SPHINX
PYTHON DOCUMENTATION GENERATOR

Sphinx Documentation

Release 4.0.0+

Georg Brandl

Feb 17, 2021

CONTENTS

CHAPTER

ONE

USING SPHINX

This guide serves to demonstrate how one can get started with Sphinx and covers everything from installing Sphinx and configuring your first Sphinx project to using some of the advanced features Sphinx provides out-of-the-box. If you are looking for guidance on extending Sphinx, refer to *Extending Sphinx*.

1.1 Getting Started

Sphinx is a *documentation generator* or a tool that translates a set of plain text source files into various output formats, automatically producing cross-references, indices, etc. That is, if you have a directory containing a bunch of *reStructuredText* or *Markdown* documents, Sphinx can generate a series of HTML files, a PDF file (via LaTeX), man pages and much more.

Sphinx focuses on documentation, in particular handwritten documentation, however, Sphinx can also be used to generate blogs, homepages and even books. Much of Sphinx's power comes from the richness of its default plain-text markup format, *reStructuredText*, along with it's *significant extensibility capabilities*.

The goal of this document is to give you a quick taste of what Sphinx is and how you might use it. When you're done here, you can check out the *installation guide* followed by the intro to the default markup format used by Sphinx, *reStucturedText*.

For a great "introduction" to writing docs in general – the whys and hows, see also Write the docs[3], written by Eric Holscher.

Setting up the documentation sources

The root directory of a Sphinx collection of plain-text document sources is called the *source directory*. This directory also contains the Sphinx configuration file `conf.py`, where you can configure all aspects of how Sphinx reads your sources and builds your documentation.[7]

Sphinx comes with a script called **`sphinx-quickstart`** that sets up a source directory and creates a default `conf.py` with the most useful configuration values from a few questions it asks you. To use this, run:

```
$ sphinx-quickstart
```

[3] http://www.writethedocs.org/guide/writing/beginners-guide-to-docs/

[7] This is the usual layout. However, `conf.py` can also live in another directory, the *configuration directory*. Refer to the *sphinx-build man page* for more information.

Defining document structure

Let's assume you've run **sphinx-quickstart**. It created a source directory with `conf.py` and a master document, `index.rst`. The main function of the *master document* is to serve as a welcome page, and to contain the root of the "table of contents tree" (or *toctree*). This is one of the main things that Sphinx adds to reStructuredText, a way to connect multiple files to a single hierarchy of documents.

reStructuredText directives

`toctree` is a reStructuredText *directive*, a very versatile piece of markup. Directives can have arguments, options and content.

Arguments are given directly after the double colon following the directive's name. Each directive decides whether it can have arguments, and how many.

Options are given after the arguments, in form of a "field list". The `maxdepth` is such an option for the `toctree` directive.

Content follows the options or arguments after a blank line. Each directive decides whether to allow content, and what to do with it.

A common gotcha with directives is that **the first line of the content must be indented to the same level as the options are**.

The `toctree` directive initially is empty, and looks like so:

```
.. toctree::
   :maxdepth: 2
```

You add documents listing them in the *content* of the directive:

```
.. toctree::
   :maxdepth: 2

   usage/installation
   usage/quickstart
   ...
```

This is exactly how the `toctree` for this documentation looks. The documents to include are given as *document names*, which in short means that you leave off the file name extension and use forward slashes (`/`) as directory separators.

Read more about *the toctree directive*.

You can now create the files you listed in the `toctree` and add content, and their section titles will be inserted (up to the `maxdepth` level) at the place where the `toctree` directive is placed. Also, Sphinx now knows about the order and hierarchy of your documents. (They may contain `toctree` directives themselves, which means you can create deeply nested hierarchies if necessary.)

Adding content

In Sphinx source files, you can use most features of standard *reStructuredText*. There are also several features added by Sphinx. For example, you can add cross-file references in a portable way (which works for all output types) using the `ref` role.

For an example, if you are viewing the HTML version, you can look at the source for this document – use the "Show Source" link in the sidebar.

Todo: Update the below link when we add new guides on these.

See *reStructuredText* for a more in-depth introduction to reStructuredText, including markup added by Sphinx.

Running the build

Now that you have added some files and content, let's make a first build of the docs. A build is started with the **`sphinx-build`** program:

```
$ sphinx-build -b html sourcedir builddir
```

where *sourcedir* is the *source directory*, and *builddir* is the directory in which you want to place the built documentation. The `-b` option selects a builder; in this example Sphinx will build HTML files.

Refer to the *sphinx-build man page* for all options that **`sphinx-build`** supports.

However, **`sphinx-quickstart`** script creates a `Makefile` and a `make.bat` which make life even easier for you. These can be executed by running **`make`** with the name of the builder. For example.

```
$ make html
```

This will build HTML docs in the build directory you chose. Execute **`make`** without an argument to see which targets are available.

How do I generate PDF documents?

`make latexpdf` runs the `LaTeX builder` and readily invokes the pdfTeX toolchain for you.

Todo: Move this whole section into a guide on rST or directives

Documenting objects

One of Sphinx's main objectives is easy documentation of *objects* (in a very general sense) in any *domain*. A domain is a collection of object types that belong together, complete with markup to create and reference descriptions of these objects.

The most prominent domain is the Python domain. For example, to document Python's built-in function `enumerate()`, you would add this to one of your source files.

```
.. py:function:: enumerate(sequence[, start=0])

   Return an iterator that yields tuples of an index and an item of the
   *sequence*. (And so on.)
```

This is rendered like this:

enumerate(*sequence*[, *start=0*])
    Return an iterator that yields tuples of an index and an item of the *sequence*. (And so on.)

The argument of the directive is the *signature* of the object you describe, the content is the documentation for it. Multiple signatures can be given, each in its own line.

The Python domain also happens to be the default domain, so you don't need to prefix the markup with the domain name.

```
.. function:: enumerate(sequence[, start=0])

   ...
```

does the same job if you keep the default setting for the default domain.

There are several more directives for documenting other types of Python objects, for example `py:class` or `py:method`. There is also a cross-referencing *role* for each of these object types. This markup will create a link to the documentation of `enumerate()`.

```
The :py:func:`enumerate` function can be used for ...
```

And here is the proof: A link to `enumerate()`.

Again, the `py:` can be left out if the Python domain is the default one. It doesn't matter which file contains the actual documentation for `enumerate()`; Sphinx will find it and create a link to it.

Each domain will have special rules for how the signatures can look like, and make the formatted output look pretty, or add specific features like links to parameter types, e.g. in the C/C++ domains.

See *Domains* for all the available domains and their directives/roles.

Basic configuration

Earlier we mentioned that the `conf.py` file controls how Sphinx processes your documents. In that file, which is executed as a Python source file, you assign configuration values. For advanced users: since it is executed by Sphinx, you can do non-trivial tasks in it, like extending `sys.path`[4] or importing a module to find out the version you are documenting.

The config values that you probably want to change are already put into the `conf.py` by **sphinx-quickstart** and initially commented out (with standard Python syntax: a `#` comments the rest of the line). To change the default value, remove the hash sign and modify the value. To customize a config value that is not automatically added by **sphinx-quickstart**, just add an additional assignment.

Keep in mind that the file uses Python syntax for strings, numbers, lists and so on. The file is saved in UTF-8 by default, as indicated by the encoding declaration in the first line.

See *Configuration* for documentation of all available config values.

[4] https://docs.python.org/3/library/sys.html#sys.path

Todo: Move this entire doc to a different section

Autodoc

When documenting Python code, it is common to put a lot of documentation in the source files, in documentation strings. Sphinx supports the inclusion of docstrings from your modules with an *extension* (an extension is a Python module that provides additional features for Sphinx projects) called *autodoc*.

In order to use *autodoc*, you need to activate it in `conf.py` by putting the string `'sphinx.ext.autodoc'` into the list assigned to the *`extensions`* config value:

```
extensions = ['sphinx.ext.autodoc']
```

Then, you have a few additional directives at your disposal. For example, to document the function `io.open()`, reading its signature and docstring from the source file, you'd write this:

```
.. autofunction:: io.open
```

You can also document whole classes or even modules automatically, using member options for the auto directives, like

```
.. automodule:: io
   :members:
```

autodoc needs to import your modules in order to extract the docstrings. Therefore, you must add the appropriate path to `sys.path`[5] in your `conf.py`.

Warning: *`autodoc`* **imports** the modules to be documented. If any modules have side effects on import, these will be executed by `autodoc` when `sphinx-build` is run.

If you document scripts (as opposed to library modules), make sure their main routine is protected by a `if __name__ == '__main__'` condition.

See *`sphinx.ext.autodoc`* for the complete description of the features of autodoc.

Todo: Move this doc to another section

Intersphinx

Many Sphinx documents including the Python documentation[6] are published on the Internet. When you want to make links to such documents from your documentation, you can do it with *`sphinx.ext.intersphinx`*.

In order to use intersphinx, you need to activate it in `conf.py` by putting the string `'sphinx.ext.intersphinx'` into the *`extensions`* list and set up the *`intersphinx_mapping`* config value.

For example, to link to `io.open()` in the Python library manual, you need to setup your *`intersphinx_mapping`* like:

```
intersphinx_mapping = {'python': ('https://docs.python.org/3', None)}
```

[5] https://docs.python.org/3/library/sys.html#sys.path
[6] https://docs.python.org/3

And now, you can write a cross-reference like `:py:func:`io.open``. Any cross-reference that has no matching target in the current documentation set, will be looked up in the documentation sets configured in *intersphinx_mapping* (this needs access to the URL in order to download the list of valid targets). Intersphinx also works for some other *domain*'s roles including `:ref:`, however it doesn't work for `:doc:` as that is non-domain role.

See *sphinx.ext.intersphinx* for the complete description of the features of intersphinx.

More topics to be covered

- *Other extensions*:
- Static files
- *Selecting a theme*
- *Setuptools integration*
- *Templating*
- Using extensions
- *Writing extensions*

1.2 Installing Sphinx

Overview

Sphinx is written in Python[8] and supports Python 3.6+. It builds upon the shoulders of many third-party libraries such as Docutils[9] and Jinja[10], which are installed when Sphinx is installed.

[8] https://docs.python-guide.org/
[9] https://docutils.sourceforge.io/
[10] https://jinja.palletsprojects.com/

Linux

Debian/Ubuntu

Install either `python3-sphinx` using **`apt-get`**:

```
$ apt-get install python3-sphinx
```

If it not already present, this will install Python for you.

RHEL, CentOS

Install `python-sphinx` using **`yum`**:

```
$ yum install python-sphinx
```

If it not already present, this will install Python for you.

Other distributions

Most Linux distributions have Sphinx in their package repositories. Usually the package is called `python3-sphinx`, `python-sphinx` or `sphinx`. Be aware that there are at least two other packages with `sphinx` in their name: a speech recognition toolkit (*CMU Sphinx*) and a full-text search database (*Sphinx search*).

macOS

Sphinx can be installed using Homebrew[11], MacPorts[12], or as part of a Python distribution such as Anaconda[13].

Homebrew

```
$ brew install sphinx-doc
```

For more information, refer to the package overview[14].

MacPorts

Install either `python3x-sphinx` using **`port`**:

```
$ sudo port install py38-sphinx
```

To set up the executable paths, use the `port select` command:

```
$ sudo port select --set python python38
$ sudo port select --set sphinx py38-sphinx
```

For more information, refer to the package overview[15].

[11] https://brew.sh/
[12] https://www.macports.org/
[13] https://www.anaconda.com/download/#macos
[14] https://formulae.brew.sh/formula/sphinx-doc
[15] https://www.macports.org/ports.php?by=library&substr=py38-sphinx

Anaconda

```
$ conda install sphinx
```

Windows

Todo: Could we start packaging this?

Most Windows users do not have Python installed by default, so we begin with the installation of Python itself. To check if you already have Python installed, open the *Command Prompt* (⊞`Win-r` and type **`cmd`**). Once the command prompt is open, type **`python --version`** and press Enter. If Python is installed, you will see the version of Python printed to the screen. If you do not have Python installed, refer to the Hitchhikers Guide to Python's[16] Python on Windows installation guides. You must install Python 3[17].

Once Python is installed, you can install Sphinx using **`pip`**. Refer to the *pip installation instructions* below for more information.

Installation from PyPI

Sphinx packages are published on the Python Package Index[18]. The preferred tool for installing packages from *PyPI* is **`pip`**. This tool is provided with all modern versions of Python.

On Linux or MacOS, you should open your terminal and run the following command.

```
$ pip install -U sphinx
```

On Windows, you should open *Command Prompt* (⊞`Win-r` and type **`cmd`**) and run the same command.

```
C:\> pip install -U sphinx
```

After installation, type **`sphinx-build --version`** on the command prompt. If everything worked fine, you will see the version number for the Sphinx package you just installed.

Installation from *PyPI* also allows you to install the latest development release. You will not generally need (or want) to do this, but it can be useful if you see a possible bug in the latest stable release. To do this, use the `--pre` flag.

```
$ pip install -U --pre sphinx
```

Docker

Docker images for Sphinx are published on the Docker Hub[19]. There are two kind of images:

- sphinxdoc/sphinx[20]
- sphinxdoc/sphinx-latexpdf[21]

[16] https://docs.python-guide.org/
[17] https://docs.python-guide.org/starting/install3/win/
[18] https://pypi.org/project/Sphinx/
[19] https://hub.docker.com/
[20] https://hub.docker.com/repository/docker/sphinxdoc/sphinx
[21] https://hub.docker.com/repository/docker/sphinxdoc/sphinx-latexpdf

Former one is used for standard usage of Sphinx, and latter one is mainly used for PDF builds using LaTeX. Please choose one for your purpose.

Note: sphinxdoc/sphinx-latexpdf contains TeXLive packages. So the image is very large (over 2GB!).

Hint: When using docker images, please use `docker run` command to invoke sphinx commands. For example, you can use following command to create a Sphinx project:

```
$ docker run -it --rm -v /path/to/document:/docs sphinxdoc/sphinx sphinx-quickstart
```

And you can following command this to build HTML document:

```
$ docker run --rm -v /path/to/document:/docs sphinxdoc/sphinx make html
```

For more details, please read README file[22] of docker images.

Installation from source

You can install Sphinx directly from a clone of the Git repository[23]. This can be done either by cloning the repo and installing from the local clone, on simply installing directly via **`git`**.

```
$ git clone https://github.com/sphinx-doc/sphinx
$ cd sphinx
$ pip install .
```

```
$ pip install git+https://github.com/sphinx-doc/sphinx
```

You can also download a snapshot of the Git repo in either tar.gz[24] or zip[25] format. Once downloaded and extracted, these can be installed with **`pip`** as above.

1.3 reStructuredText

reStructuredText (reST) is the default plaintext markup language used by both Docutils and Sphinx. Docutils provides the basic reStructuredText syntax, while Sphinx extends this to support additional functionality.

The below guides go through the most important aspects of reST. For the authoritative reStructuredText reference, refer to the docutils documentation[26].

[22] https://hub.docker.com/repository/docker/sphinxdoc/sphinx
[23] https://github.com/sphinx-doc/sphinx
[24] https://github.com/sphinx-doc/sphinx/archive/master.tar.gz
[25] https://github.com/sphinx-doc/sphinx/archive/master.zip
[26] http://docutils.sourceforge.net/rst.html

reStructuredText Primer

reStructuredText is the default plaintext markup language used by Sphinx. This section is a brief introduction to reStructuredText (reST) concepts and syntax, intended to provide authors with enough information to author documents productively. Since reST was designed to be a simple, unobtrusive markup language, this will not take too long.

See also:

The authoritative reStructuredText User Documentation[27]. The "ref" links in this document link to the description of the individual constructs in the reST reference.

Paragraphs

The paragraph (ref[28]) is the most basic block in a reST document. Paragraphs are simply chunks of text separated by one or more blank lines. As in Python, indentation is significant in reST, so all lines of the same paragraph must be left-aligned to the same level of indentation.

Inline markup

The standard reST inline markup is quite simple: use

- one asterisk: `*text*` for emphasis (italics),
- two asterisks: `**text**` for strong emphasis (boldface), and
- backquotes: ``` ``text`` ``` for code samples.

If asterisks or backquotes appear in running text and could be confused with inline markup delimiters, they have to be escaped with a backslash.

Be aware of some restrictions of this markup:

- it may not be nested,
- content may not start or end with whitespace: `* text*` is wrong,
- it must be separated from surrounding text by non-word characters. Use a backslash escaped space to work around that: `thisis\ *one*\ word`.

These restrictions may be lifted in future versions of the docutils.

It is also possible to replace or expand upon some of this inline markup with roles. Refer to *Roles* for more information.

Lists and Quote-like blocks

List markup (ref[29]) is natural: just place an asterisk at the start of a paragraph and indent properly. The same goes for numbered lists; they can also be autonumbered using a # sign:

```
* This is a bulleted list.
* It has two items, the second
  item uses two lines.

1. This is a numbered list.
2. It has two items too.
```

(continues on next page)

[27] http://docutils.sourceforge.net/rst.html
[28] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#paragraphs
[29] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#bullet-lists

(continued from previous page)

```
#. This is a numbered list.
#. It has two items too.
```

Nested lists are possible, but be aware that they must be separated from the parent list items by blank lines:

```
* this is
* a list

  * with a nested list
  * and some subitems

* and here the parent list continues
```

Definition lists (ref[30]) are created as follows:

```
term (up to a line of text)
   Definition of the term, which must be indented

   and can even consist of multiple paragraphs

next term
   Description.
```

Note that the term cannot have more than one line of text.

Quoted paragraphs (ref[31]) are created by just indenting them more than the surrounding paragraphs.

Line blocks (ref[32]) are a way of preserving line breaks:

```
| These lines are
| broken exactly like in
| the source file.
```

There are also several more special blocks available:

- field lists (ref[33], with caveats noted in *Field Lists*)
- option lists (ref[34])
- quoted literal blocks (ref[35])
- doctest blocks (ref[36])

[30] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#definition-lists
[31] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#block-quotes
[32] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#line-blocks
[33] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#field-lists
[34] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#option-lists
[35] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#quoted-literal-blocks
[36] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#doctest-blocks

Literal blocks

Literal code blocks (ref[37]) are introduced by ending a paragraph with the special marker `::`. The literal block must be indented (and, like all paragraphs, separated from the surrounding ones by blank lines):

```
This is a normal text paragraph. The next paragraph is a code sample::

   It is not processed in any way, except
   that the indentation is removed.

   It can span multiple lines.

This is a normal text paragraph again.
```

The handling of the `::` marker is smart:

- If it occurs as a paragraph of its own, that paragraph is completely left out of the document.
- If it is preceded by whitespace, the marker is removed.
- If it is preceded by non-whitespace, the marker is replaced by a single colon.

That way, the second sentence in the above example's first paragraph would be rendered as "The next paragraph is a code sample:".

Code highlighting can be enabled for these literal blocks on a document-wide basis using the *`highlight`* directive and on a project-wide basis using the *`highlight_language`* configuration option. The *`code-block`* directive can be used to set highlighting on a block-by-block basis. These directives are discussed later.

Doctest blocks

Doctest blocks (ref[38]) are interactive Python sessions cut-and-pasted into docstrings. They do not require the *literal blocks* syntax. The doctest block must end with a blank line and should *not* end with an unused prompt:

```
>>> 1 + 1
2
```

Tables

For *grid tables* (ref[39]), you have to "paint" the cell grid yourself. They look like this:

```
+------------------------+------------+----------+----------+
| Header row, column 1   | Header 2   | Header 3 | Header 4 |
| (header rows optional) |            |          |          |
+========================+============+==========+==========+
| body row 1, column 1   | column 2   | column 3 | column 4 |
+------------------------+------------+----------+----------+
| body row 2             | ...        | ...      |          |
+------------------------+------------+----------+----------+
```

Simple tables (ref[40]) are easier to write, but limited: they must contain more than one row, and the first column cells cannot contain multiple lines. They look like this:

[37] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#literal-blocks
[38] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#doctest-blocks
[39] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#grid-tables
[40] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#simple-tables

```
=====  =====  =======
A      B      A and B
=====  =====  =======
False  False  False
True   False  False
False  True   False
True   True   True
=====  =====  =======
```

Two more syntaxes are supported: *CSV tables* and *List tables*. They use an *explicit markup block*. Refer to *Tables* for more information.

Hyperlinks

External links

Use `` `Link text <https://domain.invalid/>`_ `` for inline web links. If the link text should be the web address, you don't need special markup at all, the parser finds links and mail addresses in ordinary text.

Important: There must be a space between the link text and the opening < for the URL.

You can also separate the link and the target definition (ref[41]), like this:

```
This is a paragraph that contains `a link`_.

.. _a link: https://domain.invalid/
```

Internal links

Internal linking is done via a special reST role provided by Sphinx, see the section on specific markup, *Cross-referencing arbitrary locations*.

Sections

Section headers (ref[42]) are created by underlining (and optionally overlining) the section title with a punctuation character, at least as long as the text:

```
=================
This is a heading
=================
```

Normally, there are no heading levels assigned to certain characters as the structure is determined from the succession of headings. However, this convention is used in Python's Style Guide for documenting[43] which you may follow:

- # with overline, for parts
- * with overline, for chapters
- =, for sections

[41] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#hyperlink-targets
[42] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#sections
[43] https://docs.python.org/devguide/documenting.html#style-guide

- -, for subsections
- ^, for subsubsections
- ", for paragraphs

Of course, you are free to use your own marker characters (see the reST documentation), and use a deeper nesting level, but keep in mind that most target formats (HTML, LaTeX) have a limited supported nesting depth.

Field Lists

Field lists (ref[44]) are sequences of fields marked up like this:

```
:fieldname: Field content
```

They are commonly used in Python documentation:

```
def my_function(my_arg, my_other_arg):
    """A function just for me.

    :param my_arg: The first of my arguments.
    :param my_other_arg: The second of my arguments.

    :returns: A message (just for me, of course).
    """
```

Sphinx extends standard docutils behavior and intercepts field lists specified at the beginning of documents. Refer to *Field Lists* for more information.

Roles

A role or "custom interpreted text role" (ref[45]) is an inline piece of explicit markup. It signifies that the enclosed text should be interpreted in a specific way. Sphinx uses this to provide semantic markup and cross-referencing of identifiers, as described in the appropriate section. The general syntax is `:rolename:`content``.

Docutils supports the following roles:

- emphasis[46] – equivalent of `*emphasis*`
- strong[47] – equivalent of `**strong**`
- literal[48] – equivalent of ` ``literal`` `
- subscript[49] – subscript text
- superscript[50] – superscript text
- title-reference[51] – for titles of books, periodicals, and other materials

Refer to *Roles* for roles added by Sphinx.

[44] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#field-lists
[45] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#roles
[46] http://docutils.sourceforge.net/docs/ref/rst/roles.html#emphasis
[47] http://docutils.sourceforge.net/docs/ref/rst/roles.html#strong
[48] http://docutils.sourceforge.net/docs/ref/rst/roles.html#literal
[49] http://docutils.sourceforge.net/docs/ref/rst/roles.html#subscript
[50] http://docutils.sourceforge.net/docs/ref/rst/roles.html#superscript
[51] http://docutils.sourceforge.net/docs/ref/rst/roles.html#title-reference

Explicit Markup

"Explicit markup" (ref[52]) is used in reST for most constructs that need special handling, such as footnotes, specially-highlighted paragraphs, comments, and generic directives.

An explicit markup block begins with a line starting with `..` followed by whitespace and is terminated by the next paragraph at the same level of indentation. (There needs to be a blank line between explicit markup and normal paragraphs. This may all sound a bit complicated, but it is intuitive enough when you write it.)

Directives

A directive (ref[53]) is a generic block of explicit markup. Along with roles, it is one of the extension mechanisms of reST, and Sphinx makes heavy use of it.

Docutils supports the following directives:

- Admonitions: attention[54], caution[55], danger[56], error[57], hint[58], important[59], note[60], tip[61], warning[62] and the generic admonition[63]. (Most themes style only "note" and "warning" specially.)
- Images:
  - image[64] (see also *Images* below)
  - figure[65] (an image with caption and optional legend)
- Additional body elements:
  - contents[66] (a local, i.e. for the current file only, table of contents)
  - container[67] (a container with a custom class, useful to generate an outer `<div>` in HTML)
  - rubric[68] (a heading without relation to the document sectioning)
  - topic[69], sidebar[70] (special highlighted body elements)
  - parsed-literal[71] (literal block that supports inline markup)
  - epigraph[72] (a block quote with optional attribution line)
  - highlights[73], pull-quote[74] (block quotes with their own class attribute)
  - compound[75] (a compound paragraph)

52 http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#explicit-markup-blocks
53 http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#directives
54 http://docutils.sourceforge.net/docs/ref/rst/directives.html#attention
55 http://docutils.sourceforge.net/docs/ref/rst/directives.html#caution
56 http://docutils.sourceforge.net/docs/ref/rst/directives.html#danger
57 http://docutils.sourceforge.net/docs/ref/rst/directives.html#error
58 http://docutils.sourceforge.net/docs/ref/rst/directives.html#hint
59 http://docutils.sourceforge.net/docs/ref/rst/directives.html#important
60 http://docutils.sourceforge.net/docs/ref/rst/directives.html#note
61 http://docutils.sourceforge.net/docs/ref/rst/directives.html#tip
62 http://docutils.sourceforge.net/docs/ref/rst/directives.html#warning
63 http://docutils.sourceforge.net/docs/ref/rst/directives.html#admonitions
64 http://docutils.sourceforge.net/docs/ref/rst/directives.html#image
65 http://docutils.sourceforge.net/docs/ref/rst/directives.html#figure
66 http://docutils.sourceforge.net/docs/ref/rst/directives.html#table-of-contents
67 http://docutils.sourceforge.net/docs/ref/rst/directives.html#container
68 http://docutils.sourceforge.net/docs/ref/rst/directives.html#rubric
69 http://docutils.sourceforge.net/docs/ref/rst/directives.html#topic
70 http://docutils.sourceforge.net/docs/ref/rst/directives.html#sidebar
71 http://docutils.sourceforge.net/docs/ref/rst/directives.html#parsed-literal
72 http://docutils.sourceforge.net/docs/ref/rst/directives.html#epigraph
73 http://docutils.sourceforge.net/docs/ref/rst/directives.html#highlights
74 http://docutils.sourceforge.net/docs/ref/rst/directives.html#pull-quote
75 http://docutils.sourceforge.net/docs/ref/rst/directives.html#compound-paragraph

- Special tables:
  - table[76] (a table with title)
  - csv-table[77] (a table generated from comma-separated values)
  - list-table[78] (a table generated from a list of lists)
- Special directives:
  - raw[79] (include raw target-format markup)
  - include[80] (include reStructuredText from another file) – in Sphinx, when given an absolute include file path, this directive takes it as relative to the source directory
  - class[81] (assign a class attribute to the next element)[1]
- HTML specifics:
  - meta[82] (generation of HTML `<meta>` tags, see also *HTML Metadata* below)
  - title[83] (override document title)
- Influencing markup:
  - default-role[84] (set a new default role)
  - role[85] (create a new role)

  Since these are only per-file, better use Sphinx's facilities for setting the *`default_role`*.

Warning: Do *not* use the directives sectnum[86], header[87] and footer[88].

Directives added by Sphinx are described in *Directives*.

Basically, a directive consists of a name, arguments, options and content. (Keep this terminology in mind, it is used in the next chapter describing custom directives.) Looking at this example,

```
.. function:: foo(x)
              foo(y, z)
   :module: some.module.name

   Return a line of text input from the user.
```

`function` is the directive name. It is given two arguments here, the remainder of the first line and the second line, as well as one option `module` (as you can see, options are given in the lines immediately following the arguments and indicated by the colons). Options must be indented to the same level as the directive content.

The directive content follows after a blank line and is indented relative to the directive start.

[76] http://docutils.sourceforge.net/docs/ref/rst/directives.html#table
[77] http://docutils.sourceforge.net/docs/ref/rst/directives.html#csv-table
[78] http://docutils.sourceforge.net/docs/ref/rst/directives.html#list-table
[79] http://docutils.sourceforge.net/docs/ref/rst/directives.html#raw-data-pass-through
[80] http://docutils.sourceforge.net/docs/ref/rst/directives.html#include
[81] http://docutils.sourceforge.net/docs/ref/rst/directives.html#class
[1] When the default domain contains a `class` directive, this directive will be shadowed. Therefore, Sphinx re-exports it as `rst-class`.
[82] http://docutils.sourceforge.net/docs/ref/rst/directives.html#meta
[83] http://docutils.sourceforge.net/docs/ref/rst/directives.html#metadata-document-title
[84] http://docutils.sourceforge.net/docs/ref/rst/directives.html#default-role
[85] http://docutils.sourceforge.net/docs/ref/rst/directives.html#role
[86] http://docutils.sourceforge.net/docs/ref/rst/directives.html#sectnum
[87] http://docutils.sourceforge.net/docs/ref/rst/directives.html#header
[88] http://docutils.sourceforge.net/docs/ref/rst/directives.html#footer

Images

reST supports an image directive (ref[89]), used like so:

```
.. image:: gnu.png
   (options)
```

When used within Sphinx, the file name given (here `gnu.png`) must either be relative to the source file, or absolute which means that they are relative to the top source directory. For example, the file `sketch/spam.rst` could refer to the image `images/spam.png` as `../images/spam.png` or `/images/spam.png`.

Sphinx will automatically copy image files over to a subdirectory of the output directory on building (e.g. the `_static` directory for HTML output.)

Interpretation of image size options (`width` and `height`) is as follows: if the size has no unit or the unit is pixels, the given size will only be respected for output channels that support pixels. Other units (like `pt` for points) will be used for HTML and LaTeX output (the latter replaces `pt` by `bp` as this is the TeX unit such that `72bp=1in`).

Sphinx extends the standard docutils behavior by allowing an asterisk for the extension:

```
.. image:: gnu.*
```

Sphinx then searches for all images matching the provided pattern and determines their type. Each builder then chooses the best image out of these candidates. For instance, if the file name `gnu.*` was given and two files `gnu.pdf` and `gnu.png` existed in the source tree, the LaTeX builder would choose the former, while the HTML builder would prefer the latter. Supported image types and choosing priority are defined at *Builders*.

Note that image file names should not contain spaces.

Changed in version 0.4: Added the support for file names ending in an asterisk.

Changed in version 0.6: Image paths can now be absolute.

Changed in version 1.5: latex target supports pixels (default is `96px=1in`).

Footnotes

For footnotes (ref[90]), use `[#name]_` to mark the footnote location, and add the footnote body at the bottom of the document after a "Footnotes" rubric heading, like so:

```
Lorem ipsum [#f1]_ dolor sit amet ... [#f2]_

.. rubric:: Footnotes

.. [#f1] Text of the first footnote.
.. [#f2] Text of the second footnote.
```

You can also explicitly number the footnotes (`[1]_`) or use auto-numbered footnotes without names (`[#]_`).

[89] http://docutils.sourceforge.net/docs/ref/rst/directives.html#image
[90] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#footnotes

Citations

Standard reST citations (ref[91]) are supported, with the additional feature that they are "global", i.e. all citations can be referenced from all files. Use them like so:

```
Lorem ipsum [Ref]_ dolor sit amet.

.. [Ref] Book or article reference, URL or whatever.
```

Citation usage is similar to footnote usage, but with a label that is not numeric or begins with #.

Substitutions

reST supports "substitutions" (ref[92]), which are pieces of text and/or markup referred to in the text by `|name|`. They are defined like footnotes with explicit markup blocks, like this:

```
.. |name| replace:: replacement *text*
```

or this:

```
.. |caution| image:: warning.png
             :alt: Warning!
```

See the reST reference for substitutions[93] for details.

If you want to use some substitutions for all documents, put them into *rst_prolog* or *rst_epilog* or put them into a separate file and include it into all documents you want to use them in, using the `include` directive. (Be sure to give the include file a file name extension differing from that of other source files, to avoid Sphinx finding it as a standalone document.)

Sphinx defines some default substitutions, see *Substitutions*.

Comments

Every explicit markup block which isn't a valid markup construct (like the footnotes above) is regarded as a comment (ref[94]). For example:

```
.. This is a comment.
```

You can indent text after a comment start to form multiline comments:

```
..
   This whole indented block
   is a comment.

   Still in the comment.
```

[91] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#citations
[92] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#substitution-definitions
[93] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#substitution-definitions
[94] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#comments

HTML Metadata

The meta directive (ref[95]) allows specifying the HTML metadata element[96] of a Sphinx documentation page. For example, the directive:

```
.. meta::
   :description: The Sphinx documentation builder
   :keywords: Sphinx, documentation, builder
```

will generate the following HTML output:

```
<meta name="description" content="The Sphinx documentation builder">
<meta name="keywords" content="Sphinx, documentation, builder">
```

Also, Sphinx will add the keywords as specified in the meta directive to the search index. Thereby, the lang attribute of the meta element is considered. For example, the directive:

```
.. meta::
   :keywords: backup
   :keywords lang=en: pleasefindthiskey pleasefindthiskeytoo
   :keywords lang=de: bittediesenkeyfinden
```

adds the following words to the search indices of builds with different language configurations:

- pleasefindthiskey, pleasefindthiskeytoo to *English* builds;
- bittediesenkeyfinden to *German* builds;
- backup to builds in all languages.

Source encoding

Since the easiest way to include special characters like em dashes or copyright signs in reST is to directly write them as Unicode characters, one has to specify an encoding. Sphinx assumes source files to be encoded in UTF-8 by default; you can change this with the *source_encoding* config value.

Gotchas

There are some problems one commonly runs into while authoring reST documents:

- **Separation of inline markup:** As said above, inline markup spans must be separated from the surrounding text by non-word characters, you have to use a backslash-escaped space to get around that. See the reference[97] for the details.
- **No nested inline markup:** Something like `*see :func:`foo`*` is not possible.

[95] http://docutils.sourceforge.net/docs/ref/rst/directives.html#meta
[96] https://developer.mozilla.org/en-US/docs/Web/HTML/Element/meta
[97] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#substitution-definitions

Roles

Sphinx uses interpreted text roles to insert semantic markup into documents. They are written as `:rolename:`content``.

Note: The default role (``content``) has no special meaning by default. You are free to use it for anything you like, e.g. variable names; use the *`default_role`* config value to set it to a known role – the *`any`* role to find anything or the *`py:obj`* role to find Python objects are very useful for this.

See *Domains* for roles added by domains.

Cross-referencing syntax

Cross-references are generated by many semantic interpreted text roles. Basically, you only need to write `:role:`target``, and a link will be created to the item named *target* of the type indicated by *role*. The link's text will be the same as *target*.

There are some additional facilities, however, that make cross-referencing roles more versatile:

- You may supply an explicit title and reference target, like in reST direct hyperlinks: `:role:`title <target>`` will refer to *target*, but the link text will be *title*.
- If you prefix the content with `!`, no reference/hyperlink will be created.
- If you prefix the content with `~`, the link text will only be the last component of the target. For example, `:py:meth:`~Queue.Queue.get`` will refer to `Queue.Queue.get` but only display `get` as the link text. This does not work with all cross-reference roles, but is domain specific.

  In HTML output, the link's `title` attribute (that is e.g. shown as a tool-tip on mouse-hover) will always be the full target name.

Cross-referencing anything

`:any:`

New in version 1.3.

This convenience role tries to do its best to find a valid target for its reference text.

- First, it tries standard cross-reference targets that would be referenced by *`doc`*, *`ref`* or *`option`*.

  Custom objects added to the standard domain by extensions (see *`Sphinx.add_object_type()`*) are also searched.
- Then, it looks for objects (targets) in all loaded domains. It is up to the domains how specific a match must be. For example, in the Python domain a reference of `:any:`Builder`` would match the `sphinx.builders.Builder` class.

If none or multiple targets are found, a warning will be emitted. In the case of multiple targets, you can change "any" to a specific role.

This role is a good candidate for setting *`default_role`*. If you do, you can write cross-references without a lot of markup overhead. For example, in this Python function documentation

```
.. function:: install()

   This function installs a `handler` for every signal known by the
   `signal` module.  See the section `about-signals` for more information.
```

there could be references to a glossary term (usually `:term:`handler``), a Python module (usually `:py:mod:`signal`` or `:mod:`signal``) and a section (usually `:ref:`about-signals``).

The *`any`* role also works together with the *`intersphinx`* extension: when no local cross-reference is found, all object types of intersphinx inventories are also searched.

Cross-referencing objects

These roles are described with their respective domains:

- *Python*
- *C*
- *C++*
- *JavaScript*
- *ReST*

Cross-referencing arbitrary locations

`:ref:`

To support cross-referencing to arbitrary locations in any document, the standard reST labels are used. For this to work label names must be unique throughout the entire documentation. There are two ways in which you can refer to labels:

- If you place a label directly before a section title, you can reference to it with `:ref:`label-name``. For example:

```
.. _my-reference-label:

Section to cross-reference
--------------------------

This is the text of the section.

It refers to the section itself, see :ref:`my-reference-label`.
```

  The `:ref:` role would then generate a link to the section, with the link title being "Section to cross-reference". This works just as well when section and reference are in different source files.

  Automatic labels also work with figures. For example:

```
.. _my-figure:

.. figure:: whatever

   Figure caption
```

  In this case, a reference `:ref:`my-figure`` would insert a reference to the figure with link text "Figure caption".

  The same works for tables that are given an explicit caption using the table[98] directive.

- Labels that aren't placed before a section title can still be referenced, but you must give the link an explicit title, using this syntax: `:ref:`Link title <label-name>``.

Note: Reference labels must start with an underscore. When referencing a label, the underscore must be omitted (see examples above).

Using *ref* is advised over standard reStructuredText links to sections (like `` `Section title`_ ``) because it works across files, when section headings are changed, will raise warnings if incorrect, and works for all builders that support cross-references.

Cross-referencing documents

New in version 0.6.

There is also a way to directly link to documents:

`:doc:`

Link to the specified document; the document name can be specified in absolute or relative fashion. For example, if the reference ``:doc:`parrot` `` occurs in the document `sketches/index`, then the link refers to `sketches/parrot`. If the reference is ``:doc:`/people` `` or ``:doc:`../people` ``, the link refers to `people`.

If no explicit link text is given (like usual: ``:doc:`Monty Python members </people>` ``), the link caption will be the title of the given document.

Referencing downloadable files

New in version 0.6.

`:download:`

This role lets you link to files within your source tree that are not reST documents that can be viewed, but files that can be downloaded.

When you use this role, the referenced file is automatically marked for inclusion in the output when building (obviously, for HTML output only). All downloadable files are put into a `_downloads/<unique hash>/` subdirectory of the output directory; duplicate filenames are handled.

An example:

```
See :download:`this example script <../example.py>`.
```

The given filename is usually relative to the directory the current source file is contained in, but if it absolute (starting with `/`), it is taken as relative to the top source directory.

The `example.py` file will be copied to the output directory, and a suitable link generated to it.

Not to show unavailable download links, you should wrap whole paragraphs that have this role:

```
.. only:: builder_html

   See :download:`this example script <../example.py>`.
```

[98] http://docutils.sourceforge.net/docs/ref/rst/directives.html#table

Cross-referencing figures by figure number

New in version 1.3.

Changed in version 1.5: *numref* role can also refer sections. And *numref* allows *{name}* for the link text.

`:numref:`

Link to the specified figures, tables, code-blocks and sections; the standard reST labels are used. When you use this role, it will insert a reference to the figure with link text by its figure number like "Fig. 1.1".

If an explicit link text is given (as usual: ``:numref:`Image of Sphinx (Fig. %s) <my-figure>` ``), the link caption will serve as title of the reference. As placeholders, *%s* and *{number}* get replaced by the figure number and *{name}* by the figure caption. If no explicit link text is given, the *`numfig_format`* setting is used as fall-back default.

If *`numfig`* is `False`, figures are not numbered, so this role inserts not a reference but the label or the link text.

Cross-referencing other items of interest

The following roles do possibly create a cross-reference, but do not refer to objects:

`:envvar:`

An environment variable. Index entries are generated. Also generates a link to the matching *`envvar`* directive, if it exists.

`:token:`

The name of a grammar token (used to create links between *`productionlist`* directives).

`:keyword:`

The name of a keyword in Python. This creates a link to a reference label with that name, if it exists.

`:option:`

A command-line option to an executable program. This generates a link to a *`option`* directive, if it exists.

The following role creates a cross-reference to a term in a *glossary*:

`:term:`

Reference to a term in a glossary. A glossary is created using the `glossary` directive containing a definition list with terms and definitions. It does not have to be in the same file as the `term` markup, for example the Python docs have one global glossary in the `glossary.rst` file.

If you use a term that's not explained in a glossary, you'll get a warning during build.

Math

`:math:`

Role for inline math. Use like this:

```
Since Pythagoras, we know that :math:`a^2 + b^2 = c^2`.
```

`:eq:`

Same as *`math:numref`*.

Other semantic markup

The following roles don't do anything special except formatting the text in a different style:

`:abbr:`
An abbreviation. If the role content contains a parenthesized explanation, it will be treated specially: it will be shown in a tool-tip in HTML, and output only once in LaTeX.

Example: ``:abbr:`LIFO (last-in, first-out)` ``.

New in version 0.6.

`:command:`
The name of an OS-level command, such as `rm`.

`:dfn:`
Mark the defining instance of a term in the text. (No index entries are generated.)

`:file:`
The name of a file or directory. Within the contents, you can use curly braces to indicate a "variable" part, for example:

```
... is installed in :file:`/usr/lib/python2.{x}/site-packages` ...
```

In the built documentation, the `x` will be displayed differently to indicate that it is to be replaced by the Python minor version.

`:guilabel:`
Labels presented as part of an interactive user interface should be marked using `guilabel`. This includes labels from text-based interfaces such as those created using `curses`[99] or other text-based libraries. Any label used in the interface should be marked with this role, including button labels, window titles, field names, menu and menu selection names, and even values in selection lists.

Changed in version 1.0: An accelerator key for the GUI label can be included using an ampersand; this will be stripped and displayed underlined in the output (example: ``:guilabel:`&Cancel` ``). To include a literal ampersand, double it.

`:kbd:`
Mark a sequence of keystrokes. What form the key sequence takes may depend on platform- or application-specific conventions. When there are no relevant conventions, the names of modifier keys should be spelled out, to improve accessibility for new users and non-native speakers. For example, an *xemacs* key sequence may be marked like ``:kbd:`C-x C-f` ``, but without reference to a specific application or platform, the same sequence should be marked as ``:kbd:`Control-x Control-f` ``.

`:mailheader:`
The name of an RFC 822-style mail header. This markup does not imply that the header is being used in an email message, but can be used to refer to any header of the same "style." This is also used for headers defined by the various MIME specifications. The header name should be entered in the same way it would normally be found in practice, with the camel-casing conventions being preferred where there is more than one common usage. For example: ``:mailheader:`Content-Type` ``.

`:makevar:`
The name of a **`make`** variable.

`:manpage:`
A reference to a Unix manual page including the section, e.g. ``:manpage:`ls(1)` ``. Creates a hyperlink to an external site rendering the manpage if *`manpages_url`* is defined.

`:menuselection:`
Menu selections should be marked using the `menuselection` role. This is used to mark a complete sequence of

[99] https://docs.python.org/3/library/curses.html#module-curses

menu selections, including selecting submenus and choosing a specific operation, or any subsequence of such a sequence. The names of individual selections should be separated by `-->`.

For example, to mark the selection "Start > Programs", use this markup:

```
:menuselection:`Start --> Programs`
```

When including a selection that includes some trailing indicator, such as the ellipsis some operating systems use to indicate that the command opens a dialog, the indicator should be omitted from the selection name.

`menuselection` also supports ampersand accelerators just like *`guilabel`*.

`:mimetype:`
The name of a MIME type, or a component of a MIME type (the major or minor portion, taken alone).

`:newsgroup:`
The name of a Usenet newsgroup.

Todo: Is this not part of the standard domain?

`:program:`
The name of an executable program. This may differ from the file name for the executable for some platforms. In particular, the `.exe` (or other) extension should be omitted for Windows programs.

`:regexp:`
A regular expression. Quotes should not be included.

`:samp:`
A piece of literal text, such as code. Within the contents, you can use curly braces to indicate a "variable" part, as in *`file`*. For example, in ``:samp:`print 1+{variable}` ``, the part `variable` would be emphasized.

If you don't need the "variable part" indication, use the standard ` ``code`` ` instead.

Changed in version 1.8: Allowed to escape curly braces with backslash

There is also an *`index`* role to generate index entries.

The following roles generate external links:

`:pep:`
A reference to a Python Enhancement Proposal. This generates appropriate index entries. The text "PEP *number*" is generated; in the HTML output, this text is a hyperlink to an online copy of the specified PEP. You can link to a specific section by saying ``:pep:`number#anchor` ``.

`:rfc:`
A reference to an Internet Request for Comments. This generates appropriate index entries. The text "RFC *number*" is generated; in the HTML output, this text is a hyperlink to an online copy of the specified RFC. You can link to a specific section by saying ``:rfc:`number#anchor` ``.

Note that there are no special roles for including hyperlinks as you can use the standard reST markup for that purpose.

Substitutions

The documentation system provides three substitutions that are defined by default. They are set in the build configuration file.

`|release|`
: Replaced by the project release the documentation refers to. This is meant to be the full version string including alpha/beta/release candidate tags, e.g. `2.5.2b3`. Set by *`release`*.

`|version|`
: Replaced by the project version the documentation refers to. This is meant to consist only of the major and minor version parts, e.g. `2.5`, even for version 2.5.1. Set by *`version`*.

`|today|`
: Replaced by either today's date (the date on which the document is read), or the date set in the build configuration file. Normally has the format `April 14, 2007`. Set by *`today_fmt`* and *`today`*.

Directives

As previously discussed, a directive is a generic block of explicit markup. While Docutils provides a number of directives, Sphinx provides many more and uses directives as one of the primary extension mechanisms.

See *Domains* for roles added by domains.

See also:

Refer to the *reStructuredText Primer* for an overview of the directives provided by Docutils.

Table of contents

Since reST does not have facilities to interconnect several documents, or split documents into multiple output files, Sphinx uses a custom directive to add relations between the single files the documentation is made of, as well as tables of contents. The `toctree` directive is the central element.

Note: Simple "inclusion" of one file in another can be done with the include[100] directive.

Note: To create table of contents for current document (.rst file), use the standard reST contents directive[101].

`.. toctree::`

This directive inserts a "TOC tree" at the current location, using the individual TOCs (including "sub-TOC trees") of the documents given in the directive body. Relative document names (not beginning with a slash) are relative to the document the directive occurs in, absolute names are relative to the source directory. A numeric `maxdepth` option may be given to indicate the depth of the tree; by default, all levels are included.[113]

The representation of "TOC tree" is changed in each output format. The builders that output multiple files (ex. HTML) treat it as a collection of hyperlinks. On the other hand, the builders that output a single file (ex. LaTeX, man page, etc.) replace it with the content of the documents on the TOC tree.

Consider this example (taken from the Python docs' library reference index):

[100] http://docutils.sourceforge.net/docs/ref/rst/directives.html#include
[101] http://docutils.sourceforge.net/docs/ref/rst/directives.html#table-of-contents

```
.. toctree::
   :maxdepth: 2

   intro
   strings
   datatypes
   numeric
   (many more documents listed here)
```

This accomplishes two things:

- Tables of contents from all those documents are inserted, with a maximum depth of two, that means one nested heading. `toctree` directives in those documents are also taken into account.
- Sphinx knows the relative order of the documents `intro`, `strings` and so forth, and it knows that they are children of the shown document, the library index. From this information it generates "next chapter", "previous chapter" and "parent chapter" links.

Entries

Document titles in the *`toctree`* will be automatically read from the title of the referenced document. If that isn't what you want, you can specify an explicit title and target using a similar syntax to reST hyperlinks (and Sphinx's *cross-referencing syntax*). This looks like:

```
.. toctree::

   intro
   All about strings <strings>
   datatypes
```

The second line above will link to the `strings` document, but will use the title "All about strings" instead of the title of the `strings` document.

You can also add external links, by giving an HTTP URL instead of a document name.

Section numbering

If you want to have section numbers even in HTML output, give the **toplevel** toctree a `numbered` option. For example:

```
.. toctree::
   :numbered:

   foo
   bar
```

Numbering then starts at the heading of `foo`. Sub-toctrees are automatically numbered (don't give the `numbered` flag to those).

Numbering up to a specific depth is also possible, by giving the depth as a numeric argument to `numbered`.

Additional options

You can use the `caption` option to provide a toctree caption and you can use the `name` option to provide an implicit target name that can be referenced by using *`ref`*:

```
.. toctree::
   :caption: Table of Contents
   :name: mastertoc
```

(continues on next page)

(continued from previous page)

```
   foo
```

If you want only the titles of documents in the tree to show up, not other headings of the same level, you can use the `titlesonly` option:

```
.. toctree::
   :titlesonly:

   foo
   bar
```

You can use "globbing" in toctree directives, by giving the `glob` flag option. All entries are then matched against the list of available documents, and matches are inserted into the list alphabetically. Example:

```
.. toctree::
   :glob:

   intro*
   recipe/*
   *
```

This includes first all documents whose names start with `intro`, then all documents in the `recipe` folder, then all remaining documents (except the one containing the directive, of course.)[114]

The special entry name `self` stands for the document containing the toctree directive. This is useful if you want to generate a "sitemap" from the toctree.

You can use the `reversed` flag option to reverse the order of the entries in the list. This can be useful when using the `glob` flag option to reverse the ordering of the files. Example:

```
.. toctree::
   :glob:
   :reversed:

   recipe/*
```

You can also give a "hidden" option to the directive, like this:

```
.. toctree::
   :hidden:

   doc_1
   doc_2
```

This will still notify Sphinx of the document hierarchy, but not insert links into the document at the location of the directive – this makes sense if you intend to insert these links yourself, in a different style, or in the HTML sidebar.

In cases where you want to have only one top-level toctree and hide all other lower level toctrees you can add the "includehidden" option to the top-level toctree entry:

```
.. toctree::
   :includehidden:
```

(continues on next page)

(continued from previous page)

```
   doc_1
   doc_2
```

All other toctree entries can then be eliminated by the "hidden" option.

In the end, all documents in the *source directory* (or subdirectories) must occur in some `toctree` directive; Sphinx will emit a warning if it finds a file that is not included, because that means that this file will not be reachable through standard navigation.

Use `exclude_patterns` to explicitly exclude documents or directories from building completely. Use *the "orphan" metadata* to let a document be built, but notify Sphinx that it is not reachable via a toctree.

The "master document" (selected by `master_doc`) is the "root" of the TOC tree hierarchy. It can be used as the documentation's main page, or as a "full table of contents" if you don't give a `maxdepth` option.

Changed in version 0.3: Added "globbing" option.

Changed in version 0.6: Added "numbered" and "hidden" options as well as external links and support for "self" references.

Changed in version 1.0: Added "titlesonly" option.

Changed in version 1.1: Added numeric argument to "numbered".

Changed in version 1.2: Added "includehidden" option.

Changed in version 1.3: Added "caption" and "name" option.

Special names

Sphinx reserves some document names for its own use; you should not try to create documents with these names – it will cause problems.

The special document names (and pages generated for them) are:

- `genindex`, `modindex`, `search`

  These are used for the general index, the Python module index, and the search page, respectively.

  The general index is populated with entries from modules, all index-generating *object descriptions*, and from `index` directives.

  The Python module index contains one entry per `py:module` directive.

  The search page contains a form that uses the generated JSON search index and JavaScript to full-text search the generated documents for search words; it should work on every major browser that supports modern JavaScript.

- every name beginning with `_`

  Though few such names are currently used by Sphinx, you should not create documents or document-containing directories with such names. (Using `_` as a prefix for a custom template directory is fine.)

Warning: Be careful with unusual characters in filenames. Some formats may interpret these characters in unexpected ways:

- Do not use the colon `:` for HTML based formats. Links to other parts may not work.
- Do not use the plus `+` for the ePub format. Some resources may not be found.

[113] The LaTeX writer only refers the `maxdepth` option of first toctree directive in the document.

[114] A note on available globbing syntax: you can use the standard shell constructs `*`, `?`, `[...]` and `[!...]` with the feature that these all don't match slashes. A double star `**` can be used to match any sequence of characters *including* slashes.

Paragraph-level markup

These directives create short paragraphs and can be used inside information units as well as normal text.

.. note::

An especially important bit of information about an API that a user should be aware of when using whatever bit of API the note pertains to. The content of the directive should be written in complete sentences and include all appropriate punctuation.

Example:

```
.. note::

   This function is not suitable for sending spam e-mails.
```

.. warning::

An important bit of information about an API that a user should be very aware of when using whatever bit of API the warning pertains to. The content of the directive should be written in complete sentences and include all appropriate punctuation. This differs from `note` in that it is recommended over `note` for information regarding security.

.. versionadded:: `version`

This directive documents the version of the project which added the described feature to the library or C API. When this applies to an entire module, it should be placed at the top of the module section before any prose.

The first argument must be given and is the version in question; you can add a second argument consisting of a *brief* explanation of the change.

Example:

```
.. versionadded:: 2.5
   The *spam* parameter.
```

Note that there must be no blank line between the directive head and the explanation; this is to make these blocks visually continuous in the markup.

.. versionchanged:: `version`

Similar to `versionadded`, but describes when and what changed in the named feature in some way (new parameters, changed side effects, etc.).

.. deprecated:: `version`

Similar to `versionchanged`, but describes when the feature was deprecated. An explanation can also be given, for example to inform the reader what should be used instead. Example:

```
.. deprecated:: 3.1
   Use :func:`spam` instead.
```

.. seealso::

Many sections include a list of references to module documentation or external documents. These lists are created using the `seealso` directive.

The `seealso` directive is typically placed in a section just before any subsections. For the HTML output, it is shown boxed off from the main flow of the text.

The content of the `seealso` directive should be a reST definition list. Example:

```
.. seealso::

   Module :py:mod:`zipfile`
```

(continues on next page)

(continued from previous page)

```
      Documentation of the :py:mod:`zipfile` standard module.

   `GNU tar manual, Basic Tar Format <http://link>`_
      Documentation for tar archive files, including GNU tar extensions.
```

There's also a "short form" allowed that looks like this:

```
.. seealso:: modules :py:mod:`zipfile`, :py:mod:`tarfile`
```

New in version 0.5: The short form.

.. rubric:: title

This directive creates a paragraph heading that is not used to create a table of contents node.

Note: If the *title* of the rubric is "Footnotes" (or the selected language's equivalent), this rubric is ignored by the LaTeX writer, since it is assumed to only contain footnote definitions and therefore would create an empty heading.

.. centered::

This directive creates a centered boldfaced line of text. Use it as follows:

```
.. centered:: LICENSE AGREEMENT
```

Deprecated since version 1.1: This presentation-only directive is a legacy from older versions. Use a `rst-class` directive instead and add an appropriate style.

.. hlist::

This directive must contain a bullet list. It will transform it into a more compact list by either distributing more than one item horizontally, or reducing spacing between items, depending on the builder.

For builders that support the horizontal distribution, there is a `columns` option that specifies the number of columns; it defaults to 2. Example:

```
.. hlist::
   :columns: 3

   * A list of
   * short items
   * that should be
   * displayed
   * horizontally
```

New in version 0.6.

Showing code examples

There are multiple ways to show syntax-highlighted literal code blocks in Sphinx:

- using *reST doctest blocks*;
- using *reST literal blocks*, optionally in combination with the `highlight` directive;
- using the `code-block` directive;
- and using the `literalinclude` directive.

Doctest blocks can only be used to show interactive Python sessions, while the remaining three can be used for other languages. Of these three, literal blocks are useful when an entire document, or at least large sections of it, use code blocks with the same syntax and which should be styled in the same manner. On the other hand, the `code-block` directive makes more sense when you want more fine-tuned control over the styling of each block or when you have a document containing code blocks using multiple varied syntaxes. Finally, the `literalinclude` directive is useful for including entire code files in your documentation.

In all cases, Syntax highlighting is provided by Pygments[102]. When using literal blocks, this is configured using any `highlight` directives in the source file. When a `highlight` directive is encountered, it is used until the next `highlight` directive is encountered. If there is no `highlight` directive in the file, the global highlighting language is used. This defaults to `python` but can be configured using the `highlight_language` config value. The following values are supported:

- `none` (no highlighting)
- `default` (similar to `python3` but with a fallback to `none` without warning highlighting fails; the default when `highlight_language` isn't set)
- `guess` (let Pygments guess the lexer based on contents, only works with certain well-recognizable languages)
- `python`
- `rest`
- `c`
- … and any other lexer alias that Pygments supports[103]

If highlighting with the selected language fails (i.e. Pygments emits an "Error" token), the block is not highlighted in any way.

Important: The list of lexer aliases supported is tied to the Pygment version. If you want to ensure consistent highlighting, you should fix your version of Pygments.

.. highlight:: language

Example:

```
.. highlight:: c
```

This language is used until the next `highlight` directive is encountered. As discussed previously, *language* can be any lexer alias supported by Pygments.

options

:linenothreshold: threshold (number (optional))

Enable to generate line numbers for code blocks.

This option takes an optional number as threshold parameter. If any threshold given, the directive will produce line numbers only for the code blocks longer than N lines. If not given, line numbers will be produced for all of code blocks.

Example:

```
.. highlight:: python
   :linenothreshold: 5
```

[102] http://pygments.org
[103] http://pygments.org/docs/lexers

:force: (no value)
If given, minor errors on highlighting are ignored.

New in version 2.1.

.. code-block:: [language]
Example:

```
.. code-block:: ruby

   Some Ruby code.
```

The directive's alias name `sourcecode` works as well. This directive takes a language name as an argument. It can be any lexer alias supported by Pygments. If it is not given, the setting of *highlight* directive will be used. If not set, *highlight_language* will be used.

Changed in version 2.0: The `language` argument becomes optional.

options

:linenos: (no value)
Enable to generate line numbers for the code block:

```
.. code-block:: ruby
   :linenos:

   Some more Ruby code.
```

:lineno-start: number (number)
Set the first line number of the code block. If present, `linenos` option is also automatically activated:

```
.. code-block:: ruby
   :lineno-start: 10

   Some more Ruby code, with line numbering starting at 10.
```

New in version 1.3.

:emphasize-lines: line numbers (comma separated numbers)
Emphasize particular lines of the code block:

```
.. code-block:: python
   :emphasize-lines: 3,5

   def some_function():
       interesting = False
       print 'This line is highlighted.'
       print 'This one is not...'
       print '...but this one is.'
```

New in version 1.1.

Changed in version 1.6.6: LaTeX supports the `emphasize-lines` option.

:caption: caption of code block (text)
Set a caption to the code block.

New in version 1.3.

:name: a label for hyperlink (text)

Define implicit target name that can be referenced by using *ref*. For example:

```
.. code-block:: python
   :caption: this.py
   :name: this-py

   print 'Explicit is better than implicit.'
```

In order to cross-reference a code-block using either the *ref* or the *numref* role, it is necessary that both **name** and **caption** be defined. The argument of **name** can then be given to *numref* to generate the cross-reference. Example:

```
See :numref:`this-py` for an example.
```

When using *ref*, it is possible to generate a cross-reference with only **name** defined, provided an explicit title is given. Example:

```
See :ref:`this code snippet <this-py>` for an example.
```

New in version 1.3.

:dedent: number (number or no value)

Strip indentation characters from the code block. When number given, leading N characters are removed. When no argument given, leading spaces are removed via `textwrap.dedent()`[104]. For example:

```
.. code-block:: ruby
   :dedent: 4

       some ruby code
```

New in version 1.3.

Changed in version 3.5: Support automatic dedent.

:force: (no value)

If given, minor errors on highlighting are ignored.

New in version 2.1.

.. literalinclude:: filename

Longer displays of verbatim text may be included by storing the example text in an external file containing only plain text. The file may be included using the `literalinclude` directive.[115] For example, to include the Python source file `example.py`, use:

```
.. literalinclude:: example.py
```

The file name is usually relative to the current file's path. However, if it is absolute (starting with `/`), it is relative to the top source directory.

Additional options

Like *code-block*, the directive supports the `linenos` flag option to switch on line numbers, the `lineno-start` option to select the first line number, the `emphasize-lines` option to emphasize particular lines, the `name` option to provide an implicit target name, the `dedent` option to strip indentation characters for the code block, and a `language` option to select a language different from the current file's standard language. In addition, it supports the `caption` option; however, this can be provided with no argument to use the filename as the caption. Example with options:

[104] https://docs.python.org/3/library/textwrap.html#textwrap.dedent

```
.. literalinclude:: example.rb
   :language: ruby
   :emphasize-lines: 12,15-18
   :linenos:
```

Tabs in the input are expanded if you give a `tab-width` option with the desired tab width.

Include files are assumed to be encoded in the *source_encoding*. If the file has a different encoding, you can specify it with the `encoding` option:

```
.. literalinclude:: example.py
   :encoding: latin-1
```

The directive also supports including only parts of the file. If it is a Python module, you can select a class, function or method to include using the `pyobject` option:

```
.. literalinclude:: example.py
   :pyobject: Timer.start
```

This would only include the code lines belonging to the `start()` method in the `Timer` class within the file.

Alternately, you can specify exactly which lines to include by giving a `lines` option:

```
.. literalinclude:: example.py
   :lines: 1,3,5-10,20-
```

This includes the lines 1, 3, 5 to 10 and lines 20 to the last line.

Another way to control which part of the file is included is to use the `start-after` and `end-before` options (or only one of them). If `start-after` is given as a string option, only lines that follow the first line containing that string are included. If `end-before` is given as a string option, only lines that precede the first lines containing that string are included. The `start-at` and `end-at` options behave in a similar way, but the lines containing the matched string are included.

`start-after`/`start-at` and `end-before`/`end-at` can have same string. `start-after`/`start-at` filter lines before the line that contains option string (`start-at` will keep the line). Then `end-before`/`end-at` filter lines after the line that contains option string (`end-at` will keep the line and `end-before` skip the first line).

Note: If you want to select only `[second-section]` of ini file like the following, you can use `:start-at: [second-section]` and `:end-before: [third-section]`:

```
[first-section]

var_in_first=true

[second-section]

var_in_second=true

[third-section]

var_in_third=true
```

Useful cases of these option is working with tag comments. `:start-after: [initialized]` and `:end-before: [initialized]` options keep lines between comments:

```
if __name__ == "__main__":
    # [initialize]
    app.start(":8000")
    # [initialize]
```

When lines have been selected in any of the ways described above, the line numbers in `emphasize-lines` refer to those selected lines, counted consecutively starting at 1.

When specifying particular parts of a file to display, it can be useful to display the original line numbers. This can be done using the `lineno-match` option, which is however allowed only when the selection consists of contiguous lines.

You can prepend and/or append a line to the included code, using the `prepend` and `append` option, respectively. This is useful e.g. for highlighting PHP code that doesn't include the <?php/?> markers.

If you want to show the diff of the code, you can specify the old file by giving a `diff` option:

```
.. literalinclude:: example.py
   :diff: example.py.orig
```

This shows the diff between `example.py` and `example.py.orig` with unified diff format.

A `force` option can ignore minor errors on highlighting.

Changed in version 0.4.3: Added the `encoding` option.

Changed in version 0.6: Added the `pyobject`, `lines`, `start-after` and `end-before` options, as well as support for absolute filenames.

Changed in version 1.0: Added the `prepend`, `append`, and `tab-width` options.

Changed in version 1.3: Added the `diff`, `lineno-match`, `caption`, `name`, and `dedent` options.

Changed in version 1.5: Added the `start-at`, and `end-at` options.

Changed in version 1.6: With both `start-after` and `lines` in use, the first line as per `start-after` is considered to be with line number 1 for `lines`.

Changed in version 2.1: Added the `force` option.

Changed in version 3.5: Support automatic dedent.

Glossary

.. glossary::

This directive must contain a reST definition-list-like markup with terms and definitions. The definitions will then be referenceable with the *term* role. Example:

```
.. glossary::

   environment
      A structure where information about all documents under the root is
      saved, and used for cross-referencing.  The environment is pickled
      after the parsing stage, so that successive runs only need to read
      and parse new and changed documents.

   source directory
```

(continues on next page)

[115] There is a standard `.. include` directive, but it raises errors if the file is not found. This one only emits a warning.

(continued from previous page)

```
      The directory which, including its subdirectories, contains all
      source files for one Sphinx project.
```

In contrast to regular definition lists, *multiple* terms per entry are allowed, and inline markup is allowed in terms. You can link to all of the terms. For example:

```
.. glossary::

   term 1
   term 2
      Definition of both terms.
```

(When the glossary is sorted, the first term determines the sort order.)

If you want to specify "grouping key" for general index entries, you can put a "key" as "term : key". For example:

```
.. glossary::

   term 1 : A
   term 2 : B
      Definition of both terms.
```

Note that "key" is used for grouping key as is. The "key" isn't normalized; key "A" and "a" become different groups. The whole characters in "key" is used instead of a first character; it is used for "Combining Character Sequence" and "Surrogate Pairs" grouping key.

In i18n situation, you can specify "localized term : key" even if original text only have "term" part. In this case, translated "localized term" will be categorized in "key" group.

New in version 0.6: You can now give the glossary directive a `:sorted:` flag that will automatically sort the entries alphabetically.

Changed in version 1.1: Now supports multiple terms and inline markup in terms.

Changed in version 1.4: Index key for glossary term should be considered *experimental*.

Meta-information markup

.. sectionauthor:: `name <email>`
Identifies the author of the current section. The argument should include the author's name such that it can be used for presentation and email address. The domain name portion of the address should be lower case. Example:

```
.. sectionauthor:: Guido van Rossum <guido@python.org>
```

By default, this markup isn't reflected in the output in any way (it helps keep track of contributions), but you can set the configuration value *show_authors* to `True` to make them produce a paragraph in the output.

.. codeauthor:: `name <email>`
The *codeauthor* directive, which can appear multiple times, names the authors of the described code, just like *sectionauthor* names the author(s) of a piece of documentation. It too only produces output if the *show_authors* configuration value is `True`.

Index-generating markup

Sphinx automatically creates index entries from all object descriptions (like functions, classes or attributes) like discussed in *Domains*.

However, there is also explicit markup available, to make the index more comprehensive and enable index entries in documents where information is not mainly contained in information units, such as the language reference.

.. index:: `<entries>`

This directive contains one or more index entries. Each entry consists of a type and a value, separated by a colon.

For example:

```
.. index::
   single: execution; context
   module: __main__
   module: sys
   triple: module; search; path

The execution context
---------------------

...
```

This directive contains five entries, which will be converted to entries in the generated index which link to the exact location of the index statement (or, in case of offline media, the corresponding page number).

Since index directives generate cross-reference targets at their location in the source, it makes sense to put them *before* the thing they refer to – e.g. a heading, as in the example above.

The possible entry types are:

single Creates a single index entry. Can be made a subentry by separating the subentry text with a semicolon (this notation is also used below to describe what entries are created).

pair `pair: loop; statement` is a shortcut that creates two index entries, namely `loop; statement` and `statement; loop`.

triple Likewise, `triple: module; search; path` is a shortcut that creates three index entries, which are `module; search path`, `search; path, module` and `path; module search`.

see `see: entry; other` creates an index entry that refers from `entry` to `other`.

seealso Like `see`, but inserts “see also” instead of “see”.

module, keyword, operator, object, exception, statement, builtin These all create two index entries. For example, `module: hashlib` creates the entries `module; hashlib` and `hashlib; module`. (These are Python-specific and therefore deprecated.)

You can mark up “main” index entries by prefixing them with an exclamation mark. The references to “main” entries are emphasized in the generated index. For example, if two pages contain

```
.. index:: Python
```

and one page contains

```
.. index:: ! Python
```

then the backlink to the latter page is emphasized among the three backlinks.

For index directives containing only “single” entries, there is a shorthand notation:

```
.. index:: BNF, grammar, syntax, notation
```

This creates four index entries.

Changed in version 1.1: Added `see` and `seealso` types, as well as marking main entries.

options

`:name: a label for hyperlink (text)`

Define implicit target name that can be referenced by using *`ref`*. For example:

```
.. index:: Python
   :name: py-index
```

New in version 3.0.

`:index:`

While the *`index`* directive is a block-level markup and links to the beginning of the next paragraph, there is also a corresponding role that sets the link target directly where it is used.

The content of the role can be a simple phrase, which is then kept in the text and used as an index entry. It can also be a combination of text and index entry, styled like with explicit targets of cross-references. In that case, the "target" part can be a full entry as described for the directive above. For example:

```
This is a normal reST :index:`paragraph` that contains several
:index:`index entries <pair: index; entry>`.
```

New in version 1.1.

Including content based on tags

`.. only:: <expression>`

Include the content of the directive only if the *expression* is true. The expression should consist of tags, like this:

```
.. only:: html and draft
```

Undefined tags are false, defined tags (via the `-t` command-line option or within `conf.py`, see *here*) are true. Boolean expressions, also using parentheses (like `html and (latex or draft)`) are supported.

The *format* and the *name* of the current builder (`html`, `latex` or `text`) are always set as a tag[116]. To make the distinction between format and name explicit, they are also added with the prefix `format_` and `builder_`, e.g. the epub builder defines the tags `html`, `epub`, `format_html` and `builder_epub`.

These standard tags are set *after* the configuration file is read, so they are not available there.

All tags must follow the standard Python identifier syntax as set out in the Identifiers and keywords[105] documentation. That is, a tag expression may only consist of tags that conform to the syntax of Python variables. In ASCII, this consists of the uppercase and lowercase letters `A` through `Z`, the underscore `_` and, except for the first character, the digits `0` through `9`.

New in version 0.6.

Changed in version 1.2: Added the name of the builder and the prefixes.

Warning: This directive is designed to control only content of document. It could not control sections, labels and so on.

Tables

Use *reStructuredText tables*, i.e. either

- grid table syntax (ref[106]),
- simple table syntax (ref[107]),
- csv-table[108] syntax,
- or list-table[109] syntax.

The table[110] directive serves as optional wrapper of the *grid* and *simple* syntaxes.

They work fine in HTML output, however there are some gotchas when using tables in LaTeX: the column width is hard to determine correctly automatically. For this reason, the following directive exists:

.. tabularcolumns:: `column spec`

This directive gives a "column spec" for the next table occurring in the source file. The spec is the second argument to the LaTeX `tabulary` package's environment (which Sphinx uses to translate tables). It can have values like

```
|l|l|l|
```

which means three left-adjusted, nonbreaking columns. For columns with longer text that should automatically be broken, use either the standard `p{width}` construct, or tabulary's automatic specifiers:

L	flush left column with automatic width
R	flush right column with automatic width
C	centered column with automatic width
J	justified column with automatic width

The automatic widths of the `LRCJ` columns are attributed by `tabulary` in proportion to the observed shares in a first pass where the table cells are rendered at their natural "horizontal" widths.

By default, Sphinx uses a table layout with `J` for every column.

New in version 0.3.

Changed in version 1.6: Merged cells may now contain multiple paragraphs and are much better handled, thanks to custom Sphinx LaTeX macros. This novel situation motivated the switch to `J` specifier and not `L` by default.

Hint: Sphinx actually uses `T` specifier having done `\newcolumntype{T}{J}`. To revert to previous default, insert `\newcolumntype{T}{L}` in the LaTeX preamble (see `latex_elements`).

A frequent issue with tabulary is that columns with little contents are "squeezed". The minimal column width is a tabulary parameter called `\tymin`. You may set it globally in the LaTeX preamble via `\setlength{\tymin}{40pt}` for example.

Else, use the `tabularcolumns` directive with an explicit `p{40pt}` (for example) for that column. You may use also `l` specifier but this makes the task of setting column widths more difficult if some merged cell intersects that column.

[116] For most builders name and format are the same. At the moment only builders derived from the html builder distinguish between the builder format and the builder name.

Note that the current builder tag is not available in `conf.py`, it is only available after the builder is initialized.

[105] https://docs.python.org/3/reference/lexical_analysis.html#identifiers
[106] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#grid-tables
[107] http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html#simple-tables
[108] http://docutils.sourceforge.net/docs/ref/rst/directives.html#csv-table
[109] http://docutils.sourceforge.net/docs/ref/rst/directives.html#list-table
[110] http://docutils.sourceforge.net/docs/ref/rst/directives.html#table

Warning: Tables with more than 30 rows are rendered using `longtable`, not `tabulary`, in order to allow pagebreaks. The `L`, `R`, … specifiers do not work for these tables.

Tables that contain list-like elements such as object descriptions, blockquotes or any kind of lists cannot be set out of the box with `tabulary`. They are therefore set with the standard LaTeX `tabular` (or `longtable`) environment if you don't give a `tabularcolumns` directive. If you do, the table will be set with `tabulary` but you must use the `p{width}` construct (or Sphinx's `\X` and `\Y` specifiers described below) for the columns containing these elements.

Literal blocks do not work with `tabulary` at all, so tables containing a literal block are always set with `tabular`. The verbatim environment used for literal blocks only works in `p{width}` (and `\X` or `\Y`) columns, hence Sphinx generates such column specs for tables containing literal blocks.

Since Sphinx 1.5, the `\X{a}{b}` specifier is used (there *is* a backslash in the specifier letter). It is like `p{width}` with the width set to a fraction `a/b` of the current line width. You can use it in the `tabularcolumns` (it is not a problem if some LaTeX macro is also called `\X`.)

It is *not* needed for `b` to be the total number of columns, nor for the sum of the fractions of the `\X` specifiers to add up to one. For example `|\X{2}{5}|\X{1}{5}|\X{1}{5}|` is legitimate and the table will occupy 80% of the line width, the first of its three columns having the same width as the sum of the next two.

This is used by the `:widths:` option of the table[111] directive.

Since Sphinx 1.6, there is also the `\Y{f}` specifier which admits a decimal argument, such has `\Y{0.15}`: this would have the same effect as `\X{3}{20}`.

Changed in version 1.6: Merged cells from complex grid tables (either multi-row, multi-column, or both) now allow blockquotes, lists, literal blocks, … as do regular cells.

Sphinx's merged cells interact well with `p{width}`, `\X{a}{b}`, `\Y{f}` and tabulary's columns.

Note: `tabularcolumns` conflicts with `:widths:` option of table directives. If both are specified, `:widths:` option will be ignored.

Math

The input language for mathematics is LaTeX markup. This is the de-facto standard for plain-text math notation and has the added advantage that no further translation is necessary when building LaTeX output.

Keep in mind that when you put math markup in **Python docstrings** read by *autodoc*, you either have to double all backslashes, or use Python raw strings (`r"raw"`).

.. math::

Directive for displayed math (math that takes the whole line for itself).

The directive supports multiple equations, which should be separated by a blank line:

```
.. math::

   (a + b)^2 = a^2 + 2ab + b^2

   (a - b)^2 = a^2 - 2ab + b^2
```

In addition, each single equation is set within a `split` environment, which means that you can have multiple aligned lines in an equation, aligned at `&` and separated by `\\`:

[111] http://docutils.sourceforge.net/docs/ref/rst/directives.html#table

```
.. math::

   (a + b)^2  &=  (a + b)(a + b) \\
              &=  a^2 + 2ab + b^2
```

For more details, look into the documentation of the AmSMath LaTeX package[112].

When the math is only one line of text, it can also be given as a directive argument:

```
.. math:: (a + b)^2 = a^2 + 2ab + b^2
```

Normally, equations are not numbered. If you want your equation to get a number, use the `label` option. When given, it selects an internal label for the equation, by which it can be cross-referenced, and causes an equation number to be issued. See *eq* for an example. The numbering style depends on the output format.

There is also an option `nowrap` that prevents any wrapping of the given math in a math environment. When you give this option, you must make sure yourself that the math is properly set up. For example:

```
.. math::
   :nowrap:

   \begin{eqnarray}
      y    & = & ax^2 + bx + c \\
      f(x) & = & x^2 + 2xy + y^2
   \end{eqnarray}
```

See also:

Math support for HTML outputs in Sphinx Rendering options for math with HTML builders.

latex_engine Explains how to configure LaTeX builder to support Unicode literals in math mark-up.

Grammar production displays

Special markup is available for displaying the productions of a formal grammar. The markup is simple and does not attempt to model all aspects of BNF (or any derived forms), but provides enough to allow context-free grammars to be displayed in a way that causes uses of a symbol to be rendered as hyperlinks to the definition of the symbol. There is this directive:

.. productionlist:: [productionGroup]

This directive is used to enclose a group of productions. Each production is given on a single line and consists of a name, separated by a colon from the following definition. If the definition spans multiple lines, each continuation line must begin with a colon placed at the same column as in the first line. Blank lines are not allowed within `productionlist` directive arguments.

The definition can contain token names which are marked as interpreted text (e.g., "`sum ::= `integer` "+" `integer``") – this generates cross-references to the productions of these tokens. Outside of the production list, you can reference to token productions using *token*.

The *productionGroup* argument to *productionlist* serves to distinguish different sets of production lists that belong to different grammars. Multiple production lists with the same *productionGroup* thus define rules in the same scope.

Inside of the production list, tokens implicitly refer to productions from the current group. You can refer to the production of another grammar by prefixing the token with its group name and a colon, e.g, "`otherGroup:sum`". If the group of the token should not be shown in the production, it can be prefixed by

[112] https://www.ams.org/publications/authors/tex/amslatex

a tilde, e.g., "~otherGroup:sum". To refer to a production from an unnamed grammar, the token should be prefixed by a colon, e.g., ":sum".

Outside of the production list, if you have given a *productionGroup* argument you must prefix the token name in the cross-reference with the group name and a colon, e.g., "myGroup:sum" instead of just "sum". If the group should not be shown in the title of the link either an explicit title can be given (e.g., "myTitle <myGroup:sum>"), or the target can be prefixed with a tilde (e.g., "~myGroup:sum").

Note that no further reST parsing is done in the production, so that you don't have to escape * or | characters.

The following is an example taken from the Python Reference Manual:

```
.. productionlist::
   try_stmt: try1_stmt | try2_stmt
   try1_stmt: "try" ":" `suite`
            : ("except" [`expression` ["," `target`]] ":" `suite`)+
            : ["else" ":" `suite`]
            : ["finally" ":" `suite`]
   try2_stmt: "try" ":" `suite`
            : "finally" ":" `suite`
```

Field Lists

As previously discussed, field lists are sequences of fields marked up like this:

```
:fieldname: Field content
```

Sphinx extends standard docutils behavior for field lists and adds some extra functionality that is covered in this section.

Note: The values of field lists will be parsed as strings. You cannot use Python collections such as lists or dictionaries.

File-wide metadata

A field list near the top of a file is normally parsed by docutils as the *docinfo* and shown on the page. However, in Sphinx, a field list preceding any other markup is moved from the *docinfo* to the Sphinx environment as document metadata, and is not displayed in the output.

Note: A field list appearing after the document title *will* be part of the *docinfo* as normal and will be displayed in the output.

Special metadata fields

Sphinx provides custom behavior for bibliographic fields compared to docutils.

At the moment, these metadata fields are recognized:

tocdepth The maximum depth for a table of contents of this file.

```
:tocdepth: 2
```

Note: This metadata effects to the depth of local toctree. But it does not effect to the depth of *global* toctree. So this would not be change the sidebar of some themes which uses global one.

New in version 0.4.

`nocomments` If set, the web application won't display a comment form for a page generated from this source file.

```
:nocomments:
```

`orphan` If set, warnings about this file not being included in any toctree will be suppressed.

```
:orphan:
```

New in version 1.0.

`nosearch` If set, full text search for this file is disabled.

```
:nosearch:
```

Note: object search is still available even if *nosearch* option is set.

New in version 3.0.

Domains

New in version 1.0.

Originally, Sphinx was conceived for a single project, the documentation of the Python language. Shortly afterwards, it was made available for everyone as a documentation tool, but the documentation of Python modules remained deeply built in – the most fundamental directives, like `function`, were designed for Python objects. Since Sphinx has become somewhat popular, interest developed in using it for many different purposes: C/C++ projects, JavaScript, or even reStructuredText markup (like in this documentation).

While this was always possible, it is now much easier to easily support documentation of projects using different programming languages or even ones not supported by the main Sphinx distribution, by providing a **domain** for every such purpose.

A domain is a collection of markup (reStructuredText *directive*s and *role*s) to describe and link to *object*s belonging together, e.g. elements of a programming language. Directive and role names in a domain have names like `domain:name`, e.g. `py:function`. Domains can also provide custom indices (like the Python Module Index).

Having domains means that there are no naming problems when one set of documentation wants to refer to e.g. C++ and Python classes. It also means that extensions that support the documentation of whole new languages are much easier to write.

This section describes what the domains that are included with Sphinx provide. The domain API is documented as well, in the section *Domain API*.

Basic Markup

Most domains provide a number of *object description directives*, used to describe specific objects provided by modules. Each directive requires one or more signatures to provide basic information about what is being described, and the content should be the description. A domain will typically keep an internal index of all entites to aid cross-referencing. Typically it will also add entries in the shown general index. If you want to suppress the addition of an entry in the shown index, you can give the directive option flag `:noindexentry:`. If you want to typeset an object description, without even making it available for cross-referencing, you can give the directive option flag `:noindex:` (which implies `:noindexentry:`). Though, note that not every directive en every domain may support these options.

New in version 3.2: The directive option `noindexentry` in the Python, C, C++, and Javascript domains.

An example using a Python domain directive:

```
.. py:function:: spam(eggs)
                 ham(eggs)

   Spam or ham the foo.
```

This describes the two Python functions `spam` and `ham`. (Note that when signatures become too long, you can break them if you add a backslash to lines that are continued in the next line. Example:

```
.. py:function:: filterwarnings(action, message='', category=Warning, \
                                module='', lineno=0, append=False)
   :noindex:
```

(This example also shows how to use the `:noindex:` flag.)

The domains also provide roles that link back to these object descriptions. For example, to link to one of the functions described in the example above, you could say

```
The function :py:func:`spam` does a similar thing.
```

As you can see, both directive and role names contain the domain name and the directive name.

Default Domain

For documentation describing objects from solely one domain, authors will not have to state again its name at each directive, role, etc… after having specified a default. This can be done either via the config value *`primary_domain`* or via this directive:

`.. default-domain::` `name`

Select a new default domain. While the *`primary_domain`* selects a global default, this only has an effect within the same file.

If no other default is selected, the Python domain (named `py`) is the default one, mostly for compatibility with documentation written for older versions of Sphinx.

Directives and roles that belong to the default domain can be mentioned without giving the domain name, i.e.

```
.. function:: pyfunc()

   Describes a Python function.

Reference to :func:`pyfunc`.
```

Cross-referencing syntax

For cross-reference roles provided by domains, the same facilities exist as for general cross-references. See *Cross-referencing syntax*.

In short:

- You may supply an explicit title and reference target: ``:role:`title <target>` `` will refer to *target*, but the link text will be *title*.
- If you prefix the content with `!`, no reference/hyperlink will be created.
- If you prefix the content with `~`, the link text will only be the last component of the target. For example, ``:py:meth:`~Queue.Queue.get` `` will refer to `Queue.Queue.get` but only display `get` as the link text.

The Python Domain

The Python domain (name **py**) provides the following directives for module declarations:

.. py:module:: `name`

This directive marks the beginning of the description of a module (or package submodule, in which case the name should be fully qualified, including the package name). It does not create content (like e.g. *`py:class`* does).

This directive will also cause an entry in the global module index.

options

:platform: platforms (comma separated list)

Indicate platforms which the module is available (if it is available on all platforms, the option should be omitted). The keys are short identifiers; examples that are in use include "IRIX", "Mac", "Windows" and "Unix". It is important to use a key which has already been used when applicable.

:synopsis: purpose (text)

Consist of one sentence describing the module's purpose – it is currently only used in the Global Module Index.

:deprecated: (no argument)

Mark a module as deprecated; it will be designated as such in various locations then.

.. py:currentmodule:: `name`

This directive tells Sphinx that the classes, functions etc. documented from here are in the given module (like *`py:module`*), but it will not create index entries, an entry in the Global Module Index, or a link target for *`py:mod`*. This is helpful in situations where documentation for things in a module is spread over multiple files or sections – one location has the *`py:module`* directive, the others only *`py:currentmodule`*.

The following directives are provided for module and class contents:

.. py:function:: `name(parameters)`

Describes a module-level function. The signature should include the parameters as given in the Python function definition, see *Python Signatures*. For example:

```
.. py:function:: Timer.repeat(repeat=3, number=1000000)
```

For methods you should use *`py:method`*.

The description normally includes information about the parameters required and how they are used (especially whether mutable objects passed as parameters are modified), side effects, and possible exceptions.

This information can (in any `py` directive) optionally be given in a structured form, see *Info field lists*.

options

:async: (no value)
Indicate the function is an async function.

New in version 2.1.

:canonical: (full qualified name including module name)
Describe the location where the object is defined if the object is imported from other modules

New in version 4.0.

.. py:data:: name
Describes global data in a module, including both variables and values used as "defined constants." Class and object attributes are not documented using this environment.

options

:type: type of the variable (text)
New in version 2.4.

:value: initial value of the variable (text)
New in version 2.4.

:canonical: (full qualified name including module name)
Describe the location where the object is defined if the object is imported from other modules

New in version 4.0.

.. py:exception:: name
Describes an exception class. The signature can, but need not include parentheses with constructor arguments.

options

:final: (no value)
Indicate the class is a final class.

New in version 3.1.

.. py:class:: name
.. py:class:: name(parameters)
Describes a class. The signature can optionally include parentheses with parameters which will be shown as the constructor arguments. See also *Python Signatures*.

Methods and attributes belonging to the class should be placed in this directive's body. If they are placed outside, the supplied name should contain the class name so that cross-references still work. Example:

```
.. py:class:: Foo

   .. py:method:: quux()

-- or --

.. py:class:: Bar

.. py:method:: Bar.quux()
```

The first way is the preferred one.

options

`:canonical: (full qualified name including module name)`
: Describe the location where the object is defined if the object is imported from other modules

  New in version 4.0.

`:final: (no value)`
: Indicate the class is a final class.

  New in version 3.1.

`.. py:attribute::` `name`

Describes an object data attribute. The description should include information about the type of the data to be expected and whether it may be changed directly.

options

`:type: type of the attribute (text)`
: New in version 2.4.

`:value: initial value of the attribute (text)`
: New in version 2.4.

`:canonical: (full qualified name including module name)`
: Describe the location where the object is defined if the object is imported from other modules

  New in version 4.0.

`.. py:method::` `name(parameters)`

Describes an object method. The parameters should not include the `self` parameter. The description should include similar information to that described for `function`. See also *Python Signatures* and *Info field lists*.

options

`:abstractmethod: (no value)`
: Indicate the method is an abstract method.

  New in version 2.1.

`:async: (no value)`
: Indicate the method is an async method.

  New in version 2.1.

`:canonical: (full qualified name including module name)`
: Describe the location where the object is defined if the object is imported from other modules

  New in version 4.0.

`:classmethod: (no value)`
: Indicate the method is a class method.

  New in version 2.1.

`:final: (no value)`
: Indicate the class is a final method.

  New in version 3.1.

`:property: (no value)`
: Indicate the method is a property.

  New in version 2.1.

:staticmethod: (no value)
Indicate the method is a static method.

New in version 2.1.

.. py:staticmethod:: name(parameters)
Like *py:method*, but indicates that the method is a static method.

New in version 0.4.

.. py:classmethod:: name(parameters)
Like *py:method*, but indicates that the method is a class method.

New in version 0.6.

.. py:decorator:: name
.. py:decorator:: name(parameters)
Describes a decorator function. The signature should represent the usage as a decorator. For example, given the functions

```
def removename(func):
    func.__name__ = ''
    return func

def setnewname(name):
    def decorator(func):
        func.__name__ = name
        return func
    return decorator
```

the descriptions should look like this:

```
.. py:decorator:: removename

   Remove name of the decorated function.

.. py:decorator:: setnewname(name)

   Set name of the decorated function to *name*.
```

(as opposed to `.. py:decorator:: removename(func)`.)

There is no `py:deco` role to link to a decorator that is marked up with this directive; rather, use the *py:func* role.

.. py:decoratormethod:: name
.. py:decoratormethod:: name(signature)
Same as *py:decorator*, but for decorators that are methods.

Refer to a decorator method using the *py:meth* role.

Python Signatures

Signatures of functions, methods and class constructors can be given like they would be written in Python.

Default values for optional arguments can be given (but if they contain commas, they will confuse the signature parser). Python 3-style argument annotations can also be given as well as return type annotations:

```
.. py:function:: compile(source : string, filename, symbol='file') -> ast object
```

For functions with optional parameters that don't have default values (typically functions implemented in C extension modules without keyword argument support), you can use brackets to specify the optional parts:

compile(*source*[, *filename*[, *symbol*]])

It is customary to put the opening bracket before the comma.

Info field lists

New in version 0.4.

Changed in version 3.0: meta fields are added.

Inside Python object description directives, reST field lists with these fields are recognized and formatted nicely:

- `param`, `parameter`, `arg`, `argument`, `key`, `keyword`: Description of a parameter.
- `type`: Type of a parameter. Creates a link if possible.
- `raises`, `raise`, `except`, `exception`: That (and when) a specific exception is raised.
- `var`, `ivar`, `cvar`: Description of a variable.
- `vartype`: Type of a variable. Creates a link if possible.
- `returns`, `return`: Description of the return value.
- `rtype`: Return type. Creates a link if possible.
- `meta`: Add metadata to description of the python object. The metadata will not be shown on output document. For example, `:meta private:` indicates the python object is private member. It is used in *`sphinx.ext.autodoc`* for filtering members.

Note: In current release, all `var`, `ivar` and `cvar` are represented as "Variable". There is no difference at all.

The field names must consist of one of these keywords and an argument (except for `returns` and `rtype`, which do not need an argument). This is best explained by an example:

```
.. py:function:: send_message(sender, recipient, message_body, [priority=1])

   Send a message to a recipient

   :param str sender: The person sending the message
   :param str recipient: The recipient of the message
   :param str message_body: The body of the message
   :param priority: The priority of the message, can be a number 1-5
   :type priority: integer or None
   :return: the message id
   :rtype: int
   :raises ValueError: if the message_body exceeds 160 characters
   :raises TypeError: if the message_body is not a basestring
```

This will render like this:

send_message(*sender*, *recipient*, *message_body*[, *priority=1*])

Send a message to a recipient

Parameters

- **sender** (*str*[117]) – The person sending the message
- **recipient** (*str*[118]) – The recipient of the message
- **message_body** (*str*[119]) – The body of the message
- **priority** (*integer or None*[120]) – The priority of the message, can be a number 1-5

Returns the message id

Return type int[121]

Raises

- **ValueError**[122] – if the message_body exceeds 160 characters
- **TypeError**[123] – if the message_body is not a basestring

It is also possible to combine parameter type and description, if the type is a single word, like this:

```
:param int priority: The priority of the message, can be a number 1-5
```

New in version 1.5.

Container types such as lists and dictionaries can be linked automatically using the following syntax:

```
:type priorities: list(int)
:type priorities: list[int]
:type mapping: dict(str, int)
:type mapping: dict[str, int]
:type point: tuple(float, float)
:type point: tuple[float, float]
```

Multiple types in a type field will be linked automatically if separated by the word “or”:

```
:type an_arg: int or None
:vartype a_var: str or int
:rtype: float or str
```

[117] https://docs.python.org/3/library/stdtypes.html#str
[118] https://docs.python.org/3/library/stdtypes.html#str
[119] https://docs.python.org/3/library/stdtypes.html#str
[120] https://docs.python.org/3/library/constants.html#None
[121] https://docs.python.org/3/library/functions.html#int
[122] https://docs.python.org/3/library/exceptions.html#ValueError
[123] https://docs.python.org/3/library/exceptions.html#TypeError

Cross-referencing Python objects

The following roles refer to objects in modules and are possibly hyperlinked if a matching identifier is found:

`:py:mod:`
: Reference a module; a dotted name may be used. This should also be used for package names.

`:py:func:`
: Reference a Python function; dotted names may be used. The role text needs not include trailing parentheses to enhance readability; they will be added automatically by Sphinx if the *`add_function_parentheses`* config value is `True` (the default).

`:py:data:`
: Reference a module-level variable.

`:py:const:`
: Reference a "defined" constant. This may be a Python variable that is not intended to be changed.

`:py:class:`
: Reference a class; a dotted name may be used.

`:py:meth:`
: Reference a method of an object. The role text can include the type name and the method name; if it occurs within the description of a type, the type name can be omitted. A dotted name may be used.

`:py:attr:`
: Reference a data attribute of an object.

`:py:exc:`
: Reference an exception. A dotted name may be used.

`:py:obj:`
: Reference an object of unspecified type. Useful e.g. as the *`default_role`*.

  New in version 0.4.

The name enclosed in this markup can include a module name and/or a class name. For example, `` :py:func:`filter` `` could refer to a function named `filter` in the current module, or the built-in function of that name. In contrast, `` :py:func:`foo.filter` `` clearly refers to the `filter` function in the `foo` module.

Normally, names in these roles are searched first without any further qualification, then with the current module name prepended, then with the current module and class name (if any) prepended. If you prefix the name with a dot, this order is reversed. For example, in the documentation of Python's `codecs`[124] module, `` :py:func:`open` `` always refers to the built-in function, while `` :py:func:`.open` `` refers to `codecs.open()`[125].

A similar heuristic is used to determine whether the name is an attribute of the currently documented class.

Also, if the name is prefixed with a dot, and no exact match is found, the target is taken as a suffix and all object names with that suffix are searched. For example, `` :py:meth:`.TarFile.close` `` references the `tarfile.TarFile.close()` function, even if the current module is not `tarfile`. Since this can get ambiguous, if there is more than one possible match, you will get a warning from Sphinx.

Note that you can combine the `~` and `.` prefixes: `` :py:meth:`~.TarFile.close` `` will reference the `tarfile.TarFile.close()` method, but the visible link caption will only be `close()`.

[124] https://docs.python.org/3/library/codecs.html#module-codecs
[125] https://docs.python.org/3/library/codecs.html#codecs.open

The C Domain

The C domain (name **c**) is suited for documentation of C API.

.. c:member:: `declaration`
.. c:var:: `declaration`

Describes a C struct member or variable. Example signature:

```
.. c:member:: PyObject *PyTypeObject.tp_bases
```

The difference between the two directives is only cosmetic.

.. c:function:: `function prototype`

Describes a C function. The signature should be given as in C, e.g.:

```
.. c:function:: PyObject *PyType_GenericAlloc(PyTypeObject *type, Py_ssize_t nitems)
```

Note that you don't have to backslash-escape asterisks in the signature, as it is not parsed by the reST inliner.

.. c:macro:: `name`
.. c:macro:: `name(arg list)`

Describes a C macro, i.e., a C-language `#define`, without the replacement text.

New in version 3.0: The function style variant.

.. c:struct:: `name`

Describes a C struct.

New in version 3.0.

.. c:union:: `name`

Describes a C union.

New in version 3.0.

.. c:enum:: `name`

Describes a C enum.

New in version 3.0.

.. c:enumerator:: `name`

Describes a C enumerator.

New in version 3.0.

.. c:type:: `typedef-like declaration`
.. c:type:: `name`

Describes a C type, either as a typedef, or the alias for an unspecified type.

Cross-referencing C constructs

The following roles create cross-references to C-language constructs if they are defined in the documentation:

:c:member:
:c:data:
:c:var:
:c:func:
:c:macro:
:c:struct:
:c:union:
:c:enum:
:c:enumerator:

:c:type:

Reference a C declaration, as defined above. Note that *c:member*, *c:data*, and *c:var* are equivalent.

New in version 3.0: The var, struct, union, enum, and enumerator roles.

Anonymous Entities

C supports anonymous structs, enums, and unions. For the sake of documentation they must be given some name that starts with `@`, e.g., `@42` or `@data`. These names can also be used in cross-references, though nested symbols will be found even when omitted. The `@...` name will always be rendered as **[anonymous]** (possibly as a link).

Example:

```
.. c:struct:: Data

   .. c:union:: @data

      .. c:var:: int a

      .. c:var:: double b

Explicit ref: :c:var:`Data.@data.a`. Short-hand ref: :c:var:`Data.a`.
```

This will be rendered as:

struct Data

union [anonymous]

int **a**

double **b**

Explicit ref: *Data.[anonymous].a*. Short-hand ref: *Data.a*.

New in version 3.0.

Aliasing Declarations

Sometimes it may be helpful list declarations elsewhere than their main documentation, e.g., when creating a synopsis of an interface. The following directive can be used for this purpose.

.. c:alias:: name

Insert one or more alias declarations. Each entity can be specified as they can in the `c:any` role.

For example:

```
.. c:var:: int data
.. c:function:: int f(double k)

.. c:alias:: data
             f
```

becomes

int **data**

int **f**(double *k*)

int *data*

int *f* (double k)

New in version 3.2.

Options

:maxdepth: int

Insert nested declarations as well, up to the total depth given. Use 0 for infinite depth and 1 for just the mentioned declaration. Defaults to 1.

New in version 3.3.

:noroot:

Skip the mentioned declarations and only render nested declarations. Requires `maxdepth` either 0 or at least 2.

New in version 3.5.

Inline Expressions and Types

:c:expr:
:c:texpr:

Insert a C expression or type either as inline code (`cpp:expr`) or inline text (`cpp:texpr`). For example:

```
.. c:var:: int a = 42

.. c:function:: int f(int i)

An expression: :c:expr:`a * f(a)` (or as text: :c:texpr:`a * f(a)`).

A type: :c:expr:`const Data*`
(or as text :c:texpr:`const Data*`).
```

will be rendered as follows:

int **a** = 42

int **f**(int *i*)

An expression: `a * f(a)` (or as text: *a* * *f*(*a*)).

A type: `const Data*` (or as text **const** *Data**).

New in version 3.0.

Namespacing

New in version 3.1.

The C language it self does not support namespacing, but it can sometimes be useful to emulate it in documentation, e.g., to show alternate declarations. The feature may also be used to document members of structs/unions/enums separate from their parent declaration.

The current scope can be changed using three namespace directives. They manage a stack declarations where `c:namespace` resets the stack and changes a given scope.

The `c:namespace-push` directive changes the scope to a given inner scope of the current one.

The `c:namespace-pop` directive undoes the most recent `c:namespace-push` directive.

.. c:namespace:: scope specification

Changes the current scope for the subsequent objects to the given scope, and resets the namespace directive stack. Note that nested scopes can be specified by separating with a dot, e.g.:

```
.. c:namespace:: Namespace1.Namespace2.SomeStruct.AnInnerStruct
```

All subsequent objects will be defined as if their name were declared with the scope prepended. The subsequent cross-references will be searched for starting in the current scope.

Using `NULL` or `0` as the scope will change to global scope.

.. c:namespace-push:: scope specification

Change the scope relatively to the current scope. For example, after:

```
.. c:namespace:: A.B

.. c:namespace-push:: C.D
```

the current scope will be `A.B.C.D`.

.. c:namespace-pop::

Undo the previous `c:namespace-push` directive (*not* just pop a scope). For example, after:

```
.. c:namespace:: A.B

.. c:namespace-push:: C.D

.. c:namespace-pop::
```

the current scope will be `A.B` (*not* `A.B.C`).

If no previous `c:namespace-push` directive has been used, but only a `c:namespace` directive, then the current scope will be reset to global scope. That is, `.. c:namespace:: A.B` is equivalent to:

```
.. c:namespace:: NULL

.. c:namespace-push:: A.B
```

Configuration Variables

See *Options for the C domain*.

The C++ Domain

The C++ domain (name **cpp**) supports documenting C++ projects.

Directives for Declaring Entities

The following directives are available. All declarations can start with a visibility statement (`public`, `private` or `protected`).

.. cpp:class:: `class specifier`

.. cpp:struct:: `class specifier`

Describe a class/struct, possibly with specification of inheritance, e.g.,:

```
.. cpp:class:: MyClass : public MyBase, MyOtherBase
```

The difference between *`cpp:class`* and *`cpp:struct`* is only cosmetic: the prefix rendered in the output, and the specifier shown in the index.

The class can be directly declared inside a nested scope, e.g.,:

```
.. cpp:class:: OuterScope::MyClass : public MyBase, MyOtherBase
```

A class template can be declared:

```
.. cpp:class:: template<typename T, std::size_t N> std::array
```

or with a line break:

```
.. cpp:class:: template<typename T, std::size_t N> \
               std::array
```

Full and partial template specialisations can be declared:

```
.. cpp:class:: template<> \
               std::array<bool, 256>

.. cpp:class:: template<typename T> \
               std::array<T, 42>
```

New in version 2.0: The *`cpp:struct`* directive.

.. cpp:function:: `(member) function prototype`

Describe a function or member function, e.g.,:

```
.. cpp:function:: bool myMethod(int arg1, std::string arg2)

   A function with parameters and types.

.. cpp:function:: bool myMethod(int, double)

   A function with unnamed parameters.

.. cpp:function:: const T &MyClass::operator[](std::size_t i) const

   An overload for the indexing operator.

.. cpp:function:: operator bool() const

   A casting operator.
```

(continues on next page)

(continued from previous page)

```
.. cpp:function:: constexpr void foo(std::string &bar[2]) noexcept

   A constexpr function.

.. cpp:function:: MyClass::MyClass(const MyClass&) = default

   A copy constructor with default implementation.
```

Function templates can also be described:

```
.. cpp:function:: template<typename U> \
                  void print(U &&u)
```

and function template specialisations:

```
.. cpp:function:: template<> \
                  void print(int i)
```

.. cpp:member:: `(member) variable declaration`
.. cpp:var:: `(member) variable declaration`

Describe a variable or member variable, e.g.,:

```
.. cpp:member:: std::string MyClass::myMember

.. cpp:var:: std::string MyClass::myOtherMember[N][M]

.. cpp:member:: int a = 42
```

Variable templates can also be described:

```
.. cpp:member:: template<class T> \
                constexpr T pi = T(3.1415926535897932385)
```

.. cpp:type:: `typedef declaration`
.. cpp:type:: `name`
.. cpp:type:: `type alias declaration`

Describe a type as in a typedef declaration, a type alias declaration, or simply the name of a type with unspecified type, e.g.,:

```
.. cpp:type:: std::vector<int> MyList

   A typedef-like declaration of a type.

.. cpp:type:: MyContainer::const_iterator

   Declaration of a type alias with unspecified type.

.. cpp:type:: MyType = std::unordered_map<int, std::string>

   Declaration of a type alias.
```

A type alias can also be templated:

```
.. cpp:type:: template<typename T> \
              MyContainer = std::vector<T>
```

The example are rendered as follows.

typedef std::vector<int> **MyList**
A typedef-like declaration of a type.

type MyContainer::**const_iterator**
Declaration of a type alias with unspecified type.

using MyType = std::unordered_map<int, std::string>
Declaration of a type alias.

template<typename **T**>
using MyContainer = std::vector<*T*>

.. cpp:enum:: unscoped enum declaration
.. cpp:enum-struct:: scoped enum declaration
.. cpp:enum-class:: scoped enum declaration
Describe a (scoped) enum, possibly with the underlying type specified. Any enumerators declared inside an unscoped enum will be declared both in the enum scope and in the parent scope. Examples:

```
.. cpp:enum:: MyEnum

   An unscoped enum.

.. cpp:enum:: MySpecificEnum : long

   An unscoped enum with specified underlying type.

.. cpp:enum-class:: MyScopedEnum

   A scoped enum.

.. cpp:enum-struct:: protected MyScopedVisibilityEnum : std::underlying_type
↪<MySpecificEnum>::type

   A scoped enum with non-default visibility, and with a specified
   underlying type.
```

.. cpp:enumerator:: name
.. cpp:enumerator:: name = constant
Describe an enumerator, optionally with its value defined, e.g.,:

```
.. cpp:enumerator:: MyEnum::myEnumerator

.. cpp:enumerator:: MyEnum::myOtherEnumerator = 42
```

.. cpp:union:: name
Describe a union.

New in version 1.8.

.. cpp:concept:: template-parameter-list name

Warning: The support for concepts is experimental. It is based on the current draft standard and the Concepts Technical Specification. The features may change as they evolve.

Describe a concept. It must have exactly 1 template parameter list. The name may be a nested name. Example:

```
.. cpp:concept:: template<typename It> std::Iterator

   Proxy to an element of a notional sequence that can be compared,
   indirected, or incremented.

   **Notation**

   .. cpp:var:: It r

      An lvalue.

   **Valid Expressions**

   - :cpp:expr:`*r`, when :cpp:expr:`r` is dereferenceable.
   - :cpp:expr:`++r`, with return type :cpp:expr:`It&`, when
     :cpp:expr:`r` is incrementable.
```

This will render as follows:

template<typename **It**>
concept std::**Iterator**

Proxy to an element of a notional sequence that can be compared, indirected, or incremented.

Notation

It **r**

An lvalue.

Valid Expressions

- `*r`, when `r` is dereferenceable.
- `++r`, with return type `It&`, when `r` is incrementable.

New in version 1.5.

Options

Some directives support options:

- `:noindexentry:`, see *Basic Markup*.
- `:tparam-line-spec:`, for templated declarations. If specified, each template parameter will be rendered on a separate line.

  New in version 1.6.

Anonymous Entities

C++ supports anonymous namespaces, classes, enums, and unions. For the sake of documentation they must be given some name that starts with `@`, e.g., `@42` or `@data`. These names can also be used in cross-references and (type) expressions, though nested symbols will be found even when omitted. The `@...` name will always be rendered as **[anonymous]** (possibly as a link).

Example:

```
.. cpp:class:: Data

   .. cpp:union:: @data

      .. cpp:var:: int a

      .. cpp:var:: double b

Explicit ref: :cpp:var:`Data::@data::a`. Short-hand ref: :cpp:var:`Data::a`.
```

This will be rendered as:

`class Data`

`union` [anonymous]

int **a**

double **b**

Explicit ref: *`Data::[anonymous]::a`*. Short-hand ref: *`Data::a`*.

New in version 1.8.

Aliasing Declarations

Sometimes it may be helpful list declarations elsewhere than their main documentation, e.g., when creating a synopsis of a class interface. The following directive can be used for this purpose.

`.. cpp:alias::` `name or function signature`

Insert one or more alias declarations. Each entity can be specified as they can in the *`cpp:any`* role. If the name of a function is given (as opposed to the complete signature), then all overloads of the function will be listed.

For example:

```
.. cpp:alias:: Data::a
               overload_example::C::f
```

becomes

int *a*

void *f* (double d) **`const`**

void *f* (double d)

void *f* (int i)

void *f* ()

whereas:

```
.. cpp:alias:: void overload_example::C::f(double d) const
               void overload_example::C::f(double d)
```

becomes

void *f* (double d) **const**

void *f* (double d)

New in version 2.0.

Options

:maxdepth: int
Insert nested declarations as well, up to the total depth given. Use 0 for infinite depth and 1 for just the mentioned declaration. Defaults to 1.

New in version 3.5.

:noroot:
Skip the mentioned declarations and only render nested declarations. Requires `maxdepth` either 0 or at least 2.

New in version 3.5.

Constrained Templates

Warning: The support for concepts is experimental. It is based on the current draft standard and the Concepts Technical Specification. The features may change as they evolve.

Note: Sphinx does not currently support `requires` clauses.

Placeholders

Declarations may use the name of a concept to introduce constrained template parameters, or the keyword `auto` to introduce unconstrained template parameters:

```
.. cpp:function:: void f(auto &&arg)

   A function template with a single unconstrained template parameter.

.. cpp:function:: void f(std::Iterator it)

   A function template with a single template parameter, constrained by the
   Iterator concept.
```

Template Introductions

Simple constrained function or class templates can be declared with a *template introduction* instead of a template parameter list:

```
.. cpp:function:: std::Iterator{It} void advance(It &it)

    A function template with a template parameter constrained to be an
    Iterator.

.. cpp:class:: std::LessThanComparable{T} MySortedContainer

    A class template with a template parameter constrained to be
    LessThanComparable.
```

They are rendered as follows.

std::*Iterator*{**It**}
void **advance**(*It* &*it*)

A function template with a template parameter constrained to be an Iterator.

std::LessThanComparable{**T**}
class MySortedContainer

A class template with a template parameter constrained to be LessThanComparable.

Note however that no checking is performed with respect to parameter compatibility. E.g., `Iterator{A, B, C}` will be accepted as an introduction even though it would not be valid C++.

Inline Expressions and Types

:cpp:expr:
:cpp:texpr:

Insert a C++ expression or type either as inline code (`cpp:expr`) or inline text (`cpp:texpr`). For example:

```
.. cpp:var:: int a = 42

.. cpp:function:: int f(int i)

An expression: :cpp:expr:`a * f(a)` (or as text: :cpp:texpr:`a * f(a)`).

A type: :cpp:expr:`const MySortedContainer<int>&`
(or as text :cpp:texpr:`const MySortedContainer<int>&`).
```

will be rendered as follows:

int **a** = 42

int **f**(int *i*)

An expression: `a * f(a)` (or as text: *a* * *f*(*a*)).

A type: `const MySortedContainer<int>&` (or as text **const** *MySortedContainer*<int>&).

New in version 1.7: The *cpp:expr* role.

New in version 1.8: The *cpp:texpr* role.

Namespacing

Declarations in the C++ domain are as default placed in global scope. The current scope can be changed using three namespace directives. They manage a stack declarations where `cpp:namespace` resets the stack and changes a given scope.

The `cpp:namespace-push` directive changes the scope to a given inner scope of the current one.

The `cpp:namespace-pop` directive undoes the most recent `cpp:namespace-push` directive.

.. cpp:namespace:: `scope specification`

Changes the current scope for the subsequent objects to the given scope, and resets the namespace directive stack. Note that the namespace does not need to correspond to C++ namespaces, but can end in names of classes, e.g.,:

```
.. cpp:namespace:: Namespace1::Namespace2::SomeClass::AnInnerClass
```

All subsequent objects will be defined as if their name were declared with the scope prepended. The subsequent cross-references will be searched for starting in the current scope.

Using `NULL`, `0`, or `nullptr` as the scope will change to global scope.

A namespace declaration can also be templated, e.g.,:

```
.. cpp:class:: template<typename T> \
               std::vector

.. cpp:namespace:: template<typename T> std::vector

.. cpp:function:: std::size_t size() const
```

declares `size` as a member function of the class template `std::vector`. Equivalently this could have been declared using:

```
.. cpp:class:: template<typename T> \
               std::vector

   .. cpp:function:: std::size_t size() const
```

or:

```
.. cpp:class:: template<typename T> \
               std::vector
```

.. cpp:namespace-push:: `scope specification`

Change the scope relatively to the current scope. For example, after:

```
.. cpp:namespace:: A::B

.. cpp:namespace-push:: C::D
```

the current scope will be `A::B::C::D`.

New in version 1.4.

.. cpp:namespace-pop::

Undo the previous `cpp:namespace-push` directive (*not* just pop a scope). For example, after:

```
.. cpp:namespace:: A::B

.. cpp:namespace-push:: C::D

.. cpp:namespace-pop::
```

the current scope will be `A::B` (*not* `A::B::C`).

If no previous `cpp:namespace-push` directive has been used, but only a `cpp:namespace` directive, then the current scope will be reset to global scope. That is, `.. cpp:namespace:: A::B` is equivalent to:

```
.. cpp:namespace:: nullptr

.. cpp:namespace-push:: A::B
```

New in version 1.4.

Info field lists

The C++ directives support the following info fields (see also *Info field lists*):

- *param*, *parameter*, *arg*, *argument*: Description of a parameter.
- *tparam*: Description of a template parameter.
- *returns*, *return*: Description of a return value.
- *throws*, *throw*, *exception*: Description of a possibly thrown exception.

Cross-referencing

These roles link to the given declaration types:

`:cpp:any:`
`:cpp:class:`
`:cpp:struct:`
`:cpp:func:`
`:cpp:member:`
`:cpp:var:`
`:cpp:type:`
`:cpp:concept:`
`:cpp:enum:`
`:cpp:enumerator:`

Reference a C++ declaration by name (see below for details). The name must be properly qualified relative to the position of the link.

New in version 2.0: The *`cpp:struct`* role as alias for the *`cpp:class`* role.

Note on References with Templates Parameters/Arguments

These roles follow the Sphinx *Cross-referencing syntax* rules. This means care must be taken when referencing a (partial) template specialization, e.g. if the link looks like this: ``:cpp:class:`MyClass<int>` ``. This is interpreted as a link to `int` with a title of `MyClass`. In this case, escape the opening angle bracket with a backslash, like this: ``:cpp:class:`MyClass\<int>` ``.

When a custom title is not needed it may be useful to use the roles for inline expressions, *cpp:expr* and *cpp:texpr*, where angle brackets do not need escaping.

Declarations without template parameters and template arguments

For linking to non-templated declarations the name must be a nested name, e.g., `f` or `MyClass::f`.

Overloaded (member) functions

When a (member) function is referenced using just its name, the reference will point to an arbitrary matching overload. The *cpp:any* and *cpp:func* roles use an alternative format, which simply is a complete function declaration. This will resolve to the exact matching overload. As example, consider the following class declaration:

class C

> void **f**(double *d*) **const**
>
> void **f**(double *d*)
>
> void **f**(int *i*)
>
> void **f**()

References using the *cpp:func* role:

- Arbitrary overload: `C::f`, *C::f()*
- Also arbitrary overload: `C::f()`, *C::f()*
- Specific overload: `void C::f()`, *void C::f()*
- Specific overload: `void C::f(int)`, *void C::f(int)*
- Specific overload: `void C::f(double)`, *void C::f(double)*
- Specific overload: `void C::f(double) const`, *void C::f(double) const*

Note that the *add_function_parentheses* configuration variable does not influence specific overload references.

Templated declarations

Assume the following declarations.

class Wrapper

> template<typename **TOuter**>
> **class Outer**
>
> > template<typename **TInner**>
> > **class Inner**

In general the reference must include the template parameter declarations, and template arguments for the prefix of qualified names. For example:

- `template\<typename TOuter> Wrapper::Outer` (*template<typename TOuter> Wrapper::Outer*)
- `template\<typename TOuter> template\<typename TInner> Wrapper::Outer<TOuter>::Inner` (template<typename TOuter> template<typename TInner> Wrapper::Outer<TOuter>::Inner)

Currently the lookup only succeed if the template parameter identifiers are equal strings. That is, `template\<typename UOuter> Wrapper::Outer` will not work.

As a shorthand notation, if a template parameter list is omitted, then the lookup will assume either a primary template or a non-template, but not a partial template specialisation. This means the following references work as well:

- `Wrapper::Outer` (*`Wrapper::Outer`*)
- `Wrapper::Outer::Inner` (*`Wrapper::Outer::Inner`*)
- `template\<typename TInner> Wrapper::Outer::Inner` (*`template<typename TInner> Wrapper::Outer::Inner`*)

(Full) Template Specialisations

Assume the following declarations.

template<typename **TOuter**>
class Outer

    template<typename **TInner**>
    class Inner

template<>
class Outer<int>

    template<typename **TInner**>
    class Inner

    template<>
    class Inner<bool>

In general the reference must include a template parameter list for each template argument list. The full specialisation above can therefore be referenced with `template\<> Outer\<int>` (*`template<> Outer<int>`*) and `template\<> template\<> Outer\<int>::Inner\<bool>` (*`template<> template<> Outer<int>::Inner<bool>`*). As a shorthand the empty template parameter list can be omitted, e.g., `Outer\<int>` (*`Outer<int>`*) and `Outer\<int>::Inner\<bool>` (*`Outer<int>::Inner<bool>`*).

Partial Template Specialisations

Assume the following declaration.

template<typename **T**>
class Outer<*T**>

References to partial specialisations must always include the template parameter lists, e.g., `template\<typename T> Outer\<T*>` (*`template<typename T> Outer<T*>`*). Currently the lookup only succeed if the template parameter identifiers are equal strings.

Configuration Variables

See *Options for the C++ domain*.

The Standard Domain

The so-called "standard" domain collects all markup that doesn't warrant a domain of its own. Its directives and roles are not prefixed with a domain name.

The standard domain is also where custom object descriptions, added using the `add_object_type()` API, are placed.

There is a set of directives allowing documenting command-line programs:

.. option:: `name args, name args, ...`

Describes a command line argument or switch. Option argument names should be enclosed in angle brackets. Examples:

```
.. option:: dest_dir

   Destination directory.

.. option:: -m <module>, --module <module>

   Run a module as a script.
```

The directive will create cross-reference targets for the given options, referenceable by `option` (in the example case, you'd use something like ``:option:`dest_dir` ``, ``:option:`-m` ``, or ``:option:`--module` ``).

`cmdoption` directive is a deprecated alias for the `option` directive.

.. envvar:: `name`

Describes an environment variable that the documented code or program uses or defines. Referenceable by `envvar`.

.. program:: `name`

Like `py:currentmodule`, this directive produces no output. Instead, it serves to notify Sphinx that all following `option` directives document options for the program called *name*.

If you use `program`, you have to qualify the references in your `option` roles by the program name, so if you have the following situation

```
.. program:: rm

.. option:: -r

   Work recursively.

.. program:: svn

.. option:: -r revision

   Specify the revision to work upon.
```

then ``:option:`rm -r` `` would refer to the first option, while ``:option:`svn -r` `` would refer to the second one.

The program name may contain spaces (in case you want to document subcommands like `svn add` and `svn commit` separately).

New in version 0.5.

There is also a very generic object description directive, which is not tied to any domain:

.. describe:: `text`
.. object:: `text`

This directive produces the same formatting as the specific ones provided by domains, but does not create index entries or cross-referencing targets. Example:

```
.. describe:: PAPER

   You can set this variable to select a paper size.
```

The JavaScript Domain

The JavaScript domain (name **js**) provides the following directives:

.. js:module:: `name`

This directive sets the module name for object declarations that follow after. The module name is used in the global module index and in cross references. This directive does not create an object heading like *py:class* would, for example.

By default, this directive will create a linkable entity and will cause an entry in the global module index, unless the `noindex` option is specified. If this option is specified, the directive will only update the current module name.

New in version 1.6.

.. js:function:: `name(signature)`

Describes a JavaScript function or method. If you want to describe arguments as optional use square brackets as *documented* for Python signatures.

You can use fields to give more details about arguments and their expected types, errors which may be thrown by the function, and the value being returned:

```
.. js:function:: $.getJSON(href, callback[, errback])

   :param string href: An URI to the location of the resource.
   :param callback: Gets called with the object.
   :param errback:
       Gets called in case the request fails. And a lot of other
       text so we need multiple lines.
   :throws SomeError: For whatever reason in that case.
   :returns: Something.
```

This is rendered as:

`$.`**`getJSON`**`(`*href*, *callback*[, *errback*]`)`

Arguments
- **`href`** (*`string`*) – An URI to the location of the resource.
- **`callback`** – Gets called with the object.
- **`errback`** – Gets called in case the request fails. And a lot of other text so we need multiple lines.

Throws **`SomeError`** – For whatever reason in that case.

Returns Something.

.. js:method:: `name(signature)`

This directive is an alias for *js:function*, however it describes a function that is implemented as a method on a class object.

New in version 1.6.

.. js:class:: name

Describes a constructor that creates an object. This is basically like a function but will show up with a *class* prefix:

```
.. js:class:: MyAnimal(name[, age])

   :param string name: The name of the animal
   :param number age: an optional age for the animal
```

This is rendered as:

> `class` **`MyAnimal`**(*name*[, *age*])
>
> **Arguments**
> - **name** (*string*) – The name of the animal
> - **age** (*number*) – an optional age for the animal

.. js:data:: name

Describes a global variable or constant.

.. js:attribute:: object.name

Describes the attribute *name* of *object*.

These roles are provided to refer to the described objects:

:js:mod:
:js:func:
:js:meth:
:js:class:
:js:data:
:js:attr:

The reStructuredText domain

The reStructuredText domain (name **rst**) provides the following directives:

.. rst:directive:: name

Describes a reST directive. The *name* can be a single directive name or actual directive syntax (*..* prefix and *::* suffix) with arguments that will be rendered differently. For example:

```
.. rst:directive:: foo

   Foo description.

.. rst:directive:: .. bar:: baz

   Bar description.
```

will be rendered as:

> **.. foo::**
> Foo description.
>
> **.. bar::** baz
> Bar description.

.. rst:directive:option:: name

Describes an option for reST directive. The *name* can be a single option name or option name with arguments which separated with colon (`:`). For example:

```
.. rst:directive:: toctree

   .. rst:directive:option:: caption: caption of ToC

   .. rst:directive:option:: glob
```

will be rendered as:

.. toctree::

:caption: caption of ToC

:glob:

options

:type: description of argument (text)

Describe the type of option value.

For example:

```
.. rst:directive:: toctree

   .. rst:directive:option:: maxdepth
      :type: integer or no value
```

New in version 2.1.

.. rst:role:: name

Describes a reST role. For example:

```
.. rst:role:: foo

   Foo description.
```

will be rendered as:

:foo:

Foo description.

These roles are provided to refer to the described objects:

:rst:dir:
:rst:role:

The Math Domain

The math domain (name **math**) provides the following roles:

:math:numref:

Role for cross-referencing equations defined by *math* directive via their label. Example:

```
.. math:: e^{i\pi} + 1 = 0
   :label: euler

Euler's identity, equation :math:numref:`euler`, was elected one of the
most beautiful mathematical formulas.
```

New in version 1.8.

More domains

The sphinx-contrib[126] repository contains more domains available as extensions; currently Ada[127], CoffeeScript[128], Erlang[129], HTTP[130], Lasso[131], MATLAB[132], PHP[133], and Ruby[134] domains. Also available are domains for Chapel[135], Common Lisp[136], dqn[137], Go[138], Jinja[139], Operation[140], and Scala[141].

1.4 Markdown

Markdown[142] is a lightweight markup language with a simplistic plain text formatting syntax. It exists in many syntactically different *flavors*. To support Markdown-based documentation, Sphinx can use recommonmark[143]. recommonmark is a Docutils bridge to CommonMark-py[144], a Python package for parsing the CommonMark[145] Markdown flavor.

Configuration

To configure your Sphinx project for Markdown support, proceed as follows:

1. Install the Markdown parser *recommonmark*:

```
pip install --upgrade recommonmark
```

Note: The configuration as explained here requires recommonmark version 0.5.0 or later.

2. Add *recommonmark* to the `list of configured extensions`:

```
extensions = ['recommonmark']
```

Changed in version 1.8: Version 1.8 deprecates and version 3.0 removes the `source_parsers` configuration variable that was used by older *recommonmark* versions.

3. If you want to use Markdown files with extensions other than `.md`, adjust the `source_suffix` variable. The following example configures Sphinx to parse all files with the extensions `.md` and `.txt` as Markdown:

[126] https://github.com/sphinx-contrib
[127] https://pypi.org/project/sphinxcontrib-adadomain/
[128] https://pypi.org/project/sphinxcontrib-coffee/
[129] https://pypi.org/project/sphinxcontrib-erlangdomain/
[130] https://pypi.org/project/sphinxcontrib-httpdomain/
[131] https://pypi.org/project/sphinxcontrib-lassodomain/
[132] https://pypi.org/project/sphinxcontrib-matlabdomain/
[133] https://pypi.org/project/sphinxcontrib-phpdomain/
[134] https://bitbucket.org/birkenfeld/sphinx-contrib/src/default/rubydomain
[135] https://pypi.org/project/sphinxcontrib-chapeldomain/
[136] https://pypi.org/project/sphinxcontrib-cldomain/
[137] https://pypi.org/project/sphinxcontrib-dqndomain/
[138] https://pypi.org/project/sphinxcontrib-golangdomain/
[139] https://pypi.org/project/sphinxcontrib-jinjadomain/
[140] https://pypi.org/project/sphinxcontrib-operationdomain/
[141] https://pypi.org/project/sphinxcontrib-scaladomain/
[142] https://daringfireball.net/projects/markdown/
[143] https://recommonmark.readthedocs.io/en/latest/index.html
[144] https://github.com/rtfd/CommonMark-py
[145] https://commonmark.org/

```
source_suffix = {
    '.rst': 'restructuredtext',
    '.txt': 'markdown',
    '.md': 'markdown',
}
```

4. You can further configure *recommonmark* to allow custom syntax that standard *CommonMark* doesn't support. Read more in the recommonmark documentation[146].

1.5 Configuration

The *configuration directory* must contain a file named `conf.py`. This file (containing Python code) is called the "build configuration file" and contains (almost) all configuration needed to customize Sphinx input and output behavior.

> An optional file docutils.conf[147] can be added to the configuration directory to adjust Docutils[148] configuration if not otherwise overridden or set by Sphinx.

The configuration file is executed as Python code at build time (using `execfile()`, and with the current directory set to its containing directory), and therefore can execute arbitrarily complex code. Sphinx then reads simple names from the file's namespace as its configuration.

Important points to note:

- If not otherwise documented, values must be strings, and their default is the empty string.
- The term "fully-qualified name" refers to a string that names an importable Python object inside a module; for example, the FQN `"sphinx.builders.Builder"` means the `Builder` class in the `sphinx.builders` module.
- Remember that document names use `/` as the path separator and don't contain the file name extension.
- Since `conf.py` is read as a Python file, the usual rules apply for encodings and Unicode support.
- The contents of the config namespace are pickled (so that Sphinx can find out when configuration changes), so it may not contain unpickleable values – delete them from the namespace with `del` if appropriate. Modules are removed automatically, so you don't need to `del` your imports after use.
- There is a special object named `tags` available in the config file. It can be used to query and change the tags (see *Including content based on tags*). Use `tags.has('tag')` to query, `tags.add('tag')` and `tags.remove('tag')` to change. Only tags set via the `-t` command-line option or via `tags.add('tag')` can be queried using `tags.has('tag')`. Note that the current builder tag is not available in `conf.py`, as it is created *after* the builder is initialized.

Project information

`project`
The documented project's name.

`author`
The author name(s) of the document. The default value is `'unknown'`.

`copyright`
A copyright statement in the style `'2008, Author Name'`.

[146] https://recommonmark.readthedocs.io/en/latest/auto_structify.html
[147] http://docutils.sourceforge.net/docs/user/config.html
[148] http://docutils.sourceforge.net/

project_copyright
An alias of *copyright*.

New in version 3.5.

version
The major project version, used as the replacement for `|version|`. For example, for the Python documentation, this may be something like `2.6`.

release
The full project version, used as the replacement for `|release|` and e.g. in the HTML templates. For example, for the Python documentation, this may be something like `2.6.0rc1`.

If you don't need the separation provided between *version* and *release*, just set them both to the same value.

General configuration

extensions
A list of strings that are module names of *extensions*. These can be extensions coming with Sphinx (named `sphinx.ext.*`) or custom ones.

Note that you can extend `sys.path`[149] within the conf file if your extensions live in another directory – but make sure you use absolute paths. If your extension path is relative to the *configuration directory*, use `os.path.abspath()`[150] like so:

```
import sys, os

sys.path.append(os.path.abspath('sphinxext'))

extensions = ['extname']
```

That way, you can load an extension called `extname` from the subdirectory `sphinxext`.

The configuration file itself can be an extension; for that, you only need to provide a `setup()` function in it.

source_suffix
The file extensions of source files. Sphinx considers the files with this suffix as sources. The value can be a dictionary mapping file extensions to file types. For example:

```
source_suffix = {
    '.rst': 'restructuredtext',
    '.txt': 'restructuredtext',
    '.md': 'markdown',
}
```

By default, Sphinx only supports `'restructuredtext'` file type. You can add a new file type using source parser extensions. Please read a document of the extension to know which file type the extension supports.

The value may also be a list of file extensions: then Sphinx will consider that they all map to the `'restructuredtext'` file type.

Default is `{'.rst': 'restructuredtext'}`.

Note: file extensions have to start with a dot (e.g. `.rst`).

Changed in version 1.3: Can now be a list of extensions.

[149] https://docs.python.org/3/library/sys.html#sys.path
[150] https://docs.python.org/3/library/os.path.html#os.path.abspath

Changed in version 1.8: Support file type mapping

source_encoding

The encoding of all reST source files. The recommended encoding, and the default value, is 'utf-8-sig'.

New in version 0.5: Previously, Sphinx accepted only UTF-8 encoded sources.

source_parsers

If given, a dictionary of parser classes for different source suffices. The keys are the suffix, the values can be either a class or a string giving a fully-qualified name of a parser class. The parser class can be either docutils.parsers.Parser or *sphinx.parsers.Parser*. Files with a suffix that is not in the dictionary will be parsed with the default reStructuredText parser.

For example:

```
source_parsers = {'.md': 'recommonmark.parser.CommonMarkParser'}
```

Note: Refer to *Markdown* for more information on using Markdown with Sphinx.

New in version 1.3.

Deprecated since version 1.8: Now Sphinx provides an API *Sphinx.add_source_parser()* to register a source parser. Please use it instead.

master_doc

The document name of the "master" document, that is, the document that contains the root *toctree* directive. Default is 'index'.

Changed in version 2.0: The default is changed to 'index' from 'contents'.

exclude_patterns

A list of glob-style patterns that should be excluded when looking for source files.[1] They are matched against the source file names relative to the source directory, using slashes as directory separators on all platforms.

Example patterns:

- 'library/xml.rst' – ignores the library/xml.rst file (replaces entry in unused_docs)
- 'library/xml' – ignores the library/xml directory
- 'library/xml*' – ignores all files and directories starting with library/xml
- '**/.svn' – ignores all .svn directories

exclude_patterns is also consulted when looking for static files in *html_static_path* and *html_extra_path*.

New in version 1.0.

templates_path

A list of paths that contain extra templates (or templates that overwrite builtin/theme-specific templates). Relative paths are taken as relative to the configuration directory.

Changed in version 1.3: As these files are not meant to be built, they are automatically added to *exclude_patterns*.

template_bridge

A string with the fully-qualified name of a callable (or simply a class) that returns an instance of *TemplateBridge*. This instance is then used to render HTML documents, and possibly the output of other builders (currently the changes builder). (Note that the template bridge must be made theme-aware if HTML themes are to be used.)

[1] A note on available globbing syntax: you can use the standard shell constructs *, ?, [...] and [!...] with the feature that these all don't match slashes. A double star ** can be used to match any sequence of characters *including* slashes.

rst_epilog

A string of reStructuredText that will be included at the end of every source file that is read. This is a possible place to add substitutions that should be available in every file (another being *rst_prolog*). An example:

```
rst_epilog = """
.. |psf| replace:: Python Software Foundation
"""
```

New in version 0.6.

rst_prolog

A string of reStructuredText that will be included at the beginning of every source file that is read. This is a possible place to add substitutions that should be available in every file (another being *rst_epilog*). An example:

```
rst_prolog = """
.. |psf| replace:: Python Software Foundation
"""
```

New in version 1.0.

primary_domain

The name of the default *domain*. Can also be `None` to disable a default domain. The default is `'py'`. Those objects in other domains (whether the domain name is given explicitly, or selected by a *default-domain* directive) will have the domain name explicitly prepended when named (e.g., when the default domain is C, Python functions will be named "Python function", not just "function").

New in version 1.0.

default_role

The name of a reST role (builtin or Sphinx extension) to use as the default role, that is, for text marked up `` `like this` ``. This can be set to `'py:obj'` to make `` `filter` `` a cross-reference to the Python function "filter". The default is `None`, which doesn't reassign the default role.

The default role can always be set within individual documents using the standard reST `default-role` directive.

New in version 0.4.

keep_warnings

If true, keep warnings as "system message" paragraphs in the built documents. Regardless of this setting, warnings are always written to the standard error stream when `sphinx-build` is run.

The default is `False`, the pre-0.5 behavior was to always keep them.

New in version 0.5.

suppress_warnings

A list of warning types to suppress arbitrary warning messages.

Sphinx supports following warning types:

- `app.add_node`
- `app.add_directive`
- `app.add_role`
- `app.add_generic_role`
- `app.add_source_parser`
- `download.not_readable`
- `image.not_readable`
- `ref.term`

- `ref.ref`
- `ref.numref`
- `ref.keyword`
- `ref.option`
- `ref.citation`
- `ref.footnote`
- `ref.doc`
- `ref.python`
- `misc.highlighting_failure`
- `toc.circular`
- `toc.secnum`
- `epub.unknown_project_files`
- `epub.duplicated_toc_entry`
- `autosectionlabel.*`

You can choose from these types.

Now, this option should be considered *experimental*.

New in version 1.4.

Changed in version 1.5: Added `misc.highlighting_failure`

Changed in version 1.5.1: Added `epub.unknown_project_files`

Changed in version 1.6: Added `ref.footnote`

Changed in version 2.1: Added `autosectionlabel.*`

Changed in version 3.3.0: Added `epub.duplicated_toc_entry`

`needs_sphinx`

If set to a `major.minor` version string like `'1.1'`, Sphinx will compare it with its version and refuse to build if it is too old. Default is no requirement.

New in version 1.0.

Changed in version 1.4: also accepts micro version string

`needs_extensions`

This value can be a dictionary specifying version requirements for extensions in *`extensions`*, e.g. `needs_extensions = {'sphinxcontrib.something': '1.5'}`. The version strings should be in the form `major.minor`. Requirements do not have to be specified for all extensions, only for those you want to check.

This requires that the extension specifies its version to Sphinx (see *Developing extensions for Sphinx* for how to do that).

New in version 1.3.

`manpages_url`

A URL to cross-reference *`manpage`* directives. If this is defined to `https://manpages.debian.org/{path}`, the ``:manpage:`man(1)` `` role will link to <https://manpages.debian.org/man(1)>. The patterns available are:

- `page` - the manual page (`man`)
- `section` - the manual section (`1`)
- `path` - the original manual page and section specified (`man(1)`)

This also supports manpages specified as `man.1`.

Note: This currently affects only HTML writers but could be expanded in the future.

New in version 1.7.

`nitpicky`
If true, Sphinx will warn about *all* references where the target cannot be found. Default is `False`. You can activate this mode temporarily using the *`-n`* command-line switch.

New in version 1.0.

`nitpick_ignore`
A list of `(type, target)` tuples (by default empty) that should be ignored when generating warnings in "nitpicky mode". Note that `type` should include the domain name if present. Example entries would be `('py:func', 'int')` or `('envvar', 'LD_LIBRARY_PATH')`.

New in version 1.1.

`numfig`
If true, figures, tables and code-blocks are automatically numbered if they have a caption. The *`numref`* role is enabled. Obeyed so far only by HTML and LaTeX builders. Default is `False`.

Note: The LaTeX builder always assigns numbers whether this option is enabled or not.

New in version 1.3.

`numfig_format`
A dictionary mapping `'figure'`, `'table'`, `'code-block'` and `'section'` to strings that are used for format of figure numbers. As a special character, `%s` will be replaced to figure number.

Default is to use `'Fig. %s'` for `'figure'`, `'Table %s'` for `'table'`, `'Listing %s'` for `'code-block'` and `'Section'` for `'section'`.

New in version 1.3.

`numfig_secnum_depth`

- if set to `0`, figures, tables and code-blocks are continuously numbered starting at `1`.
- if `1` (default) numbers will be `x.1`, `x.2`, … with `x` the section number (top level sectioning; no `x.` if no section). This naturally applies only if section numbering has been activated via the `:numbered:` option of the *`toctree`* directive.
- `2` means that numbers will be `x.y.1`, `x.y.2`, … if located in a sub-section (but still `x.1`, `x.2`, … if located directly under a section and `1`, `2`, … if not in any top level section.)
- etc…

New in version 1.3.

Changed in version 1.7: The LaTeX builder obeys this setting (if *`numfig`* is set to `True`).

`smartquotes`
If true, the Docutils Smart Quotes transform[151], originally based on SmartyPants[152] (limited to English) and currently applying to many languages, will be used to convert quotes and dashes to typographically correct entities. Default: `True`.

New in version 1.6.6: It replaces deprecated *`html_use_smartypants`*. It applies by default to all builders except `man` and `text` (see *`smartquotes_excludes`*.)

A docutils.conf[153] file located in the configuration directory (or a global `~/.docutils` file) is obeyed unconditionally if it *deactivates* smart quotes via the corresponding Docutils option[154]. But if it *activates* them, then *smartquotes* does prevail.

smartquotes_action

This string customizes the Smart Quotes transform. See the file `smartquotes.py` at the Docutils repository[155] for details. The default `'qDe'` educates normal **q**uote characters `"`, `'`, em- and en-**D**ashes `---`, `--`, and **e**llipses `...`.

New in version 1.6.6.

smartquotes_excludes

This is a `dict` whose default is:

```
{'languages': ['ja'], 'builders': ['man', 'text']}
```

Each entry gives a sufficient condition to ignore the *smartquotes* setting and deactivate the Smart Quotes transform. Accepted keys are as above `'builders'` or `'languages'`. The values are lists.

Note: Currently, in case of invocation of **make** with multiple targets, the first target name is the only one which is tested against the `'builders'` entry and it decides for all. Also, a `make text` following `make html` needs to be issued in the form `make text O="-E"` to force re-parsing of source files, as the cached ones are already transformed. On the other hand the issue does not arise with direct usage of **sphinx-build** as it caches (in its default usage) the parsed source files in per builder locations.

Hint: An alternative way to effectively deactivate (or customize) the smart quotes for a given builder, for example `latex`, is to use `make` this way:

```
make latex O="-D smartquotes_action="
```

This can follow some `make html` with no problem, in contrast to the situation from the prior note. It requires Docutils 0.14 or later.

New in version 1.6.6.

user_agent

A User-Agent of Sphinx. It is used for a header on HTTP access (ex. linkcheck, intersphinx and so on). Default is `"Sphinx/X.Y.Z requests/X.Y.Z python/X.Y.Z"`.

New in version 2.3.

tls_verify

If true, Sphinx verifies server certifications. Default is `True`.

New in version 1.5.

tls_cacerts

A path to a certification file of CA or a path to directory which contains the certificates. This also allows a dictionary mapping hostname to the path to certificate file. The certificates are used to verify server certifications.

New in version 1.5.

[151] http://docutils.sourceforge.net/docs/user/smartquotes.html
[152] https://daringfireball.net/projects/smartypants/
[153] http://docutils.sourceforge.net/docs/user/config.html
[154] http://docutils.sourceforge.net/docs/user/config.html#smart-quotes
[155] https://sourceforge.net/p/docutils/code/HEAD/tree/trunk/docutils/

Tip: Sphinx uses requests[156] as a HTTP library internally. Therefore, Sphinx refers a certification file on the directory pointed `REQUESTS_CA_BUNDLE` environment variable if `tls_cacerts` not set.

`today`
`today_fmt`

These values determine how to format the current date, used as the replacement for `|today|`.

- If you set *`today`* to a non-empty value, it is used.
- Otherwise, the current time is formatted using `time.strftime()`[157] and the format given in *`today_fmt`*.

The default is now *`today`* and a *`today_fmt`* of `'%b %d, %Y'` (or, if translation is enabled with *`language`*, an equivalent format for the selected locale).

`highlight_language`

The default language to highlight source code in. The default is `'python3'`. The value should be a valid Pygments lexer name, see *Showing code examples* for more details.

New in version 0.5.

Changed in version 1.4: The default is now `'default'`. It is similar to `'python3'`; it is mostly a superset of `'python'` but it fallbacks to `'none'` without warning if failed. `'python3'` and other languages will emit warning if failed. If you prefer Python 2 only highlighting, you can set it back to `'python'`.

`highlight_options`

A dictionary that maps language names to options for the lexer modules of Pygments. These are lexer-specific; for the options understood by each, see the Pygments documentation[158].

Example:

```
highlight_options = {
  'default': {'stripall': True},
  'php': {'startinline': True},
}
```

A single dictionary of options are also allowed. Then it is recognized as options to the lexer specified by *`highlight_language`*:

```
# configuration for the ``highlight_language``
highlight_options = {'stripall': True}
```

New in version 1.3.

Changed in version 3.5: Allow to configure highlight options for multiple languages

`pygments_style`

The style name to use for Pygments highlighting of source code. If not set, either the theme's default style or `'sphinx'` is selected for HTML output.

Changed in version 0.3: If the value is a fully-qualified name of a custom Pygments style class, this is then used as custom style.

`add_function_parentheses`

A boolean that decides whether parentheses are appended to function and method role text (e.g. the content of `:func:`input``) to signify that the name is callable. Default is `True`.

[156] https://requests.readthedocs.io/en/master/
[157] https://docs.python.org/3/library/time.html#time.strftime
[158] https://pygments.org/docs/lexers

`add_module_names`
A boolean that decides whether module names are prepended to all *object* names (for object types where a "module" of some kind is defined), e.g. for `py:function` directives. Default is `True`.

`show_authors`
A boolean that decides whether `codeauthor` and `sectionauthor` directives produce any output in the built files.

`modindex_common_prefix`
A list of prefixes that are ignored for sorting the Python module index (e.g., if this is set to `['foo.']`, then `foo.bar` is shown under `B`, not `F`). This can be handy if you document a project that consists of a single package. Works only for the HTML builder currently. Default is `[]`.

New in version 0.6.

`trim_footnote_reference_space`
Trim spaces before footnote references that are necessary for the reST parser to recognize the footnote, but do not look too nice in the output.

New in version 0.6.

`trim_doctest_flags`
If true, doctest flags (comments looking like `# doctest: FLAG, ...`) at the ends of lines and `<BLANKLINE>` markers are removed for all code blocks showing interactive Python sessions (i.e. doctests). Default is `True`. See the extension `doctest` for more possibilities of including doctests.

New in version 1.0.

Changed in version 1.1: Now also removes `<BLANKLINE>`.

`strip_signature_backslash`
Default is `False`. When backslash stripping is enabled then every occurrence of `\\` in a domain directive will be changed to `\`, even within string literals. This was the behaviour before version 3.0, and setting this variable to `True` will reinstate that behaviour.

New in version 3.0.

Options for internationalization

These options influence Sphinx's *Native Language Support*. See the documentation on *Internationalization* for details.

`language`
The code for the language the docs are written in. Any text automatically generated by Sphinx will be in that language. Also, Sphinx will try to substitute individual paragraphs from your documents with the translation sets obtained from `locale_dirs`. Sphinx will search language-specific figures named by `figure_language_filename` (e.g. the German version of `myfigure.png` will be `myfigure.de.png` by default setting) and substitute them for original figures. In the LaTeX builder, a suitable language will be selected as an option for the *Babel* package. Default is `None`, which means that no translation will be done.

New in version 0.5.

Changed in version 1.4: Support figure substitution

Currently supported languages by Sphinx are:

- `ar` – Arabic
- `bg` – Bulgarian
- `bn` – Bengali
- `ca` – Catalan

- `cak` – Kaqchikel
- `cs` – Czech
- `cy` – Welsh
- `da` – Danish
- `de` – German
- `el` – Greek
- `en` – English
- `eo` – Esperanto
- `es` – Spanish
- `et` – Estonian
- `eu` – Basque
- `fa` – Iranian
- `fi` – Finnish
- `fr` – French
- `he` – Hebrew
- `hi` – Hindi
- `hi_IN` – Hindi (India)
- `hr` – Croatian
- `hu` – Hungarian
- `id` – Indonesian
- `it` – Italian
- `ja` – Japanese
- `ko` – Korean
- `lt` – Lithuanian
- `lv` – Latvian
- `mk` – Macedonian
- `nb_NO` – Norwegian Bokmal
- `ne` – Nepali
- `nl` – Dutch
- `pl` – Polish
- `pt` – Portuguese
- `pt_BR` – Brazilian Portuguese
- `pt_PT` – European Portuguese
- `ro` – Romanian
- `ru` – Russian
- `si` – Sinhala
- `sk` – Slovak
- `sl` – Slovenian

- `sq` – Albanian
- `sr` – Serbian
- `sr@latin` – Serbian (Latin)
- `sr_RS` – Serbian (Cyrillic)
- `sv` – Swedish
- `ta` – Tamil
- `te` – Telugu
- `tr` – Turkish
- `uk_UA` – Ukrainian
- `ur` – Urdu
- `vi` – Vietnamese
- `zh_CN` – Simplified Chinese
- `zh_TW` – Traditional Chinese

`locale_dirs`

New in version 0.5.

Directories in which to search for additional message catalogs (see *`language`*), relative to the source directory. The directories on this path are searched by the standard `gettext`[159] module.

Internal messages are fetched from a text domain of `sphinx`; so if you add the directory `./locale` to this setting, the message catalogs (compiled from `.po` format using **`msgfmt`**) must be in `./locale/`*`language`*`/LC_MESSAGES/sphinx.mo`. The text domain of individual documents depends on *`gettext_compact`*.

The default is `['locales']`.

Changed in version 1.5: Use `locales` directory as a default value

`gettext_compact`

New in version 1.1.

If true, a document's text domain is its docname if it is a top-level project file and its very base directory otherwise.

If set to string, all document's text domain is this string, making all documents use single text domain.

By default, the document `markup/code.rst` ends up in the `markup` text domain. With this option set to `False`, it is `markup/code`.

Changed in version 3.3: The string value is now accepted.

`gettext_uuid`

If true, Sphinx generates uuid information for version tracking in message catalogs. It is used for:

- Add uid line for each msgids in .pot files.
- Calculate similarity between new msgids and previously saved old msgids. This calculation takes a long time.

If you want to accelerate the calculation, you can use `python-levenshtein` 3rd-party package written in C by using **`pip install python-levenshtein`**.

The default is `False`.

New in version 1.3.

[159] https://docs.python.org/3/library/gettext.html#module-gettext

`gettext_location`
If true, Sphinx generates location information for messages in message catalogs.

The default is `True`.

New in version 1.3.

`gettext_auto_build`
If true, Sphinx builds mo file for each translation catalog files.

The default is `True`.

New in version 1.3.

`gettext_additional_targets`
To specify names to enable gettext extracting and translation applying for i18n additionally. You can specify below names:

Index index terms

Literal-block literal blocks (`::` annotation and `code-block` directive)

Doctest-block doctest block

Raw raw content

Image image/figure uri

For example: `gettext_additional_targets = ['literal-block', 'image']`.

The default is `[]`.

New in version 1.3.

Changed in version 4.0: The alt text for image is translated by default.

`figure_language_filename`
The filename format for language-specific figures. The default value is `{root}.{language}{ext}`. It will be expanded to `dirname/filename.en.png` from `.. image:: dirname/filename.png`. The available format tokens are:

- `{root}` - the filename, including any path component, without the file extension, e.g. `dirname/filename`
- `{path}` - the directory path component of the filename, with a trailing slash if non-empty, e.g. `dirname/`
- `{docpath}` - the directory path component for the current document, with a trailing slash if non-empty.
- `{basename}` - the filename without the directory path or file extension components, e.g. `filename`
- `{ext}` - the file extension, e.g. `.png`
- `{language}` - the translation language, e.g. `en`

For example, setting this to `{path}{language}/{basename}{ext}` will expand to `dirname/en/filename.png` instead.

New in version 1.4.

Changed in version 1.5: Added `{path}` and `{basename}` tokens.

Changed in version 3.2: Added `{docpath}` token.

Options for Math

These options influence Math notations.

`math_number_all`
Set this option to `True` if you want all displayed math to be numbered. The default is `False`.

`math_eqref_format`
A string used for formatting the labels of references to equations. The `{number}` place-holder stands for the equation number.

Example: `'Eq.{number}'` gets rendered as, for example, `Eq.10`.

`math_numfig`
If `True`, displayed math equations are numbered across pages when *`numfig`* is enabled. The *`numfig_secnum_depth`* setting is respected. The *`eq`*, not *`numref`*, role must be used to reference equation numbers. Default is `True`.

New in version 1.7.

Options for HTML output

These options influence HTML as well as HTML Help output, and other builders that use Sphinx's HTMLWriter class.

`html_theme`
The "theme" that the HTML output should use. See the *section about theming*. The default is `'alabaster'`.

New in version 0.6.

`html_theme_options`
A dictionary of options that influence the look and feel of the selected theme. These are theme-specific. For the options understood by the builtin themes, see *this section*.

New in version 0.6.

`html_theme_path`
A list of paths that contain custom themes, either as subdirectories or as zip files. Relative paths are taken as relative to the configuration directory.

New in version 0.6.

`html_style`
The style sheet to use for HTML pages. A file of that name must exist either in Sphinx's `static/` path, or in one of the custom paths given in *`html_static_path`*. Default is the stylesheet given by the selected theme. If you only want to add or override a few things compared to the theme's stylesheet, use CSS `@import` to import the theme's stylesheet.

`html_title`
The "title" for HTML documentation generated with Sphinx's own templates. This is appended to the `<title>` tag of individual pages, and used in the navigation bar as the "topmost" element. It defaults to `'<project> v<revision> documentation'`.

`html_short_title`
A shorter "title" for the HTML docs. This is used for links in the header and in the HTML Help docs. If not given, it defaults to the value of *`html_title`*.

New in version 0.4.

`html_baseurl`
The base URL which points to the root of the HTML documentation. It is used to indicate the location of document using The Canonical Link Relation[160]. Default: `''`.

New in version 1.8.

html_codeblock_linenos_style
The style of line numbers for code-blocks.

- `'table'` – display line numbers using `<table>` tag
- `'inline'` – display line numbers using `<span>` tag (default)

New in version 3.2.

Changed in version 4.0: It defaults to `'inline'`.

Deprecated since version 4.0.

html_context
A dictionary of values to pass into the template engine's context for all pages. Single values can also be put in this dictionary using the *-A* command-line option of `sphinx-build`.

New in version 0.5.

html_logo
If given, this must be the name of an image file (path relative to the *configuration directory*) that is the logo of the docs, or URL that points an image file for the logo. It is placed at the top of the sidebar; its width should therefore not exceed 200 pixels. Default: `None`.

New in version 0.4.1: The image file will be copied to the `_static` directory of the output HTML, but only if the file does not already exist there.

Changed in version 4.0: Also accepts the URL for the logo file.

html_favicon
If given, this must be the name of an image file (path relative to the *configuration directory*) that is the favicon of the docs, or URL that points an image file for the favicon. Modern browsers use this as the icon for tabs, windows and bookmarks. It should be a Windows-style icon file (`.ico`), which is 16x16 or 32x32 pixels large. Default: `None`.

New in version 0.4: The image file will be copied to the `_static` directory of the output HTML, but only if the file does not already exist there.

Changed in version 4.0: Also accepts the URL for the favicon.

html_css_files
A list of CSS files. The entry must be a *filename* string or a tuple containing the *filename* string and the *attributes* dictionary. The *filename* must be relative to the *html_static_path*, or a full URI with scheme like `http://example.org/style.css`. The *attributes* is used for attributes of `<link>` tag. It defaults to an empty list.

Example:

```
html_css_files = ['custom.css',
                  'https://example.com/css/custom.css',
                  ('print.css', {'media': 'print'})]
```

As a special attribute, *priority* can be set as an integer to load the CSS file earlier or lazier step. For more information, refer `Sphinx.add_css_files()`.

New in version 1.8.

Changed in version 3.5: Support priority attribute

html_js_files
A list of JavaScript *filename*. The entry must be a *filename* string or a tuple containing the *filename* string and the *attributes* dictionary. The *filename* must be relative to the *html_static_path*, or a full URI with scheme

[160] https://tools.ietf.org/html/rfc6596

like `http://example.org/script.js`. The *attributes* is used for attributes of `<script>` tag. It defaults to an empty list.

Example:

```
html_js_files = ['script.js',
                 'https://example.com/scripts/custom.js',
                 ('custom.js', {'async': 'async'})]
```

As a special attribute, *priority* can be set as an integer to load the CSS file earlier or lazier step. For more information, refer `Sphinx.add_css_files()`.

New in version 1.8.

Changed in version 3.5: Support priority attribute

`html_static_path`

A list of paths that contain custom static files (such as style sheets or script files). Relative paths are taken as relative to the configuration directory. They are copied to the output's `_static` directory after the theme's static files, so a file named `default.css` will overwrite the theme's `default.css`.

As these files are not meant to be built, they are automatically excluded from source files.

Note: For security reasons, dotfiles under `html_static_path` will not be copied. If you would like to copy them intentionally, please add each filepath to this setting:

```
html_static_path = ['_static', '_static/.htaccess']
```

Another way to do that, you can also use *`html_extra_path`*. It allows to copy dotfiles under the directories.

Changed in version 0.4: The paths in *`html_static_path`* can now contain subdirectories.

Changed in version 1.0: The entries in *`html_static_path`* can now be single files.

Changed in version 1.8: The files under *`html_static_path`* are excluded from source files.

`html_extra_path`

A list of paths that contain extra files not directly related to the documentation, such as `robots.txt` or `.htaccess`. Relative paths are taken as relative to the configuration directory. They are copied to the output directory. They will overwrite any existing file of the same name.

As these files are not meant to be built, they are automatically excluded from source files.

New in version 1.2.

Changed in version 1.4: The dotfiles in the extra directory will be copied to the output directory. And it refers *`exclude_patterns`* on copying extra files and directories, and ignores if path matches to patterns.

`html_last_updated_fmt`

If this is not None, a 'Last updated on:' timestamp is inserted at every page bottom, using the given `strftime()` format. The empty string is equivalent to `'%b %d, %Y'` (or a locale-dependent equivalent).

`html_use_smartypants`

If true, quotes and dashes are converted to typographically correct entities. Default: `True`.

Deprecated since version 1.6: To disable smart quotes, use rather *`smartquotes`*.

`html_add_permalinks`

Sphinx will add "permalinks" for each heading and description environment as paragraph signs that become visible when the mouse hovers over them.

This value determines the text for the permalink; it defaults to `"¶"`. Set it to `None` or the empty string to disable permalinks.

New in version 0.6: Previously, this was always activated.

Changed in version 1.1: This can now be a string to select the actual text of the link. Previously, only boolean values were accepted.

Deprecated since version 3.5: This has been replaced by *html_permalinks*

html_permalinks
If true, Sphinx will add "permalinks" for each heading and description environment. Default: `True`.

New in version 3.5.

html_permalinks_icon
A text for permalinks for each heading and description environment. HTML tags are allowed. Default: a paragraph sign; ¶

New in version 3.5.

html_sidebars
Custom sidebar templates, must be a dictionary that maps document names to template names.

The keys can contain glob-style patterns[Page 75, 1], in which case all matching documents will get the specified sidebars. (A warning is emitted when a more than one glob-style pattern matches for any document.)

The values can be either lists or single strings.

- If a value is a list, it specifies the complete list of sidebar templates to include. If all or some of the default sidebars are to be included, they must be put into this list as well.

  The default sidebars (for documents that don't match any pattern) are defined by theme itself. Builtin themes are using these templates by default: `['localtoc.html', 'relations.html', 'sourcelink.html', 'searchbox.html']`.

- If a value is a single string, it specifies a custom sidebar to be added between the `'sourcelink.html'` and `'searchbox.html'` entries. This is for compatibility with Sphinx versions before 1.0.

Deprecated since version 1.7: a single string value for `html_sidebars` will be removed in 2.0

Builtin sidebar templates that can be rendered are:

- **localtoc.html** – a fine-grained table of contents of the current document
- **globaltoc.html** – a coarse-grained table of contents for the whole documentation set, collapsed
- **relations.html** – two links to the previous and next documents
- **sourcelink.html** – a link to the source of the current document, if enabled in *html_show_sourcelink*
- **searchbox.html** – the "quick search" box

Example:

```
html_sidebars = {
   '**': ['globaltoc.html', 'sourcelink.html', 'searchbox.html'],
   'using/windows': ['windowssidebar.html', 'searchbox.html'],
}
```

This will render the custom template `windowssidebar.html` and the quick search box within the sidebar of the given document, and render the default sidebars for all other pages (except that the local TOC is replaced by the global TOC).

New in version 1.0: The ability to use globbing keys and to specify multiple sidebars.

Note that this value only has no effect if the chosen theme does not possess a sidebar, like the builtin **scrolls** and **haiku** themes.

html_additional_pages

Additional templates that should be rendered to HTML pages, must be a dictionary that maps document names to template names.

Example:

```
html_additional_pages = {
    'download': 'customdownload.html',
}
```

This will render the template `customdownload.html` as the page `download.html`.

html_domain_indices

If true, generate domain-specific indices in addition to the general index. For e.g. the Python domain, this is the global module index. Default is `True`.

This value can be a bool or a list of index names that should be generated. To find out the index name for a specific index, look at the HTML file name. For example, the Python module index has the name `'py-modindex'`.

New in version 1.0.

html_use_index

If true, add an index to the HTML documents. Default is `True`.

New in version 0.4.

html_split_index

If true, the index is generated twice: once as a single page with all the entries, and once as one page per starting letter. Default is `False`.

New in version 0.4.

html_copy_source

If true, the reST sources are included in the HTML build as `_sources/`*`name`*. The default is `True`.

html_show_sourcelink

If true (and *`html_copy_source`* is true as well), links to the reST sources will be added to the sidebar. The default is `True`.

New in version 0.6.

html_sourcelink_suffix

Suffix to be appended to source links (see *`html_show_sourcelink`*), unless they have this suffix already. Default is `'.txt'`.

New in version 1.5.

html_use_opensearch

If nonempty, an OpenSearch[161] description file will be output, and all pages will contain a `<link>` tag referring to it. Since OpenSearch doesn't support relative URLs for its search page location, the value of this option must be the base URL from which these documents are served (without trailing slash), e.g. `"https://docs.python.org"`. The default is `''`.

html_file_suffix

This is the file name suffix for generated HTML files. The default is `".html"`.

New in version 0.4.

html_link_suffix

Suffix for generated links to HTML files. The default is whatever *`html_file_suffix`* is set to; it can be set differently (e.g. to support different web server setups).

New in version 0.6.

[161] http://www.opensearch.org/Home

`html_show_copyright`
If true, "(C) Copyright ..." is shown in the HTML footer. Default is `True`.

New in version 1.0.

`html_show_sphinx`
If true, "Created using Sphinx" is shown in the HTML footer. Default is `True`.

New in version 0.4.

`html_output_encoding`
Encoding of HTML output files. Default is `'utf-8'`. Note that this encoding name must both be a valid Python encoding name and a valid HTML `charset` value.

New in version 1.0.

`html_compact_lists`
If true, a list all whose items consist of a single paragraph and/or a sub-list all whose items etc... (recursive definition) will not use the `<p>` element for any of its items. This is standard docutils behavior. Default: `True`.

New in version 1.0.

`html_secnumber_suffix`
Suffix for section numbers. Default: `". "`. Set to `" "` to suppress the final dot on section numbers.

New in version 1.0.

`html_search_language`
Language to be used for generating the HTML full-text search index. This defaults to the global language selected with *`language`*. If there is no support for this language, `"en"` is used which selects the English language.

Support is present for these languages:

- `da` – Danish
- `nl` – Dutch
- `en` – English
- `fi` – Finnish
- `fr` – French
- `de` – German
- `hu` – Hungarian
- `it` – Italian
- `ja` – Japanese
- `no` – Norwegian
- `pt` – Portuguese
- `ro` – Romanian
- `ru` – Russian
- `es` – Spanish
- `sv` – Swedish
- `tr` – Turkish
- `zh` – Chinese

Accelerating build speed

Each language (except Japanese) provides its own stemming algorithm. Sphinx uses a Python implementation by default. You can use a C implementation to accelerate building the index file.

- PorterStemmer[162] (`en`)
- PyStemmer[163] (all languages)

New in version 1.1: With support for `en` and `ja`.

Changed in version 1.3: Added additional languages.

`html_search_options`

A dictionary with options for the search language support, empty by default. The meaning of these options depends on the language selected.

The English support has no options.

The Japanese support has these options:

Type type is dotted module path string to specify Splitter implementation which should be derived from `sphinx.search.ja.BaseSplitter`. If not specified or None is specified, `'sphinx.search.ja.DefaultSplitter'` will be used.

You can choose from these modules:

'sphinx.search.ja.DefaultSplitter' TinySegmenter algorithm. This is default splitter.

'sphinx.search.ja.MecabSplitter' MeCab binding. To use this splitter, 'mecab' python binding or dynamic link library ('libmecab.so' for linux, 'libmecab.dll' for windows) is required.

'sphinx.search.ja.JanomeSplitter' Janome binding. To use this splitter, Janome[164] is required.

Deprecated since version 1.6: `'mecab'`, `'janome'` and `'default'` is deprecated. To keep compatibility, `'mecab'`, `'janome'` and `'default'` are also acceptable.

Other option values depend on splitter value which you choose.

Options for `'mecab'`:

dic_enc dic_enc option is the encoding for the MeCab algorithm.

dict dict option is the dictionary to use for the MeCab algorithm.

lib lib option is the library name for finding the MeCab library via ctypes if the Python binding is not installed.

For example:

```
html_search_options = {
    'type': 'mecab',
    'dic_enc': 'utf-8',
    'dict': '/path/to/mecab.dic',
    'lib': '/path/to/libmecab.so',
}
```

Options for `'janome'`:

user_dic user_dic option is the user dictionary file path for Janome.

user_dic_enc user_dic_enc option is the encoding for the user dictionary file specified by `user_dic` option. Default is 'utf8'.

[162] https://pypi.org/project/PorterStemmer/
[163] https://pypi.org/project/PyStemmer/

New in version 1.1.

Changed in version 1.4: html_search_options for Japanese is re-organized and any custom splitter can be used by *type* settings.

The Chinese support has these options:

- `dict` – the `jieba` dictionary path if want to use custom dictionary.

`html_search_scorer`
The name of a JavaScript file (relative to the configuration directory) that implements a search results scorer. If empty, the default will be used.

New in version 1.2.

`html_scaled_image_link`
If true, images itself links to the original image if it doesn't have 'target' option or scale related options: 'scale', 'width', 'height'. The default is `True`.

Document authors can this feature manually with giving `no-scaled-link` class to the image:

```
.. image:: sphinx.png
   :scale: 50%
   :class: no-scaled-link
```

New in version 1.3.

Changed in version 3.0: It is disabled for images having `no-scaled-link` class

`html_math_renderer`
The name of math_renderer extension for HTML output. The default is `'mathjax'`.

New in version 1.8.

`html_experimental_html5_writer`
Output is processed with HTML5 writer. Default is `False`.

New in version 1.6.

Deprecated since version 2.0.

`html4_writer`
Output is processed with HTML4 writer. Default is `False`.

Options for Single HTML output

`singlehtml_sidebars`
Custom sidebar templates, must be a dictionary that maps document names to template names. And it only allows a key named *'index'*. All other keys are ignored. For more information, refer to *`html_sidebars`*. By default, it is same as *`html_sidebars`*.

[164] https://pypi.org/project/Janome/

Options for HTML help output

htmlhelp_basename
Output file base name for HTML help builder. Default is `'pydoc'`.

htmlhelp_file_suffix
This is the file name suffix for generated HTML help files. The default is `".html"`.

New in version 2.0.

htmlhelp_link_suffix
Suffix for generated links to HTML files. The default is `".html"`.

New in version 2.0.

Options for Apple Help output

New in version 1.3.

These options influence the Apple Help output. This builder derives from the HTML builder, so the HTML options also apply where appropriate.

Note: Apple Help output will only work on Mac OS X 10.6 and higher, as it requires the **hiutil** and **codesign** command line tools, neither of which are Open Source.

You can disable the use of these tools using *applehelp_disable_external_tools*, but the result will not be a valid help book until the indexer is run over the `.lproj` folders within the bundle.

applehelp_bundle_name
The basename for the Apple Help Book. Defaults to the *project* name.

applehelp_bundle_id
The bundle ID for the help book bundle.

Warning: You *must* set this value in order to generate Apple Help.

applehelp_dev_region
The development region. Defaults to `'en-us'`, which is Apple's recommended setting.

applehelp_bundle_version
The bundle version (as a string). Defaults to `'1'`.

applehelp_icon
The help bundle icon file, or `None` for no icon. According to Apple's documentation, this should be a 16-by-16 pixel version of the application's icon with a transparent background, saved as a PNG file.

applehelp_kb_product
The product tag for use with *applehelp_kb_url*. Defaults to `'<project>-<release>'`.

applehelp_kb_url
The URL for your knowledgebase server, e.g. `https://example.com/kbsearch.py?p='product'&q='query'&l='lang'`. Help Viewer will replace the values `'product'`, `'query'` and `'lang'` at runtime with the contents of *applehelp_kb_product*, the text entered by the user in the search box and the user's system language respectively.

Defaults to `None` for no remote search.

applehelp_remote_url
The URL for remote content. You can place a copy of your Help Book's Resources folder at this location and Help Viewer will attempt to use it to fetch updated content.

e.g. if you set it to `https://example.com/help/Foo/` and Help Viewer wants a copy of `index.html` for an English speaking customer, it will look at `https://example.com/help/Foo/en.lproj/index.html`.

Defaults to `None` for no remote content.

applehelp_index_anchors
If `True`, tell the help indexer to index anchors in the generated HTML. This can be useful for jumping to a particular topic using the `AHLookupAnchor` function or the `openHelpAnchor:inBook:` method in your code. It also allows you to use `help:anchor` URLs; see the Apple documentation for more information on this topic.

applehelp_min_term_length
Controls the minimum term length for the help indexer. Defaults to `None`, which means the default will be used.

applehelp_stopwords
Either a language specification (to use the built-in stopwords), or the path to a stopwords plist, or `None` if you do not want to use stopwords. The default stopwords plist can be found at `/usr/share/hiutil/Stopwords.plist` and contains, at time of writing, stopwords for the following languages:

Language	Code
English	en
German	de
Spanish	es
French	fr
Swedish	sv
Hungarian	hu
Italian	it

Defaults to *language*, or if that is not set, to `en`.

applehelp_locale
Specifies the locale to generate help for. This is used to determine the name of the `.lproj` folder inside the Help Book's `Resources`, and is passed to the help indexer.

Defaults to *language*, or if that is not set, to `en`.

applehelp_title
Specifies the help book title. Defaults to `'<project> Help'`.

applehelp_codesign_identity
Specifies the identity to use for code signing, or `None` if code signing is not to be performed.

Defaults to the value of the environment variable `CODE_SIGN_IDENTITY`, which is set by Xcode for script build phases, or `None` if that variable is not set.

applehelp_codesign_flags
A *list* of additional arguments to pass to **codesign** when signing the help book.

Defaults to a list based on the value of the environment variable `OTHER_CODE_SIGN_FLAGS`, which is set by Xcode for script build phases, or the empty list if that variable is not set.

applehelp_indexer_path
The path to the **hiutil** program. Defaults to `'/usr/bin/hiutil'`.

applehelp_codesign_path
The path to the **codesign** program. Defaults to `'/usr/bin/codesign'`.

applehelp_disable_external_tools
If `True`, the builder will not run the indexer or the code signing tool, no matter what other settings are specified.

This is mainly useful for testing, or where you want to run the Sphinx build on a non-Mac OS X platform and then complete the final steps on OS X for some reason.

Defaults to `False`.

Options for epub output

These options influence the epub output. As this builder derives from the HTML builder, the HTML options also apply where appropriate. The actual values for some of the options is not really important, they just have to be entered into the Dublin Core metadata[165].

`epub_basename`
The basename for the epub file. It defaults to the *`project`* name.

`epub_theme`
The HTML theme for the epub output. Since the default themes are not optimized for small screen space, using the same theme for HTML and epub output is usually not wise. This defaults to `'epub'`, a theme designed to save visual space.

`epub_theme_options`
A dictionary of options that influence the look and feel of the selected theme. These are theme-specific. For the options understood by the builtin themes, see *this section*.

New in version 1.2.

`epub_title`
The title of the document. It defaults to the *`html_title`* option but can be set independently for epub creation. It defaults to the *`project`* option.

Changed in version 2.0: It defaults to the `project` option.

`epub_description`
The description of the document. The default value is `'unknown'`.

New in version 1.4.

Changed in version 1.5: Renamed from `epub3_description`

`epub_author`
The author of the document. This is put in the Dublin Core metadata. It defaults to the *`author`* option.

`epub_contributor`
The name of a person, organization, etc. that played a secondary role in the creation of the content of an EPUB Publication. The default value is `'unknown'`.

New in version 1.4.

Changed in version 1.5: Renamed from `epub3_contributor`

`epub_language`
The language of the document. This is put in the Dublin Core metadata. The default is the *`language`* option or `'en'` if unset.

`epub_publisher`
The publisher of the document. This is put in the Dublin Core metadata. You may use any sensible string, e.g. the project homepage. The defaults to the *`author`* option.

`epub_copyright`
The copyright of the document. It defaults to the *`copyright`* option but can be set independently for epub creation.

[165] http://dublincore.org/

epub_identifier
An identifier for the document. This is put in the Dublin Core metadata. For published documents this is the ISBN number, but you can also use an alternative scheme, e.g. the project homepage. The default value is `'unknown'`.

epub_scheme
The publication scheme for the *epub_identifier*. This is put in the Dublin Core metadata. For published books the scheme is `'ISBN'`. If you use the project homepage, `'URL'` seems reasonable. The default value is `'unknown'`.

epub_uid
A unique identifier for the document. This is put in the Dublin Core metadata. You may use a XML's Name format[166] string. You can't use hyphen, period, numbers as a first character. The default value is `'unknown'`.

epub_cover
The cover page information. This is a tuple containing the filenames of the cover image and the html template. The rendered html cover page is inserted as the first item in the spine in `content.opf`. If the template filename is empty, no html cover page is created. No cover at all is created if the tuple is empty. Examples:

```
epub_cover = ('_static/cover.png', 'epub-cover.html')
epub_cover = ('_static/cover.png', '')
epub_cover = ()
```

The default value is `()`.

New in version 1.1.

epub_css_files
A list of CSS files. The entry must be a *filename* string or a tuple containing the *filename* string and the *attributes* dictionary. For more information, see *html_css_files*.

New in version 1.8.

epub_guide
Meta data for the guide element of `content.opf`. This is a sequence of tuples containing the *type*, the *uri* and the *title* of the optional guide information. See the OPF documentation at http://idpf.org/epub for details. If possible, default entries for the *cover* and *toc* types are automatically inserted. However, the types can be explicitly overwritten if the default entries are not appropriate. Example:

```
epub_guide = (('cover', 'cover.html', u'Cover Page'),)
```

The default value is `()`.

epub_pre_files
Additional files that should be inserted before the text generated by Sphinx. It is a list of tuples containing the file name and the title. If the title is empty, no entry is added to `toc.ncx`. Example:

```
epub_pre_files = [
    ('index.html', 'Welcome'),
]
```

The default value is `[]`.

epub_post_files
Additional files that should be inserted after the text generated by Sphinx. It is a list of tuples containing the file name and the title. This option can be used to add an appendix. If the title is empty, no entry is added to `toc.ncx`. The default value is `[]`.

[166] https://www.w3.org/TR/REC-xml/#NT-NameStartChar

epub_exclude_files
A list of files that are generated/copied in the build directory but should not be included in the epub file. The default value is `[]`.

epub_tocdepth
The depth of the table of contents in the file `toc.ncx`. It should be an integer greater than zero. The default value is 3. Note: A deeply nested table of contents may be difficult to navigate.

epub_tocdup
This flag determines if a toc entry is inserted again at the beginning of its nested toc listing. This allows easier navigation to the top of a chapter, but can be confusing because it mixes entries of different depth in one list. The default value is `True`.

epub_tocscope
This setting control the scope of the epub table of contents. The setting can have the following values:

- `'default'` – include all toc entries that are not hidden (default)
- `'includehidden'` – include all toc entries

New in version 1.2.

epub_fix_images
This flag determines if sphinx should try to fix image formats that are not supported by some epub readers. At the moment palette images with a small color table are upgraded. You need Pillow, the Python Image Library, installed to use this option. The default value is `False` because the automatic conversion may lose information.

New in version 1.2.

epub_max_image_width
This option specifies the maximum width of images. If it is set to a value greater than zero, images with a width larger than the given value are scaled accordingly. If it is zero, no scaling is performed. The default value is `0`. You need the Python Image Library (Pillow) installed to use this option.

New in version 1.2.

epub_show_urls
Control whether to display URL addresses. This is very useful for readers that have no other means to display the linked URL. The settings can have the following values:

- `'inline'` – display URLs inline in parentheses (default)
- `'footnote'` – display URLs in footnotes
- `'no'` – do not display URLs

The display of inline URLs can be customized by adding CSS rules for the class `link-target`.

New in version 1.2.

epub_use_index
If true, add an index to the epub document. It defaults to the *`html_use_index`* option but can be set independently for epub creation.

New in version 1.2.

epub_writing_mode
It specifies writing direction. It can accept `'horizontal'` (default) and `'vertical'`

epub_writing_mode	'horizontal'	'vertical'
writing-mode[167]	horizontal-tb	vertical-rl
page progression	left to right	right to left
iBook's Scroll Theme support	scroll-axis is vertical.	scroll-axis is horizontal.

[167] https://developer.mozilla.org/en-US/docs/Web/CSS/writing-mode

Options for LaTeX output

These options influence LaTeX output.

`latex_engine`

The LaTeX engine to build the docs. The setting can have the following values:

- `'pdflatex'` – PDFLaTeX (default)
- `'xelatex'` – XeLaTeX
- `'lualatex'` – LuaLaTeX
- `'platex'` – pLaTeX
- `'uplatex'` – upLaTeX (default if *`language`* is `'ja'`)

`'pdflatex'`'s support for Unicode characters is limited.

Note: 2.0 adds to `'pdflatex'` support in Latin language document of occasional Cyrillic or Greek letters or words. This is not automatic, see the discussion of the *`latex_elements`* `'fontenc'` key.

If your project uses Unicode characters, setting the engine to `'xelatex'` or `'lualatex'` and making sure to use an OpenType font with wide-enough glyph coverage is often easier than trying to make `'pdflatex'` work with the extra Unicode characters. Since Sphinx 2.0 the default is the GNU FreeFont which covers well Latin, Cyrillic and Greek.

Changed in version 2.1.0: Use `xelatex` (and LaTeX package `xeCJK`) by default for Chinese documents.

Changed in version 2.2.1: Use `xelatex` by default for Greek documents.

Changed in version 2.3: Add `uplatex` support.

Changed in version 4.0: `uplatex` becomes the default setting of Japanese documents.

Contrarily to *MathJaX math rendering in HTML output*, LaTeX requires some extra configuration to support Unicode literals in *`math`*: the only comprehensive solution (as far as we know) is to use `'xelatex'` or `'lualatex'` *and* to add `r'\usepackage{unicode-math}'` (e.g. via the *`latex_elements`* `'preamble'` key). You may prefer `r'\usepackage[math-style=literal]{unicode-math}'` to keep a Unicode literal such as α (U+03B1) for example as is in output, rather than being rendered as α.

`latex_documents`

This value determines how to group the document tree into LaTeX source files. It must be a list of tuples `(startdocname, targetname, title, author, theme, toctree_only)`, where the items are:

startdocname String that specifies the *document name* of the LaTeX file's master document. All documents referenced by the *startdoc* document in TOC trees will be included in the LaTeX file. (If you want to use the default master document for your LaTeX build, provide your *`master_doc`* here.)

targetname File name of the LaTeX file in the output directory.

title LaTeX document title. Can be empty to use the title of the *startdoc* document. This is inserted as LaTeX markup, so special characters like a backslash or ampersand must be represented by the proper LaTeX commands if they are to be inserted literally.

author Author for the LaTeX document. The same LaTeX markup caveat as for *title* applies. Use `\\and` to separate multiple authors, as in: `'John \\and Sarah'` (backslashes must be Python-escaped to reach LaTeX).

theme LaTeX theme. See *`latex_theme`*.

toctree_only Must be `True` or `False`. If true, the *startdoc* document itself is not included in the output, only the documents referenced by it via TOC trees. With this option, you can put extra stuff in the master document that shows up in the HTML, but not the LaTeX output.

New in version 1.2: In the past including your own document class required you to prepend the document class name with the string "sphinx". This is not necessary anymore.

New in version 0.3: The 6th item `toctree_only`. Tuples with 5 items are still accepted.

`latex_logo`
If given, this must be the name of an image file (relative to the configuration directory) that is the logo of the docs. It is placed at the top of the title page. Default: `None`.

`latex_toplevel_sectioning`
This value determines the topmost sectioning unit. It should be chosen from `'part'`, `'chapter'` or `'section'`. The default is `None`; the topmost sectioning unit is switched by documentclass: `section` is used if documentclass will be `howto`, otherwise `chapter` will be used.

Note that if LaTeX uses `\part` command, then the numbering of sectioning units one level deep gets off-sync with HTML numbering, because LaTeX numbers continuously `\chapter` (or `\section` for `howto`.)

New in version 1.4.

`latex_appendices`
A list of document names to append as an appendix to all manuals.

`latex_domain_indices`
If true, generate domain-specific indices in addition to the general index. For e.g. the Python domain, this is the global module index. Default is `True`.

This value can be a bool or a list of index names that should be generated, like for *`html_domain_indices`*.

New in version 1.0.

`latex_show_pagerefs`
If true, add page references after internal references. This is very useful for printed copies of the manual. Default is `False`.

New in version 1.0.

`latex_show_urls`
Control whether to display URL addresses. This is very useful for printed copies of the manual. The setting can have the following values:

- `'no'` – do not display URLs (default)
- `'footnote'` – display URLs in footnotes
- `'inline'` – display URLs inline in parentheses

New in version 1.0.

Changed in version 1.1: This value is now a string; previously it was a boolean value, and a true value selected the `'inline'` display. For backwards compatibility, `True` is still accepted.

`latex_use_latex_multicolumn`
The default is `False`: it means that Sphinx's own macros are used for merged cells from grid tables. They allow general contents (literal blocks, lists, blockquotes, ...) but may have problems if the *`tabularcolumns`* directive was used to inject LaTeX mark-up of the type `>{..}`, `<{..}`, `@{..}` as column specification.

Setting to `True` means to use LaTeX's standard `\multicolumn`; this is incompatible with literal blocks in the horizontally merged cell, and also with multiple paragraphs in such cell if the table is rendered using `tabulary`.

New in version 1.6.

`latex_use_xindy`
If `True`, the PDF build from the LaTeX files created by Sphinx will use **`xindy`** (doc[168]) rather than **`makeindex`** for preparing the index of general terms (from *`index`* usage). This means that words with UTF-8 characters will get ordered correctly for the *`language`*.

- This option is ignored if *`latex_engine`* is `'platex'` (Japanese documents; **`mendex`** replaces **`makeindex`** then).
- The default is `True` for `'xelatex'` or `'lualatex'` as **`makeindex`**, if any indexed term starts with a non-ascii character, creates `.ind` files containing invalid bytes for UTF-8 encoding. With `'lualatex'` this then breaks the PDF build.
- The default is `False` for `'pdflatex'` but `True` is recommended for non-English documents as soon as some indexed terms use non-ascii characters from the language script.

Sphinx adds to **`xindy`** base distribution some dedicated support for using `'pdflatex'` engine with Cyrillic scripts. And whether with `'pdflatex'` or Unicode engines, Cyrillic documents handle correctly the indexing of Latin names, even with diacritics.

New in version 1.8.

`latex_elements`

New in version 0.5.

Its *documentation* has moved to *LaTeX customization*.

`latex_docclass`

A dictionary mapping `'howto'` and `'manual'` to names of real document classes that will be used as the base for the two Sphinx classes. Default is to use `'article'` for `'howto'` and `'report'` for `'manual'`.

New in version 1.0.

Changed in version 1.5: In Japanese docs (*`language`* is `'ja'`), by default `'jreport'` is used for `'howto'` and `'jsbook'` for `'manual'`.

`latex_additional_files`

A list of file names, relative to the configuration directory, to copy to the build directory when building LaTeX output. This is useful to copy files that Sphinx doesn't copy automatically, e.g. if they are referenced in custom LaTeX added in `latex_elements`. Image files that are referenced in source files (e.g. via `.. image::`) are copied automatically.

You have to make sure yourself that the filenames don't collide with those of any automatically copied files.

New in version 0.6.

Changed in version 1.2: This overrides the files which is provided from Sphinx such as `sphinx.sty`.

`latex_theme`

The "theme" that the LaTeX output should use. It is a collection of settings for LaTeX output (ex. document class, top level sectioning unit and so on).

As a built-in LaTeX themes, `manual` and `howto` are bundled.

`manual` A LaTeX theme for writing a manual. It imports the `report` document class (Japanese documents use `jsbook`).

`howto` A LaTeX theme for writing an article. It imports the `article` document class (Japanese documents use `jreport` rather). *`latex_appendices`* is available only for this theme.

It defaults to `'manual'`.

New in version 3.0.

`latex_theme_options`

A dictionary of options that influence the look and feel of the selected theme.

New in version 3.1.

[168] http://xindy.sourceforge.net/

`latex_theme_path`
A list of paths that contain custom LaTeX themes as subdirectories. Relative paths are taken as relative to the configuration directory.

New in version 3.0.

Options for text output

These options influence text output.

`text_newlines`
Determines which end-of-line character(s) are used in text output.

- `'unix'`: use Unix-style line endings (`\n`)
- `'windows'`: use Windows-style line endings (`\r\n`)
- `'native'`: use the line ending style of the platform the documentation is built on

Default: `'unix'`.

New in version 1.1.

`text_sectionchars`
A string of 7 characters that should be used for underlining sections. The first character is used for first-level headings, the second for second-level headings and so on.

The default is `'*=-~"+`'`.

New in version 1.1.

`text_add_secnumbers`
A boolean that decides whether section numbers are included in text output. Default is `True`.

New in version 1.7.

`text_secnumber_suffix`
Suffix for section numbers in text output. Default: `". "`. Set to `" "` to suppress the final dot on section numbers.

New in version 1.7.

Options for manual page output

These options influence manual page output.

`man_pages`
This value determines how to group the document tree into manual pages. It must be a list of tuples `(startdocname, name, description, authors, section)`, where the items are:

startdocname String that specifies the *document name* of the manual page's master document. All documents referenced by the *startdoc* document in TOC trees will be included in the manual file. (If you want to use the default master document for your manual pages build, use your `master_doc` here.)

name Name of the manual page. This should be a short string without spaces or special characters. It is used to determine the file name as well as the name of the manual page (in the NAME section).

description Description of the manual page. This is used in the NAME section.

authors A list of strings with authors, or a single string. Can be an empty string or list if you do not want to automatically generate an AUTHORS section in the manual page.

section The manual page section. Used for the output file name as well as in the manual page header.

New in version 1.0.

`man_show_urls`
If true, add URL addresses after links. Default is `False`.

New in version 1.1.

`man_make_section_directory`
If true, make a section directory on build man page. Default is True.

New in version 3.3.

Changed in version 4.0: The default is changed to `False` from `True`.

Options for Texinfo output

These options influence Texinfo output.

`texinfo_documents`
This value determines how to group the document tree into Texinfo source files. It must be a list of tuples `(startdocname, targetname, title, author, dir_entry, description, category, toctree_only)`, where the items are:

startdocname String that specifies the *document name* of the the Texinfo file's master document. All documents referenced by the *startdoc* document in TOC trees will be included in the Texinfo file. (If you want to use the default master document for your Texinfo build, provide your *`master_doc`* here.)

targetname File name (no extension) of the Texinfo file in the output directory.

title Texinfo document title. Can be empty to use the title of the *startdoc* document. Inserted as Texinfo markup, so special characters like `@` and `{}` will need to be escaped to be inserted literally.

author Author for the Texinfo document. Inserted as Texinfo markup. Use `@*` to separate multiple authors, as in: `'John@*Sarah'`.

dir_entry The name that will appear in the top-level `DIR` menu file.

description Descriptive text to appear in the top-level `DIR` menu file.

category Specifies the section which this entry will appear in the top-level `DIR` menu file.

toctree_only Must be `True` or `False`. If true, the *startdoc* document itself is not included in the output, only the documents referenced by it via TOC trees. With this option, you can put extra stuff in the master document that shows up in the HTML, but not the Texinfo output.

New in version 1.1.

`texinfo_appendices`
A list of document names to append as an appendix to all manuals.

New in version 1.1.

`texinfo_domain_indices`
If true, generate domain-specific indices in addition to the general index. For e.g. the Python domain, this is the global module index. Default is `True`.

This value can be a bool or a list of index names that should be generated, like for *`html_domain_indices`*.

New in version 1.1.

`texinfo_show_urls`
Control how to display URL addresses.

- `'footnote'` – display URLs in footnotes (default)
- `'no'` – do not display URLs
- `'inline'` – display URLs inline in parentheses

New in version 1.1.

`texinfo_no_detailmenu`
If true, do not generate a `@detailmenu` in the "Top" node's menu containing entries for each sub-node in the document. Default is `False`.

New in version 1.2.

`texinfo_elements`
A dictionary that contains Texinfo snippets that override those Sphinx usually puts into the generated `.texi` files.

- Keys that you may want to override include:

  `'paragraphindent'` Number of spaces to indent the first line of each paragraph, default `2`. Specify `0` for no indentation.

  `'exampleindent'` Number of spaces to indent the lines for examples or literal blocks, default `4`. Specify `0` for no indentation.

  `'preamble'` Texinfo markup inserted near the beginning of the file.

  `'copying'` Texinfo markup inserted within the `@copying` block and displayed after the title. The default value consists of a simple title page identifying the project.

- Keys that are set by other options and therefore should not be overridden are:

  `'author'` `'body'` `'date'` `'direntry'` `'filename'` `'project'` `'release'` `'title'`

New in version 1.1.

Options for QtHelp output

These options influence qthelp output. As this builder derives from the HTML builder, the HTML options also apply where appropriate.

`qthelp_basename`
The basename for the qthelp file. It defaults to the *`project`* name.

`qthelp_namespace`
The namespace for the qthelp file. It defaults to `org.sphinx.<project_name>.<project_version>`.

`qthelp_theme`
The HTML theme for the qthelp output. This defaults to `'nonav'`.

`qthelp_theme_options`
A dictionary of options that influence the look and feel of the selected theme. These are theme-specific. For the options understood by the builtin themes, see *this section*.

Options for the linkcheck builder

`linkcheck_ignore`
A list of regular expressions that match URIs that should not be checked when doing a `linkcheck` build. Example:

```
linkcheck_ignore = [r'http://localhost:\d+/']
```

New in version 1.1.

`linkcheck_request_headers`
A dictionary that maps baseurls to HTTP request headers.

The key is a URL base string like `"https://sphinx-doc.org/"`. To specify headers for other hosts, `"*"` can be used. It matches all hosts only when the URL does not match other settings.

The value is a dictionary that maps header name to its value.

Example:

```
linkcheck_request_headers = {
    "https://sphinx-doc.org/": {
        "Accept": "text/html",
        "Accept-Encoding": "utf-8",
    },
    "*": {
        "Accept": "text/html,application/xhtml+xml",
    }
}
```

New in version 3.1.

`linkcheck_retries`
The number of times the linkcheck builder will attempt to check a URL before declaring it broken. Defaults to 1 attempt.

New in version 1.4.

`linkcheck_timeout`
A timeout value, in seconds, for the linkcheck builder. The default is to use Python's global socket timeout.

New in version 1.1.

`linkcheck_workers`
The number of worker threads to use when checking links. Default is 5 threads.

New in version 1.1.

`linkcheck_anchors`
If true, check the validity of `#anchors` in links. Since this requires downloading the whole document, it's considerably slower when enabled. Default is `True`.

New in version 1.2.

`linkcheck_anchors_ignore`
A list of regular expressions that match anchors Sphinx should skip when checking the validity of anchors in links. This allows skipping anchors that a website's JavaScript adds to control dynamic pages or when triggering an internal REST request. Default is `["^!"]`.

Note: If you want to ignore anchors of a specific page or of pages that match a specific pattern (but still check occurrences of the same page(s) that don't have anchors), use *`linkcheck_ignore`* instead, for example as follows:

```
linkcheck_ignore = [
   'http://www.sphinx-doc.org/en/1.7/intro.html#'
]
```

New in version 1.5.

`linkcheck_auth`
Pass authentication information when doing a `linkcheck` build.

A list of `(regex_pattern, auth_info)` tuples where the items are:

regex_pattern A regular expression that matches a URI.

auth_info Authentication information to use for that URI. The value can be anything that is understood by the `requests` library (see requests Authentication for details).

The `linkcheck` builder will use the first matching `auth_info` value it can find in the *linkcheck_auth* list, so values earlier in the list have higher priority.

Example:

```
linkcheck_auth = [
  ('https://foo\.yourcompany\.com/.+', ('johndoe', 'secret')),
  ('https://.+\.yourcompany\.com/.+', HTTPDigestAuth(...)),
]
```

New in version 2.3.

linkcheck_rate_limit_timeout

The `linkcheck` builder may issue a large number of requests to the same site over a short period of time. This setting controls the builder behavior when servers indicate that requests are rate-limited.

If a server indicates when to retry (using the Retry-After[169] header), `linkcheck` always follows the server indication.

Otherwise, `linkcheck` waits for a minute before to retry and keeps doubling the wait time between attempts until it succeeds or exceeds the `linkcheck_rate_limit_timeout`. By default, the timeout is 5 minutes.

New in version 3.4.

Options for the XML builder

xml_pretty

If true, pretty-print the XML. Default is `True`.

New in version 1.2.

Options for the C domain

c_id_attributes

A list of strings that the parser additionally should accept as attributes. This can for example be used when attributes have been `#define` d for portability.

New in version 3.0.

c_paren_attributes

A list of strings that the parser additionally should accept as attributes with one argument. That is, if `my_align_as` is in the list, then `my_align_as(X)` is parsed as an attribute for all strings `X` that have balanced braces (`()`, `[]`, and `{}`). This can for example be used when attributes have been `#define` d for portability.

New in version 3.0.

c_allow_pre_v3

A boolean (default `False`) controlling whether to parse and try to convert pre-v3 style type directives and type roles.

New in version 3.2.

Deprecated since version 3.2: Use the directives and roles added in v3.

[169] https://tools.ietf.org/html/rfc7231#section-7.1.3

c_warn_on_allowed_pre_v3

A boolean (default `True`) controlling whether to warn when a pre-v3 style type directive/role is parsed and converted.

New in version 3.2.

Deprecated since version 3.2: Use the directives and roles added in v3.

Options for the C++ domain

cpp_index_common_prefix

A list of prefixes that will be ignored when sorting C++ objects in the global index. For example `['awesome_lib::']`.

New in version 1.5.

cpp_id_attributes

A list of strings that the parser additionally should accept as attributes. This can for example be used when attributes have been `#define` d for portability.

New in version 1.5.

cpp_paren_attributes

A list of strings that the parser additionally should accept as attributes with one argument. That is, if `my_align_as` is in the list, then `my_align_as(X)` is parsed as an attribute for all strings `X` that have balanced braces (`()`, `[]`, and `{}`). This can for example be used when attributes have been `#define` d for portability.

New in version 1.5.

Example of configuration file

```
# test documentation build configuration file, created by
# sphinx-quickstart on Sun Jun 26 00:00:43 2016.
#
# This file is execfile()d with the current directory set to its
# containing dir.
#
# Note that not all possible configuration values are present in this
# autogenerated file.
#
# All configuration values have a default; values that are commented out
# serve to show the default.

# If extensions (or modules to document with autodoc) are in another directory,
# add these directories to sys.path here. If the directory is relative to the
# documentation root, use os.path.abspath to make it absolute, like shown here.
#
# import os
# import sys
# sys.path.insert(0, os.path.abspath('.'))

# -- General configuration ------------------------------------------------

# If your documentation needs a minimal Sphinx version, state it here.
#
# needs_sphinx = '1.0'
```

(continues on next page)

(continued from previous page)

```
# Add any Sphinx extension module names here, as strings. They can be
# extensions coming with Sphinx (named 'sphinx.ext.*') or your custom
# ones.
extensions = []

# Add any paths that contain templates here, relative to this directory.
templates_path = ['_templates']

# The suffix(es) of source filenames.
# You can specify multiple suffix as a list of string:
#
# source_suffix = ['.rst', '.md']
source_suffix = '.rst'

# The encoding of source files.
#
# source_encoding = 'utf-8-sig'

# The master toctree document.
master_doc = 'index'

# General information about the project.
project = u'test'
copyright = u'2016, test'
author = u'test'

# The version info for the project you're documenting, acts as replacement for
# |version| and |release|, also used in various other places throughout the
# built documents.
#
# The short X.Y version.
version = u'test'
# The full version, including alpha/beta/rc tags.
release = u'test'

# The language for content autogenerated by Sphinx. Refer to documentation
# for a list of supported languages.
#
# This is also used if you do content translation via gettext catalogs.
# Usually you set "language" from the command line for these cases.
language = None

# There are two options for replacing |today|: either, you set today to some
# non-false value, then it is used:
#
# today = ''
#
# Else, today_fmt is used as the format for a strftime call.
#
# today_fmt = '%B %d, %Y'
```

(continues on next page)

(continued from previous page)

```
# List of patterns, relative to source directory, that match files and
# directories to ignore when looking for source files.
# These patterns also affect html_static_path and html_extra_path
exclude_patterns = ['_build', 'Thumbs.db', '.DS_Store']

# The reST default role (used for this markup: `text`) to use for all
# documents.
#
# default_role = None

# If true, '()' will be appended to :func: etc. cross-reference text.
#
# add_function_parentheses = True

# If true, the current module name will be prepended to all description
# unit titles (such as .. function::).
#
# add_module_names = True

# If true, sectionauthor and moduleauthor directives will be shown in the
# output. They are ignored by default.
#
# show_authors = False

# The name of the Pygments (syntax highlighting) style to use.
pygments_style = 'sphinx'

# A list of ignored prefixes for module index sorting.
# modindex_common_prefix = []

# If true, keep warnings as "system message" paragraphs in the built documents.
# keep_warnings = False

# If true, `todo` and `todoList` produce output, else they produce nothing.
todo_include_todos = False


# -- Options for HTML output ----------------------------------------------

# The theme to use for HTML and HTML Help pages.  See the documentation for
# a list of builtin themes.
#
html_theme = 'alabaster'

# Theme options are theme-specific and customize the look and feel of a theme
# further.  For a list of options available for each theme, see the
# documentation.
#
# html_theme_options = {}

# Add any paths that contain custom themes here, relative to this directory.
# html_theme_path = []
```

(continues on next page)

(continued from previous page)

```
# The name for this set of Sphinx documents.
# "<project> v<release> documentation" by default.
#
# html_title = u'test vtest'

# A shorter title for the navigation bar.  Default is the same as html_title.
#
# html_short_title = None

# The name of an image file (relative to this directory) to place at the top
# of the sidebar.
#
# html_logo = None

# The name of an image file (relative to this directory) to use as a favicon of
# the docs.  This file should be a Windows icon file (.ico) being 16x16 or 32x32
# pixels large.
#
# html_favicon = None

# Add any paths that contain custom static files (such as style sheets) here,
# relative to this directory. They are copied after the builtin static files,
# so a file named "default.css" will overwrite the builtin "default.css".
html_static_path = ['_static']

# Add any extra paths that contain custom files (such as robots.txt or
# .htaccess) here, relative to this directory. These files are copied
# directly to the root of the documentation.
#
# html_extra_path = []

# If not None, a 'Last updated on:' timestamp is inserted at every page
# bottom, using the given strftime format.
# The empty string is equivalent to '%b %d, %Y'.
#
# html_last_updated_fmt = None

# Custom sidebar templates, maps document names to template names.
#
# html_sidebars = {}

# Additional templates that should be rendered to pages, maps page names to
# template names.
#
# html_additional_pages = {}

# If false, no module index is generated.
#
# html_domain_indices = True

# If false, no index is generated.
```

(continues on next page)

(continued from previous page)

```
#
# html_use_index = True

# If true, the index is split into individual pages for each letter.
#
# html_split_index = False

# If true, links to the reST sources are added to the pages.
#
# html_show_sourcelink = True

# If true, "Created using Sphinx" is shown in the HTML footer. Default is True.
#
# html_show_sphinx = True

# If true, "(C) Copyright ..." is shown in the HTML footer. Default is True.
#
# html_show_copyright = True

# If true, an OpenSearch description file will be output, and all pages will
# contain a <link> tag referring to it.  The value of this option must be the
# base URL from which the finished HTML is served.
#
# html_use_opensearch = ''

# This is the file name suffix for HTML files (e.g. ".xhtml").
# html_file_suffix = None

# Language to be used for generating the HTML full-text search index.
# Sphinx supports the following languages:
#   'da', 'de', 'en', 'es', 'fi', 'fr', 'hu', 'it', 'ja'
#   'nl', 'no', 'pt', 'ro', 'ru', 'sv', 'tr', 'zh'
#
# html_search_language = 'en'

# A dictionary with options for the search language support, empty by default.
# 'ja' uses this config value.
# 'zh' user can custom change `jieba` dictionary path.
#
# html_search_options = {'type': 'default'}

# The name of a javascript file (relative to the configuration directory) that
# implements a search results scorer. If empty, the default will be used.
#
# html_search_scorer = 'scorer.js'

# Output file base name for HTML help builder.
htmlhelp_basename = 'testdoc'

# -- Options for LaTeX output ---------------------------------------------

latex_elements = {
```

(continues on next page)

(continued from previous page)

```
    # The paper size ('letterpaper' or 'a4paper').
    #
    # 'papersize': 'letterpaper',

    # The font size ('10pt', '11pt' or '12pt').
    #
    # 'pointsize': '10pt',

    # Additional stuff for the LaTeX preamble.
    #
    # 'preamble': '',

    # Latex figure (float) alignment
    #
    # 'figure_align': 'htbp',
}

# Grouping the document tree into LaTeX files. List of tuples
# (source start file, target name, title,
#  author, documentclass [howto, manual, or own class]).
latex_documents = [
    (master_doc, 'test.tex', u'test Documentation',
     u'test', 'manual'),
]

# The name of an image file (relative to this directory) to place at the top of
# the title page.
#
# latex_logo = None

# If true, show page references after internal links.
#
# latex_show_pagerefs = False

# If true, show URL addresses after external links.
#
# latex_show_urls = False

# Documents to append as an appendix to all manuals.
#
# latex_appendices = []

# If false, no module index is generated.
#
# latex_domain_indices = True


# -- Options for manual page output ---------------------------------------

# One entry per manual page. List of tuples
# (source start file, name, description, authors, manual section).
man_pages = [
```

(continues on next page)

(continued from previous page)

```
    (master_doc, 'test', u'test Documentation',
     [author], 1)
]

# If true, show URL addresses after external links.
#
# man_show_urls = False


# -- Options for Texinfo output -------------------------------------------

# Grouping the document tree into Texinfo files. List of tuples
# (source start file, target name, title, author,
#  dir menu entry, description, category)
texinfo_documents = [
    (master_doc, 'test', u'test Documentation',
     author, 'test', 'One line description of project.',
     'Miscellaneous'),
]

# Documents to append as an appendix to all manuals.
#
# texinfo_appendices = []

# If false, no module index is generated.
#
# texinfo_domain_indices = True

# How to display URL addresses: 'footnote', 'no', or 'inline'.
#
# texinfo_show_urls = 'footnote'

# If true, do not generate a @detailmenu in the "Top" node's menu.
#
# texinfo_no_detailmenu = False

# -- A random example -----------------------------------------------------

import sys, os
sys.path.insert(0, os.path.abspath('.'))
exclude_patterns = ['zzz']

numfig = True
#language = 'ja'

extensions.append('sphinx.ext.todo')
extensions.append('sphinx.ext.autodoc')
#extensions.append('sphinx.ext.autosummary')
extensions.append('sphinx.ext.intersphinx')
extensions.append('sphinx.ext.mathjax')
extensions.append('sphinx.ext.viewcode')
extensions.append('sphinx.ext.graphviz')
```

(continues on next page)

(continued from previous page)

```
autosummary_generate = True
html_theme = 'default'
#source_suffix = ['.rst', '.txt']
```

1.6 Builders

These are the built-in Sphinx builders. More builders can be added by *extensions*.

The builder's "name" must be given to the **-b** command-line option of **sphinx-build** to select a builder.

class sphinx.builders.html.**StandaloneHTMLBuilder**

This is the standard HTML builder. Its output is a directory with HTML files, complete with style sheets and optionally the reST sources. There are quite a few configuration values that customize the output of this builder, see the chapter *Options for HTML output* for details.

name = 'html'

The builder's name, for the -b command line option.

format = 'html'

The builder's output format, or '' if no document output is produced.

supported_image_types = ['image/svg+xml', 'image/png', 'image/gif', 'image/jpeg']

The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

class sphinx.builders.dirhtml.**DirectoryHTMLBuilder**

This is a subclass of the standard HTML builder. Its output is a directory with HTML files, where each file is called `index.html` and placed in a subdirectory named like its page name. For example, the document `markup/rest.rst` will not result in an output file `markup/rest.html`, but `markup/rest/index.html`. When generating links between pages, the `index.html` is omitted, so that the URL would look like `markup/rest/`.

name = 'dirhtml'

The builder's name, for the -b command line option.

format = 'html'

The builder's output format, or '' if no document output is produced.

supported_image_types = ['image/svg+xml', 'image/png', 'image/gif', 'image/jpeg']

The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

New in version 0.6.

class sphinx.builders.singlehtml.**SingleFileHTMLBuilder**

This is an HTML builder that combines the whole project in one output file. (Obviously this only works with smaller projects.) The file is named like the master document. No indices will be generated.

name = 'singlehtml'

The builder's name, for the -b command line option.

format = 'html'

The builder's output format, or '' if no document output is produced.

supported_image_types = ['image/svg+xml', 'image/png', 'image/gif', 'image/jpeg']

The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

New in version 1.0.

`class sphinxcontrib.htmlhelp.HTMLHelpBuilder`

This builder produces the same output as the standalone HTML builder, but also generates HTML Help support files that allow the Microsoft HTML Help Workshop to compile them into a CHM file.

`name = 'htmlhelp'`

The builder's name, for the -b command line option.

`format = 'html'`

The builder's output format, or '' if no document output is produced.

`supported_image_types = ['image/png', 'image/gif', 'image/jpeg']`

The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

`class sphinxcontrib.qthelp.QtHelpBuilder`

This builder produces the same output as the standalone HTML builder, but also generates Qt help[170] collection support files that allow the Qt collection generator to compile them.

Changed in version 2.0: Moved to sphinxcontrib.qthelp from sphinx.builders package.

`name = 'qthelp'`

The builder's name, for the -b command line option.

`format = 'html'`

The builder's output format, or '' if no document output is produced.

`supported_image_types = ['image/svg+xml', 'image/png', 'image/gif', 'image/jpeg']`

The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

`class sphinxcontrib.applehelp.AppleHelpBuilder`

This builder produces an Apple Help Book based on the same output as the standalone HTML builder.

If the source directory contains any `.lproj` folders, the one corresponding to the selected language will have its contents merged with the generated output. These folders will be ignored by all other documentation types.

In order to generate a valid help book, this builder requires the command line tool **hiutil**, which is only available on Mac OS X 10.6 and above. You can disable the indexing step by setting *applehelp_disable_external_tools* to `True`, in which case the output will not be valid until **hiutil** has been run on all of the `.lproj` folders within the bundle.

`name = 'applehelp'`

The builder's name, for the -b command line option.

`format = 'html'`

The builder's output format, or '' if no document output is produced.

`supported_image_types = ['image/png', 'image/gif', 'image/jpeg', 'image/tiff', 'image/jp2', 'image/`

The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

New in version 1.3.

Changed in version 2.0: Moved to sphinxcontrib.applehelp from sphinx.builders package.

`class sphinxcontrib.devhelp.DevhelpBuilder`

This builder produces the same output as the standalone HTML builder, but also generates GNOME Devhelp[171] support file that allows the GNOME Devhelp reader to view them.

`name = 'devhelp'`

The builder's name, for the -b command line option.

[170] https://doc.qt.io/qt-4.8/qthelp-framework.html

format = 'html'
The builder's output format, or '' if no document output is produced.

supported_image_types = ['image/png', 'image/gif', 'image/jpeg']
The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

Changed in version 2.0: Moved to sphinxcontrib.devhelp from sphinx.builders package.

class `sphinx.builders.epub3.`**Epub3Builder**
This builder produces the same output as the standalone HTML builder, but also generates an *epub* file for ebook readers. See *Epub info* for details about it. For definition of the epub format, have a look at http://idpf.org/epub or https://en.wikipedia.org/wiki/EPUB. The builder creates *EPUB 3* files.

name = 'epub'
The builder's name, for the -b command line option.

format = 'html'
The builder's output format, or '' if no document output is produced.

supported_image_types = ['image/svg+xml', 'image/png', 'image/gif', 'image/jpeg']
The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

New in version 1.4.

Changed in version 1.5: Since Sphinx-1.5, the epub3 builder is used for the default builder of epub.

class `sphinx.builders.latex.`**LaTeXBuilder**
This builder produces a bunch of LaTeX files in the output directory. You have to specify which documents are to be included in which LaTeX files via the *`latex_documents`* configuration value. There are a few configuration values that customize the output of this builder, see the chapter *Options for LaTeX output* for details.

The produced LaTeX file uses several LaTeX packages that may not be present in a "minimal" TeX distribution installation.

On Ubuntu xenial, the following packages need to be installed for successful PDF builds:

- `texlive-latex-recommended`
- `texlive-fonts-recommended`
- `tex-gyre` (if *`latex_engine`* is `'pdflatex'`)
- `texlive-latex-extra`
- `latexmk` (this is a Sphinx requirement on GNU/Linux and MacOS X for functioning of `make latexpdf`)

Additional packages are needed in some circumstances (see the discussion of the `'fontpkg'` key of *`latex_elements`* for more information):

- `texlive-lang-cyrillic` for Cyrillic (even individual letters), and, `cm-super` or `cm-super-minimal` (if default fonts),
- `texlive-lang-greek` for Greek (even individual letters), and, `cm-super` or `cm-super-minimal` (if default fonts),
- `texlive-xetex` if *`latex_engine`* is `'xelatex'`,
- `texlive-luatex` if *`latex_engine`* is `'lualatex'`,
- `fonts-freefont-otf` if *`latex_engine`* is `'xelatex'` or `'lualatex'`.

The testing of Sphinx LaTeX is done on Ubuntu xenial whose TeX distribution is based on a TeXLive 2015 snapshot dated March 2016.

[171] https://wiki.gnome.org/Apps/Devhelp

Changed in version 1.6: Formerly, testing had been done on Ubuntu precise (TeXLive 2009).

Changed in version 2.0: Formerly, testing had been done on Ubuntu trusty (TeXLive 2013).

Changed in version 4.0.0: TeX Gyre fonts dependency for the default LaTeX font configuration.

Note: Since 1.6, `make latexpdf` uses `latexmk` (not on Windows). This makes sure the needed number of runs is automatically executed to get the cross-references, bookmarks, indices, and tables of contents right.

One can pass to `latexmk` options via the `LATEXMKOPTS` Makefile variable. For example:

```
make latexpdf LATEXMKOPTS="-silent"
```

reduces console output to a minimum.

Also, if `latexmk` is at version 4.52b or higher (January 2017) `LATEXMKOPTS="-xelatex"` speeds up PDF builds via XeLateX in case of numerous graphics inclusions.

To pass options directly to the `(pdf|xe|lua)latex` binary, use variable `LATEXOPTS`, for example:

```
make latexpdf LATEXOPTS="--halt-on-error"
```

name = 'latex'
The builder's name, for the -b command line option.

format = 'latex'
The builder's output format, or '' if no document output is produced.

supported_image_types = ['application/pdf', 'image/png', 'image/jpeg']
The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

Note that a direct PDF builder is being provided by rinohtype[172]. The builder's name is `rinoh`. Refer to the rinohtype manual[173] for details.

class sphinx.builders.text.TextBuilder
This builder produces a text file for each reST file – this is almost the same as the reST source, but with much of the markup stripped for better readability.

name = 'text'
The builder's name, for the -b command line option.

format = 'text'
The builder's output format, or '' if no document output is produced.

supported_image_types = []
The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

New in version 0.4.

class sphinx.builders.manpage.ManualPageBuilder
This builder produces manual pages in the groff format. You have to specify which documents are to be included in which manual pages via the *man_pages* configuration value.

name = 'man'
The builder's name, for the -b command line option.

format = 'man'
The builder's output format, or '' if no document output is produced.

[172] https://github.com/brechtm/rinohtype
[173] https://www.mos6581.org/rinohtype/quickstart.html#sphinx-builder

supported_image_types = []
The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

New in version 1.0.

class sphinx.builders.texinfo.**TexinfoBuilder**
This builder produces Texinfo files that can be processed into Info files by the **makeinfo** program. You have to specify which documents are to be included in which Texinfo files via the *texinfo_documents* configuration value.

The Info format is the basis of the on-line help system used by GNU Emacs and the terminal-based program **info**. See *Texinfo info* for more details. The Texinfo format is the official documentation system used by the GNU project. More information on Texinfo can be found at https://www.gnu.org/software/texinfo/.

name = 'texinfo'
The builder's name, for the -b command line option.

format = 'texinfo'
The builder's output format, or '' if no document output is produced.

supported_image_types = ['image/png', 'image/jpeg', 'image/gif']
The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

New in version 1.1.

class sphinxcontrib.serializinghtml.**SerializingHTMLBuilder**
This builder uses a module that implements the Python serialization API (*pickle*, *simplejson*, *phpserialize*, and others) to dump the generated HTML documentation. The pickle builder is a subclass of it.

A concrete subclass of this builder serializing to the PHP serialization[174] format could look like this:

```
import phpserialize

class PHPSerializedBuilder(SerializingHTMLBuilder):
    name = 'phpserialized'
    implementation = phpserialize
    out_suffix = '.file.phpdump'
    globalcontext_filename = 'globalcontext.phpdump'
    searchindex_filename = 'searchindex.phpdump'
```

implementation
A module that implements *dump()*, *load()*, *dumps()* and *loads()* functions that conform to the functions with the same names from the pickle module. Known modules implementing this interface are *simplejson*, *phpserialize*, *plistlib*, and others.

out_suffix
The suffix for all regular files.

globalcontext_filename
The filename for the file that contains the "global context". This is a dict with some general configuration values such as the name of the project.

searchindex_filename
The filename for the search index Sphinx generates.

See *Serialization builder details* for details about the output format.

New in version 0.5.

[174] https://pypi.org/project/phpserialize/

`class sphinxcontrib.serializinghtml.PickleHTMLBuilder`

This builder produces a directory with pickle files containing mostly HTML fragments and TOC information, for use of a web application (or custom postprocessing tool) that doesn't use the standard HTML templates.

See *Serialization builder details* for details about the output format.

`name = 'pickle'`

The builder's name, for the -b command line option.

The old name `web` still works as well.

`format = 'html'`

The builder's output format, or '' if no document output is produced.

`supported_image_types = ['image/svg+xml', 'image/png', 'image/gif', 'image/jpeg']`

The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

The file suffix is `.fpickle`. The global context is called `globalcontext.pickle`, the search index `searchindex.pickle`.

`class sphinxcontrib.serializinghtml.JSONHTMLBuilder`

This builder produces a directory with JSON files containing mostly HTML fragments and TOC information, for use of a web application (or custom postprocessing tool) that doesn't use the standard HTML templates.

See *Serialization builder details* for details about the output format.

`name = 'json'`

The builder's name, for the -b command line option.

`format = 'html'`

The builder's output format, or '' if no document output is produced.

`supported_image_types = ['image/svg+xml', 'image/png', 'image/gif', 'image/jpeg']`

The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

The file suffix is `.fjson`. The global context is called `globalcontext.json`, the search index `searchindex.json`.

New in version 0.5.

`class sphinx.builders.gettext.MessageCatalogBuilder`

This builder produces gettext-style message catalogs. Each top-level file or subdirectory grows a single `.pot` catalog template.

See the documentation on *Internationalization* for further reference.

`name = 'gettext'`

The builder's name, for the -b command line option.

`format = ''`

The builder's output format, or '' if no document output is produced.

`supported_image_types = []`

The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

New in version 1.1.

`class sphinx.builders.changes.ChangesBuilder`

This builder produces an HTML overview of all `versionadded`, `versionchanged` and `deprecated` directives for the current `version`. This is useful to generate a ChangeLog file, for example.

`name = 'changes'`

The builder's name, for the -b command line option.

format = ''
The builder's output format, or '' if no document output is produced.

supported_image_types = []
The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

class sphinx.builders.dummy.**DummyBuilder**
This builder produces no output. The input is only parsed and checked for consistency. This is useful for linting purposes.

name = 'dummy'
The builder's name, for the -b command line option.

supported_image_types = []
The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

New in version 1.4.

class sphinx.builders.linkcheck.**CheckExternalLinksBuilder**
This builder scans all documents for external links, tries to open them with `requests`, and writes an overview which ones are broken and redirected to standard output and to `output.txt` in the output directory.

name = 'linkcheck'
The builder's name, for the -b command line option.

format = ''
The builder's output format, or '' if no document output is produced.

supported_image_types = []
The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

Changed in version 1.5: Since Sphinx-1.5, the linkcheck builder comes to use requests module.

Changed in version 3.4: The linkcheck builder retries links when servers apply rate limits.

class sphinx.builders.xml.**XMLBuilder**
This builder produces Docutils-native XML files. The output can be transformed with standard XML tools such as XSLT processors into arbitrary final forms.

name = 'xml'
The builder's name, for the -b command line option.

format = 'xml'
The builder's output format, or '' if no document output is produced.

supported_image_types = []
The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

New in version 1.2.

class sphinx.builders.xml.**PseudoXMLBuilder**
This builder is used for debugging the Sphinx/Docutils "Reader to Transform to Writer" pipeline. It produces compact pretty-printed "pseudo-XML", files where nesting is indicated by indentation (no end-tags). External attributes for all elements are output, and internal attributes for any leftover "pending" elements are also given.

name = 'pseudoxml'
The builder's name, for the -b command line option.

format = 'pseudoxml'
The builder's output format, or '' if no document output is produced.

supported_image_types = []
The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

New in version 1.2.

Built-in Sphinx extensions that offer more builders are:

- *doctest*
- *coverage*

Serialization builder details

All serialization builders outputs one file per source file and a few special files. They also copy the reST source files in the directory `_sources` under the output directory.

The *PickleHTMLBuilder* is a builtin subclass that implements the pickle serialization interface.

The files per source file have the extensions of *out_suffix*, and are arranged in directories just as the source files are. They unserialize to a dictionary (or dictionary like structure) with these keys:

body The HTML "body" (that is, the HTML rendering of the source file), as rendered by the HTML translator.

title The title of the document, as HTML (may contain markup).

toc The table of contents for the file, rendered as an HTML `<ul>`.

display_toc A boolean that is `True` if the `toc` contains more than one entry.

current_page_name The document name of the current file.

parents, prev and next Information about related chapters in the TOC tree. Each relation is a dictionary with the keys `link` (HREF for the relation) and `title` (title of the related document, as HTML). `parents` is a list of relations, while `prev` and `next` are a single relation.

sourcename The name of the source file under `_sources`.

The special files are located in the root output directory. They are:

SerializingHTMLBuilder.globalcontext_filename A pickled dict with these keys:

project, copyright, release, version The same values as given in the configuration file.

style *html_style*.

last_updated Date of last build.

builder Name of the used builder, in the case of pickles this is always `'pickle'`.

titles A dictionary of all documents' titles, as HTML strings.

SerializingHTMLBuilder.searchindex_filename An index that can be used for searching the documentation. It is a pickled list with these entries:

- A list of indexed docnames.
- A list of document titles, as HTML strings, in the same order as the first list.
- A dict mapping word roots (processed by an English-language stemmer) to a list of integers, which are indices into the first list.

environment.pickle The build environment. This is always a pickle file, independent of the builder and a copy of the environment that was used when the builder was started.

Todo: Document common members.

Unlike the other pickle files this pickle file requires that the `sphinx` package is available on unpickling.

1.7 Extensions

Since many projects will need special features in their documentation, Sphinx allows adding "extensions" to the build process, each of which can modify almost any aspect of document processing.

This chapter describes the extensions bundled with Sphinx. For the API documentation on writing your own extension, refer to *Developing extensions for Sphinx*.

Built-in extensions

These extensions are built in and can be activated by respective entries in the `extensions` configuration value:

`sphinx.ext.autodoc` – Include documentation from docstrings

This extension can import the modules you are documenting, and pull in documentation from docstrings in a semi-automatic way.

Note: For Sphinx (actually, the Python interpreter that executes Sphinx) to find your module, it must be importable. That means that the module or the package must be in one of the directories on `sys.path`[175] – adapt your `sys.path`[176] in the configuration file accordingly.

Warning: `autodoc` **imports** the modules to be documented. If any modules have side effects on import, these will be executed by `autodoc` when `sphinx-build` is run.

If you document scripts (as opposed to library modules), make sure their main routine is protected by a `if __name__ == '__main__'` condition.

For this to work, the docstrings must of course be written in correct reStructuredText. You can then use all of the usual Sphinx markup in the docstrings, and it will end up correctly in the documentation. Together with hand-written documentation, this technique eases the pain of having to maintain two locations for documentation, while at the same time avoiding auto-generated-looking pure API documentation.

If you prefer NumPy[177] or Google[178] style docstrings over reStructuredText, you can also enable the `napoleon` extension. `napoleon` is a preprocessor that converts your docstrings to correct reStructuredText before `autodoc` processes them.

[175] https://docs.python.org/3/library/sys.html#sys.path
[176] https://docs.python.org/3/library/sys.html#sys.path
[177] https://github.com/numpy/numpy/blob/master/doc/HOWTO_DOCUMENT.rst.txt
[178] https://github.com/google/styleguide/blob/gh-pages/pyguide.md#38-comments-and-docstrings

Directives

autodoc provides several directives that are versions of the usual *py:module*, *py:class* and so forth. On parsing time, they import the corresponding module and extract the docstring of the given objects, inserting them into the page source under a suitable *py:module*, *py:class* etc. directive.

Note: Just as *py:class* respects the current *py:module*, *autoclass* will also do so. Likewise, *automethod* will respect the current *py:class*.

.. automodule::
.. autoclass::
.. autoexception::

Document a module, class or exception. All three directives will by default only insert the docstring of the object itself:

```
.. autoclass:: Noodle
```

will produce source like this:

```
.. class:: Noodle

   Noodle's docstring.
```

The "auto" directives can also contain content of their own, it will be inserted into the resulting non-auto directive source after the docstring (but before any automatic member documentation).

Therefore, you can also mix automatic and non-automatic member documentation, like so:

```
.. autoclass:: Noodle
   :members: eat, slurp

   .. method:: boil(time=10)

      Boil the noodle *time* minutes.
```

Options and advanced usage

- If you want to automatically document members, there's a `members` option:

  ```
  .. automodule:: noodle
     :members:
  ```

  will document all module members (recursively), and

  ```
  .. autoclass:: Noodle
     :members:
  ```

  will document all non-private member functions and properties (that is, those whose name doesn't start with `_`).

  For modules, `__all__` will be respected when looking for members unless you give the `ignore-module-all` flag option. Without `ignore-module-all`, the order of the members will also be the order in `__all__`.

  You can also give an explicit list of members; only these will then be documented:

```
.. autoclass:: Noodle
   :members: eat, slurp
```

- If you want to make the `members` option (or other options described below) the default, see *autodoc_default_options*.

Tip: You can use a negated form, `'no-flag'`, as an option of autodoc directive, to disable it temporarily. For example:

```
.. automodule:: foo
   :no-undoc-members:
```

Tip: You can use autodoc directive options to temporarily override or extend default options which takes list as an input. For example:

```
.. autoclass:: Noodle
   :members: eat
   :private-members: +_spicy, _garlickly
```

Changed in version 3.5: The default options can be overridden or extended temporarily.

- Members without docstrings will be left out, unless you give the `undoc-members` flag option:

```
.. automodule:: noodle
   :members:
   :undoc-members:
```

- "Private" members (that is, those named like `_private` or `__private`) will be included if the `private-members` flag option is given:

```
.. automodule:: noodle
   :members:
   :private-members:
```

It can also take an explicit list of member names to be documented as arguments:

```
.. automodule:: noodle
   :members:
   :private-members: _spicy, _garlickly
```

New in version 1.1.

Changed in version 3.2: The option can now take arguments.

- autodoc considers a member private if its docstring contains `:meta private:` in its *Info field lists*. For example:

```
def my_function(my_arg, my_other_arg):
    """blah blah blah

    :meta private:
    """
```

New in version 3.0.

- autodoc considers a member public if its docstring contains `:meta public:` in its *Info field lists*, even if it starts with an underscore. For example:

  ```
  def _my_function(my_arg, my_other_arg):
      """blah blah blah

      :meta public:
      """
  ```

  New in version 3.1.

- autodoc considers a variable member does not have any default value if its docstring contains `:meta hide-value:` in its *Info field lists*. Example:

  ```
  var1 = None  #: :meta hide-value:
  ```

  New in version 3.5.

- Python "special" members (that is, those named like `__special__`) will be included if the `special-members` flag option is given:

  ```
  .. autoclass:: my.Class
     :members:
     :private-members:
     :special-members:
  ```

  would document both "private" and "special" members of the class.

  New in version 1.1.

  Changed in version 1.2: The option can now take arguments, i.e. the special members to document.

- For classes and exceptions, members inherited from base classes will be left out when documenting all members, unless you give the `inherited-members` option, in addition to `members`:

  ```
  .. autoclass:: Noodle
     :members:
     :inherited-members:
  ```

  This can be combined with `undoc-members` to document *all* available members of the class or module.

  It can take an ancestor class not to document inherited members from it. By default, members of `object` class are not documented. To show them all, give `None` to the option.

  For example; If your class `Foo` is derived from `list` class and you don't want to document `list.__len__()`, you should specify a option `:inherited-members: list` to avoid special members of list class.

  Another example; If your class Foo has `__str__` special method and autodoc directive has both `inherited-members` and `special-members`, `__str__` will be documented as in the past, but other special method that are not implemented in your class `Foo`.

  Note: this will lead to markup errors if the inherited members come from a module whose docstrings are not reST formatted.

  New in version 0.3.

  Changed in version 3.0: It takes an ancestor class name as an argument.

- It's possible to override the signature for explicitly documented callable objects (functions, methods, classes) with the regular syntax that will override the signature gained from introspection:

```
.. autoclass:: Noodle(type)

   .. automethod:: eat(persona)
```

This is useful if the signature from the method is hidden by a decorator.

New in version 0.4.

- The *automodule*, *autoclass* and *autoexception* directives also support a flag option called `show-inheritance`. When given, a list of base classes will be inserted just below the class signature (when used with *automodule*, this will be inserted for every class that is documented in the module).

  New in version 0.4.

- All autodoc directives support the `noindex` flag option that has the same effect as for standard *py:function* etc. directives: no index entries are generated for the documented object (and all autodocumented members).

  New in version 0.4.

- *automodule* also recognizes the `synopsis`, `platform` and `deprecated` options that the standard *py:module* directive supports.

  New in version 0.5.

- *automodule* and *autoclass* also has an `member-order` option that can be used to override the global value of *autodoc_member_order* for one directive.

  New in version 0.6.

- The directives supporting member documentation also have a `exclude-members` option that can be used to exclude single member names from documentation, if all members are to be documented.

  New in version 0.6.

- In an *automodule* directive with the `members` option set, only module members whose `__module__` attribute is equal to the module name as given to `automodule` will be documented. This is to prevent documentation of imported classes or functions. Set the `imported-members` option if you want to prevent this behavior and document all available members. Note that attributes from imported modules will not be documented, because attribute documentation is discovered by parsing the source file of the current module.

  New in version 1.2.

- Add a list of modules in the *autodoc_mock_imports* to prevent import errors to halt the building process when some external dependencies are not importable at build time.

  New in version 1.3.

- As a hint to autodoc extension, you can put a `::` separator in between module name and object name to let autodoc know the correct module name if it is ambiguous.

  ```
  .. autoclass:: module.name::Noodle
  ```

.. autofunction::
.. autodecorator::
.. autodata::
.. automethod::
.. autoattribute::

These work exactly like *autoclass* etc., but do not offer the options used for automatic member documentation.

autodata and *autoattribute* support the `annotation` option. The option controls how the value of variable is shown. If specified without arguments, only the name of the variable will be printed, and its value is not shown:

```
.. autodata:: CD_DRIVE
   :annotation:
```

If the option specified with arguments, it is printed after the name as a value of the variable:

```
.. autodata:: CD_DRIVE
   :annotation: = your CD device name
```

By default, without `annotation` option, Sphinx tries to obtain the value of the variable and print it after the name.

The `no-value` option can be used instead of a blank `annotation` to show the type hint but not the value:

```
.. autodata:: CD_DRIVE
   :no-value:
```

If both the `annotation` and `no-value` options are used, `no-value` has no effect.

For module data members and class attributes, documentation can either be put into a comment with special formatting (using a `#:` to start the comment instead of just `#`), or in a docstring *after* the definition. Comments need to be either on a line of their own *before* the definition, or immediately after the assignment *on the same line*. The latter form is restricted to one line only.

This means that in the following class definition, all attributes can be autodocumented:

```
class Foo:
    """Docstring for class Foo."""

    #: Doc comment for class attribute Foo.bar.
    #: It can have multiple lines.
    bar = 1

    flox = 1.5   #: Doc comment for Foo.flox. One line only.

    baz = 2
    """Docstring for class attribute Foo.baz."""

    def __init__(self):
        #: Doc comment for instance attribute qux.
        self.qux = 3

        self.spam = 4
        """Docstring for instance attribute spam."""
```

Changed in version 0.6: *autodata* and *autoattribute* can now extract docstrings.

Changed in version 1.1: Comment docs are now allowed on the same line after an assignment.

Changed in version 1.2: *autodata* and *autoattribute* have an `annotation` option.

Changed in version 2.0: *autodecorator* added.

Changed in version 3.4: *autodata* and *autoattribute* now have a `no-value` option.

Note: If you document decorated functions or methods, keep in mind that autodoc retrieves its docstrings by importing the module and inspecting the `__doc__` attribute of the given function or method. That means that if a decorator replaces the decorated function with another, it must copy the original `__doc__` to the new function.

Configuration

There are also config values that you can set:

`autoclass_content`

This value selects what content will be inserted into the main body of an *autoclass* directive. The possible values are:

"class" Only the class' docstring is inserted. This is the default. You can still document `__init__` as a separate method using *automethod* or the `members` option to *autoclass*.

"both" Both the class' and the `__init__` method's docstring are concatenated and inserted.

"init" Only the `__init__` method's docstring is inserted.

New in version 0.3.

If the class has no `__init__` method or if the `__init__` method's docstring is empty, but the class has a `__new__` method's docstring, it is used instead.

New in version 1.4.

`autodoc_member_order`

This value selects if automatically documented members are sorted alphabetical (value `'alphabetical'`), by member type (value `'groupwise'`) or by source order (value `'bysource'`). The default is alphabetical.

Note that for source order, the module must be a Python module with the source code available.

New in version 0.6.

Changed in version 1.0: Support for `'bysource'`.

`autodoc_default_flags`

This value is a list of autodoc directive flags that should be automatically applied to all autodoc directives. The supported flags are `'members'`, `'undoc-members'`, `'private-members'`, `'special-members'`, `'inherited-members'`, `'show-inheritance'`, `'ignore-module-all'` and `'exclude-members'`.

New in version 1.0.

Deprecated since version 1.8: Integrated into *autodoc_default_options*.

`autodoc_default_options`

The default options for autodoc directives. They are applied to all autodoc directives automatically. It must be a dictionary which maps option names to the values. For example:

```
autodoc_default_options = {
    'members': 'var1, var2',
    'member-order': 'bysource',
    'special-members': '__init__',
    'undoc-members': True,
    'exclude-members': '__weakref__'
}
```

Setting `None` or `True` to the value is equivalent to giving only the option name to the directives.

The supported options are `'members'`, `'member-order'`, `'undoc-members'`, `'private-members'`, `'special-members'`, `'inherited-members'`, `'show-inheritance'`, `'ignore-module-all'`, `'imported-members'` and `'exclude-members'`.

New in version 1.8.

Changed in version 2.0: Accepts `True` as a value.

Changed in version 2.1: Added `'imported-members'`.

autodoc_docstring_signature

Functions imported from C modules cannot be introspected, and therefore the signature for such functions cannot be automatically determined. However, it is an often-used convention to put the signature into the first line of the function's docstring.

If this boolean value is set to `True` (which is the default), autodoc will look at the first line of the docstring for functions and methods, and if it looks like a signature, use the line as the signature and remove it from the docstring content.

If the signature line ends with backslash, autodoc considers the function has multiple signatures and look at the next line of the docstring. It is useful for overloaded function.

New in version 1.1.

Changed in version 3.1: Support overloaded signatures

autodoc_mock_imports

This value contains a list of modules to be mocked up. This is useful when some external dependencies are not met at build time and break the building process. You may only specify the root package of the dependencies themselves and omit the sub-modules:

```
autodoc_mock_imports = ["django"]
```

Will mock all imports under the `django` package.

New in version 1.3.

Changed in version 1.6: This config value only requires to declare the top-level modules that should be mocked.

autodoc_typehints

This value controls how to represents typehints. The setting takes the following values:

- `'signature'` – Show typehints as its signature (default)
- `'description'` – Show typehints as content of function or method
- `'none'` – Do not show typehints

New in version 2.1.

New in version 3.0: New option `'description'` is added.

autodoc_type_aliases

A dictionary for users defined type aliases[179] that maps a type name to the full-qualified object name. It is used to keep type aliases not evaluated in the document. Defaults to empty (`{}`).

The type aliases are only available if your program enables Postponed Evaluation of Annotations (PEP 563)[180] feature via `from __future__ import annotations`.

For example, there is code using a type alias:

```
from __future__ import annotations

AliasType = Union[List[Dict[Tuple[int, str], Set[int]]], Tuple[str, List[str]]]

def f() -> AliasType:
    ...
```

If `autodoc_type_aliases` is not set, autodoc will generate internal mark-up from this code as following:

```
.. py:function:: f() -> Union[List[Dict[Tuple[int, str], Set[int]]], Tuple[str,␣
↪List[str]]]

   ...
```

If you set `autodoc_type_aliases` as `{'AliasType': 'your.module.AliasType'}`, it generates the following document internally:

```
.. py:function:: f() -> your.module.AliasType:

   ...
```

New in version 3.3.

`autodoc_warningiserror`
This value controls the behavior of *`sphinx-build -W`* during importing modules. If `False` is given, autodoc forcedly suppresses the error if the imported module emits warnings. By default, `True`.

`autodoc_inherit_docstrings`
This value controls the docstrings inheritance. If set to True the docstring for classes or methods, if not explicitly set, is inherited from parents.

The default is `True`.

New in version 1.7.

`suppress_warnings`
`autodoc` supports to suppress warning messages via *`suppress_warnings`*. It allows following warnings types in addition:

- autodoc
- autodoc.import_object

Docstring preprocessing

autodoc provides the following additional events:

`autodoc-process-docstring`(*app*, *what*, *name*, *obj*, *options*, *lines*)
New in version 0.4.

Emitted when autodoc has read and processed a docstring. *lines* is a list of strings – the lines of the processed docstring – that the event handler can modify **in place** to change what Sphinx puts into the output.

Parameters

- **`app`** – the Sphinx application object
- **`what`** – the type of the object which the docstring belongs to (one of `"module"`, `"class"`, `"exception"`, `"function"`, `"method"`, `"attribute"`)
- **`name`** – the fully qualified name of the object
- **`obj`** – the object itself
- **`options`** – the options given to the directive: an object with attributes `inherited_members`, `undoc_members`, `show_inheritance` and `noindex` that are true if the flag option of same name was given to the auto directive
- **`lines`** – the lines of the docstring, see above

`autodoc-before-process-signature`(*app*, *obj*, *bound_method*)
New in version 2.4.

Emitted before autodoc formats a signature for an object. The event handler can modify an object to change its signature.

[179] https://www.python.org/dev/peps/pep-0563/
[180] https://mypy.readthedocs.io/en/latest/kinds_of_types.html#type-aliases

Parameters

- **app** – the Sphinx application object
- **obj** – the object itself
- **bound_method** – a boolean indicates an object is bound method or not

autodoc-process-signature(*app*, *what*, *name*, *obj*, *options*, *signature*, *return_annotation*)

New in version 0.5.

Emitted when autodoc has formatted a signature for an object. The event handler can return a new tuple `(signature, return_annotation)` to change what Sphinx puts into the output.

Parameters

- **app** – the Sphinx application object
- **what** – the type of the object which the docstring belongs to (one of `"module"`, `"class"`, `"exception"`, `"function"`, `"method"`, `"attribute"`)
- **name** – the fully qualified name of the object
- **obj** – the object itself
- **options** – the options given to the directive: an object with attributes `inherited_members`, `undoc_members`, `show_inheritance` and `noindex` that are true if the flag option of same name was given to the auto directive
- **signature** – function signature, as a string of the form `"(parameter_1, parameter_2)"`, or `None` if introspection didn't succeed and signature wasn't specified in the directive.
- **return_annotation** – function return annotation as a string of the form `" -> annotation"`, or `None` if there is no return annotation

The *sphinx.ext.autodoc* module provides factory functions for commonly needed docstring processing in event *autodoc-process-docstring*:

`sphinx.ext.autodoc.`**cut_lines**(*pre: int*[181], *post: int*[182] *= 0*, *what: Optional[str*[183]*] = None*) → Callable

Return a listener that removes the first *pre* and last *post* lines of every docstring. If *what* is a sequence of strings, only docstrings of a type in *what* will be processed.

Use like this (e.g. in the `setup()` function of `conf.py`):

```
from sphinx.ext.autodoc import cut_lines
app.connect('autodoc-process-docstring', cut_lines(4, what=['module']))
```

This can (and should) be used in place of `automodule_skip_lines`.

`sphinx.ext.autodoc.`**between**(*marker: str*[184], *what: Optional[Sequence[str*[185]*]] = None*, *keepempty: bool*[186] *= False*, *exclude: bool*[187] *= False*) → Callable

Return a listener that either keeps, or if *exclude* is True excludes, lines between lines that match the *marker* regular expression. If no line matches, the resulting docstring would be empty, so no change will be made unless *keepempty* is true.

If *what* is a sequence of strings, only docstrings of a type in *what* will be processed.

[181] https://docs.python.org/3/library/functions.html#int
[182] https://docs.python.org/3/library/functions.html#int
[183] https://docs.python.org/3/library/stdtypes.html#str
[184] https://docs.python.org/3/library/stdtypes.html#str
[185] https://docs.python.org/3/library/stdtypes.html#str
[186] https://docs.python.org/3/library/functions.html#bool
[187] https://docs.python.org/3/library/functions.html#bool

Skipping members

autodoc allows the user to define a custom method for determining whether a member should be included in the documentation by using the following event:

autodoc-skip-member(*app*, *what*, *name*, *obj*, *skip*, *options*)

New in version 0.5.

Emitted when autodoc has to decide whether a member should be included in the documentation. The member is excluded if a handler returns `True`. It is included if the handler returns `False`.

If more than one enabled extension handles the `autodoc-skip-member` event, autodoc will use the first non-`None` value returned by a handler. Handlers should return `None` to fall back to the skipping behavior of autodoc and other enabled extensions.

Parameters

- **app** – the Sphinx application object
- **what** – the type of the object which the docstring belongs to (one of `"module"`, `"class"`, `"exception"`, `"function"`, `"method"`, `"attribute"`)
- **name** – the fully qualified name of the object
- **obj** – the object itself
- **skip** – a boolean indicating if autodoc will skip this member if the user handler does not override the decision
- **options** – the options given to the directive: an object with attributes `inherited_members`, `undoc_members`, `show_inheritance` and `noindex` that are true if the flag option of same name was given to the auto directive

`sphinx.ext.autosectionlabel` – Allow reference sections using its title

New in version 1.4.

This extension allows you to refer sections its title. This affects to the reference role (*ref*).

For example:

```
A Plain Title
-------------

This is the text of the section.

It refers to the section title, see :ref:`A Plain Title`.
```

Internally, this extension generates the labels for each section. If same section names are used in whole of document, any one is used for a target by default. The `autosectionlabel_prefix_document` configuration variable can be used to make headings which appear multiple times but in different documents unique.

Configuration

autosectionlabel_prefix_document
True to prefix each section label with the name of the document it is in, followed by a colon. For example, `index:Introduction` for a section called `Introduction` that appears in document `index.rst`. Useful for avoiding ambiguity when the same section heading appears in different documents.

autosectionlabel_maxdepth
If set, autosectionlabel chooses the sections for labeling by its depth. For example, when set 1 to `autosectionlabel_maxdepth`, labels are generated only for top level sections, and deeper sections are not labeled. It defaults to `None` (disabled).

`sphinx.ext.autosummary` – Generate autodoc summaries

New in version 0.6.

This extension generates function/method/attribute summary lists, similar to those output e.g. by Epydoc and other API doc generation tools. This is especially useful when your docstrings are long and detailed, and putting each one of them on a separate page makes them easier to read.

The *`sphinx.ext.autosummary`* extension does this in two parts:

1. There is an *`autosummary`* directive for generating summary listings that contain links to the documented items, and short summary blurbs extracted from their docstrings.
2. Optionally, the convenience script **`sphinx-autogen`** or the new *`autosummary_generate`* config value can be used to generate short "stub" files for the entries listed in the *`autosummary`* directives. These files by default contain only the corresponding *`sphinx.ext.autodoc`* directive, but can be customized with templates.

.. autosummary::
Insert a table that contains links to documented items, and a short summary blurb (the first sentence of the docstring) for each of them.

The *`autosummary`* directive can also optionally serve as a *`toctree`* entry for the included items. Optionally, stub `.rst` files for these items can also be automatically generated when *`autosummary_generate`* is *True*.

For example,

```
.. currentmodule:: sphinx

.. autosummary::

   environment.BuildEnvironment
   util.relative_uri
```

produces a table like this:

`environment.BuildEnvironment`(app)	The environment in which the ReST files are translated.
`util.relative_uri`(base, to)	Return a relative URL from `base` to `to`.

Autosummary preprocesses the docstrings and signatures with the same *`autodoc-process-docstring`* and *`autodoc-process-signature`* hooks as *`autodoc`*.

Options

- If you want the *`autosummary`* table to also serve as a *`toctree`* entry, use the `toctree` option, for example:

```
.. autosummary::
   :toctree: DIRNAME

   sphinx.environment.BuildEnvironment
   sphinx.util.relative_uri
```

The `toctree` option also signals to the **`sphinx-autogen`** script that stub pages should be generated for the entries listed in this directive. The option accepts a directory name as an argument; **`sphinx-autogen`** will by default place its output in this directory. If no argument is given, output is placed in the same directory as the file that contains the directive.

You can also use `caption` option to give a caption to the toctree.

New in version 3.1: caption option added.

- If you don't want the *`autosummary`* to show function signatures in the listing, include the `nosignatures` option:

```
.. autosummary::
   :nosignatures:

   sphinx.environment.BuildEnvironment
   sphinx.util.relative_uri
```

- You can specify a custom template with the `template` option. For example,

```
.. autosummary::
   :template: mytemplate.rst

   sphinx.environment.BuildEnvironment
```

would use the template `mytemplate.rst` in your *`templates_path`* to generate the pages for all entries listed. See *Customizing templates* below.

New in version 1.0.

- You can specify the `recursive` option to generate documents for modules and sub-packages recursively. It defaults to disabled. For example,

```
.. autosummary::
   :recursive:

   sphinx.environment.BuildEnvironment
```

New in version 3.1.

`sphinx-autogen` – generate autodoc stub pages

The **`sphinx-autogen`** script can be used to conveniently generate stub documentation pages for items included in *`autosummary`* listings.

For example, the command

```
$ sphinx-autogen -o generated *.rst
```

will read all *`autosummary`* tables in the `*.rst` files that have the `:toctree:` option set, and output corresponding stub pages in directory `generated` for all documented items. The generated pages by default contain text of the form:

```
sphinx.util.relative_uri
========================

.. autofunction:: sphinx.util.relative_uri
```

If the `-o` option is not given, the script will place the output files in the directories specified in the `:toctree:` options.

For more information, refer to the *sphinx-autogen documentation*

Generating stub pages automatically

If you do not want to create stub pages with **sphinx-autogen**, you can also use these config values:

autosummary_context
A dictionary of values to pass into the template engine's context for autosummary stubs files.

New in version 3.1.

autosummary_generate
Boolean indicating whether to scan all found documents for autosummary directives, and to generate stub pages for each. It is disabled by default.

Can also be a list of documents for which stub pages should be generated.

The new files will be placed in the directories specified in the `:toctree:` options of the directives.

Changed in version 2.3: Emits *autodoc-skip-member* event as *autodoc* does.

autosummary_generate_overwrite
If true, autosummary overwrites existing files by generated stub pages. Defaults to true (enabled).

New in version 3.0.

autosummary_mock_imports
This value contains a list of modules to be mocked up. See *autodoc_mock_imports* for more details. It defaults to *autodoc_mock_imports*.

New in version 2.0.

autosummary_imported_members
A boolean flag indicating whether to document classes and functions imported in modules. Default is `False`

New in version 2.1.

autosummary_filename_map
A dict mapping object names to filenames. This is necessary to avoid filename conflicts where multiple objects have names that are indistinguishable when case is ignored, on file systems where filenames are case-insensitive.

New in version 3.2.

Customizing templates

New in version 1.0.

You can customize the stub page templates, in a similar way as the HTML Jinja templates, see *Templating*. (*TemplateBridge* is not supported.)

Note: If you find yourself spending much time tailoring the stub templates, this may indicate that it's a better idea to write custom narrative documentation instead.

Autosummary uses the following Jinja template files:

- `autosummary/base.rst` – fallback template
- `autosummary/module.rst` – template for modules
- `autosummary/class.rst` – template for classes
- `autosummary/function.rst` – template for functions
- `autosummary/attribute.rst` – template for class attributes
- `autosummary/method.rst` – template for class methods

The following variables available in the templates:

`name`
  Name of the documented object, excluding the module and class parts.

`objname`
  Name of the documented object, excluding the module parts.

`fullname`
  Full name of the documented object, including module and class parts.

`module`
  Name of the module the documented object belongs to.

`class`
  Name of the class the documented object belongs to. Only available for methods and attributes.

`underline`
  A string containing `len(full_name) * '='`. Use the `underline` filter instead.

`members`
  List containing names of all members of the module or class. Only available for modules and classes.

`inherited_members`
  List containing names of all inherited members of class. Only available for classes.

  New in version 1.8.0.

`functions`
  List containing names of "public" functions in the module. Here, "public" here means that the name does not start with an underscore. Only available for modules.

`classes`
  List containing names of "public" classes in the module. Only available for modules.

`exceptions`
  List containing names of "public" exceptions in the module. Only available for modules.

`methods`
  List containing names of "public" methods in the class. Only available for classes.

`attributes`
  List containing names of "public" attributes in the class/module. Only available for classes and modules.

    Changed in version 3.1: Attributes of modules are supported.

`modules`
  List containing names of "public" modules in the package. Only available for modules that are packages and the `recursive` option is on.

  New in version 3.1.

Additionally, the following filters are available

escape(*s*)
Escape any special characters in the text to be used in formatting RST contexts. For instance, this prevents asterisks making things bold. This replaces the builtin Jinja escape filter[188] that does html-escaping.

underline(*s*, *line='='*)
Add a title underline to a piece of text.

For instance, `{{ fullname | escape | underline }}` should be used to produce the title of a page.

Note: You can use the *autosummary* directive in the stub pages. Stub pages are generated also based on these directives.

`sphinx.ext.coverage` – Collect doc coverage stats

This extension features one additional builder, the *CoverageBuilder*.

class `sphinx.ext.coverage.`**CoverageBuilder**
To use this builder, activate the coverage extension in your configuration file and give `-b coverage` on the command line.

Todo: Write this section.

Several configuration values can be used to specify what the builder should check:

coverage_ignore_modules

coverage_ignore_functions

coverage_ignore_classes

coverage_ignore_pyobjects
List of Python regular expressions[189].

If any of these regular expressions matches any part of the full import path of a Python object, that Python object is excluded from the documentation coverage report.

New in version 2.1.

coverage_c_path

coverage_c_regexes

coverage_ignore_c_items

coverage_write_headline
Set to `False` to not write headlines.

New in version 1.1.

coverage_skip_undoc_in_source
Skip objects that are not documented in the source with a docstring. `False` by default.

New in version 1.1.

coverage_show_missing_items
Print objects that are missing to standard output also. `False` by default.

New in version 3.1.

[188] http://jinja.pocoo.org/docs/2.9/templates/#escape
[189] https://docs.python.org/library/re

`sphinx.ext.doctest` – Test snippets in the documentation

It is often helpful to include snippets of code in your documentation and demonstrate the results of executing them. But it is important to ensure that the documentation stays up-to-date with the code.

This extension allows you to test such code snippets in the documentation in a natural way. If you mark the code blocks as shown here, the `doctest` builder will collect them and run them as doctest tests.

Within each document, you can assign each snippet to a *group*. Each group consists of:

- zero or more *setup code* blocks (e.g. importing the module to test)
- one or more *test* blocks

When building the docs with the `doctest` builder, groups are collected for each document and run one after the other, first executing setup code blocks, then the test blocks in the order they appear in the file.

There are two kinds of test blocks:

- *doctest-style* blocks mimic interactive sessions by interleaving Python code (including the interpreter prompt) and output.
- *code-output-style* blocks consist of an ordinary piece of Python code, and optionally, a piece of output for that code.

Directives

The *group* argument below is interpreted as follows: if it is empty, the block is assigned to the group named `default`. If it is `*`, the block is assigned to all groups (including the `default` group). Otherwise, it must be a comma-separated list of group names.

.. testsetup:: `[group]`
A setup code block. This code is not shown in the output for other builders, but executed before the doctests of the group(s) it belongs to.

.. testcleanup:: `[group]`
A cleanup code block. This code is not shown in the output for other builders, but executed after the doctests of the group(s) it belongs to.

New in version 1.1.

.. doctest:: `[group]`
A doctest-style code block. You can use standard `doctest`[190] flags for controlling how actual output is compared with what you give as output. The default set of flags is specified by the *`doctest_default_flags`* configuration variable.

This directive supports five options:

- `hide`, a flag option, hides the doctest block in other builders. By default it is shown as a highlighted doctest block.
- `options`, a string option, can be used to give a comma-separated list of doctest flags that apply to each example in the tests. (You still can give explicit flags per example, with doctest comments, but they will show up in other builders too.)
- `pyversion`, a string option, can be used to specify the required Python version for the example to be tested. For instance, in the following case the example will be tested only for Python versions greater than 3.3:

```
.. doctest::
   :pyversion: > 3.3
```

The following operands are supported:

  - ~=: Compatible release clause
  - ==: Version matching clause
  - !=: Version exclusion clause
  - <=, >=: Inclusive ordered comparison clause
  - <, >: Exclusive ordered comparison clause
  - ===: Arbitrary equality clause.

  `pyversion` option is followed PEP-440: Version Specifiers[191].

  New in version 1.6.

  Changed in version 1.7: Supported PEP-440 operands and notations

- `trim-doctest-flags` and `no-trim-doctest-flags`, a flag option, doctest flags (comments looking like `# doctest:  FLAG, ...`) at the ends of lines and `<BLANKLINE>` markers are removed (or not removed) individually. Default is `trim-doctest-flags`.

Note that like with standard doctests, you have to use `<BLANKLINE>` to signal a blank line in the expected output. The `<BLANKLINE>` is removed when building presentation output (HTML, LaTeX etc.).

Also, you can give inline doctest options, like in doctest:

```
>>> datetime.date.now()   # doctest: +SKIP
datetime.date(2008, 1, 1)
```

They will be respected when the test is run, but stripped from presentation output.

.. testcode:: `[group]`

A code block for a code-output-style test.

This directive supports three options:

- `hide`, a flag option, hides the code block in other builders. By default it is shown as a highlighted code block.
- `trim-doctest-flags` and `no-trim-doctest-flags`, a flag option, doctest flags (comments looking like `# doctest:  FLAG, ...`) at the ends of lines and `<BLANKLINE>` markers are removed (or not removed) individually. Default is `trim-doctest-flags`.

Note: Code in a `testcode` block is always executed all at once, no matter how many statements it contains. Therefore, output will *not* be generated for bare expressions – use `print`. Example:

```
.. testcode::

   1+1         # this will give no output!
   print(2+2)  # this will give output

.. testoutput::

   4
```

Also, please be aware that since the doctest module does not support mixing regular output and an exception message in the same snippet, this applies to testcode/testoutput as well.

[190] https://docs.python.org/3/library/doctest.html#module-doctest
[191] https://www.python.org/dev/peps/pep-0440/#version-specifiers

`.. testoutput:: [group]`

The corresponding output, or the exception message, for the last *testcode* block.

This directive supports four options:

- `hide`, a flag option, hides the output block in other builders. By default it is shown as a literal block without highlighting.
- `options`, a string option, can be used to give doctest flags (comma-separated) just like in normal doctest blocks.
- `trim-doctest-flags` and `no-trim-doctest-flags`, a flag option, doctest flags (comments looking like `# doctest:  FLAG, ...`) at the ends of lines and `<BLANKLINE>` markers are removed (or not removed) individually. Default is `trim-doctest-flags`.

Example:

```
.. testcode::

   print('Output     text.')

.. testoutput::
   :hide:
   :options: -ELLIPSIS, +NORMALIZE_WHITESPACE

   Output text.
```

The following is an example for the usage of the directives. The test via *doctest* and the test via *testcode* and *testoutput* are equivalent.

```
The parrot module
=================

.. testsetup:: *

   import parrot

The parrot module is a module about parrots.

Doctest example:

.. doctest::

   >>> parrot.voom(3000)
   This parrot wouldn't voom if you put 3000 volts through it!

Test-Output example:

.. testcode::

   parrot.voom(3000)

This would output:

.. testoutput::

   This parrot wouldn't voom if you put 3000 volts through it!
```

Skipping tests conditionally

skipif, a string option, can be used to skip directives conditionally. This may be useful e.g. when a different set of tests should be run depending on the environment (hardware, network/VPN, optional dependencies or different versions of dependencies). The skipif option is supported by all of the doctest directives. Below are typical use cases for skipif when used for different directives:

- *testsetup* and *testcleanup*
  - conditionally skip test setup and/or cleanup
  - customize setup/cleanup code per environment
- *doctest*
  - conditionally skip both a test and its output verification
- *testcode*
  - conditionally skip a test
  - customize test code per environment
- *testoutput*
  - conditionally skip output assertion for a skipped test
  - expect different output depending on the environment

The value of the skipif option is evaluated as a Python expression. If the result is a true value, the directive is omitted from the test run just as if it wasn't present in the file at all.

Instead of repeating an expression, the *doctest_global_setup* configuration option can be used to assign it to a variable which can then be used instead.

Here's an example which skips some tests if Pandas is not installed:

Listing 1: conf.py

```
extensions = ['sphinx.ext.doctest']
doctest_global_setup = '''
try:
    import pandas as pd
except ImportError:
    pd = None
'''
```

Listing 2: contents.rst

```
.. testsetup::
   :skipif: pd is None

   data = pd.Series([42])

.. doctest::
   :skipif: pd is None

   >>> data.iloc[0]
   42

.. testcode::
```

(continues on next page)

(continued from previous page)

```
   :skipif: pd is None

   print(data.iloc[-1])

.. testoutput::
   :skipif: pd is None

   42
```

Configuration

The doctest extension uses the following configuration values:

doctest_default_flags
By default, these options are enabled:

- ELLIPSIS, allowing you to put ellipses in the expected output that match anything in the actual output;
- IGNORE_EXCEPTION_DETAIL, causing everything following the leftmost colon and any module information in the exception name to be ignored;
- DONT_ACCEPT_TRUE_FOR_1, rejecting "True" in the output where "1" is given – the default behavior of accepting this substitution is a relic of pre-Python 2.2 times.

New in version 1.5.

doctest_path
A list of directories that will be added to sys.path[192] when the doctest builder is used. (Make sure it contains absolute paths.)

doctest_global_setup
Python code that is treated like it were put in a testsetup directive for *every* file that is tested, and for every group. You can use this to e.g. import modules you will always need in your doctests.

New in version 0.6.

doctest_global_cleanup
Python code that is treated like it were put in a testcleanup directive for *every* file that is tested, and for every group. You can use this to e.g. remove any temporary files that the tests leave behind.

New in version 1.1.

doctest_test_doctest_blocks
If this is a nonempty string (the default is 'default'), standard reST doctest blocks will be tested too. They will be assigned to the group name given.

reST doctest blocks are simply doctests put into a paragraph of their own, like so:

```
Some documentation text.

>>> print(1)
1

Some more documentation text.
```

(Note that no special :: is used to introduce a doctest block; docutils recognizes them from the leading >>>. Also, no additional indentation is used, though it doesn't hurt.)

[192] https://docs.python.org/3/library/sys.html#sys.path

If this value is left at its default value, the above snippet is interpreted by the doctest builder exactly like the following:

```
Some documentation text.

.. doctest::

   >>> print(1)
   1

Some more documentation text.
```

This feature makes it easy for you to test doctests in docstrings included with the *autodoc* extension without marking them up with a special directive.

Note though that you can't have blank lines in reST doctest blocks. They will be interpreted as one block ending and another one starting. Also, removal of `<BLANKLINE>` and `# doctest:` options only works in *doctest* blocks, though you may set *trim_doctest_flags* to achieve that in all code blocks with Python console content.

sphinx.ext.duration – Measure durations of Sphinx processing

New in version 2.4.

This extension measures durations of Sphinx processing and show its result at end of the build. It is useful for inspecting what document is slowly built.

sphinx.ext.extlinks – Markup to shorten external links

Module author: Georg Brandl

New in version 1.0.

This extension is meant to help with the common pattern of having many external links that point to URLs on one and the same site, e.g. links to bug trackers, version control web interfaces, or simply subpages in other websites. It does so by providing aliases to base URLs, so that you only need to give the subpage name when creating a link.

Let's assume that you want to include many links to issues at the Sphinx tracker, at `https://github.com/sphinx-doc/sphinx/issues/num`. Typing this URL again and again is tedious, so you can use *extlinks* to avoid repeating yourself.

The extension adds a config value:

extlinks

This config value must be a dictionary of external sites, mapping unique short alias names to a base URL and a *prefix*. For example, to create an alias for the above mentioned issues, you would add

```
extlinks = {'issue': ('https://github.com/sphinx-doc/sphinx/issues/%s',
                      'issue ')}
```

Now, you can use the alias name as a new role, e.g. `:issue:`123``. This then inserts a link to https://github.com/sphinx-doc/sphinx/issues/123. As you can see, the target given in the role is substituted in the base URL in the place of `%s`.

The link *caption* depends on the second item in the tuple, the *prefix*:

- If the prefix is `None`, the link caption is the full URL.
- If the prefix is the empty string, the link caption is the partial URL given in the role content (`123` in this case.)

- If the prefix is a non-empty string, the link caption is the partial URL, prepended by the prefix – in the above example, the link caption would be `issue 123`.

You can also use the usual "explicit title" syntax supported by other roles that generate links, i.e. ``:issue:`this issue <123>` ``. In this case, the *prefix* is not relevant.

Note: Since links are generated from the role in the reading stage, they appear as ordinary links to e.g. the `linkcheck` builder.

`sphinx.ext.githubpages` – Publish HTML docs in GitHub Pages

New in version 1.4.

Changed in version 2.0: Support `CNAME` file

This extension creates `.nojekyll` file on generated HTML directory to publish the document on GitHub Pages.

It also creates a `CNAME` file for custom domains when *`html_baseurl`* set.

`sphinx.ext.graphviz` – Add Graphviz graphs

New in version 0.6.

This extension allows you to embed Graphviz[193] graphs in your documents.

It adds these directives:

.. graphviz::

Directive to embed graphviz code. The input code for `dot` is given as the content. For example:

```
.. graphviz::

   digraph foo {
      "bar" -> "baz";
   }
```

In HTML output, the code will be rendered to a PNG or SVG image (see *`graphviz_output_format`*). In LaTeX output, the code will be rendered to an embeddable PDF file.

You can also embed external dot files, by giving the file name as an argument to *`graphviz`* and no additional content:

```
.. graphviz:: external.dot
```

As for all file references in Sphinx, if the filename is absolute, it is taken as relative to the source directory.

Changed in version 1.1: Added support for external files.

[193] https://graphviz.org/

options

`:alt: alternate text (text)`
The alternate text of the graph. By default, the graph code is used to the alternate text.

New in version 1.0.

`:align: alignment of the graph (left, center or right)`
The horizontal alignment of the graph.

New in version 1.5.

`:caption: caption of the graph (text)`
The caption of the graph.

New in version 1.1.

`:layout: layout type of the graph (text)`
The layout of the graph (ex. `dot`, `neato` and so on). A path to the graphviz commands are also allowed. By default, *`graphviz_dot`* is used.

New in version 1.4.

Changed in version 2.2: Renamed from `graphviz_dot`

`:name: label (text)`
The label of the graph.

New in version 1.6.

`:class: class names (a list of class names separeted by spaces)`
The class name of the graph.

New in version 2.4.

`.. graph::`
Directive for embedding a single undirected graph. The name is given as a directive argument, the contents of the graph are the directive content. This is a convenience directive to generate `graph <name> { <content> }`.

For example:

```
.. graph:: foo

   "bar" -- "baz";
```

Note: The graph name is passed unchanged to Graphviz. If it contains non-alphanumeric characters (e.g. a dash), you will have to double-quote it.

options

Same as *`graphviz`*.

`:alt: alternate text (text)`
New in version 1.0.

`:align: alignment of the graph (left, center or right)`
New in version 1.5.

`:caption: caption of the graph (text)`
New in version 1.1.

:layout: layout type of the graph (text)
New in version 1.4.

Changed in version 2.2: Renamed from `graphviz_dot`

:name: label (text)
New in version 1.6.

:class: class names (a list of class names separeted by spaces)
The class name of the graph.

New in version 2.4.

.. digraph::
Directive for embedding a single directed graph. The name is given as a directive argument, the contents of the graph are the directive content. This is a convenience directive to generate `digraph <name> { <content> }`.

For example:

```
.. digraph:: foo

   "bar" -> "baz" -> "quux";
```

options

Same as *graphviz*.

:alt: alternate text (text)
New in version 1.0.

:align: alignment of the graph (left, center or right)
New in version 1.5.

:caption: caption of the graph (text)
New in version 1.1.

:layout: layout type of the graph (text)
New in version 1.4.

Changed in version 2.2: Renamed from `graphviz_dot`

:name: label (text)
New in version 1.6.

:class: class names (a list of class names separeted by spaces)
The class name of the graph.

New in version 2.4.

There are also these config values:

graphviz_dot
The command name with which to invoke `dot`. The default is `'dot'`; you may need to set this to a full path if `dot` is not in the executable search path.

Since this setting is not portable from system to system, it is normally not useful to set it in `conf.py`; rather, giving it on the **sphinx-build** command line via the *-D* option should be preferable, like this:

```
sphinx-build -b html -D graphviz_dot=C:\graphviz\bin\dot.exe . _build/html
```

graphviz_dot_args
Additional command-line arguments to give to dot, as a list. The default is an empty list. This is the right place to set global graph, node or edge attributes via dot's `-G`, `-N` and `-E` options.

graphviz_output_format

The output format for Graphviz when building HTML files. This must be either `'png'` or `'svg'`; the default is `'png'`. If `'svg'` is used, in order to make the URL links work properly, an appropriate `target` attribute must be set, such as `"_top"` and `"_blank"`. For example, the link in the following graph should work in the svg output:

```
.. graphviz::

     digraph example {
         a [label="sphinx", href="http://sphinx-doc.org", target="_top"];
         b [label="other"];
         a -> b;
     }
```

New in version 1.0: Previously, output always was PNG.

`sphinx.ext.ifconfig` – Include content based on configuration

This extension is quite simple, and features only one directive:

Warning: This directive is designed to control only content of document. It could not control sections, labels and so on.

.. ifconfig::

Include content of the directive only if the Python expression given as an argument is `True`, evaluated in the namespace of the project's configuration (that is, all registered variables from `conf.py` are available).

For example, one could write

```
.. ifconfig:: releaselevel in ('alpha', 'beta', 'rc')

   This stuff is only included in the built docs for unstable versions.
```

To make a custom config value known to Sphinx, use *add_config_value()* in the setup function in `conf.py`, e.g.:

```
def setup(app):
    app.add_config_value('releaselevel', '', 'env')
```

The second argument is the default value, the third should always be `'env'` for such values (it selects if Sphinx re-reads the documents if the value changes).

`sphinx.ext.imgconverter` – A reference image converter using Imagemagick

New in version 1.6.

This extension converts images in your document to appropriate format for builders. For example, it allows you to use SVG images with LaTeX builder. As a result, you don't mind what image format the builder supports.

Internally, this extension uses Imagemagick[194] to convert images.

[194] https://www.imagemagick.org/script/index.php

Note: Imagemagick rasterizes a SVG image on conversion. As a result, the image becomes not scalable. To avoid that, please use other image converters like sphinxcontrib-svg2pdfconverter[195] (which uses Inkscape or `rsvg-convert`).

Configuration

`image_converter`

A path to **`convert`** command. By default, the imgconverter uses the command from search paths.

On windows platform, **`magick`** command is used by default.

Changed in version 3.1: Use **`magick`** command by default on windows

`image_converter_args`

Additional command-line arguments to give to **`convert`**, as a list. The default is an empty list `[]`.

On windows platform, it defaults to `["convert"]`.

Changed in version 3.1: Use `["convert"]` by default on windows

`sphinx.ext.inheritance_diagram` – Include inheritance diagrams

New in version 0.6.

This extension allows you to include inheritance diagrams, rendered via the *`Graphviz extension`*.

It adds this directive:

`.. inheritance-diagram::`

This directive has one or more arguments, each giving a module or class name. Class names can be unqualified; in that case they are taken to exist in the currently described module (see *`py:module`*).

For each given class, and each class in each given module, the base classes are determined. Then, from all classes and their base classes, a graph is generated which is then rendered via the graphviz extension to a directed graph.

This directive supports an option called `parts` that, if given, must be an integer, advising the directive to keep that many dot-separated parts in the displayed names (from right to left). For example, `parts=1` will only display class names, without the names of the modules that contain them.

Changed in version 2.0: The value of for `parts` can also be negative, indicating how many parts to drop from the left. For example, if all your class names start with `lib.`, you can give `:parts:  -1` to remove that prefix from the displayed node names.

The directive also supports a `private-bases` flag option; if given, private base classes (those whose name starts with `_`) will be included.

You can use `caption` option to give a caption to the diagram.

Changed in version 1.1: Added `private-bases` option; previously, all bases were always included.

Changed in version 1.5: Added `caption` option

It also supports a `top-classes` option which requires one or more class names separated by comma. If specified inheritance traversal will stop at the specified class names. Given the following Python module:

```
"""
       A
      / \
     B   C
```

(continues on next page)

[195] https://github.com/missinglinkelectronics/sphinxcontrib-svg2pdfconverter

(continued from previous page)

```
    / \ / \
   E   D   F
"""

class A:
    pass

class B(A):
    pass

class C(A):
    pass

class D(B, C):
    pass

class E(B):
    pass

class F(C):
    pass
```

If you have specified a module in the inheritance diagram like this:

```
.. inheritance-diagram:: dummy.test
   :top-classes: dummy.test.B, dummy.test.C
```

any base classes which are ancestors to `top-classes` and are also defined in the same module will be rendered as stand alone nodes. In this example class A will be rendered as stand alone node in the graph. This is a known issue due to how this extension works internally.

If you don't want class A (or any other ancestors) to be visible then specify only the classes you would like to generate the diagram for like this:

```
.. inheritance-diagram:: dummy.test.D dummy.test.E dummy.test.F
   :top-classes: dummy.test.B, dummy.test.C
```

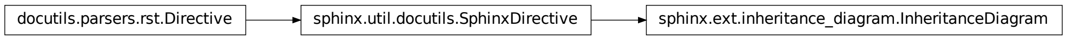
Changed in version 1.7: Added `top-classes` option to limit the scope of inheritance graphs.

Examples

The following are different inheritance diagrams for the internal `InheritanceDiagram` class that implements the directive.

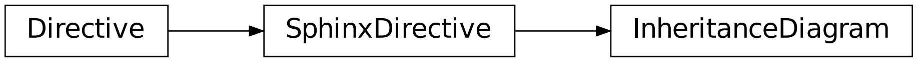
With full names:

```
.. inheritance-diagram:: sphinx.ext.inheritance_diagram.InheritanceDiagram
```


docutils.parsers.rst.Directive
sphinx.util.docutils.SphinxDirective
sphinx.ext.inheritance_diagram.InheritanceDiagram

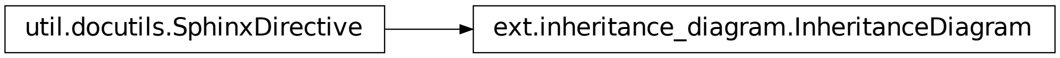
Showing class names only:

```
.. inheritance-diagram:: sphinx.ext.inheritance_diagram.InheritanceDiagram
   :parts: 1
```



Stopping the diagram at *sphinx.util.docutils.SphinxDirective* (the highest superclass still part of Sphinx), and dropping the common left-most part (`sphinx`) from all names:

```
.. inheritance-diagram:: sphinx.ext.inheritance_diagram.InheritanceDiagram
   :top-classes: sphinx.util.docutils.SphinxDirective
   :parts: -1
```


util.docutils.SphinxDirective
ext.inheritance_diagram.InheritanceDiagram

Configuration

inheritance_graph_attrs

A dictionary of graphviz graph attributes for inheritance diagrams.

For example:

```
inheritance_graph_attrs = dict(rankdir="LR", size='"6.0, 8.0"',
                               fontsize=14, ratio='compress')
```

inheritance_node_attrs

A dictionary of graphviz node attributes for inheritance diagrams.

For example:

```
inheritance_node_attrs = dict(shape='ellipse', fontsize=14, height=0.75,
                              color='dodgerblue1', style='filled')
```

inheritance_edge_attrs
A dictionary of graphviz edge attributes for inheritance diagrams.

inheritance_alias
Allows mapping the full qualified name of the class to custom values (useful when exposing the underlying path of a class is not desirable, e.g. it's a private class and should not be instantiated by the user).

For example:

```
inheritance_alias = {'_pytest.Magic': 'pytest.Magic'}
```

`sphinx.ext.intersphinx` – Link to other projects' documentation

New in version 0.5.

This extension can generate automatic links to the documentation of objects in other projects.

Usage is simple: whenever Sphinx encounters a cross-reference that has no matching target in the current documentation set, it looks for targets in the documentation sets configured in *intersphinx_mapping*. A reference like `:py:class:`zipfile.ZipFile`` can then link to the Python documentation for the ZipFile class, without you having to specify where it is located exactly.

When using the "new" format (see below), you can even force lookup in a foreign set by prefixing the link target appropriately. A link like `:ref:`comparison manual <python:comparisons>`` will then link to the label "comparisons" in the doc set "python", if it exists.

Behind the scenes, this works as follows:

- Each Sphinx HTML build creates a file named `objects.inv` that contains a mapping from object names to URIs relative to the HTML set's root.
- Projects using the Intersphinx extension can specify the location of such mapping files in the *intersphinx_mapping* config value. The mapping will then be used to resolve otherwise missing references to objects into links to the other documentation.
- By default, the mapping file is assumed to be at the same location as the rest of the documentation; however, the location of the mapping file can also be specified individually, e.g. if the docs should be buildable without Internet access.

Configuration

To use Intersphinx linking, add `'sphinx.ext.intersphinx'` to your *extensions* config value, and use these config values to activate linking:

intersphinx_mapping
This config value contains the locations and names of other projects that should be linked to in this documentation.

Relative local paths for target locations are taken as relative to the base of the built documentation, while relative local paths for inventory locations are taken as relative to the source directory.

When fetching remote inventory files, proxy settings will be read from the `$HTTP_PROXY` environment variable.

Old format for this config value

This is the format used before Sphinx 1.0. It is still recognized.

A dictionary mapping URIs to either `None` or an URI. The keys are the base URI of the foreign Sphinx documentation sets and can be local paths or HTTP URIs. The values indicate where the inventory file can be found: they can be `None` (at the same location as the base URI) or another local or HTTP URI.

New format for this config value

New in version 1.0.

A dictionary mapping unique identifiers to a tuple `(target, inventory)`. Each `target` is the base URI of a foreign Sphinx documentation set and can be a local path or an HTTP URI. The `inventory` indicates where the inventory file can be found: it can be `None` (an `objects.inv` file at the same location as the base URI) or another local file path or a full HTTP URI to an inventory file.

The unique identifier can be used to prefix cross-reference targets, so that it is clear which intersphinx set the target belongs to. A link like ``:ref:`comparison manual <python:comparisons>` `` will link to the label "comparisons" in the doc set "python", if it exists.

Example

To add links to modules and objects in the Python standard library documentation, use:

```
intersphinx_mapping = {'python': ('https://docs.python.org/3', None)}
```

This will download the corresponding `objects.inv` file from the Internet and generate links to the pages under the given URI. The downloaded inventory is cached in the Sphinx environment, so it must be re-downloaded whenever you do a full rebuild.

A second example, showing the meaning of a non-`None` value of the second tuple item:

```
intersphinx_mapping = {'python': ('https://docs.python.org/3',
                                  'python-inv.txt')}
```

This will read the inventory from `python-inv.txt` in the source directory, but still generate links to the pages under `https://docs.python.org/3`. It is up to you to update the inventory file as new objects are added to the Python documentation.

Multiple targets for the inventory

New in version 1.3.

Alternative files can be specified for each inventory. One can give a tuple for the second inventory tuple item as shown in the following example. This will read the inventory iterating through the (second) tuple items until the first successful fetch. The primary use case for this to specify mirror sites for server downtime of the primary inventory:

```
intersphinx_mapping = {'python': ('https://docs.python.org/3',
                                  (None, 'python-inv.txt'))}
```

For a set of books edited and tested locally and then published together, it could be helpful to try a local inventory file first, to check references before publication:

```
intersphinx_mapping = {
    'otherbook':
        ('https://myproj.readthedocs.io/projects/otherbook/en/latest',
            ('../../otherbook/build/html/objects.inv', None)),
}
```

`intersphinx_cache_limit`

The maximum number of days to cache remote inventories. The default is `5`, meaning five days. Set this to a negative value to cache inventories for unlimited time.

`intersphinx_timeout`

The number of seconds for timeout. The default is `None`, meaning do not timeout.

Note: timeout is not a time limit on the entire response download; rather, an exception is raised if the server has not issued a response for timeout seconds.

Showing all links of an Intersphinx mapping file

To show all Intersphinx links and their targets of an Intersphinx mapping file, run `python -msphinx.ext.intersphinx url-or-path`. This is helpful when searching for the root cause of a broken Intersphinx link in a documentation project. The following example prints the Intersphinx mapping of the Python 3 documentation:

```
$ python -msphinx.ext.intersphinx https://docs.python.org/3/objects.inv
```

Using Intersphinx with inventory file under Basic Authorization

Intersphinx supports Basic Authorization like this:

```
intersphinx_mapping = {'python': ('https://user:password@docs.python.org/3',
                                  None)}
```

The user and password will be stripped from the URL when generating the links.

`sphinx.ext.linkcode` – Add external links to source code

Module author: Pauli Virtanen

New in version 1.2.

This extension looks at your object descriptions (`.. class::`, `.. function::` etc.) and adds external links to code hosted somewhere on the web. The intent is similar to the `sphinx.ext.viewcode` extension, but assumes the source code can be found somewhere on the Internet.

In your configuration, you need to specify a *`linkcode_resolve`* function that returns an URL based on the object.

Configuration

`linkcode_resolve`

This is a function `linkcode_resolve(domain, info)`, which should return the URL to source code corresponding to the object in given domain with given information.

The function should return `None` if no link is to be added.

The argument `domain` specifies the language domain the object is in. `info` is a dictionary with the following keys guaranteed to be present (dependent on the domain):

- `py`: `module` (name of the module), `fullname` (name of the object)
- `c`: `names` (list of names for the object)
- `cpp`: `names` (list of names for the object)
- `javascript`: `object` (name of the object), `fullname` (name of the item)

Example:

```python
def linkcode_resolve(domain, info):
    if domain != 'py':
        return None
    if not info['module']:
        return None
    filename = info['module'].replace('.', '/')
    return "https://somesite/sourcerepo/%s.py" % filename
```

Math support for HTML outputs in Sphinx

New in version 0.5.

Changed in version 1.8: Math support for non-HTML builders is integrated to sphinx-core. So mathbase extension is no longer needed.

Since mathematical notation isn't natively supported by HTML in any way, Sphinx gives a math support to HTML document with several extensions. These use the reStructuredText math *directive* and *role*.

`sphinx.ext.imgmath` – Render math as images

New in version 1.4.

This extension renders math via LaTeX and dvipng[196] or dvisvgm[197] into PNG or SVG images. This of course means that the computer where the docs are built must have both programs available.

There are various configuration values you can set to influence how the images are built:

`imgmath_image_format`
: The output image format. The default is `'png'`. It should be either `'png'` or `'svg'`. The image is produced by first executing `latex` on the TeX mathematical mark-up then (depending on the requested format) either dvipng[198] or dvisvgm[199].

`imgmath_use_preview`
: `dvipng` and `dvisvgm` both have the ability to collect from LaTeX the "depth" of the rendered math: an inline image should use this "depth" in a `vertical-align` style to get correctly aligned with surrounding text.

  This mechanism requires the LaTeX preview package[200] (available as `preview-latex-style` on Ubuntu xenial). Therefore, the default for this option is `False` but it is strongly recommended to set it to `True`.

  Changed in version 2.2: This option can be used with the `'svg'` *`imgmath_image_format`*.

`imgmath_add_tooltips`
: Default: `True`. If false, do not add the LaTeX code as an "alt" attribute for math images.

`imgmath_font_size`
: The font size (in `pt`) of the displayed math. The default value is `12`. It must be a positive integer.

`imgmath_latex`
: The command name with which to invoke LaTeX. The default is `'latex'`; you may need to set this to a full path if `latex` is not in the executable search path.

  Since this setting is not portable from system to system, it is normally not useful to set it in `conf.py`; rather, giving it on the **`sphinx-build`** command line via the *`-D`* option should be preferable, like this:

[196] https://savannah.nongnu.org/projects/dvipng/
[197] https://dvisvgm.de/
[198] https://savannah.nongnu.org/projects/dvipng/
[199] https://dvisvgm.de/
[200] https://www.gnu.org/software/auctex/preview-latex.html

```
sphinx-build -b html -D imgmath_latex=C:\tex\latex.exe . _build/html
```

This value should only contain the path to the latex executable, not further arguments; use *imgmath_latex_args* for that purpose.

Hint: Some fancy LaTeX mark-up (an example was reported which used TikZ to add various decorations to the equation) require multiple runs of the LaTeX executable. To handle this, set this configuration setting to `'latexmk'` (or a full path to it) as this Perl script reliably chooses dynamically how many latex runs are needed.

imgmath_latex_args
Additional arguments to give to latex, as a list. The default is an empty list.

imgmath_latex_preamble
Additional LaTeX code to put into the preamble of the LaTeX files used to translate the math snippets. This is left empty by default. Use it e.g. to add packages which modify the fonts used for math, such as `'\\usepackage{newtxsf}'` for sans-serif fonts, or `'\\usepackage{fouriernc}'` for serif fonts. Indeed, the default LaTeX math fonts have rather thin glyphs which (in HTML output) often do not match well with the font for text.

imgmath_dvipng
The command name to invoke `dvipng`. The default is `'dvipng'`; you may need to set this to a full path if `dvipng` is not in the executable search path. This option is only used when `imgmath_image_format` is set to `'png'`.

imgmath_dvipng_args
Additional arguments to give to dvipng, as a list. The default value is `['-gamma', '1.5', '-D', '110', '-bg', 'Transparent']` which makes the image a bit darker and larger then it is by default (this compensates somewhat for the thinness of default LaTeX math fonts), and produces PNGs with a transparent background. This option is used only when `imgmath_image_format` is `'png'`.

imgmath_dvisvgm
The command name to invoke `dvisvgm`. The default is `'dvisvgm'`; you may need to set this to a full path if `dvisvgm` is not in the executable search path. This option is only used when `imgmath_image_format` is `'svg'`.

imgmath_dvisvgm_args
Additional arguments to give to dvisvgm, as a list. The default value is `['--no-fonts']`, which means that `dvisvgm` will render glyphs as path elements (cf the dvisvgm FAQ[201]). This option is used only when `imgmath_image_format` is `'svg'`.

`sphinx.ext.mathjax` – Render math via JavaScript

Warning: Version 4.0 changes the version of MathJax used to version 3. You may need to override `mathjax_path` to `https://cdn.jsdelivr.net/npm/mathjax@2/MathJax.js?config=TeX-AMS-MML_HTMLorMML` or update your configuration options for version 3.

New in version 1.1.

This extension puts math as-is into the HTML files. The JavaScript package MathJax[202] is then loaded and transforms the LaTeX markup to readable math live in the browser.

Because MathJax (and the necessary fonts) is very large, it is not included in Sphinx but is set to automatically include it from a third-party site.

[201] https://dvisvgm.de/FAQ
[202] https://www.mathjax.org/

Attention: You should use the math *directive* and *role*, not the native MathJax `$$`, `\(`, etc.

mathjax_path

The path to the JavaScript file to include in the HTML files in order to load MathJax.

The default is the `https://` URL that loads the JS files from the jsdelivr[203] Content Delivery Network. See the MathJax Getting Started page[204] for details. If you want MathJax to be available offline or without including resources from a third-party site, you have to download it and set this value to a different path.

The path can be absolute or relative; if it is relative, it is relative to the `_static` directory of the built docs.

For example, if you put MathJax into the static path of the Sphinx docs, this value would be `MathJax/MathJax.js`. If you host more than one Sphinx documentation set on one server, it is advisable to install MathJax in a shared location.

You can also give a full `https://` URL different from the CDN URL.

mathjax_options

The options to script tag for mathjax. For example, you can set integrity option with following setting:

```
mathjax_options = {
    'integrity': 'sha384-......',
}
```

The default is empty (`{}`).

New in version 1.8.

mathjax_config

The inline configuration options for mathjax. The value is used as a parameter of `MathJax.Hub.Config()`. For more information, please read Using in-line configuration options[205].

For example:

```
mathjax_config = {
    'extensions': ['tex2jax.js'],
    'jax': ['input/TeX', 'output/HTML-CSS'],
}
```

The default is empty (not configured).

New in version 1.8.

`sphinx.ext.jsmath` – Render math via JavaScript

This extension works just as the MathJax extension does, but uses the older package jsMath[206]. It provides this config value:

jsmath_path

The path to the JavaScript file to include in the HTML files in order to load JSMath. There is no default.

The path can be absolute or relative; if it is relative, it is relative to the `_static` directory of the built docs.

For example, if you put JSMath into the static path of the Sphinx docs, this value would be `jsMath/easy/load.js`. If you host more than one Sphinx documentation set on one server, it is advisable to install jsMath in a shared location.

[203] https://www.jsdelivr.com/
[204] https://www.mathjax.org/#gettingstarted
[205] https://docs.mathjax.org/en/latest/configuration.html#using-in-line-configuration-options
[206] http://www.math.union.edu/~dpvc/jsmath/

`sphinx.ext.napoleon` – Support for NumPy and Google style docstrings

Module author: Rob Ruana

New in version 1.3.

Overview

Are you tired of writing docstrings that look like this:

```
:param path: The path of the file to wrap
:type path: str
:param field_storage: The :class:`FileStorage` instance to wrap
:type field_storage: FileStorage
:param temporary: Whether or not to delete the file when the File
   instance is destructed
:type temporary: bool
:returns: A buffered writable file descriptor
:rtype: BufferedFileStorage
```

reStructuredText[207] is great, but it creates visually dense, hard to read docstrings[208]. Compare the jumble above to the same thing rewritten according to the Google Python Style Guide[209]:

```
Args:
    path (str): The path of the file to wrap
    field_storage (FileStorage): The :class:`FileStorage` instance to wrap
    temporary (bool): Whether or not to delete the file when the File
       instance is destructed

Returns:
    BufferedFileStorage: A buffered writable file descriptor
```

Much more legible, no?

Napoleon is a *extension* that enables Sphinx to parse both NumPy[210] and Google[211] style docstrings - the style recommended by Khan Academy[212].

Napoleon is a pre-processor that parses NumPy[213] and Google[214] style docstrings and converts them to reStructuredText before Sphinx attempts to parse them. This happens in an intermediate step while Sphinx is processing the documentation, so it doesn't modify any of the docstrings in your actual source code files.

[207] http://docutils.sourceforge.net/rst.html
[208] https://www.python.org/dev/peps/pep-0287/
[209] https://google.github.io/styleguide/pyguide.html
[210] https://numpydoc.readthedocs.io/en/latest/format.html#docstring-standard
[211] https://google.github.io/styleguide/pyguide.html#Comments
[212] https://github.com/Khan/style-guides/blob/master/style/python.md#docstrings
[213] https://numpydoc.readthedocs.io/en/latest/format.html#docstring-standard
[214] https://google.github.io/styleguide/pyguide.html#Comments

Getting Started

1. After *setting up Sphinx* to build your docs, enable napoleon in the Sphinx *conf.py* file:

   ```
   # conf.py

   # Add napoleon to the extensions list
   extensions = ['sphinx.ext.napoleon']
   ```

2. Use *sphinx-apidoc* to build your API documentation:

   ```
   $ sphinx-apidoc -f -o docs/source projectdir
   ```

Docstrings

Napoleon interprets every docstring that `autodoc` can find, including docstrings on: `modules`, `classes`, `attributes`, `methods`, `functions`, and `variables`. Inside each docstring, specially formatted *Sections* are parsed and converted to reStructuredText.

All standard reStructuredText formatting still works as expected.

Docstring Sections

All of the following section headers are supported:

- `Args` *(alias of Parameters)*
- `Arguments` *(alias of Parameters)*
- `Attention`
- `Attributes`
- `Caution`
- `Danger`
- `Error`
- `Example`
- `Examples`
- `Hint`
- `Important`
- `Keyword Args` *(alias of Keyword Arguments)*
- `Keyword Arguments`
- `Methods`
- `Note`
- `Notes`
- `Other Parameters`
- `Parameters`
- `Return` *(alias of Returns)*
- `Returns`

- `Raise` *(alias of Raises)*
- `Raises`
- `References`
- `See Also`
- `Tip`
- `Todo`
- `Warning`
- `Warnings` *(alias of Warning)*
- `Warn` *(alias of Warns)*
- `Warns`
- `Yield` *(alias of Yields)*
- `Yields`

Google vs NumPy

Napoleon supports two styles of docstrings: Google[215] and NumPy[216]. The main difference between the two styles is that Google uses indentation to separate sections, whereas NumPy uses underlines.

Google style:

```
def func(arg1, arg2):
    """Summary line.

    Extended description of function.

    Args:
        arg1 (int): Description of arg1
        arg2 (str): Description of arg2

    Returns:
        bool: Description of return value

    """
    return True
```

NumPy style:

```
def func(arg1, arg2):
    """Summary line.

    Extended description of function.

    Parameters
    ----------
    arg1 : int
        Description of arg1
```

(continues on next page)

[215] https://google.github.io/styleguide/pyguide.html#Comments
[216] https://numpydoc.readthedocs.io/en/latest/format.html#docstring-standard

(continued from previous page)

```
    arg2 : str
        Description of arg2

    Returns
    -------
    bool
        Description of return value

    """
    return True
```

NumPy style tends to require more vertical space, whereas Google style tends to use more horizontal space. Google style tends to be easier to read for short and simple docstrings, whereas NumPy style tends be easier to read for long and in-depth docstrings.

The Khan Academy[217] recommends using Google style.

The choice between styles is largely aesthetic, but the two styles should not be mixed. Choose one style for your project and be consistent with it.

See also:

For complete examples:

- example_google
- example_numpy

Type Annotations

PEP 484[218] introduced a standard way to express types in Python code. This is an alternative to expressing types directly in docstrings. One benefit of expressing types according to PEP 484[219] is that type checkers and IDEs can take advantage of them for static code analysis. PEP 484[220] was then extended by PEP 526[221] which introduced a similar way to annotate variables (and attributes).

Google style with Python 3 type annotations:

```
def func(arg1: int, arg2: str) -> bool:
    """Summary line.

    Extended description of function.

    Args:
        arg1: Description of arg1
        arg2: Description of arg2

    Returns:
        Description of return value

    """
    return True
```

(continues on next page)

[217] https://github.com/Khan/style-guides/blob/master/style/python.md#docstrings
[218] https://www.python.org/dev/peps/pep-0484/
[219] https://www.python.org/dev/peps/pep-0484/
[220] https://www.python.org/dev/peps/pep-0484/
[221] https://www.python.org/dev/peps/pep-0526/

(continued from previous page)

```

class Class:
    """Summary line.

    Extended description of class

    Attributes:
        attr1: Description of attr1
        attr2: Description of attr2
    """

    attr1: int
    attr2: str
```

Google style with types in docstrings:

```
def func(arg1, arg2):
    """Summary line.

    Extended description of function.

    Args:
        arg1 (int): Description of arg1
        arg2 (str): Description of arg2

    Returns:
        bool: Description of return value

    """
    return True

class Class:
    """Summary line.

    Extended description of class

    Attributes:
        attr1 (int): Description of attr1
        attr2 (str): Description of attr2
    """
```

Note: Python 2/3 compatible annotations[222] aren't currently supported by Sphinx and won't show up in the docs.

[222] https://www.python.org/dev/peps/pep-0484/#suggested-syntax-for-python-2-7-and-straddling-code

Configuration

Listed below are all the settings used by napoleon and their default values. These settings can be changed in the Sphinx *conf.py* file. Make sure that “sphinx.ext.napoleon” is enabled in *conf.py*:

```
# conf.py

# Add any Sphinx extension module names here, as strings
extensions = ['sphinx.ext.napoleon']

# Napoleon settings
napoleon_google_docstring = True
napoleon_numpy_docstring = True
napoleon_include_init_with_doc = False
napoleon_include_private_with_doc = False
napoleon_include_special_with_doc = True
napoleon_use_admonition_for_examples = False
napoleon_use_admonition_for_notes = False
napoleon_use_admonition_for_references = False
napoleon_use_ivar = False
napoleon_use_param = True
napoleon_use_rtype = True
napoleon_type_aliases = None
napoleon_attr_annotations = True
```

`napoleon_google_docstring`
True to parse Google style[223] docstrings. False to disable support for Google style docstrings. *Defaults to True.*

`napoleon_numpy_docstring`
True to parse NumPy style[224] docstrings. False to disable support for NumPy style docstrings. *Defaults to True.*

`napoleon_include_init_with_doc`
True to list `__init___` docstrings separately from the class docstring. False to fall back to Sphinx’s default behavior, which considers the `__init___` docstring as part of the class documentation. *Defaults to False.*

If True:

```
def __init__(self):
    \"\"\"
    This will be included in the docs because it has a docstring
    \"\"\"

def __init__(self):
    # This will NOT be included in the docs
```

`napoleon_include_private_with_doc`
True to include private members (like `_membername`) with docstrings in the documentation. False to fall back to Sphinx’s default behavior. *Defaults to False.*

If True:

```
def _included(self):
    """
    This will be included in the docs because it has a docstring
```

(continues on next page)

[223] https://google.github.io/styleguide/pyguide.html
[224] https://numpydoc.readthedocs.io/en/latest/format.html#docstring-standard

(continued from previous page)

```
    """
    pass

def _skipped(self):
    # This will NOT be included in the docs
    pass
```

napoleon_include_special_with_doc

True to include special members (like `__membername__`) with docstrings in the documentation. False to fall back to Sphinx's default behavior. *Defaults to True.*

If True:

```
def __str__(self):
    """
    This will be included in the docs because it has a docstring
    """
    return unicode(self).encode('utf-8')

def __unicode__(self):
    # This will NOT be included in the docs
    return unicode(self.__class__.__name__)
```

napoleon_use_admonition_for_examples

True to use the `.. admonition::` directive for the **Example** and **Examples** sections. False to use the `.. rubric::` directive instead. One may look better than the other depending on what HTML theme is used. *Defaults to False.*

This NumPy style[225] snippet will be converted as follows:

```
Example
-------
This is just a quick example
```

If True:

```
.. admonition:: Example

   This is just a quick example
```

If False:

```
.. rubric:: Example

This is just a quick example
```

napoleon_use_admonition_for_notes

True to use the `.. admonition::` directive for **Notes** sections. False to use the `.. rubric::` directive instead. *Defaults to False.*

Note: The singular **Note** section will always be converted to a `.. note::` directive.

[225] https://numpydoc.readthedocs.io/en/latest/format.html#docstring-standard

See also:

`napoleon_use_admonition_for_examples`

`napoleon_use_admonition_for_references`
True to use the `.. admonition::` directive for **References** sections. False to use the `.. rubric::` directive instead. *Defaults to False.*

See also:

`napoleon_use_admonition_for_examples`

`napoleon_use_ivar`
True to use the `:ivar:` role for instance variables. False to use the `.. attribute::` directive instead. *Defaults to False.*

This NumPy style[226] snippet will be converted as follows:

```
Attributes
----------
attr1 : int
    Description of `attr1`
```

If True:

```
:ivar attr1: Description of `attr1`
:vartype attr1: int
```

If False:

```
.. attribute:: attr1

   Description of `attr1`

   :type: int
```

`napoleon_use_param`
True to use a `:param:` role for each function parameter. False to use a single `:parameters:` role for all the parameters. *Defaults to True.*

This NumPy style[227] snippet will be converted as follows:

```
Parameters
----------
arg1 : str
    Description of `arg1`
arg2 : int, optional
    Description of `arg2`, defaults to 0
```

If True:

```
:param arg1: Description of `arg1`
:type arg1: str
:param arg2: Description of `arg2`, defaults to 0
:type arg2: :class:`int`, *optional*
```

If False:

[226] https://numpydoc.readthedocs.io/en/latest/format.html#docstring-standard

```
:parameters: * **arg1** (*str*) --
               Description of `arg1`
             * **arg2** (*int, optional*) --
               Description of `arg2`, defaults to 0
```

`napoleon_use_keyword`
True to use a `:keyword:` role for each function keyword argument. False to use a single `:keyword arguments:` role for all the keywords. *Defaults to True.*

This behaves similarly to `napoleon_use_param`. Note unlike docutils, `:keyword:` and `:param:` will not be treated the same way - there will be a separate "Keyword Arguments" section, rendered in the same fashion as "Parameters" section (type links created if possible)

See also:

`napoleon_use_param`

`napoleon_use_rtype`
True to use the `:rtype:` role for the return type. False to output the return type inline with the description. *Defaults to True.*

This NumPy style[228] snippet will be converted as follows:

```
Returns
-------
bool
    True if successful, False otherwise
```

If True:

```
:returns: True if successful, False otherwise
:rtype: bool
```

If False:

```
:returns: *bool* -- True if successful, False otherwise
```

`napoleon_type_aliases`
A mapping to translate type names to other names or references. Works only when `napoleon_use_param = True`. *Defaults to None.*

With:

```
napoleon_type_aliases = {
    "CustomType": "mypackage.CustomType",
    "dict-like": ":term:`dict-like <mapping>`",
}
```

This NumPy style[229] snippet:

```
Parameters
----------
arg1 : CustomType
    Description of `arg1`
arg2 : dict-like
    Description of `arg2`
```

[227] https://numpydoc.readthedocs.io/en/latest/format.html#docstring-standard
[228] https://numpydoc.readthedocs.io/en/latest/format.html#docstring-standard

becomes:

```
:param arg1: Description of `arg1`
:type arg1: mypackage.CustomType
:param arg2: Description of `arg2`
:type arg2: :term:`dict-like <mapping>`
```

New in version 3.2.

`napoleon_attr_annotations`

True to allow using PEP 526[230] attributes annotations in classes. If an attribute is documented in the docstring without a type and has an annotation in the class body, that type is used.

New in version 3.4.

`napoleon_custom_sections`

Add a list of custom sections to include, expanding the list of parsed sections. *Defaults to None.*

The entries can either be strings or tuples, depending on the intention:

- To create a custom "generic" section, just pass a string.
- To create an alias for an existing section, pass a tuple containing the alias name and the original, in that order.
- To create a custom section that displays like the parameters or returns section, pass a tuple containing the custom section name and a string value, "params_style" or "returns_style".

If an entry is just a string, it is interpreted as a header for a generic section. If the entry is a tuple/list/indexed container, the first entry is the name of the section, the second is the section key to emulate. If the second entry value is "params_style" or "returns_style", the custom section will be displayed like the parameters section or returns section.

New in version 1.8.

Changed in version 3.5: Support `params_style` and `returns_style`

`sphinx.ext.todo` – Support for todo items

Module author: Daniel Bültmann

New in version 0.5.

There are two additional directives when using this extension:

`.. todo::`

Use this directive like, for example, *`note`*.

It will only show up in the output if *`todo_include_todos`* is `True`.

New in version 1.3.2: This directive supports an `class` option that determines the class attribute for HTML output. If not given, the class defaults to `admonition-todo`.

`.. todolist::`

This directive is replaced by a list of all todo directives in the whole documentation, if *`todo_include_todos`* is `True`.

These can be configured as seen below.

[229] https://numpydoc.readthedocs.io/en/latest/format.html#docstring-standard
[230] https://www.python.org/dev/peps/pep-0526/

Configuration

`todo_include_todos`
If this is `True`, *`todo`* and *`todolist`* produce output, else they produce nothing. The default is `False`.

`todo_emit_warnings`
If this is `True`, *`todo`* emits a warning for each TODO entries. The default is `False`.

New in version 1.5.

`todo_link_only`
If this is `True`, *`todolist`* produce output without file path and line, The default is `False`.

New in version 1.4.

autodoc provides the following an additional event:

`todo-defined`(*app*, *node*)
New in version 1.5.

Emitted when a todo is defined. *node* is the defined `sphinx.ext.todo.todo_node` node.

`sphinx.ext.viewcode` – Add links to highlighted source code

Module author: Georg Brandl

New in version 1.0.

This extension looks at your Python object descriptions (`.. class::`, `.. function::` etc.) and tries to find the source files where the objects are contained. When found, a separate HTML page will be output for each module with a highlighted version of the source code, and a link will be added to all object descriptions that leads to the source code of the described object. A link back from the source to the description will also be inserted.

Warning: Basically, `viewcode` extension will import the modules being linked to. If any modules have side effects on import, these will be executed when `sphinx-build` is run.

If you document scripts (as opposed to library modules), make sure their main routine is protected by a `if __name__ == '__main__'` condition.

In addition, if you don't want to import the modules by `viewcode`, you can tell the location of the location of source code to `viewcode` using the *`viewcode-find-source`* event.

If *`viewcode_follow_imported_members`* is enabled, you will also need to resolve imported attributes using the *`viewcode-follow-imported`* event.

This extension works only on HTML related builders like `html`, `applehelp`, `devhelp`, `htmlhelp`, `qthelp` and so on except `singlehtml`. By default `epub` builder doesn't support this extension (see *`viewcode_enable_epub`*).

Configuration

`viewcode_follow_imported_members`
If this is `True`, viewcode extension will emit *`viewcode-follow-imported`* event to resolve the name of the module by other extensions. The default is `True`.

New in version 1.3.

Changed in version 1.8: Renamed from `viewcode_import` to `viewcode_follow_imported_members`.

`viewcode_enable_epub`

If this is `True`, viewcode extension is also enabled even if you use epub builders. This extension generates pages outside toctree, but this is not preferred as epub format.

Until 1.4.x, this extension is always enabled. If you want to generate epub as same as 1.4.x, you should set `True`, but epub format checker's score becomes worse.

The default is `False`.

New in version 1.5.

> **Warning:** Not all epub readers support pages generated by viewcode extension. These readers ignore links to pages are not under toctree.
>
> Some reader's rendering result are corrupted and epubcheck[231]'s score becomes worse even if the reader supports.

`viewcode-find-source`(*app*, *modname*)

New in version 1.8.

Find the source code for a module. An event handler for this event should return a tuple of the source code itself and a dictionary of tags. The dictionary maps the name of a class, function, attribute, etc to a tuple of its type, the start line number, and the end line number. The type should be one of "class", "def", or "other".

Parameters

- **`app`** – The Sphinx application object.
- **`modname`** – The name of the module to find source code for.

`viewcode-follow-imported`(*app*, *modname*, *attribute*)

New in version 1.8.

Find the name of the original module for an attribute.

Parameters

- **`app`** – The Sphinx application object.
- **`modname`** – The name of the module that the attribute belongs to.
- **`attribute`** – The name of the member to follow.

Third-party extensions

You can find several extensions contributed by users in the sphinx-contrib[232] organization. If you wish to include your extension in this organization, simply follow the instructions provided in the github-administration[233] project. This is optional and there are several extensions hosted elsewhere. The awesome-sphinxdoc[234] project contains a curated list of Sphinx packages, and many packages use the Framework :: Sphinx :: Extension[235] and Framework :: Sphinx :: Theme[236] trove classifiers for Sphinx extensions and themes, respectively.

[231] https://github.com/IDPF/epubcheck
[232] https://github.com/sphinx-contrib/
[233] https://github.com/sphinx-contrib/github-administration
[234] https://github.com/yoloseem/awesome-sphinxdoc
[235] https://pypi.org/search/?c=Framework+%3A%3A+Sphinx+%3A%3A+Extension
[236] https://pypi.org/search/?c=Framework+%3A%3A+Sphinx+%3A%3A+Theme

Where to put your own extensions?

Extensions local to a project should be put within the project's directory structure. Set Python's module search path, `sys.path`, accordingly so that Sphinx can find them. For example, if your extension `foo.py` lies in the `exts` subdirectory of the project root, put into `conf.py`:

```
import sys, os

sys.path.append(os.path.abspath('exts'))

extensions = ['foo']
```

You can also install extensions anywhere else on `sys.path`, e.g. in the `site-packages` directory.

1.8 HTML Theming

Sphinx provides a number of builders for HTML and HTML-based formats.

Builders

Todo: Populate when the 'builders' document is split up.

Themes

New in version 0.6.

Note: This section provides information about using pre-existing HTML themes. If you wish to create your own theme, refer to *HTML theme development*.

Sphinx supports changing the appearance of its HTML output via *themes*. A theme is a collection of HTML templates, stylesheet(s) and other static files. Additionally, it has a configuration file which specifies from which theme to inherit, which highlighting style to use, and what options exist for customizing the theme's look and feel.

Themes are meant to be project-unaware, so they can be used for different projects without change.

Using a theme

Using a *theme provided with Sphinx* is easy. Since these do not need to be installed, you only need to set the *`html_theme`* config value. For example, to enable the `classic` theme, add the following to `conf.py`:

```
html_theme = "classic"
```

You can also set theme-specific options using the *`html_theme_options`* config value. These options are generally used to change the look and feel of the theme. For example, to place the sidebar on the right side and a black background for the relation bar (the bar with the navigation links at the page's top and bottom), add the following `conf.py`:

```
html_theme_options = {
    "rightsidebar": "true",
    "relbarbgcolor": "black"
}
```

If the theme does not come with Sphinx, it can be in two static forms or as a Python package. For the static forms, either a directory (containing `theme.conf` and other needed files), or a zip file with the same contents is supported. The directory or zipfile must be put where Sphinx can find it; for this there is the config value *html_theme_path*. This can be a list of directories, relative to the directory containing `conf.py`, that can contain theme directories or zip files. For example, if you have a theme in the file `blue.zip`, you can put it right in the directory containing `conf.py` and use this configuration:

```
html_theme = "blue"
html_theme_path = ["."]
```

The third form is a Python package. If a theme you want to use is distributed as a Python package, you can use it after installing

```
# installing theme package
$ pip install sphinxjp.themes.dotted
```

Once installed, this can be used in the same manner as a directory or zipfile-based theme:

```
html_theme = "dotted"
```

For more information on the design of themes, including information about writing your own themes, refer to *HTML theme development*.

Builtin themes

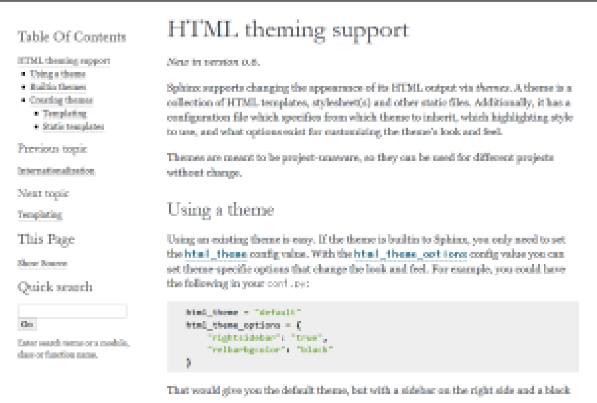
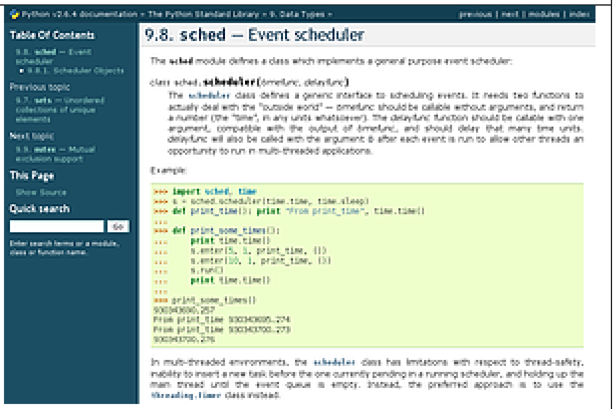
Theme overview	
 *alabaster*	 *classic*

Table Of Contents
HTML theming support
- Using a theme
- Builtin themes
- Creating themes
  - Templating
  - Static templates

Previous topic
Internationalization
Next topic
Templating
This Page
Show Source
Quick search
Go
Enter search terms or a module, class or function name.

HTML theming support

New in version 0.6.

Sphinx supports changing the appearance of its HTML output via *themes*. A theme is a collection of HTML templates, stylesheet(s) and other static files. Additionally, it has a configuration file which specifies from which theme to inherit, which highlighting style to use, and what options exist for customizing the theme's look and feel.

Themes are meant to be project-unaware, so they can be used for different projects without change.

Using a theme

Using an existing theme is easy. If the theme is builtin to Sphinx, you only need to set the `html_theme` config value. With the `html_theme_options` config value you can set theme-specific options that change the look and feel. For example, you could have the following in your `conf.py`:

```
html_theme = "default"
html_theme_options = {
    "rightsidebar": "true",
    "relbarbgcolor": "black"
}
```

That would give you the default theme, but with a sidebar on the right side and a black

Python v2.6.4 documentation » The Python Standard Library » 9. Data Types » previous | next | modules | index

Table Of Contents
9.8. sched — Event scheduler
- 9.8.1. Scheduler Objects

Previous topic
9.7. sets — Unordered collections of unique elements
Next topic
9.9. mutex — Mutual exclusion support
This Page
Show Source
Quick search
Go
Enter search terms or a module, class or function name.

9.8. sched — Event scheduler

The **sched** module defines a class which implements a general purpose event scheduler:

class sched.**scheduler**(*timefunc*, *delayfunc*)

The `scheduler` class defines a generic interface to scheduling events. It needs two functions to actually deal with the "outside world" — *timefunc* should be callable without arguments, and return a number (the "time", in any units whatsoever). The *delayfunc* function should be callable with one argument, compatible with the output of *timefunc*, and should delay that many time units. *delayfunc* will also be called with the argument 0 after each event is run to allow other threads an opportunity to run in multi-threaded applications.

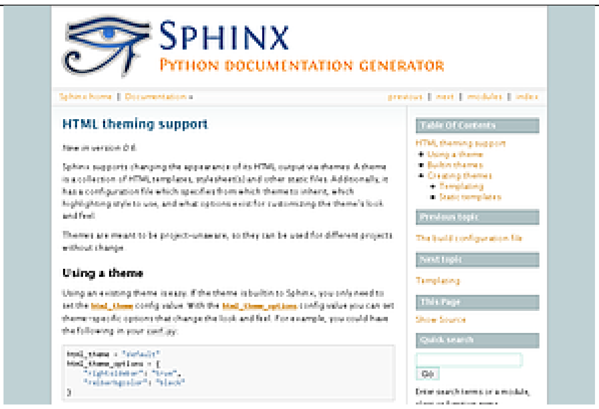
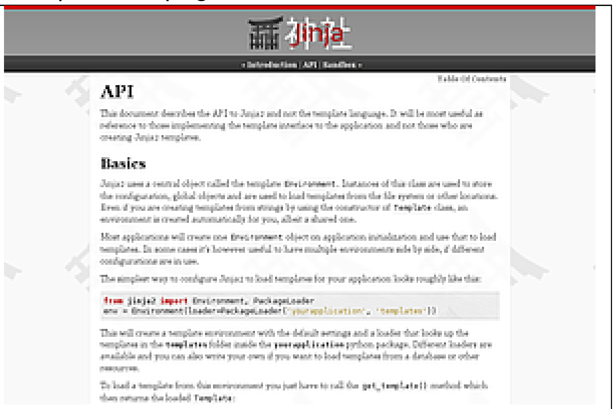
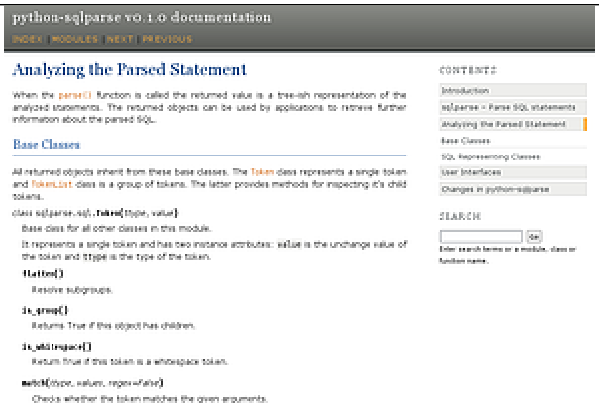
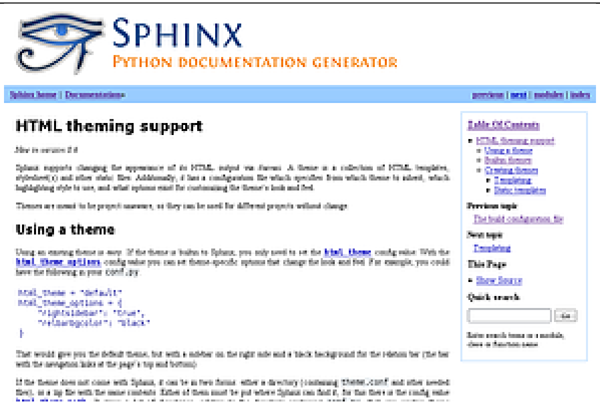
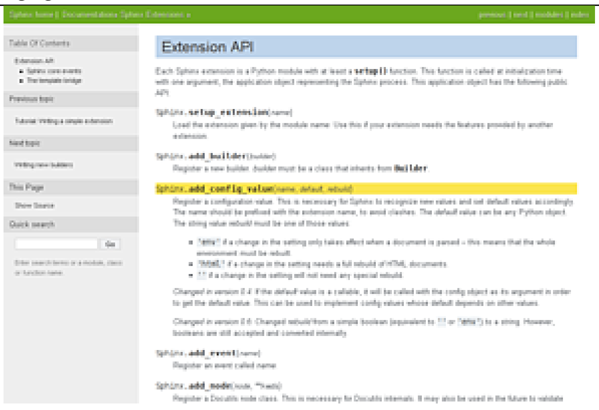
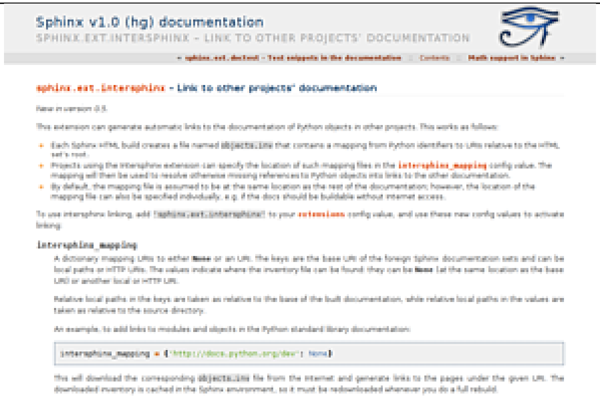
Example:

```
>>> import sched, time
>>> s = sched.scheduler(time.time, time.sleep)
>>> def print_time(): print "From print_time", time.time()
...
>>> def print_some_times():
...     print time.time()
...     s.enter(5, 1, print_time, ())
...     s.enter(10, 1, print_time, ())
...     s.run()
...     print time.time()
...
>>> print_some_times()
930343690.257
From print_time 930343695.274
From print_time 930343700.273
930343700.276
```

In multi-threaded environments, the `scheduler` class has limitations with respect to thread-safety, inability to insert a new task before the one currently pending in a running scheduler, and holding up the main thread until the event queue is empty. Instead, the preferred approach is to use the `threading.Timer` class instead.

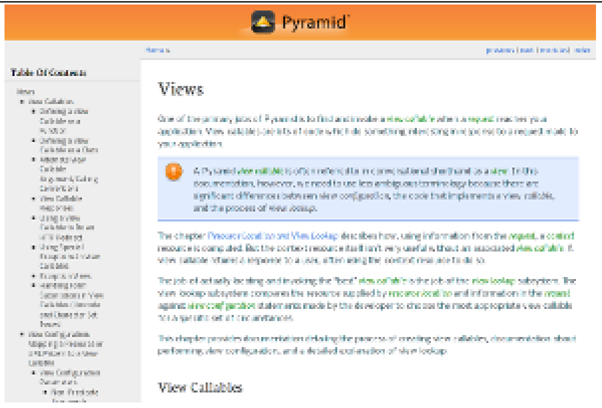
continues on next page

Table 2 – continued from previous page

 *sphinxdoc*	 *scrolls*
 *agogo*	 *traditional*
 *nature*	 *haiku*

continues on next page

Table 2 – continued from previous page

 *pyramid*	*bizstyle*

Sphinx comes with a selection of themes to choose from.

These themes are:

basic This is a basically unstyled layout used as the base for the other themes, and usable as the base for custom themes as well. The HTML contains all important elements like sidebar and relation bar. There are these options (which are inherited by the other themes):

- **nosidebar** (true or false): Don't include the sidebar. Defaults to `False`.
- **sidebarwidth** (int or str): Width of the sidebar in pixels. This can be an int, which is interpreted as pixels or a valid CSS dimension string such as '70em' or '50%'. Defaults to 230 pixels.
- **body_min_width** (int or str): Minimal width of the document body. This can be an int, which is interpreted as pixels or a valid CSS dimension string such as '70em' or '50%'. Use 0 if you don't want a width limit. Defaults may depend on the theme (often 450px).
- **body_max_width** (int or str): Maximal width of the document body. This can be an int, which is interpreted as pixels or a valid CSS dimension string such as '70em' or '50%'. Use 'none' if you don't want a width limit. Defaults may depend on the theme (often 800px).
- **navigation_with_keys** (true or false): Allow navigating to the previous/next page using the keyboard's left and right arrows. Defaults to `False`.
- **globaltoc_collapse** (true or false): Only expand subsections of the current document in `globaltoc.html` (see *`html_sidebars`*). Defaults to `True`.

  New in version 3.1.
- **globaltoc_includehidden** (true or false): Show even those subsections in `globaltoc.html` (see *`html_sidebars`*) which have been included with the `:hidden:` flag of the *`toctree`* directive. Defaults to `False`.

  New in version 3.1.
- **globaltoc_maxdepth** (int): The maximum depth of the toctree in `globaltoc.html` (see *`html_sidebars`*). Set it to -1 to allow unlimited depth. Defaults to the max depth selected in the toctree directive.

  New in version 3.2.

alabaster Alabaster theme[237] is a modified "Kr" Sphinx theme from @kennethreitz (especially as used in his Requests project), which was itself originally based on @mitsuhiko's theme used for Flask & related projects. Refer to its installation page[238] for information on how to configure *`html_sidebars`* for its use.

[237] https://pypi.org/project/alabaster/
[238] https://alabaster.readthedocs.io/en/latest/installation.html

classic This is the classic theme, which looks like the Python 2 documentation[239]. It can be customized via these options:

- **rightsidebar** (true or false): Put the sidebar on the right side. Defaults to `False`.
- **stickysidebar** (true or false): Make the sidebar "fixed" so that it doesn't scroll out of view for long body content. This may not work well with all browsers. Defaults to `False`.
- **collapsiblesidebar** (true or false): Add an *experimental* JavaScript snippet that makes the sidebar collapsible via a button on its side. Defaults to `False`.
- **externalrefs** (true or false): Display external links differently from internal links. Defaults to `False`.

There are also various color and font options that can change the color scheme without having to write a custom stylesheet:

- **footerbgcolor** (CSS color): Background color for the footer line.
- **footertextcolor** (CSS color): Text color for the footer line.
- **sidebarbgcolor** (CSS color): Background color for the sidebar.
- **sidebarbtncolor** (CSS color): Background color for the sidebar collapse button (used when *collapsiblesidebar* is `True`).
- **sidebartextcolor** (CSS color): Text color for the sidebar.
- **sidebarlinkcolor** (CSS color): Link color for the sidebar.
- **relbarbgcolor** (CSS color): Background color for the relation bar.
- **relbartextcolor** (CSS color): Text color for the relation bar.
- **relbarlinkcolor** (CSS color): Link color for the relation bar.
- **bgcolor** (CSS color): Body background color.
- **textcolor** (CSS color): Body text color.
- **linkcolor** (CSS color): Body link color.
- **visitedlinkcolor** (CSS color): Body color for visited links.
- **headbgcolor** (CSS color): Background color for headings.
- **headtextcolor** (CSS color): Text color for headings.
- **headlinkcolor** (CSS color): Link color for headings.
- **codebgcolor** (CSS color): Background color for code blocks.
- **codetextcolor** (CSS color): Default text color for code blocks, if not set differently by the highlighting style.
- **bodyfont** (CSS font-family): Font for normal text.
- **headfont** (CSS font-family): Font for headings.

sphinxdoc The theme originally used by this documentation. It features a sidebar on the right side. There are currently no options beyond *nosidebar* and *sidebarwidth*.

Note: The Sphinx documentation now uses an adjusted version of the sphinxdoc theme[240].

scrolls A more lightweight theme, based on the Jinja documentation[241]. The following color options are available:

- **headerbordercolor**

[239] https://docs.python.org/2/
[240] https://github.com/sphinx-doc/sphinx/tree/master/doc/_themes/sphinx13
[241] http://jinja.pocoo.org/

- **subheadlinecolor**
- **linkcolor**
- **visitedlinkcolor**
- **admonitioncolor**

agogo A theme created by Andi Albrecht. The following options are supported:

- **bodyfont** (CSS font family): Font for normal text.
- **headerfont** (CSS font family): Font for headings.
- **pagewidth** (CSS length): Width of the page content, default 70em.
- **documentwidth** (CSS length): Width of the document (without sidebar), default 50em.
- **sidebarwidth** (CSS length): Width of the sidebar, default 20em.
- **rightsidebar** (true or false): Put the sidebar on the right side. Defaults to `True`.
- **bgcolor** (CSS color): Background color.
- **headerbg** (CSS value for "background"): background for the header area, default a grayish gradient.
- **footerbg** (CSS value for "background"): background for the footer area, default a light gray gradient.
- **linkcolor** (CSS color): Body link color.
- **headercolor1**, **headercolor2** (CSS color): colors for <h1> and <h2> headings.
- **headerlinkcolor** (CSS color): Color for the backreference link in headings.
- **textalign** (CSS *text-align* value): Text alignment for the body, default is `justify`.

nature A greenish theme. There are currently no options beyond *nosidebar* and *sidebarwidth*.

pyramid A theme from the Pyramid web framework project, designed by Blaise Laflamme. There are currently no options beyond *nosidebar* and *sidebarwidth*.

haiku A theme without sidebar inspired by the Haiku OS user guide[242]. The following options are supported:

- **full_logo** (true or false, default `False`): If this is true, the header will only show the *`html_logo`*. Use this for large logos. If this is false, the logo (if present) will be shown floating right, and the documentation title will be put in the header.
- **textcolor**, **headingcolor**, **linkcolor**, **visitedlinkcolor**, **hoverlinkcolor** (CSS colors): Colors for various body elements.

traditional A theme resembling the old Python documentation. There are currently no options beyond *nosidebar* and *sidebarwidth*.

epub A theme for the epub builder. This theme tries to save visual space which is a sparse resource on ebook readers. The following options are supported:

- **relbar1** (true or false, default `True`): If this is true, the *relbar1* block is inserted in the epub output, otherwise it is omitted.
- **footer** (true or false, default `True`): If this is true, the *footer* block is inserted in the epub output, otherwise it is omitted.

bizstyle A simple bluish theme. The following options are supported beyond *nosidebar* and *sidebarwidth*:

- **rightsidebar** (true or false): Put the sidebar on the right side. Defaults to `False`.

New in version 1.3: 'alabaster', 'sphinx_rtd_theme' and 'bizstyle' theme.

Changed in version 1.3: The 'default' theme has been renamed to 'classic'. 'default' is still available, however it will emit a notice that it is an alias for the new 'alabaster' theme.

242 https://www.haiku-os.org/docs/userguide/en/contents.html

Third Party Themes

There are many third-party themes available for Sphinx. Some of these are for general use, while others are specific to an individual project.

sphinx-themes.org[243] is a gallery that showcases various themes for Sphinx, with demo documentation rendered under each theme. Themes can also be found on PyPI[244] (using the classifier `Framework :: Sphinx :: Theme`), GitHub[245] and GitLab[246].

1.9 Internationalization

New in version 1.1.

Complementary to translations provided for Sphinx-generated messages such as navigation bars, Sphinx provides mechanisms facilitating the translation of *documents*. See the *Options for internationalization* for details on configuration.

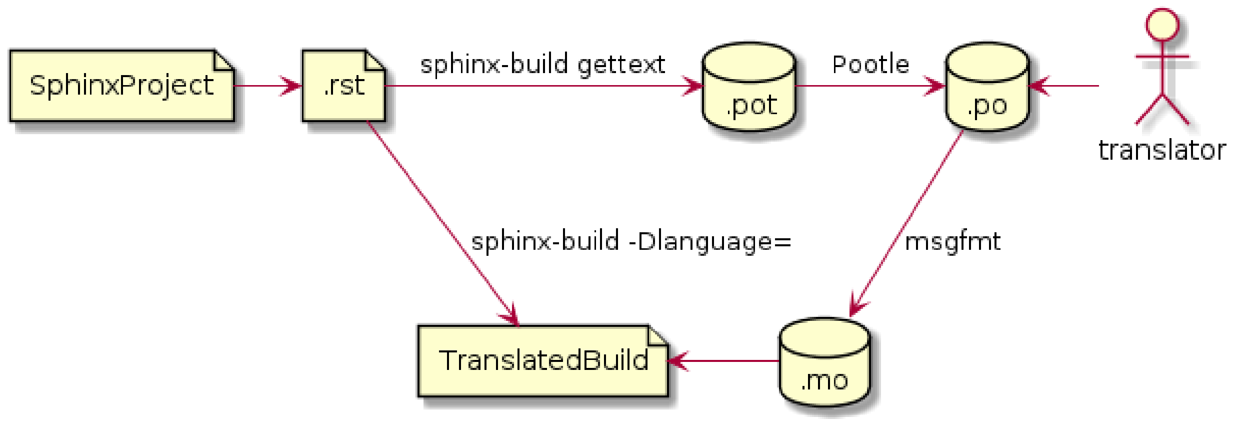


Fig. 1: Workflow visualization of translations in Sphinx. (The figure is created by plantuml[247].)

[243] https://sphinx-themes.org/
[244] https://pypi.org/search/?q=&o=&c=Framework+%3A%3A+Sphinx+%3A%3A+Theme
[245] https://github.com/search?utf8=%E2%9C%93&q=sphinx+theme
[246] https://gitlab.com/explore?name=sphinx+theme
[247] http://plantuml.com

Sphinx internationalization details

gettext[1] is an established standard for internationalization and localization. It naively maps messages in a program to a translated string. Sphinx uses these facilities to translate whole documents.

Initially project maintainers have to collect all translatable strings (also referred to as *messages*) to make them known to translators. Sphinx extracts these through invocation of `sphinx-build -b gettext`.

Every single element in the doctree will end up in a single message which results in lists being equally split into different chunks while large paragraphs will remain as coarsely-grained as they were in the original document. This grants seamless document updates while still providing a little bit of context for translators in free-text passages. It is the maintainer's task to split up paragraphs which are too large as there is no sane automated way to do that.

After Sphinx successfully ran the *MessageCatalogBuilder* you will find a collection of `.pot` files in your output directory. These are **catalog templates** and contain messages in your original language *only*.

They can be delivered to translators which will transform them to `.po` files — so called **message catalogs** — containing a mapping from the original messages to foreign-language strings.

gettext compiles them into a binary format known as **binary catalogs** through **msgfmt** for efficiency reasons. If you make these files discoverable with *locale_dirs* for your *language*, Sphinx will pick them up automatically.

An example: you have a document `usage.rst` in your Sphinx project. The *gettext* builder will put its messages into `usage.pot`. Imagine you have Spanish translations[2] stored in `usage.po` — for your builds to be translated you need to follow these instructions:

- Compile your message catalog to a locale directory, say `locale`, so it ends up in `./locale/es/LC_MESSAGES/usage.mo` in your source directory (where `es` is the language code for Spanish.)

  ```
  msgfmt "usage.po" -o "locale/es/LC_MESSAGES/usage.mo"
  ```

- Set *locale_dirs* to `["locale/"]`.
- Set *language* to `es` (also possible via *-D*).
- Run your desired build.

Translating with sphinx-intl

Quick guide

sphinx-intl[248] is a useful tool to work with Sphinx translation flow. This section describe an easy way to translate with *sphinx-intl*.

1. Install sphinx-intl[249].

   ```
   $ pip install sphinx-intl
   ```

2. Add configurations to `conf.py`.

   ```
   locale_dirs = ['locale/']   # path is example but recommended.
   gettext_compact = False     # optional.
   ```

   This case-study assumes that BUILDDIR is set to `_build`, *locale_dirs* is set to `locale/` and *gettext_compact* is set to `False` (the Sphinx document is already configured as such).

[1] See the GNU gettext utilities[257] for details on that software suite.
[257] https://www.gnu.org/software/gettext/manual/gettext.html#Introduction
[2] Because nobody expects the Spanish Inquisition!
[248] https://pypi.org/project/sphinx-intl/
[249] https://pypi.org/project/sphinx-intl/

3. Extract translatable messages into pot files.

   ```
   $ make gettext
   ```

   The generated pot files will be placed in the `_build/gettext` directory.

4. Generate po files.

   We'll use the pot files generated in the above step.

   ```
   $ sphinx-intl update -p _build/gettext -l de -l ja
   ```

   Once completed, the generated po files will be placed in the below directories:

   - `./locale/de/LC_MESSAGES/`
   - `./locale/ja/LC_MESSAGES/`

5. Translate po files.

   AS noted above, these are located in the `./locale/<lang>/LC_MESSAGES` directory. An example of one such file, from Sphinx, `builders.po`, is given below.

   ```
   # a5600c3d2e3d48fc8c261ea0284db79b
   #: ../../builders.rst:4
   msgid "Available builders"
   msgstr "<FILL HERE BY TARGET LANGUAGE>"
   ```

   Another case, msgid is multi-line text and contains reStructuredText syntax:

   ```
   # 302558364e1d41c69b3277277e34b184
   #: ../../builders.rst:9
   msgid ""
   "These are the built-in Sphinx builders. More builders can be added by "
   ":ref:`extensions <extensions>`."
   msgstr ""
   "FILL HERE BY TARGET LANGUAGE FILL HERE BY TARGET LANGUAGE FILL HERE "
   "BY TARGET LANGUAGE :ref:`EXTENSIONS <extensions>` FILL HERE."
   ```

   Please be careful not to break reST notation. Most po-editors will help you with that.

6. Build translated document.

   You need a *`language`* parameter in `conf.py` or you may also specify the parameter on the command line.

   For for BSD/GNU make, run:

   ```
   $ make -e SPHINXOPTS="-D language='de'" html
   ```

   For Windows **`cmd.exe`**, run:

   ```
   > set SPHINXOPTS=-D language=de
   > .\make.bat html
   ```

   For PowerShell, run:

   ```
   > Set-Item env:SPHINXOPTS "-D language=de"
   > .\make.bat html
   ```

Congratulations! You got the translated documentation in the `_build/html` directory.

New in version 1.3: **`sphinx-build`** that is invoked by make command will build po files into mo files.

If you are using 1.2.x or earlier, please invoke **sphinx-intl build** command before **make** command.

Translating

Update your po files by new pot files

If a document is updated, it is necessary to generate updated pot files and to apply differences to translated po files. In order to apply the updates from a pot file to the po file, use the **sphinx-intl update** command.

```
$ sphinx-intl update -p _build/locale
```

Using Transifex service for team translation

Transifex[250] is one of several services that allow collaborative translation via a web interface. It has a nifty Python-based command line client that makes it easy to fetch and push translations.

1. Install transifex-client[251].

   You need **tx** command to upload resources (pot files).

   ```
   $ pip install transifex-client
   ```

   See also:

   Transifex Client documentation[252]

2. Create your transifex[253] account and create new project for your document.

   Currently, transifex does not allow for a translation project to have more than one version of the document, so you'd better include a version number in your project name.

   For example:

   Project ID `sphinx-document-test_1_0`

   Project URL `https://www.transifex.com/projects/p/sphinx-document-test_1_0/`

3. Create config files for **tx** command.

   This process will create `.tx/config` in the current directory, as well as a `~/.transifexrc` file that includes auth information.

   ```
   $ tx init
   Creating .tx folder...
   Transifex instance [https://www.transifex.com]:
   ...
   Please enter your transifex username: <transifex-username>
   Password: <transifex-password>
   ...
   Done.
   ```

4. Upload pot files to transifex service.

   Register pot files to `.tx/config` file:

[250] https://www.transifex.com/
[251] https://pypi.org/project/transifex-client/
[252] https://docs.transifex.com/client/introduction/
[253] https://www.transifex.com/

```
$ cd /your/document/root
$ sphinx-intl update-txconfig-resources --pot-dir _build/locale \
  --transifex-project-name sphinx-document-test_1_0
```

and upload pot files:

```
$ tx push -s
Pushing translations for resource sphinx-document-test_1_0.builders:
Pushing source file (locale/pot/builders.pot)
Resource does not exist.  Creating...
...
Done.
```

5. Forward the translation on transifex.
6. Pull translated po files and make translated HTML.

   Get translated catalogs and build mo files. For example, to build mo files for German (de):

```
$ cd /your/document/root
$ tx pull -l de
Pulling translations for resource sphinx-document-test_1_0.builders (...)
 -> de: locale/de/LC_MESSAGES/builders.po
...
Done.
```

   Invoke **`make html`** (for BSD/GNU make):

```
$ make -e SPHINXOPTS="-D language='de'" html
```

That's all!

Tip: Translating locally and on Transifex

If you want to push all language's po files, you can be done by using **`tx push -t`** command. Watch out! This operation overwrites translations in transifex.

In other words, if you have updated each in the service and local po files, it would take much time and effort to integrate them.

Contributing to Sphinx reference translation

The recommended way for new contributors to translate Sphinx reference is to join the translation team on Transifex.

There is a sphinx translation page[254] for Sphinx (master) documentation.

1. Login to transifex[255] service.
2. Go to sphinx translation page[256].
3. Click `Request language` and fill form.
4. Wait acceptance by transifex sphinx translation maintainers.
5. (After acceptance) Translate on transifex.

[254] https://www.transifex.com/sphinx-doc/sphinx-doc/
[255] https://www.transifex.com/
[256] https://www.transifex.com/sphinx-doc/sphinx-doc/

Detail is here: https://docs.transifex.com/getting-started-1/translators

1.10 Setuptools integration

Sphinx supports integration with setuptools and distutils through a custom command - `BuildDoc`.

Using setuptools integration

The Sphinx build can then be triggered from distutils, and some Sphinx options can be set in `setup.py` or `setup.cfg` instead of Sphinx's own configuration file.

For instance, from `setup.py`:

```
# this is only necessary when not using setuptools/distribute
from sphinx.setup_command import BuildDoc
cmdclass = {'build_sphinx': BuildDoc}

name = 'My project'
version = '1.2'
release = '1.2.0'
setup(
    name=name,
    author='Bernard Montgomery',
    version=release,
    cmdclass=cmdclass,
    # these are optional and override conf.py settings
    command_options={
        'build_sphinx': {
            'project': ('setup.py', name),
            'version': ('setup.py', version),
            'release': ('setup.py', release),
            'source_dir': ('setup.py', 'doc')}},
)
```

Note: If you set Sphinx options directly in the `setup()` command, replace hyphens in variable names with underscores. In the example above, `source-dir` becomes `source_dir`.

Or add this section in `setup.cfg`:

```
[build_sphinx]
project = 'My project'
version = 1.2
release = 1.2.0
source-dir = 'doc'
```

Once configured, call this by calling the relevant command on `setup.py`:

```
$ python setup.py build_sphinx
```

Options for setuptools integration

fresh-env

A boolean that determines whether the saved environment should be discarded on build. Default is false.

This can also be set by passing the *-E* flag to `setup.py`:

```
$ python setup.py build_sphinx -E
```

all-files

A boolean that determines whether all files should be built from scratch. Default is false.

This can also be set by passing the *-a* flag to `setup.py`:

```
$ python setup.py build_sphinx -a
```

source-dir

The target source directory. This can be relative to the `setup.py` or `setup.cfg` file, or it can be absolute. It defaults to `./doc` or `./docs` if either contains a file named `conf.py` (checking `./doc` first); otherwise it defaults to the current directory.

This can also be set by passing the *-s* flag to `setup.py`:

```
$ python setup.py build_sphinx -s $SOURCE_DIR
```

build-dir

The target build directory. This can be relative to the `setup.py` or `setup.cfg` file, or it can be absolute. Default is `./build/sphinx`.

config-dir

Location of the configuration directory. This can be relative to the `setup.py` or `setup.cfg` file, or it can be absolute. Default is to use *source-dir*.

This can also be set by passing the *-c* flag to `setup.py`:

```
$ python setup.py build_sphinx -c $CONFIG_DIR
```

New in version 1.0.

builder

The builder or list of builders to use. Default is `html`.

This can also be set by passing the *-b* flag to `setup.py`:

```
$ python setup.py build_sphinx -b $BUILDER
```

Changed in version 1.6: This can now be a comma- or space-separated list of builders

warning-is-error

A boolean that ensures Sphinx warnings will result in a failed build. Default is false.

This can also be set by passing the *-W* flag to `setup.py`:

```
$ python setup.py build_sphinx -W
```

New in version 1.5.

project

The documented project's name. Default is `''`.

New in version 1.0.

version
The short X.Y version. Default is `''`.

New in version 1.0.

release
The full version, including alpha/beta/rc tags. Default is `''`.

New in version 1.0.

today
How to format the current date, used as the replacement for `|today|`. Default is `''`.

New in version 1.0.

link-index
A boolean that ensures index.html will be linked to the master doc. Default is false.

This can also be set by passing the *-i* flag to `setup.py`:

```
$ python setup.py build_sphinx -i
```

New in version 1.0.

copyright
The copyright string. Default is `''`.

New in version 1.3.

nitpicky
Run in nit-picky mode. Currently, this generates warnings for all missing references. See the config value *`nitpick_ignore`* for a way to exclude some references as "known missing".

New in version 1.8.

pdb
A boolean to configure pdb on exception. Default is false.

New in version 1.5.

1.11 Sphinx Web Support

New in version 1.1.

Sphinx provides a Python API to easily integrate Sphinx documentation into your web application. To learn more read the *Web Support Quick Start*.

Web Support Quick Start

Building Documentation Data

To make use of the web support package in your application you'll need to build the data it uses. This data includes pickle files representing documents, search indices, and node data that is used to track where comments and other things are in a document. To do this you will need to create an instance of the *`WebSupport`* class and call its *`build()`* method:

```
from sphinxcontrib.websupport import WebSupport

support = WebSupport(srcdir='/path/to/rst/sources/',
```

(continues on next page)

(continued from previous page)

```
                     builddir='/path/to/build/outdir',
                     search='xapian')

support.build()
```

This will read reStructuredText sources from `srcdir` and place the necessary data in `builddir`. The `builddir` will contain two sub-directories: one named "data" that contains all the data needed to display documents, search through documents, and add comments to documents. The other directory will be called "static" and contains static files that should be served from "/static".

Note: If you wish to serve static files from a path other than "/static", you can do so by providing the *staticdir* keyword argument when creating the `WebSupport` object.

Integrating Sphinx Documents Into Your Webapp

Now that the data is built, it's time to do something useful with it. Start off by creating a `WebSupport` object for your application:

```
from sphinxcontrib.websupport import WebSupport

support = WebSupport(datadir='/path/to/the/data',
                     search='xapian')
```

You'll only need one of these for each set of documentation you will be working with. You can then call its `get_document()` method to access individual documents:

```
contents = support.get_document('contents')
```

This will return a dictionary containing the following items:

- **body**: The main body of the document as HTML
- **sidebar**: The sidebar of the document as HTML
- **relbar**: A div containing links to related documents
- **title**: The title of the document
- **css**: Links to CSS files used by Sphinx
- **script**: JavaScript containing comment options

This dict can then be used as context for templates. The goal is to be easy to integrate with your existing templating system. An example using Jinja2[258] is:

```
{%- extends "layout.html" %}

{%- block title %}
    {{ document.title }}
{%- endblock %}

{% block css %}
    {{ super() }}
```

(continues on next page)

[258] http://jinja.pocoo.org/

(continued from previous page)

```
    {{ document.css|safe }}
    <link rel="stylesheet" href="/static/websupport-custom.css" type="text/css">
{% endblock %}

{%- block script %}
    {{ super() }}
    {{ document.script|safe }}
{%- endblock %}

{%- block relbar %}
    {{ document.relbar|safe }}
{%- endblock %}

{%- block body %}
    {{ document.body|safe }}
{%- endblock %}

{%- block sidebar %}
    {{ document.sidebar|safe }}
{%- endblock %}
```

Authentication

To use certain features such as voting, it must be possible to authenticate users. The details of the authentication are left to your application. Once a user has been authenticated you can pass the user's details to certain `WebSupport` methods using the *username* and *moderator* keyword arguments. The web support package will store the username with comments and votes. The only caveat is that if you allow users to change their username you must update the websupport package's data:

```
support.update_username(old_username, new_username)
```

username should be a unique string which identifies a user, and *moderator* should be a boolean representing whether the user has moderation privileges. The default value for *moderator* is `False`.

An example Flask[259] function that checks whether a user is logged in and then retrieves a document is:

```
from sphinxcontrib.websupport.errors import *

@app.route('/<path:docname>')
def doc(docname):
    username = g.user.name if g.user else ''
    moderator = g.user.moderator if g.user else False
    try:
        document = support.get_document(docname, username, moderator)
    except DocumentNotFoundError:
        abort(404)
    return render_template('doc.html', document=document)
```

The first thing to notice is that the *docname* is just the request path. This makes accessing the correct document easy from a single view. If the user is authenticated, then the username and moderation status are passed along with the

[259] http://flask.pocoo.org/

docname to *get_document()*. The web support package will then add this data to the COMMENT_OPTIONS that are used in the template.

Note: This only works if your documentation is served from your document root. If it is served from another directory, you will need to prefix the url route with that directory, and give the *docroot* keyword argument when creating the web support object:

```
support = WebSupport(..., docroot='docs')

@app.route('/docs/<path:docname>')
```

Performing Searches

To use the search form built-in to the Sphinx sidebar, create a function to handle requests to the URL 'search' relative to the documentation root. The user's search query will be in the GET parameters, with the key *q*. Then use the *get_search_results()* method to retrieve search results. In Flask[260] that would be like this:

```
@app.route('/search')
def search():
    q = request.args.get('q')
    document = support.get_search_results(q)
    return render_template('doc.html', document=document)
```

Note that we used the same template to render our search results as we did to render our documents. That's because *get_search_results()* returns a context dict in the same format that *get_document()* does.

Comments & Proposals

Now that this is done it's time to define the functions that handle the AJAX calls from the script. You will need three functions. The first function is used to add a new comment, and will call the web support method *add_comment()*:

```
@app.route('/docs/add_comment', methods=['POST'])
def add_comment():
    parent_id = request.form.get('parent', '')
    node_id = request.form.get('node', '')
    text = request.form.get('text', '')
    proposal = request.form.get('proposal', '')
    username = g.user.name if g.user is not None else 'Anonymous'
    comment = support.add_comment(text, node_id='node_id',
                                  parent_id='parent_id',
                                  username=username, proposal=proposal)
    return jsonify(comment=comment)
```

You'll notice that both a parent_id and node_id are sent with the request. If the comment is being attached directly to a node, parent_id will be empty. If the comment is a child of another comment, then node_id will be empty. Then next function handles the retrieval of comments for a specific node, and is aptly named *get_data()*:

```
@app.route('/docs/get_comments')
def get_comments():
    username = g.user.name if g.user else None
```

(continues on next page)

[260] http://flask.pocoo.org/

(continued from previous page)

```
    moderator = g.user.moderator if g.user else False
    node_id = request.args.get('node', '')
    data = support.get_data(node_id, username, moderator)
    return jsonify(**data)
```

The final function that is needed will call *process_vote()*, and will handle user votes on comments:

```
@app.route('/docs/process_vote', methods=['POST'])
def process_vote():
    if g.user is None:
        abort(401)
    comment_id = request.form.get('comment_id')
    value = request.form.get('value')
    if value is None or comment_id is None:
        abort(400)
    support.process_vote(comment_id, g.user.id, value)
    return "success"
```

Comment Moderation

By default, all comments added through *add_comment()* are automatically displayed. If you wish to have some form of moderation, you can pass the `displayed` keyword argument:

```
comment = support.add_comment(text, node_id='node_id',
                              parent_id='parent_id',
                              username=username, proposal=proposal,
                              displayed=False)
```

You can then create a new view to handle the moderation of comments. It will be called when a moderator decides a comment should be accepted and displayed:

```
@app.route('/docs/accept_comment', methods=['POST'])
def accept_comment():
    moderator = g.user.moderator if g.user else False
    comment_id = request.form.get('id')
    support.accept_comment(comment_id, moderator=moderator)
    return 'OK'
```

Rejecting comments happens via comment deletion.

To perform a custom action (such as emailing a moderator) when a new comment is added but not displayed, you can pass callable to the *WebSupport* class when instantiating your support object:

```
def moderation_callback(comment):
    """Do something..."""

support = WebSupport(..., moderation_callback=moderation_callback)
```

The moderation callback must take one argument, which will be the same comment dict that is returned by `add_comment()`.

The WebSupport Class

class `sphinxcontrib.websupport.`**`WebSupport`**

The main API class for the web support package. All interactions with the web support package should occur through this class.

The class takes the following keyword arguments:

srcdir The directory containing reStructuredText source files.

builddir The directory that build data and static files should be placed in. This should be used when creating a *`WebSupport`* object that will be used to build data.

datadir The directory that the web support data is in. This should be used when creating a *`WebSupport`* object that will be used to retrieve data.

search This may contain either a string (e.g. 'xapian') referencing a built-in search adapter to use, or an instance of a subclass of *`BaseSearch`*.

storage This may contain either a string representing a database uri, or an instance of a subclass of *`StorageBackend`*. If this is not provided, a new sqlite database will be created.

moderation_callback A callable to be called when a new comment is added that is not displayed. It must accept one argument: a dictionary representing the comment that was added.

staticdir If the static files should be created in a different location **and not in** `'/static'`, this should be a string with the name of that location (e.g. `builddir + '/static_files'`).

Note: If you specify `staticdir`, you will typically want to adjust `staticroot` accordingly.

staticroot If the static files are not served from `'/static'`, this should be a string with the name of that location (e.g. `'/static_files'`).

docroot If the documentation is not served from the base path of a URL, this should be a string specifying that path (e.g. `'docs'`).

Changed in version 1.6: WebSupport class is moved to sphinxcontrib.websupport from sphinx.websupport. Please add `sphinxcontrib-websupport` package in your dependency and use moved class instead.

Methods

`WebSupport.`**`build`**`()`

Build the documentation. Places the data into the *outdir* directory. Use it like this:

```
support = WebSupport(srcdir, builddir, search='xapian')
support.build()
```

This will read reStructured text files from *srcdir*. Then it will build the pickles and search index, placing them into *builddir*. It will also save node data to the database.

`WebSupport.`**`get_document`**`(`*docname*, *username=''*, *moderator=False*`)`

Load and return a document from a pickle. The document will be a dict object which can be used to render a template:

```
support = WebSupport(datadir=datadir)
support.get_document('index', username, moderator)
```

In most cases *docname* will be taken from the request path and passed directly to this function. In Flask, that would be something like this:

```
@app.route('/<path:docname>')
def index(docname):
    username = g.user.name if g.user else ''
    moderator = g.user.moderator if g.user else False
    try:
        document = support.get_document(docname, username,
                                        moderator)
    except DocumentNotFoundError:
        abort(404)
    render_template('doc.html', document=document)
```

The document dict that is returned contains the following items to be used during template rendering.

- **body**: The main body of the document as HTML
- **sidebar**: The sidebar of the document as HTML
- **relbar**: A div containing links to related documents
- **title**: The title of the document
- **css**: Links to css files used by Sphinx
- **script**: Javascript containing comment options

This raises `DocumentNotFoundError` if a document matching *docname* is not found.

Parameters **`docname`** – the name of the document to load.

`WebSupport.`**`get_data`**(*node_id*, *username=None*, *moderator=False*)

Get the comments and source associated with *node_id*. If *username* is given vote information will be included with the returned comments. The default CommentBackend returns a dict with two keys, *source*, and *comments*. *source* is raw source of the node and is used as the starting point for proposals a user can add. *comments* is a list of dicts that represent a comment, each having the following items:

Key	Contents
text	The comment text.
user-name	The username that was stored with the comment.
id	The comment's unique identifier.
rat-ing	The comment's current rating.
age	The time in seconds since the comment was added.
time	A dict containing time information. It contains the following keys: year, month, day, hour, minute, second, iso, and delta. *iso* is the time formatted in ISO 8601 format. *delta* is a printable form of how old the comment is (e.g. "3 hours ago").
vote	If *user_id* was given, this will be an integer representing the vote. 1 for an upvote, -1 for a downvote, or 0 if unvoted.
node	The id of the node that the comment is attached to. If the comment's parent is another comment rather than a node, this will be null.
par-ent	The id of the comment that this comment is attached to if it is not attached to a node.
chil-dren	A list of all children, in this format.
pro-posal_diff	An HTML representation of the differences between the the current source and the user's proposed source.

Parameters

- **node_id** – the id of the node to get comments for.
- **username** – the username of the user viewing the comments.
- **moderator** – whether the user is a moderator.

WebSupport.**add_comment**(*text*, *node_id=''*, *parent_id=''*, *displayed=True*, *username=None*, *time=None*, *proposal=None*, *moderator=False*)

Add a comment to a node or another comment. Returns the comment in the same format as `get_comments()`. If the comment is being attached to a node, pass in the node's id (as a string) with the node keyword argument:

```
comment = support.add_comment(text, node_id=node_id)
```

If the comment is the child of another comment, provide the parent's id (as a string) with the parent keyword argument:

```
comment = support.add_comment(text, parent_id=parent_id)
```

If you would like to store a username with the comment, pass in the optional *username* keyword argument:

```
comment = support.add_comment(text, node=node_id,
                              username=username)
```

Parameters

- **parent_id** – the prefixed id of the comment's parent.
- **text** – the text of the comment.
- **displayed** – for moderation purposes
- **username** – the username of the user making the comment.
- **time** – the time the comment was created, defaults to now.

WebSupport.**process_vote**(*comment_id*, *username*, *value*)

Process a user's vote. The web support package relies on the API user to perform authentication. The API user will typically receive a comment_id and value from a form, and then make sure the user is authenticated. A unique username must be passed in, which will also be used to retrieve the user's past voting data. An example, once again in Flask:

```
@app.route('/docs/process_vote', methods=['POST'])
def process_vote():
    if g.user is None:
        abort(401)
    comment_id = request.form.get('comment_id')
    value = request.form.get('value')
    if value is None or comment_id is None:
        abort(400)
    support.process_vote(comment_id, g.user.name, value)
    return "success"
```

Parameters

- **comment_id** – the comment being voted on
- **username** – the unique username of the user voting
- **value** – 1 for an upvote, -1 for a downvote, 0 for an unvote.

WebSupport.**get_search_results**(*q*)

Perform a search for the query *q*, and create a set of search results. Then render the search results as html and return a context dict like the one created by *get_document()*:

```
document = support.get_search_results(q)
```

Parameters **q** – the search query

Search Adapters

To create a custom search adapter you will need to subclass the *BaseSearch* class. Then create an instance of the new class and pass that as the *search* keyword argument when you create the *WebSupport* object:

```
support = WebSupport(srcdir=srcdir,
                     builddir=builddir,
                     search=MySearch())
```

For more information about creating a custom search adapter, please see the documentation of the *BaseSearch* class below.

class sphinxcontrib.websupport.search.**BaseSearch**

Defines an interface for search adapters.

Changed in version 1.6: BaseSearch class is moved to sphinxcontrib.websupport.search from sphinx.websupport.search.

Methods

The following methods are defined in the BaseSearch class. Some methods do not need to be overridden, but some (*add_document()* and *handle_query()*) must be overridden in your subclass. For a working example, look at the built-in adapter for whoosh.

BaseSearch.**init_indexing**(*changed=[]*)

Called by the builder to initialize the search indexer. *changed* is a list of pagenames that will be reindexed. You may want to remove these from the search index before indexing begins.

Parameters **changed** – a list of pagenames that will be re-indexed

BaseSearch.**finish_indexing**()

Called by the builder when writing has been completed. Use this to perform any finalization or cleanup actions after indexing is complete.

BaseSearch.**feed**(*pagename*, *filename*, *title*, *doctree*)

Called by the builder to add a doctree to the index. Converts the *doctree* to text and passes it to *add_document()*. You probably won't want to override this unless you need access to the *doctree*. Override *add_document()* instead.

Parameters

- **pagename** – the name of the page to be indexed
- **filename** – the name of the original source file
- **title** – the title of the page to be indexed
- **doctree** – is the docutils doctree representation of the page

BaseSearch.**add_document**(*pagename*, *filename*, *title*, *text*)

Called by *feed()* to add a document to the search index. This method should should do everything necessary to add a single document to the search index.

pagename is name of the page being indexed. It is the combination of the source files relative path and filename, minus the extension. For example, if the source file is "ext/builders.rst", the *pagename* would be "ext/builders". This will need to be returned with search results when processing a query.

Parameters

- **pagename** – the name of the page being indexed
- **filename** – the name of the original source file
- **title** – the page's title
- **text** – the full text of the page

BaseSearch.**query**(*q*)

Called by the web support api to get search results. This method compiles the regular expression to be used when *extracting context*, then calls *handle_query()*. You won't want to override this unless you don't want to use the included *extract_context()* method. Override *handle_query()* instead.

Parameters **q** – the search query string.

BaseSearch.**handle_query**(*q*)

Called by *query()* to retrieve search results for a search query *q*. This should return an iterable containing tuples of the following format:

```
(<path>, <title>, <context>)
```

path and *title* are the same values that were passed to *add_document()*, and *context* should be a short text snippet of the text surrounding the search query in the document.

The *extract_context()* method is provided as a simple way to create the *context*.

Parameters **q** – the search query

BaseSearch.**extract_context**(*text*, *length=240*)

Extract the context for the search query from the document's full *text*.

Parameters

- **text** – the full text of the document to create the context for
- **length** – the length of the context snippet to return.

Storage Backends

To create a custom storage backend you will need to subclass the *StorageBackend* class. Then create an instance of the new class and pass that as the *storage* keyword argument when you create the *WebSupport* object:

```
support = WebSupport(srcdir=srcdir,
                     builddir=builddir,
                     storage=MyStorage())
```

For more information about creating a custom storage backend, please see the documentation of the *StorageBackend* class below.

class sphinxcontrib.websupport.storage.**StorageBackend**

Defines an interface for storage backends.

Changed in version 1.6: StorageBackend class is moved to sphinxcontrib.websupport.storage from sphinx.websupport.storage.

Methods

StorageBackend.**pre_build**()

Called immediately before the build process begins. Use this to prepare the StorageBackend for the addition of nodes.

StorageBackend.**add_node**(*id*, *document*, *source*)

Add a node to the StorageBackend.

Parameters

- **id** – a unique id for the comment.
- **document** – the name of the document the node belongs to.
- **source** – the source files name.

StorageBackend.**post_build**()

Called after a build has completed. Use this to finalize the addition of nodes if needed.

StorageBackend.**add_comment**(*text*, *displayed*, *username*, *time*, *proposal*, *node_id*, *parent_id*, *moderator*)

Called when a comment is being added.

Parameters

- **text** – the text of the comment
- **displayed** – whether the comment should be displayed
- **username** – the name of the user adding the comment
- **time** – a date object with the time the comment was added
- **proposal** – the text of the proposal the user made
- **node_id** – the id of the node that the comment is being added to
- **parent_id** – the id of the comment's parent comment.
- **moderator** – whether the user adding the comment is a moderator

StorageBackend.**delete_comment**(*comment_id*, *username*, *moderator*)

Delete a comment.

Raises `UserNotAuthorizedError` if moderator is False and *username* doesn't match the username on the comment.

Parameters

- **comment_id** – The id of the comment being deleted.
- **username** – The username of the user requesting the deletion.
- **moderator** – Whether the user is a moderator.

StorageBackend.**get_data**(*node_id*, *username*, *moderator*)

Called to retrieve all data for a node. This should return a dict with two keys, *source* and *comments* as described by `WebSupport`'s `get_data()` method.

Parameters

- **node_id** – The id of the node to get data for.
- **username** – The name of the user requesting the data.
- **moderator** – Whether the requestor is a moderator.

StorageBackend.**process_vote**(*comment_id*, *username*, *value*)

Process a vote that is being cast. *value* will be either -1, 0, or 1.

Parameters

- **`comment_id`** – The id of the comment being voted on.
- **`username`** – The username of the user casting the vote.
- **`value`** – The value of the vote being cast.

`StorageBackend.`**`update_username`**`(`*old_username*, *new_username*`)`

If a user is allowed to change their username this method should be called so that there is not stagnate data in the storage system.

Parameters

- **`old_username`** – The username being changed.
- **`new_username`** – What the username is being changed to.

`StorageBackend.`**`accept_comment`**`(`*comment_id*`)`

Called when a moderator accepts a comment. After the method is called the comment should be displayed to all users.

Parameters **`comment_id`** – The id of the comment being accepted.

CHAPTER

TWO

EXTENDING SPHINX

This guide is aimed at giving a quick introduction for those wishing to develop their own extensions for Sphinx. Sphinx possesses significant extensibility capabilities including the ability to hook into almost every point of the build process. If you simply wish to use Sphinx with existing extensions, refer to *Using Sphinx*. For a more detailed discussion of the extension interface see *Developing extensions for Sphinx*.

2.1 Developing extensions overview

This page contains general information about developing Sphinx extensions.

Make an extension depend on another extension

Sometimes your extension depends on the functionality of another Sphinx extension. Most Sphinx extensions are activated in a project's `conf.py` file, but this is not available to you as an extension developer.

To ensure that another extension is activated as a part of your own extension, use the *`Sphinx.setup_extension()`* method. This will activate another extension at run-time, ensuring that you have access to its functionality.

For example, the following code activates the `recommonmark` extension:

```
def setup(app):
    app.setup_extension("recommonmark")
```

Note: Since your extension will depend on another, make sure to include it as a part of your extension's installation requirements.

2.2 Extension tutorials

Refer to the following tutorials to get started with extension development.

Developing a "Hello world" extension

The objective of this tutorial is to create a very basic extension that adds a new directive. This directive will output a paragraph containing "hello world".

Only basic information is provided in this tutorial. For more information, refer to the *other tutorials* that go into more details.

Warning: For this extension, you will need some basic understanding of docutils[261] and Python.

Overview

We want the extension to add the following to Sphinx:

- A `helloworld` directive, that will simply output the text "hello world".

Prerequisites

We will not be distributing this plugin via PyPI[262] and will instead include it as part of an existing project. This means you will need to use an existing project or create a new one using **`sphinx-quickstart`**.

We assume you are using separate source (`source`) and build (`build`) folders. Your extension file could be in any folder of your project. In our case, let's do the following:

1. Create an `_ext` folder in `source`
2. Create a new Python file in the `_ext` folder called `helloworld.py`

Here is an example of the folder structure you might obtain:

```
└── source
    ├── _ext
    │   └── helloworld.py
    ├── _static
    ├── conf.py
    ├── somefolder
    ├── index.rst
    ├── somefile.rst
    └── someotherfile.rst
```

Writing the extension

Open `helloworld.py` and paste the following code in it:

```
from docutils import nodes
from docutils.parsers.rst import Directive


class HelloWorld(Directive):

    def run(self):
```

(continues on next page)

261 http://docutils.sourceforge.net/
262 https://pypi.org/

(continued from previous page)

```python
        paragraph_node = nodes.paragraph(text='Hello World!')
        return [paragraph_node]


def setup(app):
    app.add_directive("helloworld", HelloWorld)

    return {
        'version': '0.1',
        'parallel_read_safe': True,
        'parallel_write_safe': True,
    }
```

Some essential things are happening in this example, and you will see them for all directives.

The directive class

Our new directive is declared in the `HelloWorld` class.

```python
class HelloWorld(Directive):

    def run(self):
        paragraph_node = nodes.paragraph(text='Hello World!')
        return [paragraph_node]
```

This class extends the docutils[263]' `Directive` class. All extensions that create directives should extend this class.

See also:

The docutils documentation on creating directives[264]

This class contains a `run` method. This method is a requirement and it is part of every directive. It contains the main logic of the directive and it returns a list of docutils nodes to be processed by Sphinx. These nodes are docutils' way of representing the content of a document. There are many types of nodes available: text, paragraph, reference, table, etc.

See also:

The docutils documentation on nodes[265]

The `nodes.paragraph` class creates a new paragraph node. A paragraph node typically contains some text that we can set during instantiation using the `text` parameter.

The `setup` function

This function is a requirement. We use it to plug our new directive into Sphinx.

```python
def setup(app):
    app.add_directive("helloworld", HelloWorld)

    return {
        'version': '0.1',
        'parallel_read_safe': True,
```

(continues on next page)

[263] http://docutils.sourceforge.net/
[264] http://docutils.sourceforge.net/docs/howto/rst-directives.html
[265] http://docutils.sourceforge.net/docs/ref/doctree.html

(continued from previous page)

```
        'parallel_write_safe': True,
    }
```

The simplest thing you can do it call the *add_directive()* method, which is what we've done here. For this particular call, the first argument is the name of the directive itself as used in a reST file. In this case, we would use `helloworld`. For example:

```
Some intro text here...

.. helloworld::

Some more text here...
```

We also return the *extension metadata* that indicates the version of our extension, along with the fact that it is safe to use the extension for both parallel reading and writing.

Using the extension

The extension has to be declared in your `conf.py` file to make Sphinx aware of it. There are two steps necessary here:

1. Add the `_ext` directory to the Python path[266] using `sys.path.append`. This should be placed at the top of the file.
2. Update or create the *extensions* list and add the extension file name to the list

For example:

```
import os
import sys

sys.path.append(os.path.abspath("./_ext"))

extensions = ['helloworld']
```

Tip: We're not distributing this extension as a Python package[267], we need to modify the Python path[268] so Sphinx can find our extension. This is why we need the call to `sys.path.append`.

You can now use the extension in a file. For example:

```
Some intro text here...

.. helloworld::

Some more text here...
```

The sample above would generate:

```
Some intro text here...

Hello World!
```

(continues on next page)

[266] https://docs.python.org/3/using/cmdline.html#envvar-PYTHONPATH
[267] https://packaging.python.org/
[268] https://docs.python.org/3/using/cmdline.html#envvar-PYTHONPATH

(continued from previous page)

```

Some more text here...
```

Further reading

This is the very basic principle of an extension that creates a new directive.

For a more advanced example, refer to *Developing a "TODO" extension*.

Developing a "TODO" extension

The objective of this tutorial is to create a more comprehensive extension than that created in *Developing a "Hello world" extension*. Whereas that guide just covered writing a custom *directive*, this guide adds multiple directives, along with custom nodes, additional config values and custom event handlers. To this end, we will cover a `todo` extension that adds capabilities to include todo entries in the documentation, and to collect these in a central place. This is similar the `sphinxext.todo` extension distributed with Sphinx.

Overview

Note: To understand the design of this extension, refer to *Important objects* and *Build Phases*.

We want the extension to add the following to Sphinx:

- A `todo` directive, containing some content that is marked with "TODO" and only shown in the output if a new config value is set. Todo entries should not be in the output by default.
- A `todolist` directive that creates a list of all todo entries throughout the documentation.

For that, we will need to add the following elements to Sphinx:

- New directives, called `todo` and `todolist`.
- New document tree nodes to represent these directives, conventionally also called `todo` and `todolist`. We wouldn't need new nodes if the new directives only produced some content representable by existing nodes.
- A new config value `todo_include_todos` (config value names should start with the extension name, in order to stay unique) that controls whether todo entries make it into the output.
- New event handlers: one for the *`doctree-resolved`* event, to replace the todo and todolist nodes, one for *`env-merge-info`* to merge intermediate results from parallel builds, and one for *`env-purge-doc`* (the reason for that will be covered later).

Prerequisites

As with *Developing a "Hello world" extension*, we will not be distributing this plugin via PyPI so once again we need a Sphinx project to call this from. You can use an existing project or create a new one using **`sphinx-quickstart`**.

We assume you are using separate source (`source`) and build (`build`) folders. Your extension file could be in any folder of your project. In our case, let's do the following:

1. Create an `_ext` folder in `source`
2. Create a new Python file in the `_ext` folder called `todo.py`

Here is an example of the folder structure you might obtain:

```
└── source
    ├── _ext
    │   └── todo.py
    ├── _static
    ├── conf.py
    ├── somefolder
    ├── index.rst
    ├── somefile.rst
    └── someotherfile.rst
```

Writing the extension

Open todo.py and paste the following code in it, all of which we will explain in detail shortly:

```python
from docutils import nodes
from docutils.parsers.rst import Directive

from sphinx.locale import _
from sphinx.util.docutils import SphinxDirective


class todo(nodes.Admonition, nodes.Element):
    pass


class todolist(nodes.General, nodes.Element):
    pass


def visit_todo_node(self, node):
    self.visit_admonition(node)


def depart_todo_node(self, node):
    self.depart_admonition(node)


class TodolistDirective(Directive):

    def run(self):
        return [todolist('')]


class TodoDirective(SphinxDirective):

    # this enables content in the directive
    has_content = True

    def run(self):
        targetid = 'todo-%d' % self.env.new_serialno('todo')
        targetnode = nodes.target('', '', ids=[targetid])

```

(continues on next page)

(continued from previous page)

```
        todo_node = todo('\n'.join(self.content))
        todo_node += nodes.title(_('Todo'), _('Todo'))
        self.state.nested_parse(self.content, self.content_offset, todo_node)

        if not hasattr(self.env, 'todo_all_todos'):
            self.env.todo_all_todos = []

        self.env.todo_all_todos.append({
            'docname': self.env.docname,
            'lineno': self.lineno,
            'todo': todo_node.deepcopy(),
            'target': targetnode,
        })

        return [targetnode, todo_node]


def purge_todos(app, env, docname):
    if not hasattr(env, 'todo_all_todos'):
        return

    env.todo_all_todos = [todo for todo in env.todo_all_todos
                          if todo['docname'] != docname]


def merge_todos(app, env, docnames, other):
    if not hasattr(env, 'todo_all_todos'):
        env.todo_all_todos = []
    if hasattr(other, 'todo_all_todos'):
        env.todo_all_todos.extend(other.todo_all_todos)


def process_todo_nodes(app, doctree, fromdocname):
    if not app.config.todo_include_todos:
        for node in doctree.traverse(todo):
            node.parent.remove(node)

    # Replace all todolist nodes with a list of the collected todos.
    # Augment each todo with a backlink to the original location.
    env = app.builder.env

    if not hasattr(env, 'todo_all_todos'):
        env.todo_all_todos = []

    for node in doctree.traverse(todolist):
        if not app.config.todo_include_todos:
            node.replace_self([])
            continue

        content = []

        for todo_info in env.todo_all_todos:
```

(continues on next page)

(continued from previous page)

```
            para = nodes.paragraph()
            filename = env.doc2path(todo_info['docname'], base=None)
            description = (
                _('(The original entry is located in %s, line %d and can be found ') %
                (filename, todo_info['lineno']))
            para += nodes.Text(description, description)

            # Create a reference
            newnode = nodes.reference('', '')
            innernode = nodes.emphasis(_('here'), _('here'))
            newnode['refdocname'] = todo_info['docname']
            newnode['refuri'] = app.builder.get_relative_uri(
                fromdocname, todo_info['docname'])
            newnode['refuri'] += '#' + todo_info['target']['refid']
            newnode.append(innernode)
            para += newnode
            para += nodes.Text('.)', '.)')

            # Insert into the todolist
            content.append(todo_info['todo'])
            content.append(para)

        node.replace_self(content)


def setup(app):
    app.add_config_value('todo_include_todos', False, 'html')

    app.add_node(todolist)
    app.add_node(todo,
                 html=(visit_todo_node, depart_todo_node),
                 latex=(visit_todo_node, depart_todo_node),
                 text=(visit_todo_node, depart_todo_node))

    app.add_directive('todo', TodoDirective)
    app.add_directive('todolist', TodolistDirective)
    app.connect('doctree-resolved', process_todo_nodes)
    app.connect('env-purge-doc', purge_todos)
    app.connect('env-merge-info', merge_todos)

    return {
        'version': '0.1',
        'parallel_read_safe': True,
        'parallel_write_safe': True,
    }
```

This is far more extensive extension than the one detailed in *Developing a “Hello world” extension*, however, we will will look at each piece step-by-step to explain what’s happening.

The node classes

Let's start with the node classes:

```
class todo(nodes.Admonition, nodes.Element):
    pass


class todolist(nodes.General, nodes.Element):
    pass


def visit_todo_node(self, node):
    self.visit_admonition(node)


def depart_todo_node(self, node):
    self.depart_admonition(node)
```

Node classes usually don't have to do anything except inherit from the standard docutils classes defined in `docutils.nodes`. `todo` inherits from `Admonition` because it should be handled like a note or warning, `todolist` is just a "general" node.

Note: Many extensions will not have to create their own node classes and work fine with the nodes already provided by docutils[269] and *Sphinx*.

Attention: It is important to know that while you can extend Sphinx without leaving your `conf.py`, if you declare an inherited node right there, you'll hit an unobvious `PickleError`. So if something goes wrong, please make sure that you put inherited nodes into a separate Python module.

For more details, see:

- https://github.com/sphinx-doc/sphinx/issues/6751
- https://github.com/sphinx-doc/sphinx/issues/1493
- https://github.com/sphinx-doc/sphinx/issues/1424

The directive classes

A directive class is a class deriving usually from *`docutils.parsers.rst.Directive`*. The directive interface is also covered in detail in the docutils documentation[270]; the important thing is that the class should have attributes that configure the allowed markup, and a `run` method that returns a list of nodes.

Looking first at the `TodolistDirective` directive:

```
class TodolistDirective(Directive):

    def run(self):
        return [todolist('')]
```

[269] http://docutils.sourceforge.net/docs/ref/doctree.html
[270] http://docutils.sourceforge.net/docs/ref/rst/directives.html

It's very simple, creating and returning an instance of our `todolist` node class. The `TodolistDirective` directive itself has neither content nor arguments that need to be handled. That brings us to the `TodoDirective` directive:

```python
class TodoDirective(SphinxDirective):

    # this enables content in the directive
    has_content = True

    def run(self):
        targetid = 'todo-%d' % self.env.new_serialno('todo')
        targetnode = nodes.target('', '', ids=[targetid])

        todo_node = todo('\n'.join(self.content))
        todo_node += nodes.title(_('Todo'), _('Todo'))
        self.state.nested_parse(self.content, self.content_offset, todo_node)

        if not hasattr(self.env, 'todo_all_todos'):
            self.env.todo_all_todos = []

        self.env.todo_all_todos.append({
            'docname': self.env.docname,
            'lineno': self.lineno,
            'todo': todo_node.deepcopy(),
            'target': targetnode,
        })

        return [targetnode, todo_node]
```

Several important things are covered here. First, as you can see, we're now subclassing the *`SphinxDirective`* helper class instead of the usual *`Directive`* class. This gives us access to the *build environment instance* using the `self.env` property. Without this, we'd have to use the rather convoluted `self.state.document.settings.env`. Then, to act as a link target (from `TodolistDirective`), the `TodoDirective` directive needs to return a target node in addition to the `todo` node. The target ID (in HTML, this will be the anchor name) is generated by using `env.new_serialno` which returns a new unique integer on each call and therefore leads to unique target names. The target node is instantiated without any text (the first two arguments).

On creating admonition node, the content body of the directive are parsed using `self.state.nested_parse`. The first argument gives the content body, and the second one gives content offset. The third argument gives the parent node of parsed result, in our case the `todo` node. Following this, the `todo` node is added to the environment. This is needed to be able to create a list of all todo entries throughout the documentation, in the place where the author puts a `todolist` directive. For this case, the environment attribute `todo_all_todos` is used (again, the name should be unique, so it is prefixed by the extension name). It does not exist when a new environment is created, so the directive must check and create it if necessary. Various information about the todo entry's location are stored along with a copy of the node.

In the last line, the nodes that should be put into the doctree are returned: the target node and the admonition node.

The node structure that the directive returns looks like this:

```
+--------------------+
| target node        |
+--------------------+
+--------------------+
| todo node          |
+--------------------+
```

(continues on next page)

(continued from previous page)

```
  \__+--------------------+
     | admonition title   |
     +--------------------+
     | paragraph          |
     +--------------------+
     | ...                |
     +--------------------+
```

The event handlers

Event handlers are one of Sphinx's most powerful features, providing a way to do hook into any part of the documentation process. There are many events provided by Sphinx itself, as detailed in *the API guide*, and we're going to use a subset of them here.

Let's look at the event handlers used in the above example. First, the one for the `env-purge-doc` event:

```python
def purge_todos(app, env, docname):
    if not hasattr(env, 'todo_all_todos'):
        return

    env.todo_all_todos = [todo for todo in env.todo_all_todos
                          if todo['docname'] != docname]
```

Since we store information from source files in the environment, which is persistent, it may become out of date when the source file changes. Therefore, before each source file is read, the environment's records of it are cleared, and the `env-purge-doc` event gives extensions a chance to do the same. Here we clear out all todos whose docname matches the given one from the `todo_all_todos` list. If there are todos left in the document, they will be added again during parsing.

The next handler, for the `env-merge-info` event, is used during parallel builds. As during parallel builds all threads have their own `env`, there's multiple `todo_all_todos` lists that need to be merged:

```python
def merge_todos(app, env, docnames, other):
    if not hasattr(env, 'todo_all_todos'):
        env.todo_all_todos = []
    if hasattr(other, 'todo_all_todos'):
        env.todo_all_todos.extend(other.todo_all_todos)
```

The other handler belongs to the `doctree-resolved` event:

```python
def process_todo_nodes(app, doctree, fromdocname):
    if not app.config.todo_include_todos:
        for node in doctree.traverse(todo):
            node.parent.remove(node)

    # Replace all todolist nodes with a list of the collected todos.
    # Augment each todo with a backlink to the original location.
    env = app.builder.env

    if not hasattr(env, 'todo_all_todos'):
        env.todo_all_todos = []

    for node in doctree.traverse(todolist):
```

(continues on next page)

(continued from previous page)

```
        if not app.config.todo_include_todos:
            node.replace_self([])
            continue

        content = []

        for todo_info in env.todo_all_todos:
            para = nodes.paragraph()
            filename = env.doc2path(todo_info['docname'], base=None)
            description = (
                _('(The original entry is located in %s, line %d and can be found ') %
                (filename, todo_info['lineno']))
            para += nodes.Text(description, description)

            # Create a reference
            newnode = nodes.reference('', '')
            innernode = nodes.emphasis(_('here'), _('here'))
            newnode['refdocname'] = todo_info['docname']
            newnode['refuri'] = app.builder.get_relative_uri(
                fromdocname, todo_info['docname'])
            newnode['refuri'] += '#' + todo_info['target']['refid']
            newnode.append(innernode)
            para += newnode
            para += nodes.Text('.)', '.)')

            # Insert into the todolist
            content.append(todo_info['todo'])
            content.append(para)

        node.replace_self(content)
```

The *doctree-resolved* event is emitted at the end of *phase 3 (resolving)* and allows custom resolving to be done. The handler we have written for this event is a bit more involved. If the `todo_include_todos` config value (which we'll describe shortly) is false, all `todo` and `todolist` nodes are removed from the documents. If not, `todo` nodes just stay where and how they are. `todolist` nodes are replaced by a list of todo entries, complete with backlinks to the location where they come from. The list items are composed of the nodes from the `todo` entry and docutils nodes created on the fly: a paragraph for each entry, containing text that gives the location, and a link (reference node containing an italic node) with the backreference. The reference URI is built by *sphinx.builders.Builder.get_relative_uri()* which creates a suitable URI depending on the used builder, and appending the todo node's (the target's) ID as the anchor name.

The `setup` function

As noted *previously*, the `setup` function is a requirement and is used to plug directives into Sphinx. However, we also use it to hook up the other parts of our extension. Let's look at our `setup` function:

```
def setup(app):
    app.add_config_value('todo_include_todos', False, 'html')

    app.add_node(todolist)
    app.add_node(todo,
                 html=(visit_todo_node, depart_todo_node),
```

(continues on next page)

(continued from previous page)

```
                 latex=(visit_todo_node, depart_todo_node),
                 text=(visit_todo_node, depart_todo_node))

    app.add_directive('todo', TodoDirective)
    app.add_directive('todolist', TodolistDirective)
    app.connect('doctree-resolved', process_todo_nodes)
    app.connect('env-purge-doc', purge_todos)
    app.connect('env-merge-info', merge_todos)

    return {
        'version': '0.1',
        'parallel_read_safe': True,
        'parallel_write_safe': True,
    }
```

The calls in this function refer to the classes and functions we added earlier. What the individual calls do is the following:

- *add_config_value()* lets Sphinx know that it should recognize the new *config value* `todo_include_todos`, whose default value should be `False` (this also tells Sphinx that it is a boolean value).

  If the third argument was `'html'`, HTML documents would be full rebuild if the config value changed its value. This is needed for config values that influence reading (build *phase 1 (reading)*).

- *add_node()* adds a new *node class* to the build system. It also can specify visitor functions for each supported output format. These visitor functions are needed when the new nodes stay until *phase 4 (writing)*. Since the `todolist` node is always replaced in *phase 3 (resolving)*, it doesn't need any.

- *add_directive()* adds a new *directive*, given by name and class.

- Finally, *connect()* adds an *event handler* to the event whose name is given by the first argument. The event handler function is called with several arguments which are documented with the event.

With this, our extension is complete.

Using the extension

As before, we need to enable the extension by declaring it in our `conf.py` file. There are two steps necessary here:

1. Add the `_ext` directory to the Python path[271] using `sys.path.append`. This should be placed at the top of the file.

2. Update or create the *extensions* list and add the extension file name to the list

In addition, we may wish to set the `todo_include_todos` config value. As noted above, this defaults to `False` but we can set it explicitly.

For example:

```
import os
import sys

sys.path.append(os.path.abspath("./_ext"))

extensions = ['todo']

todo_include_todos = False
```

[271] https://docs.python.org/3/using/cmdline.html#envvar-PYTHONPATH

You can now use the extension throughout your project. For example:

Listing 1: index.rst

```
Hello, world
============

.. toctree::
   somefile.rst
   someotherfile.rst

Hello world. Below is the list of TODOs.

.. todolist::
```

Listing 2: somefile.rst

```
foo
===

Some intro text here...

.. todo:: Fix this
```

Listing 3: someotherfile.rst

```
bar
===

Some more text here...

.. todo:: Fix that
```

Because we have configured `todo_include_todos` to `False`, we won't actually see anything rendered for the `todo` and `todolist` directives. However, if we toggle this to true, we will see the output described previously.

Further reading

For more information, refer to the docutils[272] documentation and *Developing extensions for Sphinx*.

Developing a "recipe" extension

The objective of this tutorial is to illustrate roles, directives and domains. Once complete, we will be able to use this extension to describe a recipe and reference that recipe from elsewhere in our documentation.

Note: This tutorial is based on a guide first published on opensource.com[273] and is provided here with the original author's permission.

[272] http://docutils.sourceforge.net/docs/
[273] https://opensource.com/article/18/11/building-custom-workflows-sphinx

Overview

We want the extension to add the following to Sphinx:

- A `recipe` *directive*, containing some content describing the recipe steps, along with a `:contains:` option highlighting the main ingredients of the recipe.
- A `ref` *role*, which provides a cross-reference to the recipe itself.
- A `recipe` *domain*, which allows us to tie together the above role and domain, along with things like indices.

For that, we will need to add the following elements to Sphinx:

- A new directive called `recipe`
- New indexes to allow us to reference ingredient and recipes
- A new domain called `recipe`, which will contain the `recipe` directive and `ref` role

Prerequisites

We need the same setup as in *the previous extensions*. This time, we will be putting out extension in a file called `recipe.py`.

Here is an example of the folder structure you might obtain:

```
└── source
    ├── _ext
    │   └── recipe.py
    ├── conf.py
    └── index.rst
```

Writing the extension

Open `recipe.py` and paste the following code in it, all of which we will explain in detail shortly:

```python
from collections import defaultdict

from docutils.parsers.rst import directives

from sphinx import addnodes
from sphinx.directives import ObjectDescription
from sphinx.domains import Domain, Index
from sphinx.roles import XRefRole
from sphinx.util.nodes import make_refnode


class RecipeDirective(ObjectDescription):
    """A custom directive that describes a recipe."""

    has_content = True
    required_arguments = 1
    option_spec = {
        'contains': directives.unchanged_required,
    }

    def handle_signature(self, sig, signode):
```

(continues on next page)

(continued from previous page)

```
        signode += addnodes.desc_name(text=sig)
        return sig

    def add_target_and_index(self, name_cls, sig, signode):
        signode['ids'].append('recipe' + '-' + sig)
        if 'contains' not in self.options:
            ingredients = [
                x.strip() for x in self.options.get('contains').split(',')]

            recipes = self.env.get_domain('recipe')
            recipes.add_recipe(sig, ingredients)


class IngredientIndex(Index):
    """A custom index that creates an ingredient matrix."""

    name = 'ingredient'
    localname = 'Ingredient Index'
    shortname = 'Ingredient'

    def generate(self, docnames=None):
        content = defaultdict(list)

        recipes = {name: (dispname, typ, docname, anchor)
                   for name, dispname, typ, docname, anchor, _
                   in self.domain.get_objects()}
        recipe_ingredients = self.domain.data['recipe_ingredients']
        ingredient_recipes = defaultdict(list)

        # flip from recipe_ingredients to ingredient_recipes
        for recipe_name, ingredients in recipe_ingredients.items():
            for ingredient in ingredients:
                ingredient_recipes[ingredient].append(recipe_name)

        # convert the mapping of ingredient to recipes to produce the expected
        # output, shown below, using the ingredient name as a key to group
        #
        # name, subtype, docname, anchor, extra, qualifier, description
        for ingredient, recipe_names in ingredient_recipes.items():
            for recipe_name in recipe_names:
                dispname, typ, docname, anchor = recipes[recipe_name]
                content[ingredient].append(
                    (dispname, 0, docname, anchor, docname, '', typ))

        # convert the dict to the sorted list of tuples expected
        content = sorted(content.items())

        return content, True


class RecipeIndex(Index):
    """A custom index that creates an recipe matrix."""
```

(continues on next page)

(continued from previous page)

```

    name = 'recipe'
    localname = 'Recipe Index'
    shortname = 'Recipe'

    def generate(self, docnames=None):
        content = defaultdict(list)

        # sort the list of recipes in alphabetical order
        recipes = self.domain.get_objects()
        recipes = sorted(recipes, key=lambda recipe: recipe[0])

        # generate the expected output, shown below, from the above using the
        # first letter of the recipe as a key to group thing
        #
        # name, subtype, docname, anchor, extra, qualifier, description
        for name, dispname, typ, docname, anchor, _ in recipes:
            content[dispname[0].lower()].append(
                (dispname, 0, docname, anchor, docname, '', typ))

        # convert the dict to the sorted list of tuples expected
        content = sorted(content.items())

        return content, True


class RecipeDomain(Domain):

    name = 'recipe'
    label = 'Recipe Sample'
    roles = {
        'ref': XRefRole()
    }
    directives = {
        'recipe': RecipeDirective,
    }
    indices = {
        RecipeIndex,
        IngredientIndex
    }
    initial_data = {
        'recipes': [],  # object list
        'recipe_ingredients': {},  # name -> object
    }

    def get_full_qualified_name(self, node):
        return '{}.{}'.format('recipe', node.arguments[0])

    def get_objects(self):
        for obj in self.data['recipes']:
            yield(obj)

```

(continues on next page)

(continued from previous page)

```python
    def resolve_xref(self, env, fromdocname, builder, typ, target, node,
                     contnode):
        match = [(docname, anchor)
                 for name, sig, typ, docname, anchor, prio
                 in self.get_objects() if sig == target]

        if len(match) > 0:
            todocname = match[0][0]
            targ = match[0][1]

            return make_refnode(builder, fromdocname, todocname, targ,
                                contnode, targ)
        else:
            print('Awww, found nothing')
            return None

    def add_recipe(self, signature, ingredients):
        """Add a new recipe to the domain."""
        name = '{}.{}'.format('recipe', signature)
        anchor = 'recipe-{}'.format(signature)

        self.data['recipe_ingredients'][name] = ingredients
        # name, dispname, type, docname, anchor, priority
        self.data['recipes'].append(
            (name, signature, 'Recipe', self.env.docname, anchor, 0))


def setup(app):
    app.add_domain(RecipeDomain)

    return {
        'version': '0.1',
        'parallel_read_safe': True,
        'parallel_write_safe': True,
    }
```

Let's look at each piece of this extension step-by-step to explain what's going on.

The directive class

The first thing to examine is the `RecipeDirective` directive:

```python
class RecipeDirective(ObjectDescription):
    """A custom directive that describes a recipe."""

    has_content = True
    required_arguments = 1
    option_spec = {
        'contains': directives.unchanged_required,
    }

    def handle_signature(self, sig, signode):
```

(continues on next page)

(continued from previous page)

```python
        signode += addnodes.desc_name(text=sig)
        return sig

    def add_target_and_index(self, name_cls, sig, signode):
        signode['ids'].append('recipe' + '-' + sig)
        if 'contains' not in self.options:
            ingredients = [
                x.strip() for x in self.options.get('contains').split(',')]

            recipes = self.env.get_domain('recipe')
            recipes.add_recipe(sig, ingredients)
```

Unlike *Developing a "Hello world" extension* and *Developing a "TODO" extension*, this directive doesn't derive from `docutils.parsers.rst.Directive` and doesn't define a `run` method. Instead, it derives from `sphinx.directives.ObjectDescription` and defines `handle_signature` and `add_target_and_index` methods. This is because `ObjectDescription` is a special-purpose directive that's intended for describing things like classes, functions, or, in our case, recipes. More specifically, `handle_signature` implements parsing the signature of the directive and passes on the object's name and type to its superclass, while `add_taget_and_index` adds a target (to link to) and an entry to the index for this node.

We also see that this directive defines `has_content`, `required_arguments` and `option_spec`. Unlike the `TodoDirective` directive added in the *previous tutorial*, this directive takes a single argument, the recipe name, and an option, `contains`, in addition to the nested reStructuredText in the body.

The index classes

Todo: Add brief overview of indices

```python
class IngredientIndex(Index):
    """A custom index that creates an ingredient matrix."""

    name = 'ingredient'
    localname = 'Ingredient Index'
    shortname = 'Ingredient'

    def generate(self, docnames=None):
        content = defaultdict(list)

        recipes = {name: (dispname, typ, docname, anchor)
                   for name, dispname, typ, docname, anchor, _
                   in self.domain.get_objects()}
        recipe_ingredients = self.domain.data['recipe_ingredients']
        ingredient_recipes = defaultdict(list)

        # flip from recipe_ingredients to ingredient_recipes
        for recipe_name, ingredients in recipe_ingredients.items():
            for ingredient in ingredients:
                ingredient_recipes[ingredient].append(recipe_name)

        # convert the mapping of ingredient to recipes to produce the expected
```

(continues on next page)

(continued from previous page)

```python
        # output, shown below, using the ingredient name as a key to group
        #
        # name, subtype, docname, anchor, extra, qualifier, description
        for ingredient, recipe_names in ingredient_recipes.items():
            for recipe_name in recipe_names:
                dispname, typ, docname, anchor = recipes[recipe_name]
                content[ingredient].append(
                    (dispname, 0, docname, anchor, docname, '', typ))

        # convert the dict to the sorted list of tuples expected
        content = sorted(content.items())

        return content, True
```

```python
class RecipeIndex(Index):
    """A custom index that creates an recipe matrix."""

    name = 'recipe'
    localname = 'Recipe Index'
    shortname = 'Recipe'

    def generate(self, docnames=None):
        content = defaultdict(list)

        # sort the list of recipes in alphabetical order
        recipes = self.domain.get_objects()
        recipes = sorted(recipes, key=lambda recipe: recipe[0])

        # generate the expected output, shown below, from the above using the
        # first letter of the recipe as a key to group thing
        #
        # name, subtype, docname, anchor, extra, qualifier, description
        for name, dispname, typ, docname, anchor, _ in recipes:
            content[dispname[0].lower()].append(
                (dispname, 0, docname, anchor, docname, '', typ))

        # convert the dict to the sorted list of tuples expected
        content = sorted(content.items())

        return content, True
```

Both `IngredientIndex` and `RecipeIndex` are derived from *`Index`*. They implement custom logic to generate a tuple of values that define the index. Note that `RecipeIndex` is a simple index that has only one entry. Extending it to cover more object types is not yet part of the code.

Both indices use the method *`Index.generate()`* to do their work. This method combines the information from our domain, sorts it, and returns it in a list structure that will be accepted by Sphinx. This might look complicated but all it really is is a list of tuples like `('tomato', 'TomatoSoup', 'test', 'rec-TomatoSoup',...)`. Refer to the *domain API guide* for more information on this API.

These index pages can be referred by combination of domain name and its `name` using *`ref`* role. For example, `RecipeIndex` can be referred by ``:ref:`recipe-recipe` ``.

The domain

A Sphinx domain is a specialized container that ties together roles, directives, and indices, among other things. Let's look at the domain we're creating here.

```python
class RecipeDomain(Domain):

    name = 'recipe'
    label = 'Recipe Sample'
    roles = {
        'ref': XRefRole()
    }
    directives = {
        'recipe': RecipeDirective,
    }
    indices = {
        RecipeIndex,
        IngredientIndex
    }
    initial_data = {
        'recipes': [],  # object list
        'recipe_ingredients': {},  # name -> object
    }

    def get_full_qualified_name(self, node):
        return '{}.{}'.format('recipe', node.arguments[0])

    def get_objects(self):
        for obj in self.data['recipes']:
            yield(obj)

    def resolve_xref(self, env, fromdocname, builder, typ, target, node,
                     contnode):
        match = [(docname, anchor)
                 for name, sig, typ, docname, anchor, prio
                 in self.get_objects() if sig == target]

        if len(match) > 0:
            todocname = match[0][0]
            targ = match[0][1]

            return make_refnode(builder, fromdocname, todocname, targ,
                                contnode, targ)
        else:
            print('Awww, found nothing')
            return None

    def add_recipe(self, signature, ingredients):
        """Add a new recipe to the domain."""
        name = '{}.{}'.format('recipe', signature)
        anchor = 'recipe-{}'.format(signature)

        self.data['recipe_ingredients'][name] = ingredients
```

(continues on next page)

(continued from previous page)

```
        # name, dispname, type, docname, anchor, priority
        self.data['recipes'].append(
            (name, signature, 'Recipe', self.env.docname, anchor, 0))
```

There are some interesting things to note about this `recipe` domain and domains in general. Firstly, we actually register our directives, roles and indices here, via the `directives`, `roles` and `indices` attributes, rather than via calls later on in `setup`. We can also note that we aren't actually defining a custom role and are instead reusing the `sphinx.roles.XRefRole` role and defining the *`sphinx.domains.Domain.resolve_xref`* method. This method takes two arguments, `typ` and `target`, which refer to the cross-reference type and its target name. We'll use `target` to resolve our destination from our domain's `recipes` because we currently have only one type of node.

Moving on, we can see that we've defined `initial_data`. The values defined in `initial_data` will be copied to `env.domaindata[domain_name]` as the initial data of the domain, and domain instances can access it via `self.data`. We see that we have defined two items in `initial_data`: `recipes` and `recipe2ingredient`. These contain a list of all objects defined (i.e. all recipes) and a hash that maps a canonical ingredient name to the list of objects. The way we name objects is common across our extension and is defined in the `get_full_qualified_name` method. For each object created, the canonical name is `recipe.<recipename>`, where `<recipename>` is the name the documentation writer gives the object (a recipe). This enables the extension to use different object types that share the same name. Having a canonical name and central place for our objects is a huge advantage. Both our indices and our cross-referencing code use this feature.

The `setup` function

As always, the `setup` function is a requirement and is used to hook the various parts of our extension into Sphinx. Let's look at the `setup` function for this extension.

```
def setup(app):
    app.add_domain(RecipeDomain)

    return {
        'version': '0.1',
        'parallel_read_safe': True,
        'parallel_write_safe': True,
    }
```

This looks a little different to what we're used to seeing. There are no calls to *`add_directive()`* or even *`add_role()`*. Instead, we have a single call to *`add_domain()`* followed by some initialization of the *standard domain*. This is because we had already registered our directives, roles and indexes as part of the directive itself.

Using the extension

You can now use the extension throughout your project. For example:

Listing 4: index.rst

```
Joe's Recipes
=============

Below are a collection of my favourite recipes. I highly recommend the
:recipe:ref:`TomatoSoup` recipe in particular!

.. toctree::
```

(continues on next page)

(continued from previous page)

```
   tomato-soup
```

Listing 5: tomato-soup.rst

```
The recipe contains `tomato` and `cilantro`.

.. recipe:recipe:: TomatoSoup
   :contains: tomato, cilantro, salt, pepper

   This recipe is a tasty tomato soup, combine all ingredients
   and cook.
```

The important things to note are the use of the `:recipe:ref:` role to cross-reference the recipe actually defined elsewhere (using the `:recipe:recipe:` directive.

Further reading

For more information, refer to the docutils[274] documentation and *Developing extensions for Sphinx*.

2.3 Configuring builders

Discover builders by entry point

New in version 1.6.

builder extensions can be discovered by means of entry points[275] so that they do not have to be listed in the `extensions` configuration value.

Builder extensions should define an entry point in the `sphinx.builders` group. The name of the entry point needs to match your builder's `name` attribute, which is the name passed to the `sphinx-build -b` option. The entry point value should equal the dotted name of the extension module. Here is an example of how an entry point for 'mybuilder' can be defined in the extension's `setup.py`

```
setup(
    # ...
    entry_points={
        'sphinx.builders': [
            'mybuilder = my.extension.module',
        ],
    }
)
```

Note that it is still necessary to register the builder using `add_builder()` in the extension's `setup()` function.

[274] http://docutils.sourceforge.net/docs/
[275] https://setuptools.readthedocs.io/en/latest/setuptools.html#dynamic-discovery-of-services-and-plugins

2.4 HTML theme development

New in version 0.6.

Note: This document provides information about creating your own theme. If you simply wish to use a pre-existing HTML themes, refer to *HTML Theming*.

Sphinx supports changing the appearance of its HTML output via *themes*. A theme is a collection of HTML templates, stylesheet(s) and other static files. Additionally, it has a configuration file which specifies from which theme to inherit, which highlighting style to use, and what options exist for customizing the theme's look and feel.

Themes are meant to be project-unaware, so they can be used for different projects without change.

Note: See *Developing extensions for Sphinx* for more information that may be helpful in developing themes.

Creating themes

Themes take the form of either a directory or a zipfile (whose name is the theme name), containing the following:

- A `theme.conf` file.
- HTML templates, if needed.
- A `static/` directory containing any static files that will be copied to the output static directory on build. These can be images, styles, script files.

The `theme.conf` file is in INI format[1] (readable by the standard Python `ConfigParser` module) and has the following structure:

```
[theme]
inherit = base theme
stylesheet = main CSS name
pygments_style = stylename
sidebars = localtoc.html, relations.html, sourcelink.html, searchbox.html

[options]
variable = default value
```

- The **inherit** setting gives the name of a "base theme", or `none`. The base theme will be used to locate missing templates (most themes will not have to supply most templates if they use `basic` as the base theme), its options will be inherited, and all of its static files will be used as well. If you want to also inherit the stylesheet, include it via CSS' `@import` in your own.
- The **stylesheet** setting gives the name of a CSS file which will be referenced in the HTML header. If you need more than one CSS file, either include one from the other via CSS' `@import`, or use a custom HTML template that adds `<link rel="stylesheet">` tags as necessary. Setting the *`html_style`* config value will override this setting.
- The **pygments_style** setting gives the name of a Pygments style to use for highlighting. This can be overridden by the user in the *`pygments_style`* config value.
- The **pygments_dark_style** setting gives the name of a Pygments style to use for highlighting when the CSS media query `(prefers-color-scheme: dark)` evaluates to true. It is injected into the page using `add_css_file()`.

[1] It is not an executable Python file, as opposed to `conf.py`, because that would pose an unnecessary security risk if themes are shared.

- The **sidebars** setting gives the comma separated list of sidebar templates for constructing sidebars. This can be overridden by the user in the *html_sidebars* config value.
- The **options** section contains pairs of variable names and default values. These options can be overridden by the user in *html_theme_options* and are accessible from all templates as `theme_<name>`.

New in version 1.7: sidebar settings

Distribute your theme as a Python package

As a way to distribute your theme, you can use Python package. Python package brings to users easy setting up ways.

To distribute your theme as a Python package, please define an entry point called `sphinx.html_themes` in your `setup.py` file, and write a `setup()` function to register your themes using `add_html_theme()` API in it:

```
# 'setup.py'
setup(
    ...
    entry_points = {
        'sphinx.html_themes': [
            'name_of_theme = your_package',
        ]
    },
    ...
)

# 'your_package.py'
from os import path

def setup(app):
    app.add_html_theme('name_of_theme', path.abspath(path.dirname(__file__)))
```

If your theme package contains two or more themes, please call `add_html_theme()` twice or more.

New in version 1.2: 'sphinx_themes' entry_points feature.

Deprecated since version 1.6: `sphinx_themes` entry_points has been deprecated.

New in version 1.6: `sphinx.html_themes` entry_points feature.

Templating

The *guide to templating* is helpful if you want to write your own templates. What is important to keep in mind is the order in which Sphinx searches for templates:

- First, in the user's `templates_path` directories.
- Then, in the selected theme.
- Then, in its base theme, its base's base theme, etc.

When extending a template in the base theme with the same name, use the theme name as an explicit directory: `{% extends "basic/layout.html" %}`. From a user `templates_path` template, you can still use the "exclamation mark" syntax as described in the templating document.

Static templates

Since theme options are meant for the user to configure a theme more easily, without having to write a custom stylesheet, it is necessary to be able to template static files as well as HTML files. Therefore, Sphinx supports so-called "static templates", like this:

If the name of a file in the `static/` directory of a theme (or in the user's static path, for that matter) ends with `_t`, it will be processed by the template engine. The `_t` will be left from the final file name. For example, the *classic* theme has a file `static/classic.css_t` which uses templating to put the color options into the stylesheet. When a documentation is built with the classic theme, the output directory will contain a `_static/classic.css` file where all template tags have been processed.

Use custom page metadata in HTML templates

Any key / value pairs in *field lists* that are placed *before* the page's title will be available to the Jinja template when building the page within the `meta` attribute. For example, if a page had the following text before its first title:

```
:mykey: My value

My first title
--------------
```

Then it could be accessed within a Jinja template like so:

```
{%- if meta is mapping %}
    {{ meta.get("mykey") }}
{%- endif %}
```

Note the check that `meta` is a dictionary ("mapping" in Jinja terminology) to ensure that using it in this way is valid.

Defining custom template functions

Sometimes it is useful to define your own function in Python that you wish to then use in a template. For example, if you'd like to insert a template value with logic that depends on the user's configuration in the project, or if you'd like to include non-trivial checks and provide friendly error messages for incorrect configuration in the template.

To define your own template function, you'll need to define two functions inside your module:

- A **page context event handler** (or **registration**) function. This is connected to the `Sphinx` application via an event callback.
- A **template function** that you will use in your Jinja template.

First, define the registration function, which accepts the arguments for `html-page-context`.

Within the registration function, define the template function that you'd like to use within Jinja. The template function should return a string or Python objects (lists, dictionaries) with strings inside that Jinja uses in the templating process

Note: The template function will have access to all of the variables that are passed to the registration function.

At the end of the registration function, add the template function to the Sphinx application's context with `context['template_func'] = template_func`.

Finally, in your extension's `setup()` function, add your registration function as a callback for `html-page-context`.

```
# The registration function
def setup_my_func(app, pagename, templatename, context, doctree):
    # The template function
    def my_func(mystring):
        return "Your string is %s" % mystring
    # Add it to the page's context
    context['my_func'] = my_func

# Your extension's setup function
def setup(app):
    app.connect("html-page-context", setup_my_func)
```

Now, you will have access to this function in jinja like so:

```
<div>
{{ my_func("some string") }}
</div>
```

Add your own static files to the build assets

If you are packaging your own build assets with an extension (e.g., a CSS or JavaScript file), you need to ensure that they are placed in the `_static/` folder of HTML outputs. To do so, you may copy them directly into a build's `_static/` folder at build time, generally via an event hook. Here is some sample code to accomplish this:

```
def copy_custom_files(app, exc):
    if app.builder.format == 'html' and not exc:
        staticdir = path.join(app.builder.outdir, '_static')
        copy_asset_file('path/to/myextension/_static/myjsfile.js', staticdir)

def setup(app):
    app.connect('builder-inited', copy_custom_files)
```

Inject JavaScript based on user configuration

If your extension makes use of JavaScript, it can be useful to allow users to control its behavior using their Sphinx configuration. However, this can be difficult to do if your JavaScript comes in the form of a static library (which will not be built with Jinja).

There are two ways to inject variables into the JavaScript space based on user configuration.

First, you may append `_t` to the end of any static files included with your extension. This will cause Sphinx to process these files with the templating engine, allowing you to embed variables and control behavior.

For example, the following JavaScript structure:

```
mymodule/
├── _static
│   └── myjsfile.js_t
└── mymodule.py
```

Will result in the following static file placed in your HTML's build output:

```
_build/
└── html
    └── _static
        └── myjsfile.js
```

See *Static templates* for more information.

Second, you may use the `Sphinx.add_js_file()` method without pointing it to a file. Normally, this method is used to insert a new JavaScript file into your site. However, if you do *not* pass a file path, but instead pass a string to the "body" argument, then this text will be inserted as JavaScript into your site's head. This allows you to insert variables into your project's JavaScript from Python.

For example, the following code will read in a user-configured value and then insert this value as a JavaScript variable, which your extension's JavaScript code may use:

```python
# This function reads in a variable and inserts it into JavaScript
def add_js_variable(app):
    # This is a configuration that you've specified for users in `conf.py`
    js_variable = app.config['my_javascript_variable']
    js_text = "var my_variable = '%s';" % js_variable
    app.add_js_file(None, body=js_text)
# We connect this function to the step after the builder is initialized
def setup(app):
    # Tell Sphinx about this configuration variable
    app.add_config_value('my_javascript_variable')
    # Run the function after the builder is initialized
    app.connect('builder-inited', add_js_variable)
```

As a result, in your theme you can use code that depends on the presence of this variable. Users can control the variable's value by defining it in their `conf.py` file.

CHAPTER

THREE

MAN PAGES

These are the applications provided as part of Sphinx.

3.1 Core Applications

sphinx-quickstart

Synopsis

sphinx-quickstart

Description

`sphinx-quickstart` is an interactive tool that asks some questions about your project and then generates a complete documentation directory and sample Makefile to be used with *`sphinx-build(1)`*.

Options

`-q, --quiet`
Quiet mode that skips the interactive wizard for specifying options. This option requires *-p*, *-a* and *-v* options.

`-h, --help, --version`
Display usage summary or Sphinx version.

Structure Options

`--sep`
If specified, separate source and build directories.

`--no-sep`
If specified, create build directroy under source directroy.

`--dot``=DOT`
Inside the root directory, two more directories will be created; "_templates" for custom HTML templates and "_static" for custom stylesheets and other static files. You can enter another prefix (such as ".") to replace the underscore.

Project Basic Options

-p PROJECT, **--project**=PROJECT
Project name will be set. (see *project*).

-a AUTHOR, **--author**=AUTHOR
Author names. (see *copyright*).

-v VERSION
Version of project. (see *version*).

-r RELEASE, **--release**=RELEASE
Release of project. (see *release*).

-l LANGUAGE, **--language**=LANGUAGE
Document language. (see *language*).

--suffix=SUFFIX
Source file suffix. (see *source_suffix*).

--master=MASTER
Master document name. (see *master_doc*).

Extension Options

--ext-autodoc
Enable *sphinx.ext.autodoc* extension.

--ext-doctest
Enable *sphinx.ext.doctest* extension.

--ext-intersphinx
Enable *sphinx.ext.intersphinx* extension.

--ext-todo
Enable *sphinx.ext.todo* extension.

--ext-coverage
Enable *sphinx.ext.coverage* extension.

--ext-imgmath
Enable *sphinx.ext.imgmath* extension.

--ext-mathjax
Enable *sphinx.ext.mathjax* extension.

--ext-ifconfig
Enable *sphinx.ext.ifconfig* extension.

--ext-viewcode
Enable *sphinx.ext.viewcode* extension.

--ext-githubpages
Enable *sphinx.ext.githubpages* extension.

--extensions=EXTENSIONS
Enable arbitrary extensions.

Makefile and Batchfile Creation Options

`--use-make-mode` `(-m)`, **`--no-use-make-mode`** `(-M)`
: `Makefile/make.bat` uses (or doesn't use) *make-mode*. Default is `use`, which generates a more concise `Makefile/make.bat`.

Changed in version 1.5: make-mode is default.

`--makefile`, **`--no-makefile`**
: Create (or not create) makefile.

`--batchfile`, **`--no-batchfile`**
: Create (or not create) batchfile

Project templating

New in version 1.5: Project templating options for sphinx-quickstart

`-t`, **`--templatedir`**`=TEMPLATEDIR`
: Template directory for template files. You can modify the templates of sphinx project files generated by quickstart. Following Jinja2 template files are allowed:

- `master_doc.rst_t`
- `conf.py_t`
- `Makefile_t`
- `Makefile.new_t`
- `make.bat_t`
- `make.bat.new_t`

In detail, please refer the system template files Sphinx provides. (`sphinx/templates/quickstart`)

`-d` `NAME=VALUE`
: Define a template variable

See also

`sphinx-build(1)`

sphinx-build

Synopsis

sphinx-build [*options*] <*sourcedir*> <*outputdir*> [*filenames* …]

Description

`sphinx-build` generates documentation from the files in `<sourcedir>` and places it in the `<outputdir>`.

`sphinx-build` looks for `<sourcedir>/conf.py` for the configuration settings. *sphinx-quickstart(1)* may be used to generate template files, including `conf.py`.

`sphinx-build` can create documentation in different formats. A format is selected by specifying the builder name on the command line; it defaults to HTML. Builders can also perform other tasks related to documentation processing.

By default, everything that is outdated is built. Output only for selected files can be built by specifying individual filenames.

For a list of available options, refer to *sphinx-build -b*.

Options

-b buildername
The most important option: it selects a builder. The most common builders are:

html Build HTML pages. This is the default builder.

dirhtml Build HTML pages, but with a single directory per document. Makes for prettier URLs (no `.html`) if served from a webserver.

singlehtml Build a single HTML with the whole content.

htmlhelp, qthelp, devhelp, epub Build HTML files with additional information for building a documentation collection in one of these formats.

applehelp Build an Apple Help Book. Requires **hiutil** and **codesign**, which are not Open Source and presently only available on Mac OS X 10.6 and higher.

latex Build LaTeX sources that can be compiled to a PDF document using **pdflatex**.

man Build manual pages in groff format for UNIX systems.

texinfo Build Texinfo files that can be processed into Info files using **makeinfo**.

text Build plain text files.

gettext Build gettext-style message catalogs (`.pot` files).

doctest Run all doctests in the documentation, if the *doctest* extension is enabled.

linkcheck Check the integrity of all external links.

xml Build Docutils-native XML files.

pseudoxml Build compact pretty-printed "pseudo-XML" files displaying the internal structure of the intermediate document trees.

See *Builders* for a list of all builders shipped with Sphinx. Extensions can add their own builders.

-M buildername
Alternative to *-b*. Uses the Sphinx **make_mode** module, which provides the same build functionality as a default *Makefile or Make.bat*. In addition to all Sphinx *Builders*, the following build pipelines are available:

latexpdf Build LaTeX files and run them through **pdflatex**, or as per *latex_engine* setting. If *language* is set to `'ja'`, will use automatically the **platex/dvipdfmx** latex to PDF pipeline.

info Build Texinfo files and run them through **makeinfo**.

Important: Sphinx only recognizes the `-M` option if it is placed first.

New in version 1.2.1.

-a
If given, always write all output files. The default is to only write output files for new and changed source files. (This may not apply to all builders.)

-E
Don't use a saved *environment* (the structure caching all cross-references), but rebuild it completely. The default is to only read and parse source files that are new or have changed since the last run.

-t `tag`
Define the tag *tag*. This is relevant for `only` directives that only include their content if this tag is set.

New in version 0.6.

-d `path`
Since Sphinx has to read and parse all source files before it can write an output file, the parsed source files are cached as "doctree pickles". Normally, these files are put in a directory called `.doctrees` under the build directory; with this option you can select a different cache directory (the doctrees can be shared between all builders).

-j `N`
Distribute the build over *N* processes in parallel, to make building on multiprocessor machines more effective. Note that not all parts and not all builders of Sphinx can be parallelized. If `auto` argument is given, Sphinx uses the number of CPUs as *N*.

New in version 1.2: This option should be considered *experimental*.

Changed in version 1.7: Support `auto` argument.

-c `path`
Don't look for the `conf.py` in the source directory, but use the given configuration directory instead. Note that various other files and paths given by configuration values are expected to be relative to the configuration directory, so they will have to be present at this location too.

New in version 0.3.

-C
Don't look for a configuration file; only take options via the `-D` option.

New in version 0.5.

-D `setting=value`
Override a configuration value set in the `conf.py` file. The value must be a number, string, list or dictionary value.

For lists, you can separate elements with a comma like this: `-D html_theme_path=path1,path2`.

For dictionary values, supply the setting name and key like this: `-D latex_elements.docclass=scrartcl`.

For boolean values, use `0` or `1` as the value.

Changed in version 0.6: The value can now be a dictionary value.

Changed in version 1.3: The value can now also be a list value.

-A `name=value`
Make the *name* assigned to *value* in the HTML templates.

New in version 0.5.

-n
Run in nit-picky mode. Currently, this generates warnings for all missing references. See the config value `nitpick_ignore` for a way to exclude some references as "known missing".

-N
Do not emit colored output.

`-v`
Increase verbosity (loglevel). This option can be given up to three times to get more debug logging output. It implies *-T*.

New in version 1.2.

`-q`
Do not output anything on standard output, only write warnings and errors to standard error.

`-Q`
Do not output anything on standard output, also suppress warnings. Only errors are written to standard error.

`-w file`
Write warnings (and errors) to the given file, in addition to standard error.

`-W`
Turn warnings into errors. This means that the build stops at the first warning and `sphinx-build` exits with exit status 1.

`--keep-going`
With -W option, keep going processing when getting warnings to the end of build, and `sphinx-build` exits with exit status 1.

New in version 1.8.

`-T`
Display the full traceback when an unhandled exception occurs. Otherwise, only a summary is displayed and the traceback information is saved to a file for further analysis.

New in version 1.2.

`-P`
(Useful for debugging only.) Run the Python debugger, pdb[276], if an unhandled exception occurs while building.

`-h, --help, --version`
Display usage summary or Sphinx version.

New in version 1.2.

You can also give one or more filenames on the command line after the source and build directories. Sphinx will then try to build only these output files (and their dependencies).

Environment Variables

The **`sphinx-build`** refers following environment variables:

`MAKE`
A path to make command. A command name is also allowed. **`sphinx-build`** uses it to invoke sub-build process on make-mode.

[276] https://docs.python.org/3/library/pdb.html#module-pdb

Makefile Options

The `Makefile` and `make.bat` files created by **`sphinx-quickstart`** usually run **`sphinx-build`** only with the *`-b`* and *`-d`* options. However, they support the following variables to customize behavior:

`PAPER`
This sets the `'papersize'` key of *`latex_elements`*: i.e. `PAPER=a4` sets it to `'a4paper'` and `PAPER=letter` to `'letterpaper'`.

Note: Usage of this environment variable got broken at Sphinx 1.5 as `a4` or `letter` ended up as option to LaTeX document in place of the needed `a4paper`, resp. `letterpaper`. Fixed at 1.7.7.

`SPHINXBUILD`
The command to use instead of `sphinx-build`.

`BUILDDIR`
The build directory to use instead of the one chosen in **`sphinx-quickstart`**.

`SPHINXOPTS`
Additional options for **`sphinx-build`**. These options can also be set via the shortcut variable **O** (capital 'o').

Deprecation Warnings

If any deprecation warning like `RemovedInSphinxXXXWarning` are displayed when building a user's document, some Sphinx extension is using deprecated features. In that case, please report it to author of the extension.

To disable the deprecation warnings, please set `PYTHONWARNINGS=` environment variable to your environment. For example:

- `PYTHONWARNINGS= make html` (Linux/Mac)
- `export PYTHONWARNINGS=` and do `make html` (Linux/Mac)
- `set PYTHONWARNINGS=` and do `make html` (Windows)
- modify your Makefile/make.bat and set the environment variable

See also

`sphinx-quickstart(1)`

3.2 Additional Applications

sphinx-apidoc

Synopsis

sphinx-apidoc [*OPTIONS*] -o <*OUTPUT_PATH*> <*MODULE_PATH*> [*EXCLUDE_PATTERN* …]

Description

sphinx-apidoc is a tool for automatic generation of Sphinx sources that, using the `autodoc` extension, document a whole package in the style of other automatic API documentation tools.

MODULE_PATH is the path to a Python package to document, and *OUTPUT_PATH* is the directory where the generated sources are placed. Any *EXCLUDE_PATTERN*s given are fnmatch-style[277] file and/or directory patterns that will be excluded from generation.

> **Warning:** `sphinx-apidoc` generates source files that use *`sphinx.ext.autodoc`* to document all found modules. If any modules have side effects on import, these will be executed by `autodoc` when `sphinx-build` is run.
>
> If you document scripts (as opposed to library modules), make sure their main routine is protected by a `if __name__ == '__main__'` condition.

Options

`-o` `<OUTPUT_PATH>`
: Directory to place the output files. If it does not exist, it is created.

`-q`
: Do not output anything on standard output, only write warnings and errors to standard error.

`-f, --force`
: Force overwriting of any existing generated files.

`-l, --follow-links`
: Follow symbolic links.

`-n, --dry-run`
: Do not create any files.

`-s` `<suffix>`
: Suffix for the source files generated. Defaults to `rst`.

`-d` `<MAXDEPTH>`
: Maximum depth for the generated table of contents file.

`--tocfile`
: Filename for a table of contents file. Defaults to `modules`.

`-T, --no-toc`
: Do not create a table of contents file. Ignored when *`--full`* is provided.

`-F, --full`
: Generate a full Sphinx project (`conf.py`, `Makefile` etc.) using the same mechanism as **`sphinx-quickstart`**.

`-e, --separate`
: Put documentation for each module on its own page.

  New in version 1.2.

`-E, --no-headings`
: Do not create headings for the modules/packages. This is useful, for example, when docstrings already contain headings.

`-P, --private`
: Include "_private" modules.

  New in version 1.2.

[277] https://docs.python.org/3/library/fnmatch.html

--implicit-namespaces
: By default sphinx-apidoc processes sys.path searching for modules only. Python 3.3 introduced **PEP 420**[278] implicit namespaces that allow module path structures such as `foo/bar/module.py` or `foo/bar/baz/__init__.py` (notice that `bar` and `foo` are namespaces, not modules).

  Interpret paths recursively according to PEP-0420.

-M, --module-first
: Put module documentation before submodule documentation.

These options are used when *--full* is specified:

-a
: Append module_path to sys.path.

-H <project>
: Sets the project name to put in generated files (see *project*).

-A <author>
: Sets the author name(s) to put in generated files (see *copyright*).

-V <version>
: Sets the project version to put in generated files (see *version*).

-R <release>
: Sets the project release to put in generated files (see *release*).

Project templating

New in version 2.2: Project templating options for sphinx-apidoc

-t, --templatedir=TEMPLATEDIR
: Template directory for template files. You can modify the templates of sphinx project files generated by apidoc. Following Jinja2 template files are allowed:

  - `module.rst_t`
  - `package.rst_t`
  - `toc.rst_t`
  - `master_doc.rst_t`
  - `conf.py_t`
  - `Makefile_t`
  - `Makefile.new_t`
  - `make.bat_t`
  - `make.bat.new_t`

  In detail, please refer the system template files Sphinx provides. (`sphinx/templates/apidoc` and `sphinx/templates/quickstart`)

[278] https://www.python.org/dev/peps/pep-0420

Environment

SPHINX_APIDOC_OPTIONS
: A comma-separated list of option to append to generated `automodule` directives. Defaults to `members,undoc-members,show-inheritance`.

See also

sphinx-build(1), *sphinx-autogen(1)*

sphinx-autogen

Synopsis

sphinx-autogen [*options*] <sourcefile> …

Description

sphinx-autogen is a tool for automatic generation of Sphinx sources that, using the `autodoc` extension, document items included in *autosummary* listing(s).

sourcefile is the path to one or more reStructuredText documents containing *autosummary* entries with the `:toctree::` option set. *sourcefile* can be an `fnmatch`[279]-style pattern.

Options

-o <outputdir>
: Directory to place the output file. If it does not exist, it is created. Defaults to the value passed to the `:toctree:` option.

-s <suffix>, **--suffix** <suffix>
: Default suffix to use for generated files. Defaults to `rst`.

-t <templates>, **--templates** <templates>
: Custom template directory. Defaults to `None`.

-i, --imported-members
: Document imported members.

Example

Given the following directory structure:

```
docs
├── index.rst
└── ...
foobar
├── foo
│   └── __init__.py
└── bar
    ├── __init__.py
```

(continues on next page)

[279] https://docs.python.org/3/library/fnmatch.html#module-fnmatch

(continued from previous page)

```
    └── baz
        └── __init__.py
```

and assuming docs/index.rst contained the following:

```
Modules
=======

.. autosummary::
   :toctree: modules

   foobar.foo
   foobar.bar
   foobar.bar.baz
```

If you run the following:

```
$ PYTHONPATH=. sphinx-autogen docs/index.rst
```

then the following stub files will be created in docs:

```
docs
├── index.rst
└── modules
    ├── foobar.bar.rst
    ├── foobar.bar.baz.rst
    └── foobar.foo.rst
```

and each of those files will contain a autodoc directive and some other information.

See also

sphinx-build(1), *sphinx-apidoc(1)*

CHAPTER

FOUR

TEMPLATING

Sphinx uses the Jinja[280] templating engine for its HTML templates. Jinja is a text-based engine, inspired by Django templates, so anyone having used Django will already be familiar with it. It also has excellent documentation for those who need to make themselves familiar with it.

4.1 Do I need to use Sphinx's templates to produce HTML?

No. You have several other options:

- You can write a `TemplateBridge` subclass that calls your template engine of choice, and set the `template_bridge` configuration value accordingly.
- You can *write a custom builder* that derives from `StandaloneHTMLBuilder` and calls your template engine of choice.
- You can use the `PickleHTMLBuilder` that produces pickle files with the page contents, and postprocess them using a custom tool, or use them in your Web application.

4.2 Jinja/Sphinx Templating Primer

The default templating language in Sphinx is Jinja. It's Django/Smarty inspired and easy to understand. The most important concept in Jinja is *template inheritance*, which means that you can overwrite only specific blocks within a template, customizing it while also keeping the changes at a minimum.

To customize the output of your documentation you can override all the templates (both the layout templates and the child templates) by adding files with the same name as the original filename into the template directory of the structure the Sphinx quickstart generated for you.

Sphinx will look for templates in the folders of `templates_path` first, and if it can't find the template it's looking for there, it falls back to the selected theme's templates.

A template contains **variables**, which are replaced with values when the template is evaluated, **tags**, which control the logic of the template and **blocks** which are used for template inheritance.

Sphinx's *basic* theme provides base templates with a couple of blocks it will fill with data. These are located in the `themes/basic` subdirectory of the Sphinx installation directory, and used by all builtin Sphinx themes. Templates with the same name in the `templates_path` override templates supplied by the selected theme.

For example, to add a new link to the template area containing related links all you have to do is to add a new template called `layout.html` with the following contents:

[280] http://jinja.pocoo.org

```
{% extends "!layout.html" %}
{% block rootrellink %}
    <li><a href="https://project.invalid/">Project Homepage</a> &raquo;</li>
    {{ super() }}
{% endblock %}
```

By prefixing the name of the overridden template with an exclamation mark, Sphinx will load the layout template from the underlying HTML theme.

Important: If you override a block, call `{{ super() }}` somewhere to render the block's original content in the extended template – unless you don't want that content to show up.

4.3 Working with the builtin templates

The builtin **basic** theme supplies the templates that all builtin Sphinx themes are based on. It has the following elements you can override or use:

Blocks

The following blocks exist in the `layout.html` template:

`doctype` The doctype of the output format. By default this is XHTML 1.0 Transitional as this is the closest to what Sphinx and Docutils generate and it's a good idea not to change it unless you want to switch to HTML 5 or a different but compatible XHTML doctype.

`linktags` This block adds a couple of `<link>` tags to the head section of the template.

`extrahead` This block is empty by default and can be used to add extra contents into the `<head>` tag of the generated HTML file. This is the right place to add references to JavaScript or extra CSS files.

`relbar1`, `relbar2` This block contains the *relation bar*, the list of related links (the parent documents on the left, and the links to index, modules etc. on the right). `relbar1` appears before the document, `relbar2` after the document. By default, both blocks are filled; to show the relbar only before the document, you would override *relbar2* like this:

```
{% block relbar2 %}{% endblock %}
```

`rootrellink`, `relbaritems` Inside the relbar there are three sections: The `rootrellink`, the links from the documentation and the custom `relbaritems`. The `rootrellink` is a block that by default contains a list item pointing to the master document by default, the `relbaritems` is an empty block. If you override them to add extra links into the bar make sure that they are list items and end with the *`reldelim1`*.

`document` The contents of the document itself. It contains the block "body" where the individual content is put by subtemplates like `page.html`.

Note: In order for the built-in JavaScript search to show a page preview on the results page, the document or body content should be wrapped in an HTML element containing the `role="main"` attribute. For example:

```
<div role="main">
  {% block document %}{% endblock %}
</div>
```

sidebar1, sidebar2 A possible location for a sidebar. `sidebar1` appears before the document and is empty by default, `sidebar2` after the document and contains the default sidebar. If you want to swap the sidebar location override this and call the `sidebar` helper:

```
{% block sidebar1 %}{{ sidebar() }}{% endblock %}
{% block sidebar2 %}{% endblock %}
```

(The `sidebar2` location for the sidebar is needed by the `sphinxdoc.css` stylesheet, for example.)

sidebarlogo The logo location within the sidebar. Override this if you want to place some content at the top of the sidebar.

footer The block for the footer div. If you want a custom footer or markup before or after it, override this one.

The following four blocks are *only* used for pages that do not have assigned a list of custom sidebars in the *html_sidebars* config value. Their use is deprecated in favor of separate sidebar templates, which can be included via *html_sidebars*.

sidebartoc The table of contents within the sidebar.

Deprecated since version 1.0.

sidebarrel The relation links (previous, next document) within the sidebar.

Deprecated since version 1.0.

sidebarsourcelink The "Show source" link within the sidebar (normally only shown if this is enabled by *html_show_sourcelink*).

Deprecated since version 1.0.

sidebarsearch The search box within the sidebar. Override this if you want to place some content at the bottom of the sidebar.

Deprecated since version 1.0.

Configuration Variables

Inside templates you can set a couple of variables used by the layout template using the `{% set %}` tag:

reldelim1
The delimiter for the items on the left side of the related bar. This defaults to `' &raquo;'` Each item in the related bar ends with the value of this variable.

reldelim2
The delimiter for the items on the right side of the related bar. This defaults to `' |'`. Each item except of the last one in the related bar ends with the value of this variable.

Overriding works like this:

```
{% extends "!layout.html" %}
{% set reldelim1 = ' &gt;' %}
```

script_files
Add additional script files here, like this:

```
{% set script_files = script_files + ["_static/myscript.js"] %}
```

Deprecated since version 1.8.0: Please use `.Sphinx.add_js_file()` instead.

Helper Functions

Sphinx provides various Jinja functions as helpers in the template. You can use them to generate links or output multiply used elements.

`pathto`(*document*)
: Return the path to a Sphinx document as a URL. Use this to refer to built documents.

`pathto`(*file*, *1*)
: Return the path to a *file* which is a filename relative to the root of the generated output. Use this to refer to static files.

`hasdoc`(*document*)
: Check if a document with the name *document* exists.

`sidebar`()
: Return the rendered sidebar.

`relbar`()
: Return the rendered relation bar.

`warning`(*message*)
: Emit a warning message.

Global Variables

These global variables are available in every template and are safe to use. There are more, but most of them are an implementation detail and might change in the future.

`builder`
: The name of the builder (e.g. `html` or `htmlhelp`).

`copyright`
: The value of *`copyright`*.

`docstitle`
: The title of the documentation (the value of *`html_title`*), except when the “single-file” builder is used, when it is set to `None`.

`embedded`
: True if the built HTML is meant to be embedded in some viewing application that handles navigation, not the web browser, such as for HTML help or Qt help formats. In this case, the sidebar is not included.

`favicon`
: The path to the HTML favicon in the static path, or URL to the favicon, or `''`.

  Deprecated since version 4.0: Recommend to use `favicon_url` instead.

`favicon_url`
: The relative path to the HTML favicon image from the current document, or URL to the favicon, or `''`.

  New in version 4.0.

`file_suffix`
: The value of the builder’s *`out_suffix`* attribute, i.e. the file name extension that the output files will get. For a standard HTML builder, this is usually `.html`.

`has_source`
: True if the reST document sources are copied (if *`html_copy_source`* is `True`).

`language`
: The value of *`language`*.

`last_updated`
: The build date.

`logo`
: The path to the HTML logo image in the static path, or URL to the logo, or `''`.

  Deprecated since version 4.0: Recommend to use `logo_url` instead.

`logo_url`
: The relative path to the HTML logo image from the current document, or URL to the logo, or `''`.

  New in version 4.0.

`master_doc`
: The value of *`master_doc`*, for usage with *`pathto()`*.

`pagename`
: The "page name" of the current file, i.e. either the document name if the file is generated from a reST source, or the equivalent hierarchical name relative to the output directory (`[directory/]filename_without_extension`).

`project`
: The value of *`project`*.

`release`
: The value of *`release`*.

`rellinks`
: A list of links to put at the left side of the relbar, next to "next" and "prev". This usually contains links to the general index and other indices, such as the Python module index. If you add something yourself, it must be a tuple `(pagename, link title, accesskey, link text)`.

`shorttitle`
: The value of *`html_short_title`*.

`show_source`
: True if *`html_show_sourcelink`* is `True`.

`sphinx_version`
: The version of Sphinx used to build.

`style`
: The name of the main stylesheet, as given by the theme or *`html_style`*.

`title`
: The title of the current document, as used in the `<title>` tag.

`use_opensearch`
: The value of *`html_use_opensearch`*.

`version`
: The value of *`version`*.

In addition to these values, there are also all **theme options** available (prefixed by `theme_`), as well as the values given by the user in *`html_context`*.

In documents that are created from source files (as opposed to automatically-generated files like the module index, or documents that already are in HTML form), these variables are also available:

`body`
: A string containing the content of the page in HTML form as produced by the HTML builder, before the theme is applied.

`display_toc`
: A boolean that is True if the toc contains more than one entry.

`meta`
: Document metadata (a dictionary), see *File-wide metadata*.

`metatags`
: A string containing the page's HTML meta[281] tags.

`next`
: The next document for the navigation. This variable is either false or has two attributes *link* and *title*. The title contains HTML markup. For example, to generate a link to the next page, you can use this snippet:

```
{% if next %}
<a href="{{ next.link|e }}">{{ next.title }}</a>
{% endif %}
```

`page_source_suffix`
: The suffix of the file that was rendered. Since we support a list of *`source_suffix`*, this will allow you to properly link to the original source file.

`parents`
: A list of parent documents for navigation, structured like the *`next`* item.

`prev`
: Like *`next`*, but for the previous page.

`sourcename`
: The name of the copied source file for the current document. This is only nonempty if the *`html_copy_source`* value is `True`. This has empty value on creating automatically-generated files.

`toc`
: The local table of contents for the current page, rendered as HTML bullet lists.

`toctree`
: A callable yielding the global TOC tree containing the current page, rendered as HTML bullet lists. Optional keyword arguments:

`collapse` If true, all TOC entries that are not ancestors of the current page are collapsed. `True` by default.

`maxdepth` The maximum depth of the tree. Set it to `-1` to allow unlimited depth. Defaults to the max depth selected in the toctree directive.

`titles_only` If true, put only top-level document titles in the tree. `False` by default.

`includehidden` If true, the ToC tree will also contain hidden entries. `False` by default.

[281] http://docutils.sourceforge.net/docs/ref/rst/directives.html#meta

CHAPTER

FIVE

LATEX CUSTOMIZATION

Unlike *the HTML builders*, the `latex` builder does not benefit from prepared themes. The *Options for LaTeX output*, and particularly the *latex_elements* variable, provides much of the interface for customization. For example:

```
# inside conf.py
latex_engine = 'xelatex'
latex_elements = {
    'fontpkg': r'''
\setmainfont{DejaVu Serif}
\setsansfont{DejaVu Sans}
\setmonofont{DejaVu Sans Mono}
''',
    'preamble': r'''
\usepackage[titles]{tocloft}
\cftsetpnumwidth {1.25cm}\cftsetrmarg{1.5cm}
\setlength{\cftchapnumwidth}{0.75cm}
\setlength{\cftsecindent}{\cftchapnumwidth}
\setlength{\cftsecnumwidth}{1.25cm}
''',
    'fncychap': r'\usepackage[Bjornstrup]{fncychap}',
    'printindex': r'\footnotesize\raggedright\printindex',
}
latex_show_urls = 'footnote'
```

Note: Keep in mind that backslashes must be doubled in Python string literals to avoid interpretation as escape sequences. Alternatively, you may use raw strings as is done above.

5.1 The `latex_elements` configuration setting

A dictionary that contains LaTeX snippets overriding those Sphinx usually puts into the generated `.tex` files. Its `'sphinxsetup'` key is described *separately*.

Keys that you may want to override include:

`'papersize'` Paper size option of the document class (`'a4paper'` or `'letterpaper'`)

Default: `'letterpaper'`

`'pointsize'` Point size option of the document class (`'10pt'`, `'11pt'` or `'12pt'`)

Default: `'10pt'`

'pxunit' The value of the `px` when used in image attributes `width` and `height`. The default value is `'0.75bp'` which achieves `96px=1in` (in TeX `1in = 72bp = 72.27pt`.) To obtain for example `100px=1in` use `'0.01in'` or `'0.7227pt'` (the latter leads to TeX computing a more precise value, due to the smaller unit used in the specification); for `72px=1in`, simply use `'1bp'`; for `90px=1in`, use `'0.8bp'` or `'0.803pt'`.

Default: `'0.75bp'`

New in version 1.5.

'passoptionstopackages' A string which will be positioned early in the preamble, designed to contain `\\PassOptionsToPackage{options}{foo}` commands.

Hint: It may be also used for loading LaTeX packages very early in the preamble. For example package `fancybox` is incompatible with being loaded via the `'preamble'` key, it must be loaded earlier.

Default: `''`

New in version 1.4.

'babel' "babel" package inclusion, default `'\\usepackage{babel}'` (the suitable document language string is passed as class option, and `english` is used if no language.) For Japanese documents, the default is the empty string.

With XeLaTeX and LuaLaTeX, Sphinx configures the LaTeX document to use polyglossia[282], but one should be aware that current babel[283] has improved its support for Unicode engines in recent years and for some languages it may make sense to prefer `babel` over `polyglossia`.

Hint: After modifiying a core LaTeX key like this one, clean up the LaTeX build repertory before next PDF build, else left-over auxiliary files are likely to break the build.

Default: `'\\usepackage{babel}'` (`''` for Japanese documents)

Changed in version 1.5: For *`latex_engine`* set to `'xelatex'`, the default is `'\\usepackage{polyglossia}\n\\setmainlanguage{<language>}'`.

Changed in version 1.6: `'lualatex'` uses same default setting as `'xelatex'`

Changed in version 1.7.6: For French, `xelatex` and `lualatex` default to using `babel`, not `polyglossia`.

'fontpkg' Font package inclusion. The default is:

```
r"""\usepackage{tgtermes}
\usepackage{tgheros}
\renewcommand\ttdefault{txtt}
"""
```

For `'xelatex'` and `'lualatex'` however the default is to use the GNU FreeFont.

Changed in version 1.2: Defaults to `''` when the *`language`* uses the Cyrillic script.

Changed in version 2.0: Incorporates some font substitution commands to help support occasional Greek or Cyrillic in a document using `'pdflatex'` engine.

Changed in version 4.0.0: - The font substitution commands added at `2.0` have been moved to the `'fontsubstitution'` key, as their presence here made it complicated for user to customize the value of `'fontpkg'`. - The default font setting has changed: it still uses Times and Helvetica clones for serif and sans serif, but via better, more complete TeX fonts and associated LaTeX packages. The monospace font has been changed to better match the Times clone.

[282] https://ctan.org/pkg/polyglossia
[283] https://ctan.org/pkg/babel

'fncychap' Inclusion of the "fncychap" package (which makes fancy chapter titles), default '\\usepackage[Bjarne]{fncychap}' for English documentation (this option is slightly customized by Sphinx), '\\usepackage[Sonny]{fncychap}' for internationalized docs (because the "Bjarne" style uses numbers spelled out in English). Other "fncychap" styles you can try are "Lenny", "Glenn", "Conny", "Rejne" and "Bjornstrup". You can also set this to '' to disable fncychap.

Default: '\\usepackage[Bjarne]{fncychap}' for English documents, '\\usepackage[Sonny]{fncychap}' for internationalized documents, and '' for Japanese documents.

'preamble' Additional preamble content. One may move all needed macros into some file mystyle.tex.txt of the project source repertory, and get LaTeX to import it at run time:

```
'preamble': r'\input{mystyle.tex.txt}',
# or, if the \ProvidesPackage LaTeX macro is used in a file mystyle.sty
'preamble': r'\usepackage{mystyle}',
```

It is then needed to set appropriately *latex_additional_files*, for example:

```
latex_additional_files = ["mystyle.sty"]
```

Default: ''

'figure_align' Latex figure float alignment. Whenever an image doesn't fit into the current page, it will be 'floated' into the next page but may be preceded by any other text. If you don't like this behavior, use 'H' which will disable floating and position figures strictly in the order they appear in the source.

Default: 'htbp' (here, top, bottom, page)

New in version 1.3.

'atendofbody' Additional document content (right before the indices).

Default: ''

New in version 1.5.

'extrapackages' Additional LaTeX packages. For example:

```
latex_elements = {
    'packages': r'\usepackage{isodate}'
}
```

The specified LaTeX packages will be loaded before hyperref package and packages loaded from Sphinx extensions.

Hint: If you'd like to load additional LaTeX packages after hyperref, use 'preamble' key instead.

Default: ''

New in version 2.3.

'footer' Additional footer content (before the indices).

Default: ''

Deprecated since version 1.5: Use 'atendofbody' key instead.

Keys that don't need to be overridden unless in special cases are:

'extraclassoptions' The default is the empty string. Example: 'extraclassoptions': 'openany' will allow chapters (for documents of the 'manual' type) to start on any page.

Default: ''

New in version 1.2.

Changed in version 1.6: Added this documentation.

`'maxlistdepth'` LaTeX allows by default at most 6 levels for nesting list and quote-like environments, with at most 4 enumerated lists, and 4 bullet lists. Setting this key for example to `'10'` (as a string) will allow up to 10 nested levels (of all sorts). Leaving it to the empty string means to obey the LaTeX default.

Warning:

- Using this key may prove incompatible with some LaTeX packages or special document classes which do their own list customization.
- The key setting is silently *ignored* if `\usepackage{enumitem}` is executed inside the document preamble. Use then rather the dedicated commands of this LaTeX package.

Default: `6`

New in version 1.5.

`'inputenc'` "inputenc" package inclusion.

Default: `'\\usepackage[utf8]{inputenc}'` when using pdflatex, else `''`

Changed in version 1.4.3: Previously `'\\usepackage[utf8]{inputenc}'` was used for all compilers.

`'cmappkg'` "cmap" package inclusion.

Default: `'\\usepackage{cmap}'`

New in version 1.2.

`'fontenc'` Customize this from its default `'\\usepackage[T1]{fontenc}'` to:

- `'\\usepackage[X2,T1]{fontenc}'` if you need occasional Cyrillic letters (физика частиц),
- `'\\usepackage[LGR,T1]{fontenc}'` if you need occasional Greek letters (Σωματιδιακή φυσική).

Use `[LGR,X2,T1]` rather if both are needed.

Attention:

- Do not use this key for a *`latex_engine`* other than `'pdflatex'`.
- If Greek is main language, do not use this key. Since Sphinx 2.2.1, `xelatex` will be used automatically as *`latex_engine`*.
- The TeX installation may need some extra packages. For example, on Ubuntu xenial, packages `texlive-lang-greek` and `cm-super` are needed for LGR to work. And `texlive-lang-cyrillic` and `cm-super` are needed for support of Cyrillic.

Changed in version 1.5: Defaults to `'\\usepackage{fontspec}'` when *`latex_engine`* is `'xelatex'`.

Changed in version 1.6: `'lualatex'` uses `fontspec` per default like `'xelatex'`.

Changed in version 2.0: `'lualatex'` executes `\defaultfontfeatures[\rmfamily,\sffamily]{}` to disable TeX ligatures transforming << and >> as escaping working with `pdflatex/xelatex` failed with `lualatex`.

Changed in version 2.0: Detection of `LGR`, `T2A`, `X2` to trigger support of occasional Greek or Cyrillic letters (`'pdflatex'`).

Changed in version 2.3.0: `'xelatex'` executes `\defaultfontfeatures[\rmfamily,\sffamily]{}` in order to avoid contractions of `--` into en-dash or transforms of straight quotes into curly ones in PDF (in non-literal text paragraphs) despite *`smartquotes`* being set to `False`.

`'fontsubstitution'` Ignored if `'fontenc'` was not configured to use `LGR` or `X2` (or `T2A`). In case `'fontpkg'` key is configured for usage with some TeX fonts known to be available in the `LGR` or `X2` encodings, set this one to be the empty string. Else leave to its default.

Ignored with *`latex_engine`* other than `'pdflatex'`.

New in version 4.0.0.

`'textgreek'` For the support of occasional Greek letters.

It is ignored with `'platex'`, `'xelatex'` or `'lualatex'` as *`latex_engine`* and defaults to either the empty string or to `'\\usepackage{textalpha}'` for `'pdflatex'` depending on whether the `'fontenc'` key was used with `LGR` or not. Only expert LaTeX users may want to customize this key.

It can also be used as `r'\usepackage{textalpha,alphabeta}'` to let `'pdflatex'` support Greek Unicode input in *`math`* context. For example ``:math:`α` `` (U+03B1) will render as α.

Default: `'\\usepackage{textalpha}'` or `''` if `fontenc` does not include the `LGR` option.

New in version 2.0.

`'geometry'` "geometry" package inclusion, the default definition is:

```
'\\usepackage{geometry}'
```

with an additional `[dvipdfm]` for Japanese documents. The Sphinx LaTeX style file executes:

```
\PassOptionsToPackage{hmargin=1in,vmargin=1in,marginpar=0.5in}{geometry}
```

which can be customized via corresponding *'sphinxsetup' options*.

Default: `'\\usepackage{geometry}'` (or `'\\usepackage[dvipdfm]{geometry}'` for Japanese documents)

New in version 1.5.

Changed in version 1.5.2: `dvipdfm` option if *`latex_engine`* is `'platex'`.

New in version 1.5.3: The *'sphinxsetup' keys for the margins*.

Changed in version 1.5.3: The location in the LaTeX file has been moved to after `\usepackage{sphinx}` and `\sphinxsetup{..}`, hence also after insertion of `'fontpkg'` key. This is in order to handle the paper layout options in a special way for Japanese documents: the text width will be set to an integer multiple of the *zenkaku* width, and the text height to an integer multiple of the baseline. See the *hmargin* documentation for more.

`'hyperref'` "hyperref" package inclusion; also loads package "hypcap" and issues `\urlstyle{same}`. This is done after `sphinx.sty` file is loaded and before executing the contents of `'preamble'` key.

Attention: Loading of packages "hyperref" and "hypcap" is mandatory.

New in version 1.5: Previously this was done from inside `sphinx.sty`.

`'maketitle'` "maketitle" call. Override if you want to generate a differently styled title page.

Hint: If the key value is set to `r'\newcommand\sphinxbackoftitlepage{<Extra material>}\sphinxmaketitle'`, then `<Extra material>` will be typeset on back of title page (`'manual'` docclass only).

Default: `'\\sphinxmaketitle'`

Changed in version 1.8.3: Original `\maketitle` from document class is not overwritten, hence is re-usable as part of some custom setting for this key.

New in version 1.8.3: `\sphinxbackoftitlepage` optional macro. It can also be defined inside `'preamble'` key rather than this one.

`'releasename'` Value that prefixes `'release'` element on title page. As for *title* and *author* used in the tuples of `latex_documents`, it is inserted as LaTeX markup.

Default: `'Release'`

`'tableofcontents'` "tableofcontents" call. The default of `'\\sphinxtableofcontents'` is a wrapper of unmodified `\tableofcontents`, which may itself be customized by user loaded packages. Override if you want to generate a different table of contents or put content between the title page and the TOC.

Default: `'\\sphinxtableofcontents'`

Changed in version 1.5: Previously the meaning of `\tableofcontents` itself was modified by Sphinx. This created an incompatibility with dedicated packages modifying it also such as "tocloft" or "etoc".

`'transition'` Commands used to display transitions. Override if you want to display transitions differently.

Default: `'\n\n\\bigskip\\hrule\\bigskip\n\n'`

New in version 1.2.

Changed in version 1.6: Remove unneeded `{}` after `\\hrule`.

`'makeindex'` "makeindex" call, the last thing before `\begin{document}`. With `'\\usepackage[columns=1]{idxlayout}\\makeindex'` the index will use only one column. You may have to install `idxlayout` LaTeX package.

Default: `'\\makeindex'`

`'printindex'` "printindex" call, the last thing in the file. Override if you want to generate the index differently, append some content after the index, or change the font. As LaTeX uses two-column mode for the index it is often advisable to set this key to `'\\footnotesize\\raggedright\\printindex'`. Or, to obtain a one-column index, use `'\\def\\twocolumn[#1]{#1}\\printindex'` (this trick may fail if using a custom document class; then try the `idxlayout` approach described in the documentation of the `'makeindex'` key).

Default: `'\\printindex'`

`'fvset'` Customization of `fancyvrb` LaTeX package.

The default value is `'\\fvset{fontsize=auto}'` which means that the font size will adjust correctly if a code-block ends up in a footnote. You may need to modify this if you use custom fonts: `'\\fvset{fontsize=\\small}'` if the monospace font is Courier-like.

Default: `'\\fvset{fontsize=auto}'`

New in version 1.8.

Changed in version 2.0: For `'xelatex'` and `'lualatex'` defaults to `'\\fvset{fontsize=\\small}'` as this is adapted to the relative widths of the FreeFont family.

Changed in version 4.0.0: Changed default for `'pdflatex'`. Previously it was using `'\\fvset{fontsize=\\small}'`.

Keys that are set by other options and therefore should not be overridden are:

`'docclass'` `'classoptions'` `'title'` `'release'` `'author'`

5.2 The `sphinxsetup` configuration setting

New in version 1.5.

The `'sphinxsetup'` key of *latex_elements* provides a LaTeX-type customization interface:

```
latex_elements = {
    'sphinxsetup': 'key1=value1, key2=value2, ...',
}
```

It defaults to empty. If non-empty, it will be passed as argument to the `\sphinxsetup` macro inside the document preamble, like this:

```
\usepackage{sphinx}
\sphinxsetup{key1=value1, key2=value2,...}
```

The colors used in the above are provided by the `svgnames` option of the "xcolor" package:

```
latex_elements = {
    'passoptionstopackages': r'\PassOptionsToPackage{svgnames}{xcolor}',
}
```

It is possible to insert further uses of the `\sphinxsetup` LaTeX macro directly into the body of the document, via the help of the `raw` directive. This chapter is styled in the PDF output using the following at the start of the chaper:

```
.. raw:: latex

   \begingroup
   \sphinxsetup{%
         verbatimwithframe=false,
         VerbatimColor={named}{OldLace},
         TitleColor={named}{DarkGoldenrod},
         hintBorderColor={named}{LightCoral},
         attentionborder=3pt,
         attentionBorderColor={named}{Crimson},
         attentionBgColor={named}{FloralWhite},
         noteborder=2pt,
         noteBorderColor={named}{Olive},
         cautionborder=3pt,
         cautionBorderColor={named}{Cyan},
         cautionBgColor={named}{LightCyan}}
```

The below is included at the end of the chapter:

```
.. raw:: latex

   \endgroup
```

LaTeX syntax for boolean keys requires *lowercase* `true` or `false` e.g `'sphinxsetup': "verbatimwrapslines=false"`. If setting the boolean key to `true`, `=true` is optional. Spaces around the commas and equal signs are ignored, spaces inside LaTeX macros may be significant.

`hmargin, vmargin` The dimensions of the horizontal (resp. vertical) margins, passed as `hmargin` (resp. `vmargin`) option to the `geometry` package. Example:

```
'sphinxsetup': 'hmargin={2in,1.5in}, vmargin={1.5in,2in}, marginpar=1in',
```

Japanese documents currently accept only the one-dimension format for these parameters. The `geometry` package is then passed suitable options to get the text width set to an exact multiple of the *zenkaku* width, and the text height set to an integer multiple of the baselineskip, with the closest fit for the margins.

Default: `1in` (equivalent to `{1in,1in}`)

Hint: For Japanese `'manual'` docclass with pointsize `11pt` or `12pt`, use the `nomag` extra document class option (cf. `'extraclassoptions'` key of *`latex_elements`*) or so-called TeX "true" units:

```
'sphinxsetup': 'hmargin=1.5truein, vmargin=1.5truein, marginpar=5zw',
```

New in version 1.5.3.

`marginpar` The `\marginparwidth` LaTeX dimension. For Japanese documents, the value is modified to be the closest integer multiple of the *zenkaku* width.

Default: `0.5in`

New in version 1.5.3.

`verbatimwithframe` Boolean to specify if *`code-block`*s and literal includes are framed. Setting it to `false` does not deactivate use of package "framed", because it is still in use for the optional background colour.

Default: `true`.

`verbatimwrapslines` Boolean to specify if long lines in *`code-block`*'s contents are wrapped.

If `true`, line breaks may happen at spaces (the last space before the line break will be rendered using a special symbol), and at ascii punctuation characters (i.e. not at letters or digits). Whenever a long string has no break points, it is moved to next line. If its length is longer than the line width it will overflow.

Default: `true`

`verbatimforcewraps` Boolean to specify if long lines in *`code-block`*'s contents should be forcefully wrapped to never overflow due to long strings.

Note: It is assumed that the Pygments[284] LaTeXFormatter has not been used with its `texcomments` or similar options which allow additional (arbitrary) LaTeX mark-up.

Also, in case of *`latex_engine`* set to `'pdflatex'`, only the default LaTeX handling of Unicode code points, i.e. `utf8` not `utf8x` is allowed.

Default: `false`

New in version 3.5.0.

`verbatimmaxoverfull` A number. If an unbreakable long string has length larger than the total linewidth plus this number of characters, and if `verbatimforcewraps` mode is on, the input line will be reset using the forceful algorithm which applies breakpoints at each character.

Default: `3`

New in version 3.5.0.

`verbatimmaxunderfull` A number. If `verbatimforcewraps` mode applies, and if after applying the line wrapping at spaces and punctuation, the first part of the split line is lacking at least that number of characters to fill the available width, then the input line will be reset using the forceful algorithm.

[284] https://pygments.org/

As the default is set to a high value, the forceful algorithm is triggered only in overfull case, i.e. in presence of a string longer than full linewidth. Set this to `0` to force all input lines to be hard wrapped at the current avaiable linewidth:

```
latex_elements = {
    'sphinxsetup': "verbatimforcewraps, verbatimmaxunderfull=0",
}
```

This can be done locally for a given code-block via the use of raw latex directives to insert suitable `\sphinxsetup` (before and after) into the latex file.

Default: `100`

New in version 3.5.0.

`verbatimhintsturnover` Boolean to specify if code-blocks display "continued on next page" and "continued from previous page" hints in case of pagebreaks.

Default: `true`

New in version 1.6.3.

Changed in version 1.7: the default changed from `false` to `true`.

`verbatimcontinuedalign`, `verbatimcontinuesalign` Horizontal position relative to the framed contents: either `l` (left aligned), `r` (right aligned) or `c` (centered).

Default: `r`

New in version 1.7.

`parsedliteralwraps` Boolean to specify if long lines in parsed-literal[285]'s contents should wrap.

Default: `true`

New in version 1.5.2: set this option value to `false` to recover former behaviour.

`inlineliteralwraps` Boolean to specify if line breaks are allowed inside inline literals: but extra potential break-points (additionally to those allowed by LaTeX at spaces or for hyphenation) are currently inserted only after the characters `. , ; ? ! /` and `\`. Due to TeX internals, white space in the line will be stretched (or shrunk) in order to accommodate the linebreak.

Default: `true`

New in version 1.5: set this option value to `false` to recover former behaviour.

Changed in version 2.3.0: added potential breakpoint at `\` characters.

`verbatimvisiblespace` When a long code line is split, the last space character from the source code line right before the linebreak location is typeset using this.

Default: `\textcolor{red}{\textvisiblespace}`

`verbatimcontinued` A LaTeX macro inserted at start of continuation code lines. Its (complicated…) default typesets a small red hook pointing to the right:

```
\makebox[2\fontcharwd\font`\x][r]{\textcolor{red}{\tiny$\hookrightarrow$}}
```

Changed in version 1.5: The breaking of long code lines was added at 1.4.2. The default definition of the continuation symbol was changed at 1.5 to accomodate various font sizes (e.g. code-blocks can be in footnotes).

`TitleColor` The colour for titles (as configured via use of package "titlesec".)

Default: `{rgb}{0.126,0.263,0.361}`

[285] http://docutils.sourceforge.net/docs/ref/rst/directives.html#parsed-literal

Warning: Colours set via `'sphinxsetup'` must obey the syntax of the argument of the `color/xcolor` packages `\definecolor` command.

`InnerLinkColor` A colour passed to `hyperref` as value of `linkcolor` and `citecolor`.

Default: `{rgb}{0.208,0.374,0.486}`.

`OuterLinkColor` A colour passed to `hyperref` as value of `filecolor`, `menucolor`, and `urlcolor`.

Default: `{rgb}{0.216,0.439,0.388}`

`VerbatimColor` The background colour for *`code-block`*s.

Default: `{rgb}{1,1,1}` (white)

`VerbatimBorderColor` The frame color.

Default: `{rgb}{0,0,0}` (black)

`VerbatimHighlightColor` The color for highlighted lines.

Default: `{rgb}{0.878,1,1}`

New in version 1.6.6.

Note: Starting with this colour, and for all others following, the names declared to "color" or "xcolor" are prefixed with "sphinx".

`verbatimsep` The separation between code lines and the frame.

Default: `\fboxsep`

`verbatimborder` The width of the frame around *`code-block`*s.

Default: `\fboxrule`

`shadowsep` The separation between contents and frame for contents[286] and topic[287] boxes.

Default: `5pt`

`shadowsize` The width of the lateral "shadow" to the right.

Default: `4pt`

`shadowrule` The width of the frame around topic[288] boxes.

Default: `\fboxrule`

`noteBorderColor, hintBorderColor, importantBorderColor, tipBorderColor` The colour for the two horizontal rules used by Sphinx in LaTeX for styling a note[289] type admonition.

Default: `{rgb}{0,0,0}` (black)

`noteborder, hintborder, importantborder, tipborder` The width of the two horizontal rules.

Default: `0.5pt`

`warningBorderColor, and (caution|attention|danger|error)BorderColor` The colour for the admonition frame.

Default: `{rgb}{0,0,0}` (black)

[286] http://docutils.sourceforge.net/docs/ref/rst/directives.html#contents
[287] http://docutils.sourceforge.net/docs/ref/rst/directives.html#topic
[288] http://docutils.sourceforge.net/docs/ref/rst/directives.html#topic
[289] http://docutils.sourceforge.net/docs/ref/rst/directives.html#note

`warningBgColor`, `cautionBgColor`, `attentionBgColor`, `dangerBgColor`, `errorBgColor` The background colours for the respective admonitions.

Default: `{rgb}{1,1,1}` (white)

`warningborder`, `cautionborder`, `attentionborder`, `dangerborder`, `errorborder` The width of the frame.

Default: `1pt`

`AtStartFootnote` LaTeX macros inserted at the start of the footnote text at bottom of page, after the footnote number.

Default: `\mbox{ }`

`BeforeFootnote` LaTeX macros inserted before the footnote mark. The default removes possible space before it (else, TeX could insert a line break there).

Default: `\leavevmode\unskip`

New in version 1.5.

`HeaderFamily` default `\sffamily\bfseries`. Sets the font used by headings.

5.3 LaTeX macros and environments

The "LaTeX package" file `sphinx.sty` loads various components providing support macros (aka commands), and environments, which are used in the mark-up produced on output from the `latex` builder, before conversion to `pdf` via the LaTeX toolchain. Also the "LaTeX class" files `sphinxhowto.cls` and `sphinxmanual.cls` define or customize some environments. All of these files can be found in the latex build repertory.

Some of these provide facilities not available from pre-existing LaTeX packages and work around LaTeX limitations with lists, table cells, verbatim rendering, footnotes, etc…

Others simply define macros with public names to make overwriting their defaults easy via user-added contents to the preamble. We will survey most of those public names here, but defaults have to be looked at in their respective definition files.

Hint: Sphinx LaTeX support code is split across multiple smaller-sized files. Rather than adding code to the preamble via *latex_elements*`['preamble']` it is also possible to replace entirely one of the component files of Sphinx LaTeX code with a custom version, simply by including a modified copy in the project source and adding the filename to the `latex_additional_files` list. Check the LaTeX build repertory for the filenames and contents.

Changed in version 4.0.0: split of `sphinx.sty` into multiple smaller units, to facilitate customization of many aspects simultaneously.

Macros

- Text styling commands:
  - `\sphinxstrong`,
  - `\sphinxbfcode`,
  - `\sphinxemail`,
  - `\sphinxtablecontinued`,
  - `\sphinxtitleref`,
  - `\sphinxmenuselection`,
  - `\sphinxaccelerator`,

  - `\sphinxcrossref`,
  - `\sphinxtermref`,
  - `\sphinxoptional`.

  New in version 1.4.5: Use of `\sphinx` prefixed macro names to limit possibilities of conflict with LaTeX packages.

- More text styling:
  - `\sphinxstyleindexentry`,
  - `\sphinxstyleindexextra`,
  - `\sphinxstyleindexpageref`,
  - `\sphinxstyletopictitle`,
  - `\sphinxstylesidebartitle`,
  - `\sphinxstyleothertitle`,
  - `\sphinxstylesidebarsubtitle`,
  - `\sphinxstyletheadfamily`,
  - `\sphinxstyleemphasis`,
  - `\sphinxstyleliteralemphasis`,
  - `\sphinxstylestrong`,
  - `\sphinxstyleliteralstrong`,
  - `\sphinxstyleabbreviation`,
  - `\sphinxstyleliteralintitle`,
  - `\sphinxstylecodecontinued`,
  - `\sphinxstylecodecontinues`.

  New in version 1.5: These macros were formerly hard-coded as non customizable `\texttt`, `\emph`, etc…

  New in version 1.6: `\sphinxstyletheadfamily` which defaults to `\sffamily` and allows multiple paragraphs in header cells of tables.

  New in version 1.6.3: `\sphinxstylecodecontinued` and `\sphinxstylecodecontinues`.

  New in version 3.0: `\sphinxkeyboard`

- `\sphinxtableofcontents`: A wrapper (defined differently in `sphinxhowto.cls` and in `sphinxmanual.cls`) of standard `\tableofcontents`. The macro `\sphinxtableofcontentshook` is executed during its expansion right before `\tableofcontents` itself.

  Changed in version 1.5: Formerly, the meaning of `\tableofcontents` was modified by Sphinx.

  Changed in version 2.0: Hard-coded redefinitions of `\l@section` and `\l@subsection` formerly done during loading of `'manual'` docclass are now executed later via `\sphinxtableofcontentshook`. This macro is also executed by the `'howto'` docclass, but defaults to empty with it.

- `\sphinxmaketitle`: Used as the default setting of the `'maketitle'` *`latex_elements`* key. Defined in the class files `sphinxmanual.cls` and `sphinxhowto.cls`.

  Changed in version 1.8.3: Formerly, `\maketitle` from LaTeX document class was modified by Sphinx.

- `\sphinxbackoftitlepage`: For `'manual'` docclass, and if it is defined, it gets executed at end of `\sphinxmaketitle`, before the final `\clearpage`. Use either the `'maketitle'` key or the `'preamble'` key of *`latex_elements`* to add a custom definition of `\sphinxbackoftitlepage`.

  New in version 1.8.3.

- `\sphinxcite`: A wrapper of standard `\cite` for citation references.

Environments

- A figure[290] may have an optional legend with arbitrary body elements: they are rendered in a `sphinxlegend` environment. The default definition issues `\small`, and ends with `\par`.

  New in version 1.5.6: Formerly, the `\small` was hardcoded in LaTeX writer and the ending `\par` was lacking.

- Environments associated with admonitions:
  - `sphinxnote`,
  - `sphinxhint`,
  - `sphinximportant`,
  - `sphinxtip`,
  - `sphinxwarning`,
  - `sphinxcaution`,
  - `sphinxattention`,
  - `sphinxdanger`,
  - `sphinxerror`.

  They may be `\renewenvironment` 'd individually, and must then be defined with one argument (it is the heading of the notice, for example `Warning:` for warning[291] directive, if English is the document language). Their default definitions use either the *sphinxheavybox* (for the last 5 ones) or the *sphinxlightbox* environments, configured to use the parameters (colours, border thickness) specific to each type, which can be set via `'sphinxsetup'` string.

  Changed in version 1.5: Use of public environment names, separate customizability of the parameters, such as `noteBorderColor`, `noteborder`, `warningBgColor`, `warningBorderColor`, `warningborder`, …

- The contents[292] directive (with `:local:` option) and the topic[293] directive are implemented by environment `sphinxShadowBox`.

  New in version 1.4.2: Former code refactored into an environment allowing page breaks.

  Changed in version 1.5: Options `shadowsep`, `shadowsize`, `shadowrule`.

- The literal blocks (via `::` or *`code-block`*), are implemented using `sphinxVerbatim` environment which is a wrapper of `Verbatim` environment from package `fancyvrb.sty`. It adds the handling of the top caption and the wrapping of long lines, and a frame which allows pagebreaks. Inside tables the used environment is `sphinxVerbatimintable` (it does not draw a frame, but allows a caption).

  Changed in version 1.5: `Verbatim` keeps exact same meaning as in `fancyvrb.sty` (also under the name `OriginalVerbatim`); `sphinxVerbatimintable` is used inside tables.

  New in version 1.5: Options `verbatimwithframe`, `verbatimwrapslines`, `verbatimsep`, `verbatimborder`.

  New in version 1.6.6: Support for `:emphasize-lines:` option

  New in version 1.6.6: Easier customizability of the formatting via exposed to user LaTeX macros such as `\sphinxVerbatimHighlightLine`.

- The bibliography uses `sphinxthebibliography` and the Python Module index as well as the general index both use `sphinxtheindex`; these environments are wrappers of the `thebibliography` and respectively `theindex` environments as provided by the document class (or packages).

[290] http://docutils.sourceforge.net/docs/ref/rst/directives.html#figure
[291] http://docutils.sourceforge.net/docs/ref/rst/directives.html#warning
[292] http://docutils.sourceforge.net/docs/ref/rst/directives.html#contents
[293] http://docutils.sourceforge.net/docs/ref/rst/directives.html#topic

Changed in version 1.5: Formerly, the original environments were modified by Sphinx.

Miscellany

- Every text paragraph in document body starts with `\sphinxAtStartPar`. Currently, this is used to insert a zero width horizontal skip which is a trick to allow TeX hyphenation of the first word of a paragraph in a narrow context (like a table cell). For `'lualatex'` which does not need the trick, the `\sphinxAtStartPar` does nothing.

  New in version 3.5.0.
- The section, subsection, … headings are set using *titlesec*'s `\titleformat` command.
- For the `'manual'` docclass, the chapter headings can be customized using *fncychap*'s commands `\ChNameVar`, `\ChNumVar`, `\ChTitleVar`. File `sphinx.sty` has custom re-definitions in case of *fncychap* option `Bjarne`.

  Changed in version 1.5: Formerly, use of *fncychap* with other styles than `Bjarne` was dysfunctional.

Hint: As an experimental feature, Sphinx can use user-defined template file for LaTeX source if you have a file named `_templates/latex.tex_t` in your project.

Additional files `longtable.tex_t`, `tabulary.tex_t` and `tabular.tex_t` can be added to `_templates/` to configure some aspects of table rendering (such as the caption position).

New in version 1.6: currently all template variables are unstable and undocumented.

CHAPTER

SIX

DEVELOPING EXTENSIONS FOR SPHINX

Since many projects will need special features in their documentation, Sphinx is designed to be extensible on several levels.

Here are a few things you can do in an extension:

- Add new *builder*s to support new output formats or actions on the parsed documents.
- Register custom reStructuredText roles and directives, extending the markup using the *Docutils markup API*.
- Add custom code to so-called "hook points" at strategic places throughout the build process, allowing you to register a hook and run specialized code. For example, see the *Sphinx core events*.

An extension is simply a Python module with a `setup()` function. A user activates the extension by placing the extension's module name (or a sub-module) in their `extensions` configuration value.

When **`sphinx-build`** is executed, Sphinx will attempt to import each module that is listed, and execute `yourmodule.setup(app)`. This function is used to prepare the extension (e.g., by executing Python code), linking resources that Sphinx uses in the build process (like CSS or HTML files), and notifying Sphinx of everything the extension offers (such as directive or role definitions). The `app` argument is an instance of `Sphinx` and gives you control over most aspects of the Sphinx build.

Note: The configuration file itself can be treated as an extension if it contains a `setup()` function. All other extensions to load must be listed in the `extensions` configuration value.

The rest of this page describes some high-level aspects of developing extensions and various parts of Sphinx's behavior that you can control. For some examples of how extensions can be built and used to control different parts of Sphinx, see the *Extension tutorials*.

6.1 Important objects

There are several key objects whose API you will use while writing an extension. These are:

Application The application object (usually called `app`) is an instance of `Sphinx`. It controls most high-level functionality, such as the setup of extensions, event dispatching and producing output (logging).

If you have the environment object, the application is available as `env.app`.

Environment The build environment object (usually called `env`) is an instance of `BuildEnvironment`. It is responsible for parsing the source documents, stores all metadata about the document collection and is serialized to disk after each build.

Its API provides methods to do with access to metadata, resolving references, etc. It can also be used by extensions to cache information that should persist for incremental rebuilds.

If you have the application or builder object, the environment is available as `app.env` or `builder.env`.

Builder The builder object (usually called `builder`) is an instance of a specific subclass of *`Builder`*. Each builder class knows how to convert the parsed documents into an output format, or otherwise process them (e.g. check external links).

If you have the application object, the builder is available as `app.builder`.

Config The config object (usually called `config`) provides the values of configuration values set in `conf.py` as attributes. It is an instance of *`Config`*.

The config is available as `app.config` or `env.config`.

To see an example of use of these objects, refer to *Extension tutorials*.

6.2 Build Phases

One thing that is vital in order to understand extension mechanisms is the way in which a Sphinx project is built: this works in several phases.

Phase 0: Initialization

In this phase, almost nothing of interest to us happens. The source directory is searched for source files, and extensions are initialized. Should a stored build environment exist, it is loaded, otherwise a new one is created.

Phase 1: Reading

In Phase 1, all source files (and on subsequent builds, those that are new or changed) are read and parsed. This is the phase where directives and roles are encountered by docutils, and the corresponding code is executed. The output of this phase is a *doctree* for each source file; that is a tree of docutils nodes. For document elements that aren't fully known until all existing files are read, temporary nodes are created.

There are nodes provided by docutils, which are documented in the docutils documentation[294]. Additional nodes are provided by Sphinx and *documented here*.

During reading, the build environment is updated with all meta- and cross reference data of the read documents, such as labels, the names of headings, described Python objects and index entries. This will later be used to replace the temporary nodes.

The parsed doctrees are stored on the disk, because it is not possible to hold all of them in memory.

Phase 2: Consistency checks

Some checking is done to ensure no surprises in the built documents.

Phase 3: Resolving

Now that the metadata and cross-reference data of all existing documents is known, all temporary nodes are replaced by nodes that can be converted into output using components called transforms. For example, links are created for object references that exist, and simple literal nodes are created for those that don't.

Phase 4: Writing

This phase converts the resolved doctrees to the desired output format, such as HTML or LaTeX. This happens via a so-called docutils writer that visits the individual nodes of each doctree and produces some output in the process.

Note: Some builders deviate from this general build plan, for example, the builder that checks external links does not need anything more than the parsed doctrees and therefore does not have phases 2–4.

To see an example of application, refer to *Developing a "TODO" extension*.

[294] http://docutils.sourceforge.net/docs/ref/doctree.html

6.3 Extension metadata

New in version 1.3.

The `setup()` function can return a dictionary. This is treated by Sphinx as metadata of the extension. Metadata keys currently recognized are:

- `'version'`: a string that identifies the extension version. It is used for extension version requirement checking (see *`needs_extensions`*) and informational purposes. If not given, `"unknown version"` is substituted.
- `'env_version'`: an integer that identifies the version of env data structure if the extension stores any data to environment. It is used to detect the data structure has been changed from last build. The extensions have to increment the version when data structure has changed. If not given, Sphinx considers the extension does not stores any data to environment.
- `'parallel_read_safe'`: a boolean that specifies if parallel reading of source files can be used when the extension is loaded. It defaults to `False`, i.e. you have to explicitly specify your extension to be parallel-read-safe after checking that it is.
- `'parallel_write_safe'`: a boolean that specifies if parallel writing of output files can be used when the extension is loaded. Since extensions usually don't negatively influence the process, this defaults to `True`.

6.4 APIs used for writing extensions

These sections provide a more complete description of the tools at your disposal when developing Sphinx extensions. Some are core to Sphinx (such as the *Application API*) while others trigger specific behavior (such as the *i18n API*)

Application API

Each Sphinx extension is a Python module with at least a `setup()` function. This function is called at initialization time with one argument, the application object representing the Sphinx process.

class `sphinx.application.`**`Sphinx`**

This application object has the public API described in the following.

Extension setup

These methods are usually called in an extension's `setup()` function.

Examples of using the Sphinx extension API can be seen in the `sphinx.ext` package.

`Sphinx.`**`setup_extension`**(*extname: str*[295]) → None[296]

Import and setup a Sphinx extension module.

Load the extension given by the module *name*. Use this if your extension needs the features provided by another extension. No-op if called twice.

`Sphinx.`**`require_sphinx`**(*version: str*[297]) → None[298]

Check the Sphinx version if requested.

Compare *version* with the version of the running Sphinx, and abort the build when it is too old.

Parameters **`version`** – The required version in the form of `major.minor`.

New in version 1.0.

[295] https://docs.python.org/3/library/stdtypes.html#str
[296] https://docs.python.org/3/library/constants.html#None

`Sphinx.connect`(*event: str*[299], *callback: Callable*, *priority: int*[300] *= 500*) → int[301]

Register *callback* to be called when *event* is emitted.

For details on available core events and the arguments of callback functions, please see *Sphinx core events*.

Parameters

- **`event`** – The name of target event
- **`callback`** – Callback function for the event
- **`priority`** – The priority of the callback. The callbacks will be invoked in the order of *priority* in asending.

Returns A listener ID. It can be used for `disconnect()`.

Changed in version 3.0: Support *priority*

`Sphinx.disconnect`(*listener_id: int*[302]) → None[303]

Unregister callback by *listener_id*.

Parameters **`listener_id`** – A listener_id that `connect()` returns

`Sphinx.add_builder`(*builder: Type[Builder]*, *override: bool*[304] *= False*) → None[305]

Register a new builder.

Parameters

- **`builder`** – A builder class
- **`override`** – If true, install the builder forcedly even if another builder is already installed as the same name

Changed in version 1.8: Add *override* keyword.

`Sphinx.add_config_value`(*name: str*[306], *default: Any*, *rebuild: Union[bool*[307], *str*[308]*]*, *types: Any = ()*) → None[309]

Register a configuration value.

This is necessary for Sphinx to recognize new values and set default values accordingly.

Parameters

- **`name`** – The name of configuration value. It is recommended to be prefixed with the extension name (ex. `html_logo`, `epub_title`)
- **`default`** – The default value of the configuration.
- **`rebuild`** – The condition of rebuild. It must be one of those values:
  - `'env'` if a change in the setting only takes effect when a document is parsed – this means that the whole environment must be rebuilt.
  - `'html'` if a change in the setting needs a full rebuild of HTML documents.
  - `''` if a change in the setting will not need any special rebuild.
- **`types`** – The type of configuration value. A list of types can be specified. For example, `[str]` is used to describe a configuration that takes string value.

[297] https://docs.python.org/3/library/stdtypes.html#str
[298] https://docs.python.org/3/library/constants.html#None
[299] https://docs.python.org/3/library/stdtypes.html#str
[300] https://docs.python.org/3/library/functions.html#int
[301] https://docs.python.org/3/library/functions.html#int
[302] https://docs.python.org/3/library/functions.html#int
[303] https://docs.python.org/3/library/constants.html#None
[304] https://docs.python.org/3/library/functions.html#bool
[305] https://docs.python.org/3/library/constants.html#None

Changed in version 0.4: If the *default* value is a callable, it will be called with the config object as its argument in order to get the default value. This can be used to implement config values whose default depends on other values.

Changed in version 0.6: Changed *rebuild* from a simple boolean (equivalent to `''` or `'env'`) to a string. However, booleans are still accepted and converted internally.

`Sphinx.`**`add_event`**`(`*name: str*[310]`)` → None[311]

Register an event called *name*.

This is needed to be able to emit it.

Parameters **`name`** – The name of the event

`Sphinx.`**`set_translator`**`(`*name: str*[312], *translator_class: Type[docutils.nodes.NodeVisitor]*, *override: bool*[313] *= False*`)` → None[314]

Register or override a Docutils translator class.

This is used to register a custom output translator or to replace a builtin translator. This allows extensions to use custom translator and define custom nodes for the translator (see `add_node()`).

Parameters

- **`name`** – The name of builder for the translator
- **`translator_class`** – A translator class
- **`override`** – If true, install the translator forcedly even if another translator is already installed as the same name

New in version 1.3.

Changed in version 1.8: Add *override* keyword.

`Sphinx.`**`add_node`**`(`*node: Type[docutils.nodes.Element]*, *override: bool*[315] *= False*, ***kwargs: Tuple[Callable, Callable]*`)` → None[316]

Register a Docutils node class.

This is necessary for Docutils internals. It may also be used in the future to validate nodes in the parsed documents.

Parameters

- **`node`** – A node class
- **`kwargs`** – Visitor functions for each builder (see below)
- **`override`** – If true, install the node forcedly even if another node is already installed as the same name

Node visitor functions for the Sphinx HTML, LaTeX, text and manpage writers can be given as keyword arguments: the keyword should be one or more of `'html'`, `'latex'`, `'text'`, `'man'`, `'texinfo'` or any other supported translators, the value a 2-tuple of `(visit, depart)` methods. `depart` can be `None` if the `visit` function raises `docutils.nodes.SkipNode`. Example:

[306] https://docs.python.org/3/library/stdtypes.html#str
[307] https://docs.python.org/3/library/functions.html#bool
[308] https://docs.python.org/3/library/stdtypes.html#str
[309] https://docs.python.org/3/library/constants.html#None
[310] https://docs.python.org/3/library/stdtypes.html#str
[311] https://docs.python.org/3/library/constants.html#None
[312] https://docs.python.org/3/library/stdtypes.html#str
[313] https://docs.python.org/3/library/functions.html#bool
[314] https://docs.python.org/3/library/constants.html#None

```
class math(docutils.nodes.Element): pass

def visit_math_html(self, node):
    self.body.append(self.starttag(node, 'math'))
def depart_math_html(self, node):
    self.body.append('</math>')

app.add_node(math, html=(visit_math_html, depart_math_html))
```

Obviously, translators for which you don't specify visitor methods will choke on the node when encountered in a document to translate.

Changed in version 0.5: Added the support for keyword arguments giving visit functions.

`Sphinx.`**`add_enumerable_node`**(*node: Type[docutils.nodes.Element]*, *figtype: str*[317], *title_getter: Optional[Callable[[docutils.nodes.Node], str*[318]*]] = None*, *override: bool*[319] *= False*, ***kwargs: Tuple[Callable, Callable]*) → None[320]

Register a Docutils node class as a numfig target.

Sphinx numbers the node automatically. And then the users can refer it using *numref*.

Parameters

- **node** – A node class
- **figtype** – The type of enumerable nodes. Each figtypes have individual numbering sequences. As a system figtypes, `figure`, `table` and `code-block` are defined. It is able to add custom nodes to these default figtypes. It is also able to define new custom figtype if new figtype is given.
- **title_getter** – A getter function to obtain the title of node. It takes an instance of the enumerable node, and it must return its title as string. The title is used to the default title of references for *ref*. By default, Sphinx searches `docutils.nodes.caption` or `docutils.nodes.title` from the node as a title.
- **kwargs** – Visitor functions for each builder (same as *add_node()*)
- **override** – If true, install the node forcedly even if another node is already installed as the same name

New in version 1.4.

`Sphinx.`**`add_directive`**(*name: str*[321], *cls: Type[docutils.parsers.rst.Directive]*, *override: bool*[322] *= False*) → None[323]

Register a Docutils directive.

Parameters

- **name** – The name of directive
- **cls** – A directive class
- **override** – If true, install the directive forcedly even if another directive is already installed as the same name

For example, a custom directive named `my-directive` would be added like this:

[315] https://docs.python.org/3/library/functions.html#bool
[316] https://docs.python.org/3/library/constants.html#None
[317] https://docs.python.org/3/library/stdtypes.html#str
[318] https://docs.python.org/3/library/stdtypes.html#str
[319] https://docs.python.org/3/library/functions.html#bool
[320] https://docs.python.org/3/library/constants.html#None

```python
from docutils.parsers.rst import Directive, directives

class MyDirective(Directive):
    has_content = True
    required_arguments = 1
    optional_arguments = 0
    final_argument_whitespace = True
    option_spec = {
        'class': directives.class_option,
        'name': directives.unchanged,
    }

    def run(self):
        ...

def setup(app):
    add_directive('my-directive', MyDirective)
```

For more details, see the Docutils docs[324] .

Changed in version 0.6: Docutils 0.5-style directive classes are now supported.

Deprecated since version 1.8: Docutils 0.4-style (function based) directives support is deprecated.

Changed in version 1.8: Add *override* keyword.

`Sphinx.`**`add_role`**(*name: str*[325], *role: Any*, *override: bool*[326] *= False*) → None[327]

Register a Docutils role.

Parameters

- **`name`** – The name of role
- **`role`** – A role function
- **`override`** – If true, install the role forcedly even if another role is already installed as the same name

For more details about role functions, see the Docutils docs[328] .

Changed in version 1.8: Add *override* keyword.

`Sphinx.`**`add_generic_role`**(*name: str*[329], *nodeclass: Any*, *override: bool*[330] *= False*) → None[331]

Register a generic Docutils role.

Register a Docutils role that does nothing but wrap its contents in the node given by *nodeclass*.

If *override* is True, the given *nodeclass* is forcedly installed even if a role named as *name* is already installed.

New in version 0.6.

Changed in version 1.8: Add *override* keyword.

[321] https://docs.python.org/3/library/stdtypes.html#str
[322] https://docs.python.org/3/library/functions.html#bool
[323] https://docs.python.org/3/library/constants.html#None
[324] http://docutils.sourceforge.net/docs/howto/rst-directives.html
[325] https://docs.python.org/3/library/stdtypes.html#str
[326] https://docs.python.org/3/library/functions.html#bool
[327] https://docs.python.org/3/library/constants.html#None
[328] http://docutils.sourceforge.net/docs/howto/rst-roles.html
[329] https://docs.python.org/3/library/stdtypes.html#str
[330] https://docs.python.org/3/library/functions.html#bool
[331] https://docs.python.org/3/library/constants.html#None

`Sphinx.add_domain`(*domain: Type[*sphinx.domains.Domain*], override: bool*[332] *= False*) → None[333]

Register a domain.

Parameters

- **domain** – A domain class
- **override** – If true, install the domain forcedly even if another domain is already installed as the same name

New in version 1.0.

Changed in version 1.8: Add *override* keyword.

`Sphinx.add_directive_to_domain`(*domain: str*[334], *name: str*[335], *cls: Type[*docutils.parsers.rst.Directive*]*, *override: bool*[336] *= False*) → None[337]

Register a Docutils directive in a domain.

Like `add_directive()`, but the directive is added to the domain named *domain*.

Parameters

- **domain** – The name of target domain
- **name** – A name of directive
- **cls** – A directive class
- **override** – If true, install the directive forcedly even if another directive is already installed as the same name

New in version 1.0.

Changed in version 1.8: Add *override* keyword.

`Sphinx.add_role_to_domain`(*domain: str*[338], *name: str*[339], *role: Union[Callable[[str*[340], *str*[341], *str*[342], *int*[343], *docutils.parsers.rst.states.Inliner, Dict[str*[344], *Any], List[str*[345]*]], Tuple[List[docutils.nodes.Node], List[docutils.nodes.system_message]]], sphinx.roles.XRefRole]*, *override: bool*[346] *= False*) → None[347]

Register a Docutils role in a domain.

Like `add_role()`, but the role is added to the domain named *domain*.

Parameters

- **domain** – The name of target domain
- **name** – A name of role
- **role** – A role function
- **override** – If true, install the role forcedly even if another role is already installed as the same name

New in version 1.0.

Changed in version 1.8: Add *override* keyword.

[332] https://docs.python.org/3/library/functions.html#bool
[333] https://docs.python.org/3/library/constants.html#None
[334] https://docs.python.org/3/library/stdtypes.html#str
[335] https://docs.python.org/3/library/stdtypes.html#str
[336] https://docs.python.org/3/library/functions.html#bool
[337] https://docs.python.org/3/library/constants.html#None

Sphinx.**add_index_to_domain**(*domain: str*[348], *index: Type[sphinx.domains.Index]*, *override: bool*[349] = *False*) → None[350]

Register a custom index for a domain.

Add a custom *index* class to the domain named *domain*.

Parameters

- **domain** – The name of target domain
- **index** – A index class
- **override** – If true, install the index forcedly even if another index is already installed as the same name

New in version 1.0.

Changed in version 1.8: Add *override* keyword.

Sphinx.**add_object_type**(*directivename: str*[351], *rolename: str*[352], *indextemplate: str*[353] = *''*, *parse_node: Optional[Callable]* = *None*, *ref_nodeclass: Optional[Type[docutils.nodes.TextElement]]* = *None*, *objname: str*[354] = *''*, *doc_field_types: List* = *[]*, *override: bool*[355] = *False*) → None[356]

Register a new object type.

This method is a very convenient way to add a new *object* type that can be cross-referenced. It will do this:

- Create a new directive (called *directivename*) for documenting an object. It will automatically add index entries if *indextemplate* is nonempty; if given, it must contain exactly one instance of `%s`. See the example below for how the template will be interpreted.
- Create a new role (called *rolename*) to cross-reference to these object descriptions.
- If you provide *parse_node*, it must be a function that takes a string and a docutils node, and it must populate the node with children parsed from the string. It must then return the name of the item to be used in cross-referencing and index entries. See the `conf.py` file in the source for this documentation for an example.
- The *objname* (if not given, will default to *directivename*) names the type of object. It is used when listing objects, e.g. in search results.

For example, if you have this call in a custom Sphinx extension:

```
app.add_object_type('directive', 'dir', 'pair: %s; directive')
```

you can use this markup in your documents:

```
.. rst:directive:: function

   Document a function.

<...>
```

(continues on next page)

[338] https://docs.python.org/3/library/stdtypes.html#str
[339] https://docs.python.org/3/library/stdtypes.html#str
[340] https://docs.python.org/3/library/stdtypes.html#str
[341] https://docs.python.org/3/library/stdtypes.html#str
[342] https://docs.python.org/3/library/stdtypes.html#str
[343] https://docs.python.org/3/library/functions.html#int
[344] https://docs.python.org/3/library/stdtypes.html#str
[345] https://docs.python.org/3/library/stdtypes.html#str
[346] https://docs.python.org/3/library/functions.html#bool
[347] https://docs.python.org/3/library/constants.html#None
[348] https://docs.python.org/3/library/stdtypes.html#str
[349] https://docs.python.org/3/library/functions.html#bool
[350] https://docs.python.org/3/library/constants.html#None

(continued from previous page)

```
See also the :rst:dir:`function` directive.
```

For the directive, an index entry will be generated as if you had prepended

```
.. index:: pair: function; directive
```

The reference node will be of class `literal` (so it will be rendered in a proportional font, as appropriate for code) unless you give the *ref_nodeclass* argument, which must be a docutils node class. Most useful are `docutils.nodes.emphasis` or `docutils.nodes.strong` – you can also use `docutils.nodes.generated` if you want no further text decoration. If the text should be treated as literal (e.g. no smart quote replacement), but not have typewriter styling, use `sphinx.addnodes.literal_emphasis` or `sphinx.addnodes.literal_strong`.

For the role content, you have the same syntactical possibilities as for standard Sphinx roles (see *Cross-referencing syntax*).

If *override* is True, the given object_type is forcedly installed even if an object_type having the same name is already installed.

Changed in version 1.8: Add *override* keyword.

`Sphinx.`**`add_crossref_type`**(*directivename: str*[357], *rolename: str*[358], *indextemplate: str*[359] *= ''*, *ref_nodeclass: Optional[Type[docutils.nodes.TextElement]] = None*, *objname: str*[360] *= ''*, *override: bool*[361] *= False*) → None[362]

Register a new crossref object type.

This method is very similar to *`add_object_type()`* except that the directive it generates must be empty, and will produce no output.

That means that you can add semantic targets to your sources, and refer to them using custom roles instead of generic ones (like *`ref`*). Example call:

```
app.add_crossref_type('topic', 'topic', 'single: %s',
                      docutils.nodes.emphasis)
```

Example usage:

```
.. topic:: application API

The application API
-------------------

Some random text here.

See also :topic:`this section <application API>`.
```

(Of course, the element following the `topic` directive needn't be a section.)

If *override* is True, the given crossref_type is forcedly installed even if a crossref_type having the same name is already installed.

Changed in version 1.8: Add *override* keyword.

[351] https://docs.python.org/3/library/stdtypes.html#str
[352] https://docs.python.org/3/library/stdtypes.html#str
[353] https://docs.python.org/3/library/stdtypes.html#str
[354] https://docs.python.org/3/library/stdtypes.html#str
[355] https://docs.python.org/3/library/functions.html#bool
[356] https://docs.python.org/3/library/constants.html#None

`Sphinx.add_transform`(*transform: Type[docutils.transforms.Transform]*) → None[363]

Register a Docutils transform to be applied after parsing.

Add the standard docutils `Transform` subclass *transform* to the list of transforms that are applied after Sphinx parses a reST document.

Parameters **`transform`** – A transform class

Table 1: priority range categories for Sphinx transforms

Priority	Main purpose in Sphinx
0-99	Fix invalid nodes by docutils. Translate a doctree.
100-299	Preparation
300-399	early
400-699	main
700-799	Post processing. Deadline to modify text and referencing.
800-899	Collect referencing and referenced nodes. Domain processing.
900-999	Finalize and clean up.

refs: Transform Priority Range Categories[364]

`Sphinx.add_post_transform`(*transform: Type[docutils.transforms.Transform]*) → None[365]

Register a Docutils transform to be applied before writing.

Add the standard docutils `Transform` subclass *transform* to the list of transforms that are applied before Sphinx writes a document.

Parameters **`transform`** – A transform class

`Sphinx.add_js_file`(*filename: str[366], priority: int[367] = 500, **kwargs: Any*) → None[368]

Register a JavaScript file to include in the HTML output.

Add *filename* to the list of JavaScript files that the default HTML template will include in order of *priority* (ascending). The filename must be relative to the HTML static path , or a full URI with scheme. If the priority of JavaScript file is the same as others, the JavaScript files will be included in order of the registration. If the keyword argument `body` is given, its value will be added between the `<script>` tags. Extra keyword arguments are included as attributes of the `<script>` tag.

Example:

```
app.add_js_file('example.js')
# => <script src="_static/example.js"></script>

app.add_js_file('example.js', async="async")
# => <script src="_static/example.js" async="async"></script>

app.add_js_file(None, body="var myVariable = 'foo';")
# => <script>var myVariable = 'foo';</script>
```

[357] https://docs.python.org/3/library/stdtypes.html#str
[358] https://docs.python.org/3/library/stdtypes.html#str
[359] https://docs.python.org/3/library/stdtypes.html#str
[360] https://docs.python.org/3/library/stdtypes.html#str
[361] https://docs.python.org/3/library/functions.html#bool
[362] https://docs.python.org/3/library/constants.html#None
[363] https://docs.python.org/3/library/constants.html#None
[364] http://docutils.sourceforge.net/docs/ref/transforms.html#transform-priority-range-categories
[365] https://docs.python.org/3/library/constants.html#None

Table 2: priority range for JavaScript files

Priority	Main purpose in Sphinx
200	default priority for built-in JavaScript files
500	default priority for extensions
800	default priority for *html_js_files*

A JavaScript file can be added to the specific HTML page when on extension calls this method on *html-page-context* event.

New in version 0.5.

Changed in version 1.8: Renamed from `app.add_javascript()`. And it allows keyword arguments as attributes of script tag.

Changed in version 3.5: Take priority argument. Allow to add a JavaScript file to the specific page.

`Sphinx.`**`add_css_file`**(*filename: str*[369], *priority: int*[370] *= 500*, ***kwargs: Any*) → None[371]

Register a stylesheet to include in the HTML output.

Add *filename* to the list of CSS files that the default HTML template will include in order of *priority* (ascending). The filename must be relative to the HTML static path, or a full URI with scheme. If the priority of CSS file is the same as others, the CSS files will be included in order of the registration. The keyword arguments are also accepted for attributes of `<link>` tag.

Example:

```
app.add_css_file('custom.css')
# => <link rel="stylesheet" href="_static/custom.css" type="text/css" />

app.add_css_file('print.css', media='print')
# => <link rel="stylesheet" href="_static/print.css"
#          type="text/css" media="print" />

app.add_css_file('fancy.css', rel='alternate stylesheet', title='fancy')
# => <link rel="alternate stylesheet" href="_static/fancy.css"
#          type="text/css" title="fancy" />
```

Table 3: priority range for CSS files

Priority	Main purpose in Sphinx
200	default priority for built-in CSS files
500	default priority for extensions
800	default priority for *html_css_files*

A CSS file can be added to the specific HTML page when on extension calls this method on *html-page-context* event.

New in version 1.0.

Changed in version 1.6: Optional `alternate` and/or `title` attributes can be supplied with the *alternate* (of boolean type) and *title* (a string) arguments. The default is no title and *alternate* = `False`. For more information, refer to the documentation[372].

Changed in version 1.8: Renamed from `app.add_stylesheet()`. And it allows keyword arguments as attributes of link tag.

[366] https://docs.python.org/3/library/stdtypes.html#str
[367] https://docs.python.org/3/library/functions.html#int
[368] https://docs.python.org/3/library/constants.html#None

Changed in version 3.5: Take priority argument. Allow to add a CSS file to the specific page.

`Sphinx.add_latex_package`(*packagename: str*[373], *options: Optional[str*[374]*] = None*, *after_hyperref: bool*[375] *= False*) → None[376]

Register a package to include in the LaTeX source code.

Add *packagename* to the list of packages that LaTeX source code will include. If you provide *options*, it will be taken to *usepackage* declaration. If you set *after_hyperref* truthy, the package will be loaded after `hyperref` package.

```
app.add_latex_package('mypackage')
# => \usepackage{mypackage}
app.add_latex_package('mypackage', 'foo,bar')
# => \usepackage[foo,bar]{mypackage}
```

New in version 1.3.

New in version 3.1: *after_hyperref* option.

`Sphinx.add_lexer`(*alias: str*[377], *lexer: Type[pygments.lexer.Lexer]*) → None[378]

Register a new lexer for source code.

Use *lexer* to highlight code blocks with the given language *alias*.

New in version 0.6.

Changed in version 2.1: Take a lexer class as an argument. An instance of lexers are still supported until Sphinx-3.x.

`Sphinx.add_autodocumenter`(*cls: Any*, *override: bool*[379] *= False*) → None[380]

Register a new documenter class for the autodoc extension.

Add *cls* as a new documenter class for the `sphinx.ext.autodoc` extension. It must be a subclass of `sphinx.ext.autodoc.Documenter`. This allows to auto-document new types of objects. See the source of the autodoc module for examples on how to subclass `Documenter`.

If *override* is True, the given *cls* is forcedly installed even if a documenter having the same name is already installed.

Todo: Add real docs for Documenter and subclassing

New in version 0.6.

Changed in version 2.2: Add *override* keyword.

`Sphinx.add_autodoc_attrgetter`(*typ: Type*, *getter: Callable[[Any, str*[381]*, Any], Any]*) → None[382]

Register a new `getattr`-like function for the autodoc extension.

Add *getter*, which must be a function with an interface compatible to the `getattr()`[383] builtin, as the autodoc attribute getter for objects that are instances of *typ*. All cases where autodoc needs to get an attribute of a type are then handled by this function instead of `getattr()`[384].

[369] https://docs.python.org/3/library/stdtypes.html#str
[370] https://docs.python.org/3/library/functions.html#int
[371] https://docs.python.org/3/library/constants.html#None
[372] https://mdn.io/Web/CSS/Alternative_style_sheets
[373] https://docs.python.org/3/library/stdtypes.html#str
[374] https://docs.python.org/3/library/stdtypes.html#str
[375] https://docs.python.org/3/library/functions.html#bool
[376] https://docs.python.org/3/library/constants.html#None
[377] https://docs.python.org/3/library/stdtypes.html#str
[378] https://docs.python.org/3/library/constants.html#None
[379] https://docs.python.org/3/library/functions.html#bool
[380] https://docs.python.org/3/library/constants.html#None

New in version 0.6.

`Sphinx.add_search_language`(*cls: Any*) → None[385]

Register a new language for the HTML search index.

Add *cls*, which must be a subclass of `sphinx.search.SearchLanguage`, as a support language for building the HTML full-text search index. The class must have a *lang* attribute that indicates the language it should be used for. See `html_search_language`.

New in version 1.1.

`Sphinx.add_source_suffix`(*suffix: str*[386], *filetype: str*[387], *override: bool*[388] *= False*) → None[389]

Register a suffix of source files.

Same as `source_suffix`. The users can override this using the setting.

If *override* is True, the given *suffix* is forcedly installed even if a same suffix is already installed.

New in version 1.8.

`Sphinx.add_source_parser`(*parser: Type[docutils.parsers.Parser]*, *override: bool*[390] *= False*) → None[391]

Register a parser class.

If *override* is True, the given *parser* is forcedly installed even if a parser for the same suffix is already installed.

New in version 1.4.

Changed in version 1.8: *suffix* argument is deprecated. It only accepts *parser* argument. Use `add_source_suffix()` API to register suffix instead.

Changed in version 1.8: Add *override* keyword.

`Sphinx.add_env_collector`(*collector: Type[*sphinx.environment.collectors.EnvironmentCollector*]*) → None[392]

Register an environment collector class.

Refer to *Environment Collector API*.

New in version 1.6.

`Sphinx.add_html_theme`(*name: str*[393], *theme_path: str*[394]) → None[395]

Register a HTML Theme.

The *name* is a name of theme, and *path* is a full path to the theme (refs: *Distribute your theme as a Python package*).

New in version 1.6.

`Sphinx.add_html_math_renderer`(*name: str*[396], *inline_renderers: Optional[Tuple[Callable, Callable]] = None*, *block_renderers: Optional[Tuple[Callable, Callable]] = None*) → None[397]

Register a math renderer for HTML.

[381] https://docs.python.org/3/library/stdtypes.html#str
[382] https://docs.python.org/3/library/constants.html#None
[383] https://docs.python.org/3/library/functions.html#getattr
[384] https://docs.python.org/3/library/functions.html#getattr
[385] https://docs.python.org/3/library/constants.html#None
[386] https://docs.python.org/3/library/stdtypes.html#str
[387] https://docs.python.org/3/library/stdtypes.html#str
[388] https://docs.python.org/3/library/functions.html#bool
[389] https://docs.python.org/3/library/constants.html#None
[390] https://docs.python.org/3/library/functions.html#bool
[391] https://docs.python.org/3/library/constants.html#None
[392] https://docs.python.org/3/library/constants.html#None
[393] https://docs.python.org/3/library/stdtypes.html#str
[394] https://docs.python.org/3/library/stdtypes.html#str
[395] https://docs.python.org/3/library/constants.html#None

The *name* is a name of math renderer. Both *inline_renderers* and *block_renderers* are used as visitor functions for the HTML writer: the former for inline math node (`nodes.math`), the latter for block math node (`nodes.math_block`). Regarding visitor functions, see *add_node()* for details.

New in version 1.8.

`Sphinx.`**`add_message_catalog`**(*catalog: str*[398], *locale_dir: str*[399]) → None[400]

Register a message catalog.

Parameters

- **`catalog`** – A name of catalog
- **`locale_dir`** – The base path of message catalog

For more details, see *sphinx.locale.get_translation()*.

New in version 1.8.

`Sphinx.`**`is_parallel_allowed`**(*typ: str*[401]) → bool[402]

Check parallel processing is allowed or not.

Parameters **`typ`** – A type of processing; `'read'` or `'write'`.

`exception` `sphinx.application.`**`ExtensionError`**

All these methods raise this exception if something went wrong with the extension API.

Emitting events

`class` `sphinx.application.`**`Sphinx`**

`emit`(*event: str*[403], **args: Any*, *allowed_exceptions: Tuple[Type[Exception*[404]*], . . .] = ()*) → List

Emit *event* and pass *arguments* to the callback functions.

Return the return values of all callbacks as a list. Do not emit core Sphinx events in extensions!

Parameters

- **`event`** – The name of event that will be emitted
- **`args`** – The arguments for the event
- **`allowed_exceptions`** – The list of exceptions that are allowed in the callbacks

Changed in version 3.1: Added *allowed_exceptions* to specify path-through exceptions

`emit_firstresult`(*event: str*[405], **args: Any*, *allowed_exceptions: Tuple[Type[Exception*[406]*], . . .] = ()*) → Any

Emit *event* and pass *arguments* to the callback functions.

Return the result of the first callback that doesn't return `None`.

Parameters

- **`event`** – The name of event that will be emitted
- **`args`** – The arguments for the event
- **`allowed_exceptions`** – The list of exceptions that are allowed in the callbacks

[396] https://docs.python.org/3/library/stdtypes.html#str
[397] https://docs.python.org/3/library/constants.html#None
[398] https://docs.python.org/3/library/stdtypes.html#str
[399] https://docs.python.org/3/library/stdtypes.html#str
[400] https://docs.python.org/3/library/constants.html#None
[401] https://docs.python.org/3/library/stdtypes.html#str
[402] https://docs.python.org/3/library/functions.html#bool

New in version 0.5.

Changed in version 3.1: Added *allowed_exceptions* to specify path-through exceptions

Sphinx runtime information

The application object also provides runtime information as attributes.

`Sphinx.`**`project`**
Target project. See *`Project`*.

`Sphinx.`**`srcdir`**
Source directory.

`Sphinx.`**`confdir`**
Directory containing `conf.py`.

`Sphinx.`**`doctreedir`**
Directory for storing pickled doctrees.

`Sphinx.`**`outdir`**
Directory for storing built document.

Sphinx core events

These events are known to the core. The arguments shown are given to the registered event handlers. Use *`Sphinx.connect()`* in an extension's `setup` function (note that `conf.py` can also have a `setup` function) to connect handlers to the events. Example:

```
def source_read_handler(app, docname, source):
    print('do something here...')

def setup(app):
    app.connect('source-read', source_read_handler)
```

Below is an overview of each event that happens during a build. In the list below, we include the event name, its callback parameters, and the input and output type for that event:

```
1. event.config-inited(app,config)
2. event.builder-inited(app)
3. event.env-get-outdated(app, env, added, changed, removed)
4. event.env-before-read-docs(app, env, docnames)

for docname in docnames:
   5. event.env-purge-doc(app, env, docname)

   if doc changed and not removed:
      6. source-read(app, docname, source)
      7. run source parsers: text -> docutils.document
         - parsers can be added with the app.add_source_parser() API
      8. apply transforms based on priority: docutils.document -> docutils.document
         - event.doctree-read(app, doctree) is called in the middle of transforms,
```

(continues on next page)

[403] https://docs.python.org/3/library/stdtypes.html#str
[404] https://docs.python.org/3/library/exceptions.html#Exception
[405] https://docs.python.org/3/library/stdtypes.html#str
[406] https://docs.python.org/3/library/exceptions.html#Exception

(continued from previous page)

```
           transforms come before/after this event depending on their priority.

9. event.env-merged-info(app, env, docnames, other)
   - if running in parallel mode, this event will be emitted for each process

10. event.env-updated(app, env)
11. event.env-get-updated(app, env)
12. event.env-check-consistency(app, env)

# The updated-docs list can be builder dependent, but generally includes all new/changed␣
↪documents,
# plus any output from `env-get-updated`, and then all "parent" documents in the ToC tree
# For builders that output a single page, they are first joined into a single doctree␣
↪before post-transforms
# or the doctree-resolved event is emitted
for docname in updated-docs:
   13. apply post-transforms (by priority): docutils.document -> docutils.document
   14. event.doctree-resolved(app, doctree, docname)
       - In the event that any reference nodes fail to resolve, the following may emit:
       - event.missing-reference(env, node, contnode)
       - event.warn-missing-reference(domain, node)

15. Generate output files
16. event.build-finished(app, exception)
```

Here is a more detailed list of these events.

builder-inited(*app*)
Emitted when the builder object has been created. It is available as `app.builder`.

config-inited(*app*, *config*)
Emitted when the config object has been initialized.

New in version 1.8.

env-get-outdated(*app*, *env*, *added*, *changed*, *removed*)
Emitted when the environment determines which source files have changed and should be re-read. *added*, *changed* and *removed* are sets of docnames that the environment has determined. You can return a list of docnames to re-read in addition to these.

New in version 1.1.

env-purge-doc(*app*, *env*, *docname*)
Emitted when all traces of a source file should be cleaned from the environment, that is, if the source file is removed or before it is freshly read. This is for extensions that keep their own caches in attributes of the environment.

For example, there is a cache of all modules on the environment. When a source file has been changed, the cache's entries for the file are cleared, since the module declarations could have been removed from the file.

New in version 0.5.

env-before-read-docs(*app*, *env*, *docnames*)
Emitted after the environment has determined the list of all added and changed files and just before it reads them. It allows extension authors to reorder the list of docnames (*inplace*) before processing, or add more docnames that Sphinx did not consider changed (but never add any docnames that are not in `env.found_docs`).

You can also remove document names; do this with caution since it will make Sphinx treat changed files as unchanged.

New in version 1.3.

`source-read`(*app*, *docname*, *source*)
Emitted when a source file has been read. The *source* argument is a list whose single element is the contents of the source file. You can process the contents and replace this item to implement source-level transformations.

For example, if you want to use `$` signs to delimit inline math, like in LaTeX, you can use a regular expression to replace `$...$` by ``:math:`...` ``.

New in version 0.5.

`object-description-transform`(*app*, *domain*, *objtype*, *contentnode*)
Emitted when an object description directive has run. The *domain* and *objtype* arguments are strings indicating object description of the object. And *contentnode* is a content for the object. It can be modified in-place.

New in version 2.4.

`doctree-read`(*app*, *doctree*)
Emitted when a doctree has been parsed and read by the environment, and is about to be pickled. The *doctree* can be modified in-place.

`missing-reference`(*app*, *env*, *node*, *contnode*)
Emitted when a cross-reference to an object cannot be resolved. If the event handler can resolve the reference, it should return a new docutils node to be inserted in the document tree in place of the node *node*. Usually this node is a `reference` node containing *contnode* as a child. If the handler can not resolve the cross-reference, it can either return `None` to let other handlers try, or raise `NoUri` to prevent other handlers in trying and suppress a warning about this cross-reference being unresolved.

Parameters

- **env** – The build environment (`app.builder.env`).
- **node** – The `pending_xref` node to be resolved. Its attributes `reftype`, `reftarget`, `modname` and `classname` attributes determine the type and target of the reference.
- **contnode** – The node that carries the text and formatting inside the future reference and should be a child of the returned reference node.

New in version 0.5.

`warn-missing-reference`(*app*, *domain*, *node*)
Emitted when a cross-reference to an object cannot be resolved even after *`missing-reference`*. If the event handler can emit warnings for the missing reference, it should return `True`.

New in version 3.4.

`doctree-resolved`(*app*, *doctree*, *docname*)
Emitted when a doctree has been “resolved” by the environment, that is, all references have been resolved and TOCs have been inserted. The *doctree* can be modified in place.

Here is the place to replace custom nodes that don’t have visitor methods in the writers, so that they don’t cause errors when the writers encounter them.

`env-merge-info`(*app*, *env*, *docnames*, *other*)
This event is only emitted when parallel reading of documents is enabled. It is emitted once for every subprocess that has read some documents.

You must handle this event in an extension that stores data in the environment in a custom location. Otherwise the environment in the main process will not be aware of the information stored in the subprocess.

other is the environment object from the subprocess, *env* is the environment from the main process. *docnames* is a set of document names that have been read in the subprocess.

New in version 1.3.

`env-updated`(*app*, *env*)
Emitted when the `update()` method of the build environment has completed, that is, the environment and all doctrees are now up-to-date.

You can return an iterable of docnames from the handler. These documents will then be considered updated, and will be (re-)written during the writing phase.

New in version 0.5.

Changed in version 1.3: The handlers' return value is now used.

`env-check-consistency`(*app*, *env*)
Emitted when Consistency checks phase. You can check consistency of metadata for whole of documents.

New in version 1.6: As a **experimental** event

`html-collect-pages`(*app*)
Emitted when the HTML builder is starting to write non-document pages. You can add pages to write by returning an iterable from this event consisting of `(pagename, context, templatename)`.

New in version 1.0.

`html-page-context`(*app*, *pagename*, *templatename*, *context*, *doctree*)
Emitted when the HTML builder has created a context dictionary to render a template with – this can be used to add custom elements to the context.

The *pagename* argument is the canonical name of the page being rendered, that is, without `.html` suffix and using slashes as path separators. The *templatename* is the name of the template to render, this will be `'page.html'` for all pages from reST documents.

The *context* argument is a dictionary of values that are given to the template engine to render the page and can be modified to include custom values. Keys must be strings.

The *doctree* argument will be a doctree when the page is created from a reST documents; it will be `None` when the page is created from an HTML template alone.

You can return a string from the handler, it will then replace `'page.html'` as the HTML template for this page.

Note: You can install JS/CSS files for the specific page via *`Sphinx.add_js_file()`* and *`Sphinx.add_css_file()`* since v3.5.0.

New in version 0.4.

Changed in version 1.3: The return value can now specify a template name.

`build-finished`(*app*, *exception*)
Emitted when a build has finished, before Sphinx exits, usually used for cleanup. This event is emitted even when the build process raised an exception, given as the *exception* argument. The exception is reraised in the application after the event handlers have run. If the build process raised no exception, *exception* will be `None`. This allows to customize cleanup actions depending on the exception status.

New in version 0.5.

Checking the Sphinx version

Use this to adapt your extension to API changes in Sphinx.

`sphinx.version_info = (4, 0, 0, 'beta', 0)`

Version info for better programmatic use.

A tuple of five elements; for Sphinx version 1.2.1 beta 3 this would be `(1, 2, 1, 'beta', 3)`. The fourth element can be one of: `alpha`, `beta`, `rc`, `final`. `final` always has 0 as the last element.

New in version 1.2: Before version 1.2, check the string `sphinx.__version__`.

The Config object

class `sphinx.config.Config`(*config: Dict[str*[407]*, Any] = {}*, *overrides: Dict[str*[408]*, Any] = {}*)

Configuration file abstraction.

The config object makes the values of all config values available as attributes.

It is exposed via the `sphinx.application.Application.config` and `sphinx.environment.Environment.config` attributes. For example, to get the value of *language*, use either `app.config.language` or `env.config.language`.

The template bridge

class `sphinx.application.TemplateBridge`

This class defines the interface for a "template bridge", that is, a class that renders templates given a template name and a context.

init(*builder: Builder*, *theme: sphinx.theming.Theme = None*, *dirs: List[str*[409]*] = None*) → None[410]

Called by the builder to initialize the template system.

builder is the builder object; you'll probably want to look at the value of `builder.config.templates_path`.

theme is a `sphinx.theming.Theme` object or None; in the latter case, *dirs* can be list of fixed directories to look for templates.

newest_template_mtime() → float[411]

Called by the builder to determine if output files are outdated because of template changes. Return the mtime of the newest template file that was changed. The default implementation returns `0`.

render(*template: str*[412], *context: Dict*) → None[413]

Called by the builder to render a template given as a filename with a specified context (a Python dictionary).

render_string(*template: str*[414], *context: Dict*) → str[415]

Called by the builder to render a template given as a string with a specified context (a Python dictionary).

[407] https://docs.python.org/3/library/stdtypes.html#str
[408] https://docs.python.org/3/library/stdtypes.html#str
[409] https://docs.python.org/3/library/stdtypes.html#str
[410] https://docs.python.org/3/library/constants.html#None
[411] https://docs.python.org/3/library/functions.html#float
[412] https://docs.python.org/3/library/stdtypes.html#str
[413] https://docs.python.org/3/library/constants.html#None
[414] https://docs.python.org/3/library/stdtypes.html#str
[415] https://docs.python.org/3/library/stdtypes.html#str

Exceptions

exception sphinx.errors.**SphinxError**

Base class for Sphinx errors.

This is the base class for "nice" exceptions. When such an exception is raised, Sphinx will abort the build and present the exception category and message to the user.

Extensions are encouraged to derive from this exception for their custom errors.

Exceptions *not* derived from `SphinxError` are treated as unexpected and shown to the user with a part of the traceback (and the full traceback saved in a temporary file).

category

Description of the exception "category", used in converting the exception to a string ("category: message"). Should be set accordingly in subclasses.

exception sphinx.errors.**ConfigError**

Configuration error.

exception sphinx.errors.**ExtensionError**(*message: str*[416], *orig_exc: Optional[Exception*[417]*] = None*, *modname: Optional[str*[418]*] = None*)

Extension error.

exception sphinx.errors.**ThemeError**

Theme error.

exception sphinx.errors.**VersionRequirementError**

Incompatible Sphinx version error.

Project API

class sphinx.project.**Project**(*srcdir*, *source_suffix*)

A project is source code set of Sphinx document.

discover(*exclude_paths=[]*)

Find all document files in the source directory and put them in `docnames`.

doc2path(*docname*, *basedir=True*)

Return the filename for the document name.

If *basedir* is True, return as an absolute path. Else, return as a relative path to the source directory.

path2doc(*filename*)

Return the docname for the filename if the file is document.

filename should be absolute or relative to the source directory.

restore(*other*)

Take over a result of last build.

docnames

The name of documents belongs to this project.

source_suffix

source_suffix. Same as `source_suffix`.

srcdir

Source directory.

[416] https://docs.python.org/3/library/stdtypes.html#str
[417] https://docs.python.org/3/library/exceptions.html#Exception
[418] https://docs.python.org/3/library/stdtypes.html#str

Build environment API

`class sphinx.environment.BuildEnvironment`

Attributes

`app`
Reference to the *Sphinx* (application) object.

`config`
Reference to the *Config* object.

`project`
Target project. See *Project*.

`srcdir`
Source directory.

`doctreedir`
Directory for storing pickled doctrees.

`events`
An *EventManager* object.

`found_docs`
A set of all existing docnames.

`metadata`
Dictionary mapping docnames to "metadata" (see *File-wide metadata*).

`titles`
Dictionary mapping docnames to the docutils node for their main title.

`docname`
Returns the docname of the document currently being parsed.

Utility methods

`doc2path`(*docname: str*[419], *base: bool*[420] *= True*) → str[421]
Return the filename for the document name.

If *base* is True, return absolute path under self.srcdir. If *base* is False, return relative path to self.srcdir.

`relfn2path`(*filename: str*[422], *docname: Optional[str*[423]*] = None*) → Tuple[str[424], str[425]]
Return paths to a file referenced from a document, relative to documentation root and absolute.

In the input "filename", absolute filenames are taken as relative to the source dir, while relative filenames are relative to the dir of the containing document.

`note_dependency`(*filename: str*[426]) → None[427]
Add *filename* as a dependency of the current document.

This means that the document will be rebuilt if this file changes.

filename should be absolute or relative to the source directory.

`new_serialno`(*category: str*[428] *= ''*) → int[429]
Return a serial number, e.g. for index entry targets.

The number is guaranteed to be unique in the current document.

`note_reread`() → None[430]
Add the current document to the list of documents that will automatically be re-read at the next build.

Builder API

Todo: Expand this.

`class sphinx.builders.Builder`

This is the base class for all builders.

These attributes should be set on builder classes:

`name = ''`

The builder's name, for the -b command line option.

`format = ''`

The builder's output format, or '' if no document output is produced.

`epilog = ''`

The message emitted upon successful build completion. This can be a printf-style template string with the following keys: `outdir`, `project`

`supported_image_types = []`

The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

`supported_remote_images = False`

The builder supports remote images or not.

`supported_data_uri_images = False`

The builder supports data URIs or not.

`default_translator_class = None`

default translator class for the builder. This can be overridden by `app.set_translator()`.

These methods are predefined and will be called from the application:

`get_relative_uri`(*from_: str*[431], *to: str*[432], *typ: Optional[str*[433]*] = None*) → str[434]

Return a relative URI between two source filenames.

May raise environment.NoUri if there's no way to return a sensible URI.

`build_all`() → None[435]

Build all source files.

`build_specific`(*filenames: List[str*[436]*]*) → None[437]

Only rebuild as much as needed for changes in the *filenames*.

`build_update`() → None[438]

Only rebuild what was changed or added since last build.

`build`(*docnames: Iterable[str*[439]*]*, *summary: Optional[str*[440]*] = None*, *method: str*[441] *= 'update'*) → None[442]

Main build method.

[419] https://docs.python.org/3/library/stdtypes.html#str
[420] https://docs.python.org/3/library/functions.html#bool
[421] https://docs.python.org/3/library/stdtypes.html#str
[422] https://docs.python.org/3/library/stdtypes.html#str
[423] https://docs.python.org/3/library/stdtypes.html#str
[424] https://docs.python.org/3/library/stdtypes.html#str
[425] https://docs.python.org/3/library/stdtypes.html#str
[426] https://docs.python.org/3/library/stdtypes.html#str
[427] https://docs.python.org/3/library/constants.html#None
[428] https://docs.python.org/3/library/stdtypes.html#str
[429] https://docs.python.org/3/library/functions.html#int
[430] https://docs.python.org/3/library/constants.html#None

First updates the environment, and then calls `write()`.

These methods can be overridden in concrete builder classes:

`init`() → None[443]
: Load necessary templates and perform initialization. The default implementation does nothing.

`get_outdated_docs`() → Union[str[444], Iterable[str[445]]]
: Return an iterable of output files that are outdated, or a string describing what an update build will build.

  If the builder does not output individual files corresponding to source files, return a string here. If it does, return an iterable of those files that need to be written.

`get_target_uri`(*docname: str*[446], *typ: Optional[str*[447]*] = None*) → str[448]
: Return the target URI for a document name.

  typ can be used to qualify the link characteristic for individual builders.

`prepare_writing`(*docnames: Set[str*[449]*]*) → None[450]
: A place where you can add logic before *`write_doc()`* is run

`write_doc`(*docname: str*[451], *doctree: docutils.nodes.document*) → None[452]
: Where you actually write something to the filesystem.

`finish`() → None[453]
: Finish the building process.

  The default implementation does nothing.

Attributes

`events`
: An *`EventManager`* object.

[431] https://docs.python.org/3/library/stdtypes.html#str
[432] https://docs.python.org/3/library/stdtypes.html#str
[433] https://docs.python.org/3/library/stdtypes.html#str
[434] https://docs.python.org/3/library/stdtypes.html#str
[435] https://docs.python.org/3/library/constants.html#None
[436] https://docs.python.org/3/library/stdtypes.html#str
[437] https://docs.python.org/3/library/constants.html#None
[438] https://docs.python.org/3/library/constants.html#None
[439] https://docs.python.org/3/library/stdtypes.html#str
[440] https://docs.python.org/3/library/stdtypes.html#str
[441] https://docs.python.org/3/library/stdtypes.html#str
[442] https://docs.python.org/3/library/constants.html#None
[443] https://docs.python.org/3/library/constants.html#None
[444] https://docs.python.org/3/library/stdtypes.html#str
[445] https://docs.python.org/3/library/stdtypes.html#str
[446] https://docs.python.org/3/library/stdtypes.html#str
[447] https://docs.python.org/3/library/stdtypes.html#str
[448] https://docs.python.org/3/library/stdtypes.html#str
[449] https://docs.python.org/3/library/stdtypes.html#str
[450] https://docs.python.org/3/library/constants.html#None
[451] https://docs.python.org/3/library/stdtypes.html#str
[452] https://docs.python.org/3/library/constants.html#None
[453] https://docs.python.org/3/library/constants.html#None

Environment Collector API

class sphinx.environment.collectors.**EnvironmentCollector**

An EnvironmentCollector is a specific data collector from each document.

It gathers data and stores *BuildEnvironment* as a database. Examples of specific data would be images, download files, section titles, metadatas, index entries and toctrees, etc.

clear_doc(*app: Sphinx*, *env:* sphinx.environment.BuildEnvironment, *docname: str*[454]) → None[455]

Remove specified data of a document.

This method is called on the removal of the document.

get_outdated_docs(*app: Sphinx*, *env:* sphinx.environment.BuildEnvironment, *added: Set[str*[456]*]*, *changed: Set[str*[457]*]*, *removed: Set[str*[458]*]*) → List[str[459]]

Return a list of docnames to re-read.

This methods is called before reading the documents.

get_updated_docs(*app: Sphinx*, *env:* sphinx.environment.BuildEnvironment) → List[str[460]]

Return a list of docnames to re-read.

This methods is called after reading the whole of documents (experimental).

merge_other(*app: Sphinx*, *env:* sphinx.environment.BuildEnvironment, *docnames: Set[str*[461]*]*, *other:* sphinx.environment.BuildEnvironment) → None[462]

Merge in specified data regarding docnames from a different *BuildEnvironment* object which coming from a subprocess in parallel builds.

process_doc(*app: Sphinx*, *doctree: docutils.nodes.document*) → None[463]

Process a document and gather specific data from it.

This method is called after the document is read.

Docutils markup API

This section describes the API for adding ReST markup elements (roles and directives).

Roles

Directives

Directives are handled by classes derived from `docutils.parsers.rst.Directive`. They have to be registered by an extension using *Sphinx.add_directive()* or *Sphinx.add_directive_to_domain()*.

class docutils.parsers.rst.**Directive**

The markup syntax of the new directive is determined by the follow five class attributes:

required_arguments = 0

Number of required directive arguments.

[454] https://docs.python.org/3/library/stdtypes.html#str
[455] https://docs.python.org/3/library/constants.html#None
[456] https://docs.python.org/3/library/stdtypes.html#str
[457] https://docs.python.org/3/library/stdtypes.html#str
[458] https://docs.python.org/3/library/stdtypes.html#str
[459] https://docs.python.org/3/library/stdtypes.html#str
[460] https://docs.python.org/3/library/stdtypes.html#str
[461] https://docs.python.org/3/library/stdtypes.html#str
[462] https://docs.python.org/3/library/constants.html#None
[463] https://docs.python.org/3/library/constants.html#None

`optional_arguments = 0`
: Number of optional arguments after the required arguments.

`final_argument_whitespace = False`
: May the final argument contain whitespace?

`option_spec = None`
: Mapping of option names to validator functions.

  Option validator functions take a single parameter, the option argument (or `None` if not given), and should validate it or convert it to the proper form. They raise `ValueError`[464] or `TypeError`[465] to indicate failure.

  There are several predefined and possibly useful validators in the `docutils.parsers.rst.directives` module.

`has_content = False`
: May the directive have content?

New directives must implement the *`run()`* method:

`run()`
: This method must process the directive arguments, options and content, and return a list of Docutils/Sphinx nodes that will be inserted into the document tree at the point where the directive was encountered.

Instance attributes that are always set on the directive are:

`name`
: The directive name (useful when registering the same directive class under multiple names).

`arguments`
: The arguments given to the directive, as a list.

`options`
: The options given to the directive, as a dictionary mapping option names to validated/converted values.

`content`
: The directive content, if given, as a `ViewList`.

`lineno`
: The absolute line number on which the directive appeared. This is not always a useful value; use `srcline` instead.

`content_offset`
: Internal offset of the directive content. Used when calling `nested_parse` (see below).

`block_text`
: The string containing the entire directive.

`state`
`state_machine`
: The state and state machine which controls the parsing. Used for `nested_parse`.

[464] https://docs.python.org/3/library/exceptions.html#ValueError
[465] https://docs.python.org/3/library/exceptions.html#TypeError

ViewLists

Docutils represents document source lines in a class `docutils.statemachine.ViewList`. This is a list with extended functionality – for one, slicing creates views of the original list, and also the list contains information about the source line numbers.

The *`Directive.content`* attribute is a ViewList. If you generate content to be parsed as ReST, you have to create a ViewList yourself. Important for content generation are the following points:

- The constructor takes a list of strings (lines) and a source (document) name.
- The `.append()` method takes a line and a source name as well.

Parsing directive content as ReST

Many directives will contain more markup that must be parsed. To do this, use one of the following APIs from the *`Directive.run()`* method:

- `self.state.nested_parse`
- `sphinx.util.nodes.nested_parse_with_titles()` – this allows titles in the parsed content.

Both APIs parse the content into a given node. They are used like this:

```
node = docutils.nodes.paragraph()
# either
nested_parse_with_titles(self.state, self.result, node)
# or
self.state.nested_parse(self.result, 0, node)
```

Note: `sphinx.util.docutils.switch_source_input()` allows to change a target file during nested_parse. It is useful to mixed contents. For example, `sphinx. ext.autodoc` uses it to parse docstrings:

```
from sphinx.util.docutils import switch_source_input

# Switch source_input between parsing content.
# Inside this context, all parsing errors and warnings are reported as
# happened in new source_input (in this case, ``self.result``).
with switch_source_input(self.state, self.result):
    node = docutils.nodes.paragraph()
    self.state.nested_parse(self.result, 0, node)
```

Deprecated since version 1.7: Until Sphinx-1.6, `sphinx.ext.autodoc.AutodocReporter` is used for this purpose. For now, it is replaced by `switch_source_input()`.

If you don't need the wrapping node, you can use any concrete node type and return `node.children` from the Directive.

See also:

Creating directives[466] HOWTO of the Docutils documentation

[466] http://docutils.sourceforge.net/docs/howto/rst-directives.html

Domain API

class sphinx.domains.**Domain**(*env: BuildEnvironment*)

A Domain is meant to be a group of "object" description directives for objects of a similar nature, and corresponding roles to create references to them. Examples would be Python modules, classes, functions etc., elements of a templating language, Sphinx roles and directives, etc.

Each domain has a separate storage for information about existing objects and how to reference them in *self.data*, which must be a dictionary. It also must implement several functions that expose the object information in a uniform way to parts of Sphinx that allow the user to reference or search for objects in a domain-agnostic way.

About *self.data*: since all object and cross-referencing information is stored on a BuildEnvironment instance, the *domain.data* object is also stored in the *env.domaindata* dict under the key *domain.name*. Before the build process starts, every active domain is instantiated and given the environment object; the *domaindata* dict must then either be nonexistent or a dictionary whose 'version' key is equal to the domain class' `data_version` attribute. Otherwise, *OSError* is raised and the pickled environment is discarded.

add_object_type(*name: str*[467], *objtype:* sphinx.domains.ObjType) → None[468]

Add an object type.

check_consistency() → None[469]

Do consistency checks (**experimental**).

clear_doc(*docname: str*[470]) → None[471]

Remove traces of a document in the domain-specific inventories.

directive(*name: str*[472]) → Callable

Return a directive adapter class that always gives the registered directive its full name ('domain:name') as `self.name`.

get_enumerable_node_type(*node: docutils.nodes.Node*) → str[473]

Get type of enumerable nodes (experimental).

get_full_qualified_name(*node: docutils.nodes.Element*) → str[474]

Return full qualified name for given node.

get_objects() → Iterable[Tuple[str[475], str[476], str[477], str[478], str[479], int[480]]]

Return an iterable of "object descriptions".

Object descriptions are tuples with six items:

name Fully qualified name.

dispname Name to display when searching/linking.

type Object type, a key in `self.object_types`.

docname The document where it is to be found.

anchor The anchor name for the object.

priority How "important" the object is (determines placement in search results). One of:

1 Default priority (placed before full-text matches).

0 Object is important (placed before default-priority objects).

2 Object is unimportant (placed after full-text matches).

-1 Object should not show up in search at all.

get_type_name(*type:* sphinx.domains.ObjType, *primary: bool*[481] *= False*) → str[482]

Return full name for given ObjType.

merge_domaindata(*docnames: List[str*[483]*]*, *otherdata: Dict*) → None[484]

Merge in data regarding *docnames* from a different domaindata inventory (coming from a subprocess in parallel builds).

process_doc(*env: BuildEnvironment*, *docname: str*[485], *document: docutils.nodes.document*) → None[486]
Process a document after it is read by the environment.

process_field_xref(*pnode: sphinx.addnodes.pending_xref*) → None[487]
Process a pending xref created in a doc field. For example, attach information about the current scope.

resolve_any_xref(*env: BuildEnvironment*, *fromdocname: str*[488], *builder: Builder*, *target: str*[489], *node: sphinx.addnodes.pending_xref*, *contnode: docutils.nodes.Element*) → List[Tuple[str[490], docutils.nodes.Element]]
Resolve the pending_xref *node* with the given *target*.

The reference comes from an "any" or similar role, which means that we don't know the type. Otherwise, the arguments are the same as for `resolve_xref()`.

The method must return a list (potentially empty) of tuples `('domain:role', newnode)`, where `'domain:role'` is the name of a role that could have created the same reference, e.g. `'py:func'`. `newnode` is what `resolve_xref()` would return.

New in version 1.3.

resolve_xref(*env: BuildEnvironment*, *fromdocname: str*[491], *builder: Builder*, *typ: str*[492], *target: str*[493], *node: sphinx.addnodes.pending_xref*, *contnode: docutils.nodes.Element*) → docutils.nodes.Element
Resolve the pending_xref *node* with the given *typ* and *target*.

This method should return a new node, to replace the xref node, containing the *contnode* which is the markup content of the cross-reference.

If no resolution can be found, None can be returned; the xref node will then given to the `missing-reference` event, and if that yields no resolution, replaced by *contnode*.

The method can also raise `sphinx.environment.NoUri` to suppress the `missing-reference` event being emitted.

role(*name: str*[494]) → Callable[[str[495], str[496], str[497], int[498], docutils.parsers.rst.states.Inliner, Dict[str[499], Any], List[str[500]]], Tuple[List[docutils.nodes.Node], List[docutils.nodes.system_message]]]
Return a role adapter function that always gives the registered role its full name ('domain:name') as the first argument.

setup() → None[501]
Set up domain object.

dangling_warnings = {}
role name -> a warning message if reference is missing

data = None
data value

data_version = 0
data version, bump this when the format of *self.data* changes

directives = {}
directive name -> directive class

enumerable_nodes = {}
node_class -> (enum_node_type, title_getter)

indices = []
a list of Index subclasses

initial_data = {}
data value for a fresh environment

label = ''
domain label: longer, more descriptive (used in messages)

name = ''
domain name: should be short, but unique

object_types = {}
type (usually directive) name -> ObjType instance

roles = {}
role name -> role callable

class sphinx.domains.**ObjType**(*lname: str*[502], **roles: Any*, ***attrs: Any*)
An ObjType is the description for a type of object that a domain can document. In the object_types attribute of Domain subclasses, object type names are mapped to instances of this class.

Constructor arguments:

- *lname*: localized name of the type (do not include domain name)
- *roles*: all the roles that can refer to an object of this type
- *attrs*: object attributes – currently only "searchprio" is known, which defines the object's priority in the full-text search index, see `Domain.get_objects()`.

class sphinx.domains.**Index**(*domain:* sphinx.domains.Domain)
An Index is the description for a domain-specific index. To add an index to a domain, subclass Index, overriding the three name attributes:

- *name* is an identifier used for generating file names. It is also used for a hyperlink target for the index. Therefore, users can refer the index page using `ref` role and a string which is combined domain name and `name` attribute (ex. `` :ref:`py-modindex` ``).

[467] https://docs.python.org/3/library/stdtypes.html#str
[468] https://docs.python.org/3/library/constants.html#None
[469] https://docs.python.org/3/library/constants.html#None
[470] https://docs.python.org/3/library/stdtypes.html#str
[471] https://docs.python.org/3/library/constants.html#None
[472] https://docs.python.org/3/library/stdtypes.html#str
[473] https://docs.python.org/3/library/stdtypes.html#str
[474] https://docs.python.org/3/library/stdtypes.html#str
[475] https://docs.python.org/3/library/stdtypes.html#str
[476] https://docs.python.org/3/library/stdtypes.html#str
[477] https://docs.python.org/3/library/stdtypes.html#str
[478] https://docs.python.org/3/library/stdtypes.html#str
[479] https://docs.python.org/3/library/stdtypes.html#str
[480] https://docs.python.org/3/library/functions.html#int
[481] https://docs.python.org/3/library/functions.html#bool
[482] https://docs.python.org/3/library/stdtypes.html#str
[483] https://docs.python.org/3/library/stdtypes.html#str
[484] https://docs.python.org/3/library/constants.html#None
[485] https://docs.python.org/3/library/stdtypes.html#str
[486] https://docs.python.org/3/library/constants.html#None
[487] https://docs.python.org/3/library/constants.html#None
[488] https://docs.python.org/3/library/stdtypes.html#str
[489] https://docs.python.org/3/library/stdtypes.html#str
[490] https://docs.python.org/3/library/stdtypes.html#str
[491] https://docs.python.org/3/library/stdtypes.html#str
[492] https://docs.python.org/3/library/stdtypes.html#str
[493] https://docs.python.org/3/library/stdtypes.html#str
[494] https://docs.python.org/3/library/stdtypes.html#str
[495] https://docs.python.org/3/library/stdtypes.html#str
[496] https://docs.python.org/3/library/stdtypes.html#str
[497] https://docs.python.org/3/library/stdtypes.html#str
[498] https://docs.python.org/3/library/functions.html#int
[499] https://docs.python.org/3/library/stdtypes.html#str
[500] https://docs.python.org/3/library/stdtypes.html#str
[501] https://docs.python.org/3/library/constants.html#None
[502] https://docs.python.org/3/library/stdtypes.html#str

- *localname* is the section title for the index.
- *shortname* is a short name for the index, for use in the relation bar in HTML output. Can be empty to disable entries in the relation bar.

and providing a `generate()` method. Then, add the index class to your domain's *indices* list. Extensions can add indices to existing domains using `add_index_to_domain()`.

Changed in version 3.0: Index pages can be referred by domain name and index name via `ref` role.

abstract generate(*docnames: Optional[Iterable[str[503]]] = None*) → Tuple[List[Tuple[str[504], List[sphinx.domains.IndexEntry]]], bool[505]]

Get entries for the index.

If `docnames` is given, restrict to entries referring to these docnames.

The return value is a tuple of `(content, collapse)`:

collapse A boolean that determines if sub-entries should start collapsed (for output formats that support collapsing sub-entries).

content: A sequence of `(letter, entries)` tuples, where `letter` is the "heading" for the given `entries`, usually the starting letter, and `entries` is a sequence of single entries. Each entry is a sequence `[name, subtype, docname, anchor, extra, qualifier, descr]`. The items in this sequence have the following meaning:

name The name of the index entry to be displayed.

subtype The sub-entry related type. One of:

0 A normal entry.

1 An entry with sub-entries.

2 A sub-entry.

docname *docname* where the entry is located.

anchor Anchor for the entry within `docname`

extra Extra info for the entry.

qualifier Qualifier for the description.

descr Description for the entry.

Qualifier and description are not rendered for some output formats such as LaTeX.

Python Domain

class sphinx.domains.python.**PythonDomain**(*env: BuildEnvironment*)

Python language domain.

objects

modules

note_object(*name: str[506], objtype: str[507], node_id: str[508], canonical: bool[509] = False, location: Optional[Any] = None*) → None[510]

Note a python object for cross reference.

New in version 2.1.

[503] https://docs.python.org/3/library/stdtypes.html#str
[504] https://docs.python.org/3/library/stdtypes.html#str
[505] https://docs.python.org/3/library/functions.html#bool

note_module(*name: str*[511], *node_id: str*[512], *synopsis: str*[513], *platform: str*[514], *deprecated: bool*[515]) → None[516]

Note a python module for cross reference.

New in version 2.1.

Parser API

The docutils documentation describes[517] parsers as follows:

> The Parser analyzes the input document and creates a node tree representation.

In Sphinx, the parser modules works as same as docutils. The parsers are registered to Sphinx by extensions using Application APIs; `Sphinx.add_source_suffix()` and `Sphinx.add_source_parser()`.

The *source suffix* is a mapping from file suffix to file type. For example, `.rst` file is mapped to `'restructuredtext'` type. Sphinx uses the file type to looking for parsers from registered list. On searching, Sphinx refers to the `Parser.supported` attribute and picks up a parser which contains the file type in the attribute.

The users can override the source suffix mappings using `source_suffix` like following:

```
# a mapping from file suffix to file types
source_suffix = {
    '.rst': 'restructuredtext',
    '.md': 'markdown',
}
```

You should indicate file types your parser supports. This will allow users to configure their settings appropriately.

class sphinx.parsers.**Parser**

A base class of source parsers. The additional parsers should inherit this class instead of `docutils.parsers.Parser`. Compared with `docutils.parsers.Parser`, this class improves accessibility to Sphinx APIs.

The subclasses can access following objects and functions:

self.app The application object (`sphinx.application.Sphinx`)

self.config The config object (`sphinx.config.Config`)

self.env The environment object (`sphinx.environment.BuildEnvironment`)

self.warn() Emit a warning. (Same as `sphinx.application.Sphinx.warn()`)

self.info() Emit a informational message. (Same as `sphinx.application.Sphinx.info()`)

Deprecated since version 1.6: `warn()` and `info()` is deprecated. Use `sphinx.util.logging` instead.

Deprecated since version 3.0: parser.app is deprecated.

[506] https://docs.python.org/3/library/stdtypes.html#str
[507] https://docs.python.org/3/library/stdtypes.html#str
[508] https://docs.python.org/3/library/stdtypes.html#str
[509] https://docs.python.org/3/library/functions.html#bool
[510] https://docs.python.org/3/library/constants.html#None
[511] https://docs.python.org/3/library/stdtypes.html#str
[512] https://docs.python.org/3/library/stdtypes.html#str
[513] https://docs.python.org/3/library/stdtypes.html#str
[514] https://docs.python.org/3/library/stdtypes.html#str
[515] https://docs.python.org/3/library/functions.html#bool
[516] https://docs.python.org/3/library/constants.html#None
[517] http://docutils.sourceforge.net/docs/dev/hacking.html#parsing-the-document

Doctree node classes added by Sphinx

Nodes for domain-specific object descriptions

class sphinx.addnodes.**desc**(*rawsource=''*, **children*, ***attributes*)
Node for object descriptions.

This node is similar to a "definition list" with one definition. It contains one or more `desc_signature` and a `desc_content`.

class sphinx.addnodes.**desc_signature**(*rawsource=''*, *text=''*, **children*, ***attributes*)
Node for object signatures.

The "term" part of the custom Sphinx definition list.

As default the signature is a single line signature, but set `is_multiline = True` to describe a multi-line signature. In that case all child nodes must be `desc_signature_line` nodes.

class sphinx.addnodes.**desc_signature_line**(*rawsource=''*, *text=''*, **children*, ***attributes*)
Node for a line in a multi-line object signatures.

It should only be used in a `desc_signature` with `is_multiline` set. Set `add_permalink = True` for the line that should get the permalink.

class sphinx.addnodes.**desc_addname**(*rawsource=''*, *text=''*, **children*, ***attributes*)
Node for additional name parts (module name, class name).

class sphinx.addnodes.**desc_type**(*rawsource=''*, *text=''*, **children*, ***attributes*)
Node for return types or object type names.

class sphinx.addnodes.**desc_returns**(*rawsource=''*, *text=''*, **children*, ***attributes*)
Node for a "returns" annotation (a la -> in Python).

class sphinx.addnodes.**desc_name**(*rawsource=''*, *text=''*, **children*, ***attributes*)
Node for the main object name.

class sphinx.addnodes.**desc_parameterlist**(*rawsource=''*, *text=''*, **children*, ***attributes*)
Node for a general parameter list.

class sphinx.addnodes.**desc_parameter**(*rawsource=''*, *text=''*, **children*, ***attributes*)
Node for a single parameter.

class sphinx.addnodes.**desc_optional**(*rawsource=''*, *text=''*, **children*, ***attributes*)
Node for marking optional parts of the parameter list.

class sphinx.addnodes.**desc_annotation**(*rawsource=''*, *text=''*, **children*, ***attributes*)
Node for signature annotations (not Python 3-style annotations).

class sphinx.addnodes.**desc_content**(*rawsource=''*, **children*, ***attributes*)
Node for object description content.

This is the "definition" part of the custom Sphinx definition list.

New admonition-like constructs

class sphinx.addnodes.**versionmodified**(*rawsource=''*, *text=''*, **children*, ***attributes*)

Node for version change entries.

Currently used for "versionadded", "versionchanged" and "deprecated" directives.

class sphinx.addnodes.**seealso**(*rawsource=''*, **children*, ***attributes*)

Custom "see also" admonition.

Other paragraph-level nodes

class sphinx.addnodes.**compact_paragraph**(*rawsource=''*, *text=''*, **children*, ***attributes*)

Node for a compact paragraph (which never makes a <p> node).

New inline nodes

class sphinx.addnodes.**index**(*rawsource=''*, *text=''*, **children*, ***attributes*)

Node for index entries.

This node is created by the `index` directive and has one attribute, `entries`. Its value is a list of 5-tuples of `(entrytype, entryname, target, ignored, key)`.

entrytype is one of "single", "pair", "double", "triple".

key is categorization characters (usually a single character) for general index page. For the details of this, please see also: *glossary* and issue #2320.

class sphinx.addnodes.**pending_xref**(*rawsource=''*, **children*, ***attributes*)

Node for cross-references that cannot be resolved without complete information about all documents.

These nodes are resolved before writing output, in BuildEnvironment.resolve_references.

class sphinx.addnodes.**literal_emphasis**(*rawsource=''*, *text=''*, **children*, ***attributes*)

Node that behaves like *emphasis*, but further text processors are not applied (e.g. smartypants for HTML output).

class sphinx.addnodes.**download_reference**(*rawsource=''*, *text=''*, **children*, ***attributes*)

Node for download references, similar to pending_xref.

Special nodes

class sphinx.addnodes.**only**(*rawsource=''*, **children*, ***attributes*)

Node for "only" directives (conditional inclusion based on tags).

class sphinx.addnodes.**meta**(*rawsource=''*, **children*, ***attributes*)

Node for meta directive – same as docutils' standard meta node, but pickleable.

class sphinx.addnodes.**highlightlang**(*rawsource=''*, **children*, ***attributes*)

Inserted to set the highlight language and line number options for subsequent code blocks.

You should not need to generate the nodes below in extensions.

class sphinx.addnodes.**glossary**(*rawsource=''*, **children*, ***attributes*)

Node to insert a glossary.

class sphinx.addnodes.**toctree**(*rawsource=''*, **children*, ***attributes*)

Node for inserting a "TOC tree".

class sphinx.addnodes.**start_of_file**(*rawsource=''*, **children*, ***attributes*)

Node to mark start of a new file, used in the LaTeX builder only.

class `sphinx.addnodes.`**`productionlist`**(*rawsource=''*, **children*, ***attributes*)
Node for grammar production lists.

Contains `production` nodes.

class `sphinx.addnodes.`**`production`**(*rawsource=''*, *text=''*, **children*, ***attributes*)
Node for a single grammar production rule.

Logging API

`sphinx.util.logging.`**`getLogger`**(*name*)
Get logger wrapped by *`sphinx.util.logging.SphinxLoggerAdapter`*.

Sphinx logger always uses `sphinx.*` namespace to be independent from settings of root logger. It ensures logging is consistent even if a third-party extension or imported application resets logger settings.

Example usage:

```
>>> from sphinx.util import logging
>>> logger = logging.getLogger(__name__)
>>> logger.info('Hello, this is an extension!')
Hello, this is an extension!
```

class `sphinx.util.logging.`**`SphinxLoggerAdapter`**(*logging.LoggerAdapter*)
LoggerAdapter allowing `type` and `subtype` keywords.

`error`(*msg*, **args*, ***kwargs*)

`critical`(*msg*, **args*, ***kwargs*)

`warning`(*msg*, **args*, ***kwargs*)
Logs a message on this logger with the specified level. Basically, the arguments are as with python's logging module.

In addition, Sphinx logger supports following keyword arguments:

type, *subtype* Categories of warning logs. It is used to suppress warnings by *`suppress_warnings`* setting.

location Where the warning happened. It is used to include the path and line number in each log. It allows docname, tuple of docname and line number and nodes:

```
logger = sphinx.util.logging.getLogger(__name__)
logger.warning('Warning happened!', location='index')
logger.warning('Warning happened!', location=('chapter1/index', 10))
logger.warning('Warning happened!', location=some_node)
```

color The color of logs. By default, error level logs are colored as `"darkred"`, critical level ones is not colored, and warning level ones are colored as `"red"`.

`log`(*level*, *msg*, **args*, ***kwargs*)

`info`(*msg*, **args*, ***kwargs*)

`verbose`(*msg*, **args*, ***kwargs*)

`debug`(*msg*, **args*, ***kwargs*)
Logs a message to this logger with the specified level. Basically, the arguments are as with python's logging module.

In addition, Sphinx logger supports following keyword arguments:

nonl If true, the logger does not fold lines at the end of the log message. The default is `False`.

location Where the message emitted. For more detail, see *SphinxLoggerAdapter.warning()*.

color The color of logs. By default, info and verbose level logs are not colored, and debug level ones are colored as "darkgray".

sphinx.util.logging.**pending_logging**()

Contextmanager to pend logging all logs temporary.

For example:

```
>>> with pending_logging():
>>>     logger.warning('Warning message!')  # not flushed yet
>>>     some_long_process()
>>>
Warning message!  # the warning is flushed here
```

sphinx.util.logging.**pending_warnings**()

Contextmanager to pend logging warnings temporary.

Similar to *pending_logging()*.

sphinx.util.logging.**prefixed_warnings**()

Prepend prefix to all records for a while.

For example:

```
>>> with prefixed_warnings("prefix:"):
>>>     logger.warning('Warning message!')  # => prefix: Warning message!
```

New in version 2.0.

i18n API

sphinx.locale.**init**(*locale_dirs: List[Optional[str*[518]*]], language: Optional[str*[519]*], catalog: str*[520] *= 'sphinx', namespace: str*[521] *= 'general'*) → Tuple[gettext.NullTranslations[522], bool[523]]

Look for message catalogs in *locale_dirs* and *ensure* that there is at least a NullTranslations catalog set in *translators*. If called multiple times or if several .mo files are found, their contents are merged together (thus making init reentrant).

sphinx.locale.**init_console**(*locale_dir: str*[524], *catalog: str*[525]) → Tuple[gettext.NullTranslations[526], bool[527]]

Initialize locale for console.

New in version 1.8.

sphinx.locale.**get_translation**(*catalog: str*[528], *namespace: str*[529] *= 'general'*) → Callable

Get a translation function based on the *catalog* and *namespace*.

The extension can use this API to translate the messages on the extension:

[518] https://docs.python.org/3/library/stdtypes.html#str
[519] https://docs.python.org/3/library/stdtypes.html#str
[520] https://docs.python.org/3/library/stdtypes.html#str
[521] https://docs.python.org/3/library/stdtypes.html#str
[522] https://docs.python.org/3/library/gettext.html#gettext.NullTranslations
[523] https://docs.python.org/3/library/functions.html#bool
[524] https://docs.python.org/3/library/stdtypes.html#str
[525] https://docs.python.org/3/library/stdtypes.html#str
[526] https://docs.python.org/3/library/gettext.html#gettext.NullTranslations
[527] https://docs.python.org/3/library/functions.html#bool

```
import os
from sphinx.locale import get_translation

MESSAGE_CATALOG_NAME = 'myextension'  # name of *.pot, *.po and *.mo files
_ = get_translation(MESSAGE_CATALOG_NAME)
text = _('Hello Sphinx!')


def setup(app):
    package_dir = path.abspath(path.dirname(__file__))
    locale_dir = os.path.join(package_dir, 'locales')
    app.add_message_catalog(MESSAGE_CATALOG_NAME, locale_dir)
```

With this code, sphinx searches a message catalog from `${package_dir}/locales/${language}/LC_MESSAGES/myextension.mo`. The *`language`* is used for the searching.

New in version 1.8.

`sphinx.locale._`(*message: str*[530], **args: Any*) → str[531]

Translation function for messages on documentation (menu, labels, themes and so on). This function follows *`language`* setting.

`sphinx.locale.__`(*message: str*[532], **args: Any*) → str[533]

Translation function for console messages This function follows locale setting (*LC_ALL*, *LC_MESSAGES* and so on).

Extension internationalization (*i18n*) and localization (*l10n*) using i18n API

New in version 1.8.

An extension may naturally come with message translations. This is briefly summarized in *`sphinx.locale.get_translation()`* help.

In practice, you have to:

1. Choose a name for your message catalog, which must be unique. Usually the name of your extension is used for the name of message catalog.
2. Mark in your extension sources all messages as translatable, via *`sphinx.locale.get_translation()`* function, usually renamed `_()`, e.g.:

Listing 1: src/__init__.py

```
from sphinx.locale import get_translation

MESSAGE_CATALOG_NAME = 'myextension'
_ = get_translation(MESSAGE_CATALOG_NAME)

translated_text = _('Hello Sphinx!')
```

3. Set up your extension to be aware of its dedicated translations:

[528] https://docs.python.org/3/library/stdtypes.html#str
[529] https://docs.python.org/3/library/stdtypes.html#str
[530] https://docs.python.org/3/library/stdtypes.html#str
[531] https://docs.python.org/3/library/stdtypes.html#str
[532] https://docs.python.org/3/library/stdtypes.html#str
[533] https://docs.python.org/3/library/stdtypes.html#str

Listing 2: src/__init__.py

```
def setup(app):
    package_dir = path.abspath(path.dirname(__file__))
    locale_dir = os.path.join(package_dir, 'locales')
    app.add_message_catalog(MESSAGE_CATALOG_NAME, locale_dir)
```

4. Generate message catalog template `*.pot` file, usually in `locale/` source directory, for example via Babel[534]:

```
$ pybabel extract --output=src/locale/myextension.pot src/
```

5. Create message catalogs (`*.po`) for each language which your extension will provide localization, for example via Babel[535]:

```
$ pybabel init --input-file=src/locale/myextension.pot --domain=myextension --
↪output-dir=src/locale --locale=fr_FR
```

6. Translate message catalogs for each language manually

7. Compile message catalogs into `*.mo` files, for example via Babel[536]:

```
$ pybabel compile --directory=src/locale --domain=myextension
```

8. Ensure that message catalog files are distributed when your package will be installed, by adding equivalent line in your extension `MANIFEST.in`:

534 http://babel.pocoo.org/
535 http://babel.pocoo.org/
536 http://babel.pocoo.org/

Listing 3: MANIFEST.in

```
recursive-include src *.pot *.po *.mo
```

When the messages on your extension has been changed, you need to also update message catalog template and message catalogs, for example via Babel[537]:

```
$ pybabel extract --output=src/locale/myextension.pot src/
$ pybabel update --input-file=src/locale/myextension.pot --domain=myextension --output-
↪dir=src/locale
```

Utilities

Sphinx provides utility classes and functions to develop extensions.

Base classes for components

These base classes are useful to allow your extensions to obtain Sphinx components (e.g. *Config*, *BuildEnvironment* and so on) easily.

Note: The subclasses of them might not work with bare docutils because they are strongly coupled with Sphinx.

`class sphinx.transforms.`**`SphinxTransform`**(*document*, *startnode=None*)

A base class of Transforms.

Compared with `docutils.transforms.Transform`, this class improves accessibility to Sphinx APIs.

`property app`

Reference to the *Sphinx* object.

`property config`

Reference to the *Config* object.

`property env`

Reference to the *BuildEnvironment* object.

`class sphinx.transforms.post_transforms.`**`SphinxPostTransform`**(*document*, *startnode=None*)

A base class of post-transforms.

Post transforms are invoked to modify the document to restructure it for outputting. They do resolving references, convert images, special transformation for each output formats and so on. This class helps to implement these post transforms.

`apply`(***kwargs: Any*) → None[538]

Override to apply the transform to the document tree.

`is_supported`() → bool[539]

Check this transform working for current builder.

`run`(***kwargs: Any*) → None[540]

main method of post transforms.

Subclasses should override this method instead of `apply()`.

[537] http://babel.pocoo.org/

class `sphinx.util.docutils.`**`SphinxDirective`**(*name*, *arguments*, *options*, *content*, *lineno*, *content_offset*, *block_text*, *state*, *state_machine*)

A base class for Sphinx directives.

This class provides helper methods for Sphinx directives.

Note: The subclasses of this class might not work with docutils. This class is strongly coupled with Sphinx.

`get_source_info``()` → Tuple[str[541], int[542]]

Get source and line number.

`set_source_info`(*node: docutils.nodes.Node*) → None[543]

Set source and line number to the node.

`property config`

Reference to the *`Config`* object.

`property env`

Reference to the *`BuildEnvironment`* object.

class `sphinx.util.docutils.`**`SphinxRole`**

A base class for Sphinx roles.

This class provides helper methods for Sphinx roles.

Note: The subclasses of this class might not work with docutils. This class is strongly coupled with Sphinx.

`property config`

Reference to the *`Config`* object.

`content: List[str`[544]**`]`**

A list of strings, the directive content for customization

`property env`

Reference to the *`BuildEnvironment`* object.

`inliner: docutils.parsers.rst.states.Inliner`

The `docutils.parsers.rst.states.Inliner` object.

`lineno: int`[545]

The line number where the interpreted text begins.

`name: str`[546]

The role name actually used in the document.

`options: Dict`

A dictionary of directive options for customization

`rawtext: str`[547]

A string containing the entire interpreted text input.

`text: str`[548]

The interpreted text content.

[538] https://docs.python.org/3/library/constants.html#None
[539] https://docs.python.org/3/library/functions.html#bool
[540] https://docs.python.org/3/library/constants.html#None
[541] https://docs.python.org/3/library/stdtypes.html#str
[542] https://docs.python.org/3/library/functions.html#int
[543] https://docs.python.org/3/library/constants.html#None

class `sphinx.util.docutils.`**`ReferenceRole`**

A base class for reference roles.

The reference roles can accpet `link title <target>` style as a text for the role. The parsed result; link title and target will be stored to `self.title` and `self.target`.

`disabled:` `bool`[549]

A boolean indicates the reference is disabled.

`has_explicit_title:` `bool`[550]

A boolean indicates the role has explicit title or not.

`target:` `str`[551]

The link target for the interpreted text.

`title:` `str`[552]

The link title for the interpreted text.

class `sphinx.transforms.post_transforms.images.`**`ImageConverter`**(**args: Any*, ***kwargs: Any*)

A base class for image converters.

An image converter is kind of Docutils transform module. It is used to convert image files which does not supported by builder to appropriate format for that builder.

For example, *`LaTeX builder`* supports PDF, PNG and JPEG as image formats. However it does not support SVG images. For such case, to use image converters allows to embed these unsupported images into the document. One of image converters; *sphinx.ext.imgconverter* can convert a SVG image to PNG format using Imagemagick internally.

There are three steps to make your custom image converter:

1. Make a subclass of `ImageConverter` class
2. Override `conversion_rules`, `is_available()` and `convert()`
3. Register your image converter to Sphinx using *`Sphinx.add_post_transform()`*

`convert`(*_from: str*[553], *_to: str*[554]) → bool[555]

Convert a image file to expected format.

_from is a path for source image file, and *_to* is a path for destination file.

`is_available`() → bool[556]

Return the image converter is available or not.

`available = None`

The converter is available or not. Will be filled at the first call of the build. The result is shared in the same process.

Todo: This should be refactored not to store the state without class variable.

`conversion_rules = []`

A conversion rules the image converter supports. It is represented as a list of pair of source image format (mimetype) and destination one:

[544] https://docs.python.org/3/library/stdtypes.html#str
[545] https://docs.python.org/3/library/functions.html#int
[546] https://docs.python.org/3/library/stdtypes.html#str
[547] https://docs.python.org/3/library/stdtypes.html#str
[548] https://docs.python.org/3/library/stdtypes.html#str
[549] https://docs.python.org/3/library/functions.html#bool
[550] https://docs.python.org/3/library/functions.html#bool
[551] https://docs.python.org/3/library/stdtypes.html#str
[552] https://docs.python.org/3/library/stdtypes.html#str

```
conversion_rules = [
    ('image/svg+xml', 'image/png'),
    ('image/gif', 'image/png'),
    ('application/pdf', 'image/png'),
]
```

Utility components

class sphinx.events.**EventManager**(*app: Sphinx*)

Event manager for Sphinx.

add(*name: str*[557]) → None[558]

Register a custom Sphinx event.

connect(*name: str*[559], *callback: Callable*, *priority: int*[560]) → int[561]

Connect a handler to specific event.

disconnect(*listener_id: int*[562]) → None[563]

Disconnect a handler.

emit(*name: str*[564], **args: Any*, *allowed_exceptions: Tuple[Type[Exception*[565]*], ...] = ()*) → List

Emit a Sphinx event.

emit_firstresult(*name: str*[566], **args: Any*, *allowed_exceptions: Tuple[Type[Exception*[567]*], ...] = ()*) → Any

Emit a Sphinx event and returns first result.

This returns the result of the first handler that doesn't return `None`.

Deprecated APIs

On developing Sphinx, we are always careful to the compatibility of our APIs. But, sometimes, the change of interface are needed for some reasons. In such cases, we've marked them as deprecated. And they are kept during the two major versions (for more details, please see *Deprecation policy*).

The following is a list of deprecated interfaces.

Table 4: deprecated APIs

Target	Deprecated	(will be) Removed	Alternatives
`favicon` variable in HTML templates	4.0	TBD	`favicon_url`
`logo` variable in HTML templates	4.0	TBD	`logo_url`

continues on next page

[553] https://docs.python.org/3/library/stdtypes.html#str
[554] https://docs.python.org/3/library/stdtypes.html#str
[555] https://docs.python.org/3/library/functions.html#bool
[556] https://docs.python.org/3/library/functions.html#bool
[557] https://docs.python.org/3/library/stdtypes.html#str
[558] https://docs.python.org/3/library/constants.html#None
[559] https://docs.python.org/3/library/stdtypes.html#str
[560] https://docs.python.org/3/library/functions.html#int
[561] https://docs.python.org/3/library/functions.html#int
[562] https://docs.python.org/3/library/functions.html#int
[563] https://docs.python.org/3/library/constants.html#None
[564] https://docs.python.org/3/library/stdtypes.html#str
[565] https://docs.python.org/3/library/exceptions.html#Exception
[566] https://docs.python.org/3/library/stdtypes.html#str
[567] https://docs.python.org/3/library/exceptions.html#Exception

Table 4 – continued from previous page

Target	Deprecated	(will be) Removed	Alternatives
`sphinx.directives.patches.CSVTable`	4.0	6.0	`docutils.parsers.rst.diretives.tables.CSVTable`
`sphinx.directives.patches.ListTable`	4.0	6.0	`docutils.parsers.rst.diretives.tables.ListSVTable`
`sphinx.directives.patches.RSTTable`	4.0	6.0	`docutils.parsers.rst.diretives.tables.RSTTable`
`sphinx.transforms.FigureAligner`	4.0	6.0	N/A
`sphinx.util.pycompat.convert_with_2to3()`	4.0	6.0	N/A
`sphinx.util.pycompat.execfile_()`	4.0	6.0	N/A
`sphinx.util.smartypants`	4.0	6.0	`docutils.utils.smartyquotes`
pending_xref node for viewcode extension	3.5	5.0	`sphinx.ext.viewcode.viewcode_anchor`
`sphinx.builders.linkcheck.CheckExternalLinksBuilder.anchors_ignore`	3.5	5.0	N/A
`sphinx.builders.linkcheck.CheckExternalLinksBuilder.auth`	3.5	5.0	N/A
`sphinx.builders.linkcheck.CheckExternalLinksBuilder.broken`	3.5	5.0	N/A
`sphinx.builders.linkcheck.CheckExternalLinksBuilder.good`	3.5	5.0	N/A
`sphinx.builders.linkcheck.CheckExternalLinksBuilder.redirected`	3.5	5.0	N/A
`sphinx.builders.linkcheck.CheckExternalLinksBuilder.rqueue`	3.5	5.0	N/A
`sphinx.builders.linkcheck.CheckExternalLinksBuilder.to_ignore`	3.5	5.0	N/A
`sphinx.builders.linkcheck.CheckExternalLinksBuilder.workers`	3.5	5.0	N/A
`sphinx.builders.linkcheck.CheckExternalLinksBuilder.wqueue`	3.5	5.0	N/A
`sphinx.builders.linkcheck.node_line_or_0()`	3.5	5.0	`sphinx.util.nodes.get_node_line()`
`sphinx.ext.autodoc.AttributeDocumenter.isinstanceattribute()`	3.5	5.0	N/A
`sphinx.ext.autodoc.importer.get_module_members()`	3.5	5.0	`sphinx.ext.autodoc.ModuleDocumenter.get_module_members()`
`sphinx.ext.autosummary.generate._simple_info()`	3.5	5.0	*Logging API*
`sphinx.ext.autosummary.generate._simple_warn()`	3.5	5.0	*Logging API*
`sphinx.writers.html.HTMLTranslator.permalink_text`	3.5	5.0	*`html_permalinks_icon`*

continues on next page

Table 4 – continued from previous page

Target	Deprecated	(will be) Removed	Alternatives
`sphinx.writers.html5.HTML5Translator.permalink_text`	3.5	5.0	*html_permalinks_icon*
The `follow_wrapped` argument of `sphinx.util.inspect.signature()`	3.4	5.0	N/A
The `no_docstring` argument of `sphinx.ext.autodoc.Documenter.add_content()`	3.4	5.0	`sphinx.ext.autodoc.Documenter.get_doc()`
`sphinx.ext.autodoc.Documenter.get_object_members()`	3.4	6.0	`sphinx.ext.autodoc.ClassDocumenter.get_object_members()`
`sphinx.ext.autodoc.DataDeclarationDocumenter`	3.4	5.0	`sphinx.ext.autodoc.DataDocumenter`
`sphinx.ext.autodoc.GenericAliasDocumenter`	3.4	5.0	`sphinx.ext.autodoc.DataDocumenter`
`sphinx.ext.autodoc.InstanceAttributeDocumenter`	3.4	5.0	`sphinx.ext.autodoc.AttributeDocumenter`
`sphinx.ext.autodoc.SlotsAttributeDocumenter`	3.4	5.0	`sphinx.ext.autodoc.AttributeDocumenter`
`sphinx.ext.autodoc.TypeVarDocumenter`	3.4	5.0	`sphinx.ext.autodoc.DataDocumenter`
`sphinx.ext.autodoc.directive.DocumenterBridge.reporter`	3.5	5.0	`sphinx.util.logging`
`sphinx.ext.autodoc.importer._getannotations()`	3.4	4.0	`sphinx.util.inspect.getannotations()`
`sphinx.ext.autodoc.importer._getmro()`	3.4	4.0	`sphinx.util.inspect.getmro()`
`sphinx.pycode.ModuleAnalyzer.parse()`	3.4	5.0	`sphinx.pycode.ModuleAnalyzer.analyze()`
`sphinx.util.osutil.movefile()`	3.4	5.0	`os.replace()`
`sphinx.util.requests.is_ssl_error()`	3.4	5.0	N/A
`sphinx.builders.latex.LaTeXBuilder.usepackages`	3.3	5.0	N/A
`sphinx.builders.latex.LaTeXBuilder.usepackages_afger_hyperref`	3.3	5.0	N/A
`sphinx.ext.autodoc.SingledispatchFunctionDocumenter`	3.3	5.0	`sphinx.ext.autodoc.FunctionDocumenter`
`sphinx.ext.autodoc.SingledispatchMethodDocumenter`	3.3	5.0	`sphinx.ext.autodoc.MethodDocumenter`
`sphinx.ext.autodoc.members_set_option()`	3.2	5.0	N/A
`sphinx.ext.autodoc.merge_special_members_option()`	3.2	5.0	`sphinx.ext.autodoc.merge_members_option()`
`sphinx.writers.texinfo.TexinfoWriter.desc`	3.2	5.0	`sphinx.writers.texinfo.TexinfoWriter.descs`

continues on next page

Table 4 – continued from previous page

Target	Depre-cated	(will be) Removed	Alternatives
The first argument for `sphinx.ext.autosummary.generate.AutosummaryRenderer` has been changed to Sphinx object	3.1	5.0	N/A
`sphinx.ext.autosummary.generate.AutosummaryRenderer` takes an object type as an argument	3.1	5.0	N/A
The `ignore` argument of `sphinx.ext.autodoc.Documenter.get_doc()`	3.1	5.0	N/A
The `template_dir` argument of `sphinx.ext.autosummary.generate.AutosummaryRenderer`	3.1	5.0	N/A
The `module` argument of `sphinx.ext.autosummary.generate.find_autosummary_in_docstring()`	3.0	5.0	N/A
The `builder` argument of `sphinx.ext.autosummary.generate.generate_autosummary_docs()`	3.1	5.0	N/A
The `template_dir` argument of `sphinx.ext.autosummary.generate.generate_autosummary_docs()`	3.1	5.0	N/A
`sphinx.ext.autosummary.generate.AutosummaryRenderer.exists()`	3.1	5.0	N/A
The `ignore` argument of `sphinx.util.docstring.prepare_docstring()`	3.1	5.0	N/A
`sphinx.util.rpartition()`	3.1	5.0	`str.rpartition()`
`desc_signature['first']`		3.0	N/A
`sphinx.directives.DescDirective`	3.0	5.0	`sphinx.directives.ObjectDescription`
`sphinx.domains.std.StandardDomain.add_object()`	3.0	5.0	`sphinx.domains.std.StandardDomain.note_object()`
`sphinx.domains.python.PyDecoratorMixin`	3.0	5.0	N/A
`sphinx.ext.autodoc.get_documenters()`	3.0	5.0	`sphinx.registry.documenters`
`sphinx.ext.autosummary.process_autosummary_toc()`	3.0	5.0	N/A
`sphinx.parsers.Parser.app`	3.0	5.0	N/A
`sphinx.testing.path.Path.text()`	3.0	5.0	`sphinx.testing.path.Path.read_text()`
`sphinx.testing.path.Path.bytes()`	3.0	5.0	`sphinx.testing.path.Path.read_bytes()`
`sphinx.util.inspect.getargspec()`	3.0	5.0	`inspect.getargspec()`
`sphinx.writers.latex.LaTeXWriter.format_docclass()`	3.0	5.0	LaTeX Themes
decode argument of `sphinx.pycode.ModuleAnalyzer()`	2.4	4.0	N/A
`sphinx.directives.other.Index`	2.4	4.0	`sphinx.domains.index.IndexDirective`

continues on next page

Table 4 – continued from previous page

Target	Deprecated	(will be) Removed	Alternatives
`sphinx.environment.temp_data['gloss_entries']`	2.4	4.0	`documents.nameids`
`sphinx.environment.BuildEnvironment.indexentries`	2.4	4.0	`sphinx.domains.index.IndexDomain`
`sphinx.environment.collectors.indexentries.IndexEntriesCollector`	2.4	4.0	`sphinx.domains.index.IndexDomain`
`sphinx.io.FiletypeNotFoundError`	2.4	4.0	`sphinx.errors.FiletypeNotFoundError`
`sphinx.ext.apidoc.INITPY`	2.4	4.0	N/A
`sphinx.ext.apidoc.shall_skip()`	2.4	4.0	`sphinx.ext.apidoc.is_skipped_package`
`sphinx.io.get_filetype()`	2.4	4.0	`sphinx.util.get_filetype()`
`sphinx.pycode.ModuleAnalyzer.encoding`	2.4	4.0	N/A
`sphinx.roles.Index`	2.4	4.0	`sphinx.domains.index.IndexRole`
`sphinx.util.detect_encoding()`	2.4	4.0	`tokenize.detect_encoding()`
`sphinx.util.get_module_source()`	2.4	4.0	N/A
`sphinx.util.inspect.Signature`	2.4	4.0	`sphinx.util.inspect.signature` and `sphinx.util.inspect.stringify_signature()`
`sphinx.util.inspect.safe_getmembers()`	2.4	4.0	`inspect.getmembers()`
`sphinx.writers.latex.LaTeXTranslator.settings.author`	2.4	4.0	N/A
`sphinx.writers.latex.LaTeXTranslator.settings.contentsname`	2.4	4.0	`document['contentsname']`
`sphinx.writers.latex.LaTeXTranslator.settings.docclass`	2.4	4.0	`document['docclass']`
`sphinx.writers.latex.LaTeXTranslator.settings.docname`	2.4	4.0	N/A
`sphinx.writers.latex.LaTeXTranslator.settings.title`	2.4	4.0	N/A
`sphinx.writers.latex.ADDITIONAL_SETTINGS`	2.4	4.0	`sphinx.builders.latex.constants.ADDITIONAL_SETTINGS`
`sphinx.writers.latex.DEFAULT_SETTINGS`	2.4	4.0	`sphinx.builders.latex.constants.DEFAULT_SETTINGS`
`sphinx.writers.latex.LUALATEX_DEFAULT_FONTPKG`	2.4	4.0	`sphinx.builders.latex.constants.LUALATEX_DEFAULT_FONTPKG`
`sphinx.writers.latex.PDFLATEX_DEFAULT_FONTPKG`	2.4	4.0	`sphinx.builders.latex.constants.PDFLATEX_DEFAULT_FONTPKG`
`sphinx.writers.latex.XELATEX_DEFAULT_FONTPKG`	2.4	4.0	`sphinx.builders.latex.constants.XELATEX_DEFAULT_FONTPKG`

continues on next page

Table 4 – continued from previous page

Target	Deprecated	(will be) Removed	Alternatives
`sphinx.writers.latex.XELATEX_GREEK_DEFAULT_FONTPKG`	2.4	4.0	`sphinx.builders.latex.constants.XELATEX_GREEK_DEFAULT_FONTPKG`
`sphinx.builders.gettext.POHEADER`	2.3	4.0	`sphinx/templates/gettext/message.pot_t` (template file)
`sphinx.io.SphinxStandaloneReader.app`	2.3	4.0	`sphinx.io.SphinxStandaloneReader.setup()`
`sphinx.io.SphinxStandaloneReader.env`	2.3	4.0	`sphinx.io.SphinxStandaloneReader.setup()`
`sphinx.util.texescape.tex_escape_map`	2.3	4.0	`sphinx.util.texescape.escape()`
`sphinx.util.texescape.tex_hl_escape_map_new`	2.3	4.0	`sphinx.util.texescape.hlescape()`
`sphinx.writers.latex.LaTeXTranslator.no_contractions`	2.3	4.0	N/A
`sphinx.domains.math.MathDomain.add_equation()`	2.2	4.0	`sphinx.domains.math.MathDomain.note_equation()`
`sphinx.domains.math.MathDomain.get_next_equation_number()`	2.2	4.0	`sphinx.domains.math.MathDomain.note_equation()`
The `info` and `warn` arguments of `sphinx.ext.autosummary.generate.generate_autosummary_docs()`	2.2	4.0	`logging.info()` and `logging.warning()`
`sphinx.ext.autosummary.generate._simple_info()`	2.2	4.0	`logging.info()`
`sphinx.ext.autosummary.generate._simple_warn()`	2.2	4.0	`logging.warning()`
`sphinx.ext.todo.merge_info()`	2.2	4.0	`sphinx.ext.todo.TodoDomain`
`sphinx.ext.todo.process_todo_nodes()`	2.2	4.0	`sphinx.ext.todo.TodoDomain`
`sphinx.ext.todo.process_todos()`	2.2	4.0	`sphinx.ext.todo.TodoDomain`
`sphinx.ext.todo.purge_todos()`	2.2	4.0	`sphinx.ext.todo.TodoDomain`
`sphinx.builders.latex.LaTeXBuilder.apply_transforms()`	2.1	4.0	N/A
`sphinx.builders._epub_base.EpubBuilder.esc()`	2.1	4.0	`html.escape()`
`sphinx.directives.Acks`	2.1	4.0	`sphinx.directives.other.Acks`
`sphinx.directives.Author`	2.1	4.0	`sphinx.directives.other.Author`
`sphinx.directives.Centered`	2.1	4.0	`sphinx.directives.other.Centered`
`sphinx.directives.Class`	2.1	4.0	`sphinx.directives.other.Class`
`sphinx.directives.CodeBlock`	2.1	4.0	`sphinx.directives.code.CodeBlock`
`sphinx.directives.Figure`	2.1	4.0	`sphinx.directives.patches.Figure`
`sphinx.directives.HList`	2.1	4.0	`sphinx.directives.other.HList`
`sphinx.directives.Highlight`	2.1	4.0	`sphinx.directives.code.Highlight`
`sphinx.directives.Include`	2.1	4.0	`sphinx.directives.other.Include`

continues on next page

Table 4 – continued from previous page

Target	Deprecated	(will be) Removed	Alternatives
sphinx.directives.Index	2.1	4.0	sphinx.directives.other.Index
sphinx.directives.LiteralInclude	2.1	4.0	sphinx.directives.code.LiteralInclude
sphinx.directives.Meta	2.1	4.0	sphinx.directives.patches.Meta
sphinx.directives.Only	2.1	4.0	sphinx.directives.other.Only
sphinx.directives.SeeAlso	2.1	4.0	sphinx.directives.other.SeeAlso
sphinx.directives.TabularColumns	2.1	4.0	sphinx.directives.other.TabularColumns
sphinx.directives.TocTree	2.1	4.0	sphinx.directives.other.TocTree
sphinx.directives.VersionChange	2.1	4.0	sphinx.directives.other.VersionChange
sphinx.domains.python.PyClassmember	2.1	4.0	sphinx.domains.python.PyAttribute, sphinx.domains.python.PyMethod, sphinx.domains.python.PyClassMethod, sphinx.domains.python.PyObject and sphinx.domains.python.PyStaticMethod
sphinx.domains.python.PyModulelevel	2.1	4.0	sphinx.domains.python.PyFunction, sphinx.domains.python.PyObject and sphinx.domains.python.PyVariable
sphinx.domains.std.StandardDomain._resolve_citation_xref()	2.1	4.0	sphinx.domains.citation.CitationDomain.resolve_xref()
sphinx.domains.std.StandardDomain.note_citations()	2.1	4.0	sphinx.domains.citation.CitationDomain.note_citation()
sphinx.domains.std.StandardDomain.note_citation_refs()	2.1	4.0	sphinx.domains.citation.CitationDomain.note_citation_reference()
sphinx.domains.std.StandardDomain.note_labels()	2.1	4.0	sphinx.domains.std.StandardDomain.process_doc()
sphinx.environment.NoUri	2.1	4.0	sphinx.errors.NoUri
sphinx.ext.apidoc.format_directive()	2.1	4.0	N/A
sphinx.ext.apidoc.format_heading()	2.1	4.0	N/A
sphinx.ext.apidoc.makename()	2.1	4.0	sphinx.ext.apidoc.module_join()
sphinx.ext.autodoc.importer.MockFinder	2.1	4.0	sphinx.ext.autodoc.mock.MockFinder
sphinx.ext.autodoc.importer.MockLoader	2.1	4.0	sphinx.ext.autodoc.mock.MockLoader
sphinx.ext.autodoc.importer.mock()	2.1	4.0	sphinx.ext.autodoc.mock.mock()
sphinx.ext.autosummary.autolink_role()	2.1	4.0	sphinx.ext.autosummary.AutoLink
sphinx.ext.imgmath.DOC_BODY	2.1	4.0	N/A

continues on next page

Table 4 – continued from previous page

Target	Depre-cated	(will be) Removed	Alternatives
`sphinx.ext.imgmath.DOC_BODY_PREVIEW`	2.1	4.0	N/A
`sphinx.ext.imgmath.DOC_HEAD`	2.1	4.0	N/A
`sphinx.transforms.CitationReferences`	2.1	4.0	`sphinx.domains.citation.CitationReferenceTransform`
`sphinx.transforms.SmartQuotesSkipper`	2.1	4.0	`sphinx.domains.citation.CitationDefinitionTransform`
`sphinx.util.docfields.DocFieldTransformer.preprocess_fieldtypes()`	2.1	4.0	`sphinx.directives.ObjectDescription.get_field_type_map()`
`sphinx.util.node.find_source_node()`	2.1	4.0	`sphinx.util.node.get_node_source()`
`sphinx.util.i18n.find_catalog()`	2.1	4.0	`sphinx.util.i18n.docname_to_domain()`
`sphinx.util.i18n.find_catalog_files()`	2.1	4.0	`sphinx.util.i18n.CatalogRepository`
`sphinx.util.i18n.find_catalog_source_files()`	2.1	4.0	`sphinx.util.i18n.CatalogRepository`
`encoding` argument of `autodoc.Documenter.get_doc()`, `autodoc.DocstringSignatureMixin.get_doc()`, `autodoc.DocstringSignatureMixin._find_signature()`, and `autodoc.ClassDocumenter.get_doc()`	2.0	4.0	N/A
arguments of `EpubBuilder.build_mimetype()`, `EpubBuilder.build_container()`, `EpubBuilder.build_content()`, `EpubBuilder.build_toc()` and `EpubBuilder.build_epub()`	2.0	4.0	N/A
arguments of `Epub3Builder.build_navigation_doc()`	2.0	4.0	N/A
`nodetype` argument of `sphinx.search.WordCollector.is_meta_keywords()`	2.0	4.0	N/A
`suffix` argument of `BuildEnvironment.doc2path()`	2.0	4.0	N/A
string style `base` argument of `BuildEnvironment.doc2path()`	2.0	4.0	`os.path.join()`
`sphinx.addnodes.abbreviation`	2.0	4.0	`docutils.nodes.abbreviation`
`sphinx.builders.applehelp`	2.0	4.0	`sphinxcontrib.applehelp`
`sphinx.builders.devhelp`	2.0	4.0	`sphinxcontrib.devhelp`
`sphinx.builders.epub3.Epub3Builder.validate_config_value()`	2.0	4.0	`sphinx.builders.epub3.validate_config_values()`
`sphinx.builders.html.JSONHTMLBuilder`	2.0	4.0	`sphinx.builders.serializinghtml.JSONHTMLBuilder`

continues on next page

Table 4 – continued from previous page

Target	Deprecated	(will be) Removed	Alternatives
sphinx.builders.html.PickleHTMLBuilder	2.0	4.0	sphinx.builders.serializinghtml.PickleHTMLBuilder
sphinx.builders.html.SerializingHTMLBuilder	2.0	4.0	sphinx.builders.serializinghtml.SerializingHTMLBuilder
sphinx.builders.html.SingleFileHTMLBuilder	2.0	4.0	sphinx.builders.singlehtml.SingleFileHTMLBuilder
sphinx.builders.html.WebHTMLBuilder	2.0	4.0	sphinx.builders.serializinghtml.PickleHTMLBuilder
sphinx.builders.htmlhelp	2.0	4.0	sphinxcontrib.htmlhelp
sphinx.builders.htmlhelp.HTMLHelpBuilder.open_file()	2.0	4.0	open()
sphinx.builders.qthelp	2.0	4.0	sphinxcontrib.qthelp
sphinx.cmd.quickstart.term_decode()	2.0	4.0	N/A
sphinx.cmd.quickstart.TERM_ENCODING	2.0	4.0	sys.stdin.encoding
sphinx.config.check_unicode()	2.0	4.0	N/A
sphinx.config.string_classes	2.0	4.0	[str]
sphinx.domains.cpp.DefinitionError.description	2.0	4.0	str(exc)
sphinx.domains.cpp.NoOldIdError.description	2.0	4.0	str(exc)
sphinx.domains.cpp.UnsupportedMultiCharacterCharLiteral.decoded	2.0	4.0	str(exc)
sphinx.ext.autosummary.Autosummary.warn()	2.0	4.0	N/A
sphinx.ext.autosummary.Autosummary.genopt	2.0	4.0	N/A
sphinx.ext.autosummary.Autosummary.warnings	2.0	4.0	N/A
sphinx.ext.autosummary.Autosummary.result	2.0	4.0	N/A
sphinx.ext.doctest.doctest_encode()	2.0	4.0	N/A
sphinx.ext.jsmath	2.0	4.0	sphinxcontrib.jsmath
sphinx.roles.abbr_role()	2.0	4.0	sphinx.roles.Abbreviation
sphinx.roles.emph_literal_role()	2.0	4.0	sphinx.roles.EmphasizedLiteral
sphinx.roles.menusel_role()	2.0	4.0	sphinx.roles.GUILabel or sphinx.roles.MenuSelection
sphinx.roles.index_role()	2.0	4.0	sphinx.roles.Index
sphinx.roles.indexmarkup_role()	2.0	4.0	sphinx.roles.PEP or sphinx.roles.RFC
sphinx.testing.util.remove_unicode_literal()	2.0	4.0	N/A
sphinx.util.attrdict	2.0	4.0	N/A

continues on next page

Table 4 – continued from previous page

Target	Deprecated	(will be) Removed	Alternatives
`sphinx.util.force_decode()`	2.0	4.0	N/A
`sphinx.util.get_matching_docs()`	2.0	4.0	`sphinx.util.get_matching_files()`
`sphinx.util.inspect.Parameter`	2.0	3.0	N/A
`sphinx.util.jsonimpl`	2.0	4.0	`sphinxcontrib.serializinghtml.jsonimpl`
`sphinx.util.osutil.EEXIST`	2.0	4.0	`errno.EEXIST` or `FileExistsError`
`sphinx.util.osutil.EINVAL`	2.0	4.0	`errno.EINVAL`
`sphinx.util.osutil.ENOENT`	2.0	4.0	`errno.ENOENT` or `FileNotFoundError`
`sphinx.util.osutil.EPIPE`	2.0	4.0	`errno.ENOENT` or `BrokenPipeError`
`sphinx.util.osutil.walk()`	2.0	4.0	`os.walk()`
`sphinx.util.pycompat.NoneType`	2.0	4.0	`sphinx.util.typing.NoneType`
`sphinx.util.pycompat.TextIOWrapper`	2.0	4.0	`io.TextIOWrapper`
`sphinx.util.pycompat.UnicodeMixin`	2.0	4.0	N/A
`sphinx.util.pycompat.htmlescape()`	2.0	4.0	`html.escape()`
`sphinx.util.pycompat.indent()`	2.0	4.0	`textwrap.indent()`
`sphinx.util.pycompat.sys_encoding`	2.0	4.0	`sys.getdefaultencoding()`
`sphinx.util.pycompat.terminal_safe()`	2.0	4.0	`sphinx.util.console.terminal_safe()`
`sphinx.util.pycompat.u`	2.0	4.0	N/A
`sphinx.util.PeekableIterator`	2.0	4.0	N/A
Omitting the `filename` argument in an overriddent `IndexBuilder.feed()` method.	2.0	4.0	`IndexBuilder.feed(docname, filename, title, doctree)`
`sphinx.writers.latex.ExtBabel`	2.0	4.0	`sphinx.builders.latex.util.ExtBabel`
`sphinx.writers.latex.LaTeXTranslator.babel_defmacro()`	2.0	4.0	N/A
`sphinx.application.Sphinx._setting_up_extension`	2.0	3.0	N/A
The `importer` argument of `sphinx.ext.autodoc.importer._MockModule`	2.0	3.0	N/A
`sphinx.ext.autodoc.importer._MockImporter`	2.0	3.0	N/A
`sphinx.io.SphinxBaseFileInput`	2.0	3.0	N/A
`sphinx.io.SphinxFileInput.supported`	2.0	3.0	N/A
`sphinx.io.SphinxRSTFileInput`	2.0	3.0	N/A
`sphinx.registry.SphinxComponentRegistry.add_source_input()`	2.0	3.0	N/A
`sphinx.writers.latex.LaTeXTranslator._make_visit_admonition()`	2.0	3.0	N/A
`sphinx.writers.latex.LaTeXTranslator.collect_footnotes()`	2.0	4.0	N/A

continues on next page

Table 4 – continued from previous page

Target	Deprecated	(will be) Removed	Alternatives
`sphinx.writers.texinfo.TexinfoTranslator._make_visit_admonition()`	2.0	3.0	N/A
`sphinx.writers.text.TextTranslator._make_depart_admonition()`	2.0	3.0	N/A
`sphinx.writers.latex.LaTeXTranslator.generate_numfig_format()`	2.0	4.0	N/A
`highlightlang`	1.8	4.0	*highlight*
`add_stylesheet()`	1.8	4.0	*add_css_file()*
`add_javascript()`	1.8	4.0	*add_js_file()*
autodoc_default_flags	1.8	4.0	*autodoc_default_options*
`content` arguments of `sphinx.util.image.guess_mimetype()`	1.8	3.0	N/A
`gettext_compact` arguments of `sphinx.util.i18n.find_catalog_source_files()`	1.8	3.0	N/A
`sphinx.io.SphinxI18nReader.set_lineno_for_reporter()`	1.8	3.0	N/A
`sphinx.io.SphinxI18nReader.line`	1.8	3.0	N/A
`sphinx.directives.other.VersionChanges`	1.8	3.0	`sphinx.domains.changeset.VersionChanges`
`sphinx.highlighting.PygmentsBridge.unhighlight()`	1.8	3.0	N/A
`trim_doctest_flags` arguments of `sphinx.highlighting.PygmentsBridge`	1.8	3.0	N/A
`sphinx.ext.mathbase`	1.8	3.0	N/A
`sphinx.ext.mathbase.MathDomain`	1.8	3.0	`sphinx.domains.math.MathDomain`
`sphinx.ext.mathbase.MathDirective`	1.8	3.0	`sphinx.directives.patches.MathDirective`
`sphinx.ext.mathbase.math_role()`	1.8	3.0	`docutils.parsers.rst.roles.math_role()`
`sphinx.ext.mathbase.setup_math()`	1.8	3.0	*add_html_math_renderer()*
`sphinx.ext.mathbase.is_in_section_title()`	1.8	3.0	N/A
`sphinx.ext.mathbase.get_node_equation_number()`	1.8	3.0	`sphinx.util.math.get_node_equation_number()`
`sphinx.ext.mathbase.wrap_displaymath()`	1.8	3.0	`sphinx.util.math.wrap_displaymath()`
`sphinx.ext.mathbase.math` (node)	1.8	3.0	`docutils.nodes.math`
`sphinx.ext.mathbase.displaymath` (node)	1.8	3.0	`docutils.nodes.math_block`
`sphinx.ext.mathbase.eqref` (node)	1.8	3.0	`sphinx.builders.latex.nodes.math_reference`
`viewcode_import` (config value)	1.8	3.0	*viewcode_follow_imported_members*
`sphinx.writers.latex.Table.caption_footnotetexts`	1.8	3.0	N/A

continues on next page

Table 4 – continued from previous page

Target	Deprecated	(will be) Removed	Alternatives
`sphinx.writers.latex.Table.header_footnotetexts`	1.8	3.0	N/A
`sphinx.writers.latex.LaTeXTranslator.footnotestack`	1.8	3.0	N/A
`sphinx.writers.latex.LaTeXTranslator.in_container_literal_block`	1.8	3.0	N/A
`sphinx.writers.latex.LaTeXTranslator.next_section_ids`	1.8	3.0	N/A
`sphinx.writers.latex.LaTeXTranslator.next_hyperlink_ids`	1.8	3.0	N/A
`sphinx.writers.latex.LaTeXTranslator.restrict_footnote()`	1.8	3.0	N/A
`sphinx.writers.latex.LaTeXTranslator.unrestrict_footnote()`	1.8	3.0	N/A
`sphinx.writers.latex.LaTeXTranslator.push_hyperlink_ids()`	1.8	3.0	N/A
`sphinx.writers.latex.LaTeXTranslator.pop_hyperlink_ids()`	1.8	3.0	N/A
`sphinx.writers.latex.LaTeXTranslator.bibitems`	1.8	3.0	N/A
`sphinx.writers.latex.LaTeXTranslator.hlsettingstack`	1.8	3.0	N/A
`sphinx.writers.latex.ExtBabel.get_shorthandoff()`	1.8	3.0	N/A
`sphinx.writers.html.HTMLTranslator.highlightlang()`	1.8	3.0	N/A
`sphinx.writers.html.HTMLTranslator.highlightlang_base()`	1.8	3.0	N/A
`sphinx.writers.html.HTMLTranslator.highlightlangopts()`	1.8	3.0	N/A
`sphinx.writers.html.HTMLTranslator.highlightlinenothreshold()`	1.8	3.0	N/A
`sphinx.writers.html5.HTMLTranslator.highlightlang()`	1.8	3.0	N/A
`sphinx.writers.html5.HTMLTranslator.highlightlang_base()`	1.8	3.0	N/A
`sphinx.writers.html5.HTMLTranslator.highlightlangopts()`	1.8	3.0	N/A

continues on next page

Table 4 – continued from previous page

Target	Deprecated	(will be) Removed	Alternatives
`sphinx.writers.html5.HTMLTranslator.highlightlinenothreshold()`	1.8	3.0	N/A
`sphinx.writers.latex.LaTeXTranslator.check_latex_elements()`	1.8	3.0	Nothing
`sphinx.application.CONFIG_FILENAME`	1.8	3.0	`sphinx.config.CONFIG_FILENAME`
`Config.check_unicode()`	1.8	3.0	`sphinx.config.check_unicode()`
`Config.check_types()`	1.8	3.0	`sphinx.config.check_confval_types()`
`dirname`, `filename` and `tags` arguments of `Config.__init__()`	1.8	3.0	`Config.read()`
The value of *html_search_options*	1.8	3.0	see *html_search_options*
`sphinx.versioning.prepare()`	1.8	3.0	`sphinx.versioning.UIDTransform`
`Sphinx.override_domain()`	1.8	3.0	*add_domain()*
`Sphinx.import_object()`	1.8	3.0	`sphinx.util.import_object()`
`suffix` argument of *add_source_parser()*	1.8	3.0	*add_source_suffix()*
`BuildEnvironment.load()`	1.8	3.0	`pickle.load()`
`BuildEnvironment.loads()`	1.8	3.0	`pickle.loads()`
`BuildEnvironment.frompickle()`	1.8	3.0	`pickle.load()`
`BuildEnvironment.dump()`	1.8	3.0	`pickle.dump()`
`BuildEnvironment.dumps()`	1.8	3.0	`pickle.dumps()`
`BuildEnvironment.topickle()`	1.8	3.0	`pickle.dump()`
`BuildEnvironment._nitpick_ignore`	1.8	3.0	*nitpick_ignore*
`BuildEnvironment.versionchanges`	1.8	3.0	N/A
`BuildEnvironment.update()`	1.8	3.0	`Builder.read()`
`BuildEnvironment.read_doc()`	1.8	3.0	`Builder.read_doc()`
`BuildEnvironment._read_serial()`	1.8	3.0	`Builder.read()`
`BuildEnvironment._read_parallel()`	1.8	3.0	`Builder.read()`
`BuildEnvironment.write_doctree()`	1.8	3.0	`Builder.write_doctree()`
`BuildEnvironment.note_versionchange()`	1.8	3.0	`ChangesDomain.note_changeset()`
`warn()` (template helper function)	1.8	3.0	`warning()`
source_parsers	1.8	3.0	*add_source_parser()*
`sphinx.util.docutils.directive_helper()`	1.8	3.0	`Directive` class of docutils
`sphinx.cmdline`	1.8	3.0	`sphinx.cmd.build`
`sphinx.make_mode`	1.8	3.0	`sphinx.cmd.make_mode`
`sphinx.locale.l_()`	1.8	3.0	*sphinx.locale._()*
`sphinx.locale.lazy_gettext()`	1.8	3.0	*sphinx.locale._()*
`sphinx.locale.mygettext()`	1.8	3.0	*sphinx.locale._()*
`sphinx.util.copy_static_entry()`	1.5	3.0	`sphinx.util.fileutil.copy_asset()`
`sphinx.build_main()`	1.7	2.0	`sphinx.cmd.build.build_main()`
`sphinx.ext.intersphinx.debug()`	1.7	2.0	`sphinx.ext.intersphinx.inspect_main()`

continues on next page

Table 4 – continued from previous page

Target	Deprecated	(will be) Removed	Alternatives
`sphinx.ext.autodoc.format_annotation()`	1.7	2.0	`sphinx.util.inspect.Signature`
`sphinx.ext.autodoc.formatargspec()`	1.7	2.0	`sphinx.util.inspect.Signature`
`sphinx.ext.autodoc.AutodocReporter`	1.7	2.0	`sphinx.util.docutils.switch_source_input()`
`sphinx.ext.autodoc.add_documenter()`	1.7	2.0	*`add_autodocumenter()`*
`sphinx.ext.autodoc.AutoDirective._register`	1.7	2.0	*`add_autodocumenter()`*
`AutoDirective._special_attrgetters`	1.7	2.0	*`add_autodoc_attrgetter()`*
`Sphinx.warn()`, `Sphinx.info()`	1.6	2.0	*Logging API*
`BuildEnvironment.set_warnfunc()`	1.6	2.0	*Logging API*
`BuildEnvironment.note_toctree()`	1.6	2.0	`Toctree.note()` (in `sphinx.environment.adapters.toctree`)
`BuildEnvironment.get_toc_for()`	1.6	2.0	`Toctree.get_toc_for()` (in `sphinx.environment.adapters.toctree`)
`BuildEnvironment.get_toctree_for()`	1.6	2.0	`Toctree.get_toctree_for()` (in `sphinx.environment.adapters.toctree`)
`BuildEnvironment.create_index()`	1.6	2.0	`IndexEntries.create_index()` (in `sphinx.environment.adapters.indexentries`)
`sphinx.websupport`	1.6	2.0	sphinxcontrib-websupport[568]
`StandaloneHTMLBuilder.css_files`	1.6	2.0	`add_stylesheet()`
`document.settings.gettext_compact`	1.8	1.8	*`gettext_compact`*
`Sphinx.status_iterator()`	1.6	1.7	`sphinx.util.status_iterator()`
`Sphinx.old_status_iterator()`	1.6	1.7	`sphinx.util.old_status_iterator()`
`Sphinx._directive_helper()`	1.6	1.7	`sphinx.util.docutils.directive_helper()`
`sphinx.util.compat.Directive`	1.6	1.7	`docutils.parsers.rst.Directive`
`sphinx.util.compat.docutils_version`	1.6	1.7	`sphinx.util.docutils.__version_info__`

Note: On deprecating on public APIs (internal functions and classes), we also follow the policy as much as possible.

[568] https://pypi.org/project/sphinxcontrib-websupport/

CHAPTER

SEVEN

SPHINX INTERNALS

This guide contains information about the Sphinx open source project itself. This is where you can find information about how Sphinx is managed and learn how to contribute to the project.

7.1 Contributing to Sphinx

There are many ways you can contribute to Sphinx, be it filing bug reports or feature requests, writing new documentation or submitting patches for new or fixed behavior. This guide serves to illustrate how you can get started with this.

Getting help

The Sphinx community maintains a number of mailing lists and IRC channels.

Stack Overflow with tag python-sphinx[569] Questions and answers about use and development.

sphinx-users <sphinx-users@googlegroups.com> Mailing list for user support.

sphinx-dev <sphinx-dev@googlegroups.com> Mailing list for development related discussions.

#sphinx-doc on irc.freenode.net IRC channel for development questions and user support.

Bug Reports and Feature Requests

If you have encountered a problem with Sphinx or have an idea for a new feature, please submit it to the issue tracker[570] on GitHub or discuss it on the sphinx-dev[571] mailing list.

For bug reports, please include the output produced during the build process and also the log file Sphinx creates after it encounters an unhandled exception. The location of this file should be shown towards the end of the error message.

Including or providing a link to the source files involved may help us fix the issue. If possible, try to create a minimal project that produces the error and post that instead.

[569] https://stackoverflow.com/questions/tagged/python-sphinx
[570] https://github.com/sphinx-doc/sphinx/issues
[571] sphinx-dev@googlegroups.com

Writing code

The Sphinx source code is managed using Git and is hosted on GitHub[572]. The recommended way for new contributors to submit code to Sphinx is to fork this repository and submit a pull request after committing changes to their fork. The pull request will then need to be approved by one of the core developers before it is merged into the main repository.

Getting started

Before starting on a patch, we recommend checking for open issues or open a fresh issue to start a discussion around a feature idea or a bug. If you feel uncomfortable or uncertain about an issue or your changes, feel free to email the *sphinx-dev* mailing list.

These are the basic steps needed to start developing on Sphinx.

1. Create an account on GitHub.
2. Fork the main Sphinx repository (sphinx-doc/sphinx[573]) using the GitHub interface.
3. Clone the forked repository to your machine.

   ```
   git clone https://github.com/USERNAME/sphinx
   cd sphinx
   ```

4. Checkout the appropriate branch.

   Sphinx adopts Semantic Versioning 2.0.0 (refs: https://semver.org/).

   For changes that preserves backwards-compatibility of API and features, they should be included in the next MINOR release, use the `A.x` branch.

   ```
   git checkout A.x
   ```

   For incompatible or other substantial changes that should wait until the next MAJOR release, use the `master` branch.

   For urgent release, a new PATCH branch must be branched from the newest release tag (see *Sphinx's release process* for detail).

5. Setup a virtual environment.

   This is not necessary for unit testing, thanks to `tox`, but it is necessary if you wish to run `sphinx-build` locally or run unit tests without the help of `tox`:

   ```
   virtualenv ~/.venv
   . ~/.venv/bin/activate
   pip install -e .
   ```

6. Create a new working branch. Choose any name you like.

   ```
   git checkout -b feature-xyz
   ```

7. Hack, hack, hack.

   Write your code along with tests that shows that the bug was fixed or that the feature works as expected.

8. Add a bullet point to `CHANGES` if the fix or feature is not trivial (small doc updates, typo fixes), then commit:

   ```
   git commit -m '#42: Add useful new feature that does this.'
   ```

[572] https://github.com/sphinx-doc/sphinx
[573] https://github.com/sphinx-doc/sphinx

GitHub recognizes certain phrases that can be used to automatically update the issue tracker. For example:

```
git commit -m 'Closes #42: Fix invalid markup in docstring of Foo.bar.'
```

would close issue #42.

9. Push changes in the branch to your forked repository on GitHub:

```
git push origin feature-xyz
```

10. Submit a pull request from your branch to the respective branch (`master` or `A.x`).
11. Wait for a core developer to review your changes.

Coding style

Please follow these guidelines when writing code for Sphinx:

- Try to use the same code style as used in the rest of the project.
- For non-trivial changes, please update the `CHANGES` file. If your changes alter existing behavior, please document this.
- New features should be documented. Include examples and use cases where appropriate. If possible, include a sample that is displayed in the generated output.
- When adding a new configuration variable, be sure to document it and update `sphinx/cmd/quickstart.py` if it's important enough.
- Add appropriate unit tests.

Style and type checks can be run using `tox`:

```
tox -e mypy
tox -e flake8
```

Unit tests

Sphinx is tested using pytest[574] for Python code and Karma[575] for JavaScript.

To run Python unit tests, we recommend using `tox`, which provides a number of targets and allows testing against multiple different Python environments:

- To list all possible targets:

```
tox -av
```

- To run unit tests for a specific Python version, such as Python 3.6:

```
tox -e py36
```

- To run unit tests for a specific Python version and turn on deprecation warnings on so they're shown in the test output:

```
PYTHONWARNINGS=all tox -e py36
```

- Arguments to `pytest` can be passed via `tox`, e.g. in order to run a particular test:

574 https://docs.pytest.org/en/latest/
575 https://karma-runner.github.io

```
tox -e py36 tests/test_module.py::test_new_feature
```

You can also test by installing dependencies in your local environment:

```
pip install .[test]
```

To run JavaScript tests, use `npm`:

```
npm install
npm run test
```

New unit tests should be included in the `tests` directory where necessary:

- For bug fixes, first add a test that fails without your changes and passes after they are applied.
- Tests that need a `sphinx-build` run should be integrated in one of the existing test modules if possible. New tests that to `@with_app` and then `build_all` for a few assertions are not good since *the test suite should not take more than a minute to run*.

New in version 1.8: Sphinx also runs JavaScript tests.

New in version 1.6: `sphinx.testing` is added as a experimental.

Changed in version 1.5.2: Sphinx was switched from nose to pytest.

Todo: The below belongs in the developer guide

Utility functions and pytest fixtures for testing are provided in `sphinx.testing`. If you are a developer of Sphinx extensions, you can write unit tests with using pytest. At this time, `sphinx.testing` will help your test implementation.

How to use pytest fixtures that are provided by `sphinx.testing`? You can require `'sphinx.testing.fixtures'` in your test modules or `conftest.py` files like this:

```
pytest_plugins = 'sphinx.testing.fixtures'
```

If you want to know more detailed usage, please refer to `tests/conftest.py` and other `test_*.py` files under `tests` directory.

Writing documentation

Todo: Add a more extensive documentation contribution guide.

You can build documentation using `tox`:

```
tox -e docs
```

Translations

The parts of messages in Sphinx that go into builds are translated into several locales. The translations are kept as gettext `.po` files translated from the master template `sphinx/locale/sphinx.pot`.

Sphinx uses Babel[576] to extract messages and maintain the catalog files. It is integrated in `setup.py`:

- Use `python setup.py extract_messages` to update the `.pot` template.
- Use `python setup.py update_catalog` to update all existing language catalogs in `sphinx/locale/*/LC_MESSAGES` with the current messages in the template file.
- Use `python setup.py compile_catalog` to compile the `.po` files to binary `.mo` files and `.js` files.

When an updated `.po` file is submitted, run compile_catalog to commit both the source and the compiled catalogs.

When a new locale is submitted, add a new directory with the ISO 639-1 language identifier and put `sphinx.po` in there. Don't forget to update the possible values for *`language`* in `doc/usage/configuration.rst`.

The Sphinx core messages can also be translated on Transifex[577]. There `tx` client tool, which is provided by the `transifex_client` Python package, can be used to pull translations in `.po` format from Transifex. To do this, go to `sphinx/locale` and then run `tx pull -f -l LANG` where `LANG` is an existing language identifier. It is good practice to run `python setup.py update_catalog` afterwards to make sure the `.po` file has the canonical Babel formatting.

Debugging tips

- Delete the build cache before building documents if you make changes in the code by running the command `make clean` or using the *`sphinx-build -E`* option.
- Use the *`sphinx-build -P`* option to run `pdb` on exceptions.
- Use `node.pformat()` and `node.asdom().toxml()` to generate a printable representation of the document structure.
- Set the configuration variable *`keep_warnings`* to `True` so warnings will be displayed in the generated output.
- Set the configuration variable *`nitpicky`* to `True` so that Sphinx will complain about references without a known target.
- Set the debugging options in the Docutils configuration file[578].
- JavaScript stemming algorithms in `sphinx/search/*.py` (except `en.py`) are generated by this modified snowballcode generator[579]. Generated JSX[580] files are in this repository[581]. You can get the resulting JavaScript files using the following command:

```
npm install
node_modules/.bin/grunt build # -> dest/*.global.js
```

[576] http://babel.pocoo.org/en/latest/
[577] https://www.transifex.com/sphinx-doc/sphinx-1/
[578] http://docutils.sourceforge.net/docs/user/config.html
[579] https://github.com/shibukawa/snowball
[580] https://jsx.github.io/
[581] https://github.com/shibukawa/snowball-stemmer.jsx

7.2 Sphinx's release process

Branch Model

Sphinx project uses following branches for developing that conforms to Semantic Versioning 2.0.0 (refs: https://semver.org/).

`master` Development for MAJOR version. All changes including incompatible behaviors and public API updates are allowed.

`A.x` (ex. `2.x`) Where `A.x` is the `MAJOR.MINOR` release. Used to maintain current MINOR release. All changes are allowed if the change preserves backwards-compatibility of API and features.

Only the most recent `MAJOR.MINOR` branch is currently retained. When a new MAJOR version is released, the old `MAJOR.MINOR` branch will be deleted and replaced by an equivalent tag.

`A.B.x` (ex. `2.4.x`) Where `A.B.x` is the `MAJOR.MINOR.PATCH` release. Only backwards-compatible bug fixes are allowed. In Sphinx project, PATCH version is used for urgent bug fix.

`MAJOR.MINOR.PATCH` branch will be branched from the `v` prefixed release tag (ex. make 2.3.1 that branched from v2.3.0) when a urgent release is needed. When new PATCH version is released, the branch will be deleted and replaced by an equivalent tag (ex. v2.3.1).

Deprecating a feature

There are a couple reasons that code in Sphinx might be deprecated:

- If a feature has been improved or modified in a backwards-incompatible way, the old feature or behavior will be deprecated.
- Sometimes Sphinx will include a backport of a Python library that's not included in a version of Python that Sphinx currently supports. When Sphinx no longer needs to support the older version of Python that doesn't include the library, the library will be deprecated in Sphinx.

As the *Deprecation policy* describes, the first release of Sphinx that deprecates a feature (`A.B`) should raise a `RemovedInSphinxXXWarning` (where `XX` is the Sphinx version where the feature will be removed) when the deprecated feature is invoked. Assuming we have good test coverage, these warnings are converted to errors when running the test suite with warnings enabled:

```
pytest -Wall
```

Thus, when adding a `RemovedInSphinxXXWarning` you need to eliminate or silence any warnings generated when running the tests.

Deprecation policy

MAJOR and MINOR releases may deprecate certain features from previous releases. If a feature is deprecated in a release A.x, it will continue to work in all A.x.x versions (for all versions of x). It will continue to work in all B.x.x versions but raise deprecation warnings. Deprecated features will be removed at the C.0.0. It means the deprecated feature will work during 2 MAJOR releases at least.

So, for example, if we decided to start the deprecation of a function in Sphinx 2.x:

- Sphinx 2.x will contain a backwards-compatible replica of the function which will raise a `RemovedInSphinx40Warning`. This is a subclass of `PendingDeprecationWarning`[582], i.e. it will not get displayed by default.

[582] https://docs.python.org/3/library/exceptions.html#PendingDeprecationWarning

- Sphinx 3.x will still contain the backwards-compatible replica, but `RemovedInSphinx40Warning` will be a subclass of `DeprecationWarning`[583] then, and gets displayed by default.
- Sphinx 4.0 will remove the feature outright.

Deprecation warnings

Sphinx will enable its `RemovedInNextVersionWarning` warnings by default, if `PYTHONWARNINGS`[584] is not set. Therefore you can disable them using:

- `PYTHONWARNINGS= make html` (Linux/Mac)
- `export PYTHONWARNINGS=` and do `make html` (Linux/Mac)
- `set PYTHONWARNINGS=` and do `make html` (Windows)

But you can also explicitly enable the pending ones using e.g. `PYTHONWARNINGS=default` (see the Python docs on configuring warnings[585]) for more details.

Release procedures

The release procedures are listed in `utils/release-checklist`.

7.3 Organization of the Sphinx project

The guide explains how the Sphinx project is organized.

Core developers

The core developers of Sphinx have write access to the main repository. They can commit changes, accept/reject pull requests, and manage items on the issue tracker.

Guidelines

The following are some general guidelines for core developers:

- Questionable or extensive changes should be submitted as a pull request instead of being committed directly to the main repository. The pull request should be reviewed by another core developer before it is merged.
- Trivial changes can be committed directly but be sure to keep the repository in a good working state and that all tests pass before pushing your changes.
- When committing code written by someone else, please attribute the original author in the commit message and any relevant `CHANGES` entry.

[583] https://docs.python.org/3/library/exceptions.html#DeprecationWarning
[584] https://docs.python.org/3/using/cmdline.html#envvar-PYTHONWARNINGS
[585] https://docs.python.org/3/library/warnings.html#describing-warning-filters

Membership

Core membership is predicated on continued active contribution to the project. In general, prospective cores should demonstrate:

- a good understanding of one of more components of Sphinx
- a history of helpful, constructive contributions
- a willingness to invest time improving Sphinx

Refer to *Contributing to Sphinx* for more information on how you can get started.

Other contributors

You do not need to be a core developer or have write access to be involved in the development of Sphinx. You can submit patches or create pull requests from forked repositories and have a core developer add the changes for you.

Similarly, contributions are not limited to code patches. We also welcome help triaging bugs, input on design decisions, reviews of existing patches and documentation improvements. More information can be found in *Contributing to Sphinx*.

A list of people that have contributed to Sphinx can be found in *Sphinx authors*.

7.4 Sphinx Code of Conduct

Like the technical community as a whole, the Sphinx team and community is made up of volunteers from all over the world. Diversity is a strength, but it can also lead to communication issues and unhappiness. To that end, we have a few ground rules that we ask people to adhere to.

- **Be friendly and patient.**
- **Be welcoming.** We strive to be a community that welcomes and supports people of all backgrounds and identities. This includes, but is not limited to members of any race, ethnicity, culture, national origin, colour, immigration status, social and economic class, educational level, sex, sexual orientation, gender identity and expression, age, size, family status, political belief, religion, and mental and physical ability.
- **Be considerate.** Your work will be used by other people, and you in turn will depend on the work of others. Any decision you take will affect users and colleagues, and you should take those consequences into account when making decisions. Remember that we're a world-wide community, so you might not be communicating in someone else's primary language.
- **Be respectful.** Not all of us will agree all the time, but disagreement is no excuse for poor behavior and poor manners. We might all experience some frustration now and then, but we cannot allow that frustration to turn into a personal attack. It's important to remember that a community where people feel uncomfortable or threatened is not a productive one. Members of the Sphinx community should be respectful when dealing with other members as well as with people outside the Sphinx community.
- **Be careful in the words that you choose.** We are a community of professionals, and we conduct ourselves professionally. Be kind to others. Do not insult or put down other participants. Harassment and other exclusionary behavior aren't acceptable. This includes, but is not limited to:
  - Violent threats or language directed against another person.
  - Discriminatory jokes and language.
  - Posting sexually explicit or violent material.
  - Posting (or threatening to post) other people's personally identifying information ("doxing").
  - Personal insults, especially those using racist or sexist terms.

  - Unwelcome sexual attention.
  - Advocating for, or encouraging, any of the above behavior.
  - Repeated harassment of others. In general, if someone asks you to stop, then stop.
- **When we disagree, try to understand why.** Disagreements, both social and technical, happen all the time and Sphinx is no exception. It is important that we resolve disagreements and differing views constructively. Remember that we're different. Different people have different perspectives on issues. Being unable to understand why someone holds a viewpoint doesn't mean that they're wrong. Don't forget that it is human to err and blaming each other doesn't get us anywhere. Instead, focus on helping to resolve issues and learning from mistakes.

This isn't an exhaustive list of things that you can't do. Rather, take it in the spirit in which it's intended - a guide to make it easier to enrich all of us and the technical communities in which we participate. This code of conduct applies to all spaces of the Sphinx community.

Attribution

Original text courtesy of the Speak Up! project: http://web.archive.org/web/20141109123859/http://speakup.io/coc.html.

7.5 Sphinx authors

Sphinx is written and maintained by Georg Brandl <georg@python.org>.

Substantial parts of the templates were written by Armin Ronacher <armin.ronacher@active-4.com>.

Other co-maintainers:

- Takayuki Shimizukawa <shimizukawa@gmail.com>
- Daniel Neuhäuser <@DasIch>
- Jon Waltman <@jonwaltman>
- Rob Ruana <@RobRuana>
- Robert Lehmann <@lehmannro>
- Roland Meister <@rolmei>
- Takeshi Komiya <@tk0miya>
- Jean-François Burnol <@jfbu>
- Yoshiki Shibukawa <@shibu_jp>
- Timotheus Kampik - <@TimKam>

Other contributors, listed alphabetically, are:

- Alastair Houghton – Apple Help builder
- Alexander Todorov – inheritance_diagram tests and improvements
- Andi Albrecht – agogo theme
- Jakob Lykke Andersen – Rewritten C++ domain
- Henrique Bastos – SVG support for graphviz extension
- Daniel Bültmann – todo extension
- Marco Buttu – doctest extension (pyversion option)
- Nathan Damon – bugfix in validation of static paths in html builders

- Etienne Desautels – apidoc module
- Michael Droettboom – inheritance_diagram extension
- Charles Duffy – original graphviz extension
- Kevin Dunn – MathJax extension
- Josip Dzolonga – coverage builder
- Buck Evan – dummy builder
- Matthew Fernandez – todo extension fix
- Hernan Grecco – search improvements
- Horst Gutmann – internationalization support
- Martin Hans – autodoc improvements
- Zac Hatfield-Dodds – doctest reporting improvements, intersphinx performance
- Doug Hellmann – graphviz improvements
- Tim Hoffmann – theme improvements
- Antti Kaihola – doctest extension (skipif option)
- Dave Kuhlman – original LaTeX writer
- Blaise Laflamme – pyramid theme
- Chris Lamb – reproducibility fixes
- Thomas Lamb – linkcheck builder
- Łukasz Langa – partial support for autodoc
- Martin Larralde – additional napoleon admonitions
- Ian Lee – quickstart improvements
- Robert Lehmann – gettext builder (GSOC project)
- Dan MacKinlay – metadata fixes
- Martin Mahner – nature theme
- Will Maier – directory HTML builder
- Jacob Mason – websupport library (GSOC project)
- Glenn Matthews – python domain signature improvements
- Kurt McKee – documentation updates
- Roland Meister – epub builder
- Ezio Melotti – collapsible sidebar JavaScript
- Bruce Mitchener – Minor epub improvement
- Daniel Neuhäuser – JavaScript domain, Python 3 support (GSOC)
- Julien Palard – Colspan and rowspan in text builder
- Christopher Perkins – autosummary integration
- Benjamin Peterson – unittests
- T. Powers – HTML output improvements
- Jeppe Pihl – literalinclude improvements
- Rob Ruana – napoleon extension

- Vince Salvino – JavaScript search improvements
- Stefan Seefeld – toctree improvements
- Gregory Szorc – performance improvements
- Taku Shimizu – epub3 builder
- Antonio Valentino – qthelp builder, docstring inheritance
- Filip Vavera – napoleon todo directive
- Pauli Virtanen – autodoc improvements, autosummary extension
- Eric N. Vander Weele – autodoc improvements
- Stefan van der Walt – autosummary extension
- Thomas Waldmann – apidoc module fixes
- John Waltman – Texinfo builder
- Barry Warsaw – setup command improvements
- Sebastian Wiesner – image handling, distutils support
- Michael Wilson – Intersphinx HTTP basic auth support
- Matthew Woodcraft – text output improvements
- Joel Wurtz – cellspanning support in LaTeX
- Hong Xu – svg support in imgmath extension and various bug fixes
- Stephen Finucane – setup command improvements and documentation
- Daniel Pizetta – inheritance diagram improvements
- KINEBUCHI Tomohiko – typing Sphinx as well as docutils
- Adrián Chaves (Gallaecio) – coverage builder improvements
- Lars Hupfeldt Nielsen - OpenSSL FIPS mode md5 bug fix

Many thanks for all contributions!

There are also a few modules or functions incorporated from other authors and projects:

- sphinx.util.jsdump uses the basestring encoding from simplejson, written by Bob Ippolito, released under the MIT license
- sphinx.util.stemmer was written by Vivake Gupta, placed in the Public Domain

CHAPTER

EIGHT

SPHINX FAQ

This is a list of Frequently Asked Questions about Sphinx. Feel free to suggest new entries!

8.1 How do I…

… create PDF files without LaTeX? rinohtype[586] provides a PDF builder that can be used as a drop-in replacement for the LaTeX builder.

… get section numbers? They are automatic in LaTeX output; for HTML, give a `:numbered:` option to the *`toctree`* directive where you want to start numbering.

… customize the look of the built HTML files? Use themes, see *HTML Theming*.

… add global substitutions or includes? Add them in the *`rst_prolog`* or *`rst_epilog`* config value.

… display the whole TOC tree in the sidebar? Use the *`toctree`* callable in a custom layout template, probably in the `sidebartoc` block.

… write my own extension? See the *Extension tutorials*.

… convert from my existing docs using MoinMoin markup? The easiest way is to convert to xhtml, then convert xhtml to reST[587]. You'll still need to mark up classes and such, but the headings and code examples come through cleanly.

For many more extensions and other contributed stuff, see the sphinx-contrib[588] repository.

8.2 Using Sphinx with…

Read the Docs Read the Docs[589] is a documentation hosting service based around Sphinx. They will host sphinx documentation, along with supporting a number of other features including version support, PDF generation, and more. The Getting Started[590] guide is a good place to start.

Epydoc There's a third-party extension providing an api role[591] which refers to Epydoc's API docs for a given identifier.

Doxygen Michael Jones is developing a reST/Sphinx bridge to doxygen called breathe[592].

SCons Glenn Hutchings has written a SCons build script to build Sphinx documentation; it is hosted here: https://bitbucket.org/zondo/sphinx-scons

[586] https://github.com/brechtm/rinohtype
[587] http://docutils.sourceforge.net/sandbox/xhtml2rest/xhtml2rest.py
[588] https://bitbucket.org/birkenfeld/sphinx-contrib/
[589] https://readthedocs.org
[590] https://docs.readthedocs.io/en/stable/intro/getting-started-with-sphinx.html
[591] https://git.savannah.gnu.org/cgit/kenozooid.git/tree/doc/extapi.py
[592] https://github.com/michaeljones/breathe/tree/master

PyPI Jannis Leidel wrote a setuptools command[593] that automatically uploads Sphinx documentation to the PyPI package documentation area at https://pythonhosted.org/.

GitHub Pages Please add *sphinx.ext.githubpages* to your project. It allows you to publish your document in GitHub Pages. It generates helper files for GitHub Pages on building HTML document automatically.

MediaWiki See https://bitbucket.org/kevindunn/sphinx-wiki/wiki/Home, a project by Kevin Dunn.

Google Analytics You can use a custom `layout.html` template, like this:

```
{% extends "!layout.html" %}

{%- block extrahead %}
{{ super() }}
<script>
  var _gaq = _gaq || [];
  _gaq.push(['_setAccount', 'XXX account number XXX']);
  _gaq.push(['_trackPageview']);
</script>
{% endblock %}

{% block footer %}
{{ super() }}
<div class="footer">This page uses <a href="https://analytics.google.com/">
Google Analytics</a> to collect statistics. You can disable it by blocking
the JavaScript coming from www.google-analytics.com.
<script>
  (function() {
    var ga = document.createElement('script');
    ga.src = ('https:' == document.location.protocol ?
              'https://ssl' : 'http://www') + '.google-analytics.com/ga.js';
    ga.setAttribute('async', 'true');
    document.documentElement.firstChild.appendChild(ga);
  })();
</script>
</div>
{% endblock %}
```

Google Search To replace Sphinx's built-in search function with Google Search, proceed as follows:

1. Go to https://cse.google.com/cse/all to create the Google Search code snippet.
2. Copy the code snippet and paste it into `_templates/searchbox.html` in your Sphinx project:

```
<div>
   <h3>{{ _('Quick search') }}</h3>
   <script>
      (function() {
         var cx = '......';
         var gcse = document.createElement('script');
         gcse.async = true;
         gcse.src = 'https://cse.google.com/cse.js?cx=' + cx;
         var s = document.getElementsByTagName('script')[0];
         s.parentNode.insertBefore(gcse, s);
      })();
```

(continues on next page)

[593] https://pypi.org/project/Sphinx-PyPI-upload/

(continued from previous page)

```
  </script>
  <gcse:search></gcse:search>
</div>
```

3. Add `searchbox.html` to the *html_sidebars* configuration value.

8.3 Sphinx vs. Docutils

tl;dr: *docutils* converts reStructuredText to multiple output formats. Sphinx builds upon docutils to allow construction of cross-referenced and indexed bodies of documentation.

docutils[594] is a text processing system for converting plain text documentation into other, richer formats. As noted in the docutils documentation[595], docutils uses *readers* to read a document, *parsers* for parsing plain text formats into an internal tree representation made up of different types of *nodes*, and *writers* to output this tree in various document formats. docutils provides parsers for one plain text format - reStructuredText[596] - though other, *out-of-tree* parsers have been implemented including Sphinx's *Markdown parser*. On the other hand, it provides writers for many different formats including HTML, LaTeX, man pages, Open Document Format and XML.

docutils exposes all of its functionality through a variety of front-end tools[597], such as `rst2html`, `rst2odt` and `rst2xml`. Crucially though, all of these tools, and docutils itself, are concerned with individual documents. They don't support concepts such as cross-referencing, indexing of documents, or the construction of a document hierarchy (typically manifesting in a table of contents).

Sphinx builds upon docutils by harnessing docutils' readers and parsers and providing its own *Builders*. As a result, Sphinx wraps some of the *writers* provided by docutils. This allows Sphinx to provide many features that would simply not be possible with docutils, such as those outlined above.

8.4 Epub info

The following list gives some hints for the creation of epub files:

- Split the text into several files. The longer the individual HTML files are, the longer it takes the ebook reader to render them. In extreme cases, the rendering can take up to one minute.
- Try to minimize the markup. This also pays in rendering time.
- For some readers you can use embedded or external fonts using the CSS `@font-face` directive. This is *extremely* useful for code listings which are often cut at the right margin. The default Courier font (or variant) is quite wide and you can only display up to 60 characters on a line. If you replace it with a narrower font, you can get more characters on a line. You may even use FontForge[598] and create narrow variants of some free font. In my case I get up to 70 characters on a line.

  You may have to experiment a little until you get reasonable results.
- Test the created epubs. You can use several alternatives. The ones I am aware of are Epubcheck[599], Calibre[600], FBreader[601] (although it does not render the CSS), and Bookworm[602]. For Bookworm, you can download the source from https://code.google.com/archive/p/threepress and run your own local server.

[594] http://docutils.sourceforge.io/
[595] http://docutils.sourceforge.io/docs/dev/hacking.html
[596] http://docutils.sourceforge.io/rst.html
[597] http://docutils.sourceforge.net/docs/user/tools.html
[598] https://fontforge.github.io/
[599] https://github.com/IDPF/epubcheck
[600] https://calibre-ebook.com/
[601] https://fbreader.org/
[602] https://www.oreilly.com/bookworm/index.html

- Large floating divs are not displayed properly. If they cover more than one page, the div is only shown on the first page. In that case you can copy the `epub.css` from the `sphinx/themes/epub/static/` directory to your local `_static/` directory and remove the float settings.
- Files that are inserted outside of the `toctree` directive must be manually included. This sometimes applies to appendixes, e.g. the glossary or the indices. You can add them with the *epub_post_files* option.
- The handling of the epub cover page differs from the reStructuredText procedure which automatically resolves image paths and puts the images into the `_images` directory. For the epub cover page put the image in the *html_static_path* directory and reference it with its full path in the *epub_cover* config option.
- kindlegen[603] command can convert from epub3 resulting file to `.mobi` file for Kindle. You can get `yourdoc.mobi` under `_build/epub` after the following command:

  ```
  $ make epub
  $ kindlegen _build/epub/yourdoc.epub
  ```

  The kindlegen command doesn't accept documents that have section titles surrounding `toctree` directive:

  ```
  Section Title
  =============

  .. toctree::

     subdocument

  Section After Toc Tree
  ======================
  ```

  kindlegen assumes all documents order in line, but the resulting document has complicated order for kindlegen:

  ```
  ``parent.xhtml`` -> ``child.xhtml`` -> ``parent.xhtml``
  ```

  If you get the following error, fix your document structure:

  ```
  Error(prcgen):E24011: TOC section scope is not included in the parent␣
  ↪chapter:(title)
  Error(prcgen):E24001: The table of content could not be built.
  ```

8.5 Texinfo info

There are two main programs for reading Info files, `info` and GNU Emacs. The `info` program has less features but is available in most Unix environments and can be quickly accessed from the terminal. Emacs provides better font and color display and supports extensive customization (of course).

[603] https://www.amazon.com/gp/feature.html?docId=1000765211

Displaying Links

One noticeable problem you may encounter with the generated Info files is how references are displayed. If you read the source of an Info file, a reference to this section would look like:

```
* note Displaying Links: target-id
```

In the stand-alone reader, `info`, references are displayed just as they appear in the source. Emacs, on the other-hand, will by default replace `*note:` with `see` and hide the `target-id`. For example:

> *Displaying Links*

The exact behavior of how Emacs displays references is dependent on the variable `Info-hide-note-references`. If set to the value of `hide`, Emacs will hide both the `*note:` part and the `target-id`. This is generally the best way to view Sphinx-based documents since they often make frequent use of links and do not take this limitation into account. However, changing this variable affects how all Info documents are displayed and most do take this behavior into account.

If you want Emacs to display Info files produced by Sphinx using the value `hide` for `Info-hide-note-references` and the default value for all other Info files, try adding the following Emacs Lisp code to your start-up file, `~/.emacs.d/init.el`.

```
(defadvice info-insert-file-contents (after
                                      sphinx-info-insert-file-contents
                                      activate)
  "Hack to make `Info-hide-note-references' buffer-local and
automatically set to `hide' iff it can be determined that this file
was created from a Texinfo file generated by Docutils or Sphinx."
  (set (make-local-variable 'Info-hide-note-references)
       (default-value 'Info-hide-note-references))
  (save-excursion
    (save-restriction
      (widen) (goto-char (point-min))
      (when (re-search-forward
             "^Generated by \\(Sphinx\\|Docutils\\)"
             (save-excursion (search-forward "\x1f" nil t)) t)
        (set (make-local-variable 'Info-hide-note-references)
             'hide)))))
```

Notes

The following notes may be helpful if you want to create Texinfo files:

- Each section corresponds to a different `node` in the Info file.
- Colons (`:`) cannot be properly escaped in menu entries and xrefs. They will be replaced with semicolons (`;`).
- Links to external Info files can be created using the somewhat official URI scheme `info`. For example:

  ```
  info:Texinfo#makeinfo_options
  ```

- Inline markup

  The standard formatting for `*strong*` and `_emphasis_` can result in ambiguous output when used to markup parameter names and other values. Since this is a fairly common practice, the default formatting has been changed so that `emphasis` and `strong` are now displayed like `` `literal' ``s.

  The standard formatting can be re-enabled by adding the following to your `conf.py`:

```
texinfo_elements = {'preamble': """
@definfoenclose strong,*,*
@definfoenclose emph,_,_
"""}
```

CHAPTER NINE

GLOSSARY

builder A class (inheriting from `Builder`) that takes parsed documents and performs an action on them. Normally, builders translate the documents to an output format, but it is also possible to use builders that e.g. check for broken links in the documentation, or build coverage information.

See *Builders* for an overview over Sphinx's built-in builders.

configuration directory The directory containing `conf.py`. By default, this is the same as the *source directory*, but can be set differently with the **-c** command-line option.

directive A reStructuredText markup element that allows marking a block of content with special meaning. Directives are supplied not only by docutils, but Sphinx and custom extensions can add their own. The basic directive syntax looks like this:

```
.. directivename:: argument ...
   :option: value

   Content of the directive.
```

See *Directives* for more information.

document name Since reST source files can have different extensions (some people like `.txt`, some like `.rst` – the extension can be configured with `source_suffix`) and different OSes have different path separators, Sphinx abstracts them: *document names* are always relative to the *source directory*, the extension is stripped, and path separators are converted to slashes. All values, parameters and such referring to "documents" expect such document names.

Examples for document names are `index`, `library/zipfile`, or `reference/datamodel/types`. Note that there is no leading or trailing slash.

domain A domain is a collection of markup (reStructuredText *directive*s and *role*s) to describe and link to *object*s belonging together, e.g. elements of a programming language. Directive and role names in a domain have names like `domain:name`, e.g. `py:function`.

Having domains means that there are no naming problems when one set of documentation wants to refer to e.g. C++ and Python classes. It also means that extensions that support the documentation of whole new languages are much easier to write.

For more information, refer to *Domains*.

environment A structure where information about all documents under the root is saved, and used for cross-referencing. The environment is pickled after the parsing stage, so that successive runs only need to read and parse new and changed documents.

extension A custom *role*, *directive* or other aspect of Sphinx that allows users to modify any aspect of the build process within Sphinx.

For more information, refer to *Extensions*.

master document The document that contains the root `toctree` directive.

object The basic building block of Sphinx documentation. Every "object directive" (e.g. `function` or *`object`*) creates such a block; and most objects can be cross-referenced to.

RemoveInSphinxXXXWarning The feature which is warned will be removed in Sphinx-XXX version. It usually caused from Sphinx extensions which is using deprecated. See also *Deprecation Warnings*.

role A reStructuredText markup element that allows marking a piece of text. Like directives, roles are extensible. The basic syntax looks like this: ``:rolename:`content` ``. See *Inline markup* for details.

source directory The directory which, including its subdirectories, contains all source files for one Sphinx project.

reStructuredText An easy-to-read, what-you-see-is-what-you-get plaintext markup syntax and parser system.

CHAPTER

TEN

CHANGELOG

10.1 Release 4.0.0 (in development)

Dependencies

- Drop python 3.5 support
- Drop docutils 0.12 and 0.13 support
- LaTeX: add `tex-gyre` font dependency

Incompatible changes

- domain: The `Index` class becomes subclasses of `abc.ABC` to indicate methods that must be overrided in the concrete classes
- #4826: py domain: The structure of python objects is changed. A boolean value is added to indicate that the python object is canonical one
- #7425: MathJax: The MathJax was changed from 2 to 3. Users using a custom MathJax configuration may have to set the old MathJax path or update their configuration for version 3. See *sphinx.ext.mathjax*.
- #7784: i18n: The msgid for alt text of image is changed
- #5560: napoleon: *napoleon_use_param* also affect "other parameters" section
- #7996: manpage: Make a section directory on build manpage by default (see *man_make_section_directory*)
- #7849: html: Change the default setting of *html_codeblock_linenos_style* to `'inline'`
- #8380: html search: search results are wrapped with <p> instead of <div>
- html theme: Move a script tag for documentation_options.js in basic/layout.html to `script_files` variable
- html theme: Move CSS tags in basic/layout.html to `css_files` variable
- #8508: LaTeX: uplatex becomes a default setting of latex_engine for Japanese documents
- #5977: py domain: `:var:`, `:cvar:` and `:ivar:` fields do not create cross-references
- #4550: The `align` attribute of `figure` and `table` nodes becomes `None` by default instead of `'default'`
- #8769: LaTeX refactoring: split sphinx.sty into multiple files and rename some auxiliary files created in `latex` build output repertory

Deprecated

- *html_codeblock_linenos_style*
- `favicon` and `logo` variable in HTML templates
- `sphinx.directives.patches.CSVTable`
- `sphinx.directives.patches.ListTable`
- `sphinx.directives.patches.RSTTable`
- `sphinx.transforms.FigureAligner`
- `sphinx.util.pycompat.convert_with_2to3()`
- `sphinx.util.pycompat.execfile_()`
- `sphinx.util.smartypants`

Features added

- #4826: py domain: Add `:canonical:` option to python directives to describe the location where the object is defined
- #7784: i18n: The alt text for image is translated by default (without *gettext_additional_targets* setting)
- #2018: html: *html_favicon* and *html_logo* now accept URL for the image
- #8070: html search: Support searching for 2characters word
- #7830: Add debug logs for change detection of sources and templates
- #8201: Emit a warning if toctree contains duplicated entries

Bugs fixed

- #8380: html search: Paragraphs in search results are not identified as <p>
- #8342: Emit a warning if a unknown domain is given for directive or role (ex. `:unknown:doc:`)
- #8711: LaTeX: backticks in code-blocks trigger latexpdf build warning (and font change) with late TeXLive 2019
- #8253: LaTeX: Figures with no size defined get overscaled (compared to images with size explicitly set in pixels) (fixed for `'pdflatex'`/`'lualatex'` only)

Testing

10.2 Release 3.5.2 (in development)

Dependencies

Incompatible changes

Deprecated

Features added

Bugs fixed

Testing

10.3 Release 3.5.1 (released Feb 16, 2021)

Bugs fixed

- #8883: autodoc: AttributeError is raised on assigning __annotations__ on read-only class
- #8884: html: minified js stemmers not included in the distributed package
- #8885: html: AttributeError is raised if CSS/JS files are installed via *html_context*
- #8880: viewcode: ExtensionError is raised on incremental build after unparsable python module found

10.4 Release 3.5.0 (released Feb 14, 2021)

Dependencies

- LaTeX: `multicol` (it is anyhow a required part of the official latex2e base distribution)

Incompatible changes

- Update Underscore.js to 1.12.0
- #6550: html: The config variable `html_add_permalinks` is replaced by *html_permalinks* and *html_permalinks_icon*

Deprecated

- pending_xref node for viewcode extension
- `sphinx.builders.linkcheck.CheckExternalLinksBuilder.anchors_ignore`
- `sphinx.builders.linkcheck.CheckExternalLinksBuilder.auth`
- `sphinx.builders.linkcheck.CheckExternalLinksBuilder.broken`
- `sphinx.builders.linkcheck.CheckExternalLinksBuilder.good`
- `sphinx.builders.linkcheck.CheckExternalLinksBuilder.redirected`
- `sphinx.builders.linkcheck.CheckExternalLinksBuilder.rqueue`
- `sphinx.builders.linkcheck.CheckExternalLinksBuilder.to_ignore`
- `sphinx.builders.linkcheck.CheckExternalLinksBuilder.workers`
- `sphinx.builders.linkcheck.CheckExternalLinksBuilder.wqueue`
- `sphinx.builders.linkcheck.node_line_or_0()`
- `sphinx.ext.autodoc.AttributeDocumenter.isinstanceattribute()`
- `sphinx.ext.autodoc.directive.DocumenterBridge.reporter`
- `sphinx.ext.autodoc.importer.get_module_members()`
- `sphinx.ext.autosummary.generate._simple_info()`

- `sphinx.ext.autosummary.generate._simple_warn()`
- `sphinx.writers.html.HTMLTranslator.permalink_text`
- `sphinx.writers.html5.HTML5Translator.permalink_text`

Features added

- #8022: autodoc: autodata and autoattribute directives does not show right-hand value of the variable if docstring contains `:meta hide-value:` in info-field-list
- #8514: autodoc: Default values of overloaded functions are taken from actual implementation if they're ellipsis
- #8775: autodoc: Support type union operator (PEP-604) in Python 3.10 or above
- #8297: autodoc: Allow to extend *`autodoc_default_options`* via directive options
- #8619: html: kbd role generates customizable HTML tags for compound keys
- #8634: html: Allow to change the order of JS/CSS via `priority` parameter for `Sphinx.add_js_file()` and `Sphinx.add_css_file()`
- #6241: html: Allow to add JS/CSS files to the specific page when an extension calls `app.add_js_file()` or `app.add_css_file()` on *`html-page-context`* event
- #6550: html: Allow to use HTML permalink texts via *`html_permalinks_icon`*
- #1638: html: Add permalink icons to glossary terms
- #8868: html search: performance issue with massive lists
- #8867: html search: Update JavaScript stemmer code to the latest version of Snowball (v2.1.0)
- #8852: i18n: Allow to translate heading syntax in MyST-Parser
- #8649: imgconverter: Skip availability check if builder supports the image type
- #8573: napoleon: Allow to change the style of custom sections using `napoleon_custom_styles`
- #8004: napoleon: Type definitions in Google style docstrings are rendered as references when `napoleon_preprocess_types` enabled
- #6241: mathjax: Include mathjax.js only on the document using equations
- #8775: py domain: Support type union operator (PEP-604)
- #8651: std domain: cross-reference for a rubric having inline item is broken
- #7642: std domain: Optimize case-insensitive match of term
- #8681: viewcode: Support incremental build
- #8132: Add *`project_copyright`* as an alias of *`copyright`*
- #207: Now *`highlight_language`* supports multiple languages
- #2030: *`code-block`* and *`literalinclude`* supports automatic dedent via no-argument `:dedent:` option
- C++, also hyperlink operator overloads in expressions and alias declarations.
- #8247: Allow production lists to refer to tokens from other production groups
- #8813: Show what extension (or module) caused it on errors on event handler
- #8213: C++: add `maxdepth` option to *`cpp:alias`* to insert nested declarations.
- C, add `noroot` option to *`c:alias`* to render only nested declarations.
- C++, add `noroot` option to *`cpp:alias`* to render only nested declarations.

Bugs fixed

- #8727: apidoc: namespace module file is not generated if no submodules there
- #741: autodoc: inherited-members doesn't work for instance attributes on super class
- #8592: autodoc: `:meta public:` does not effect to variables
- #8594: autodoc: empty __all__ attribute is ignored
- #8315: autodoc: Failed to resolve struct.Struct type annotation
- #8652: autodoc: All variable comments in the module are ignored if the module contains invalid type comments
- #8693: autodoc: Default values for overloaded functions are rendered as string
- #8134: autodoc: crashes when mocked decorator takes arguments
- #8800: autodoc: Uninitialized attributes in superclass are recognized as undocumented
- #8655: autodoc: Failed to generate document if target module contains an object that raises an exception on `hasattr()`
- #8306: autosummary: mocked modules are documented as empty page when using :recursive: option
- #8232: graphviz: Image node is not rendered if graph file is in subdirectory
- #8618: html: kbd role produces incorrect HTML when compound-key separators (-, + or ^) are used as keystrokes
- #8629: html: A type warning for html_use_opensearch is shown twice
- #8714: html: kbd role with "Caps Lock" rendered incorrectly
- #8123: html search: fix searching for terms containing + (Requires a custom search language that does not split on +)
- #8665: html theme: Could not override globaltoc_maxdepth in theme.conf
- #8446: html: consecutive spaces are displayed as single space
- #8745: i18n: crashes with KeyError when translation message adds a new auto footnote reference
- #4304: linkcheck: Fix race condition that could lead to checking the availability of the same URL twice
- #8791: linkcheck: The docname for each hyperlink is not displayed
- #7118: sphinx-quickstart: questionare got Mojibake if libreadline unavailable
- #8094: texinfo: image files on the different directory with document are not copied
- #8782: todo: Cross references in todolist get broken
- #8720: viewcode: module pages are generated for epub on incremental build
- #8704: viewcode: anchors are generated in incremental build after singlehtml
- #8756: viewcode: highlighted code is generated even if not referenced
- #8671: *`highlight_options`* is not working
- #8341: C, fix intersphinx lookup types for names in declarations.
- C, C++: in general fix intersphinx and role lookup types.
- #8683: *`html_last_updated_fmt`* does not support UTC offset (%z)
- #8683: *`html_last_updated_fmt`* generates wrong time zone for %Z
- #1112: `download` role creates duplicated copies when relative path is specified
- #2616 (fifth item): LaTeX: footnotes from captions are not clickable, and for manually numbered footnotes only first one with same number is an hyperlink

- #7576: LaTeX with French babel and memoir crash: "Illegal parameter number in definition of `\FNH@prefntext`"
- #8055: LaTeX (docs): A potential display bug with the LaTeX generation step in Sphinx (how to generate one-column index)
- #8072: LaTeX: Directive `hlist` not implemented in LaTeX
- #8214: LaTeX: The `index` role and the glossary generate duplicate entries in the LaTeX index (if both used for same term)
- #8735: LaTeX: wrong internal links in pdf to captioned code-blocks when `numfig` is not True
- #8442: LaTeX: some indexed terms are ignored when using xelatex engine (or pdflatex and `latex_use_xindy` set to True) with memoir class
- #8750: LaTeX: URLs as footnotes fail to show in PDF if originating from inside function type signatures
- #8780: LaTeX: long words in narrow columns may not be hyphenated
- #8788: LaTeX: `\titleformat` last argument in sphinx.sty should be bracketed, not braced (and is anyhow not needed)
- #8849: LaTex: code-block printed out of margin (see the opt-in LaTeX syntax boolean *verbatimforcewraps* for use via the *'sphinxsetup'* key of `latex_elements`)
- #8183: LaTeX: Remove substitution_reference nodes from doctree only on LaTeX builds
- #8865: LaTeX: Restructure the index nodes inside title nodes only on LaTeX builds
- #8796: LaTeX: potentially critical low level TeX coding mistake has gone unnoticed so far
- C, `c:alias` skip symbols without explicit declarations instead of crashing.
- C, `c:alias` give a warning when the root symbol is not declared.
- C, `expr` role should start symbol lookup in the current scope.

10.5 Release 3.4.3 (released Jan 08, 2021)

Bugs fixed

- #8655: autodoc: Failed to generate document if target module contains an object that raises an exception on `hasattr()`

10.6 Release 3.4.2 (released Jan 04, 2021)

Bugs fixed

- #8164: autodoc: Classes that inherit mocked class are not documented
- #8602: autodoc: The `autodoc-process-docstring` event is emitted to the non-datadescriptors unexpectedly
- #8616: autodoc: AttributeError is raised on non-class object is passed to autoclass directive

10.7 Release 3.4.1 (released Dec 25, 2020)

Bugs fixed

- #8559: autodoc: AttributeError is raised when using forward-reference type annotations
- #8568: autodoc: TypeError is raised on checking slots attribute
- #8567: autodoc: Instance attributes are incorrectly added to Parent class
- #8566: autodoc: The `autodoc-process-docstring` event is emitted to the alias classes unexpectedly
- #8583: autodoc: Unnecessary object comparision via `__eq__` method
- #8565: linkcheck: Fix PriorityQueue crash when link tuples are not comparable

10.8 Release 3.4.0 (released Dec 20, 2020)

Incompatible changes

- #8105: autodoc: the signature of class constructor will be shown for decorated classes, not a signature of decorator

Deprecated

- The `follow_wrapped` argument of `sphinx.util.inspect.signature()`
- The `no_docstring` argument of `sphinx.ext.autodoc.Documenter.add_content()`
- `sphinx.ext.autodoc.Documenter.get_object_members()`
- `sphinx.ext.autodoc.DataDeclarationDocumenter`
- `sphinx.ext.autodoc.GenericAliasDocumenter`
- `sphinx.ext.autodoc.InstanceAttributeDocumenter`
- `sphinx.ext.autodoc.SlotsAttributeDocumenter`
- `sphinx.ext.autodoc.TypeVarDocumenter`
- `sphinx.ext.autodoc.importer._getannotations()`
- `sphinx.ext.autodoc.importer._getmro()`
- `sphinx.pycode.ModuleAnalyzer.parse()`
- `sphinx.util.osutil.movefile()`
- `sphinx.util.requests.is_ssl_error()`

Features added

- #8119: autodoc: Allow to determine whether a member not included in `__all__` attribute of the module should be documented or not via *autodoc-skip-member* event
- #8219: autodoc: Parameters for generic class are not shown when super class is a generic class and show-inheritance option is given (in Python 3.7 or above)
- autodoc: Add `Documenter.config` as a shortcut to access the config object
- autodoc: Add Optional[t] to annotation of function and method if a default value equal to None is set.
- #8209: autodoc: Add `:no-value:` option to *autoattribute* and *autodata* directive to suppress the default value of the variable
- #8460: autodoc: Support custom types defined by typing.NewType
- #8285: napoleon: Add *napoleon_attr_annotations* to merge type hints on source code automatically if any type is specified in docstring
- #8236: napoleon: Support numpydoc's "Receives" section
- #6914: Add a new event *warn-missing-reference* to custom warning messages when failed to resolve a cross-reference
- #6914: Emit a detailed warning when failed to resolve a `:ref:` reference
- #6629: linkcheck: The builder now handles rate limits. See `linkcheck_retry_on_rate_limit` for details.

Bugs fixed

- #7613: autodoc: autodoc does not respect __signature__ of the class
- #4606: autodoc: the location of the warning is incorrect for inherited method
- #8105: autodoc: the signature of class constructor is incorrect if the class is decorated
- #8434: autodoc: *autodoc_type_aliases* does not effect to variables and attributes
- #8443: autodoc: autodata directive can't create document for PEP-526 based type annotated variables
- #8443: autodoc: autoattribute directive can't create document for PEP-526 based uninitalized variables
- #8480: autodoc: autoattribute could not create document for __slots__ attributes
- #8503: autodoc: autoattribute could not create document for a GenericAlias as class attributes correctly
- #8534: autodoc: autoattribute could not create document for a commented attribute in alias class
- #8452: autodoc: autodoc_type_aliases doesn't work when autodoc_typehints is set to "description"
- #8541: autodoc: autodoc_type_aliases doesn't work for the type annotation to instance attributes
- #8460: autodoc: autodata and autoattribute directives do not display type information of TypeVars
- #8493: autodoc: references to builtins not working in class aliases
- #8522: autodoc: `__bool__` method could be called
- #8067: autodoc: A typehint for the instance variable having type_comment on super class is not displayed
- #8545: autodoc: a __slots__ attribute is not documented even having docstring
- #741: autodoc: inherited-members doesn't work for instance attributes on super class
- #8477: autosummary: non utf-8 reST files are generated when template contains multibyte characters
- #8501: autosummary: summary extraction splits text after "el at." unexpectedly
- #8524: html: Wrong url_root has been generated on a document named "index"

- #8419: html search: Do not load `language_data.js` in non-search pages
- #8549: i18n: `-D gettext_compact=0` is no longer working
- #8454: graphviz: The layout option for graph and digraph directives don't work
- #8131: linkcheck: Use GET when HEAD requests cause Too Many Redirects, to accommodate infinite redirect loops on HEAD
- #8437: Makefile: `make clean` with empty BUILDDIR is dangerous
- #8365: py domain: `:type:` and `:rtype:` gives false ambiguous class lookup warnings
- #8352: std domain: Failed to parse an option that starts with bracket
- #8519: LaTeX: Prevent page brake in the middle of a seealso
- #8520: C, fix copying of AliasNode.

10.9 Release 3.3.1 (released Nov 12, 2020)

Bugs fixed

- #8372: autodoc: autoclass directive became slower than Sphinx-3.2
- #7727: autosummary: raise PycodeError when documenting python package without __init__.py
- #8350: autosummary: autosummary_mock_imports causes slow down builds
- #8364: C, properly initialize attributes in empty symbols.
- #8399: i18n: Put system locale path after the paths specified by configuration

10.10 Release 3.3.0 (released Nov 02, 2020)

Deprecated

- `sphinx.builders.latex.LaTeXBuilder.usepackages`
- `sphinx.builders.latex.LaTeXBuilder.usepackages_afger_hyperref`
- `sphinx.ext.autodoc.SingledispatchFunctionDocumenter`
- `sphinx.ext.autodoc.SingledispatchMethodDocumenter`

Features added

- #8100: html: Show a better error message for failures on copying html_static_files
- #8141: C: added a `maxdepth` option to *`c:alias`* to insert nested declarations.
- #8081: LaTeX: Allow to add LaTeX package via `app.add_latex_package()` until just before writing .tex file
- #7996: manpage: Add *`man_make_section_directory`* to make a section directory on build man page
- #8289: epub: Allow to suppress "duplicated ToC entry found" warnings from epub builder using *`suppress_warnings`*.
- #8298: sphinx-quickstart: Add *`sphinx-quickstart --no-sep`* option
- #8304: sphinx.testing: Register public markers in sphinx.testing.fixtures
- #8051: napoleon: use the obj role for all See Also items

- #8050: napoleon: Apply `napoleon_preprocess_types` to every field
- C and C++, show line numbers for previous declarations when duplicates are detected.
- #8183: Remove substitution_reference nodes from doctree only on LaTeX builds

Bugs fixed

- #8085: i18n: Add support for having single text domain
- #6640: i18n: Failed to override system message translation
- #8143: autodoc: AttributeError is raised when False value is passed to autodoc_default_options
- #8103: autodoc: functools.cached_property is not considered as a property
- #8190: autodoc: parsing error is raised if some extension replaces docstring by string not ending with blank lines
- #8142: autodoc: Wrong constructor signature for the class derived from typing.Generic
- #8157: autodoc: TypeError is raised when annotation has invalid __args__
- #7964: autodoc: Tuple in default value is wrongly rendered
- #8200: autodoc: type aliases break type formatting of autoattribute
- #7786: autodoc: can't detect overloaded methods defined in other file
- #8294: autodoc: single-string __slots__ is not handled correctly
- #7785: autodoc: autodoc_typehints='none' does not effect to overloaded functions
- #8192: napoleon: description is disappeared when it contains inline literals
- #8142: napoleon: Potential of regex denial of service in google style docs
- #8169: LaTeX: pxjahyper loaded even when latex_engine is not platex
- #8215: LaTeX: 'oneside' classoption causes build warning
- #8175: intersphinx: Potential of regex denial of service by broken inventory
- #8277: sphinx-build: missing and redundant spacing (and etc) for console output on building
- #7973: imgconverter: Check availability of imagemagick many times
- #8255: py domain: number in default argument value is changed from hexadecimal to decimal
- #8316: html: Prevent arrow keys changing page when button elements are focused
- #8343: html search: Fix unnecessary load of images when parsing the document
- #8254: html theme: Line numbers misalign with code lines
- #8093: The highlight warning has wrong location in some builders (LaTeX, singlehtml and so on)
- #8215: Eliminate Fancyhdr build warnings for oneside documents
- #8239: Failed to refer a token in productionlist if it is indented
- #8268: linkcheck: Report HTTP errors when `linkcheck_anchors` is `True`
- #8245: linkcheck: take source directory into account for local files
- #8321: linkcheck: `tel:` schema hyperlinks are detected as errors
- #8323: linkcheck: An exit status is incorrect when links having unsupported schema found
- #8188: C, add missing items to internal object types dictionary, e.g., preventing intersphinx from resolving them.
- C, fix anon objects in intersphinx.

- #8270, C++, properly reject functions as duplicate declarations if a non-function declaration of the same name already exists.
- C, fix references to function parameters. Link to the function instead of a non-existing anchor.
- #6914: figure numbers are unexpectedly assigned to uncaptioned items
- #8320: make "inline" line numbers un-selectable

Testing

- #8257: Support parallel build in sphinx.testing

10.11 Release 3.2.1 (released Aug 14, 2020)

Features added

- #8095: napoleon: Add `napoleon_preprocess_types` to enable the type preprocessor for numpy style docstrings
- #8114: C and C++, parse function attributes after parameters and qualifiers.

Bugs fixed

- #8074: napoleon: Crashes during processing C-ext module
- #8088: napoleon: "Inline literal start-string without end-string" warning in Numpy style Parameters section
- #8084: autodoc: KeyError is raised on documenting an attribute of the broken class
- #8091: autodoc: AttributeError is raised on documenting an attribute on Python 3.5.2
- #8099: autodoc: NameError is raised when target code uses `TYPE_CHECKING`
- C++, fix parsing of template template paramters, broken by the fix of #7944

10.12 Release 3.2.0 (released Aug 08, 2020)

Deprecated

- `sphinx.ext.autodoc.members_set_option()`
- `sphinx.ext.autodoc.merge_special_members_option()`
- `sphinx.writers.texinfo.TexinfoWriter.desc`
- C, parsing of pre-v3 style type directives and roles, along with the options *`c_allow_pre_v3`* and *`c_warn_on_allowed_pre_v3`*.

Features added

- #2076: autodoc: Allow overriding of exclude-members in skip-member function
- #8034: autodoc: `:private-member:` can take an explicit list of member names to be documented
- #2024: autosummary: Add *autosummary_filename_map* to avoid conflict of filenames between two object with different case
- #8011: autosummary: Support instance attributes as a target of autosummary directive
- #7849: html: Add *html_codeblock_linenos_style* to change the style of line numbers for code-blocks
- #7853: C and C++, support parameterized GNU style attributes.
- #7888: napoleon: Add aliases Warn and Raise.
- #7690: napoleon: parse type strings and make them hyperlinks as possible. The conversion rule can be updated via *napoleon_type_aliases*
- #8049: napoleon: Create a hyperlink for each the type of parameter when `napoleon_use_params` is False
- C, added *c:alias* directive for inserting copies of existing declarations.
- #7745: html: inventory is broken if the docname contains a space
- #7991: html search: Allow searching for numbers
- #7902: html theme: Add a new option `globaltoc_maxdepth` to control the behavior of globaltoc in sidebar
- #7840: i18n: Optimize the dependencies check on bootstrap
- #7768: i18n: *figure_language_filename* supports `docpath` token
- #5208: linkcheck: Support checks for local links
- #5090: setuptools: Link verbosity to distutils' -v and -q option
- #6698: doctest: Add `:trim-doctest-flags:` and `:no-trim-doctest-flags:` options to doctest, testcode and testoutput directives
- #7052: add `:noindexentry:` to the Python, C, C++, and Javascript domains. Update the documentation to better reflect the relationship between this option and the `:noindex:` option.
- #7899: C, add possibility of parsing of some pre-v3 style type directives and roles and try to convert them to equivalent v3 directives/roles. Set the new option *c_allow_pre_v3* to `True` to enable this. The warnings printed from this functionality can be suppressed by setting ``c_warn_on_allowed_pre_v3` `` to `True`. The functionality is immediately deprecated.
- #7999: C, add support for named variadic macro arguments.
- #8071: Allow to suppress "self referenced toctrees" warning

Bugs fixed

- #7886: autodoc: TypeError is raised on mocking generic-typed classes
- #7935: autodoc: function signature is not shown when the function has a parameter having `inspect._empty` as its default value
- #7901: autodoc: type annotations for overloaded functions are not resolved
- #904: autodoc: An instance attribute cause a crash of autofunction directive
- #1362: autodoc: `private-members` option does not work for class attributes
- #7983: autodoc: Generator type annotation is wrongly rendered in py36

- #8030: autodoc: An uninitialized annotated instance variable is not documented when `:inherited-members:` option given
- #8032: autodoc: A type hint for the instance variable defined at parent class is not shown in the document of the derived class
- #8041: autodoc: An annotated instance variable on super class is not documented when derived class has other annotated instance variables
- #7839: autosummary: cannot handle umlauts in function names
- #7865: autosummary: Failed to extract summary line when abbreviations found
- #7866: autosummary: Failed to extract correct summary line when docstring contains a hyperlink target
- #7469: autosummary: "Module attributes" header is not translatable
- #7940: apidoc: An extra newline is generated at the end of the rst file if a module has submodules
- #4258: napoleon: decorated special methods are not shown
- #7799: napoleon: parameters are not escaped for combined params in numpydoc
- #7780: napoleon: multiple paramaters declaration in numpydoc was wrongly recognized when napoleon_use_params=True
- #7715: LaTeX: `numfig_secnum_depth > 1` leads to wrong figure links
- #7846: html theme: XML-invalid files were generated
- #7894: gettext: Wrong source info is shown when using rst_epilog
- #7691: linkcheck: HEAD requests are not used for checking
- #4888: i18n: Failed to add an explicit title to `:ref:` role on translation
- #7928: py domain: failed to resolve a type annotation for the attribute
- #8008: py domain: failed to parse a type annotation containing ellipsis
- #7994: std domain: option directive does not generate old node_id compatible with 2.x or older
- #7968: i18n: The content of `math` directive is interpreted as reST on translation
- #7768: i18n: The `root` element for *`figure_language_filename`* is not a path that user specifies in the document
- #7993: texinfo: TypeError is raised for nested object descriptions
- #7993: texinfo: a warning not supporting desc_signature_line node is shown
- #7869: *`abbr`* role without an explanation will show the explanation from the previous abbr role
- #8048: graphviz: graphviz.css was copied on building non-HTML document
- C and C++, removed `noindex` directive option as it did nothing.
- #7619: Duplicated node IDs are generated if node has multiple IDs
- #2050: Symbols sections are appeared twice in the index page
- #8017: Fix circular import in sphinx.addnodes
- #7986: CSS: make "highlight" selector more robust
- #7944: C++, parse non-type template parameters starting with a dependent qualified name.
- C, don't deepcopy the entire symbol table and make a mess every time an enumerator is handled.

10.13 Release 3.1.2 (released Jul 05, 2020)

Incompatible changes

- #7650: autodoc: the signature of base function will be shown for decorated functions, not a signature of decorator

Bugs fixed

- #7844: autodoc: Failed to detect module when relative module name given
- #7856: autodoc: AttributeError is raised when non-class object is given to the autoclass directive
- #7850: autodoc: KeyError is raised for invalid mark up when autodoc_typehints is 'description'
- #7812: autodoc: crashed if the target name matches to both an attribute and module that are same name
- #7650: autodoc: function signature becomes `(*args, **kwargs)` if the function is decorated by generic decorator
- #7812: autosummary: generates broken stub files if the target code contains an attribute and module that are same name
- #7806: viewcode: Failed to resolve viewcode references on 3rd party builders
- #7838: html theme: List items have extra vertical space
- #7878: html theme: Undesired interaction between "overflow" and "float"

10.14 Release 3.1.1 (released Jun 14, 2020)

Incompatible changes

- #7808: napoleon: a type for attribute are represented as typed field

Features added

- #7807: autodoc: Show detailed warning when type_comment is mismatched with its signature

Bugs fixed

- #7808: autodoc: Warnings raised on variable and attribute type annotations
- #7802: autodoc: EOFError is raised on parallel build
- #7821: autodoc: TypeError is raised for overloaded C-ext function
- #7805: autodoc: an object which descriptors returns is unexpectedly documented
- #7807: autodoc: wrong signature is shown for the function using contextmanager
- #7812: autosummary: generates broken stub files if the target code contains an attribute and module that are same name
- #7808: napoleon: Warnings raised on variable and attribute type annotations
- #7811: sphinx.util.inspect causes circular import problem

10.15 Release 3.1.0 (released Jun 08, 2020)

Dependencies

- #7746: mathjax: Update to 2.7.5

Incompatible changes

- #7477: imgconverter: Invoke "magick convert" command by default on Windows

Deprecated

- The first argument for sphinx.ext.autosummary.generate.AutosummaryRenderer has been changed to Sphinx object
- `sphinx.ext.autosummary.generate.AutosummaryRenderer` takes an object type as an argument
- The `ignore` argument of `sphinx.ext.autodoc.Documenter.get_doc()`
- The `template_dir` argument of `sphinx.ext.autosummary.generate. AutosummaryRenderer`
- The `module` argument of `sphinx.ext.autosummary.generate. find_autosummary_in_docstring()`
- The `builder` argument of `sphinx.ext.autosummary.generate. generate_autosummary_docs()`
- The `template_dir` argument of `sphinx.ext.autosummary.generate. generate_autosummary_docs()`
- The `ignore` argument of `sphinx.util.docstring.prepare_docstring()`
- `sphinx.ext.autosummary.generate.AutosummaryRenderer.exists()`
- `sphinx.util.rpartition()`

Features added

- LaTeX: Make the `toplevel_sectioning` setting optional in LaTeX theme
- LaTeX: Allow to override papersize and pointsize from LaTeX themes
- LaTeX: Add *`latex_theme_options`* to override theme options
- #7410: Allow to suppress "circular toctree references detected" warnings using *`suppress_warnings`*
- C, added scope control directives, *`c:namespace`*, *`c:namespace-push`*, and *`c:namespace-pop`*.
- #2044: autodoc: Suppress default value for instance attributes
- #7473: autodoc: consider a member public if docstring contains `:meta public:` in info-field-list
- #7487: autodoc: Allow to generate docs for singledispatch functions by py:autofunction
- #7143: autodoc: Support final classes and methods
- #7384: autodoc: Support signatures defined by `__new__()`, metaclasses and builtin base classes
- #2106: autodoc: Support multiple signatures on docstring
- #4422: autodoc: Support GenericAlias in Python 3.7 or above
- #3610: autodoc: Support overloaded functions
- #7722: autodoc: Support TypeVar

- #7466: autosummary: headings in generated documents are not translated
- #7490: autosummary: Add `:caption:` option to autosummary directive to set a caption to the toctree
- #7469: autosummary: Support module attributes
- #248, #6040: autosummary: Add `:recursive:` option to autosummary directive to generate stub files recursively
- #4030: autosummary: Add *`autosummary_context`* to add template variables for custom templates
- #7530: html: Support nested <kbd> elements
- #7481: html theme: Add right margin to footnote/citation labels
- #7482, #7717: html theme: CSS spacing for code blocks with captions and line numbers
- #7443: html theme: Add new options `globaltoc_collapse` and `globaltoc_includehidden` to control the behavior of globaltoc in sidebar
- #7484: html theme: Avoid clashes between sidebar and other blocks
- #7476: html theme: Relbar breadcrumb should contain current page
- #7506: html theme: A canonical URL is not escaped
- #7533: html theme: Avoid whitespace at the beginning of genindex.html
- #7541: html theme: Add a "clearer" at the end of the "body"
- #7542: html theme: Make admonition/topic/sidebar scrollable
- #7543: html theme: Add top and bottom margins to tables
- #7695: html theme: Add viewport meta tag for basic theme
- #7721: html theme: classic: default codetextcolor/codebgcolor doesn't override Pygments
- C and C++: allow semicolon in the end of declarations.
- C++, parse parameterized noexcept specifiers.
- #7294: C++, parse expressions with user-defined literals.
- C++, parse trailing return types.
- #7143: py domain: Add `:final:` option to `py:class:`, `py:exception:` and `py:method:` directives
- #7596: py domain: Change a type annotation for variables to a hyperlink
- #7770: std domain: *`option`* directive support arguments in the form of `foo[=bar]`
- #7582: napoleon: a type for attribute are represented like type annotation
- #7734: napoleon: overescaped trailing underscore on attribute
- #7247: linkcheck: Add *`linkcheck_request_headers`* to send custom HTTP headers for specific host
- #7792: setuptools: Support `--verbosity` option
- #7683: Add `allowed_exceptions` parameter to `Sphinx.emit()` to allow handlers to raise specified exceptions
- #7295: C++, parse (trailing) requires clauses.

Bugs fixed

- #6703: autodoc: incremental build does not work for imported objects
- #7564: autodoc: annotations not to be shown for descriptors
- #6588: autodoc: Decorated inherited method has no documentation
- #7469: autodoc: The change of autodoc-process-docstring for variables is cached unexpectedly
- #7559: autodoc: misdetects a sync function is async
- #6857: autodoc: failed to detect a classmethod on Enum class
- #7562: autodoc: a typehint contains spaces is wrongly rendered under autodoc_typehints='description' mode
- #7551: autodoc: failed to import nested class
- #7362: autodoc: does not render correct signatures for built-in functions
- #7654: autodoc: `Optional[Union[foo, bar]]` is presented as `Union[foo, bar, None]`
- #7629: autodoc: autofunction emits an unfriendly warning if an invalid object specified
- #7650: autodoc: undecorated signature is shown for decorated functions
- #7676: autodoc: typo in the default value of autodoc_member_order
- #7676: autodoc: wrong value for :member-order: option is ignored silently
- #7676: autodoc: member-order="bysource" does not work for C module
- #3673: autodoc: member-order="bysource" does not work for a module having __all__
- #7668: autodoc: wrong retann value is passed to a handler of autodoc-proccess-signature
- #7711: autodoc: fails with ValueError when processing numpy objects
- #7791: autodoc: TypeError is raised on documenting singledispatch function
- #7551: autosummary: a nested class is indexed as non-nested class
- #7661: autosummary: autosummary directive emits warnings twices if failed to import the target module
- #7685: autosummary: The template variable "members" contains imported members even if `autossummary_imported_members` is False
- #7671: autosummary: The location of import failure warning is missing
- #7535: sphinx-autogen: crashes when custom template uses inheritance
- #7536: sphinx-autogen: crashes when template uses i18n feature
- #7781: sphinx-build: Wrong error message when outdir is not directory
- #7653: sphinx-quickstart: Fix multiple directory creation for nested relpath
- #2785: html: Bad alignment of equation links
- #7718: html theme: some themes does not respect background color of Pygments style (agogo, haiku, nature, pyramid, scrolls, sphinxdoc and traditional)
- #7544: html theme: inconsistent padding in admonitions
- #7581: napoleon: bad parsing of inline code in attribute docstrings
- #7628: imgconverter: runs imagemagick once unnecessary for builders not supporting images
- #7610: incorrectly renders consecutive backslashes for docutils-0.16
- #7646: handle errors on event handlers
- #4187: LaTeX: EN DASH disappears from PDF bookmarks in Japanese documents

- #7701: LaTeX: Anonymous indirect hyperlink target causes duplicated labels
- #7723: LaTeX: pdflatex crashed when URL contains a single quote
- #7756: py domain: The default value for positional only argument is not shown
- #7760: coverage: Add *coverage_show_missing_items* to show coverage result to console
- C++, fix rendering and xrefs in nested names explicitly starting in global scope, e.g., `::A::B`.
- C, fix rendering and xrefs in nested names explicitly starting in global scope, e.g., `.A.B`.
- #7763: C and C++, don't crash during display stringification of unary expressions and fold expressions.

10.16 Release 3.0.4 (released May 27, 2020)

Bugs fixed

- #7567: autodoc: parametrized types are shown twice for generic types
- #7637: autodoc: system defined TypeVars are shown in Python 3.9
- #7696: html: Updated jQuery version from 3.4.1 to 3.5.1 for security reasons
- #7611: md5 fails when OpenSSL FIPS is enabled
- #7626: release package does not contain `CODE_OF_CONDUCT`

10.17 Release 3.0.3 (released Apr 26, 2020)

Features added

- C, parse array declarators with static, qualifiers, and VLA specification.

Bugs fixed

- #7516: autodoc: crashes if target object raises an error on accessing its attributes

10.18 Release 3.0.2 (released Apr 19, 2020)

Features added

- C, parse attributes and add *c_id_attributes* and *c_paren_attributes* to support user-defined attributes.

Bugs fixed

- #7461: py domain: fails with IndexError for empty tuple in type annotation
- #7510: py domain: keyword-only arguments are documented as having a default of None
- #7418: std domain: *term* role could not match case-insensitively
- #7461: autodoc: empty tuple in type annotation is not shown correctly
- #7479: autodoc: Sphinx builds has been slower since 3.0.0 on mocking
- C++, fix spacing issue in east-const declarations.
- #7414: LaTeX: Xindy language options were incorrect
- sphinx crashes with ImportError on python3.5.1

10.19 Release 3.0.1 (released Apr 11, 2020)

Incompatible changes

- #7418: std domain: `term` role becomes case sensitive

Bugs fixed

- #7428: py domain: a reference to class `None` emits a nitpicky warning
- #7445: py domain: a return annotation `None` in the function signature is not converted to a hyperlink when using intersphinx
- #7418: std domain: duplication warning for glossary terms is case insensitive
- #7438: C++, fix merging overloaded functions in parallel builds.
- #7422: autodoc: fails with ValueError when using autodoc_mock_imports
- #7435: autodoc: `autodoc_typehints='description'` doesn't suppress typehints in signature for classes/methods
- #7451: autodoc: fails with AttributeError when an object returns non-string object as a `__doc__` member
- #7423: crashed when giving a non-string object to logger
- #7479: html theme: Do not include xmlns attribute with HTML 5 doctype
- #7426: html theme: Escape some links in HTML templates

10.20 Release 3.0.0 (released Apr 06, 2020)

Dependencies

3.0.0b1

- LaTeX: drop dependency on **`extractbb`** for image inclusion in Japanese documents as `.xbb` files are unneeded by **`dvipdfmx`** since TeXLive2015 (refs: #6189)
- babel-2.0 or above is available (Unpinned)

Incompatible changes

3.0.0b1

- Drop features and APIs deprecated in 1.8.x
- #247: autosummary: stub files are overwritten automatically by default. see *autosummary_generate_overwrite* to change the behavior
- #5923: autodoc: the members of `object` class are not documented by default when `:inherited-members:` and `:special-members:` are given.
- #6830: py domain: `meta` fields in info-field-list becomes reserved. They are not displayed on output document now
- #6417: py domain: doctree of desc_parameterlist has been changed. The argument names, annotations and default values are wrapped with inline node
- The structure of `sphinx.events.EventManager.listeners` has changed
- Due to the scoping changes for *productionlist* some uses of *token* must be modified to include the scope which was previously ignored.
- #6903: Internal data structure of Python, reST and standard domains have changed. The node_id is added to the index of objects and modules. Now they contains a pair of docname and node_id for cross reference.
- #7276: C++ domain: Non intended behavior is removed such as `say_hello_` links to `.. cpp:function:: say_hello()`
- #7210: js domain: Non intended behavior is removed such as `parseInt_` links to `.. js:function:: parseInt`
- #7229: rst domain: Non intended behavior is removed such as `numref_` links to `.. rst:role:: numref`
- #6903: py domain: Non intended behavior is removed such as `say_hello_` links to `.. py:function:: say_hello()`
- #7246: py domain: Drop special cross reference helper for exceptions, functions and methods
- The C domain has been rewritten, with additional directives and roles. The existing ones are now more strict, resulting in new warnings.
- The attribute `sphinx_cpp_tagname` in the `desc_signature_line` node has been renamed to `sphinx_line_type`.
- #6462: double backslashes in domain directives are no longer replaced by single backslashes as default. A new configuration value *strip_signature_backslash* can be used by users to reenable it.

3.0.0 final

- #7222: `sphinx.util.inspect.unwrap()` is renamed to `unwrap_all()`

Deprecated

3.0.0b1

- `desc_signature['first']`
- `sphinx.directives.DescDirective`
- `sphinx.domains.std.StandardDomain.add_object()`
- `sphinx.domains.python.PyDecoratorMixin`
- `sphinx.ext.autodoc.get_documenters()`
- `sphinx.ext.autosummary.process_autosummary_toc()`

- `sphinx.parsers.Parser.app`
- `sphinx.testing.path.Path.text()`
- `sphinx.testing.path.Path.bytes()`
- `sphinx.util.inspect.getargspec()`
- `sphinx.writers.latex.LaTeXWriter.format_docclass()`

Features added

3.0.0b1

- #247: autosummary: Add *`autosummary_generate_overwrite`* to overwrite old stub file
- #5923: autodoc: `:inherited-members:` option takes a name of anchestor class not to document inherited members of the class and uppers
- #6830: autodoc: consider a member private if docstring contains `:meta private:` in info-field-list
- #7165: autodoc: Support Annotated type (PEP-593)
- #2815: autodoc: Support singledispatch functions and methods
- #7079: autodoc: *`autodoc_typehints`* accepts `"description"` configuration. It shows typehints as object description
- #7314: apidoc: Propagate `--maxdepth` option through package documents
- #6558: glossary: emit a warning for duplicated glossary entry
- #3106: domain: Register hyperlink target for index page automatically
- #6558: std domain: emit a warning for duplicated generic objects
- #6830: py domain: Add new event: *`object-description-transform`*
- #6895: py domain: Do not emit nitpicky warnings for built-in types
- py domain: Support lambda functions in function signature
- #6417: py domain: Allow to make a style for arguments of functions and methods
- #7238, #7239: py domain: Emit a warning on describing a python object if the entry is already added as the same name
- #7341: py domain: type annotations in singature are converted to cross refs
- Support priority of event handlers. For more detail, see *`Sphinx.connect()`*
- #3077: Implement the scoping for *`productionlist`* as indicated in the documentation.
- #1027: Support backslash line continuation in *`productionlist`*.
- #7108: config: Allow to show an error message from conf.py via `ConfigError`
- #7032: html: *`html_scaled_image_link`* will be disabled for images having `no-scaled-link` class
- #7144: Add CSS class indicating its domain for each desc node
- #7211: latex: Use babel for Chinese document when using XeLaTeX
- #6672: LaTeX: Support LaTeX Theming (experimental)
- #7005: LaTeX: Add LaTeX styling macro for *`kbd`* role
- #7220: genindex: Show “main” index entries at first
- #7103: linkcheck: writes all links to `output.json`

- #7025: html search: full text search can be disabled for individual document using `:nosearch:` file-wide metadata
- #7293: html search: Allow to override JavaScript splitter via `SearchLanguage.js_splitter_code`
- #7142: html theme: Add a theme option: `pygments_dark_style` to switch the style of code-blocks in dark mode
- The C domain has been rewritten adding for example:
  - Cross-referencing respecting the current scope.
  - Possible to document anonymous entities.
  - More specific directives and roles for each type of entitiy, e.g., handling scoping of enumerators.
  - New role *`c:expr`* for rendering expressions and types in text.
- Added `SphinxDirective.get_source_info()` and `SphinxRole.get_source_info()`.
- #7324: sphinx-build: Emit a warning if multiple files having different file extensions for same document found

3.0.0 final

- Added `ObjectDescription.transform_content()`.

Bugs fixed

3.0.0b1

- C++, fix cross reference lookup in certain cases involving function overloads.
- #5078: C++, fix cross reference lookup when a directive contains multiple declarations.
- C++, suppress warnings for directly dependent typenames in cross references generated automatically in signatures.
- #5637: autodoc: Incorrect handling of nested class names on show-inheritance
- #7267: autodoc: error message for invalid directive options has wrong location
- #7329: autodoc: info-field-list is wrongly generated from type hints into the class description even if `autoclass_content='class'` set
- #7331: autodoc: a cython-function is not recognized as a function
- #5637: inheritance_diagram: Incorrect handling of nested class names
- #7139: `code-block:: guess` does not work
- #7325: html: source_suffix containing dot leads to wrong source link
- #7357: html: Resizing SVG image fails with ValueError
- #7278: html search: Fix use of `html_file_suffix` instead of `html_link_suffix` in search results
- #7297: html theme: `bizstyle` does not support `sidebarwidth`
- #3842: singlehtml: Path to images broken when master doc is not in source root
- #7179: std domain: Fix whitespaces are suppressed on referring GenericObject
- #7289: console: use bright colors instead of bold
- #1539: C, parse array types.
- #2377: C, parse function pointers even in complex types.
- #7345: sphinx-build: Sphinx crashes if output directory exists as a file
- #7290: sphinx-build: Ignore bdb.BdbQuit when handling exceptions

- #6240: napoleon: Attributes and Methods sections ignore :noindex: option

3.0.0 final

- #7364: autosummary: crashed when `autosummary_generate` is False
- #7370: autosummary: raises UnboundLocalError when unknown module given
- #7367: C++, alternate operator spellings are now supported.
- C, alternate operator spellings are now supported.
- #7368: C++, comma operator in expressions, pack expansion in template argument lists, and more comprehensive error messages in some cases.
- C, C++, fix crash and wrong duplicate warnings related to anon symbols.
- #6477: Escape first "!" in a cross reference linking no longer possible
- #7219: py domain: The index entry generated by `py:function` directive is different with one from `index` directive with "builtin" type
- #7301: capital characters are not allowed for node_id
- #7301: epub: duplicated node_ids are generated
- #6564: html: a width of table was ignored on HTML builder
- #7401: Incorrect argument is passed for `env-get-outdated` handlers
- #7355: autodoc: a signature of cython-function is not recognized well
- #7222: autodoc: `__wrapped__` functions are not documented correctly
- #7409: intersphinx: ValueError is raised when an extension sets up `intersphinx_mapping` on `config-inited` event
- #7343: Sphinx builds has been slower since 2.4.0 on debug mode

10.21 Release 2.4.4 (released Mar 05, 2020)

Bugs fixed

- #7197: LaTeX: platex cause error to build image directive with target url
- #7223: Sphinx builds has been slower since 2.4.0

10.22 Release 2.4.3 (released Feb 22, 2020)

Bugs fixed

- #7184: autodoc: `*args` and `**kwarg` in type comments are not handled properly
- #7189: autodoc: classmethod coroutines are not detected
- #7183: intersphinx: `:attr:` reference to property is broken
- #6244, #6387: html search: Search breaks/hangs when built with dirhtml builder
- #7195: todo: emit doctree-resolved event with non-document node incorrectly

10.23 Release 2.4.2 (released Feb 19, 2020)

Bugs fixed

- #7138: autodoc: `autodoc.typehints` crashed when variable has unbound object as a value
- #7156: autodoc: separator for keyword only arguments is not shown
- #7146: autodoc: IndexError is raised on suppressed type_comment found
- #7161: autodoc: typehints extension does not support parallel build
- #7178: autodoc: TypeError is raised on fetching type annotations
- #7151: crashed when extension assigns a value to `env.indexentries`
- #7170: text: Remove debug print
- #7137: viewcode: Avoid to crash when non-python code given

10.24 Release 2.4.1 (released Feb 11, 2020)

Bugs fixed

- #7120: html: crashed when on scaling SVG images which have float dimensions
- #7126: autodoc: TypeError: 'getset_descriptor' object is not iterable

10.25 Release 2.4.0 (released Feb 09, 2020)

Deprecated

- The decode argument of `sphinx.pycode.ModuleAnalyzer()`
- `sphinx.directives.other.Index`
- `sphinx.environment.temp_data['gloss_entries']`
- `sphinx.environment.BuildEnvironment.indexentries`
- `sphinx.environment.collectors.indexentries.IndexEntriesCollector`
- `sphinx.ext.apidoc.INITPY`
- `sphinx.ext.apidoc.shall_skip()`
- `sphinx.io.FiletypeNotFoundError`
- `sphinx.io.get_filetype()`
- `sphinx.pycode.ModuleAnalyzer.encoding`
- `sphinx.roles.Index`
- `sphinx.util.detect_encoding()`
- `sphinx.util.get_module_source()`
- `sphinx.util.inspect.Signature`
- `sphinx.util.inspect.safe_getmembers()`
- `sphinx.writers.latex.LaTeXTranslator.settings.author`

- `sphinx.writers.latex.LaTeXTranslator.settings.contentsname`
- `sphinx.writers.latex.LaTeXTranslator.settings.docclass`
- `sphinx.writers.latex.LaTeXTranslator.settings.docname`
- `sphinx.writers.latex.LaTeXTranslator.settings.title`
- `sphinx.writers.latex.ADDITIONAL_SETTINGS`
- `sphinx.writers.latex.DEFAULT_SETTINGS`
- `sphinx.writers.latex.LUALATEX_DEFAULT_FONTPKG`
- `sphinx.writers.latex.PDFLATEX_DEFAULT_FONTPKG`
- `sphinx.writers.latex.XELATEX_DEFAULT_FONTPKG`
- `sphinx.writers.latex.XELATEX_GREEK_DEFAULT_FONTPKG`

Features added

- #6910: inheritance_diagram: Make the background of diagrams transparent
- #6446: duration: Add `sphinx.ext.durations` to inspect which documents slow down the build
- #6837: LaTeX: Support a nested table
- #7115: LaTeX: Allow to override LATEXOPTS and LATEXMKOPTS via environment variable
- #6966: graphviz: Support `:class:` option
- #6696: html: `:scale:` option of image/figure directive not working for SVG images (imagesize-1.2.0 or above is required)
- #6994: imgconverter: Support illustrator file (.ai) to .png conversion
- autodoc: Support Positional-Only Argument separator (PEP-570 compliant)
- autodoc: Support type annotations for variables
- #2755: autodoc: Add new event: *`autodoc-before-process-signature`*
- #2755: autodoc: Support type_comment style (ex. `# type: (str) -> str`) annotation (python3.8+ or typed_ast[604] is required)
- #7051: autodoc: Support instance variables without defaults (PEP-526)
- #6418: autodoc: Add a new extension `sphinx.ext.autodoc.typehints`. It shows typehints as object description if `autodoc_typehints = "description"` set. This is an experimental extension and it will be integrated into autodoc core in Sphinx-3.0
- SphinxTranslator now calls visitor/departure method for super node class if visitor/departure method for original node class not found
- #6418: Add new event: *`object-description-transform`*
- py domain: *`py:data`* and *`py:attribute`* take new options named `:type:` and `:value:` to describe its type and initial value
- #6785: py domain: `:py:attr:` is able to refer properties again
- #6772: apidoc: Add `-q` option for quiet mode

[604] https://github.com/python/typed_ast

Bugs fixed

- #6925: html: Remove redundant type="text/javascript" from <script> elements
- #7112: html: SVG image is not layouted as float even if aligned
- #6906, #6907: autodoc: failed to read the source codes encoeded in cp1251
- #6961: latex: warning for babel shown twice
- #7059: latex: LaTeX compilation falls into infinite loop (wrapfig issue)
- #6581: latex: `:reversed:` option for toctree does not effect to LaTeX build
- #6559: Wrong node-ids are generated in glossary directive
- #6986: apidoc: misdetects module name for .so file inside module
- #6899: apidoc: private members are not shown even if `--private` given
- #6327: apidoc: Support a python package consisted of __init__.so file
- #6999: napoleon: fails to parse tilde in :exc: role
- #7019: gettext: Absolute path used in message catalogs
- #7023: autodoc: nested partial functions are not listed
- #7023: autodoc: partial functions imported from other modules are listed as module members without :impoprted-members: option
- #6889: autodoc: Trailing comma in `:members::` option causes cryptic warning
- #6568: autosummary: `autosummary_imported_members` is ignored on generating a stub file for submodule
- #7055: linkcheck: redirect is treated as an error
- #7088: HTML template: If `navigation_with_keys` option is activated, modifier keys are ignored, which means the feature can interfere with browser features
- #7090: std domain: Can't assign numfig-numbers for custom container nodes
- #7106: std domain: enumerated nodes are marked as duplicated when extensions call `note_explicit_target()`
- #7095: dirhtml: Cross references are broken via intersphinx and `:doc:` role
- C++:
  - Don't crash when using the `struct` role in some cases.
  - Don't warn when using the `var`/`member` role for function parameters.
  - Render call and braced-init expressions correctly.
- #7097: Filenames of images generated by `sphinx.transforms.post_transforms.images.ImageConverter` or its subclasses (used for latex build) are now sanitized, to prevent broken paths

10.26 Release 2.3.1 (released Dec 22, 2019)

Bugs fixed

- #6936: sphinx-autogen: raises AttributeError

10.27 Release 2.3.0 (released Dec 15, 2019)

Incompatible changes

- #6742: `end-before` option of *`literalinclude`* directive does not match the first line of the code block.
- #1331: Change default User-Agent header to `"Sphinx/X.Y.Z requests/X.Y.Z python/X.Y.Z"`. It can be changed via *`user_agent`*.
- #6867: text: content of admonitions starts after a blank line

Deprecated

- `sphinx.builders.gettext.POHEADER`
- `sphinx.io.SphinxStandaloneReader.app`
- `sphinx.io.SphinxStandaloneReader.env`
- `sphinx.util.texescape.tex_escape_map`
- `sphinx.util.texescape.tex_hl_escape_map_new`
- `sphinx.writers.latex.LaTeXTranslator.no_contractions`

Features added

- #6707: C++, support bit-fields.
- #267: html: Eliminate prompt characters of doctest block from copyable text
- #6548: html: Use favicon for OpenSearch if available
- #6729: html theme: agogo theme now supports `rightsidebar` option
- #6780: Add PEP-561 Support
- #6762: latex: Allow to load additional LaTeX packages via `extrapackages` key of *`latex_elements`*
- #1331: Add new config variable: *`user_agent`*
- #6000: LaTeX: have backslash also be an inline literal word wrap break character
- #4186: LaTeX: Support upLaTeX as a new *`latex_engine`* (experimental)
- #6812: Improve a warning message when extensions are not parallel safe
- #6818: Improve Intersphinx performance for multiple remote inventories.
- #2546: apidoc: .so file support
- #6798: autosummary: emit `autodoc-skip-member` event on generating stub file
- #6483: i18n: make explicit titles in toctree translatable

- #6816: linkcheck: Add *linkcheck_auth* option to provide authentication information when doing `linkcheck` builds
- #6872: linkcheck: Handles HTTP 308 Permanent Redirect
- #6613: html: Wrap section number in span tag
- #6781: gettext: Add `gettext_last_translator' and :confval:`gettext_language_team` to customize headers of POT file

Bugs fixed

- #6668: LaTeX: Longtable before header has incorrect distance (refs: latex3/latex2e#173[605])
- #6618: LaTeX: Avoid section names at the end of a page
- #6738: LaTeX: Do not replace unicode characters by LaTeX macros on unicode supported LaTeX engines: ¶, §, €, ∞, ±, →, →, –, superscript and subscript digits go through "as is" (as default OpenType font supports them)
- #6704: linkcheck: Be defensive and handle newly defined HTTP error code
- #6806: linkcheck: Failure on parsing content
- #6655: image URLs containing `data:` causes gettext builder crashed
- #6584: i18n: Error when compiling message catalogs on Hindi
- #6718: i18n: KeyError is raised if section title and table title are same
- #6743: i18n: *rst_prolog* breaks the translation
- #6708: mathbase: Some deprecated functions have removed
- #6709: autodoc: mock object does not work as a class decorator
- #5070: epub: Wrong internal href fragment links
- #6712: Allow not to install sphinx.testing as runtime (mainly for ALT Linux)
- #6741: html: search result was broken with empty *html_file_suffix*
- #6001: LaTeX does not wrap long code lines at backslash character
- #6804: LaTeX: PDF build breaks if admonition of danger type contains code-block long enough not to fit on one page
- #6809: LaTeX: code-block in a danger type admonition can easily spill over bottom of page
- #6793: texinfo: Code examples broken following "sidebar"
- #6813: An orphan warning is emitted for included document on Windows. Thanks to @drillan
- #6850: Fix smartypants module calls re.sub() with wrong options
- #6824: HTML search: If a search term is partially matched in the title and fully matched in a text paragraph on the same page, the search does not include this match.
- #6848: config.py shouldn't pop extensions from overrides
- #6867: text: extra spaces are inserted to hyphenated words on folding lines
- #6886: LaTeX: xelatex converts straight double quotes into right curly ones (shows when *smartquotes* is `False`)
- #6890: LaTeX: even with smartquotes off, PDF output transforms straight quotes and consecutive hyphens into curly quotes and dashes
- #6876: LaTeX: multi-line display of authors on title page has ragged edges

605 https://github.com/latex3/latex2e/issues/173

- #6887: Sphinx crashes with docutils-0.16b0
- #6920: sphinx-build: A console message is wrongly highlighted
- #6900: sphinx-build: `-D` option does not considers `0` and `1` as a boolean value

10.28 Release 2.2.2 (released Dec 03, 2019)

Incompatible changes

- #6803: For security reason of python, parallel mode is disabled on macOS and Python3.8+

Bugs fixed

- #6776: LaTeX: 2019-10-01 LaTeX release breaks `sphinxcyrillic.sty`
- #6815: i18n: French, Hindi, Chinese, Japanese and Korean translation messages has been broken
- #6803: parallel build causes AttributeError on macOS and Python3.8

10.29 Release 2.2.1 (released Oct 26, 2019)

Bugs fixed

- #6641: LaTeX: Undefined control sequence `\sphinxmaketitle`
- #6710: LaTeX not well configured for Greek language as main language
- #6759: validation of html static paths and extra paths no longer throws an error if the paths are in different directories

10.30 Release 2.2.0 (released Aug 19, 2019)

Incompatible changes

- apidoc: template files are renamed to `.rst_t`
- html: Field lists will be styled by grid layout

Deprecated

- `sphinx.domains.math.MathDomain.add_equation()`
- `sphinx.domains.math.MathDomain.get_next_equation_number()`
- The `info` and `warn` arguments of `sphinx.ext.autosummary.generate.generate_autosummary_docs()`
- `sphinx.ext.autosummary.generate._simple_info()`
- `sphinx.ext.autosummary.generate._simple_warn()`
- `sphinx.ext.todo.merge_info()`
- `sphinx.ext.todo.process_todo_nodes()`
- `sphinx.ext.todo.process_todos()`

- `sphinx.ext.todo.purge_todos()`

Features added

- #5124: graphviz: `:graphviz_dot:` option is renamed to `:layout:`
- #1464: html: emit a warning if *`html_static_path`* and *`html_extra_path`* directories are inside output directory
- #6514: html: Add a label to search input for accessability purposes
- #5602: apidoc: Add `--templatedir` option
- #6475: Add `override` argument to `app.add_autodocumenter()`
- #6310: imgmath: let *`imgmath_use_preview`* work also with the SVG format for images rendering inline math
- #6533: LaTeX: refactor visit_enumerated_list() to use `\sphinxsetlistlabels`
- #6628: quickstart: Use `https://docs.python.org/3/` for default setting of *`intersphinx_mapping`*
- #6419: sphinx-build: give reasons why rebuilded

Bugs fixed

- py domain: duplicated warning does not point the location of source code
- #6499: html: Sphinx never updates a copy of *`html_logo`* even if original file has changed
- #1125: html theme: scrollbar is hard to see on classic theme and macOS
- #5502: linkcheck: Consider HTTP 503 response as not an error
- #6439: Make generated download links reproducible
- #6486: UnboundLocalError is raised if broken extension installed
- #6567: autodoc: *`autodoc_inherit_docstrings`* does not effect to `__init__()` and `__new__()`
- #6574: autodoc: *`autodoc_member_order`* does not refer order of imports when `'bysource'` order
- #6574: autodoc: missing type annotation for variadic and keyword parameters
- #6589: autodoc: Formatting issues with autodoc_typehints='none'
- #6605: autodoc: crashed when target code contains custom method-like objects
- #6498: autosummary: crashed with wrong autosummary_generate setting
- #6507: autosummary: crashes without no autosummary_generate setting
- #6511: LaTeX: autonumbered list can not be customized in LaTeX since Sphinx 1.8.0 (refs: #6533)
- #6531: Failed to load last environment object when extension added
- #736: Invalid sort in pair index
- #6527: `last_updated` wrongly assumes timezone as UTC
- #5592: std domain: *`option`* directive registers an index entry for each comma separated option
- #6549: sphinx-build: Escaped characters in error messages
- #6545: doctest comments not getting trimmed since Sphinx 1.8.0
- #6561: glossary: Wrong hyperlinks are generated for non alphanumeric terms
- #6620: i18n: classifiers of definition list are not translated with docutils-0.15

- #6474: `DocFieldTransformer` raises AttributeError when given directive is not a subclass of ObjectDescription

10.31 Release 2.1.2 (released Jun 19, 2019)

Bugs fixed

- #6497: custom lexers fails highlighting when syntax error
- #6478, #6488: info field lists are incorrectly recognized

10.32 Release 2.1.1 (released Jun 10, 2019)

Incompatible changes

- #6447: autodoc: Stop to generate document for undocumented module variables

Bugs fixed

- #6442: LaTeX: admonitions of *`note`* type can get separated from immediately preceding section title by pagebreak
- #6448: autodoc: crashed when autodocumenting classes with `__slots__ = None`
- #6451: autodoc: generates docs for "optional import"ed modules as variables
- #6452: autosummary: crashed when generating document of properties
- #6455: napoleon: docstrings for properties are not processed
- #6436: napoleon: "Unknown target name" error if variable name ends with underscore
- #6440: apidoc: missing blank lines between modules

10.33 Release 2.1.0 (released Jun 02, 2019)

Incompatible changes

- Ignore filenames without file extension given to `Builder.build_specific()` API directly
- #6230: The anchor of term in glossary directive is changed if it is consisted by non-ASCII characters
- #4550: html: Centering tables by default using CSS
- #6239: latex: xelatex and xeCJK are used for Chinese documents by default
- `Sphinx.add_lexer()` now takes a Lexer class instead of instance. An instance of lexers are still supported until Sphinx-3.x.

Deprecated

- sphinx.builders.latex.LaTeXBuilder.apply_transforms()
- sphinx.builders._epub_base.EpubBuilder.esc()
- sphinx.directives.Acks
- sphinx.directives.Author
- sphinx.directives.Centered
- sphinx.directives.Class
- sphinx.directives.CodeBlock
- sphinx.directives.Figure
- sphinx.directives.HList
- sphinx.directives.Highlight
- sphinx.directives.Include
- sphinx.directives.Index
- sphinx.directives.LiteralInclude
- sphinx.directives.Meta
- sphinx.directives.Only
- sphinx.directives.SeeAlso
- sphinx.directives.TabularColumns
- sphinx.directives.TocTree
- sphinx.directives.VersionChange
- sphinx.domains.python.PyClassmember
- sphinx.domains.python.PyModulelevel
- sphinx.domains.std.StandardDomain._resolve_citation_xref()
- sphinx.domains.std.StandardDomain.note_citations()
- sphinx.domains.std.StandardDomain.note_citation_refs()
- sphinx.domains.std.StandardDomain.note_labels()
- sphinx.environment.NoUri
- sphinx.ext.apidoc.format_directive()
- sphinx.ext.apidoc.format_heading()
- sphinx.ext.apidoc.makename()
- sphinx.ext.autodoc.importer.MockFinder
- sphinx.ext.autodoc.importer.MockLoader
- sphinx.ext.autodoc.importer.mock()
- sphinx.ext.autosummary.autolink_role()
- sphinx.ext.imgmath.DOC_BODY
- sphinx.ext.imgmath.DOC_BODY_PREVIEW
- sphinx.ext.imgmath.DOC_HEAD
- sphinx.transforms.CitationReferences

- `sphinx.transforms.SmartQuotesSkipper`
- `sphinx.util.docfields.DocFieldTransformer.preprocess_fieldtypes()`
- `sphinx.util.node.find_source_node()`
- `sphinx.util.i18n.find_catalog()`
- `sphinx.util.i18n.find_catalog_files()`
- `sphinx.util.i18n.find_catalog_source_files()`

For more details, see *deprecation APIs list*.

Features added

- Add a helper class `sphinx.transforms.post_transforms.SphinxPostTransform`
- Add helper methods
  - `PythonDomain.note_module()`
  - `PythonDomain.note_object()`
  - `SphinxDirective.set_source_info()`
- #6180: Support `--keep-going` with BuildDoc setup command
- `math` directive now supports `:class:` option
- todo: `todo` directive now supports `:name:` option
- Enable override via environment of `SPHINXOPTS` and `SPHINXBUILD` Makefile variables (refs: #6232, #6303)
- #6287: autodoc: Unable to document bound instance methods exported as module functions
- #6289: autodoc: *`autodoc_default_options`* now supports `imported-members` option
- #4777: autodoc: Support coroutine
- #744: autodoc: Support abstractmethod
- #6325: autodoc: Support attributes in __slots__. For dict-style __slots__, autodoc considers values as a docstring of the attribute
- #6361: autodoc: Add *`autodoc_typehints`* to suppress typehints from signature
- #1063: autodoc: `automodule` directive now handles undocumented module level variables
- #6212 autosummary: Add *`autosummary_imported_members`* to display imported members on autosummary
- #6271: `make clean` is catastrophically broken if building into '.'
- #6363: Support `%O%` environment variable in make.bat
- #4777: py domain: Add `:async:` option to *`py:function`* directive
- py domain: Add new options to *`py:method`* directive
  - `:abstractmethod:`
  - `:async:`
  - `:classmethod:`
  - `:property:`
  - `:staticmethod:`
- rst domain: Add `directive:option` directive to describe the option for directive
- #6306: html: Add a label to search form for accessability purposes

- #4390: html: Consistent and semantic CSS for signatures
- #6358: The `rawsource` property of `production` nodes now contains the full production rule
- #6373: autosectionlabel: Allow suppression of warnings
- coverage: Support a new `coverage_ignore_pyobjects` option
- #6239: latex: Support to build Chinese documents

Bugs fixed

- #6230: Inappropriate node_id has been generated by glossary directive if term is consisted by non-ASCII characters
- #6213: ifconfig: contents after headings are not shown
- commented term in glossary directive is wrongly recognized
- #6299: rst domain: rst:directive directive generates waste space
- #6379: py domain: Module index (py-modindex.html) has duplicate titles
- #6331: man: invalid output when doctest follows rubric
- #6351: "Hyperlink target is not referenced" message is shown even if referenced
- #6165: autodoc: `tab_width` setting of docutils has been ignored
- #6347: autodoc: crashes with a plain Tuple on Python 3.6 and 3.5
- #6311: autosummary: autosummary table gets confused by complex type hints
- #6350: autosummary: confused by an argument having some kind of default value
- Generated Makefiles lack a final EOL (refs: #6232)
- #6375: extlinks: Cannot escape angle brackets in link caption
- #6378: linkcheck: Send commonly used User-Agent
- #6387: html search: failed to search document with haiku and scrolls themes
- #6408: html search: Fix the ranking of search results
- #6406: Wrong year is returned for `SOURCE_DATE_EPOCH`
- #6402: image directive crashes by unknown image format
- #6286: C++, allow 8 and 9 in hexadecimal integer literals.
- #6305: Fix the string in quickstart for 'path' argument of parser
- LaTeX: Figures in admonitions produced errors (refs: #6364)

10.34 Release 2.0.1 (released Apr 08, 2019)

Bugs fixed

- LaTeX: some system labels are not translated
- RemovedInSphinx30Warning is marked as pending
- deprecation warnings are not emitted
  - sphinx.application.CONFIG_FILENAME
  - sphinx.builders.htmlhelp

    - `viewcode_import`

- #6208: C++, properly parse full xrefs that happen to have a short xref as prefix
- #6220, #6225: napoleon: AttributeError is raised for raised section having references
- #6245: circular import error on importing SerializingHTMLBuilder
- #6243: LaTeX: 'releasename' setting for latex_elements is ignored
- #6244: html: Search function is broken with 3rd party themes
- #6263: html: HTML5Translator crashed with invalid field node
- #6262: html theme: The style of field lists has changed in bizstyle theme

10.35 Release 2.0.0 (released Mar 29, 2019)

Dependencies

2.0.0b1

- LaTeX builder now depends on TeX Live 2015 or above.
- LaTeX builder (with `'pdflatex'` *latex_engine*) will process Unicode Greek letters in text (not in math mark-up) via the text font and will not escape them to math mark-up. See the discussion of the `'fontenc'` key of *latex_elements*; such (optional) support for Greek adds, for example on Ubuntu xenial, the `texlive-lang-greek` and (if default font set-up is not modified) `cm-super(-minimal)` as additional Sphinx LaTeX requirements.
- LaTeX builder with *latex_engine* set to `'xelatex'` or to `'lualatex'` requires (by default) the `FreeFont` fonts, which in Ubuntu xenial are provided by package `fonts-freefont-otf`, and e.g. in Fedora 29 via package `texlive-gnu-freefont`.
- requests 2.5.0 or above
- The six package is no longer a dependency
- The sphinxcontrib-websupport package is no longer a dependency
- Some packages are separated to sub packages:
    - sphinxcontrib.applehelp
    - sphinxcontrib.devhelp
    - sphinxcontrib.htmlhelp
    - sphinxcontrib.jsmath
    - sphinxcontrib.serializinghtml
    - sphinxcontrib.qthelp

Incompatible changes

2.0.0b1

- Drop python 2.7 and 3.4 support
- Drop docutils 0.11 support
- Drop features and APIs deprecated in 1.7.x
- The default setting for *master_doc* is changed to `'index'` which has been longly used as default of sphinx-quickstart.
- LaTeX: Move message resources to `sphinxmessage.sty`
- LaTeX: Stop using `\captions<lang>` macro for some labels
- LaTeX: for `'xelatex'` and `'lualatex'`, use the `FreeFont` OpenType fonts as default choice (refs: #5645)
- LaTeX: `'xelatex'` and `'lualatex'` now use `\small` in code-blocks (due to `FreeMono` character width) like `'pdflatex'` already did (due to `Courier` character width). You may need to adjust this via *latex_elements* `'fvset'` key, in case of usage of some other OpenType fonts (refs: #5768)
- LaTeX: Greek letters in text are not escaped to math mode mark-up, and they will use the text font not the math font. The `LGR` font encoding must be added to the `'fontenc'` key of *latex_elements* for this to work (only if it is needed by the document, of course).
- LaTeX: setting the *language* to `'en'` triggered `Sonny` option of `fncychap`, now it is `Bjarne` to match case of no language specified. (refs: #5772)
- #5770: doctest: Follow *highlight_language* on highlighting doctest block. As a result, they are highlighted as python3 by default.
- The order of argument for `HTMLTranslator`, `HTML5Translator` and `ManualPageTranslator` are changed
- LaTeX: hard-coded redefinitions of `\l@section` and `\l@subsection` formerly done during loading of `'manual'` docclass get executed later, at time of `\sphinxtableofcontents`. This means that custom user definitions from LaTeX preamble now get overwritten. Use `\sphinxtableofcontentshook` to insert custom user definitions. See *Macros*.
- quickstart: Simplify generated `conf.py`
- #4148: quickstart: some questions are removed. They are still able to specify via command line options
- websupport: unbundled from sphinx core. Please use sphinxcontrib-websupport
- C++, the visibility of base classes is now always rendered as present in the input. That is, `private` is now shown, where it was ellided before.
- LaTeX: graphics inclusion of oversized images rescales to not exceed the text width and height, even if width and/or height option were used. (refs: #5956)
- epub: `epub_title` defaults to the *project* option
- #4550: All tables and figures without `align` option are displayed to center
- #4587: html: Output HTML5 by default

2.0.0b2

- texinfo: image files are copied into `name-figure` directory

Deprecated

2.0.0b1

- Support for evaluating Python 2 syntax is deprecated. This includes configuration files which should be converted to Python 3.
- The arguments of `EpubBuilder.build_mimetype()`, `EpubBuilder.build_container()`, `EpubBuilder.bulid_content()`, `EpubBuilder.build_toc()` and `EpubBuilder.build_epub()`
- The arguments of `Epub3Builder.build_navigation_doc()`
- The config variables
  - *html_experimental_html5_writer*
- The `encoding` argument of `autodoc.Documenter.get_doc()`, `autodoc.DocstringSignatureMixin.get_doc()`, `autodoc.DocstringSignatureMixin._find_signature()`, and `autodoc.ClassDocumenter.get_doc()` are deprecated.
- The `importer` argument of `sphinx.ext.autodoc.importer._MockModule`
- The `nodetype` argument of `sphinx.search.WordCollector. is_meta_keywords()`
- The `suffix` argument of `env.doc2path()` is deprecated.
- The string style `base` argument of `env.doc2path()` is deprecated.
- The fallback to allow omitting the `filename` argument from an overridden `IndexBuilder.feed()` method is deprecated.
- `sphinx.addnodes.abbreviation`
- `sphinx.application.Sphinx._setting_up_extension`
- `sphinx.builders.epub3.Epub3Builder.validate_config_value()`
- `sphinx.builders.html.SingleFileHTMLBuilder`
- `sphinx.builders.htmlhelp.HTMLHelpBuilder.open_file()`
- `sphinx.cmd.quickstart.term_decode()`
- `sphinx.cmd.quickstart.TERM_ENCODING`
- `sphinx.config.check_unicode()`
- `sphinx.config.string_classes`
- `sphinx.domains.cpp.DefinitionError.description`
- `sphinx.domains.cpp.NoOldIdError.description`
- `sphinx.domains.cpp.UnsupportedMultiCharacterCharLiteral.decoded`
- `sphinx.ext.autodoc.importer._MockImporter`
- `sphinx.ext.autosummary.Autosummary.warn()`
- `sphinx.ext.autosummary.Autosummary.genopt`
- `sphinx.ext.autosummary.Autosummary.warnings`
- `sphinx.ext.autosummary.Autosummary.result`
- `sphinx.ext.doctest.doctest_encode()`
- `sphinx.io.SphinxBaseFileInput`
- `sphinx.io.SphinxFileInput.supported`
- `sphinx.io.SphinxRSTFileInput`

- `sphinx.registry.SphinxComponentRegistry.add_source_input()`
- `sphinx.roles.abbr_role()`
- `sphinx.roles.emph_literal_role()`
- `sphinx.roles.menusel_role()`
- `sphinx.roles.index_role()`
- `sphinx.roles.indexmarkup_role()`
- `sphinx.testing.util.remove_unicode_literal()`
- `sphinx.util.attrdict`
- `sphinx.util.force_decode()`
- `sphinx.util.get_matching_docs()`
- `sphinx.util.inspect.Parameter`
- `sphinx.util.jsonimpl`
- `sphinx.util.osutil.EEXIST`
- `sphinx.util.osutil.EINVAL`
- `sphinx.util.osutil.ENOENT`
- `sphinx.util.osutil.EPIPE`
- `sphinx.util.osutil.walk()`
- `sphinx.util.PeekableIterator`
- `sphinx.util.pycompat.NoneType`
- `sphinx.util.pycompat.TextIOWrapper`
- `sphinx.util.pycompat.UnicodeMixin`
- `sphinx.util.pycompat.htmlescape`
- `sphinx.util.pycompat.indent`
- `sphinx.util.pycompat.sys_encoding`
- `sphinx.util.pycompat.terminal_safe()`
- `sphinx.util.pycompat.u`
- `sphinx.writers.latex.ExtBabel`
- `sphinx.writers.latex.LaTeXTranslator._make_visit_admonition()`
- `sphinx.writers.latex.LaTeXTranslator.babel_defmacro()`
- `sphinx.writers.latex.LaTeXTranslator.collect_footnotes()`
- `sphinx.writers.latex.LaTeXTranslator.generate_numfig_format()`
- `sphinx.writers.texinfo.TexinfoTranslator._make_visit_admonition()`
- `sphinx.writers.text.TextTranslator._make_depart_admonition()`
- template variables for LaTeX template
  - `logo`
  - `numfig_format`
  - `pageautorefname`
  - `translatablestrings`

For more details, see *deprecation APIs list*.

Features added

2.0.0b1

- #1618: The search results preview of generated HTML documentation is reader-friendlier: instead of showing the snippets as raw reStructuredText markup, Sphinx now renders the corresponding HTML. This means the Sphinx extension Sphinx: pretty search results[606] is no longer necessary. Note that changes to the search function of your custom or 3rd-party HTML template might overwrite this improvement.
- #4182: autodoc: Support `suppress_warnings`
- #5533: autodoc: `autodoc_default_options` supports `member-order`
- #5394: autodoc: Display readable names in type annotations for mocked objects
- #5459: autodoc: `autodoc_default_options` accepts `True` as a value
- #1148: autodoc: Add `autodecorator` directive for decorators
- #5635: autosummary: Add `autosummary_mock_imports` to mock external libraries on importing targets
- #4018: htmlhelp: Add `htmlhelp_file_suffix` and `htmlhelp_link_suffix`
- #5559: text: Support complex tables (colspan and rowspan)
- LaTeX: support rendering (not in math, yet) of Greek and Cyrillic Unicode letters in non-Cyrillic document even with `'pdflatex'` as `latex_engine` (refs: #5645)
- #5660: The `versionadded`, `versionchanged` and `deprecated` directives are now generated with their own specific CSS classes (`added`, `changed` and `deprecated`, respectively) in addition to the generic `versionmodified` class.
- #5841: apidoc: Add –extensions option to sphinx-apidoc
- #4981: C++, added an alias directive for inserting lists of declarations, that references existing declarations (e.g., for making a synopsis).
- C++: add `cpp:struct` to complement `cpp:class`.
- #1341 the HTML search considers words that contain a search term of length three or longer a match.
- #4611: epub: Show warning for duplicated ToC entries
- #1851: Allow to omit an argument for `code-block` directive. If omitted, it follows `highlight` or `highlight_language`
- #4587: html: Add `html4_writer` to use old HTML4 writer
- #6016: HTML search: A placeholder for the search summary prevents search result links from changing their position when the search terminates. This makes navigating search results easier.
- #5196: linkcheck also checks remote images exist
- #5924: githubpages: create CNAME file for custom domains when `html_baseurl` set
- #4261: autosectionlabel: restrict the labeled sections by new config value; `autosectionlabel_maxdepth`

[606] https://github.com/sphinx-contrib/sphinx-pretty-searchresults

Bugs fixed

2.0.0b1

- #1682: LaTeX: writer should not translate Greek unicode, but use textgreek package
- #5247: LaTeX: PDF does not build with default font config for Russian language and `'xelatex'` or `'lualatex'` as *latex_engine* (refs: #5251)
- #5248: LaTeX: Greek letters in section titles disappear from PDF bookmarks
- #5249: LaTeX: Unicode Greek letters in math directive break PDF build (fix requires extra set-up, see *latex_elements* `'textgreek'` key and/or *latex_engine* setting)
- #5772: LaTeX: should the Bjarne style of fncychap be used for English also if passed as language option?
- #5179: LaTeX: (lualatex only) escaping of > by `\textgreater{}` is not enough as `\textgreater{}\textgreater{}` applies TeX-ligature
- LaTeX: project name is not escaped if *latex_documents* omitted
- LaTeX: authors are not shown if *latex_documents* omitted
- HTML: Invalid HTML5 file is generated for a glossary having multiple terms for one description (refs: #4611)
- QtHelp: OS dependent path separator is used in .qhp file
- HTML search: search always returns nothing when multiple search terms are used and one term is shorter than three characters

2.0.0b2

- #6096: html: Anchor links are not added to figures
- #3620: html: Defer searchindex.js rather than loading it via ajax
- #6113: html: Table cells and list items have large margins
- #5508: `linenothreshold` option for `highlight` directive was ignored
- texinfo: `make install-info` causes syntax error
- texinfo: `make install-info` fails on macOS
- #3079: texinfo: image files are not copied on `make install-info`
- #5391: A cross reference in heading is rendered as literal
- #5946: C++, fix `cpp:alias` problems in LaTeX (and singlehtml)
- #6147: classes attribute of `citation_reference` node is lost
- AssertionError is raised when custom `citation_reference` node having classes attribute refers missing citation (refs: #6147)
- #2155: Support `code` directive
- C++, fix parsing of braced initializers.
- #6172: AttributeError is raised for old styled index nodes
- #4872: inheritance_diagram: correctly describe behavior of `parts` option in docs, allow negative values.
- #6178: i18n: Captions missing in translations for hidden TOCs

2.0.0 final

- #6196: py domain: unexpected prefix is generated

Testing

2.0.0b1

- Stop to use `SPHINX_TEST_TEMPDIR` envvar

2.0.0b2

- Add a helper function: `sphinx.testing.restructuredtext.parse()`

10.36 Release 1.8.5 (released Mar 10, 2019)

Bugs fixed

- LaTeX: Remove extraneous space after author names on PDF title page (refs: #6004)
- #6026: LaTeX: A cross reference to definition list does not work
- #6046: LaTeX: `TypeError` is raised when invalid latex_elements given
- #6067: LaTeX: images having a target are concatenated to next line
- #6067: LaTeX: images having a target are not aligned even if specified
- #6149: LaTeX: `:index:` role in titles causes `Use of \@icentercr doesn't match its definition` error on latexpdf build
- #6019: imgconverter: Including multipage PDF fails
- #6047: autodoc: `autofunction` emits a warning for method objects
- #6028: graphviz: Ensure the graphviz filenames are reproducible
- #6068: doctest: `skipif` option may remove the code block from documentation
- #6136: `:name:` option for `math` directive causes a crash
- #6139: intersphinx: ValueError on failure reporting
- #6135: changes: Fix UnboundLocalError when any module found
- #3859: manpage: code-block captions are not displayed correctly

10.37 Release 1.8.4 (released Feb 03, 2019)

Bugs fixed

- #3707: latex: no bold checkmark (✓) available.
- #5605: with the documentation language set to Chinese, English words could not be searched.
- #5889: LaTeX: user `numfig_format` is stripped of spaces and may cause build failure
- C++, fix hyperlinks for declarations involving east cv-qualifiers.
- #5755: C++, fix duplicate declaration error on function templates with constraints in the return type.
- C++, parse unary right fold expressions and binary fold expressions.
- pycode could not handle egg files on windows
- #5928: KeyError: 'DOCUTILSCONFIG' when running build
- #5936: LaTeX: PDF build broken by inclusion of image taller than page height in an admonition

- #5231: "make html" does not read and build "po" files in "locale" dir
- #5954: `:scale:` image option may break PDF build if image in an admonition
- #5966: mathjax has not been loaded on incremental build
- #5960: LaTeX: modified PDF layout since September 2018 TeXLive update of `parskip.sty`
- #5948: LaTeX: duplicated labels are generated for sections
- #5958: versionadded directive causes crash with Python 3.5.0
- #5995: autodoc: autodoc_mock_imports conflict with metaclass on Python 3.7
- #5871: texinfo: a section title `.` is not allowed

10.38 Release 1.8.3 (released Dec 26, 2018)

Features added

- LaTeX: it is possible to insert custom material to appear on back of title page, see discussion of `'maketitle'` key of *`latex_elements`* (`'manual'` docclass only)

Bugs fixed

- #5725: mathjax: Use CDN URL for "latest" version by default
- #5460: html search does not work with some 3rd party themes
- #5520: LaTeX, caption package incompatibility since Sphinx 1.6
- #5614: autodoc: incremental build is broken when builtin modules are imported
- #5627: qthelp: index.html missing in QtHelp
- #5659: linkcheck: crashes for a hyperlink containing multibyte character
- #5754: DOC: Fix some mistakes in *LaTeX customization*
- #5810: LaTeX: sphinxVerbatim requires explicit "hllines" set-up since 1.6.6 (refs: #1238)
- #5636: C++, fix parsing of floating point literals.
- #5496 (again): C++, fix assertion in partial builds with duplicates.
- #5724: quickstart: sphinx-quickstart fails when $LC_ALL is empty
- #1956: Default conf.py is not PEP8-compliant
- #5849: LaTeX: document class `\maketitle` is overwritten with no possibility to use original meaning in place of Sphinx custom one
- #5834: apidoc: wrong help for `--tocfile`
- #5800: todo: crashed if todo is defined in TextElement
- #5846: htmlhelp: convert hex escaping to decimal escaping in .hhc/.hhk files
- htmlhelp: broken .hhk file generated when title contains a double quote

10.39 Release 1.8.2 (released Nov 11, 2018)

Incompatible changes

- #5497: Do not include MathJax.js and jsmath.js unless it is really needed

Features added

- #5471: Show appropriate deprecation warnings

Bugs fixed

- #5490: latex: enumerated list causes a crash with recommonmark
- #5492: sphinx-build fails to build docs w/ Python < 3.5.2
- #3704: latex: wrong `\label` positioning for figures with a legend
- #5496: C++, fix assertion when a symbol is declared more than twice.
- #5493: gettext: crashed with broken template
- #5495: csv-table directive with file option in included file is broken (refs: #4821)
- #5498: autodoc: unable to find type hints for a `functools.partial`
- #5480: autodoc: unable to find type hints for unresolvable Forward references
- #5419: incompatible math_block node has been generated
- #5548: Fix ensuredir() in case of pre-existing file
- #5549: graphviz Correctly deal with non-existing static dir
- #3002: i18n: multiple footnote_references referring same footnote cause duplicated node_ids
- #5563: latex: footnote_references generated by extension causes a LaTeX builder crash
- #5561: make all-pdf fails with old xindy version
- #5557: quickstart: –no-batchfile isn't honored
- #3080: texinfo: multiline rubrics are broken
- #3080: texinfo: multiline citations are broken

10.40 Release 1.8.1 (released Sep 22, 2018)

Incompatible changes

- LaTeX `\pagestyle` commands have been moved to the LaTeX template. No changes in PDF, except possibly if `\sphinxtableofcontents`, which contained them, had been customized in `conf.py`. (refs: #5455)

Bugs fixed

- #5418: Incorrect default path for sphinx-build -d/doctrees files
- #5421: autodoc emits deprecation warning for *autodoc_default_flags*
- #5422: lambda object causes PicklingError on storing environment
- #5417: Sphinx fails to build with syntax error in Python 2.7.5
- #4911: add latexpdf to make.bat for non make-mode
- #5436: Autodoc does not work with enum subclasses with properties/methods
- #5437: autodoc: crashed on modules importing eggs
- #5433: latex: ImportError: cannot import name 'DEFAULT_SETTINGS'
- #5431: autodoc: `autofunction` emits a warning for callable objects
- #5457: Fix TypeError in error message when override is prohibited
- #5453: PDF builds of 'howto' documents have no page numbers
- #5463: mathbase: math_role and MathDirective was disappeared in 1.8.0
- #5454: latex: Index has disappeared from PDF for Japanese documents
- #5432: py domain: `:type:` field can't process `:term:` references
- #5426: py domain: TypeError has been raised for class attribute

10.41 Release 1.8.0 (released Sep 13, 2018)

Dependencies

1.8.0b1

- LaTeX: *latex_use_xindy*, if `True` (default for `xelatex/lualatex`), instructs `make latexpdf` to use **`xindy`** for general index. Make sure your LaTeX distribution includes it. (refs: #5134)
- LaTeX: `latexmk` is required for `make latexpdf` on Windows

Incompatible changes

1.8.0b2

- #5282: html theme: refer `pygments_style` settings of HTML themes preferentially
- The URL of download files are changed
- #5127: quickstart: `Makefile` and `make.bat` are not overwritten if exists

1.8.0b1

- #5156: the `sphinx.ext.graphviz:  extension runs `dot` in the directory of the document being built instead of in the root directory of the documentation.
- #4460: extensions which stores any data to environment should return the version of its env data structure as metadata. In detail, please see *Extension metadata*.
- Sphinx expects source parser modules to have supported file formats as `Parser.supported` attribute
- The default value of *epub_author* and *epub_publisher* are changed from `'unknown'` to the value of *author*. This is same as a `conf.py` file sphinx-build generates.

- The `gettext_compact` attribute is removed from `document.settings` object. Please use `config.gettext_compact` instead.
- The processing order on reading phase is changed. smart_quotes, sphinx domains, *`doctree-read`* event and versioning doctrees are invoked earlier than so far. For more details, please read a description of *`Sphinx.add_transform()`*
- #4827: All `substitution_definition` nodes are removed from doctree on reading phase
- `docutils.conf` in `$HOME` or `/etc` directories are ignored. Only `docutils.conf` from confdir is obeyed.
- #789: `:samp:` role supports to escape curly braces with backslash
- #4811: The files under *`html_static_path`* are excluded from source files.
- latex: Use `\sphinxcite` for citation references instead `\hyperref`
- The config value `viewcode_import` is renamed to *`viewcode_follow_imported_members`* (refs: #4035)
- #1857: latex: *`latex_show_pagerefs`* does not add pagerefs for citations
- #4648: latex: Now "rubric" elements are rendered as unnumbered section title
- #4983: html: The anchor for productionlist tokens has been changed
- Modifying a template variable `script_files` in templates is allowed now. Please use `app.add_js_file()` instead.
- #5072: Save environment object also with only new documents
- #5035: qthelp builder allows dashes in *`qthelp_namespace`*
- LaTeX: with lualatex or xelatex use by default **`xindy`** as UTF-8 able replacement of **`makeindex`** (refs: #5134). After upgrading Sphinx, please clean latex build repertory of existing project before new build.
- #5163: html: hlist items are now aligned to top
- `highlightlang` directive is processed on resolving phase
- #4000: LaTeX: template changed. Following elements moved to it:
  - `\begin{document}`
  - `shorthandoff` variable
  - `maketitle` variable
  - `tableofcontents` variable

Deprecated

1.8.0b2

- `sphinx.io.SphinxI18nReader.set_lineno_for_reporter()` is deprecated
- `sphinx.io.SphinxI18nReader.line` is deprecated
- `sphinx.util.i18n.find_catalog_source_file()` has changed; the *gettext_compact* argument has been deprecated
- #5403: `sphinx.util.images.guess_mimetype()` has changed; the *content* argument has been deprecated

1.8.0b1

- *`source_parsers`* is deprecated
- *`autodoc_default_flags`* is deprecated
- quickstart: `--epub` option becomes default, so it is deprecated
- Drop function based directive support. For now, Sphinx only supports class based directives (see `Directive`)

- `sphinx.util.docutils.directive_helper()` is deprecated
- `sphinx.cmdline` is deprecated
- `sphinx.make_mode` is deprecated
- `sphinx.locale.l_()` is deprecated
- #2157: helper function `warn()` for HTML themes is deprecated
- `app.override_domain()` is deprecated
- `app.add_stylesheet()` is deprecated
- `app.add_javascript()` is deprecated
- `app.import_object()` is deprecated
- `app.add_source_parser()` has changed; the *suffix* argument has been deprecated
- `sphinx.versioning.prepare()` is deprecated
- `Config.__init__()` has changed; the *dirname*, *filename* and *tags* argument has been deprecated
- `Config.check_types()` is deprecated
- `Config.check_unicode()` is deprecated
- `sphinx.application.CONFIG_FILENAME` is deprecated
- `highlightlang` directive is deprecated
- `BuildEnvironment.load()` is deprecated
- `BuildEnvironment.loads()` is deprecated
- `BuildEnvironment.frompickle()` is deprecated
- `env.read_doc()` is deprecated
- `env.update()` is deprecated
- `env._read_serial()` is deprecated
- `env._read_parallel()` is deprecated
- `env.write_doctree()` is deprecated
- `env._nitpick_ignore` is deprecated
- `env.versionchanges` is deprecated
- `env.dump()` is deprecated
- `env.dumps()` is deprecated
- `env.topickle()` is deprecated
- `env.note_versionchange()` is deprecated
- `sphinx.writers.latex.Table.caption_footnotetexts` is deprecated
- `sphinx.writers.latex.Table.header_footnotetexts` is deprecated
- `sphinx.writers.latex.LaTeXTranslator.footnotestack` is deprecated
- `sphinx.writers.latex.LaTeXTranslator.in_container_literal_block` is deprecated
- `sphinx.writers.latex.LaTeXTranslator.next_section_ids` is deprecated
- `sphinx.writers.latex.LaTeXTranslator.next_hyperlink_ids` is deprecated
- `sphinx.writers.latex.LaTeXTranslator.restrict_footnote()` is deprecated
- `sphinx.writers.latex.LaTeXTranslator.unrestrict_footnote()` is deprecated

- `sphinx.writers.latex.LaTeXTranslator.push_hyperlink_ids()` is deprecated
- `sphinx.writers.latex.LaTeXTranslator.pop_hyperlink_ids()` is deprecated
- `sphinx.writers.latex.LaTeXTranslator.check_latex_elements()` is deprecated
- `sphinx.writers.latex.LaTeXTranslator.bibitems` is deprecated
- `sphinx.writers.latex.LaTeXTranslator.hlsettingstack` is deprecated
- `sphinx.writers.latex.ExtBabel.get_shorthandoff()` is deprecated
- `sphinx.writers.html.HTMLTranslator.highlightlang` is deprecated
- `sphinx.writers.html.HTMLTranslator.highlightlang_base` is deprecated
- `sphinx.writers.html.HTMLTranslator.highlightlangopts` is deprecated
- `sphinx.writers.html.HTMLTranslator.highlightlinenothreshold` is deprecated
- `sphinx.writers.html5.HTMLTranslator.highlightlang` is deprecated
- `sphinx.writers.html5.HTMLTranslator.highlightlang_base` is deprecated
- `sphinx.writers.html5.HTMLTranslator.highlightlangopts` is deprecated
- `sphinx.writers.html5.HTMLTranslator.highlightlinenothreshold` is deprecated
- `sphinx.ext.mathbase` extension is deprecated
- `sphinx.ext.mathbase.math` node is deprecated
- `sphinx.ext.mathbase.displaymath` node is deprecated
- `sphinx.ext.mathbase.eqref` node is deprecated
- `sphinx.ext.mathbase.is_in_section_title()` is deprecated
- `sphinx.ext.mathbase.MathDomain` is deprecated
- `sphinx.ext.mathbase.MathDirective` is deprecated
- `sphinx.ext.mathbase.math_role` is deprecated
- `sphinx.ext.mathbase.setup_math()` is deprecated
- `sphinx.directives.other.VersionChanges` is deprecated
- `sphinx.highlighting.PygmentsBridge.unhighlight()` is deprecated
- `sphinx.ext.mathbase.get_node_equation_number()` is deprecated
- `sphinx.ext.mathbase.wrap_displaymath()` is deprecated
- The `trim_doctest_flags` argument of `sphinx.highlighting.PygmentsBridge` is deprecated

For more details, see deprecation APIs list[607]

Features added

1.8.0b2

- #5388: Ensure frozen object descriptions are reproducible
- #5362: apidoc: Add `--tocfile` option to change the filename of ToC

1.8.0b1

- Add *`config-inited`* event
- Add `sphinx.config.Any` to represent the config value accepts any type of value

[607] http://www.sphinx-doc.org/en/master/extdev/index.html#deprecated-apis

- *source_suffix* allows a mapping fileext to file types
- Add *author* as a configuration value
- #2852: imgconverter: Support to convert GIF to PNG
- `sphinx-build` command supports i18n console output
- Add `app.add_message_catalog()` and `sphinx.locale.get_translations()` to support translation for 3rd party extensions
- helper function `warning()` for HTML themes is added
- Add `Domain.enumerable_nodes` to manage own enumerable nodes for domains (experimental)
- Add a new keyword argument `override` to Application APIs
- LaTeX: new key `'fvset'` for *latex_elements*. For XeLaTeX/LuaLaTeX its default sets `fanvyvrb` to use normal, not small, fontsize in code-blocks (refs: #4793)
- Add *html_css_files* and *epub_css_files* for adding CSS files from configuration
- Add *html_js_files* for adding JS files from configuration
- #4834: Ensure set object descriptions are reproducible.
- #4828: Allow to override *numfig_format* partially. Full definition is not needed.
- Improve warning messages during including (refs: #4818)
- LaTeX: separate customizability of *guilabel* and *menuselection* (refs: #4830)
- Add `Config.read()` classmethod to create a new config object from configuration file
- #4866: Wrap graphviz diagrams in `<div>` tag
- viewcode: Add *viewcode-find-source* and *viewcode-follow-imported* to load source code without loading
- #4785: napoleon: Add strings to translation file for localisation
- #4927: Display a warning when invalid values are passed to linenothreshold option of highlight directive
- C++:
  - Add a `cpp:texpr` role as a sibling to `cpp:expr`.
  - Add support for unions.
  - #3593, #2683: add support for anonymous entities using names staring with `@`.
  - #5147: add support for (most) character literals.
  - Cross-referencing entities inside primary templates is supported, and now properly documented.
  - #1552: add new cross-referencing format for `cpp:any` and `cpp:func` roles, for referencing specific function overloads.
- #3606: MathJax should be loaded with async attribute
- html: Output `canonical_url` metadata if *html_baseurl* set (refs: #4193)
- #5029: autosummary: expose `inherited_members` to template
- #3784: mathjax: Add *mathjax_options* to give options to script tag for mathjax
- #726, #969: mathjax: Add *mathjax_config* to give in-line configurations for mathjax
- #4362: latex: Don't overwrite .tex file if document not changed
- #1431: latex: Add alphanumeric enumerated list support
- Add *latex_use_xindy* for UTF-8 savvy indexing, defaults to `True` if *latex_engine* is `'xelatex'` or `'lualatex'`. (refs: #5134, #5192, #5212)

- #4976: `SphinxLoggerAdapter.info()` now supports `location` parameter
- #5122: setuptools: support nitpicky option
- #2820: autoclass directive supports nested class
- Add `app.add_html_math_renderer()` to register a math renderer for HTML
- Apply *`trim_doctest_flags`* to all builders (cf. text, manpages)
- #5140: linkcheck: Add better Accept header to HTTP client
- #4614: sphinx-build: Add `--keep-going` option to show all warnings
- Add *`math:numref`* role to refer equations (Same as *`eq`*)
- quickstart: epub builder is enabled by default
- #5246: Add *`singlehtml_sidebars`* to configure sidebars for singlehtml builder
- #5273: doctest: Skip doctest conditionally
- #5306: autodoc: emit a warning for invalid typehints
- #4075, #5215: autodoc: Add *`autodoc_default_options`* which accepts option values as dict

Bugs fixed

1.8.0b2

- html: search box overrides to other elements if scrolled
- i18n: warnings for translation catalogs have wrong line numbers (refs: #5321)
- #5325: latex: cross references has been broken by multiply labeled objects
- C++, fixes for symbol addition and lookup. Lookup should no longer break in partial builds. See also #5337.
- #5348: download reference to remote file is not displayed
- #5282: html theme: `pygments_style` of theme was overridden by `conf.py` by default
- #4379: toctree shows confusing warning when document is excluded
- #2401: autodoc: `:members:` causes `:special-members:` not to be shown
- autodoc: ImportError is replaced by AttributeError for deeper module
- #2720, #4034: Incorrect links with `:download:`, duplicate names, and parallel builds
- #5290: autodoc: failed to analyze source code in egg package
- #5399: Sphinx crashes if unknown po file exists

1.8.0b1

- i18n: message catalogs were reset on each initialization
- #4850: latex: footnote inside footnote was not rendered
- #4945: i18n: fix lang_COUNTRY not fallback correctly for IndexBuilder. Thanks to Shengjing Zhu.
- #4983: productionlist directive generates invalid IDs for the tokens
- #5132: lualatex: PDF build fails if indexed word starts with Unicode character
- #5133: latex: index headings “Symbols” and “Numbers” not internationalized
- #5114: sphinx-build: Handle errors on scanning documents
- epub: spine has been broken when “self” is listed on toctree (refs: #4611)
- #344: autosummary does not understand docstring of module level attributes

- #5191: C++, prevent nested declarations in functions to avoid lookup problems.
- #5126: C++, add missing isPack method for certain template parameter types.
- #5187: C++, parse attributes on declarators as well.
- C++, parse delete expressions and basic new expressions as well.
- #5002: graphviz: SVGs do not adapt to the column width

Features removed

1.8.0b1

- `sphinx.ext.pngmath` extension

Documentation

1.8.0b1

- #5083: Fix wrong make.bat option for internationalization.
- #5115: napoleon: add admonitions added by #4613 to the docs.

10.42 Release 1.7.9 (released Sep 05, 2018)

Features added

- #5359: Make generated texinfo files reproducible by sorting the anchors

Bugs fixed

- #5361: crashed on incremental build if document uses include directive

10.43 Release 1.7.8 (released Aug 29, 2018)

Incompatible changes

- The type of `env.included` has been changed to dict of set

Bugs fixed

- #5320: intersphinx: crashed if invalid url given
- #5326: manpage: crashed when invalid docname is specified as `man_pages`
- #5322: autodoc: `Any` typehint causes formatting error
- #5327: “document isn’t included in any toctree” warning on rebuild with generated files
- #5335: quickstart: escape sequence has been displayed with MacPorts’ python

10.44 Release 1.7.7 (released Aug 19, 2018)

Bugs fixed

- #5198: document not in toctree warning when including files only for parallel builds
- LaTeX: reduce "Token not allowed in a PDF string" hyperref warnings in latex console output (refs: #5236)
- LaTeX: suppress "remreset Warning: The remreset package is obsolete" in latex console output with recent LaTeX (refs: #5237)
- #5234: PDF output: usage of PAPER environment variable is broken since Sphinx 1.5
- LaTeX: fix the *latex_engine* documentation regarding Latin Modern font with XeLaTeX/LuaLateX (refs: #5251)
- #5280: autodoc: Fix wrong type annotations for complex typing
- autodoc: Optional types are wrongly rendered
- #5291: autodoc crashed by ForwardRef types
- #5211: autodoc: No docs generated for functools.partial functions
- #5306: autodoc: `getargspec()` raises NameError for invalid typehints
- #5298: imgmath: math_number_all causes equations to have two numbers in html
- #5294: sphinx-quickstart blank prompts in PowerShell

10.45 Release 1.7.6 (released Jul 17, 2018)

Bugs fixed

- #5037: LaTeX `\sphinxupquote{}` breaks in Russian
- sphinx.testing uses deprecated pytest API; `Node.get_marker(name)`
- #5016: crashed when recommonmark.AutoStrictify is enabled
- #5022: latex: crashed with docutils package provided by Debian/Ubuntu
- #5009: latex: a label for table is vanished if table does not have a caption
- #5048: crashed with numbered toctree
- #2410: C, render empty argument lists for macros.
- C++, fix lookup of full template specializations with no template arguments.
- #4667: C++, fix assertion on missing references in global scope when using intersphinx. Thanks to Alan M. Carroll.
- #5019: autodoc: crashed by Form Feed Character
- #5032: autodoc: loses the first staticmethod parameter for old styled classes
- #5036: quickstart: Typing Ctrl-U clears the whole of line
- #5066: html: "relations" sidebar is not shown by default
- #5091: latex: curly braces in index entries are not handled correctly
- #5070: epub: Wrong internal href fragment links
- #5104: apidoc: Interface of `sphinx.apidoc:main()` has changed

- #4272: PDF builds of French projects have issues with XeTeX
- #5076: napoleon raises RuntimeError with python 3.7
- #5125: sphinx-build: Interface of `sphinx:main()` has changed
- sphinx-build: `sphinx.cmd.build.main()` refers `sys.argv` instead of given argument
- #5146: autosummary: warning is emitted when the first line of docstring ends with literal notation
- autosummary: warnings of autosummary indicates wrong location (refs: #5146)
- #5143: autodoc: crashed on inspecting dict like object which does not support sorting
- #5139: autodoc: Enum argument missing if it shares value with another
- #4946: py domain: rtype field could not handle "None" as a type
- #5176: LaTeX: indexing of terms containing `@`, `!`, or `"` fails
- #5161: html: crashes if copying static files are failed
- #5167: autodoc: Fix formatting type annotations for tuples with more than two arguments
- #3329: i18n: crashed by auto-symbol footnote references
- #5158: autosummary: module summary has been broken when it starts with heading

10.46 Release 1.7.5 (released May 29, 2018)

Bugs fixed

- #4924: html search: Upper characters problem in any other languages
- #4932: apidoc: some subpackage is ignored if sibling subpackage contains a module starting with underscore
- #4863, #4938, #4939: i18n doesn't handle correctly node.title as used for contents, topic, admonition, table and section.
- #4913: i18n: literal blocks in bullet list are not translated
- #4962: C++, raised TypeError on duplicate declaration.
- #4825: C++, properly parse expr roles and give better error messages when (escaped) line breaks are present.
- C++, properly use `desc_addname` nodes for prefixes of names.
- C++, parse pack expansions in function calls.
- #4915, #4916: links on search page are broken when using dirhtml builder
- #4969: autodoc: constructor method should not have return annotation
- latex: deeply nested enumerated list which is beginning with non-1 causes LaTeX engine crashed
- #4978: latex: shorthandoff is not set up for Brazil locale
- #4928: i18n: Ignore dot-directories like .git/ in LC_MESSAGES/
- #4946: py domain: type field could not handle "None" as a type
- #4979: latex: Incorrect escaping of curly braces in index entries
- #4956: autodoc: Failed to extract document from a subclass of the class on mocked module
- #4973: latex: glossary directive adds whitespace to each item
- #4980: latex: Explicit labels on code blocks are duplicated
- #4919: node.asdom() crashes if toctree has :numbered: option

- #4914: autodoc: Parsing error when using dataclasses without default values
- #4931: autodoc: crashed when handler for autodoc-skip-member raises an error
- #4931: autodoc: crashed when subclass of mocked class are processed by napoleon module
- #5007: sphinx-build crashes when error log contains a "%" character

10.47 Release 1.7.4 (released Apr 25, 2018)

Bugs fixed

- #4885, #4887: domains: Crashed with duplicated objects
- #4889: latex: sphinx.writers.latex causes recursive import

10.48 Release 1.7.3 (released Apr 23, 2018)

Bugs fixed

- #4769: autodoc loses the first staticmethod parameter
- #4790: autosummary: too wide two column tables in PDF builds
- #4795: Latex customization via `_templates/longtable.tex_t` is broken
- #4789: imgconverter: confused by convert.exe of Windows
- #4783: On windows, Sphinx crashed when drives of srcdir and outdir are different
- #4812: autodoc ignores type annotated variables
- #4817: wrong URLs on warning messages
- #4784: latex: *`latex_show_urls`* assigns incorrect footnote numbers if hyperlinks exists inside substitutions
- #4837: latex with class memoir Error: Font command `\sf` is not supported
- #4803: latex: too slow in proportion to number of auto numbered footnotes
- #4838: htmlhelp: The entries in .hhp file is not ordered
- toctree directive tries to glob for URL having query_string
- #4871: html search: Upper characters problem in German
- #4717: latex: Compilation for German docs failed with LuaLaTeX and XeLaTeX
- #4459: duplicated labels detector does not work well in parallel build
- #4878: Crashed with extension which returns invalid metadata

10.49 Release 1.7.2 (released Mar 21, 2018)

Incompatible changes

- #4520: apidoc: folders with an empty __init__.py are no longer excluded from TOC

Bugs fixed

- #4669: sphinx.build_main and sphinx.make_main throw NameError
- #4685: autosummary emits meaningless warnings
- autodoc: crashed when invalid options given
- pydomain: always strip parenthesis if empty (refs: #1042)
- #4689: autosummary: unexpectedly strips docstrings containing "i.e."
- #4701: viewcode: Misplaced `<div>` in viewcode html output
- #4444: Don't require numfig to use :numref: on sections
- #4727: Option clash for package textcomp
- #4725: Sphinx does not work with python 3.5.0 and 3.5.1
- #4716: Generation PDF file with TexLive on Windows, file not found error
- #4574: vertical space before equation in latex
- #4720: message when an image is mismatched for builder is not clear
- #4655, #4684: Incomplete localization strings in Polish and Chinese
- #2286: Sphinx crashes when error is happens in rendering HTML pages
- #4688: Error to download remote images having long URL
- #4754: sphinx/pycode/__init__.py raises AttributeError
- #1435: qthelp builder should htmlescape keywords
- epub: Fix docTitle elements of toc.ncx is not escaped
- #4520: apidoc: Subpackage not in toc (introduced in 1.6.6) now fixed
- #4767: html: search highlighting breaks mathjax equations

10.50 Release 1.7.1 (released Feb 23, 2018)

Deprecated

- #4623: `sphinx.build_main()` is deprecated.
- autosummary: The interface of `sphinx.ext.autosummary.get_documenter()` has been changed (Since 1.7.0)
- #4664: `sphinx.ext.intersphinx.debug()` is deprecated.

For more details, see deprecation APIs list[608]

[608] http://www.sphinx-doc.org/en/master/extdev/index.html#deprecated-apis

Bugs fixed

- #4608: epub: Invalid meta tag is generated
- #4260: autodoc: keyword only argument separator is not disappeared if it is appeared at top of the argument list
- #4622: epub: *epub_scheme* does not effect to content.opf
- #4627: graphviz: Fit graphviz images to page
- #4617: quickstart: PROJECT_DIR argument is required
- #4623: sphinx.build_main no longer exists in 1.7.0
- #4615: The argument of `sphinx.build` has been changed in 1.7.0
- autosummary: The interface of `sphinx.ext.autosummary.get_documenter()` has been changed
- #4630: Have order on msgids in sphinx.pot deterministic
- #4563: autosummary: Incorrect end of line punctuation detection
- #4577: Enumerated sublists with explicit start with wrong number
- #4641: A external link in TOC cannot contain "?" with `:glob:` option
- C++, add missing parsing of explicit casts and typeid in expression parsing.
- C++, add missing parsing of `this` in expression parsing.
- #4655: Fix incomplete localization strings in Polish
- #4653: Fix error reporting for parameterless ImportErrors
- #4664: Reading objects.inv fails again
- #4662: `any` refs with `term` targets crash when an ambiguity is encountered

10.51 Release 1.7.0 (released Feb 12, 2018)

Dependencies

1.7.0b1

- Add `packaging` package

Incompatible changes

1.7.0b1

- #3668: The arguments has changed of main functions for each command
- #3893: Unknown html_theme_options throw warnings instead of errors
- #3927: Python parameter/variable types should match classes, not all objects
- #3962: sphinx-apidoc now recognizes given directory as an implicit namespace package when `--implicit-namespaces` option given, not subdirectories of given directory.
- #3929: apidoc: Move sphinx.apidoc to sphinx.ext.apidoc
- #4226: apidoc: Generate new style makefile (make-mode)
- #4274: sphinx-build returns 2 as an exit code on argument error
- #4389: output directory will be created after loading extensions

- autodoc does not generate warnings messages to the generated document even if *keep_warnings* is True. They are only emitted to stderr.
- shebang line is removed from generated conf.py
- #2557: autodoc: *autodoc_mock_imports* only mocks specified modules with their descendants. It does not mock their ancestors. If you want to mock them, please specify the name of ancestors explicitly.
- #3620: html theme: move DOCUMENTATION_OPTIONS to independent JavaScript file (refs: #4295)
- #4246: Limit width of text body for all themes. Configurable via theme options `body_min_width` and `body_max_width`.
- #4771: apidoc: The `exclude_patterns` arguments are ignored if they are placed just after command line options

1.7.0b2

- #4467: html theme: Rename `csss` block to `css`

Deprecated

1.7.0b1

- using a string value for *html_sidebars* is deprecated and only list values will be accepted at 2.0.
- `format_annotation()` and `formatargspec()` is deprecated. Please use `sphinx.util.inspect.Signature` instead.
- `sphinx.ext.autodoc.AutodocReporter` is replaced by `sphinx.util.docutils.switch_source_input()` and now deprecated. It will be removed in Sphinx-2.0.
- `sphinx.ext.autodoc.add_documenter()` and `AutoDirective._register` is now deprecated. Please use `app.add_autodocumenter()` instead.
- `AutoDirective._special_attrgetters` is now deprecated. Please use `app.add_autodoc_attrgetter()` instead.

Features added

1.7.0b1

- C++, handle `decltype(auto)`.
- #2406: C++, add proper parsing of expressions, including linking of identifiers.
- C++, add a `cpp:expr` role for inserting inline C++ expressions or types.
- C++, support explicit member instantiations with shorthand `template` prefix
- C++, make function parameters linkable, like template params.
- #3638: Allow to change a label of reference to equation using `math_eqref_format`
- Now *suppress_warnings* accepts following configurations:
  - `ref.python` (ref: #3866)
- #3872: Add latex key to configure literal blocks caption position in PDF output (refs #3792, #1723)
- In case of missing docstring try to retrieve doc from base classes (ref: #3140)
- #4023: Clarify error message when any role has more than one target.
- #3973: epub: allow to override build date
- #3972: epub: Sort manifest entries by filename

- #4052: viewcode: Sort before highlighting module code
- #1448: qthelp: Add new config value; *qthelp_namespace*
- #4140: html themes: Make body tag inheritable
- #4168: improve zh search with jieba
- HTML themes can set up default sidebars through `theme.conf`
- #3160: html: Use <`kdb`> to represent `:kbd:` role
- #4212: autosummary: catch all exceptions when importing modules
- #4166: Add *math_numfig* for equation numbering by section (refs: #3991, #4080). Thanks to Oliver Jahn.
- #4311: Let LaTeX obey *numfig_secnum_depth* for figures, tables, and code-blocks
- #947: autodoc now supports ignore-module-all to ignore a module's `__all__`
- #4332: Let LaTeX obey *math_numfig* for equation numbering
- #4093: sphinx-build creates empty directories for unknown targets/builders
- Add `top-classes` option for the `sphinx.ext.inheritance_diagram` extension to limit the scope of inheritance graphs.
- #4183: doctest: `:pyversion:` option also follows PEP-440 specification
- #4235: html: Add *manpages_url* to make manpage roles to hyperlinks
- #3570: autodoc: Do not display 'typing.' module for type hints
- #4354: sphinx-build now emits finish message. Builders can modify it through `Builder.epilog` attribute
- #4245: html themes: Add `language` to javascript vars list
- #4079: html: Add `notranslate` class to each code-blocks, literals and maths to let Google Translate know they are not translatable
- #4137: doctest: doctest block is always highlighted as python console (pycon)
- #4137: doctest: testcode block is always highlighted as python
- #3998: text: Assign section numbers by default. You can control it using *text_add_secnumbers* and *text_secnumber_suffix*

1.7.0b2

- #4271: sphinx-build supports an option called `-j auto` to adjust numbers of processes automatically.
- Napoleon: added option to specify custom section tags.

Features removed

1.7.0b1

- Configuration variables
  - html_use_smartypants
  - latex_keep_old_macro_names
  - latex_elements['footer']
- utility methods of `sphinx.application.Sphinx` class
  - buildername (property)
  - _display_chunk()
  - old_status_iterator()

  - status_iterator()
  - _directive_helper()
- utility methods of `sphinx.environment.BuildEnvironment` class
  - currmodule (property)
  - currclass (property)
- epub2 builder
- prefix and colorfunc parameter for warn()
- `sphinx.util.compat` module
- `sphinx.util.nodes.process_only_nodes()`
- LaTeX environment `notice`, use `sphinxadmonition` instead
- LaTeX `\sphinxstylethead`, use `\sphinxstyletheadfamily`
- C++, support of function concepts. Thanks to mickk-on-cpp.
- Not used and previously not documented LaTeX macros `\shortversion` and `\setshortversion`

Bugs fixed

1.7.0b1

- #3882: Update the order of files for HTMLHelp and QTHelp
- #3962: sphinx-apidoc does not recognize implicit namespace packages correctly
- #4094: C++, allow empty template argument lists.
- C++, also hyperlink types in the name of declarations with qualified names.
- C++, do not add index entries for declarations inside concepts.
- C++, support the template disambiguator for dependent names.
- #4314: For PDF 'howto' documents, numbering of code-blocks differs from the one of figures and tables
- #4330: PDF 'howto' documents have an incoherent default LaTeX tocdepth counter setting
- #4198: autosummary emits multiple 'autodoc-process-docstring' event. Thanks to Joel Nothman.
- #4081: Warnings and errors colored the same when building
- latex: Do not display 'Release' label if *`release`* is not set

1.7.0b2

- #4415: autodoc classifies inherited classmethods as regular methods
- #4415: autodoc classifies inherited staticmethods as regular methods
- #4472: DOCUMENTATION_OPTIONS is not defined
- #4491: autodoc: prefer _MockImporter over other importers in sys.meta_path
- #4490: autodoc: type annotation is broken with python 3.7.0a4+
- utils package is no longer installed
- #3952: apidoc: module header is too escaped
- #4275: Formats accepted by sphinx.util.i18n.format_date are limited
- #4493: recommonmark raises AttributeError if AutoStructify enabled
- #4209: intersphinx: In link title, "v" should be optional if target has no version

- #4230: slowdown in writing pages with sphinx 1.6
- #4522: epub: document is not rebuilt even if config changed

1.7.0b3

- #4019: inheritance_diagram AttributeError stopping make process
- #4531: autosummary: methods are not treated as attributes
- #4538: autodoc: `sphinx.ext.autodoc.Options` has been moved
- #4539: autodoc emits warnings for partialmethods
- #4223: doctest: failing tests reported in wrong file, at wrong line
- i18n: message catalogs are not compiled if specific filenames are given for `sphinx-build` as arguments (refs: #4560)
- #4027: sphinx.ext.autosectionlabel now expects labels to be the same as they are in the raw source; no smart quotes, nothig fancy.
- #4581: apidoc: Excluded modules still included

Testing

1.7.0b1

- Add support for docutils 0.14
- Add tests for the `sphinx.ext.inheritance_diagram` extension.

10.52 Release 1.6.7 (released Feb 04, 2018)

Bugs fixed

- #1922: html search: Upper characters problem in French
- #4412: Updated jQuery version from 3.1.0 to 3.2.1
- #4438: math: math with labels with whitespace cause html error
- #2437: make full reference for classes, aliased with "alias of"
- #4434: pure numbers as link targets produce warning
- #4477: Build fails after building specific files
- #4449: apidoc: include "empty" packages that contain modules
- #3917: citation labels are transformed to ellipsis
- #4501: graphviz: epub3 validation error caused if graph is not clickable
- #4514: graphviz: workaround for wrong map ID which graphviz generates
- #4525: autosectionlabel does not support parallel build
- #3953: Do not raise warning when there is a working intersphinx inventory
- #4487: math: ValueError is raised on parallel build. Thanks to jschueller.
- #2372: autosummary: invalid signatures are shown for type annotated functions
- #3942: html: table is not aligned to center even if `:align:  center`

10.53 Release 1.6.6 (released Jan 08, 2018)

Features added

- #4181: autodoc: Sort dictionary keys when possible
- `VerbatimHighlightColor` is a new *LaTeX 'sphinxsetup'* key (refs: #4285)
- Easier customizability of LaTeX macros involved in rendering of code-blocks
- Show traceback if conf.py raises an exception (refs: #4369)
- Add *`smartquotes`* to disable smart quotes through `conf.py` (refs: #3967)
- Add *`smartquotes_action`* and *`smartquotes_excludes`* (refs: #4142, #4357)

Bugs fixed

- #4334: sphinx-apidoc: Don't generate references to non-existing files in TOC
- #4206: latex: reST label between paragraphs loses paragraph break
- #4231: html: Apply fixFirefoxAnchorBug only under Firefox
- #4221: napoleon depends on autodoc, but users need to load it manually
- #2298: automodule fails to document a class attribute
- #4099: C++: properly link class reference to class from inside constructor
- #4267: PDF build broken by Unicode U+2116 NUMERO SIGN character
- #4249: PDF output: Pygments error highlighting increases line spacing in code blocks
- #1238: Support `:emphasize-lines:` in PDF output
- #4279: Sphinx crashes with pickling error when run with multiple processes and remote image
- #1421: Respect the quiet flag in sphinx-quickstart
- #4281: Race conditions when creating output directory
- #4315: For PDF 'howto' documents, `latex_toplevel_sectioning='part'` generates `\chapter` commands
- #4214: Two todolist directives break sphinx-1.6.5
- Fix links to external option docs with intersphinx (refs: #3769)
- #4091: Private members not documented without :undoc-members:

10.54 Release 1.6.5 (released Oct 23, 2017)

Features added

- #4107: Make searchtools.js compatible with pre-Sphinx1.5 templates
- #4112: Don't override the smart_quotes setting if it was already set
- #4125: Display reference texts of original and translated passages on i18n warning message
- #4147: Include the exception when logging PO/MO file read/write

Bugs fixed

- #4085: Failed PDF build from image in parsed-literal using `:align:` option
- #4100: Remove debug print from autodoc extension
- #3987: Changing theme from alabaster causes HTML build to fail
- #4096: C++, don't crash when using the wrong role type. Thanks to mitya57.
- #4070, #4111: crashes when the warning message contains format strings (again)
- #4108: Search word highlighting breaks SVG images
- #3692: Unable to build HTML if writing .buildinfo failed
- #4152: HTML writer crashes if a field list is placed on top of the document
- #4063: Sphinx crashes when labeling directive `.. todolist::`
- #4134: [doc] `docutils.conf` is not documented explicitly
- #4169: Chinese language doesn't trigger Chinese search automatically
- #1020: ext.todo todolist not linking to the page in pdflatex
- #3965: New quickstart generates wrong SPHINXBUILD in Makefile
- #3739: `:module:` option is ignored at content of pyobjects
- #4149: Documentation: Help choosing *`latex_engine`*
- #4090: [doc] *`latex_additional_files`* with extra LaTeX macros should not use `.tex` extension
- Failed to convert reST parser error to warning (refs: #4132)

10.55 Release 1.6.4 (released Sep 26, 2017)

Features added

- #3926: Add `autodoc_warningiserror` to suppress the behavior of `-W` option during importing target modules on autodoc

Bugs fixed

- #3924: docname lost after dynamically parsing RST in extension
- #3946: Typo in sphinx.sty (this was a bug with no effect in default context)
- **pep** and :rfc: does not supports `default-role` directive (refs: #3960)
- #3960: default_role = 'guilabel' not functioning
- Missing `texinputs_win/Makefile` to be used in latexpdf builder on windows.
- #4026: nature: Fix macOS Safari scrollbar color
- #3877: Fix for C++ multiline signatures.
- #4006: Fix crash on parallel build
- #3969: private instance attributes causes AttributeError
- #4041: C++, remove extra name linking in function pointers.
- #4038: C, add missing documentation of `member` role.

- #4044: An empty multicolumn cell causes extra row height in PDF output
- #4049: Fix typo in output of sphinx-build -h
- #4062: hashlib.sha1() must take bytes, not unicode on Python 3
- Avoid indent after index entries in latex (refs: #4066)
- #4070: crashes when the warning message contains format strings
- #4067: Return non-zero exit status when make subprocess fails
- #4055: graphviz: the :align: option does not work for SVG output
- #4055: graphviz: the :align: center option does not work for latex output
- #4051: `warn()` function for HTML theme outputs 'None' string

10.56 Release 1.6.3 (released Jul 02, 2017)

Features added

- latex: hint that code-block continues on next page (refs: #3764, #3792)

Bugs fixed

- #3821: Failed to import sphinx.util.compat with docutils-0.14rc1
- #3829: sphinx-quickstart template is incomplete regarding use of alabaster
- #3772: 'str object' has no attribute 'filename'
- Emit wrong warnings if citation label includes hyphens (refs: #3565)
- #3858: Some warnings are not colored when using –color option
- #3775: Remove unwanted whitespace in default template
- #3835: sphinx.ext.imgmath fails to convert SVG images if project directory name contains spaces
- #3850: Fix color handling in make mode's help command
- #3865: use of self.env.warn in sphinx extension fails
- #3824: production lists apply smart quotes transform since Sphinx 1.6.1
- latex: fix `\sphinxbfcode` swallows initial space of argument
- #3878: Quotes in auto-documented class attributes should be straight quotes in PDF output
- #3881: LaTeX figure floated to next page sometimes leaves extra vertical whitespace
- #3885: duplicated footnotes raises IndexError
- #3873: Failure of deprecation warning mechanism of `sphinx.util.compat.Directive`
- #3874: Bogus warnings for "citation not referenced" for cross-file citations
- #3860: Don't download images when builders not supported images
- #3860: Remote image URIs without filename break builders not supported remote images
- #3833: command line messages are translated unintentionally with `language` setting.
- #3840: make checking `epub_uid` strict
- #3851, #3706: Fix about box drawing characters for PDF output

- #3900: autosummary could not find methods
- #3902: Emit error if `latex_documents` contains non-unicode string in py2

10.57 Release 1.6.2 (released May 28, 2017)

Incompatible changes

- #3789: Do not require typing module for python>=3.5

Bugs fixed

- #3754: HTML builder crashes if HTML theme appends own stylesheets
- #3756: epub: Entity 'mdash' not defined
- #3758: Sphinx crashed if logs are emitted in conf.py
- #3755: incorrectly warns about dedent with literalinclude
- #3742: RTD[609] PDF builds of Sphinx own docs are missing an index entry in the bookmarks and table of contents. This is rtfd/readthedocs.org#2857[610] issue, a workaround is obtained using some extra LaTeX code in Sphinx's own `conf.py`
- #3770: Build fails when a "code-block" has the option emphasize-lines and the number indicated is higher than the number of lines
- #3774: Incremental HTML building broken when using citations
- #3763: got epubcheck validations error if epub_cover is set
- #3779: 'ImportError' in sphinx.ext.autodoc due to broken 'sys.meta_path'. Thanks to Tatiana Tereshchenko.
- #3796: env.resolve_references() crashes when non-document node given
- #3803: Sphinx crashes with invalid PO files
- #3791: PDF "continued on next page" for long tables isn't internationalized
- #3788: smartquotes emits warnings for unsupported languages
- #3807: latex Makefile for `make latexpdf` is only for unixen
- #3781: double hyphens in option directive are compiled as endashes
- #3817: latex builder raises AttributeError

10.58 Release 1.6.1 (released May 16, 2017)

Dependencies

1.6b1

- (updated) latex output is tested with Ubuntu trusty's texlive packages (Feb. 2014) and earlier tex installations may not be fully compliant, particularly regarding Unicode engines xelatex and lualatex
- (added) latexmk is required for `make latexpdf` on GNU/Linux and Mac OS X (refs: #3082)

[609] https://readthedocs.org/
[610] https://github.com/rtfd/readthedocs.org/issues/2857

Incompatible changes

1.6b1

- #1061, #2336, #3235: Now generation of autosummary doesn't contain imported members by default. Thanks to Luc Saffre.
- LaTeX `\includegraphics` command isn't overloaded: only `\sphinxincludegraphics` has the custom code to fit image to available width if oversized.
- The subclasses of `sphinx.domains.Index` should override `generate()` method. The default implementation raises NotImplementedError
- LaTeX positioned long tables horizontally centered, and short ones flushed left (no text flow around table.) The position now defaults to center in both cases, and it will obey Docutils 0.13 `:align:` option (refs #3415, #3377)
- option directive also allows all punctuations for the option name (refs: #3366)
- #3413: if *literalinclude*'s `:start-after:` is used, make `:lines:` relative (refs #3412)
- `literalinclude` directive does not allow the combination of `:diff:` option and other options (refs: #3416)
- LuaLaTeX engine uses `fontspec` like XeLaTeX. It is advised `latex_engine = 'lualatex'` be used only on up-to-date TeX installs (refs #3070, #3466)
- `latex_keep_old_macro_names` default value has been changed from `True` to `False`. This means that some LaTeX macros for styling are by default defined only with `\sphinx..` prefixed names. (refs: #3429)
- Footer "Continued on next page" of LaTeX longtable's now not framed (refs: #3497)
- #3529: The arguments of `BuildEnvironment.__init__` is changed
- #3082: Use latexmk for pdf (and dvi) targets (Unix-like platforms only)
- #3558: Emit warnings if footnotes and citations are not referenced. The warnings can be suppressed by `suppress_warnings`.
- latex made available (non documented) colour macros from a file distributed with pdftex engine for Plain TeX. This is removed in order to provide better support for multiple TeX engines. Only interface from `color` or `xcolor` packages should be used by extensions of Sphinx latex writer. (refs #3550)
- `Builder.env` is not filled at instantiation
- #3594: LaTeX: single raw directive has been considered as block level element
- #3639: If `html_experimental_html5_writer` is available, epub builder use it by default.
- `Sphinx.add_source_parser()` raises an error if duplicated

1.6b2

- #3345: Replace the custom smartypants code with Docutils' smart_quotes. Thanks to Dmitry Shachnev, and to Günter Milde at Docutils.

1.6b3

- LaTeX package `eqparbox` is not used and not loaded by Sphinx anymore
- LaTeX package `multirow` is not used and not loaded by Sphinx anymore
- Add line numbers to citation data in std domain

1.6 final

- LaTeX package `threeparttable` is not used and not loaded by Sphinx anymore (refs #3686, #3532, #3377)

Features removed

- Configuration variables
  - epub3_contributor
  - epub3_description
  - epub3_page_progression_direction
  - html_translator_class
  - html_use_modindex
  - latex_font_size
  - latex_paper_size
  - latex_preamble
  - latex_use_modindex
  - latex_use_parts
- `termsep` node
- defindex.html template
- LDML format support in `today`, `today_fmt` and `html_last_updated_fmt`
- `:inline:` option for the directives of sphinx.ext.graphviz extension
- sphinx.ext.pngmath extension
- `sphinx.util.compat.make_admonition()`

Features added

1.6b1

- #3136: Add `:name:` option to the directives in `sphinx.ext.graphviz`
- #2336: Add `imported_members` option to `sphinx-autogen` command to document imported members.
- C++, add `:tparam-line-spec:` option to templated declarations. When specified, each template parameter will be rendered on a separate line.
- #3359: Allow sphinx.js in a user locale dir to override sphinx.js from Sphinx
- #3303: Add `:pyversion:` option to the doctest directive.
- #3378: (latex) support for `:widths:` option of table directives (refs: #3379, #3381)
- #3402: Allow to suppress "download file not readable" warnings using *`suppress_warnings`*.
- #3377: latex: Add support for Docutils 0.13 `:align:` option for tables (but does not implement text flow around table).
- latex: footnotes from inside tables are hyperlinked (except from captions or headers) (refs: #3422)
- Emit warning if over dedent has detected on `literalinclude` directive (refs: #3416)
- Use for LuaLaTeX same default settings as for XeLaTeX (i.e. `fontspec` and `polyglossia`). (refs: #3070, #3466)
- Make `'extraclassoptions'` key of `latex_elements` public (refs #3480)
- #3463: Add warning messages for required EPUB3 metadata. Add default value to `epub_description` to avoid warning like other settings.
- #3476: setuptools: Support multiple builders

- latex: merged cells in LaTeX tables allow code-blocks, lists, blockquotes… as do normal cells (refs: #3435)
- HTML builder uses experimental HTML5 writer if `html_experimental_html5_writer` is True and docutils 0.13 or later is installed.
- LaTeX macros to customize space before and after tables in PDF output (refs #3504)
- #3348: Show decorators in literalinclude and viewcode directives
- #3108: Show warning if :start-at: and other literalinclude options does not match to the text
- #3609: Allow to suppress "duplicate citation" warnings using `suppress_warnings`
- #2803: Discovery of builders by entry point
- #1764, #1676: Allow setting 'rel' and 'title' attributes for stylesheets
- #3589: Support remote images on non-HTML builders
- #3589: Support images in Data URI on non-HTML builders
- #2961: improve *`autodoc_mock_imports`*. Now the config value only requires to declare the top-level modules that should be mocked. Thanks to Robin Jarry.
- #3449: On py3, autodoc use inspect.signature for more accurate signature calculation. Thanks to Nathaniel J. Smith.
- #3641: Epub theme supports HTML structures that are generated by HTML5 writer.
- #3644 autodoc uses inspect instead of checking types. Thanks to Jeroen Demeyer.
- Add a new extension; `sphinx.ext.imgconverter`. It converts images in the document to appropriate format for builders
- latex: Use templates to render tables (refs #3389, 2a37b0e)

1.6b2

- `LATEXMKOPTS` variable for the Makefile in `$BUILDDIR/latex` to pass options to `latexmk` when executing `make latexpdf` (refs #3695, #3720)
- Add a new event *`env-check-consistency`* to check consistency to extensions
- Add *`Domain.check_consistency()`* to check consistency

Bugs fixed

1.6b1

- `literalinclude` directive expands tabs after dedent-ing (refs: #3416)
- #1574: Paragraphs in table cell doesn't work in Latex output
- #3288: Table with merged headers not wrapping text
- #3491: Inconsistent vertical space around table and longtable in PDF
- #3506: Depart functions for all admonitions in HTML writer now properly pass `node` to `depart_admonition`.
- #2693: Sphinx latex style file wrongly inhibits colours for section headings for latex+dvi(ps,pdf,pdfmx)
- C++, properly look up `any` references.
- #3624: sphinx.ext.intersphinx couldn't load inventories compressed with gzip
- #3551: PDF information dictionary is lacking author and title data
- #3351: intersphinx does not refers context like `py:module`, `py:class` and so on.
- Fail to load template file if the parent template is archived

1.6b2

- #3661: sphinx-build crashes on parallel build
- #3669: gettext builder fails with "ValueError: substring not found"
- #3660: Sphinx always depends on sphinxcontrib-websupport and its dependencies
- #3472: smart quotes getting wrong in latex (at least with list of strings via autoattribute) (refs: #3345, #3666)

1.6b3

- #3588: No compact (p tag) html output in the i18n document build even when *`html_compact_lists`* is True.
- The `make latexpdf` from 1.6b1 (for GNU/Linux and Mac OS, using `latexmk`) aborted earlier in case of LaTeX errors than was the case with 1.5 series, due to hard-coded usage of `--halt-on-error` option (refs #3695)
- #3683: sphinx.websupport module is not provided by default
- #3683: Failed to build document if builder.css_file.insert() is called
- #3714: viewcode extension not taking `highlight_code='none'` in account
- #3698: Moving :doc: to std domain broke backwards compatibility
- #3633: misdetect unreferenced citations

1.6 final

- LaTeX tables do not allow multiple paragraphs in a header cell
- LATEXOPTS is not passed over correctly to pdflatex since 1.6b3
- #3532: Figure or literal block captions in cells of short tables cause havoc in PDF output
- Fix: in PDF captions of tables are rendered differently whether table is of longtable class or not (refs #3686)
- #3725: Todo looks different from note in LaTeX output
- #3479: stub-columns have no effect in LaTeX output
- #3738: Nonsensical code in theming.py
- #3746: PDF builds fail with latexmk 4.48 or earlier due to undefined options `-pdfxe` and `-pdflua`

Deprecated

1.6b1

- `sphinx.util.compat.Directive` class is now deprecated. Please use instead `docutils.parsers.rst.Directive`
- `sphinx.util.compat.docutils_version` is now deprecated
- #2367: `Sphinx.warn()`, `Sphinx.info()` and other logging methods are now deprecated. Please use `sphinx.util.logging` (*Logging API*) instead.
- #3318: `notice` is now deprecated as LaTeX environment name and will be removed at Sphinx 1.7. Extension authors please use `sphinxadmonition` instead (as Sphinx does since 1.5.)
- `Sphinx.status_iterator()` and `Sphinx.old_status_iterator()` is now deprecated. Please use `sphinx.util:status_iterator()` instead.
- `Sphinx._directive_helper()` is deprecated. Please use `sphinx.util.docutils.directive_helper()` instead.
- `BuildEnvironment.set_warnfunc()` is now deprecated
- Following methods of `BuildEnvironment` is now deprecated.
  - `BuildEnvironment.note_toctree()`

  - `BuildEnvironment.get_toc_for()`
  - `BuildEnvironment.get_toctree_for()`
  - `BuildEnvironment.create_index()`

  Please use `sphinx.environment.adapters` modules instead.

- latex package `footnote` is not loaded anymore by its bundled replacement `footnotehyper-sphinx`. The redefined macros keep the same names as in the original package.
- #3429: deprecate config setting `latex_keep_old_macro_names`. It will be removed at 1.7, and already its default value has changed from `True` to `False`.
- #3221: epub2 builder is deprecated
- #3254: `sphinx.websupport` is now separated into independent package; `sphinxcontrib-websupport`. `sphinx.websupport` will be removed in Sphinx-2.0.
- #3628: `sphinx_themes` entry_point is deprecated. Please use `sphinx.html_themes` instead.

1.6b2

- #3662: `builder.css_files` is deprecated. Please use `add_stylesheet()` API instead.

1.6 final

- LaTeX `\sphinxstylethead` is deprecated at 1.6 and will be removed at 1.7. Please move customization into new macro `\sphinxstyletheadfamily`.

Testing

1.6 final

- #3458: Add `sphinx.testing` (experimental)

10.59 Release 1.6 (unreleased)

- not released (because of package script error)

10.60 Release 1.5.6 (released May 15, 2017)

Bugs fixed

- #3614: Sphinx crashes with requests-2.5.0
- #3618: autodoc crashes with tupled arguments
- #3664: No space after the bullet in items of a latex list produced by Sphinx
- #3657: EPUB builder crashes if a document starting with genindex exists
- #3588: No compact (p tag) html output in the i18n document build even when *`html_compact_lists`* is True.
- #3685: AttributeError when using 3rd party domains
- #3702: LaTeX writer styles figure legends with a hard-coded `\small`
- #3708: LaTeX writer allows irc scheme
- #3717: Stop enforcing that favicon's must be .ico
- #3731, #3732: Protect isenumclass predicate against non-class arguments

- #3320: Warning about reference target not being found for container types
- Misspelled ARCHIVEPREFIX in Makefile for latex build repertory

10.61 Release 1.5.5 (released Apr 03, 2017)

Bugs fixed

- #3597: python domain raises UnboundLocalError if invalid name given
- #3599: Move to new MathJax CDN

10.62 Release 1.5.4 (released Apr 02, 2017)

Features added

- #3470: Make genindex support all kinds of letters, not only Latin ones

Bugs fixed

- #3445: setting `'inputenc'` key to `\\usepackage[utf8x]{inputenc}` leads to failed PDF build
- EPUB file has duplicated `nav.xhtml` link in `content.opf` except first time build
- #3488: objects.inv has broken when `release` or `version` contain return code
- #2073, #3443, #3490: gettext builder that writes pot files unless the content are same without creation date. Thanks to Yoshiki Shibukawa.
- #3487: intersphinx: failed to refer options
- #3496: latex longtable's last column may be much wider than its contents
- #3507: wrong quotes in latex output for productionlist directive
- #3533: Moving from Sphinx 1.3.1 to 1.5.3 breaks LaTeX compilation of links rendered as code
- #2665, #2607: Link names in C++ docfields, and make it possible for other domains.
- #3542: C++, fix parsing error of non-type template argument with template.
- #3065, #3520: python domain fails to recognize nested class
- #3575: Problems with pdflatex in a Turkish document built with sphinx has reappeared (refs #2997, #2397)
- #3577: Fix intersphinx debug tool
- A LaTeX command such as `\\large` inserted in the title items of *`latex_documents`* causes failed PDF build (refs #3551, #3567)

10.63 Release 1.5.3 (released Feb 26, 2017)

Features added

- Support requests-2.0.0 (experimental) (refs: #3367)
- (latex) PDF page margin dimensions may be customized (refs: #3387)
- `literalinclude` directive allows combination of `:pyobject:` and `:lines:` options (refs: #3416)
- #3400: make-mode doesn't use subprocess on building docs

Bugs fixed

- #3370: the caption of code-block is not picked up for translation
- LaTeX: *`release`* is not escaped (refs: #3362)
- #3364: sphinx-quickstart prompts overflow on Console with 80 chars width
- since 1.5, PDF's TOC and bookmarks lack an entry for general Index (refs: #3383)
- #3392: `'releasename'` in *`latex_elements`* is not working
- #3356: Page layout for Japanese `'manual'` docclass has a shorter text area
- #3394: When `'pointsize'` is not `10pt`, Japanese `'manual'` document gets wrong PDF page dimensions
- #3399: quickstart: conf.py was not overwritten by template
- #3366: option directive does not allow punctuations
- #3410: return code in *`release`* breaks html search
- #3427: autodoc: memory addresses are not stripped on Windows
- #3428: xetex build tests fail due to fontspec v2.6 defining `\strong`
- #3349: Result of `IndexBuilder.load()` is broken
- #3450: &nbsp is appeared in EPUB docs
- #3418: Search button is misaligned in nature and pyramid theme
- #3421: Could not translate a caption of tables
- #3552: linkcheck raises UnboundLocalError

10.64 Release 1.5.2 (released Jan 22, 2017)

Incompatible changes

- Dependency requirement updates: requests 2.4.0 or above (refs: #3268, #3310)

Features added

- #3241: emit latex warning if buggy titlesec (ref #3210)
- #3194: Refer the $MAKE environment variable to determine `make` command
- Emit warning for nested numbered toctrees (refs: #3142)
- #978: *`intersphinx_mapping`* also allows a list as a parameter
- #3340: (LaTeX) long lines in parsed-literal[611] are wrapped like in *`code-block`*, inline math and footnotes are fully functional.

Bugs fixed

- #3246: xapian search adapter crashes
- #3253: In Py2 environment, building another locale with a non-captioned toctree produces `None` captions
- #185: References to section title including raw node has broken
- #3255: In Py3.4 environment, autodoc doesn't support documentation for attributes of Enum class correctly.
- #3261: `latex_use_parts` makes sphinx crash
- The warning type `misc.highlighting_failure` does not work
- #3294: `add_latex_package()` make crashes non-LaTeX builders
- The caption of table are rendered as invalid HTML (refs: #3287)
- #3268: Sphinx crashes with requests package from Debian jessie
- #3284: Sphinx crashes on parallel build with an extension which raises unserializable exception
- #3315: Bibliography crashes on latex build with docclass 'memoir'
- #3328: Could not refer rubric implicitly
- #3329: emit warnings if po file is invalid and can't read it. Also writing mo
- #3337: Ugly rendering of definition list term's classifier
- #3335: gettext does not extract field_name of a field in a field_list
- #2952: C++, fix refs to operator() functions.
- Fix Unicode super- and subscript digits in *`code-block`* and parsed-literal LaTeX output (ref #3342)
- LaTeX writer: leave `"` character inside parsed-literal as is (ref #3341)
- #3234: intersphinx failed for encoded inventories
- #3158: too much space after captions in PDF output
- #3317: An URL in parsed-literal contents gets wrongly rendered in PDF if with hyphen
- LaTeX crash if the filename of an image inserted in parsed-literal via a substitution contains an hyphen (ref #3340)
- LaTeX rendering of inserted footnotes in parsed-literal is wrong (ref #3340)
- Inline math in parsed-literal is not rendered well by LaTeX (ref #3340)
- #3308: Parsed-literals don't wrap very long lines with pdf builder (ref #3340)
- #3295: Could not import extension sphinx.builders.linkcheck
- #3285: autosummary: asterisks are escaped twice

[611] http://docutils.sourceforge.net/docs/ref/rst/directives.html#parsed-literal

- LaTeX, pass dvipdfm option to geometry package for Japanese documents (ref #3363)
- Fix parselinenos() could not parse left half open range (cf. "-4")

10.65 Release 1.5.1 (released Dec 13, 2016)

Features added

- #3214: Allow to suppress "unknown mimetype" warnings from epub builder using *suppress_warnings*.

Bugs fixed

- #3195: Can not build in parallel
- #3198: AttributeError is raised when toctree has 'self'
- #3211: Remove untranslated sphinx locale catalogs (it was covered by untranslated it_IT)
- #3212: HTML Builders crashes with docutils-0.13
- #3207: more latex problems with references inside parsed-literal directive (`\DUrole`)
- #3205: sphinx.util.requests crashes with old pyOpenSSL (< 0.14)
- #3220: KeyError when having a duplicate citation
- #3200: LaTeX: xref inside desc_name not allowed
- #3228: `build_sphinx` command crashes when missing dependency
- #2469: Ignore updates of catalog files for gettext builder. Thanks to Hiroshi Ohkubo.
- #3183: Randomized jump box order in generated index page.

10.66 Release 1.5 (released Dec 5, 2016)

Incompatible changes

1.5a1

- latex, package fancybox is not any longer a dependency of sphinx.sty
- Use `'locales'` as a default value of *locale_dirs*
- latex, package ifthen is not any longer a dependency of sphinx.sty
- latex, style file does not modify fancyvrb's Verbatim (also available as OriginalVerbatim) but uses sphinxVerbatim for name of custom wrapper.
- latex, package newfloat is not used (and not included) anymore (ref #2660; it was used since 1.3.4 and shipped with Sphinx since 1.4).
- latex, literal blocks in tables do not use OriginalVerbatim but sphinxVerbatimintable which handles captions and wraps lines (ref #2704).
- latex, replace `pt` by TeX equivalent `bp` if found in `width` or `height` attribute of an image.
- latex, if `width` or `height` attribute of an image is given with no unit, use `px` rather than ignore it.
- latex: Separate stylesheets of pygments to independent .sty file

- #2454: The filename of sourcelink is now changed. The value of *html_sourcelink_suffix* will be appended to the original filename (like index.rst.txt).
- sphinx.util.copy_static_entry() is now deprecated. Use sphinx.util.fileutil.copy_asset() instead.
- sphinx.util.osutil.filecopy() skips copying if the file has not been changed (ref: #2510, #2753)
- Internet Explorer 6-8, Opera 12.1x or Safari 5.1+ support is dropped because jQuery version is updated from 1.11.0 to 3.1.0 (ref: #2634, #2773)
- QtHelpBuilder doesn't generate search page (ref: #2352)
- QtHelpBuilder uses nonav theme instead of default one to improve readability.
- latex: To provide good default settings to Japanese documents, Sphinx uses jreport and jsbook as docclass if *language* is ja.
- sphinx-quickstart now allows a project version is empty
- Fix :download: role on epub/qthelp builder. They ignore the role because they don't support it.
- sphinx.ext.viewcode doesn't work on epub building by default. viewcode_enable_epub option
- sphinx.ext.viewcode disabled on singlehtml builder.
- Use make-mode of sphinx-quickstart by default. To disable this, use -M option
- Fix genindex.html, Sphinx's document template, link address to itself to satisfy xhtml standard.
- Use epub3 builder by default. And the old epub builder is renamed to epub2.
- Fix epub and epub3 builders that contained links to genindex even if epub_use_index = False.
- html_translator_class is now deprecated. Use *Sphinx.set_translator()* API instead.
- Drop python 2.6 and 3.3 support
- Drop epub3 builder's epub3_page_progression_direction option (use epub3_writing_mode).
- #2877: Rename latex_elements['footer'] to latex_elements['atendofbody']

1.5a2

- #2983: Rename epub3_description and epub3_contributor to epub_description and epub_contributor.
- Remove themes/basic/defindex.html; no longer used
- Sphinx does not ship anymore (but still uses) LaTeX style file fncychap
- #2435: Slim down quickstarted conf.py
- The sphinx.sty latex package does not load itself "hyperref", as this is done later in the preamble of the latex output via 'hyperref' key.
- Sphinx does not ship anymore a custom modified LaTeX style file tabulary. The non-modified package is used.
- #3057: By default, footnote marks in latex PDF output are not preceded by a space anymore, \sphinxBeforeFootnote allows user customization if needed.
- LaTeX target requires that option hyperfootnotes of package hyperref be left unchanged to its default (i.e. true) (refs: #3022)

1.5 final

- #2986: themes/basic/defindex.html is now deprecated
- Emit warnings that will be deprecated in Sphinx 1.6 by default. Users can change the behavior by setting the environment variable PYTHONWARNINGS. Please refer *Deprecation Warnings*.
- #2454: new JavaScript variable SOURCELINK_SUFFIX is added

Deprecated

These features are removed in Sphinx-1.6:

- LDML format support in i18n feature
- `sphinx.addnodes.termsep`
- Some functions and classes in `sphinx.util.pycompat`: `zip_longest`, `product`, `all`, `any`, `next`, `open`, `class_types`, `base_exception`, `relpath`, `StringIO`, `BytesIO`. Please use the standard library version instead;

If any deprecation warning like `RemovedInSphinxXXXWarning` are displayed, please refer *Deprecation Warnings*.

Features added

1.5a1

- #2951: Add `--implicit-namespaces` PEP-0420 support to apidoc.
- Add `:caption:` option for sphinx.ext.inheritance_diagram.
- #2471: Add config variable for default doctest flags.
- Convert linkcheck builder to requests for better encoding handling
- #2463, #2516: Add keywords of "meta" directive to search index
- `:maxdepth:` option of toctree affects `secnumdepth` (ref: #2547)
- #2575: Now `sphinx.ext.graphviz` allows `:align:` option
- Show warnings if unknown key is specified to *`latex_elements`*
- Show warnings if no domains match with *`primary_domain`* (ref: #2001)
- C++, show warnings when the kind of role is misleading for the kind of target it refers to (e.g., using the *`class`* role for a function).
- latex, writer abstracts more of text styling into customizable macros, e.g. the `visit_emphasis` will output `\sphinxstyleemphasis` rather than `\emph` (which may be in use elsewhere or in an added LaTeX package). See list at end of `sphinx.sty` (ref: #2686)
- latex, public names for environments and parameters used by note, warning, and other admonition types, allowing full customizability from the `'preamble'` key or an input file (ref: feature request #2674, #2685)
- latex, better computes column widths of some tables (as a result, there will be slight changes as tables now correctly fill the line width; ref: #2708)
- latex, sphinxVerbatim environment is more easily customizable (ref: #2704). In addition to already existing VerbatimColor and VerbatimBorderColor:
  - two lengths `\sphinxverbatimsep` and `\sphinxverbatimborder`,
  - booleans `\ifsphinxverbatimwithframe` and `\ifsphinxverbatimwrapslines`.
- latex, captions for literal blocks inside tables are handled, and long code lines wrapped to fit table cell (ref: #2704)
- #2597: Show warning messages as darkred
- latex, allow image dimensions using px unit (default is 96px=1in)
- Show warnings if invalid dimension units found
- #2650: Add `--pdb` option to setup.py command
- latex, make the use of `\small` for code listings customizable (ref #2721)

- #2663: Add `--warning-is-error` option to setup.py command
- Show warnings if deprecated latex options are used
- Add sphinx.config.ENUM to check the config values is in candidates
- math: Add hyperlink marker to each equations in HTML output
- Add new theme `nonav` that doesn't include any navigation links. This is for any help generator like qthelp.
- #2680: *`sphinx.ext.todo`* now emits warnings if *`todo_emit_warnings`* enabled. Also, it emits an additional event named *`todo-defined`* to handle the TODO entries in 3rd party extensions.
- Python domain signature parser now uses the xref mixin for 'exceptions', allowing exception classes to be autolinked.
- #2513: Add *`latex_engine`* to switch the LaTeX engine by conf.py
- #2682: C++, basic support for attributes (C++11 style and GNU style). The new configuration variables 'cpp_id_attributes' and 'cpp_paren_attributes' can be used to introduce custom attributes.
- #1958: C++, add configuration variable 'cpp_index_common_prefix' for removing prefixes from the index text of C++ objects.
- C++, added concept directive. Thanks to mickk-on-cpp.
- C++, added support for template introduction syntax. Thanks to mickk-on-cpp.
- #2725: latex builder: allow to use user-defined template file (experimental)
- apidoc now avoids invalidating cached files by not writing to files whose content doesn't change. This can lead to significant performance wins if apidoc is run frequently.
- #2851: `sphinx.ext.math` emits missing-reference event if equation not found
- #1210: `eqref` role now supports cross reference
- #2892: Added `-a` (`--append-syspath`) option to `sphinx-apidoc`
- #1604: epub3 builder: Obey font-related CSS when viewing in iBooks.
- #646: `option` directive support '.' character as a part of options
- Add document about kindlegen and fix document structure for it.
- #2474: Add `intersphinx_timeout` option to `sphinx.ext.intersphinx`
- #2926: EPUB3 builder supports vertical mode (`epub3_writing_mode` option)
- #2695: `build_sphinx` subcommand for setuptools handles exceptions as same as `sphinx-build` does
- #326: *`numref`* role can also refer sections
- #2916: *`numref`* role can also refer caption as an its linktext

1.5a2

- #3008: `linkcheck` builder ignores self-signed certificate URL
- #3020: new `'geometry'` key to `latex_elements` whose default uses LaTeX style file `geometry.sty` to set page layout
- #2843: Add :start-at: and :end-at: options to literalinclude directive
- #2527: Add `:reversed:` option to toctree directive
- Add `-t` and `-d` option to `sphinx-quickstart` to support templating generated sphinx project.
- #3028: Add `{path}` and `{basename}` to the format of `figure_language_filename`
- new `'hyperref'` key in the `latex_elements` dictionary (ref #3030)
- #3022: Allow code-blocks in footnotes for LaTeX PDF output

1.5b1

- #2513: A better default settings for XeLaTeX
- #3096: `'maxlistdepth'` key to work around LaTeX list limitations
- #3060: autodoc supports documentation for attributes of Enum class. Now autodoc render just the value of Enum attributes instead of Enum attribute representation.
- Add `--extensions` to `sphinx-quickstart` to support enable arbitrary extensions from command line (ref: #2904)
- #3104, #3122: `'sphinxsetup'` for key=value styling of Sphinx LaTeX
- #3071: Autodoc: Allow mocked module decorators to pass-through functions unchanged
- #2495: linkcheck: Allow skipping anchor checking using *`linkcheck_anchors_ignore`*
- #3083: let Unicode no-break space act like LaTeX ~ (fixed #3019)
- #3116: allow word wrap in PDF output for inline literals (ref #3110)
- #930: sphinx-apidoc allow wildcards for excluding paths. Thanks to Nick Coghlan.
- #3121: add `inlineliteralwraps` option to control if inline literal word-wraps in latex

1.5 final

- #3095: Add *`tls_verify`* and *`tls_cacerts`* to support self-signed HTTPS servers in linkcheck and intersphinx
- #2215: make.bat generated by sphinx-quickstart can be called from another dir. Thanks to Timotheus Kampik.
- #3185: Add new warning type `misc.highlighting_failure`

Bugs fixed

1.5a1

- #2707: (latex) the column width is badly computed for tabular
- #2799: Sphinx installs roles and directives automatically on importing sphinx module. Now Sphinx installs them on running application.
- *`sphinx.ext.autodoc`* crashes if target code imports * from mock modules by *`autodoc_mock_imports`*.
- #1953: `Sphinx.add_node` does not add handlers the translator installed by `html_translator_class`
- #1797: text builder inserts blank line on top
- #2894: quickstart main() doesn't use argv argument
- #2874: gettext builder could not extract all text under the `only` directives
- #2485: autosummary crashes with multiple source_suffix values
- #1734: Could not translate the caption of toctree directive
- Could not translate the content of meta directive (ref: #1734)
- #2550: external links are opened in help viewer
- #2687: Running Sphinx multiple times produces 'already registered' warnings

1.5a2

- #2810: Problems with pdflatex in an Italian document
- Use `latex_elements.papersize` to specify papersize of LaTeX in Makefile
- #2988: linkcheck: retry with GET request if denied HEAD request
- #2990: linkcheck raises "Can't convert 'bytes' object to str implicitly" error if linkcheck_anchors enabled

- #3004: Invalid link types "top" and "up" are used
- #3009: Bad rendering of parsed-literals in LaTeX since Sphinx 1.4.4
- #3000: `option` directive generates invalid HTML anchors
- #2984: Invalid HTML has been generated if *`html_split_index`* enabled
- #2986: themes/basic/defindex.html should be changed for html5 friendly
- #2987: Invalid HTML has been generated if multiple IDs are assigned to a list
- #2891: HTML search does not provide all the results
- #1986: Title in PDF Output
- #147: Problem with latex chapter style
- #3018: LaTeX problem with page layout dimensions and chapter titles
- Fix an issue with `\pysigline` in LaTeX style file (ref #3023)
- #3038: `sphinx.ext.math*` raises TypeError if labels are duplicated
- #3031: incompatibility with LaTeX package `tocloft`
- #3003: literal blocks in footnotes are not supported by Latex
- #3047: spacing before footnote in pdf output is not coherent and allows breaks
- #3045: HTML search index creator should ignore "raw" content if now html
- #3039: English stemmer returns wrong word if the word is capitalized
- Fix make-mode Makefile template (ref #3056, #2936)

1.5b1

- #2432: Fix unwanted * between varargs and keyword only args. Thanks to Alex Grönholm.
- #3062: Failed to build PDF using 1.5a2 (undefined `\hypersetup` for Japanese documents since PR#3030)
- Better rendering of multiline signatures in html.
- #777: LaTeX output "too deeply nested" (ref #3096)
- Let LaTeX image inclusion obey `scale` before textwidth fit (ref #2865, #3059)
- #3019: LaTeX fails on description of C function with arguments (ref #3083)
- fix latex inline literals where `< > -` gobbled a space

1.5 final

- #3069: Even if `'babel'` key is set to empty string, LaTeX output contains one `\addto\captions...`
- #3123: user `'babel'` key setting is not obeyed anymore
- #3155: Fix JavaScript for *`html_sourcelink_suffix`* fails with IE and Opera
- #3085: keep current directory after breaking build documentation. Thanks to Timotheus Kampik.
- #3181: pLaTeX crashes with a section contains endash
- #3180: latex: add stretch/shrink between successive singleline or multipleline cpp signatures (ref #3072)
- #3128: globing images does not support .svgz file
- #3015: fix a broken test on Windows.
- #1843: Fix documentation of descriptor classes that have a custom metaclass. Thanks to Erik Bray.
- #3190: util.split_docinfo fails to parse multi-line field bodies

- #3024, #3037: In Python3, application.Sphinx._log crushed when the log message cannot be encoded into console encoding.

Testing

- To simplify, sphinx uses external mock package even if unittest.mock exists.

10.67 Release 1.4.9 (released Nov 23, 2016)

Bugs fixed

- #2936: Fix doc/Makefile that can't build man because doc/man exists
- #3058: Using the same 'caption' attribute in multiple 'toctree' directives results in warning / error
- #3068: Allow the '=' character in the -D option of sphinx-build.py
- #3074: `add_source_parser()` crashes in debug mode
- #3135: `sphinx.ext.autodoc` crashes with plain Callable
- #3150: Fix query word splitter in JavaScript. It behaves as same as Python's regular expression.
- #3093: gettext build broken on substituted images.
- #3093: gettext build broken on image node under `note` directive.
- imgmath: crashes on showing error messages if image generation failed
- #3117: LaTeX writer crashes if admonition is placed before first section title
- #3164: Change search order of `sphinx.ext.inheritance_diagram`

10.68 Release 1.4.8 (released Oct 1, 2016)

Bugs fixed

- #2996: The wheel package of Sphinx got crash with ImportError

10.69 Release 1.4.7 (released Oct 1, 2016)

Bugs fixed

- #2890: Quickstart should return an error consistently on all error conditions
- #2870: flatten genindex columns' heights.
- #2856: Search on generated HTML site doesn't find some symbols
- #2882: Fall back to a GET request on 403 status in linkcheck
- #2902: jsdump.loads fails to load search index if keywords starts with underscore
- #2900: Fix epub content.opf: add auto generated orphan files to spine.
- #2899: Fix `hasdoc()` function in Jinja2 template. It will detect `genindex`, `search` also.
- #2901: Fix epub result: skip creating links from image tags to original image files.

- #2917: inline code is hyphenated on HTML
- #1462: autosummary warns for namedtuple with attribute with trailing underscore
- Could not reference equations if `:nowrap:` option specified
- #2873: code-block overflow in latex (due to commas)
- #1060, #2056: sphinx.ext.intersphinx: broken links are generated if relative paths are used in *intersphinx_mapping*
- #2931: code-block directive with same :caption: causes warning of duplicate target. Now *code-block* and *literalinclude* does not define hyperlink target using its caption automatically.
- #2962: latex: missing label of longtable
- #2968: autodoc: show-inheritance option breaks docstrings

10.70 Release 1.4.6 (released Aug 20, 2016)

Incompatible changes

- #2867: linkcheck builder crashes with six-1.4. Now Sphinx depends on six-1.5 or later

Bugs fixed

- applehelp: Sphinx crashes if `hiutil` or `codesign` commands not found
- Fix `make clean` abort issue when build dir contains regular files like `DS_Store`.
- Reduce epubcheck warnings/errors:
  - Fix DOCTYPE to html5
  - Change extension from .html to .xhtml.
  - Disable search page on epub results
- #2778: Fix autodoc crashes if obj.__dict__ is a property method and raises exception
- Fix duplicated toc in epub3 output.
- #2775: Fix failing linkcheck with servers not supporting identity encoding
- #2833: Fix formatting instance annotations in ext.autodoc.
- #1911: `-D` option of `sphinx-build` does not override the `extensions` variable
- #2789: *sphinx.ext.intersphinx* generates wrong hyperlinks if the inventory is given
- parsing errors for caption of code-blocks are displayed in document (ref: #2845)
- #2846: `singlehtml` builder does not include figure numbers
- #2816: Fix data from builds cluttering the `Domain.initial_data` class attributes

10.71 Release 1.4.5 (released Jul 13, 2016)

Incompatible changes

- latex, inclusion of non-inline images from image directive resulted in non-coherent whitespaces depending on original image width; new behaviour by necessity differs from earlier one in some cases. (ref: #2672)
- latex, use of `\includegraphics` to refer to Sphinx custom variant is deprecated; in future it will revert to original LaTeX macro, custom one already has alternative name `\sphinxincludegraphics`.

Features added

- new config option `latex_keep_old_macro_names`, defaults to True. If False, lets macros (for text styling) be defined only with `\sphinx`-prefixed names
- latex writer allows user customization of "shadowed" boxes (topics), via three length variables.
- woff-format web font files now supported by the epub builder.

Bugs fixed

- jsdump fix for python 3: fixes the HTML search on python > 3
- #2676: (latex) Error with verbatim text in captions since Sphinx 1.4.4
- #2629: memoir class crashes LaTeX. Fixed by `latex_keep_old_macro_names=False` (ref 2675)
- #2684: *sphinx.ext.intersphinx* crashes with six-1.4.1
- #2679: `float` package needed for `'figure_align': 'H'` latex option
- #2671: image directive may lead to inconsistent spacing in pdf
- #2705: `toctree` generates empty bullet_list if `:titlesonly:` specified
- #2479: *sphinx.ext.viewcode* uses python2 highlighter by default
- #2700: HtmlHelp builder has hard coded index.html
- latex, since 1.4.4 inline literal text is followed by spurious space
- #2722: C++, fix id generation for var/member declarations to include namespaces.
- latex, images (from image directive) in lists or quoted blocks did not obey indentation (fixed together with #2671)
- #2733: since Sphinx-1.4.4 `make latexpdf` generates lots of hyperref warnings
- #2731: *sphinx.ext.autodoc* does not access propertymethods which raises any exceptions
- #2666: C++, properly look up nested names involving constructors.
- #2579: Could not refer a label including both spaces and colons via *sphinx.ext.intersphinx*
- #2718: Sphinx crashes if the document file is not readable
- #2699: hyperlinks in help HTMLs are broken if *html_file_suffix* is set
- #2723: extra spaces in latex pdf output from multirow cell
- #2735: latexpdf `Underfull \hbox (badness 10000)` warnings from title page
- #2667: latex crashes if resized images appeared in section title
- #2763: (html) Provide default value for required `alt` attribute for image tags of SVG source, required to validate and now consistent w/ other formats.

10.72 Release 1.4.4 (released Jun 12, 2016)

Bugs fixed

- #2630: latex: sphinx.sty notice environment formatting problem
- #2632: Warning directives fail in quote environment latex build
- #2633: Sphinx crashes with old styled indices
- Fix a `\begin{\minipage}` typo in sphinx.sty from 1.4.2 (ref: 68becb1)
- #2622: Latex produces empty pages after title and table of contents
- #2640: 1.4.2 LaTeX crashes if code-block inside warning directive
- Let LaTeX use straight quotes also in inline code (ref #2627)
- #2351: latex crashes if enumerated lists are placed on footnotes
- #2646: latex crashes if math contains twice empty lines
- #2480: *`sphinx.ext.autodoc`*: memory addresses were shown
- latex: allow code-blocks appearing inside lists and quotes at maximal nesting depth (ref #777, #2624, #2651)
- #2635: Latex code directives produce inconsistent frames based on viewing resolution
- #2639: Sphinx now bundles iftex.sty
- Failed to build PDF with framed.sty 0.95
- Sphinx now bundles needspace.sty

10.73 Release 1.4.3 (released Jun 5, 2016)

Bugs fixed

- #2530: got "Counter too large" error on building PDF if large numbered footnotes existed in admonitions
- `width` option of figure directive does not work if `align` option specified at same time (ref: #2595)
- #2590: The `inputenc` package breaks compiling under lualatex and xelatex
- #2540: date on latex front page use different font
- Suppress "document isn't included in any toctree" warning if the document is included (ref: #2603)
- #2614: Some tables in PDF output will end up shifted if user sets non zero parindent in preamble
- #2602: URL redirection breaks the hyperlinks generated by *`sphinx.ext.intersphinx`*
- #2613: Show warnings if merged extensions are loaded
- #2619: make sure amstext LaTeX package always loaded (ref: d657225, 488ee52, 9d82cad and #2615)
- #2593: latex crashes if any figures in the table

10.74 Release 1.4.2 (released May 29, 2016)

Features added

- Now `suppress_warnings` accepts following configurations (ref: #2451, #2466):
  - `app.add_node`
  - `app.add_directive`
  - `app.add_role`
  - `app.add_generic_role`
  - `app.add_source_parser`
  - `image.data_uri`
  - `image.nonlocal_uri`
- #2453: LaTeX writer allows page breaks in topic contents; and their horizontal extent now fits in the line width (with shadow in margin). Also warning-type admonitions allow page breaks and their vertical spacing has been made more coherent with the one for hint-type notices (ref #2446).
- #2459: the framing of literal code-blocks in LaTeX output (and not only the code lines themselves) obey the indentation in lists or quoted blocks.
- #2343: the long source lines in code-blocks are wrapped (without modifying the line numbering) in LaTeX output (ref #1534, #2304).

Bugs fixed

- #2370: the equations are slightly misaligned in LaTeX writer
- #1817, #2077: suppress pep8 warnings on conf.py generated by sphinx-quickstart
- #2407: building docs crash if document includes large data image URIs
- #2436: Sphinx does not check version by `needs_sphinx` if loading extensions failed
- #2397: Setup shorthandoff for Turkish documents
- #2447: VerbatimBorderColor wrongly used also for captions of PDF
- #2456: C++, fix crash related to document merging (e.g., singlehtml and Latex builders).
- #2446: latex(pdf) sets local tables of contents (or more generally topic nodes) in unbreakable boxes, causes overflow at bottom
- #2476: Omit MathJax markers if :nowrap: is given
- #2465: latex builder fails in case no caption option is provided to toctree directive
- Sphinx crashes if self referenced toctree found
- #2481: spelling mistake for mecab search splitter. Thanks to Naoki Sato.
- #2309: Fix could not refer "indirect hyperlink targets" by ref-role
- intersphinx fails if mapping URL contains any port
- #2088: intersphinx crashes if the mapping URL requires basic auth
- #2304: auto line breaks in latexpdf codeblocks
- #1534: Word wrap long lines in Latex verbatim blocks
- #2460: too much white space on top of captioned literal blocks in PDF output

- Show error reason when multiple math extensions are loaded (ref: #2499)
- #2483: any figure number was not assigned if figure title contains only non text objects
- #2501: Unicode subscript numbers are normalized in LaTeX
- #2492: Figure directive with :figwidth: generates incorrect Latex-code
- The caption of figure is always put on center even if `:align:` was specified
- #2526: LaTeX writer crashes if the section having only images
- #2522: Sphinx touches mo files under installed directory that caused permission error.
- #2536: C++, fix crash when an immediately nested scope has the same name as the current scope.
- #2555: Fix crash on any-references with unicode.
- #2517: wrong bookmark encoding in PDF if using LuaLaTeX
- #2521: generated Makefile causes BSD make crashed if sphinx-build not found
- #2470: `typing` backport package causes autodoc errors with python 2.7
- `sphinx.ext.intersphinx` crashes if non-string value is used for key of *`intersphinx_mapping`*
- #2518: *`intersphinx_mapping`* disallows non alphanumeric keys
- #2558: unpack error on devhelp builder
- #2561: Info builder crashes when a footnote contains a link
- #2565: The descriptions of objects generated by `sphinx.ext.autosummary` overflow lines at LaTeX writer
- Extend pdflatex config in sphinx.sty to subparagraphs (ref: #2551)
- #2445: *`rst_prolog`* and *`rst_epilog`* affect to non reST sources
- #2576: `sphinx.ext.imgmath` crashes if subprocess raises error
- #2577: `sphinx.ext.imgmath`: Invalid argument are passed to dvisvgm
- #2556: Xapian search does not work with Python 3
- #2581: The search doesn't work if language="es" (Spanish)
- #2382: Adjust spacing after abbreviations on figure numbers in LaTeX writer
- #2383: The generated footnote by *`latex_show_urls`* overflows lines
- #2497, #2552: The label of search button does not fit for the button itself

10.75 Release 1.4.1 (released Apr 12, 2016)

Incompatible changes

- The default format of *`today_fmt`* and *`html_last_updated_fmt`* is back to strftime format again. Locale Date Markup Language is also supported for backward compatibility until Sphinx-1.5.

Translations

- Added Welsh translation, thanks to Geraint Palmer.
- Added Greek translation, thanks to Stelios Vitalis.
- Added Esperanto translation, thanks to Dinu Gherman.
- Added Hindi translation, thanks to Purnank H. Ghumalia.
- Added Romanian translation, thanks to Razvan Stefanescu.

Bugs fixed

- C++, added support for `extern` and `thread_local`.
- C++, type declarations are now using the prefixes `typedef`, `using`, and `type`, depending on the style of declaration.
- #2413: C++, fix crash on duplicate declarations
- #2394: Sphinx crashes when html_last_updated_fmt is invalid
- #2408: dummy builder not available in Makefile and make.bat
- #2412: hyperlink targets are broken in LaTeX builder
- figure directive crashes if non paragraph item is given as caption
- #2418: time formats no longer allowed in today_fmt
- #2395: Sphinx crashes if unicode character in image filename
- #2396: "too many values to unpack" in genindex-single
- #2405: numref link in PDF jumps to the wrong location
- #2414: missing number in PDF hyperlinks to code listings
- #2440: wrong import for gmtime. Thanks to Uwe L. Korn.

10.76 Release 1.4 (released Mar 28, 2016)

Incompatible changes

- Drop `PorterStemmer` package support. Use `PyStemmer` instead of `PorterStemmer` to accelerate stemming.
- sphinx_rtd_theme has become optional. Please install it manually. Refs #2087, #2086, #1845 and #2097. Thanks to Victor Zverovich.
- #2231: Use DUrole instead of DUspan for custom roles in LaTeX writer. It enables to take title of roles as an argument of custom macros.
- #2022: 'Thumbs.db' and '.DS_Store' are added to *`exclude_patterns`* default values in conf.py that will be provided on sphinx-quickstart.
- #2027, #2208: The `html_title` accepts string values only. And The None value cannot be accepted.
- `sphinx.ext.graphviz`: show graph image in inline by default
- #2060, #2224: The `manpage` role now generate `sphinx.addnodes.manpage` node instead of `sphinx.addnodes.literal_emphasis` node.
- #2022: *`html_extra_path`* also copies dotfiles in the extra directory, and refers to *`exclude_patterns`* to exclude extra files and directories.

- #2300: enhance autoclass:: to use the docstring of __new__ if __init__ method's is missing of empty
- #2251: Previously, under glossary directives, multiple terms for one definition are converted into single `term` node and the each terms in the term node are separated by `termsep` node. In new implementation, each terms are converted into individual `term` nodes and `termsep` node is removed. By this change, output layout of every builders are changed a bit.
- The default highlight language is now Python 3. This means that source code is highlighted as Python 3 (which is mostly a superset of Python 2), and no parsing is attempted to distinguish valid code. To get the old behavior back, add `highlight_language = "python"` to conf.py.
- Locale Date Markup Language[612] like `"MMMM dd, YYYY"` is default format for *`today_fmt`* and *`html_last_updated_fmt`*. However strftime format like `"%B %d, %Y"` is also supported for backward compatibility until Sphinx-1.5. Later format will be disabled from Sphinx-1.5.
- #2327: `latex_use_parts` is deprecated now. Use *`latex_toplevel_sectioning`* instead.
- #2337: Use `\url{URL}` macro instead of `\href{URL}{URL}` in LaTeX writer.
- #1498: manpage writer: don't make whole of item in definition list bold if it includes strong node.
- #582: Remove hint message from quick search box for html output.
- #2378: Sphinx now bundles newfloat.sty

Features added

- #2092: add todo directive support in napoleon package.
- #1962: when adding directives, roles or nodes from an extension, warn if such an element is already present (built-in or added by another extension).
- #1909: Add "doc" references to Intersphinx inventories.
- C++ type alias support (e.g., `.. type:: T = int`).
- C++ template support for classes, functions, type aliases, and variables (#1729, #1314).
- C++, added new scope management directives `namespace-push` and `namespace-pop`.
- #1970: Keyboard shortcuts to navigate Next and Previous topics
- Intersphinx: Added support for fetching Intersphinx inventories with URLs using HTTP basic auth.
- C++, added support for template parameter in function info field lists.
- C++, added support for pointers to member (function).
- #2113: Allow `:class:` option to code-block directive.
- #2192: Imgmath (pngmath with svg support).
- #2200: Support XeTeX and LuaTeX for the LaTeX builder.
- #1906: Use xcolor over color for fcolorbox where available for LaTeX output.
- #2216: Texinputs makefile improvements.
- #2170: Support for Chinese language search index.
- #2214: Add sphinx.ext.githubpages to publish the docs on GitHub Pages
- #1030: Make page reference names for latex_show_pagerefs translatable
- #2162: Add Sphinx.add_source_parser() to add source_suffix and source_parsers from extension
- #2207: Add sphinx.parsers.Parser class; a base class for new parsers

[612] https://unicode.org/reports/tr35/tr35-dates.html#Date_Format_Patterns

- #656: Add graphviz_dot option to graphviz directives to switch the dot command
- #1939: Added the dummy builder: syntax check without output.
- #2230: Add math_number_all option to number all displayed math in math extensions
- #2235: needs_sphinx supports micro version comparison
- #2282: Add "language" attribute to html tag in the "basic" theme
- #1779: Add EPUB 3 builder
- #1751: Add *todo_link_only* to avoid file path and line indication on *todolist*. Thanks to Francesco Montesano.
- #2199: Use imagesize package to obtain size of images.
- #1099: Add configurable retries to the linkcheck builder. Thanks to Alex Gaynor. Also don't check anchors starting with !.
- #2300: enhance autoclass:: to use the docstring of __new__ if __init__ method's is missing of empty
- #1858: Add Sphinx.add_enumerable_node() to add enumerable nodes for numfig feature
- #1286, #2099: Add sphinx.ext.autosectionlabel extension to allow reference sections using its title. Thanks to Tadhg O'Higgins.
- #1854: Allow to choose Janome for Japanese splitter.
- #1853: support custom text splitter on html search with language='ja'.
- #2320: classifier of glossary terms can be used for index entries grouping key The classifier also be used for translation. See also *Glossary*.
- #2308: Define \tablecontinued macro to redefine the style of continued label for longtables.
- Select an image by similarity if multiple images are globbed by .. image:: filename.*
- #1921: Support figure substitutions by *language* and *figure_language_filename*
- #2245: Add latex_elements["passoptionstopackages"] option to call PassOptionsToPackages in early stage of preambles.
- #2340: Math extension: support alignment of multiple equations for MathJax.
- #2338: Define \titleref macro to redefine the style of title-reference roles.
- Define \menuselection and \accelerator macros to redefine the style of *menuselection* roles.
- Define \crossref macro to redefine the style of references
- #2301: Texts in the classic html theme should be hyphenated.
- #2355: Define \termref macro to redefine the style of term roles.
- Add *suppress_warnings* to suppress arbitrary warning message (experimental)
- #2229: Fix no warning is given for unknown options
- #2327: Add *latex_toplevel_sectioning* to switch the top level sectioning of LaTeX document.

Bugs fixed

- #1913: C++, fix assert bug for enumerators in next-to-global and global scope.
- C++, fix parsing of 'signed char' and 'unsigned char' as types.
- C++, add missing support for 'friend' functions.
- C++, add missing support for virtual base classes (thanks to Rapptz).
- C++, add support for final classes.
- C++, fix parsing of types prefixed with 'enum'.
- #2023: Dutch search support uses Danish stemming info.
- C++, add support for user-defined literals.
- #1804: Now html output wraps overflowed long-line-text in the sidebar. Thanks to Hassen ben tanfous.
- #2183: Fix porterstemmer causes `make json` to fail.
- #1899: Ensure list is sent to OptParse.
- #2164: Fix wrong check for pdftex inside sphinx.sty (for graphicx package option).
- #2165, #2218: Remove faulty and non-need conditional from sphinx.sty.
- Fix broken LaTeX code is generated if unknown language is given
- #1944: Fix rst_prolog breaks file-wide metadata
- #2074: make gettext should use canonical relative paths for .pot. Thanks to anatoly techtonik.
- #2311: Fix sphinx.ext.inheritance_diagram raises AttributeError
- #2251: Line breaks in .rst files are transferred to .pot files in a wrong way.
- #794: Fix date formatting in latex output is not localized
- Remove `image/gif` from supported_image_types of LaTeX writer (#2272)
- Fix ValueError is raised if LANGUAGE is empty string
- Fix unpack warning is shown when the directives generated from `Sphinx.add_crossref_type` is used
- The default highlight language is now `default`. This means that source code is highlighted as Python 3 (which is mostly a superset of Python 2) if possible. To get the old behavior back, add `highlight_language = "python"` to conf.py.
- #2329: Refresh environment forcedly if source directory has changed.
- #2331: Fix code-blocks are filled by block in dvi; remove `xcdraw` option from xcolor package
- Fix the confval type checker emits warnings if unicode is given to confvals which expects string value
- #2360: Fix numref in LaTeX output is broken
- #2361: Fix additional paragraphs inside the "compound" directive are indented
- #2364: Fix KeyError 'rootSymbol' on Sphinx upgrade from older version.
- #2348: Move amsmath and amssymb to before fontpkg on LaTeX writer.
- #2368: Ignore emacs lock files like `.#foo.rst` by default.
- #2262: literal_block and its caption has been separated by pagebreak in LaTeX output.
- #2319: Fix table counter is overridden by code-block's in LaTeX. Thanks to jfbu.
- Fix unpack warning if combined with 3rd party domain extensions.
- #1153: Fix figures in sidebar causes latex build error.

- #2358: Fix user-preamble could not override the tocdepth definition.
- #2358: Reduce tocdepth if `part` or `chapter` is used for top_sectionlevel
- #2351: Fix footnote spacing
- #2363: Fix `toctree()` in templates generates broken links in SingleHTMLBuilder.
- #2366: Fix empty hyperref is generated on toctree in HTML builder.

Documentation

- #1757: Fix for usage of *`html_last_updated_fmt`*. Thanks to Ralf Hemmecke.

10.77 Release 1.3.6 (released Feb 29, 2016)

Features added

- #1873, #1876, #2278: Add `page_source_suffix` html context variable. This should be introduced with *`source_parsers`* feature. Thanks for Eric Holscher.

Bugs fixed

- #2265: Fix babel is used in spite of disabling it on `latex_elements`
- #2295: Avoid mutating dictionary errors while enumerating members in autodoc with Python 3
- #2291: Fix pdflatex "Counter too large" error from footnotes inside tables of contents
- #2292: Fix some footnotes disappear from LaTeX output
- #2287: `sphinx.transforms.Locale` always uses rst parser. Sphinx i18n feature should support parsers that specified source_parsers.
- #2290: Fix `sphinx.ext.mathbase` use of amsfonts may break user choice of math fonts
- #2324: Print a hint how to increase the recursion limit when it is hit.
- #1565, #2229: Revert new warning; the new warning will be triggered from version 1.4 on.
- #2329: Refresh environment forcedly if source directory has changed.
- #2019: Fix the domain objects in search result are not escaped

10.78 Release 1.3.5 (released Jan 24, 2016)

Bugs fixed

- Fix line numbers was not shown on warnings in LaTeX and texinfo builders
- Fix filenames were not shown on warnings of citations
- Fix line numbers was not shown on warnings in LaTeX and texinfo builders
- Fix line numbers was not shown on warnings of indices
- #2026: Fix LaTeX builder raises error if parsed-literal includes links
- #2243: Ignore strange docstring types for classes, do not crash

- #2247: Fix #2205 breaks make html for definition list with classifiers that contains regular-expression like string
- #1565: Sphinx will now emit a warning that highlighting was skipped if the syntax is incorrect for *code-block*, *literalinclude* and so on.
- #2211: Fix paragraphs in table cell doesn't work in Latex output
- #2253: `:pyobject:` option of `literalinclude` directive can't detect indented body block when the block starts with blank or comment lines.
- Fix TOC is not shown when no `:maxdepth:` for toctrees (ref: #771)
- Fix warning message for `:numref:` if target is in orphaned doc (ref: #2244)

10.79 Release 1.3.4 (released Jan 12, 2016)

Bugs fixed

- #2134: Fix figure caption with reference causes latex build error
- #2094: Fix rubric with reference not working in Latex
- #2147: Fix literalinclude code in latex does not break in pages
- #1833: Fix email addresses is showed again if latex_show_urls is not None
- #2176: sphinx.ext.graphviz: use <object> instead of <img> to embed svg
- #967: Fix SVG inheritance diagram is not hyperlinked (clickable)
- #1237: Fix footnotes not working in definition list in LaTeX
- #2168: Fix raw directive does not work for text writer
- #2171: Fix cannot linkcheck url with unicode
- #2182: LaTeX: support image file names with more than 1 dots
- #2189: Fix previous sibling link for first file in subdirectory uses last file, not intended previous from root toctree
- #2003: Fix decode error under python2 (only) when `make linkcheck` is run
- #2186: Fix LaTeX output of mathbb in math
- #1480, #2188: LaTeX: Support math in section titles
- #2071: Fix same footnote in more than two section titles => LaTeX/PDF Bug
- #2040: Fix UnicodeDecodeError in sphinx-apidoc when author contains non-ascii characters
- #2193: Fix shutil.SameFileError if source directory and destination directory are same
- #2178: Fix unparsable C++ cross-reference when referencing a function with :cpp:any:
- #2206: Fix Sphinx latex doc build failed due to a footnotes
- #2201: Fix wrong table caption for tables with over 30 rows
- #2213: Set <blockquote> in the classic theme to fit with <p>
- #1815: Fix linkcheck does not raise an exception if warniserror set to true and link is broken
- #2197: Fix slightly cryptic error message for missing index.rst file
- #1894: Unlisted phony targets in quickstart Makefile
- #2125: Fix unifies behavior of collapsed fields (`GroupedField` and `TypedField`)
- #1408: Check latex_logo validity before copying

- #771: Fix latex output doesn't set tocdepth
- #1820: On Windows, console coloring is broken with colorama version 0.3.3. Now sphinx use colorama>=0.3.5 to avoid this problem.
- #2072: Fix footnotes in chapter-titles do not appear in PDF output
- #1580: Fix paragraphs in longtable don't work in Latex output
- #1366: Fix centered image not centered in latex
- #1860: Fix man page using `:samp:` with braces - font doesn't reset
- #1610: Sphinx crashes in Japanese indexing in some systems
- Fix Sphinx crashes if mecab initialization failed
- #2160: Fix broken TOC of PDFs if section includes an image
- #2172: Fix dysfunctional admonition `\py@lightbox` in sphinx.sty. Thanks to jfbu.
- #2198,#2205: `make gettext` generate broken msgid for definition lists.
- #2062: Escape characters in doctests are treated incorrectly with Python 2.
- #2225: Fix if the option does not begin with dash, linking is not performed
- #2226: Fix math is not HTML-encoded when :nowrap: is given (jsmath, mathjax)
- #1601, #2220: 'any' role breaks extended domains behavior. Affected extensions doesn't support resolve_any_xref and resolve_xref returns problematic node instead of None. sphinxcontrib-httpdomain is one of them.
- #2229: Fix no warning is given for unknown options

10.80 Release 1.3.3 (released Dec 2, 2015)

Bugs fixed

- #2177: Fix parallel hangs
- #2012: Fix exception occurred if `numfig_format` is invalid
- #2142: Provide non-minified JS code in `sphinx/search/non-minified-js/*.js` for source distribution on PyPI.
- #2148: Error while building devhelp target with non-ASCII document.

10.81 Release 1.3.2 (released Nov 29, 2015)

Features added

- #1935: Make "numfig_format" overridable in latex_elements.

Bugs fixed

- #1976: Avoid "2.0" version of Babel because it doesn't work with Windows environment.
- Add a "default.css" stylesheet (which imports "classic.css") for compatibility
- #1788: graphviz extension raises exception when caption option is present.
- #1789: `:pyobject:` option of `literalinclude` directive includes following lines after class definitions
- #1790: `literalinclude` strips empty lines at the head and tail
- #1802: load plugin themes automatically when theme.conf use it as 'inherit'. Thanks to Takayuki Hirai.
- #1794: custom theme extended from alabaster or sphinx_rtd_theme can't find base theme.
- #1834: compatibility for docutils-0.13: handle_io_errors keyword argument for docutils.io.FileInput cause TypeError.
- #1823: '.' as <module_path> for sphinx-apidoc cause an unfriendly error. Now '.' is converted to absolute path automatically.
- Fix a crash when setting up extensions which do not support metadata.
- #1784: Provide non-minified JS code in `sphinx/search/non-minified-js/*.js`
- #1822, #1892: Fix regression for #1061. autosummary can't generate doc for imported members since sphinx-1.3b3. Thanks to Eric Larson.
- #1793, #1819: "see also" misses a linebreak in text output. Thanks to Takayuki Hirai.
- #1780, #1866: "make text" shows "class" keyword twice. Thanks to Takayuki Hirai.
- #1871: Fix for LaTeX output of tables with one column and multirows.
- Work around the lack of the HTMLParserError exception in Python 3.5.
- #1949: Use `safe_getattr` in the coverage builder to avoid aborting with descriptors that have custom behavior.
- #1915: Do not generate smart quotes in doc field type annotations.
- #1796: On py3, automated .mo building caused UnicodeDecodeError.
- #1923: Use babel features only if the babel latex element is nonempty.
- #1942: Fix a KeyError in websupport.
- #1903: Fix strange id generation for glossary terms.
- `make text` will crush if a definition list item has more than 1 classifiers as: `term : classifier1 : classifier2`.
- #1855: make gettext generates broken po file for definition lists with classifier.
- #1869: Fix problems when dealing with files containing non-ASCII characters. Thanks to Marvin Schmidt.
- #1798: Fix building LaTeX with references in titles.
- #1725: On py2 environment, doctest with using non-ASCII characters causes `'ascii' codec can't decode byte` exception.
- #1540: Fix RuntimeError with circular referenced toctree
- #1983: i18n translation feature breaks references which uses section name.
- #1990: Use caption of toctree to title of tableofcontents in LaTeX
- #1987: Fix ampersand is ignored in `:menuselection:` and `:guilabel:` on LaTeX builder
- #1994: More supporting non-standard parser (like recommonmark parser) for Translation and WebSupport feature. Now node.rawsource is fall backed to node.astext() during docutils transforming.

- #1989: "make blahblah" on Windows indicate help messages for sphinx-build every time. It was caused by wrong make.bat that generated by Sphinx-1.3.0/1.3.1.
- On Py2 environment, conf.py that is generated by sphinx-quickstart should have u prefixed config value for 'version' and 'release'.
- #2102: On Windows + Py3, using `|today|` and non-ASCII date format will raise UnicodeEncodeError.
- #1974: UnboundLocalError: local variable 'domain' referenced before assignment when using *any* role and *sphinx.ext.intersphinx* in same time.
- #2121: multiple words search doesn't find pages when words across on the page title and the page content.
- #1884, #1885: plug-in html themes cannot inherit another plug-in theme. Thanks to Suzumizaki.
- #1818: *sphinx.ext.todo* directive generates broken html class attribute as 'admonition-' when *language* is specified with non-ASCII linguistic area like 'ru' or 'ja'. To fix this, now `todo` directive can use `:class:` option.
- #2140: Fix footnotes in table has broken in LaTeX
- #2127: MecabBinder for html searching feature doesn't work with Python 3. Thanks to Tomoko Uchida.

10.82 Release 1.3.1 (released Mar 17, 2015)

Bugs fixed

- #1769: allows generating quickstart files/dirs for destination dir that doesn't overwrite existent files/dirs. Thanks to WAKAYAMA shirou.
- #1773: sphinx-quickstart doesn't accept non-ASCII character as a option argument.
- #1766: the message "least Python 2.6 to run" is at best misleading.
- #1772: cross reference in docstrings like `:param .write:` breaks building.
- #1770, #1774: `literalinclude` with empty file occurs exception. Thanks to Takayuki Hirai.
- #1777: Sphinx 1.3 can't load extra theme. Thanks to tell-k.
- #1776: `source_suffix = ['.rst']` cause unfriendly error on prior version.
- #1771: automated .mo building doesn't work properly.
- #1783: Autodoc: Python2 Allow unicode string in __all__. Thanks to Jens Hedegaard Nielsen.
- #1781: Setting *html_domain_indices* to a list raises a type check warnings.

10.83 Release 1.3 (released Mar 10, 2015)

Incompatible changes

- Roles `ref`, `term` and `menusel` now don't generate emphasis[613] nodes anymore. If you want to keep italic style, adapt your stylesheet.
- Role `numref` uses `%s` as special character to indicate position of figure numbers instead # symbol.

[613] http://docutils.sourceforge.net/docs/ref/rst/roles.html#emphasis

Features added

- Add convenience directives and roles to the C++ domain: directive `cpp:var` as alias for `cpp:member`, role `:cpp:var` as alias for `:cpp:member`, and role *any* for cross-reference to any C++ declaraction. #1577, #1744
- The *source_suffix* config value can now be a list of multiple suffixes.
- Add the ability to specify source parsers by source suffix with the *source_parsers* config value.
- #1675: A new builder, AppleHelpBuilder, has been added that builds Apple Help Books.

Bugs fixed

- 1.3b3 change breaks a previous gettext output that contains duplicated msgid such as "foo bar" and "version changes in 1.3: foo bar".
- #1745: latex builder cause maximum recursion depth exceeded when a footnote has a footnote mark itself.
- #1748: SyntaxError in sphinx/ext/ifconfig.py with Python 2.6.
- #1658, #1750: No link created (and warning given) if option does not begin with -, / or +. Thanks to Takayuki Hirai.
- #1753: C++, added missing support for more complex declarations.
- #1700: Add `:caption:` option for *toctree*.
- #1742: `:name:` option is provided for *toctree*, *code-block* and *literalinclude* directives.
- #1756: Incorrect section titles in search that was introduced from 1.3b3.
- #1746: C++, fixed name lookup procedure, and added missing lookups in declarations.
- #1765: C++, fix old id generation to use fully qualified names.

Documentation

- #1651: Add `vartype` field description for python domain.

10.84 Release 1.3b3 (released Feb 24, 2015)

Incompatible changes

- Dependency requirement updates: docutils 0.11, Pygments 2.0
- The `gettext_enables` config value has been renamed to *gettext_additional_targets*.
- #1735: Use https://docs.python.org/ instead of `http` protocol. It was used for *sphinx.ext.intersphinx* and some documentation.

Features added

- #1346: Add new default theme;
  - Add 'alabaster' theme.
  - Add 'sphinx_rtd_theme' theme.
  - The 'default' html theme has been renamed to 'classic'. 'default' is still available, however it will emit notice a recommendation that using new 'alabaster' theme.
- Added `highlight_options` configuration value.
- The `language` config value is now available in the HTML templates.
- The `env-updated` event can now return a value, which is interpreted as an iterable of additional docnames that need to be rewritten.
- #772: Support for scoped and unscoped enums in C++. Enumerators in unscoped enums are injected into the parent scope in addition to the enum scope.
- Add `todo_include_todos` config option to quickstart conf file, handled as described in documentation.
- HTML breadcrumb items tag has class "nav-item" and "nav-item-N" (like nav-item-0, 1, 2. . .).
- New option *`sphinx-quickstart --use-make-mode`* for generating Makefile that use sphinx-build make-mode.
- #1235: i18n: several node can be translated if it is set to *`gettext_additional_targets`* in conf.py. Supported nodes are:
  - 'literal-block'
  - 'doctest-block'
  - 'raw'
  - 'image'
- #1227: Add *`html_scaled_image_link`* config option to conf.py, to control scaled image link.

Bugs fixed

- LaTeX writer now generates correct markup for cells spanning multiple rows.
- #1674: Do not crash if a module's `__all__` is not a list of strings.
- #1629: Use VerbatimBorderColor to add frame to code-block in LaTeX
- On windows, make-mode didn't work on Win32 platform if sphinx was invoked as `python sphinx-build.py`.
- #1687: linkcheck now treats 401 Unauthorized responses as "working".
- #1690: toctrees with `glob` option now can also contain entries for single documents with explicit title.
- #1591: html search results for C++ elements now has correct interpage links.
- bizstyle theme: nested long title pages make long breadcrumb that breaks page layout.
- bizstyle theme: all breadcrumb items become 'Top' on some mobile browser (iPhone5s safari).
- #1722: restore `toctree()` template function behavior that was changed at 1.3b1.
- #1732: i18n: localized table caption raises exception.
- #1718: `:numref:` does not work with capital letters in the label
- #1630: resolve CSS conflicts, `div.container` css target for literal block wrapper now renamed to `div.literal-block-wrapper`.

- `sphinx.util.pycompat` has been restored in its backwards-compatibility; slated for removal in Sphinx 1.4.
- #1719: LaTeX writer does not respect `numref_format` option in captions

10.85 Release 1.3b2 (released Dec 5, 2014)

Incompatible changes

- update bundled ez_setup.py for setuptools-7.0 that requires Python 2.6 or later.

Features added

- #1597: Added possibility to return a new template name from *`html-page-context`*.
- PR#314, #1150: Configuration values are now checked for their type. A warning is raised if the configured and the default value do not have the same type and do not share a common non-trivial base class.

Bugs fixed

- PR#311: sphinx-quickstart does not work on python 3.4.
- Fix *`autodoc_docstring_signature`* not working with signatures in class docstrings.
- Rebuilding cause crash unexpectedly when source files were added.
- #1607: Fix a crash when building latexpdf with "howto" class
- #1251: Fix again. Sections which depth are lower than :tocdepth: should not be shown on localtoc sidebar.
- make-mode didn't work on Win32 platform if sphinx was installed by wheel package.

10.86 Release 1.3b1 (released Oct 10, 2014)

Incompatible changes

- Dropped support for Python 2.5, 3.1 and 3.2.
- Dropped support for docutils versions up to 0.9.
- Removed the `sphinx.ext.oldcmarkup` extension.
- The deprecated config values `exclude_trees`, `exclude_dirnames` and `unused_docs` have been removed.
- A new node, `sphinx.addnodes.literal_strong`, has been added, for text that should appear literally (i.e. no smart quotes) in strong font. Custom writers will have to be adapted to handle this node.
- PR#269, #1476: replace <`tt`> tag by <`code`>. User customized stylesheets should be updated If the css contain some styles for `tt`> tag. Thanks to Takeshi Komiya.
- #1543: *`templates_path`* is automatically added to *`exclude_patterns`* to avoid reading autosummary rst templates in the templates directory.
- Custom domains should implement the new *`Domain.resolve_any_xref`* method to make the *`any`* role work properly.
- gettext builder: gettext doesn't emit uuid information to generated pot files by default. Please set `True` to *`gettext_uuid`* to emit uuid information. Additionally, if the `python-levenshtein` 3rd-party package is installed, it will improve the calculation time.

- gettext builder: disable extracting/apply 'index' node by default. Please set 'index' to `gettext_enables` to enable extracting index entries.
- PR#307: Add frame to code-block in LaTeX. Thanks to Takeshi Komiya.

Features added

- Add support for Python 3.4.
- Add support for docutils 0.12
- Added `sphinx.ext.napoleon` extension for NumPy and Google style docstring support.
- Added support for parallel reading (parsing) of source files with the *sphinx-build -j* option. Third-party extensions will need to be checked for compatibility and may need to be adapted if they store information in the build environment object. See *env-merge-info*.
- Added the *any* role that can be used to find a cross-reference of *any* type in *any* domain. Custom domains should implement the new *Domain.resolve_any_xref* method to make this work properly.
- Exception logs now contain the last 10 messages emitted by Sphinx.
- Added support for extension versions (a string returned by `setup()`, these can be shown in the traceback log files). Version requirements for extensions can be specified in projects using the new *needs_extensions* config value.
- Changing the default role within a document with the default-role[614] directive is now supported.
- PR#214: Added stemming support for 14 languages, so that the built-in document search can now handle these. Thanks to Shibukawa Yoshiki.
- PR#296, PR#303, #76: numfig feature: Assign numbers to figures, tables and code-blocks. This feature is configured with *numfig*, *numfig_secnum_depth* and *numfig_format*. Also *numref* role is available. Thanks to Takeshi Komiya.
- PR#202: Allow "." and "~" prefixed references in `:param:` doc fields for Python.
- PR#184: Add *autodoc_mock_imports*, allowing to mock imports of external modules that need not be present when autodocumenting.
- #925: Allow list-typed config values to be provided on the command line, like `-D key=val1,val2`.
- #668: Allow line numbering of *code-block* and *literalinclude* directives to start at an arbitrary line number, with a new `lineno-start` option.
- PR#172, PR#266: The *code-block* and *literalinclude* directives now can have a `caption` option that shows a filename before the code in the output. Thanks to Nasimul Haque, Takeshi Komiya.
- Prompt for the document language in sphinx-quickstart.
- PR#217: Added config values to suppress UUID and location information in generated gettext catalogs.
- PR#236, #1456: apidoc: Add a -M option to put module documentation before submodule documentation. Thanks to Wes Turner and Luc Saffre.
- #1434: Provide non-minified JS files for jquery.js and underscore.js to clarify the source of the minified files.
- PR#252, #1291: Windows color console support. Thanks to meu31.
- PR#255: When generating latex references, also insert latex target/anchor for the ids defined on the node. Thanks to Olivier Heurtier.
- PR#229: Allow registration of other translators. Thanks to Russell Sim.
- Add app.set_translator() API to register or override a Docutils translator class like `html_translator_class`.

[614] http://docutils.sourceforge.net/docs/ref/rst/directives.html#default-role

- PR#267, #1134: add 'diff' parameter to literalinclude. Thanks to Richard Wall and WAKAYAMA shirou.
- PR#272: Added 'bizstyle' theme. Thanks to Shoji KUMAGAI.
- Automatically compile `*.mo` files from `*.po` files when *gettext_auto_build* is True (default) and `*.po` is newer than `*.mo` file.
- #623: *sphinx.ext.viewcode* supports imported function/class aliases.
- PR#275: *sphinx.ext.intersphinx* supports multiple target for the inventory. Thanks to Brigitta Sipocz.
- PR#261: Added the *env-before-read-docs* event that can be connected to modify the order of documents before they are read by the environment.
- #1284: Program options documented with *option* can now start with +.
- PR#291: The caption of *code-block* is recognized as a title of ref target. Thanks to Takeshi Komiya.
- PR#298: Add new API: *add_latex_package()*. Thanks to Takeshi Komiya.
- #1344: add `gettext_enables` to enable extracting 'index' to gettext catalog output / applying translation catalog to generated documentation.
- PR#301, #1583: Allow the line numbering of the directive *literalinclude* to match that of the included file, using a new `lineno-match` option. Thanks to Jeppe Pihl.
- PR#299: add various options to sphinx-quickstart. Quiet mode option `--quiet` will skips wizard mode. Thanks to WAKAYAMA shirou.
- #1623: Return types specified with `:rtype:` are now turned into links if possible.

Bugs fixed

- #1438: Updated jQuery version from 1.8.3 to 1.11.1.
- #1568: Fix a crash when a "centered" directive contains a reference.
- Now sphinx.ext.autodoc works with python-2.5 again.
- #1563: *add_search_language()* raises AssertionError for correct type of argument. Thanks to rikoman.
- #1174: Fix smart quotes being applied inside roles like *program* or *makevar*.
- PR#235: comment db schema of websupport lacked a length of the node_id field. Thanks to solos.
- #1466,PR#241: Fix failure of the cpp domain parser to parse C+11 "variadic templates" declarations. Thanks to Victor Zverovich.
- #1459,PR#244: Fix default mathjax js path point to `http://` that cause mixed-content error on HTTPS server. Thanks to sbrandtb and robo9k.
- PR#157: autodoc remove spurious signatures from @property decorated attributes. Thanks to David Ham.
- PR#159: Add coverage targets to quickstart generated Makefile and make.bat. Thanks to Matthias Troffaes.
- #1251: When specifying toctree :numbered: option and :tocdepth: metadata, sub section number that is larger depth than `:tocdepth:` is shrunk.
- PR#260: Encode underscore in citation labels for latex export. Thanks to Lennart Fricke.
- PR#264: Fix could not resolve xref for figure node with :name: option. Thanks to Takeshi Komiya.
- PR#265: Fix could not capture caption of graphviz node by xref. Thanks to Takeshi Komiya.
- PR#263, #1013, #1103: Rewrite of C++ domain. Thanks to Jakob Lykke Andersen.
  - Hyperlinks to all found nested names and template arguments (#1103).
  - Support for function types everywhere, e.g., in std::function<bool(int, int)> (#1013).

  - Support for virtual functions.
  - Changed interpretation of function arguments to following standard prototype declarations, i.e., void f(arg) means that arg is the type of the argument, instead of it being the name.
  - Updated tests.
  - Updated documentation with elaborate description of what declarations are supported and how the namespace declarations influence declaration and cross-reference lookup.
  - Index names may be different now. Elements are indexed by their fully qualified name. It should be rather easy to change this behaviour and potentially index by namespaces/classes as well.
- PR#258, #939: Add dedent option for `code-block` and `literalinclude`. Thanks to Zafar Siddiqui.
- PR#268: Fix numbering section does not work at singlehtml mode. It still ad-hoc fix because there is a issue that section IDs are conflicted. Thanks to Takeshi Komiya.
- PR#273, #1536: Fix RuntimeError with numbered circular toctree. Thanks to Takeshi Komiya.
- PR#274: Set its URL as a default title value if URL appears in toctree. Thanks to Takeshi Komiya.
- PR#276, #1381: `rfc` and `pep` roles support custom link text. Thanks to Takeshi Komiya.
- PR#277, #1513: highlights for function pointers in argument list of `c:function`. Thanks to Takeshi Komiya.
- PR#278: Fix section entries were shown twice if toctree has been put under only directive. Thanks to Takeshi Komiya.
- #1547: pgen2 tokenizer doesn't recognize `...` literal (Ellipsis for py3).
- PR#294: On LaTeX builder, wrap float environment on writing literal_block to avoid separation of caption and body. Thanks to Takeshi Komiya.
- PR#295, #1520: `make.bat latexpdf` mechanism to `cd` back to the current directory. Thanks to Peter Suter.
- PR#297, #1571: Add imgpath property to all builders. It make easier to develop builder extensions. Thanks to Takeshi Komiya.
- #1584: Point to master doc in HTML "top" link.
- #1585: Autosummary of modules broken in Sphinx-1.2.3.
- #1610: Sphinx cause AttributeError when MeCab search option is enabled and python-mecab is not installed.
- #1674: Do not crash if a module's `__all__` is not a list of strings.
- #1673: Fix crashes with `nitpick_ignore` and `:doc:` references.
- #1686: ifconfig directive doesn't care about default config values.
- #1642: Fix only one search result appearing in Chrome.

Documentation

- Add clarification about the syntax of tags. (`doc/markup/misc.rst`)

10.87 Release 1.2.3 (released Sep 1, 2014)

Features added

- #1518: `sphinx-apidoc` command now has a `--version` option to show version information and exit
- New locales: Hebrew, European Portuguese, Vietnamese.

Bugs fixed

- #636: Keep straight single quotes in literal blocks in the LaTeX build.
- #1419: Generated i18n sphinx.js files are missing message catalog entries from '.js_t' and '.html'. The issue was introduced from Sphinx-1.1
- #1363: Fix i18n: missing python domain's cross-references with currentmodule directive or currentclass directive.
- #1444: autosummary does not create the description from attributes docstring.
- #1457: In python3 environment, make linkcheck cause "Can't convert 'bytes' object to str implicitly" error when link target url has a hash part. Thanks to Jorge_C.
- #1467: Exception on Python3 if nonexistent method is specified by automethod
- #1441: autosummary can't handle nested classes correctly.
- #1499: With non-callable `setup` in a conf.py, now sphinx-build emits a user-friendly error message.
- #1502: In autodoc, fix display of parameter defaults containing backslashes.
- #1226: autodoc, autosummary: importing setup.py by automodule will invoke setup process and execute `sys.exit()`. Now sphinx avoids SystemExit exception and emits warnings without unexpected termination.
- #1503: py:function directive generate incorrectly signature when specifying a default parameter with an empty list `[]`. Thanks to Geert Jansen.
- #1508: Non-ASCII filename raise exception on make singlehtml, latex, man, texinfo and changes.
- #1531: On Python3 environment, docutils.conf with 'source_link=true' in the general section cause type error.
- PR#270, #1533: Non-ASCII docstring cause UnicodeDecodeError when uses with inheritance-diagram directive. Thanks to WAKAYAMA shirou.
- PR#281, PR#282, #1509: TODO extension not compatible with websupport. Thanks to Takeshi Komiya.
- #1477: gettext does not extract nodes.line in a table or list.
- #1544: `make text` generates wrong table when it has empty table cells.
- #1522: Footnotes from table get displayed twice in LaTeX. This problem has been appeared from Sphinx-1.2.1 by #949.
- #508: Sphinx every time exit with zero when is invoked from setup.py command. ex. `python setup.py build_sphinx -b doctest` return zero even if doctest failed.

10.88 Release 1.2.2 (released Mar 2, 2014)

Bugs fixed

- PR#211: When checking for existence of the *html_logo* file, check the full relative path and not the basename.
- PR#212: Fix traceback with autodoc and `__init__` methods without docstring.
- PR#213: Fix a missing import in the setup command.
- #1357: Option names documented by *option* are now again allowed to not start with a dash or slash, and referencing them will work correctly.
- #1358: Fix handling of image paths outside of the source directory when using the "wildcard" style reference.
- #1374: Fix for autosummary generating overly-long summaries if first line doesn't end with a period.
- #1383: Fix Python 2.5 compatibility of sphinx-apidoc.
- #1391: Actually prevent using "pngmath" and "mathjax" extensions at the same time in sphinx-quickstart.
- #1386: Fix bug preventing more than one theme being added by the entry point mechanism.
- #1370: Ignore "toctree" nodes in text writer, instead of raising.
- #1364: Fix 'make gettext' fails when the '.. todolist::' directive is present.
- #1367: Fix a change of PR#96 that break sphinx.util.docfields.Field.make_field interface/behavior for `item` argument usage.

Documentation

- Extended the *documentation about building extensions*.

10.89 Release 1.2.1 (released Jan 19, 2014)

Bugs fixed

- #1335: Fix autosummary template overloading with exclamation prefix like `{% extends "!autosummary/class.rst" %}` cause infinite recursive function call. This was caused by PR#181.
- #1337: Fix autodoc with `autoclass_content="both"` uses useless `object.__init__` docstring when class does not have `__init__`. This was caused by a change for #1138.
- #1340: Can't search alphabetical words on the HTML quick search generated with language='ja'.
- #1319: Do not crash if the *html_logo* file does not exist.
- #603: Do not use the HTML-ized title for building the search index (that resulted in "literal" being found on every page with a literal in the title).
- #751: Allow production lists longer than a page in LaTeX by using longtable.
- #764: Always look for stopwords lowercased in JS search.
- #814: autodoc: Guard against strange type objects that don't have `__bases__`.
- #932: autodoc: Do not crash if `__doc__` is not a string.
- #933: Do not crash if an *option* value is malformed (contains spaces but no option name).
- #908: On Python 3, handle error messages from LaTeX correctly in the pngmath extension.
- #943: In autosummary, recognize "first sentences" to pull from the docstring if they contain uppercase letters.

- #923: Take the entire LaTeX document into account when caching pngmath-generated images. This rebuilds them correctly when `pngmath_latex_preamble` changes.
- #901: Emit a warning when using docutils' new "math" markup without a Sphinx math extension active.
- #845: In code blocks, when the selected lexer fails, display line numbers nevertheless if configured.
- #929: Support parsed-literal blocks in LaTeX output correctly.
- #949: Update the tabulary.sty packed with Sphinx.
- #1050: Add anonymous labels into `objects.inv` to be referenced via *`intersphinx`*.
- #1095: Fix print-media stylesheet being included always in the "scrolls" theme.
- #1085: Fix current classname not getting set if class description has `:noindex:` set.
- #1181: Report option errors in autodoc directives more gracefully.
- #1155: Fix autodocumenting C-defined methods as attributes in Python 3.
- #1233: Allow finding both Python classes and exceptions with the "class" and "exc" roles in intersphinx.
- #1198: Allow "image" for the "figwidth" option of the figure[615] directive as documented by docutils.
- #1152: Fix pycode parsing errors of Python 3 code by including two grammar versions for Python 2 and 3, and loading the appropriate version for the running Python version.
- #1017: Be helpful and tell the user when the argument to *`option`* does not match the required format.
- #1345: Fix two bugs with *`nitpick_ignore`*; now you don't have to remove the store environment for changes to have effect.
- #1072: In the JS search, fix issues searching for upper-cased words by lowercasing words before stemming.
- #1299: Make behavior of the *`math`* directive more consistent and avoid producing empty environments in LaTeX output.
- #1308: Strip HTML tags from the content of "raw" nodes before feeding it to the search indexer.
- #1249: Fix duplicate LaTeX page numbering for manual documents.
- #1292: In the linkchecker, retry HEAD requests when denied by HTTP 405. Also make the redirect code apparent and tweak the output a bit to be more obvious.
- #1285: Avoid name clashes between C domain objects and section titles.
- #848: Always take the newest code in incremental rebuilds with the *`sphinx.ext.viewcode`* extension.
- #979, #1266: Fix exclude handling in `sphinx-apidoc`.
- #1302: Fix regression in *`sphinx.ext.inheritance_diagram`* when documenting classes that can't be pickled.
- #1316: Remove hard-coded `font-face` resources from epub theme.
- #1329: Fix traceback with empty translation msgstr in .po files.
- #1300: Fix references not working in translated documents in some instances.
- #1283: Fix a bug in the detection of changed files that would try to access doctrees of deleted documents.
- #1330: Fix *`exclude_patterns`* behavior with subdirectories in the *`html_static_path`*.
- #1323: Fix emitting empty `<ul>` tags in the HTML writer, which is not valid HTML.
- #1147: Don't emit a sidebar search box in the "singlehtml" builder.

[615] http://docutils.sourceforge.net/docs/ref/rst/directives.html#figure

Documentation

- #1325: Added a "Intersphinx" tutorial section. (`doc/tutorial.rst`)

10.90 Release 1.2 (released Dec 10, 2013)

Features added

- Added `sphinx.version_info` tuple for programmatic checking of the Sphinx version.

Incompatible changes

- Removed the `sphinx.ext.refcounting` extension – it is very specific to CPython and has no place in the main distribution.

Bugs fixed

- Restore `versionmodified` CSS class for versionadded/changed and deprecated directives.
- PR#181: Fix `html_theme_path = ['.']` is a trigger of rebuild all documents always (This change keeps the current "theme changes cause a rebuild" feature).
- #1296: Fix invalid charset in HTML help generated HTML files for default locale.
- PR#190: Fix gettext does not extract figure caption and rubric title inside other blocks. Thanks to Michael Schlenker.
- PR#176: Make sure setup_command test can always import Sphinx. Thanks to Dmitry Shachnev.
- #1311: Fix test_linkcode.test_html fails with C locale and Python 3.
- #1269: Fix ResourceWarnings with Python 3.2 or later.
- #1138: Fix: When `autodoc_docstring_signature = True` and `autoclass_content = 'init'` or `'both'`, __init__ line should be removed from class documentation.

10.91 Release 1.2 beta3 (released Oct 3, 2013)

Features added

- The Sphinx error log files will now include a list of the loaded extensions for help in debugging.

Incompatible changes

- PR#154: Remove "sphinx" prefix from LaTeX class name except 'sphinxmanual' and 'sphinxhowto'. Now you can use your custom document class without 'sphinx' prefix. Thanks to Erik B.

Bugs fixed

- #1265: Fix i18n: crash when translating a section name that is pointed to from a named target.
- A wrong condition broke the search feature on first page that is usually index.rst. This issue was introduced in 1.2b1.
- #703: When Sphinx can't decode filenames with non-ASCII characters, Sphinx now catches UnicodeError and will continue if possible instead of raising the exception.

10.92 Release 1.2 beta2 (released Sep 17, 2013)

Features added

- `apidoc` now ignores "_private" modules by default, and has an option `-P` to include them.
- `apidoc` now has an option to not generate headings for packages and modules, for the case that the module docstring already includes a reST heading.
- PR#161: `apidoc` can now write each module to a standalone page instead of combining all modules in a package on one page.
- Builders: rebuild i18n target document when catalog updated.
- Support docutils.conf 'writers' and 'html4css1 writer' section in the HTML writer. The latex, manpage and texinfo writers also support their respective 'writers' sections.
- The new *`html_extra_path`* config value allows to specify directories with files that should be copied directly to the HTML output directory.
- Autodoc directives for module data and attributes now support an `annotation` option, so that the default display of the data/attribute value can be overridden.
- PR#136: Autodoc directives now support an `imported-members` option to include members imported from different modules.
- New locales: Macedonian, Sinhala, Indonesian.
- Theme package collection by using setuptools plugin mechanism.

Incompatible changes

- PR#144, #1182: Force timezone offset to LocalTimeZone on POT-Creation-Date that was generated by gettext builder. Thanks to masklinn and Jakub Wilk.

Bugs fixed

- PR#132: Updated jQuery version to 1.8.3.
- PR#141, #982: Avoid crash when writing PNG file using Python 3. Thanks to Marcin Wojdyr.
- PR#145: In parallel builds, sphinx drops second document file to write. Thanks to tychoish.
- PR#151: Some styling updates to tables in LaTeX.
- PR#153: The "extensions" config value can now be overridden.
- PR#155: Added support for some C++11 function qualifiers.
- Fix: 'make gettext' caused UnicodeDecodeError when templates contain utf-8 encoded strings.

- #828: use inspect.getfullargspec() to be able to document functions with keyword-only arguments on Python 3.
- #1090: Fix i18n: multiple cross references (term, ref, doc) in the same line return the same link.
- #1157: Combination of 'globaltoc.html' and hidden toctree caused exception.
- #1159: fix wrong generation of objects inventory for Python modules, and add a workaround in intersphinx to fix handling of affected inventories.
- #1160: Citation target missing caused an AssertionError.
- #1162, PR#139: singlehtml builder didn't copy images to _images/.
- #1173: Adjust setup.py dependencies because Jinja2 2.7 discontinued compatibility with Python < 3.3 and Python < 2.6. Thanks to Alexander Dupuy.
- #1185: Don't crash when a Python module has a wrong or no encoding declared, and non-ASCII characters are included.
- #1188: sphinx-quickstart raises UnicodeEncodeError if "Project version" includes non-ASCII characters.
- #1189: "Title underline is too short" WARNING is given when using fullwidth characters to "Project name" on quickstart.
- #1190: Output TeX/texinfo/man filename has no basename (only extension) when using non-ASCII characters in the "Project name" on quickstart.
- #1192: Fix escaping problem for hyperlinks in the manpage writer.
- #1193: Fix i18n: multiple link references in the same line return the same link.
- #1176: Fix i18n: footnote reference number missing for auto numbered named footnote and auto symbol footnote.
- PR#146,#1172: Fix ZeroDivisionError in parallel builds. Thanks to tychoish.
- #1204: Fix wrong generation of links to local intersphinx targets.
- #1206: Fix i18n: gettext did not translate admonition directive's title.
- #1232: Sphinx generated broken ePub files on Windows.
- #1259: Guard the debug output call when emitting events; to prevent the repr() implementation of arbitrary objects causing build failures.
- #1142: Fix NFC/NFD normalizing problem of rst filename on Mac OS X.
- #1234: Ignoring the string consists only of white-space characters.

10.93 Release 1.2 beta1 (released Mar 31, 2013)

Incompatible changes

- Removed `sphinx.util.compat.directive_dwim()` and `sphinx.roles.xfileref_role()` which were deprecated since version 1.0.
- PR#122: the files given in *`latex_additional_files`* now override TeX files included by Sphinx, such as `sphinx.sty`.
- PR#124: the node generated by *`versionadded`*, *`versionchanged`* and *`deprecated`* directives now includes all added markup (such as "New in version X") as child nodes, and no additional text must be generated by writers.
- PR#99: the *`seealso`* directive now generates admonition nodes instead of the custom `seealso` node.

Features added

- Markup
  - The *toctree* directive and the `toctree()` template function now have an `includehidden` option that includes hidden toctree entries (bugs #790 and #1047). A bug in the `maxdepth` option for the `toctree()` template function has been fixed (bug #1046).
  - PR#99: Strip down seealso directives to normal admonitions. This removes their unusual CSS classes (admonition-see-also), inconsistent LaTeX admonition title ("See Also" instead of "See also"), and spurious indentation in the text builder.
- HTML builder
  - #783: Create a link to full size image if it is scaled with width or height.
  - #1067: Improve the ordering of the JavaScript search results: matches in titles come before matches in full text, and object results are better categorized. Also implement a pluggable search scorer.
  - #1053: The "rightsidebar" and "collapsiblesidebar" HTML theme options now work together.
  - Update to jQuery 1.7.1 and Underscore.js 1.3.1.
- Texinfo builder
  - An "Index" node is no longer added when there are no entries.
  - "deffn" categories are no longer capitalized if they contain capital letters.
  - `desc_annotation` nodes are now rendered.
  - `strong` and `emphasis` nodes are now formatted like `literal`s. The reason for this is because the standard Texinfo markup (`*strong*` and `_emphasis_`) resulted in confusing output due to the common usage of using these constructs for documenting parameter names.
  - Field lists formatting has been tweaked to better display "Info field lists".
  - `system_message` and `problematic` nodes are now formatted in a similar fashion as done by the text builder.
  - "en-dash" and "em-dash" conversion of hyphens is no longer performed in option directive signatures.
  - `@ref` is now used instead of `@pxref` for cross-references which prevents the word "see" from being added before the link (does not affect the Info output).
  - The `@finalout` command has been added for better TeX output.
  - `transition` nodes are now formatted using underscores ("_") instead of asterisks ("*").
  - The default value for the `paragraphindent` has been changed from 2 to 0 meaning that paragraphs are no longer indented by default.
  - #1110: A new configuration value *texinfo_no_detailmenu* has been added for controlling whether a `@detailmenu` is added in the "Top" node's menu.
  - Detailed menus are no longer created except for the "Top" node.
  - Fixed an issue where duplicate domain indices would result in invalid output.
- LaTeX builder:
  - PR#115: Add `'transition'` item in *latex_elements* for customizing how transitions are displayed. Thanks to Jeff Klukas.
  - PR#114: The LaTeX writer now includes the "cmap" package by default. The `'cmappkg'` item in *latex_elements* can be used to control this. Thanks to Dmitry Shachnev.
  - The `'fontpkg'` item in *latex_elements* now defaults to `''` when the *language* uses the Cyrillic script. Suggested by Dmitry Shachnev.

  - The *latex_documents*, *texinfo_documents*, and *man_pages* configuration values will be set to default values based on the *master_doc* if not explicitly set in `conf.py`. Previously, if these values were not set, no output would be generated by their respective builders.

- Internationalization:
  - Add i18n capabilities for custom templates. For example: The Sphinx reference documentation in doc directory provides a `sphinx.pot` file with message strings from `doc/_templates/*.html` when using `make gettext`.
  - PR#61,#703: Add support for non-ASCII filename handling.
- Other builders:
  - Added the Docutils-native XML and pseudo-XML builders. See `XMLBuilder` and `PseudoXMLBuilder`.
  - PR#45: The linkcheck builder now checks `#anchors` for existence.
  - PR#123, #1106: Add *epub_use_index* configuration value. If provided, it will be used instead of *html_use_index* for epub builder.
  - PR#126: Add *epub_tocscope* configuration value. The setting controls the generation of the epub toc. The user can now also include hidden toc entries.
  - PR#112: Add *epub_show_urls* configuration value.
- Extensions:
  - PR#52: `special_members` flag to autodoc now behaves like `members`.
  - PR#47: Added *sphinx.ext.linkcode* extension.
  - PR#25: In inheritance diagrams, the first line of the class docstring is now the tooltip for the class.
- Command-line interfaces:
  - PR#75: Added `--follow-links` option to sphinx-apidoc.
  - #869: sphinx-build now has the option `-T` for printing the full traceback after an unhandled exception.
  - sphinx-build now supports the standard `--help` and `--version` options.
  - sphinx-build now provides more specific error messages when called with invalid options or arguments.
  - sphinx-build now has a verbose option `-v` which can be repeated for greater effect. A single occurrence provides a slightly more verbose output than normal. Two or more occurrences of this option provides more detailed output which may be useful for debugging.
- Locales:
  - PR#74: Fix some Russian translation.
  - PR#54: Added Norwegian bokmaal translation.
  - PR#35: Added Slovak translation.
  - PR#28: Added Hungarian translation.
  - #1113: Add Hebrew locale.
  - #1097: Add Basque locale.
  - #1037: Fix typos in Polish translation. Thanks to Jakub Wilk.
  - #1012: Update Estonian translation.
- Optimizations:
  - Speed up building the search index by caching the results of the word stemming routines. Saves about 20 seconds when building the Python documentation.
  - PR#108: Add experimental support for parallel building with a new *sphinx-build -j* option.

Documentation

- PR#88: Added the "Sphinx Developer's Guide" (`doc/devguide.rst`) which outlines the basic development process of the Sphinx project.
- Added a detailed "Installing Sphinx" document (`doc/install.rst`).

Bugs fixed

- PR#124: Fix paragraphs in versionmodified are ignored when it has no dangling paragraphs. Fix wrong html output (nested <p> tag). Fix versionmodified is not translatable. Thanks to Nozomu Kaneko.
- PR#111: Respect add_autodoc_attrgetter() even when inherited-members is set. Thanks to A. Jesse Jiryu Davis.
- PR#97: Fix footnote handling in translated documents.
- Fix text writer not handling visit_legend for figure directive contents.
- Fix text builder not respecting wide/fullwidth characters: title underline width, table layout width and text wrap width.
- Fix leading space in LaTeX table header cells.
- #1132: Fix LaTeX table output for multi-row cells in the first column.
- #1128: Fix Unicode errors when trying to format time strings with a non-standard locale.
- #1127: Fix traceback when autodoc tries to tokenize a non-Python file.
- #1126: Fix double-hyphen to en-dash conversion in wrong places such as command-line option names in LaTeX.
- #1123: Allow whitespaces in filenames given to *`literalinclude`*.
- #1120: Added improvements about i18n for themes "basic", "haiku" and "scrolls" that Sphinx built-in. Thanks to Leonardo J. Caballero G.
- #1118: Updated Spanish translation. Thanks to Leonardo J. Caballero G.
- #1117: Handle .pyx files in sphinx-apidoc.
- #1112: Avoid duplicate download files when referenced from documents in different ways (absolute/relative).
- #1111: Fix failure to find uppercase words in search when *`html_search_language`* is 'ja'. Thanks to Tomo Saito.
- #1108: The text writer now correctly numbers enumerated lists with non-default start values (based on patch by Ewan Edwards).
- #1102: Support multi-context "with" statements in autodoc.
- #1090: Fix gettext not extracting glossary terms.
- #1074: Add environment version info to the generated search index to avoid compatibility issues with old builds.
- #1070: Avoid un-pickling issues when running Python 3 and the saved environment was created under Python 2.
- #1069: Fixed error caused when autodoc would try to format signatures of "partial" functions without keyword arguments (patch by Artur Gaspar).
- #1062: sphinx.ext.autodoc use __init__ method signature for class signature.
- #1055: Fix web support with relative path to source directory.
- #1043: Fix sphinx-quickstart asking again for yes/no questions because `input()` returns values with an extra 'r' on Python 3.2.0 + Windows. Thanks to Régis Décamps.
- #1041: Fix failure of the cpp domain parser to parse a const type with a modifier.

- #1038: Fix failure of the cpp domain parser to parse C+11 "static constexpr" declarations. Thanks to Jakub Wilk.
- #1029: Fix intersphinx_mapping values not being stable if the mapping has plural key/value set with Python 3.3.
- #1028: Fix line block output in the text builder.
- #1024: Improve Makefile/make.bat error message if Sphinx is not found. Thanks to Anatoly Techtonik.
- #1018: Fix "container" directive handling in the text builder.
- #1015: Stop overriding jQuery contains() in the JavaScript.
- #1010: Make pngmath images transparent by default; IE7+ should handle it.
- #1008: Fix test failures with Python 3.3.
- #995: Fix table-of-contents and page numbering for the LaTeX "howto" class.
- #976: Fix gettext does not extract index entries.
- PR#72: #975: Fix gettext not extracting definition terms before docutils 0.10.
- #961: Fix LaTeX output for triple quotes in code snippets.
- #958: Do not preserve `environment.pickle` after a failed build.
- #955: Fix i18n transformation.
- #940: Fix gettext does not extract figure caption.
- #920: Fix PIL packaging issue that allowed to import `Image` without PIL namespace. Thanks to Marc Schlaich.
- #723: Fix the search function on local files in WebKit based browsers.
- #440: Fix coarse timestamp resolution in some filesystem generating a wrong list of outdated files.

10.94 Release 1.1.3 (Mar 10, 2012)

- PR#40: Fix `safe_repr` function to decode bytestrings with non-ASCII characters correctly.
- PR#37: Allow configuring sphinx-apidoc via `SPHINX_APIDOC_OPTIONS`.
- PR#34: Restore Python 2.4 compatibility.
- PR#36: Make the "bibliography to TOC" fix in LaTeX output specific to the document class.
- #695: When the highlight language "python" is specified explicitly, do not try to parse the code to recognize non-Python snippets.
- #859: Fix exception under certain circumstances when not finding appropriate objects to link to.
- #860: Do not crash when encountering invalid doctest examples, just emit a warning.
- #864: Fix crash with some settings of *`modindex_common_prefix`*.
- #862: Fix handling of `-D` and `-A` options on Python 3.
- #851: Recognize and warn about circular toctrees, instead of running into recursion errors.
- #853: Restore compatibility with docutils trunk.
- #852: Fix HtmlHelp index entry links again.
- #854: Fix inheritance_diagram raising attribute errors on builtins.
- #832: Fix crashes when putting comments or lone terms in a glossary.
- #834, #818: Fix HTML help language/encoding mapping for all Sphinx supported languages.
- #844: Fix crashes when dealing with Unicode output in doctest extension.
- #831: Provide `--project` flag in setup_command as advertised.

- #875: Fix reading config files under Python 3.
- #876: Fix quickstart test under Python 3.
- #870: Fix spurious KeyErrors when removing documents.
- #892: Fix single-HTML builder misbehaving with the master document in a subdirectory.
- #873: Fix assertion errors with empty `only` directives.
- #816: Fix encoding issues in the Qt help builder.

10.95 Release 1.1.2 (Nov 1, 2011) – 1.1.1 is a silly version number anyway!

- #809: Include custom fixers in the source distribution.

10.96 Release 1.1.1 (Nov 1, 2011)

- #791: Fix QtHelp, DevHelp and HtmlHelp index entry links.
- #792: Include "sphinx-apidoc" in the source distribution.
- #797: Don't crash on a misformatted glossary.
- #801: Make intersphinx work properly without SSL support.
- #805: Make the `Sphinx.add_index_to_domain` method work correctly.
- #780: Fix Python 2.5 compatibility.

10.97 Release 1.1 (Oct 9, 2011)

Incompatible changes

- The *`py:module`* directive doesn't output its `platform` option value anymore. (It was the only thing that the directive did output, and therefore quite inconsistent.)
- Removed support for old dependency versions; requirements are now:
  - Pygments >= 1.2
  - Docutils >= 0.7
  - Jinja2 >= 2.3

Features added

- Added Python 3.x support.
- New builders and subsystems:
  - Added a Texinfo builder.
  - Added i18n support for content, a `gettext` builder and related utilities.
  - Added the `websupport` library and builder.

  - #98: Added a `sphinx-apidoc` script that autogenerates a hierarchy of source files containing autodoc directives to document modules and packages.
  - #273: Add an API for adding full-text search support for languages other than English. Add support for Japanese.

- Markup:
  - #138: Added an *index* role, to make inline index entries.
  - #454: Added more index markup capabilities: marking see/seealso entries, and main entries for a given key.
  - #460: Allowed limiting the depth of section numbers for HTML using the *toctree*'s `numbered` option.
  - #586: Implemented improved *glossary* markup which allows multiple terms per definition.
  - #478: Added *py:decorator* directive to describe decorators.
  - C++ domain now supports array definitions.
  - C++ domain now supports doc fields (`:param x:` inside directives).
  - Section headings in *only* directives are now correctly handled.
  - Added `emphasize-lines` option to source code directives.
  - #678: C++ domain now supports superclasses.
- HTML builder:
  - Added `pyramid` theme.
  - #559: *html_add_permalinks* is now a string giving the text to display in permalinks.
  - #259: HTML table rows now have even/odd CSS classes to enable "Zebra styling".
  - #554: Add theme option `sidebarwidth` to the basic theme.
- Other builders:
  - #516: Added new value of the *latex_show_urls* option to show the URLs in footnotes.
  - #209: Added *text_newlines* and *text_sectionchars* config values.
  - Added *man_show_urls* config value.
  - #472: linkcheck builder: Check links in parallel, use HTTP HEAD requests and allow configuring the timeout. New config values: *linkcheck_timeout* and *linkcheck_workers*.
  - #521: Added *linkcheck_ignore* config value.
  - #28: Support row/colspans in tables in the LaTeX builder.
- Configuration and extensibility:
  - #537: Added *nitpick_ignore*.
  - #306: Added *env-get-outdated* event.
  - `Application.add_stylesheet()` now accepts full URIs.
- Autodoc:
  - #564: Add *autodoc_docstring_signature*. When enabled (the default), autodoc retrieves the signature from the first line of the docstring, if it is found there.
  - #176: Provide `private-members` option for autodoc directives.
  - #520: Provide `special-members` option for autodoc directives.
  - #431: Doc comments for attributes can now be given on the same line as the assignment.
  - #437: autodoc now shows values of class data attributes.

  - autodoc now supports documenting the signatures of `functools.partial` objects.
- Other extensions:
  - Added the *sphinx.ext.mathjax* extension.
  - #443: Allow referencing external graphviz files.
  - Added `inline` option to graphviz directives, and fixed the default (block-style) in LaTeX output.
  - #590: Added `caption` option to graphviz directives.
  - #553: Added *testcleanup* blocks in the doctest extension.
  - #594: *trim_doctest_flags* now also removes `<BLANKLINE>` indicators.
  - #367: Added automatic exclusion of hidden members in inheritance diagrams, and an option to selectively enable it.
  - Added `pngmath_add_tooltips`.
  - The math extension displaymath directives now support `name` in addition to `label` for giving the equation label, for compatibility with Docutils.
- New locales:
  - #221: Added Swedish locale.
  - #526: Added Iranian locale.
  - #694: Added Latvian locale.
  - Added Nepali locale.
  - #714: Added Korean locale.
  - #766: Added Estonian locale.
- Bugs fixed:
  - #778: Fix "hide search matches" link on pages linked by search.
  - Fix the source positions referenced by the "viewcode" extension.

10.98 Release 1.0.8 (Sep 23, 2011)

- #627: Fix tracebacks for AttributeErrors in autosummary generation.
- Fix the `abbr` role when the abbreviation has newlines in it.
- #727: Fix the links to search results with custom object types.
- #648: Fix line numbers reported in warnings about undefined references.
- #696, #666: Fix C++ array definitions and template arguments that are not type names.
- #633: Allow footnotes in section headers in LaTeX output.
- #616: Allow keywords to be linked via intersphinx.
- #613: Allow Unicode characters in production list token names.
- #720: Add dummy visitors for graphviz nodes for text and man.
- #704: Fix image file duplication bug.
- #677: Fix parsing of multiple signatures in C++ domain.
- #637: Ignore Emacs lock files when looking for source files.
- #544: Allow .pyw extension for importable modules in autodoc.

- #700: Use `$(MAKE)` in quickstart-generated Makefiles.
- #734: Make sidebar search box width consistent in browsers.
- #644: Fix spacing of centered figures in HTML output.
- #767: Safely encode SphinxError messages when printing them to sys.stderr.
- #611: Fix LaTeX output error with a document with no sections but a link target.
- Correctly treat built-in method descriptors as methods in autodoc.
- #706: Stop monkeypatching the Python textwrap module.
- #657: viewcode now works correctly with source files that have non-ASCII encoding.
- #669: Respect the `noindex` flag option in py:module directives.
- #675: Fix IndexErrors when including nonexisting lines with *`literalinclude`*.
- #676: Respect custom function/method parameter separator strings.
- #682: Fix JS incompatibility with jQuery >= 1.5.
- #693: Fix double encoding done when writing HTMLHelp .hhk files.
- #647: Do not apply SmartyPants in parsed-literal blocks.
- C++ domain now supports array definitions.

10.99 Release 1.0.7 (Jan 15, 2011)

- #347: Fix wrong generation of directives of static methods in autosummary.
- #599: Import PIL as `from PIL import Image`.
- #558: Fix longtables with captions in LaTeX output.
- Make token references work as hyperlinks again in LaTeX output.
- #572: Show warnings by default when reference labels cannot be found.
- #536: Include line number when complaining about missing reference targets in nitpicky mode.
- #590: Fix inline display of graphviz diagrams in LaTeX output.
- #589: Build using app.build() in setup command.
- Fix a bug in the inheritance diagram exception that caused base classes to be skipped if one of them is a builtin.
- Fix general index links for C++ domain objects.
- #332: Make admonition boundaries in LaTeX output visible.
- #573: Fix KeyErrors occurring on rebuild after removing a file.
- Fix a traceback when removing files with globbed toctrees.
- If an autodoc object cannot be imported, always re-read the document containing the directive on next build.
- If an autodoc object cannot be imported, show the full traceback of the import error.
- Fix a bug where the removal of download files and images wasn't noticed.
- #571: Implement ~ cross-reference prefix for the C domain.
- Fix regression of LaTeX output with the fix of #556.
- #568: Fix lookup of class attribute documentation on descriptors so that comment documentation now works.
- Fix traceback with `only` directives preceded by targets.

- Fix tracebacks occurring for duplicate C++ domain objects.
- Fix JavaScript domain links to objects with `$` in their name.

10.100 Release 1.0.6 (Jan 04, 2011)

- #581: Fix traceback in Python domain for empty cross-reference targets.
- #283: Fix literal block display issues on Chrome browsers.
- #383, #148: Support sorting a limited range of accented characters in the general index and the glossary.
- #570: Try decoding `-D` and `-A` command-line arguments with the locale's preferred encoding.
- #528: Observe *`locale_dirs`* when looking for the JS translations file.
- #574: Add special code for better support of Japanese documents in the LaTeX builder.
- Regression of #77: If there is only one parameter given with `:param:` markup, the bullet list is now suppressed again.
- #556: Fix missing paragraph breaks in LaTeX output in certain situations.
- #567: Emit the `autodoc-process-docstring` event even for objects without a docstring so that it can add content.
- #565: In the LaTeX builder, not only literal blocks require different table handling, but also quite a few other list-like block elements.
- #515: Fix tracebacks in the viewcode extension for Python objects that do not have a valid signature.
- Fix strange reports of line numbers for warnings generated from autodoc-included docstrings, due to different behavior depending on docutils version.
- Several fixes to the C++ domain.

10.101 Release 1.0.5 (Nov 12, 2010)

- #557: Add CSS styles required by docutils 0.7 for aligned images and figures.
- In the Makefile generated by LaTeX output, do not delete pdf files on clean; they might be required images.
- #535: Fix LaTeX output generated for line blocks.
- #544: Allow `.pyw` as a source file extension.

10.102 Release 1.0.4 (Sep 17, 2010)

- #524: Open intersphinx inventories in binary mode on Windows, since version 2 contains zlib-compressed data.
- #513: Allow giving non-local URIs for JavaScript files, e.g. in the JSMath extension.
- #512: Fix traceback when `intersphinx_mapping` is empty.

10.103 Release 1.0.3 (Aug 23, 2010)

- #495: Fix internal vs. external link distinction for links coming from a docutils table-of-contents.
- #494: Fix the `maxdepth` option for the `toctree()` template callable when used with `collapse=True`.
- #507: Fix crash parsing Python argument lists containing brackets in string literals.
- #501: Fix regression when building LaTeX docs with figures that don't have captions.
- #510: Fix inheritance diagrams for classes that are not picklable.
- #497: Introduce separate background color for the sidebar collapse button, making it easier to see.
- #502, #503, #496: Fix small layout bugs in several builtin themes.

10.104 Release 1.0.2 (Aug 14, 2010)

- #490: Fix cross-references to objects of types added by the `add_object_type()` API function.
- Fix handling of doc field types for different directive types.
- Allow breaking long signatures, continuing with backlash-escaped newlines.
- Fix unwanted styling of C domain references (because of a namespace clash with Pygments styles).
- Allow references to PEPs and RFCs with explicit anchors.
- #471: Fix LaTeX references to figures.
- #482: When doing a non-exact search, match only the given type of object.
- #481: Apply non-exact search for Python reference targets with `.name` for modules too.
- #484: Fix crash when duplicating a parameter in an info field list.
- #487: Fix setting the default role to one provided by the `oldcmarkup` extension.
- #488: Fix crash when json-py is installed, which provides a `json` module but is incompatible to simplejson.
- #480: Fix handling of target naming in intersphinx.
- #486: Fix removal of `!` for all cross-reference roles.

10.105 Release 1.0.1 (Jul 27, 2010)

- #470: Fix generated target names for reST domain objects; they are not in the same namespace.
- #266: Add Bengali language.
- #473: Fix a bug in parsing JavaScript object names.
- #474: Fix building with SingleHTMLBuilder when there is no toctree.
- Fix display names for objects linked to by intersphinx with explicit targets.
- Fix building with the JSON builder.
- Fix hyperrefs in object descriptions for LaTeX.

10.106 Release 1.0 (Jul 23, 2010)

Incompatible changes

- Support for domains has been added. A domain is a collection of directives and roles that all describe objects belonging together, e.g. elements of a programming language. A few builtin domains are provided:
  - Python
  - C
  - C++
  - JavaScript
  - reStructuredText
- The old markup for defining and linking to C directives is now deprecated. It will not work anymore in future versions without activating the `oldcmarkup` extension; in Sphinx 1.0, it is activated by default.
- Removed support for old dependency versions; requirements are now:
  - docutils >= 0.5
  - Jinja2 >= 2.2
- Removed deprecated elements:
  - `exclude_dirs` config value
  - `sphinx.builder` module

Features added

- General:
  - Added a "nitpicky" mode that emits warnings for all missing references. It is activated by the *`sphinx-build -n`* command-line switch or the *`nitpicky`* config value.
  - Added `latexpdf` target in quickstart Makefile.
- Markup:
  - The *`menuselection`* and *`guilabel`* roles now support ampersand accelerators.
  - New more compact doc field syntax is now recognized: `:param type name:  description`.
  - Added `tab-width` option to *`literalinclude`* directive.
  - Added `titlesonly` option to *`toctree`* directive.
  - Added the `prepend` and `append` options to the *`literalinclude`* directive.
  - #284: All docinfo metadata is now put into the document metadata, not just the author.
  - The *`ref`* role can now also reference tables by caption.
  - The include[616] directive now supports absolute paths, which are interpreted as relative to the source directory.
  - In the Python domain, references like ``:func:`.name` `` now look for matching names with any prefix if no direct match is found.
- Configuration:
  - Added *`rst_prolog`* config value.

[616] http://docutils.sourceforge.net/docs/ref/rst/directives.html#include

  - Added *html_secnumber_suffix* config value to control section numbering format.
  - Added *html_compact_lists* config value to control docutils' compact lists feature.
  - The *html_sidebars* config value can now contain patterns as keys, and the values can be lists that explicitly select which sidebar templates should be rendered. That means that the builtin sidebar contents can be included only selectively.
  - *html_static_path* can now contain single file entries.
  - The new universal config value *exclude_patterns* makes the old `unused_docs`, `exclude_trees` and `exclude_dirnames` obsolete.
  - Added *html_output_encoding* config value.
  - Added the *latex_docclass* config value and made the "twoside" documentclass option overridable by "oneside".
  - Added the *trim_doctest_flags* config value, which is true by default.
  - Added *html_show_copyright* config value.
  - Added *latex_show_pagerefs* and *latex_show_urls* config values.
  - The behavior of *html_file_suffix* changed slightly: the empty string now means "no suffix" instead of "default suffix", use `None` for "default suffix".
- New builders:
  - Added a builder for the Epub format.
  - Added a builder for manual pages.
  - Added a single-file HTML builder.
- HTML output:
  - Inline roles now get a CSS class with their name, allowing styles to customize their appearance. Domain-specific roles get two classes, `domain` and `domain-rolename`.
  - References now get the class `internal` if they are internal to the whole project, as opposed to internal to the current page.
  - External references can be styled differently with the new `externalrefs` theme option for the default theme.
  - In the default theme, the sidebar can experimentally now be made collapsible using the new `collapsiblesidebar` theme option.
  - #129: Toctrees are now wrapped in a `div` tag with class `toctree-wrapper` in HTML output.
  - The *toctree* callable in templates now has a `maxdepth` keyword argument to control the depth of the generated tree.
  - The *toctree* callable in templates now accepts a `titles_only` keyword argument.
  - Added `htmltitle` block in layout template.
  - In the JavaScript search, allow searching for object names including the module name, like `sys.argv`.
  - Added new theme `haiku`, inspired by the Haiku OS user guide.
  - Added new theme `nature`.
  - Added new theme `agogo`, created by Andi Albrecht.
  - Added new theme `scrolls`, created by Armin Ronacher.
  - #193: Added a `visitedlinkcolor` theme option to the default theme.
  - #322: Improved responsiveness of the search page by loading the search index asynchronously.

- Extension API:
  - Added *html-collect-pages*.
  - Added *needs_sphinx* config value and *require_sphinx()* application API method.
  - #200: Added `add_stylesheet()` application API method.
- Extensions:
  - Added the *viewcode* extension.
  - Added the *extlinks* extension.
  - Added support for source ordering of members in autodoc, with `autodoc_member_order = 'bysource'`.
  - Added *autodoc_default_flags* config value, which can be used to select default flags for all autodoc directives.
  - Added a way for intersphinx to refer to named labels in other projects, and to specify the project you want to link to.
  - #280: Autodoc can now document instance attributes assigned in `__init__` methods.
  - Many improvements and fixes to the *autosummary* extension, thanks to Pauli Virtanen.
  - #309: The *graphviz* extension can now output SVG instead of PNG images, controlled by the *graphviz_output_format* config value.
  - Added `alt` option to *graphviz* extension directives.
  - Added `exclude` argument to *autodoc.between()*.
- Translations:
  - Added Croatian translation, thanks to Bojan Mihelač.
  - Added Turkish translation, thanks to Firat Ozgul.
  - Added Catalan translation, thanks to Pau Fernández.
  - Added simplified Chinese translation.
  - Added Danish translation, thanks to Hjorth Larsen.
  - Added Lithuanian translation, thanks to Dalius Dobravolskas.
- Bugs fixed:
  - #445: Fix links to result pages when using the search function of HTML built with the `dirhtml` builder.
  - #444: In templates, properly re-escape values treated with the "striptags" Jinja filter.

10.107 Previous versions

The changelog for versions before 1.0 can be found in the file `CHANGES.old` in the source distribution or at GitHub[617].

[617] https://github.com/sphinx-doc/sphinx/raw/master/CHANGES.old

CHAPTER

ELEVEN

PROJECTS USING SPHINX

This is an (incomplete) alphabetic list of projects that use Sphinx or are experimenting with using it for their documentation. If you like to be included, please mail to the Google group[618].

I've grouped the list into sections to make it easier to find interesting examples.

11.1 Documentation using the alabaster theme

- AIOHTTP[619]
- Alabaster[620]
- Blinker[621]
- Calibre[622]
- Click[623] (customized)
- coala[624] (customized)
- CodePy[625]
- Eve[626] (Python REST API framework)
- Fabric[627]
- Fityk[628]
- Flask[629]
- Flask-OpenID[630]
- Invoke[631]
- Jinja[632]

[618] https://groups.google.com/forum/#!forum/sphinx-users
[619] https://docs.aiohttp.org/
[620] https://alabaster.readthedocs.io/
[621] https://pythonhosted.org/blinker/
[622] https://manual.calibre-ebook.com/
[623] https://click.palletsprojects.com/
[624] https://docs.coala.io/
[625] https://documen.tician.de/codepy/
[626] https://docs.python-eve.org/
[627] https://docs.fabfile.org/
[628] https://fityk.nieto.pl/
[629] https://flask.palletsprojects.com/
[630] https://pythonhosted.org/Flask-OpenID/
[631] https://docs.pyinvoke.org/
[632] https://jinja.palletsprojects.com/

- Lino[633] (customized)
- marbl[634]
- MDAnalysis[635] (customized)
- MeshPy[636]
- Molecule[637]
- PyCUDA[638]
- PyOpenCL[639]
- PyLangAcq[640]
- pytest[641] (customized)
- python-apt[642]
- PyVisfile[643]
- Requests[644]
- searx[645]
- Spyder[646] (customized)
- Tablib[647]
- urllib3[648] (customized)
- Werkzeug[649]
- Write the Docs[650]

11.2 Documentation using the classic theme

- Advanced Generic Widgets[651] (customized)
- Apache CouchDB[652] (customized)
- APSW[653]
- Arb[654]

633 http://www.lino-framework.org/
634 https://getmarbl.readthedocs.io/
635 https://www.mdanalysis.org/docs/
636 https://documen.tician.de/meshpy/
637 https://molecule.readthedocs.io/
638 https://documen.tician.de/pycuda/
639 https://documen.tician.de/pyopencl/
640 http://pylangacq.org/
641 https://docs.pytest.org/
642 https://apt.alioth.debian.org/python-apt-doc/
643 https://documen.tician.de/pyvisfile/
644 http://www.python-requests.org/
645 https://asciimoo.github.io/searx/
646 https://docs.spyder-ide.org/
647 http://docs.python-tablib.org/
648 https://urllib3.readthedocs.io/
649 https://werkzeug.palletsprojects.com/
650 https://writethedocs-www.readthedocs.io/
651 http://xoomer.virgilio.it/infinity77/AGW_Docs/
652 http://docs.couchdb.org/
653 https://rogerbinns.github.io/apsw/
654 http://arblib.org/

- Bazaar[655] (customized)
- Beautiful Soup[656]
- Blender API[657]
- Bugzilla[658]
- Buildbot[659]
- CMake[660] (customized)
- Chaco[661] (customized)
- CORE[662]
- CORE Python modules[663]
- Cormoran[664]
- DEAP[665] (customized)
- Director[666]
- EZ-Draw[667] (customized)
- F2py[668]
- Generic Mapping Tools (GMT)[669] (customized)
- Genomedata[670]
- GetFEM++[671] (customized)
- Glasgow Haskell Compiler[672] (customized)
- Grok[673] (customized)
- GROMACS[674]
- GSL Shell[675]
- Hands-on Python Tutorial[676]
- Kaa[677] (customized)
- Leo[678]

[655] http://doc.bazaar.canonical.com/
[656] https://www.crummy.com/software/BeautifulSoup/bs4/doc/
[657] https://docs.blender.org/api/current/
[658] https://bugzilla.readthedocs.io/
[659] https://docs.buildbot.net/latest/
[660] https://cmake.org/documentation/
[661] https://docs.enthought.com/chaco/
[662] https://downloads.pf.itd.nrl.navy.mil/docs/core/core-html/
[663] https://downloads.pf.itd.nrl.navy.mil/docs/core/core-python-html/
[664] http://cormoran.nhopkg.org/docs/
[665] https://deap.readthedocs.io/
[666] https://pythonhosted.org/director/
[667] https://pageperso.lif.univ-mrs.fr/~edouard.thiel/ez-draw/doc/en/html/ez-manual.html
[668] http://f2py.sourceforge.net/docs/
[669] https://gmt.soest.hawaii.edu/doc/latest/
[670] https://noble.gs.washington.edu/proj/genomedata/doc/1.3.3/
[671] http://getfem.org/
[672] https://downloads.haskell.org/~ghc/latest/docs/html/users_guide/
[673] http://grok.zope.org/doc/current/
[674] http://manual.gromacs.org/documentation/
[675] https://www.nongnu.org/gsl-shell/
[676] https://anh.cs.luc.edu/python/hands-on/3.1/handsonHtml/
[677] https://api.freevo.org/kaa-base/
[678] https://leoeditor.com/

- LEPL[679] (customized)
- Mayavi[680] (customized)
- MediaGoblin[681] (customized)
- mpmath[682]
- OpenCV[683] (customized)
- OpenEXR[684]
- OpenGDA[685]
- Peach^3[686] (customized)
- Plone[687] (customized)
- PyEMD[688]
- Pyevolve[689]
- Pygame[690] (customized)
- PyMQI[691]
- PyQt4[692] (customized)
- PyQt5[693] (customized)
- Python 2[694]
- Python 3[695] (customized)
- Python Packaging Authority[696] (customized)
- Ring programming language[697] (customized)
- SageMath[698] (customized)
- Segway[699]
- simuPOP[700] (customized)
- Sprox[701] (customized)
- SymPy[702]

[679] http://www.acooke.org/lepl/
[680] https://docs.enthought.com/mayavi/mayavi/
[681] https://mediagoblin.readthedocs.io/
[682] http://mpmath.org/doc/current/
[683] https://docs.opencv.org/
[684] http://excamera.com/articles/26/doc/index.html
[685] http://www.opengda.org/gdadoc/html/
[686] https://peach3.nl/doc/latest/userdoc/
[687] https://docs.plone.org/
[688] https://pyemd.readthedocs.io/
[689] http://pyevolve.sourceforge.net/
[690] https://www.pygame.org/docs/
[691] https://pythonhosted.org/pymqi/
[692] http://pyqt.sourceforge.net/Docs/PyQt4/
[693] http://pyqt.sourceforge.net/Docs/PyQt5/
[694] https://docs.python.org/2/
[695] https://docs.python.org/3/
[696] https://www.pypa.io/
[697] http://ring-lang.sourceforge.net/doc/
[698] https://doc.sagemath.org/
[699] https://noble.gs.washington.edu/proj/segway/doc/1.1.0/segway.html
[700] http://simupop.sourceforge.net/manual_release/build/userGuide.html
[701] http://sprox.org/
[702] https://docs.sympy.org/

- TurboGears[703] (customized)
- tvtk[704]
- Varnish[705] (customized, alabaster for index)
- Waf[706]
- wxPython Phoenix[707] (customized)
- Yum[708]
- z3c[709]
- zc.async[710] (customized)
- Zope[711] (customized)

11.3 Documentation using the sphinxdoc theme

- ABRT[712]
- cartopy[713]
- Jython[714]
- LLVM[715]
- Matplotlib[716]
- MDAnalysis Tutorial[717]
- NetworkX[718]
- PyCantonese[719]
- Pyre[720]
- pySPACE[721]
- Pysparse[722]
- PyTango[723]
- Python Wild Magic[724] (customized)

[703] https://turbogears.readthedocs.io/
[704] https://docs.enthought.com/mayavi/tvtk/
[705] https://www.varnish-cache.org/docs/
[706] https://waf.io/apidocs/
[707] https://wxpython.org/Phoenix/docs/html/main.html
[708] http://yum.baseurl.org/api/yum/
[709] https://www.ibiblio.org/paulcarduner/z3ctutorial/
[710] https://pythonhosted.org/zc.async/
[711] https://docs.zope.org/zope2/
[712] https://abrt.readthedocs.io/
[713] https://scitools.org.uk/cartopy/docs/latest/
[714] http://www.jython.org/docs/
[715] https://llvm.org/docs/
[716] https://matplotlib.org/
[717] https://www.mdanalysis.org/MDAnalysisTutorial/
[718] https://networkx.github.io/
[719] http://pycantonese.org/
[720] http://docs.danse.us/pyre/sphinx/
[721] https://pyspace.github.io/pyspace/
[722] http://pysparse.sourceforge.net/
[723] https://www.esrf.eu/computing/cs/tango/tango_doc/kernel_doc/pytango/latest/
[724] https://vmlaker.github.io/pythonwildmagic/

- RDKit[725]
- Reteisi[726] (customized)
- Sqlkit[727] (customized)
- Turbulenz[728]

11.4 Documentation using the nature theme

- Alembic[729]
- Cython[730]
- easybuild[731]
- jsFiddle[732]
- libLAS[733] (customized)
- Lmod[734]
- MapServer[735] (customized)
- Pandas[736]
- pyglet[737] (customized)
- PyWavelets[738]
- Setuptools[739]
- Spring Python[740]
- StatsModels[741] (customized)
- Sylli[742]

[725] https://www.rdkit.org/docs/
[726] http://www.reteisi.org/contents.html
[727] http://sqlkit.argolinux.org/
[728] http://docs.turbulenz.com/
[729] https://alembic.sqlalchemy.org/
[730] http://docs.cython.org/
[731] https://easybuild.readthedocs.io/
[732] http://doc.jsfiddle.net/
[733] https://www.liblas.org/
[734] https://lmod.readthedocs.io/
[735] https://mapserver.org/
[736] https://pandas.pydata.org/pandas-docs/stable/
[737] https://pyglet.readthedocs.io/
[738] https://pywavelets.readthedocs.io/
[739] https://setuptools.readthedocs.io/
[740] https://docs.spring.io/spring-python/1.2.x/sphinx/html/
[741] https://www.statsmodels.org/
[742] http://sylli.sourceforge.net/

11.5 Documentation using another builtin theme

- Breathe[743] (haiku)
- MPipe[744] (sphinx13)
- NLTK[745] (agogo)
- Programmieren mit PyGTK und Glade (German)[746] (agogo, customized)
- PyPubSub[747] (bizstyle)
- Pylons[748] (pyramid)
- Pyramid web framework[749] (pyramid)
- RxDock[750]
- Sphinx[751] (sphinx13) :-)
- Valence[752] (haiku, customized)

11.6 Documentation using sphinx_rtd_theme

- Annotator[753]
- Ansible[754] (customized)
- Arcade[755]
- aria2[756]
- ASE[757]
- Autofac[758]
- BigchainDB[759]
- Blender Reference Manual[760]
- Blocks[761]
- bootstrap-datepicker[762]
- Certbot[763]

[743] https://breathe.readthedocs.io/
[744] https://vmlaker.github.io/mpipe/
[745] https://www.nltk.org/
[746] https://www.florian-diesch.de/doc/python-und-glade/online/
[747] https://pypubsub.readthedocs.io/
[748] https://docs.pylonsproject.org/projects/pylons-webframework/
[749] https://docs.pylonsproject.org/projects/pyramid/
[750] https://www.rxdock.org/documentation/html/devel/
[751] http://www.sphinx-doc.org/
[752] https://docs.valence.desire2learn.com/
[753] http://docs.annotatorjs.org/
[754] https://docs.ansible.com/
[755] http://arcade.academy/
[756] https://aria2.github.io/manual/en/html/
[757] https://wiki.fysik.dtu.dk/ase/
[758] http://docs.autofac.org/
[759] https://docs.bigchaindb.com/
[760] https://docs.blender.org/manual/
[761] https://blocks.readthedocs.io/
[762] https://bootstrap-datepicker.readthedocs.io/
[763] https://letsencrypt.readthedocs.io/

- Chainer[764] (customized)
- CherryPy[765]
- cloud-init[766]
- CodeIgniter[767]
- Conda[768]
- Corda[769]
- Dask[770]
- Databricks[771] (customized)
- Dataiku DSS[772]
- DNF[773]
- Django-cas-ng[774]
- edX[775]
- Electrum[776]
- Elemental[777]
- ESWP3[778]
- Ethereum Homestead[779]
- Exhale[780]
- Faker[781]
- Fidimag[782]
- Flake8[783]
- Flatpak[784]
- FluidDyn[785]
- Fluidsim[786]
- Gallium[787]

[764] https://docs.chainer.org/
[765] https://docs.cherrypy.org/
[766] https://cloudinit.readthedocs.io/
[767] https://www.codeigniter.com/user_guide/
[768] https://conda.io/docs/
[769] https://docs.corda.net/
[770] https://dask.pydata.org/
[771] https://docs.databricks.com/
[772] https://doc.dataiku.com/
[773] https://dnf.readthedocs.io/
[774] https://djangocas.dev/docs/
[775] https://docs.edx.org/
[776] http://docs.electrum.org/
[777] http://libelemental.org/documentation/dev/
[778] https://eswp3.readthedocs.io/
[779] http://www.ethdocs.org/
[780] https://exhale.readthedocs.io/
[781] https://faker.readthedocs.io/
[782] https://fidimag.readthedocs.io/
[783] http://flake8.pycqa.org/
[784] http://docs.flatpak.org/
[785] https://fluiddyn.readthedocs.io/
[786] https://fluidsim.readthedocs.io/
[787] https://gallium.readthedocs.io/

- GeoNode[788]
- Glances[789]
- Godot[790]
- Graylog[791]
- GPAW[792] (customized)
- HDF5 for Python (h5py)[793]
- Hyperledger Fabric[794]
- Hyperledger Sawtooth[795]
- IdentityServer[796]
- Idris[797]
- javasphinx[798]
- Julia[799]
- Jupyter Notebook[800]
- Lasagne[801]
- latexindent.pl[802]
- Learning Apache Spark with Python[803]
- LibCEED[804]
- Linguistica[805]
- Linux kernel[806]
- Mailman[807]
- MathJax[808]
- MDTraj[809] (customized)
- Mesa 3D[810]
- micca - MICrobial Community Analysis[811]

[788] http://docs.geonode.org/
[789] https://glances.readthedocs.io/
[790] https://godot.readthedocs.io/
[791] http://docs.graylog.org/
[792] https://wiki.fysik.dtu.dk/gpaw/
[793] http://docs.h5py.org/
[794] https://hyperledger-fabric.readthedocs.io/
[795] https://intelledger.github.io/
[796] http://docs.identityserver.io/
[797] http://docs.idris-lang.org/
[798] https://bronto-javasphinx.readthedocs.io/
[799] https://julia.readthedocs.io/
[800] https://jupyter-notebook.readthedocs.io/
[801] https://lasagne.readthedocs.io/
[802] https://latexindentpl.readthedocs.io/
[803] https://runawayhorse001.github.io/LearningApacheSpark
[804] https://libceed.readthedocs.io/
[805] https://linguistica-uchicago.github.io/lxa5/
[806] https://www.kernel.org/doc/html/latest/index.html
[807] http://docs.list.org/
[808] https://docs.mathjax.org/
[809] http://mdtraj.org/latest/
[810] https://docs.mesa3d.org/
[811] https://micca.readthedocs.io/

- MicroPython[812]
- Minds[813] (customized)
- Mink[814]
- Mockery[815]
- mod_wsgi[816]
- MoinMoin[817]
- Mopidy[818]
- mpi4py[819]
- MyHDL[820]
- Nextflow[821]
- NICOS[822] (customized)
- OpenFAST[823]
- Pelican[824]
- picamera[825]
- Pillow[826]
- pip[827]
- Paver[828]
- peewee[829]
- Phinx[830]
- phpMyAdmin[831]
- PROS[832] (customized)
- Pushkin[833]
- Pweave[834]
- PyPy[835]

[812] https://docs.micropython.org/
[813] https://www.minds.org/docs/
[814] http://mink.behat.org/
[815] http://docs.mockery.io/
[816] https://modwsgi.readthedocs.io/
[817] https://moin-20.readthedocs.io/
[818] https://docs.mopidy.com/
[819] https://mpi4py.readthedocs.io/
[820] http://docs.myhdl.org/
[821] https://www.nextflow.io/docs/latest/index.html
[822] https://forge.frm2.tum.de/nicos/doc/nicos-master/
[823] https://openfast.readthedocs.io/
[824] http://docs.getpelican.com/
[825] https://picamera.readthedocs.io/
[826] https://pillow.readthedocs.io/
[827] https://pip.pypa.io/
[828] https://paver.readthedocs.io/
[829] http://docs.peewee-orm.com/
[830] http://docs.phinx.org/
[831] https://docs.phpmyadmin.net/
[832] https://pros.cs.purdue.edu/v5/
[833] http://docs.pushkin.io/
[834] http://mpastell.com/pweave/
[835] http://doc.pypy.org/

- python-sqlparse[836]
- PyVISA[837]
- pyvista[838]
- Read The Docs[839]
- ROCm Platform[840]
- Free your information from their silos (French)[841] (customized)
- Releases Sphinx extension[842]
- Qtile[843]
- Quex[844]
- QuTiP[845]
- Satchmo[846]
- Scapy[847]
- SimGrid[848]
- SimPy[849]
- six[850]
- SlamData[851]
- Solidity[852]
- Sonos Controller (SoCo)[853]
- Sphinx AutoAPI[854]
- sphinx-argparse[855]
- Sphinx-Gallery[856] (customized)
- Sphinx with Github Webpages[857]
- SpotBugs[858]
- StarUML[859]

[836] https://sqlparse.readthedocs.io/
[837] https://pyvisa.readthedocs.io/
[838] https://docs.pyvista.org/
[839] https://docs.readthedocs.io/
[840] https://rocm-documentation.readthedocs.io/
[841] http://redaction-technique.org/
[842] https://releases.readthedocs.io/
[843] http://docs.qtile.org/
[844] http://quex.sourceforge.net/doc/html/main.html
[845] http://qutip.org/docs/latest/
[846] http://docs.satchmoproject.com/
[847] https://scapy.readthedocs.io/
[848] https://simgrid.org/doc/latest/
[849] https://simpy.readthedocs.io/
[850] https://six.readthedocs.io/
[851] https://newdocs.slamdata.com
[852] https://solidity.readthedocs.io/
[853] http://docs.python-soco.com/
[854] https://sphinx-autoapi.readthedocs.io/
[855] https://sphinx-argparse.readthedocs.io/
[856] https://sphinx-gallery.readthedocs.io/
[857] https://runawayhorse001.github.io/SphinxGithub
[858] https://spotbugs.readthedocs.io/
[859] https://docs.staruml.io/

- Sublime Text Unofficial Documentation[860]
- SunPy[861]
- Sylius[862]
- Syncthing[863]
- Tango Controls[864] (customized)
- Topshelf[865]
- Theano[866]
- ThreatConnect[867]
- Tuleap[868]
- TYPO3[869] (customized)
- Veyon[870]
- uWSGI[871]
- virtualenv[872]
- Wagtail[873]
- Web Application Attack and Audit Framework (w3af)[874]
- Weblate[875]
- x265[876]
- Zulip[877]

11.7 Documentation using sphinx_bootstrap_theme

- Bootstrap Theme[878]
- C/C++ Software Development with Eclipse[879]
- Dataverse[880]
- e-cidadania[881]

[860] http://docs.sublimetext.info/
[861] https://docs.sunpy.org/
[862] http://docs.sylius.org/
[863] https://docs.syncthing.net/
[864] https://tango-controls.readthedocs.io/
[865] http://docs.topshelf-project.com/
[866] http://www.deeplearning.net/software/theano/
[867] https://docs.threatconnect.com/
[868] https://tuleap.net/doc/en/
[869] https://docs.typo3.org/
[870] https://docs.veyon.io/
[871] https://uwsgi-docs.readthedocs.io/
[872] https://virtualenv.readthedocs.io/
[873] https://docs.wagtail.io/
[874] http://docs.w3af.org/
[875] https://docs.weblate.org/
[876] https://x265.readthedocs.io/
[877] https://zulip.readthedocs.io/
[878] https://ryan-roemer.github.io/sphinx-bootstrap-theme/
[879] https://eclipsebook.in/
[880] http://guides.dataverse.org/
[881] https://e-cidadania.readthedocs.io/

- Hangfire[882]
- Hedge[883]
- ObsPy[884]
- Open Dylan[885]
- OPNFV[886]
- Pootle[887]
- PyUblas[888]
- seaborn[889]

11.8 Documentation using a custom theme or integrated in a website

- Apache Cassandra[890]
- Astropy[891]
- Bokeh[892]
- Boto 3[893]
- CakePHP[894]
- CasperJS[895]
- Ceph[896]
- Chef[897]
- CKAN[898]
- Confluent Platform[899]
- Django[900]
- Doctrine[901]
- Enterprise Toolkit for Acrobat products[902]
- FreeFEM[903]

[882] http://docs.hangfire.io/
[883] https://documen.tician.de/hedge/
[884] https://docs.obspy.org/
[885] https://opendylan.org/documentation/
[886] https://docs.opnfv.org/
[887] http://docs.translatehouse.org/projects/pootle/
[888] https://documen.tician.de/pyublas/
[889] https://seaborn.pydata.org/
[890] https://cassandra.apache.org/doc/
[891] http://docs.astropy.org/
[892] https://bokeh.pydata.org/
[893] https://boto3.readthedocs.io/
[894] https://book.cakephp.org/
[895] http://docs.casperjs.org/
[896] http://docs.ceph.com/docs/master/
[897] https://docs.chef.io/
[898] https://docs.ckan.org/
[899] https://docs.confluent.io/
[900] https://docs.djangoproject.com/
[901] https://www.doctrine-project.org/
[902] https://www.adobe.com/devnet-docs/acrobatetk/
[903] https://doc.freefem.org/introduction/

- fmt[904]
- Gameduino[905]
- gensim[906]
- GeoServer[907]
- gevent[908]
- GHC - Glasgow Haskell Compiler[909]
- Guzzle[910]
- H2O.ai[911]
- Heka[912]
- Istihza (Turkish Python documentation project)[913]
- JupyterHub[914]
- Kombu[915]
- Lasso[916]
- Mako[917]
- MirrorBrain[918]
- Mitiq[919]
- MongoDB[920]
- Music21[921]
- MyHDL[922]
- ndnSIM[923]
- nose[924]
- ns-3[925]
- NumPy[926]
- ObjectListView[927]

[904] https://fmt.dev/
[905] http://excamera.com/sphinx/gameduino/
[906] https://radimrehurek.com/gensim/
[907] http://docs.geoserver.org/
[908] http://www.gevent.org/
[909] https://downloads.haskell.org/~ghc/master/users-guide/
[910] http://docs.guzzlephp.org/
[911] http://docs.h2o.ai/
[912] https://hekad.readthedocs.io/
[913] https://belgeler.yazbel.com/python-istihza/
[914] https://jupyterhub.readthedocs.io/
[915] http://docs.kombu.me/
[916] http://lassoguide.com/
[917] http://docs.makotemplates.org/
[918] http://mirrorbrain.org/docs/
[919] https://mitiq.readthedocs.io/
[920] https://docs.mongodb.com/
[921] https://web.mit.edu/music21/doc/
[922] http://docs.myhdl.org/
[923] https://ndnsim.net/current/
[924] https://nose.readthedocs.io/
[925] https://www.nsnam.org/documentation/
[926] https://docs.scipy.org/doc/numpy/reference/
[927] http://objectlistview.sourceforge.net/python/

- OpenERP[928]
- OpenCV[929]
- OpenLayers[930]
- OpenTURNS[931]
- Open vSwitch[932]
- PlatformIO[933]
- PyEphem[934]
- Pygments[935]
- Plone User Manual (German)[936]
- PSI4[937]
- PyMOTW[938]
- python-aspectlib[939] (sphinx_py3doc_enhanced_theme[940])
- QGIS[941]
- qooxdoo[942]
- Roundup[943]
- SaltStack[944]
- scikit-learn[945]
- SciPy[946]
- Scrapy[947]
- Seaborn[948]
- Selenium[949]
- Self[950]
- Substance D[951]

[928] https://doc.odoo.com/
[929] https://docs.opencv.org/
[930] http://docs.openlayers.org/
[931] https://openturns.github.io/openturns/master/
[932] http://docs.openvswitch.org/
[933] https://docs.platformio.org/
[934] http://rhodesmill.org/pyephem/
[935] http://pygments.org/docs/
[936] https://www.hasecke.com/plone-benutzerhandbuch/4.0/
[937] http://www.psicode.org/psi4manual/master/index.html
[938] https://pymotw.com/2/
[939] https://python-aspectlib.readthedocs.io/
[940] https://pypi.org/project/sphinx_py3doc_enhanced_theme/
[941] https://qgis.org/en/docs/index.html
[942] https://www.qooxdoo.org/current/
[943] http://www.roundup-tracker.org/
[944] https://docs.saltstack.com/
[945] http://scikit-learn.org/stable/
[946] https://docs.scipy.org/doc/scipy/reference/
[947] https://doc.scrapy.org/
[948] https://seaborn.pydata.org/
[949] https://docs.seleniumhq.org/docs/
[950] http://www.selflanguage.org/
[951] https://docs.pylonsproject.org/projects/substanced/

- Sulu[952]
- SQLAlchemy[953]
- tinyTiM[954]
- Twisted[955]
- Ubuntu Packaging Guide[956]
- WebFaction[957]
- WTForms[958]

11.9 Homepages and other non-documentation sites

- Arizona State University PHY494/PHY598/CHM598 Simulation approaches to Bio-and Nanophysics[959] (classic)
- Benoit Boissinot[960] (classic, customized)
- Computer Networks, Parallelization, and Simulation Laboratory (CNPSLab)[961] (sphinx_rtd_theme)
- Deep Learning Tutorials[962] (sphinxdoc)
- Eric Holscher[963] (alabaster)
- Lei Ma's Statistical Mechanics lecture notes[964] (sphinx_bootstrap_theme)
- Loyola University Chicago COMP 339-439 Distributed Systems course[965] (sphinx_bootstrap_theme)
- Pylearn2[966] (sphinxdoc, customized)
- PyXLL[967] (sphinx_bootstrap_theme, customized)
- SciPy Cookbook[968] (sphinx_rtd_theme)
- Tech writer at work blog[969] (custom theme)
- The Wine Cellar Book[970] (sphinxdoc)
- Thomas Cokelaer's Python, Sphinx and reStructuredText tutorials[971] (standard)
- UC Berkeley ME233 Advanced Control Systems II course[972] (sphinxdoc)
- Željko Svedružić's Biomolecular Structure and Function Laboratory (BioSFLab)[973] (sphinx_bootstrap_theme)

[952] http://docs.sulu.io/
[953] https://docs.sqlalchemy.org/
[954] http://tinytim.sourceforge.net/docs/2.0/
[955] https://twistedmatrix.com/documents/current/
[956] http://packaging.ubuntu.com/html/
[957] https://docs.webfaction.com/
[958] https://wtforms.readthedocs.io/
[959] https://becksteinlab.physics.asu.edu/pages/courses/2013/SimBioNano/
[960] https://bboissin.appspot.com/
[961] https://lab.miletic.net/
[962] http://www.deeplearning.net/tutorial/
[963] http://ericholscher.com/
[964] http://statisticalphysics.openmetric.org/
[965] https://books.cs.luc.edu/distributedsystems/
[966] http://www.deeplearning.net/software/pylearn2/
[967] https://www.pyxll.com/
[968] https://scipy-cookbook.readthedocs.io/
[969] https://blog.documatt.com/
[970] https://www.thewinecellarbook.com/doc/en/
[971] https://thomas-cokelaer.info/tutorials/
[972] https://berkeley-me233.github.io/
[973] https://www.svedruziclab.com/

11.10 Books produced using Sphinx

- "The Art of Community" (Japanese translation)[974]
- "Die Wahrheit des Sehens. Der DEKALOG von Krzysztof Kieślowski"[975]
- "Expert Python Programming"[976]
- "Expert Python Programming" (Japanese translation)[977]
- "Expert Python Programming 2nd Edition" (Japanese translation)[978]
- "The Hitchhiker's Guide to Python"[979]
- "LassoGuide"[980]
- "Learning Sphinx" (in Japanese)[981]
- "Learning System Programming with Go (Japanese)"[982]
- "Mercurial: the definitive guide (Second edition)"[983]
- "Mithril – The fastest clientside MVC (Japanese)"[984]
- "Pioneers and Prominent Men of Utah"[985]
- "Pomodoro Technique Illustrated" (Japanese translation)[986]
- "Professional Software Development"[987]
- "Python Professional Programming" (in Japanese)[988]
- "Python Professional Programming 2nd Edition" (in Japanese)[989]
- "Python Professional Programming 3rd Edition" (in Japanese)[990]
- Python Course by Yuri Petrov (Russian)[991]
- "Real World HTTP – Learning The Internet and Web Technology via its history and code (Japanese)"[992]
- "Redmine Primer 5th Edition (in Japanese)"[993]
- "The repoze.bfg Web Application Framework"[994]
- "The Self-Taught Programmer" (Japanese translation)[995]
- "Simple and Steady Way of Learning for Software Engineering" (in Japanese)[996]

[974] https://www.oreilly.co.jp/books/9784873114958/
[975] https://literatur.hasecke.com/post/die-wahrheit-des-sehens-dekalog-kieslowski/
[976] https://www.packtpub.com/application-development/expert-python-programming
[977] https://www.amazon.co.jp/dp/4048686291/
[978] https://www.amazon.co.jp/dp/4048930613/
[979] https://docs.python-guide.org/
[980] http://www.lassosoft.com/Lasso-Documentation
[981] https://www.oreilly.co.jp/books/9784873116488/
[982] https://www.lambdanote.com/products/go
[983] https://book.mercurial-scm.org/
[984] https://www.oreilly.co.jp/books/9784873117447/
[985] http://pioneers.rstebbing.com/
[986] https://www.amazon.co.jp/dp/4048689525/
[987] https://mixmastamyk.bitbucket.io/pro_soft_dev/
[988] http://www.amazon.co.jp/dp/4798032948/
[989] https://www.amazon.co.jp/dp/479804315X/
[990] https://www.amazon.co.jp/dp/4798053821/
[991] https://www.yuripetrov.ru/edu/python
[992] https://www.oreilly.co.jp/books/9784873118048/
[993] https://www.shuwasystem.co.jp/products/7980html/4825.html
[994] https://www.amazon.com/repoze-bfg-Web-Application-Framework-Version/dp/0615345379
[995] https://www.amazon.co.jp/dp/4822292274/
[996] https://www.amazon.co.jp/dp/477414259X/

- "Software-Dokumentation mit Sphinx"[997]
- "Theoretical Physics Reference"[998]
- "The Varnish Book"[999]

11.11 Theses produced using Sphinx

- "A Web-Based System for Comparative Analysis of OpenStreetMap Data by the Use of CouchDB"[1000]
- "Content Conditioning and Distribution for Dynamic Virtual Worlds"[1001]
- "The Sphinx Thesis Resource"[1002]

11.12 Projects integrating Sphinx functionality

- Read the Docs[1003], a software-as-a-service documentation hosting platform, uses Sphinx to automatically build documentation updates that are pushed to GitHub.
- Spyder[1004], the Scientific Python Development Environment, uses Sphinx in its help pane to render rich documentation for functions, classes and methods automatically or on-demand.

[997] https://www.amazon.de/dp/1497448689/
[998] https://www.theoretical-physics.net/
[999] https://info.varnish-software.com/the-varnish-book
[1000] https://www.yumpu.com/et/document/view/11722645/masterthesis-markusmayr-0542042
[1001] https://www.cs.princeton.edu/research/techreps/TR-941-12
[1002] https://jterrace.github.io/sphinxtr/
[1003] https://readthedocs.org/
[1004] https://docs.spyder-ide.org/help.html

PYTHON MODULE INDEX

INDEX