

Sphinx Documentation

Release 8.1.2

the Sphinx developers

THE BASICS

1	Insta	lling Sphinx	1
	1.1	PyPI package	1
	1.2	Conda package	2
	1.3	OS-specific package manager	2
		Linux	2
		Debian/Ubuntu	2
		RHEL, CentOS	3
		Other distributions	3
		macOS	3
		Homebrew	3
		MacPorts	3
		Windows	3
		Chocolatey	
	1.4	Docker	
	1.5	Installation of the latest development release	
	1.6	Installation from source	
2	Getti	ng started	7
	2.1	Setting up the documentation sources	7
	2.2	Defining document structure	7
	2.3	Adding content	8
	2.4	Running the build	
	2.5	Documenting objects	9
	2.6	Basic configuration	O
	2.7	Autodoc	O
	2.8	Intersphinx	1
	2.9	More topics to be covered	1
		1	
3	Build	your first project	3
	3.1	Getting started	3
		Setting up your project and development environment	3
		Creating the documentation layout	4
	3.2	First steps to document your project using Sphinx	5
		Building your HTML documentation	5
		Building your documentation in other formats	6
	3.3	More Sphinx customization	6
		Enabling a built-in extension	6
		Using a third-party HTML theme	7
	3.4	Narrative documentation in Sphinx	7
		Structuring your documentation across multiple pages	

		Adding cross-references	9
	3.5	Describing code in Sphinx	0
		Python	0
		Documenting Python objects	0
		Cross-referencing Python objects	
		Including doctests in your documentation	
		Other languages (C, C++, others)	
		Documenting and cross-referencing objects	
	3.6	Automatic documentation generation from code	
		Reusing signatures and docstrings with autodoc	
		Generating comprehensive API references	6
	3.7	Appendix: Deploying a Sphinx project online	7
		Sphinx-friendly deployment options	8
		Embracing the "Docs as Code" philosophy	
		Publishing your documentation sources	
		GitLab	
		Publishing your HTML documentation	
		Read the Docs	9
		GitHub Pages	0
		GitLab Pages	1
	3.8	Where to go from here	
	5.0	There to go from here	_
4	Using	g Sphinx 3.	3
	4.1	reStructuredText	
	1.1	reStructuredText Primer	
		Paragraphs	
		Inline markup	
		Lists and Quote-like blocks	
		Literal blocks	
		Doctest blocks	5
		Tables	6
		Hyperlinks	6
		Sections	
		Field Lists	
		Explicit Markup	
		Directives	
		Images	0
		Footnotes	1
		Citations	1
		Substitutions	2
		Comments	2
		HTML Metadata	
		Source encoding	
		Gotchas	
		Roles	
		Cross-references	
		Inline code highlighting	4
		Math	4
		Other semantic markup	4
		Substitutions	
		Directives	
		Table of contents	o

	Paragraph-level markup	51
	Showing code examples	56
	Glossary	63
	Meta-information markup	64
	Index-generating markup	64
	Including content based on tags	66
	Tables	67
	Math	68
	Grammar production displays	69
	Field Lists	70
	File-wide metadata	70
	Special metadata fields	71
	MOVED: Domains	71
	MOVED: Basic Markup	71
	MOVED: Python Domain	71
	MOVED: C Domain	72
	MOVED: C++ Domain	72
	MOVED: Standard Domain	72
	MOVED: JavaScript Domain	72
	MOVED: reStructuredText Domain	72
	MOVED: Math Domain	72
	MOVED: More domains	72
4.2	Markdown	72
+. ∠	Configuration	72
4.3		73
4.3	Cross-references	
	Syntax	73
	Cross-referencing objects	73
	Cross-referencing arbitrary locations	74
	Cross-referencing documents	74
	Referencing downloadable files	75
	Cross-referencing figures by figure number	75
	Cross-referencing other items of interest	75
	Cross-referencing anything	76
4.4	Configuration	77
	Project tags	77
	Project information	77
	General configuration	79
	Options for figure numbering	81
	Options for highlighting	82
	Options for HTTP requests	83
	Options for internationalisation	84
	Options for markup	89
	Options for Maths	91
	Options for the nitpicky mode	92
	Options for object signatures	93
	Options for source files	94
	Options for smart quotes	96
	Options for templating	98
	Options for warning control	98
		100
		100
		113
		114
		114

	Options for EPUB output	
	Options for LaTeX output	
	Options for text output	
	Options for manual page output	
	Options for Texinfo output	
	Options for QtHelp output	
	Options for XML output	
	Options for the linkcheck builder	
	Domain options	
	Options for the C domain	
	Options for the C++ domain	143
	Options for the Javascript domain	144
	Options for the Python domain	144
	Extension options	146
	Example configuration file	146
4.5	Builders	147
	Serialization builder details	
4.6	Domains	
	Basic Markup	
	Cross-referencing syntax	
	Built-in domains	
	The Standard Domain	
	The C Domain	
	The C++ Domain	
	The JavaScript Domain	
	The Mathematics Domain	
	The Python Domain	
	The reStructuredText Domain	
	More domains	
4.7	Extensions	
1.,	Built-in extensions	
	sphinx.ext.autodoc – Include documentation from docstrings	
	sphinx.ext.autosectionlabel – Allow referencing sections by their title	
	sphinx.ext.autosummary – Generate autodoc summaries	
	sphinx.ext.coverage - Collect doc coverage stats	
	sphinx.ext.doctest – Test snippets in the documentation	
	sphinx.ext.duration – Measure durations of Sphinx processing	
	sphinx.ext.extlinks – Markup to shorten external links	
	sphinx.ext.githubpages — Publish HTML docs in GitHub Pages	
	sphinx.ext.graphviz – Add Graphviz graphs	
	sphinx.ext.ifconfig – Include content based on configuration	
	sphinx.ext.imgconverter – A reference image converter using Imagemagick	
	sphinx.ext.inheritance_diagram – Include inheritance diagrams	
	sphinx.ext.intersphinx – Link to other projects' documentation	
	sphinx.ext.linkcode – Add external links to source code	
	Math support for HTML outputs in Sphinx	
	sphinx.ext.napoleon – Support for NumPy and Google style docstrings	
	sphinx.ext.tado – Support for todo items	
	sphinx.ext.viewcode – Add links to highlighted source code	
	Third-party extensions	
	Where to put your own extensions?	
4.8	HTML theming	
+.0	Builders	
	Themes	233

		Using a theme
		Builtin themes
		Third Party Themes
	4.9	Internationalization
		Sphinx internationalization details
		Translating with sphinx-intl
		Quick guide
		Translating
		· · · · · · · · · · · · · · · · · · ·
		Update your po files by new pot files
		Using Transifex service for team translation
		Using Weblate service for team translation
		Contributing to Sphinx reference translation
		Translation progress and statistics
	4.10	Sphinx Web Support
		Web Support Quick Start
		Building documentation data
		Integrating Sphinx documents into your web-app
		Performing searches
		Comments & proposals
		Comment moderation
		The WebSupport class
		Methods
		Search adapters
		Methods
		Storage backends
		Methods
_	_	
5		nding Sphinx 279
	5.1	Tutorials
		Extending syntax with roles and directives
		Overview
		Setting up the project
		Writing the extension
		Using the extension
		Further reading
		Further reading 283 Extending the build process 283 Overview 283 Prerequisites 284 Writing the extension 284 Using the extension 292 Further reading 293
		Further reading 283 Extending the build process 283 Overview 283 Prerequisites 284 Writing the extension 284 Using the extension 295 Further reading 293 Adding a reference domain 293
		Further reading 283 Extending the build process 283 Overview 283 Prerequisites 284 Writing the extension 284 Using the extension 293 Further reading 293 Adding a reference domain 293 Overview 293
		Further reading 283 Extending the build process 283 Overview 283 Prerequisites 284 Writing the extension 284 Using the extension 293 Further reading 293 Adding a reference domain 293 Overview 293 Prerequisites 293
		Further reading 283 Extending the build process 283 Overview 283 Prerequisites 284 Writing the extension 284 Using the extension 293 Further reading 293 Adding a reference domain 293 Overview 293 Prerequisites 293 Writing the extension 293 Writing the extension 294
		Further reading 283 Extending the build process 283 Overview 283 Prerequisites 284 Writing the extension 284 Using the extension 293 Further reading 293 Adding a reference domain 293 Overview 293 Prerequisites 293 Writing the extension 294 Using the extension 303
		Further reading 283 Extending the build process 283 Overview 283 Prerequisites 284 Writing the extension 284 Using the extension 293 Further reading 293 Adding a reference domain 293 Overview 293 Prerequisites 293 Writing the extension 293 Writing the extension 294
		Further reading 283 Extending the build process 283 Overview 283 Prerequisites 284 Writing the extension 284 Using the extension 293 Further reading 293 Adding a reference domain 293 Overview 293 Prerequisites 293 Writing the extension 294 Using the extension 303
		Further reading 283 Extending the build process 283 Overview 283 Prerequisites 284 Writing the extension 284 Using the extension 293 Further reading 293 Adding a reference domain 293 Overview 293 Prerequisites 293 Writing the extension 294 Using the extension 303 Further reading 303
		Further reading 283 Extending the build process 283 Overview 283 Prerequisites 284 Writing the extension 284 Using the extension 293 Further reading 293 Adding a reference domain 293 Overview 293 Prerequisites 293 Writing the extension 293 Writing the extension 293 Using the extension 293 Writing the extension 293 Using the extension 293 Using the extension 303 Further reading 303 Developing autodoc extensions 303
		Further reading 28 Extending the build process 28 Overview 28 Prerequisites 284 Writing the extension 29 Further reading 29 Adding a reference domain 29 Overview 29 Prerequisites 29 Writing the extension 29 Using the extension 30 Further reading 30 Developing autodoc extensions 30 Overview 30 Overview 30
		Further reading 28 Extending the build process 28 Overview 28 Prerequisites 28 Writing the extension 28 Using the extension 29 Further reading 29 Adding a reference domain 29 Overview 29 Prerequisites 29 Writing the extension 30 Further reading 30 Developing autodoc extensions 30 Overview 30 Prerequisites 30 Writing the extension 30
		Further reading 28 Extending the build process 28 Overview 28 Prerequisites 28 Writing the extension 29 Further reading 29 Adding a reference domain 29 Overview 29 Prerequisites 29 Writing the extension 29 Using the extension 30 Further reading 30 Further reading 30 Overview 30 Overview 30 Prerequisites 30 Writing the extension 30 Writing the extension 30 Using the extension 30
	5.2	Further reading 28 Extending the build process 28 Overview 28 Prerequisites 28 Writing the extension 29 Further reading 29 Adding a reference domain 29 Overview 29 Prerequisites 29 Writing the extension 29 Using the extension 30 Further reading 30 Further reading 30 Overview 30 Overview 30 Prerequisites 30 Writing the extension 30 Writing the extension 30 Using the extension 30

		Depend on another extension	305
	5.3	Discover builders by entry point	
	5.5	HTML theme development	
		Theme configuration (theme.toml)	
		Theme configuration (theme.conf)	
		Convert theme.conf to theme.toml	
		Distribute your theme as a Python package	
		Styling with CSS	
		Styling search result entries by category	
		Templating	
		Templating	
		Static templates	
		Use custom page metadata in HTML templates	
		Defining custom template functions	
		Add your own static files to the build assets	
		Inject JavaScript based on user configuration	
		inject Javascript based on user configuration	310
6	Sphir	nx API	321
	6.1	Important objects	
	6.2	Build phases	
	6.3	Extension metadata	
	6.4	APIs used for writing extensions	
		Application API	
		Extension setup	
		Emitting events	
		Sphinx runtime information	
		Sphinx core events	
		Checking the Sphinx version	341
		The Config object	342
		The template bridge	342
		Exceptions	343
		Event callbacks API	343
		Core events overview	344
		Core event details	345
		Builder specific events	350
		Project API	351
		Build environment API	352
		Builder API	353
		Environment Collector API	357
		Docutils markup API	359
		Roles	359
		Directives	359
		Domain API	362
		Python Domain	369
		Parser API	370
		Doctree node classes added by Sphinx	371
		Nodes for domain-specific object descriptions	371
			374
		Other paragraph-level nodes	374
			374
		Special nodes	
		Logging API	375

		i18n API	77
		Extension internationalization (i18n) and localization (l10n) using i18n API	78
		Utilities	79
		Base classes for components	
		Utility components	
		Utility functions	
		Utility types	
		Testing API	
		pytest configuration	
		Usage	
		Deprecated APIs	39
7	т «Т»	V anatomination	0.4
/	7.1	X customization The latex_elements configuration setting	
	7.1		
	7.2	The sphinxsetup configuration setting	
	7.3 7.4	LaTeX macros and environments	
	7.4	Macros	
		The \sphinxbox command	
		Environments	
		Miscellany	
		Wilscending)]
8	Get s	upport 43	33
9	Cont	ribute to Sphinx 43	34
	9.1	Contributing to Sphinx	
	7.1	Get help	
		Bug Reports and Feature Requests	
		Contribute code	
		Getting started	
		Coding style	
		Unit tests	
		Contribute documentation	
		Build the documentation	
		Translations	
		Debugging tips	
		Updating generated files	
	9.2	Sphinx's release process	
	·-	Versioning	
		Deprecating a feature	
		Deprecation policy	
		Deprecation warnings	
		Python version support policy	
		Release procedures	
	9.3	Organization of the Sphinx project	
		Core developers	
		Guidelines	
		Membership	
		Other contributors	42
	9.4	Sphinx Code of Conduct	
10	_	ax FAQ	
	10.1	How do I	
		Using Sphinx with	
	10.3	Sphinx vs. Docutils	17

	10.4	Epub info	17
	10.5	Texinfo info	49
		Displaying Links	
		Notes	
11	Sphin	x authors 45	51
		Maintainers	51
		Contributors	51
		Former maintainers	53
12	Comr	nand-line tools 45	55
	12.1	Core applications	55
		sphinx-quickstart	55
		Synopsis	
		Description	
		Options	
		See also	
		sphinx-build	
		TE TO THE TOTAL TOTAL TO THE TH	
		3 1	
		F	
		Options	
		Environment Variables	
		Deprecation Warnings	
		See also	
	12.2	Additional applications	
		sphinx-apidoc	
		Synopsis	52
		Description	53
		Options	53
		Environment	55
		See also	55
		sphinx-autogen	55
		Synopsis	55
		Description	
		Options	
		Example	
		See also	
		See also	,
13	Gloss	ary 40	67
	01000	•••	•
14	Chan	gelog 40	59
	14.1	Release 8.1.2 (in development)	59
	14.2	Release 8.1.1 (released Oct 11, 2024)	59
	14.3	Release 8.1.0 (released Oct 10, 2024)	
	14.4	Prior releases	
	11,1	11101 10100000	' 1
15	Proie	cts using Sphinx 63	39
	15.1	Documentation using the alabaster theme	
	15.2	Documentation using the classic theme	
	15.3	Documentation using the chassic meme	
	15.4	Documentation using the sprintxdoc theme	_
	15.4		_
	15.6	Documentation using Read the Docs Sphinx Theme	
	15.7	Documentation using Sphinx Bootstrap Theme	
	15.8	Documentation using PyData Sphinx Theme	۱ د

Index		661	
Python Module Index			
15.14	Projects integrating Sphinx functionality	657	
15.13	Theses produced using Sphinx	657	
15.12	Books produced using Sphinx	656	
15.11	Homepages and other non-documentation sites	655	
15.10	Documentation using a custom theme or integrated in a website	653	
15.9	Documentation using Furo theme	652	

INSTALLING SPHINX

Sphinx is a Python application. It can be installed in one of the ways described below.

Installation methods

- PyPI package
- Conda package
- OS-specific package manager
 - Linux
 - macOS
 - Windows
- Docker
- Installation of the latest development release
- Installation from source

After installation, you can check that Sphinx is available by running

\$ sphinx-build --version

This should print out the Sphinx version number.



For local development, it is generally recommended to install Sphinx into a non-global environment (using for example $venv^3$ or $conda^4$ environments). This will allow for the use of separate sphinx versions and third-party extensions for each sphinx project.

1.1 PyPI package

Sphinx packages are published on the Python Package Index⁵ (PyPI). The preferred tool for installing packages from PyPI is **pip**, which is included in all modern versions of Python.

³ https://docs.python.org/3/library/venv.html

⁴ https://conda.io/projects/conda/en/latest/user-guide/getting-started.html

⁵ https://pypi.org/project/Sphinx/

Run the following command:

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



To avoid issues when rebuilding your environment, it is advisable to pin sphinx and third-party extension versions in a requirements.txt file⁶:

```
$ pip install -r requirements.txt
```

Or, if writing documentation for a Python package, place the dependencies in the pyproject.toml file⁷:

```
$ pip install .[docs]
```

1.2 Conda package

To work with **conda**, you need a conda-based Python distribution such as anaconda⁸, miniconda⁹, miniforge¹⁰ or micromamba¹¹.

Sphinx is available both via the anaconda main channel (maintained by Anaconda Inc.)

```
$ conda install sphinx
```

as well as via the conda-forge community channel

```
$ conda install -c conda-forge sphinx
```

1.3 OS-specific package manager

You may install a global version of Sphinx into your system using OS-specific package managers. However, be aware that this is less flexible and you may run into compatibility issues if you want to work across different projects.

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.

 $^{^6\} https://pip.pypa.io/en/stable/reference/requirements-file-format/$

⁷ https://packaging.python.org/en/latest/guides/writing-pyproject-toml/#dependencies-optional-dependencies

⁸ https://docs.anaconda.com/anaconda/

⁹ https://docs.anaconda.com/miniconda/

¹⁰ https://github.com/conda-forge/miniforge/

¹¹ https://mamba.readthedocs.io/en/latest/installation/micromamba-installation.html

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 Homebrew¹², MacPorts¹³.

Homebrew

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

For more information, refer to the package overview¹⁴.

MacPorts

Install either python3x-sphinx using port:

```
$ sudo port install py312-sphinx
```

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

```
$ sudo port select --set python python312
$ sudo port select --set sphinx py312-sphinx
```

For more information, refer to the package overview¹⁵.

Windows

Sphinx can be install using Chocolatey¹⁶.

Chocolatey

```
$ choco install sphinx
```

You would need to install Chocolatey¹⁷ before running this.

For more information, refer to the chocolatey page ¹⁸.

```
12 https://brew.sh/
```

13 https://www.macports.org/

14 https://formulae.brew.sh/formula/sphinx-doc

https://www.macports.org/ports.php?by=library&substr=py312-sphinx

16 https://chocolatey.org/

17 https://chocolatey.org/install

18 https://chocolatey.org/packages/sphinx/

1.4 Docker

Docker images for Sphinx are published on the Docker Hub¹⁹. There are two kind of images:

- sphinxdoc/sphinx²⁰
- sphinxdoc/sphinx-latexpdf²¹

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.



1 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 use the following command to build HTML document:

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

For more details, please read README file²² of docker images.

1.5 Installation of the latest development release

You can install the latest development from PyPI using the --pre flag:

\$ pip install -U --pre sphinx



Warning

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.

1.6 Installation from source

You can install Sphinx directly from a clone of the Git repository²³. This can be done either by cloning the repo and installing from the local clone, on simply installing directly via git.

¹⁹ https://hub.docker.com/

²⁰ https://hub.docker.com/r/sphinxdoc/sphinx

²¹ https://hub.docker.com/r/sphinxdoc/sphinx-latexpdf

²² https://hub.docker.com/r/sphinxdoc/sphinx

²³ https://github.com/sphinx-doc/sphinx

```
$ 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.gz²⁴ or zip²⁵ format. Once downloaded and extracted, these can be installed with pip as above.

https://github.com/sphinx-doc/sphinx/archive/master.tar.gz
 https://github.com/sphinx-doc/sphinx/archive/master.zip

CHAPTER

TWO

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 its *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, reStructuredText.

For a great "introduction" to writing docs in general – the whys and hows, see also Write the docs²⁶, written by Eric Holscher.

2.1 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.²⁹

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

2.2 Defining document structure

Let's assume you've run **sphinx-quickstart**. It created a source directory with **conf.py** and a root document, **index.rst**. The main function of the *root 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.

1 reStructuredText directives

²⁶ https://www.writethedocs.org/guide/writing/beginners-guide-to-docs/

²⁹ 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.

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.

See also

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.)

2.3 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.

→ See also

reStructuredText, including markup added by Sphinx.

2.4 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 -M html sourcedir outputdir
```

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

See also

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.

1 How do I generate PDF documents?

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

2.5 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.



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

2.6 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.path²⁷ 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.



Configuration for documentation of all available config values.

2.7 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*.

²⁷ https://docs.python.org/3/library/sys.html#sys.path



sphinx.ext.autodoc for the complete description of the features of autodoc.

2.8 Intersphinx

Many Sphinx documents including the Python documentation²⁸ 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 <code>intersphinx_mapping</code> like:

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

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 <code>intersphinx_mapping</code> (this needs access to the URL in order to download the list of valid targets). Intersphinx also works for some other <code>domain</code>'s roles including:ref:, however it doesn't work for :doc: as that is non-domain role.

See also

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

2.9 More topics to be covered

- Other extensions:
- · Static files
- · Selecting a theme
- Templating
- Using extensions
- Writing extensions

2.8. Intersphinx

²⁸ https://docs.python.org/3

CHAPTER

THREE

BUILD YOUR FIRST PROJECT

In this tutorial you will build a simple documentation project using Sphinx, and view it in your browser as HTML. The project will include narrative, handwritten documentation, as well as autogenerated API documentation.

The tutorial is aimed towards Sphinx newcomers willing to learn the fundamentals of how projects are created and structured. You will create a fictional software library to generate random food recipes that will serve as a guide throughout the process, with the objective of properly documenting it.

To showcase Sphinx capabilities for code documentation you will use Python, which also supports *automatic* documentation generation.



Several other languages are natively supported in Sphinx for *manual* code documentation, however they require extensions for *automatic* code documentation, like Breathe³⁰.

To follow the instructions you will need access to a Linux-like command line and a basic understanding of how it works, as well as a working Python installation for development, since you will use *Python virtual environments* to create the project.

3.1 Getting started

Setting up your project and development environment

In a new directory, create a file called README.rst with the following content.

Listing 1: README.rst

Lumache

I umache (/lu!make

Lumache (/lu'make/) is a Python library for cooks and food lovers that creates recipes mixing random ingredients.

It is a good moment to create a Python virtual environment and install the required tools. For that, open a command line terminal, cd into the directory you just created, and run the following commands:

```
$ python -m venv .venv
$ source .venv/bin/activate
(.venv) $ python -m pip install sphinx
```

³⁰ https://breathe.readthedocs.io/

1 Note

The installation method used above is described in more detail in *PyPI package*. For the rest of this tutorial, the instructions will assume a Python virtual environment.

If you executed these instructions correctly, you should have the Sphinx command line tools available. You can do a basic verification running this command:

```
(.venv) $ sphinx-build --version sphinx-build 4.0.2
```

If you see a similar output, you are on the right path!

Creating the documentation layout

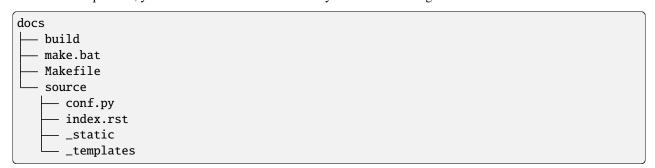
Then from the command line, run the following command:

```
(.venv) $ sphinx-quickstart docs
```

This will present to you a series of questions required to create the basic directory and configuration layout for your project inside the docs folder. To proceed, answer each question as follows:

- > Separate source and build directories (y/n) [n]: Write "y" (without quotes) and press Enter.
- > Project name: Write "Lumache" (without quotes) and press Enter.
- > Author name(s): Write "Graziella" (without quotes) and press Enter.
- > Project release []: Write "0.1" (without quotes) and press Enter.
- > Project language [en]: Leave it empty (the default, English) and press Enter.

After the last question, you will see the new docs directory with the following content.



The purpose of each of these files is:

build/

An empty directory (for now) that will hold the rendered documentation.

make.bat and Makefile

Convenience scripts to simplify some common Sphinx operations, such as rendering the content.

source/conf.py

A Python script holding the configuration of the Sphinx project. It contains the project name and release you specified to sphinx-quickstart, as well as some extra configuration keys.

source/index.rst

The *root document* of the project, which serves as welcome page and contains the root of the "table of contents tree" (or *toctree*).

Thanks to this bootstrapping step, you already have everything needed to render the documentation as HTML for the first time. To do that, run this command:

```
(.venv) $ sphinx-build -M html docs/source/ docs/build/
```

And finally, open docs/build/html/index.html in your browser. You should see something like this:



©2021, Graziella. | Powered by Sphinx 4.0.2 & Alabaster 0.7.12 | Page source

Fig. 1: Freshly created documentation of Lumache

There we go! You created your first HTML documentation using Sphinx. Now you can start *customizing it*.

3.2 First steps to document your project using Sphinx

Building your HTML documentation

The index.rst file that sphinx-quickstart created has some content already, and it gets rendered as the front page of your HTML documentation. It is written in reStructuredText, a powerful markup language.

Modify the file as follows:

Listing 2: docs/source/index.rst

This showcases several features of the reStructuredText syntax, including:

- a **section header** using === for the underline,
- two examples of *Inline markup*: **strong emphasis** (typically bold) and *emphasis* (typically italics),
- an inline external link,
- and a note **admonition** (one of the available *directives*)

Now to render it with the new content, you can use the sphinx-build command as before, or leverage the convenience script as follows:

```
(.venv) $ cd docs
(.venv) $ make html
```

After running this command, you will see that index.html reflects the new changes!

Building your documentation in other formats

Sphinx supports a variety of formats apart from HTML, including PDF, EPUB, *and more*. For example, to build your documentation in EPUB format, run this command from the docs directory:

```
(.venv) $ make epub
```

After that, you will see the files corresponding to the e-book under docs/build/epub/. You can either open Lumache epub with an EPUB-compatible e-book viewer, like Calibre³¹, or preview index.xhtml on a web browser.

1 Note

To quickly display a complete list of possible output formats, plus some extra useful commands, you can run make help.

Each output format has some specific configuration options that you can tune, *including EPUB*. For instance, the default value of *epub_show_urls* is inline, which means that, by default, URLs are shown right after the corresponding link, in parentheses. You can change that behavior by adding the following code at the end of your conf.py:

```
# EPUB options
epub_show_urls = 'footnote'
```

With this configuration value, and after running make epub again, you will notice that URLs appear now as footnotes, which avoids cluttering the text. Sweet! Read on to explore *other ways to customize Sphinx*.

1 Note

Generating a PDF using Sphinx can be done running make latexpdf, provided that the system has a working LaTeX installation, as explained in the documentation of *sphinx.builders.latex.LaTeXBuilder*. Although this is perfectly feasible, such installations are often big, and in general LaTeX requires careful configuration in some cases, so PDF generation is out of scope for this tutorial.

3.3 More Sphinx customization

There are two main ways to customize your documentation beyond what is possible with core Sphinx: extensions and themes.

Enabling a built-in extension

In addition to these configuration values, you can customize Sphinx even more by using *extensions*. Sphinx ships several *builtin ones*, and there are many more *maintained by the community*.

For example, to enable the *sphinx.ext.duration* extension, locate the extensions list in your conf.py and add one element as follows:

³¹ https://calibre-ebook.com/

Listing 3: docs/source/conf.py

```
# Add any Sphinx extension module names here, as strings. They can be
# extensions coming with Sphinx (named 'sphinx.ext.*') or your custom
# ones.
extensions = [
    'sphinx.ext.duration',
]
```

After that, every time you generate your documentation, you will see a short durations report at the end of the console output, like this one:

Using a third-party HTML theme

Themes, on the other hand, are a way to customize the appearance of your documentation. Sphinx has several *builtin themes*, and there are also third-party ones³².

For example, to use the Furo³³ third-party theme in your HTML documentation, first you will need to install it with pip in your Python virtual environment, like this:

```
(.venv) $ pip install furo
```

And then, locate the html_theme variable on your conf.py and replace its value as follows:

Listing 4: docs/source/conf.py

```
# The theme to use for HTML and HTML Help pages. See the documentation for
# a list of builtin themes.
#
html_theme = 'furo'
```

With this change, you will notice that your HTML documentation has now a new appearance:

It is now time to expand the narrative documentation and split it into several documents.

3.4 Narrative documentation in Sphinx

Structuring your documentation across multiple pages

The file index.rst created by sphinx-quickstart is the *root document*, whose main function is to serve as a welcome page and to contain the root of the "table of contents tree" (or *toctree*). Sphinx allows you to assemble a project from different files, which is helpful when the project grows.

As an example, create a new file docs/source/usage.rst (next to index.rst) with these contents:

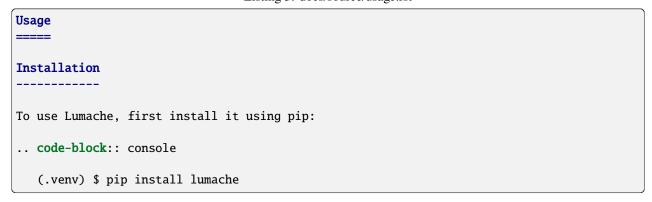
³² https://sphinx-themes.org/

³³ https://pradyunsg.me/furo/



Fig. 2: HTML documentation of Lumache with the Furo theme

Listing 5: docs/source/usage.rst



This new file contains two *section* headers, normal paragraph text, and a *code-block* directive that renders a block of content as source code, with appropriate syntax highlighting (in this case, generic console text).

The structure of the document is determined by the succession of heading styles, which means that, by using --- for the "Installation" section after === for the "Usage" section, you have declared "Installation" to be a *subsection* of "Usage".

To complete the process, add a toctree *directive* at the end of index.rst including the document you just created, as follows:

Listing 6: docs/source/index.rst



This step inserts that document in the root of the *toctree*, so now it belongs to the structure of your project, which so far looks like this:

```
index
└─ usage
```

If you build the HTML documentation running make html, you will see that the toctree gets rendered as a list of hyperlinks, and this allows you to navigate to the new page you just created. Neat!

A Warning

Documents outside a *toctree* will result in WARNING: document isn't included in any toctree messages during the build process, and will be unreachable for users.

Adding cross-references

One powerful feature of Sphinx is the ability to seamlessly add *cross-references* to specific parts of the documentation: a document, a section, a figure, a code object, etc. This tutorial is full of them!

To add a cross-reference, write this sentence right after the introduction paragraph in index.rst:

Listing 7: docs/source/index.rst

```
Check out the :doc:`usage` section for further information.
```

The *doc role* you used automatically references a specific document in the project, in this case the usage.rst you created earlier.

Alternatively, you can also add a cross-reference to an arbitrary part of the project. For that, you need to use the ref role, and add an explicit *label* that acts as a target³⁴.

For example, to reference the "Installation" subsection, add a label right before the heading, as follows:

Listing 8: docs/source/usage.rst

Usage ===== .. _installation: Installation ------

And make the sentence you added in index.rst look like this:

³⁴ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#hyperlink-targets

Listing 9: docs/source/index.rst

```
Check out the :doc:`usage` section for further information, including how to :ref:`install <installation>` the project.
```

Notice a trick here: the install part specifies how the link will look like (we want it to be a specific word, so the sentence makes sense), whereas the <installation> part refers to the actual label we want to add a cross-reference to. If you do not include an explicit title, hence using :ref:`installation`, the section title will be used (in this case, Installation). Both the :doc: and the :ref: roles will be rendered as hyperlinks in the HTML documentation.

What about documenting code objects in Sphinx? Read on!

3.5 Describing code in Sphinx

In the *previous sections of the tutorial* you can read how to write narrative or prose documentation in Sphinx. In this section you will describe code objects instead.

Sphinx supports documenting code objects in several languages, namely Python, C, C++, JavaScript, and reStructured-Text. Each of them can be documented using a series of directives and roles grouped by *domain*. For the remainder of the tutorial you will use the Python domain, but all the concepts seen in this section apply for the other domains as well.

Python

Documenting Python objects

Sphinx offers several roles and directives to document Python objects, all grouped together in *the Python domain*. For example, you can use the *py:function* directive to document a Python function, as follows:

Listing 10: docs/source/usage.rst

```
Creating recipes
-----
To retrieve a list of random ingredients,
you can use the ``lumache.get_random_ingredients()`` function:
.. py:function:: lumache.get_random_ingredients(kind=None)

Return a list of random ingredients as strings.

:param kind: Optional "kind" of ingredients.
:type kind: list[str] or None
:return: The ingredients list.
:rtype: list[str]
```

Which will render like this:

Notice several things:

- Sphinx parsed the argument of the .. py:function directive and highlighted the module, the function name, and the parameters appropriately.
- The directive content includes a one-line description of the function, as well as an *info field list* containing the function parameter, its expected type, the return value, and the return type.

Creating recipes

To retrieve a list of random ingredients, you can use the lumache.get_random_ingredients() function:

```
| Iumache.get_random_ingredients(kind=None)
| Return a list of random ingredients as strings.
| PARAMETERS | kind (list[str] or None) - Optional "kind" of ingredients.
| RETURNS | The ingredients list.
| RETURN TYPE | list[str]
```

Fig. 3: The rendered result of documenting a Python function in Sphinx

1 Note

The py: prefix specifies the *domain*. You may configure the default domain so you can omit the prefix, either globally using the *primary_domain* configuration, or use the *default-domain* directive to change it from the point it is called until the end of the file. For example, if you set it to py (the default), you can write . . function:: directly.

Cross-referencing Python objects

By default, most of these directives generate entities that can be cross-referenced from any part of the documentation by using *a corresponding role*. For the case of functions, you can use *py:func* for that, as follows:

Listing 11: docs/source/usage.rst

```
The ``kind`` parameter should be either ``"meat"``, ``"fish"``, or ``"veggies"``. Otherwise, :py:func:`lumache.get_random_ingredients` will raise an exception.
```

When generating code documentation, Sphinx will generate a cross-reference automatically just by using the name of the object, without you having to explicitly use a role for that. For example, you can describe the custom exception raised by the function using the *py:exception* directive:

Listing 12: docs/source/usage.rst

```
.. py:exception:: lumache.InvalidKindError

Raised if the kind is invalid.
```

Then, add this exception to the original description of the function:

Listing 13: docs/source/usage.rst

```
.. py:function:: lumache.get_random_ingredients(kind=None)

Return a list of random ingredients as strings.
```

(continues on next page)

(continued from previous page)

```
:param kind: Optional "kind" of ingredients.
:type kind: list[str] or None
:raise lumache.InvalidKindError: If the kind is invalid.
:return: The ingredients list.
:rtype: list[str]
```

And finally, this is how the result would look:

Creating recipes

```
lumache.get_random_ingredients(kind=None)

Return a list of random ingredients as strings.
```

To retrieve a list of random ingredients, you can use the lumache.get_random_ingredients() function:

```
PARAMETERS kind (list[str] or None) – Optional "kind" of ingredients.
```

RAISES

lumache.InvalidKindError – If the kind is invalid.

RETURNS

The ingredients list.

RETURN TYPE

list[str]
The kind parameter should be either "meat", "fish", or "veggies". Otherwise,

lumache.get_random_ingredients() will raise an exception.

exception lumache.InvalidKindError

Raised if the kind is invalid.

Fig. 4: HTML result of documenting a Python function in Sphinx with cross-references

Beautiful, isn't it?

Including doctests in your documentation

Since you are now describing code from a Python library, it will become useful to keep both the documentation and the code as synchronized as possible. One of the ways to do that in Sphinx is to include code snippets in the documentation, called *doctests*, that are executed when the documentation is built.

To demonstrate doctests and other Sphinx features covered in this tutorial, Sphinx will need to be able to import the code. To achieve that, write this at the beginning of conf.py:

Listing 14: docs/source/conf.py

```
# If extensions (or modules to document with autodoc) are in another directory,
# add these directories to sys.path here.
import pathlib
import sys
sys.path.insert(0, pathlib.Path(__file__).parents[2].resolve().as_posix())
```

1 Note

An alternative to changing the sys.path³⁵ variable is to create a pyproject.toml file and make the code installable, so it behaves like any other Python library. However, the sys.path approach is simpler.

Then, before adding doctests to your documentation, enable the *doctest* extension in conf.py:

Listing 15: docs/source/conf.py

```
extensions = [
    'sphinx.ext.duration',
    'sphinx.ext.doctest',
]
```

Next, write a doctest block as follows:

Listing 16: docs/source/usage.rst

```
>>> import lumache
>>> lumache.get_random_ingredients()
['shells', 'gorgonzola', 'parsley']
```

Doctests include the Python instructions to be run preceded by >>>, the standard Python interpreter prompt, as well as the expected output of each instruction. This way, Sphinx can check whether the actual output matches the expected one.

To observe how a doctest failure looks like (rather than a code error as above), let's write the return value incorrectly first. Therefore, add a function get_random_ingredients like this:

Listing 17: lumache.py

```
def get_random_ingredients(kind=None):
    return ["eggs", "bacon", "spam"]
```

You can now run make doctest to execute the doctests of your documentation. Initially this will display an error, since the actual code does not behave as specified:

³⁵ https://docs.python.org/3/library/sys.html#sys.path

(continued from previous page)

As you can see, doctest reports the expected and the actual results, for easy examination. It is now time to fix the function:

Listing 18: lumache.py

```
def get_random_ingredients(kind=None):
    return ["shells", "gorgonzola", "parsley"]
```

And finally, make doctest reports success!

For big projects though, this manual approach can become a bit tedious. In the next section, you will see *how to automate the process*.

Other languages (C, C++, others)

Documenting and cross-referencing objects

Sphinx also supports documenting and cross-referencing objects written in other programming languages. There are four additional built-in domains: C, C++, JavaScript, and reStructuredText. Third-party extensions may define domains for more languages, such as

- Fortran³⁶,
- Julia³⁷, or
- PHP³⁸.

For example, to document a C++ type definition, you would use the built-in cpp: type directive, as follows:

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

A typedef-like declaration of a type.
```

Which would give the following result:

typedef std::vector<int> CustomList

A typedef-like declaration of a type.

All such directives then generate references that can be cross-referenced by using the corresponding role. For example, to reference the previous type definition, you can use the *cpp:type* role as follows:

```
Cross reference to :cpp:type:`CustomList`.
```

Which would produce a hyperlink to the previous definition: CustomList.

³⁶ https://sphinx-fortran.readthedocs.io

³⁷ https://bastikr.github.io/sphinx-julia

³⁸ https://github.com/markstory/sphinxcontrib-phpdomain

3.6 Automatic documentation generation from code

In the *previous section* of the tutorial you manually documented a Python function in Sphinx. However, the description was out of sync with the code itself, since the function signature was not the same. Besides, it would be nice to reuse **Python docstrings**³⁹ in the documentation, rather than having to write the information in two places.

Fortunately, the autodoc extension provides this functionality.

Reusing signatures and docstrings with autodoc

To use autodoc, first add it to the list of enabled extensions:

Listing 19: docs/source/conf.py

```
extensions = [
    'sphinx.ext.duration',
    'sphinx.ext.doctest',
    'sphinx.ext.autodoc',
]
```

Next, move the content of the .. py:function directive to the function doestring in the original Python file, as follows:

Listing 20: lumache.py

```
def get_random_ingredients(kind=None):
    """
    Return a list of random ingredients as strings.

    :param kind: Optional "kind" of ingredients.
    :type kind: list[str] or None
    :raise lumache.InvalidKindError: If the kind is invalid.
    :return: The ingredients list.
    :rtype: list[str]
    """
    return ["shells", "gorgonzola", "parsley"]
```

Finally, replace the . . py: function directive from the Sphinx documentation with autofunction:

Listing 21: docs/source/usage.rst

```
you can use the ``lumache.get_random_ingredients()`` function:
.. autofunction:: lumache.get_random_ingredients
```

If you now build the HTML documentation, the output will be the same! With the advantage that it is generated from the code itself. Sphinx took the reStructuredText from the docstring and included it, also generating proper cross-references.

You can also autogenerate documentation from other objects. For example, add the code for the InvalidKindError exception:

³⁹ https://peps.python.org/pep-0257/#what-is-a-docstring

Listing 22: lumache.py

```
class InvalidKindError(Exception):
    """Raised if the kind is invalid."""
    pass
```

And replace the .. py:exception directive with autoexception as follows:

Listing 23: docs/source/usage.rst

```
or ``"veggies"``. Otherwise, :py:func:`lumache.get_random_ingredients`
will raise an exception.

.. autoexception:: lumache.InvalidKindError
```

And again, after running make html, the output will be the same as before.

Generating comprehensive API references

While using sphinx.ext.autodoc makes keeping the code and the documentation in sync much easier, it still requires you to write an auto* directive for every object you want to document. Sphinx provides yet another level of automation: the *autosummary* extension.

The *autosummary* directive generates documents that contain all the necessary autodoc directives. To use it, first enable the autosummary extension:

Listing 24: docs/source/conf.py

```
extensions = [
   'sphinx.ext.duration',
   'sphinx.ext.doctest',
   'sphinx.ext.autodoc',
   'sphinx.ext.autosummary',
]
```

Next, create a new api.rst file with these contents:

Listing 25: docs/source/api.rst

```
API ===

.. autosummary::
    :toctree: generated

lumache
```

Remember to include the new document in the root toctree:

Listing 26: docs/source/index.rst

```
Contents
-----
.. toctree::
(continues on next page)
```

(continued from previous page)

usage api

Finally, after you build the HTML documentation running make html, it will contain two new pages:

- api.html, corresponding to docs/source/api.rst and containing a table with the objects you included in the autosummary directive (in this case, only one).
- generated/lumache.html, corresponding to a newly created reStructuredText file generated/lumache.rst and containing a summary of members of the module, in this case one function and one exception.

lumache

Lumache - Python library for cooks and food lovers.

FUNCTIONS

get_random_ingredients ([kind]) Return a list of random ingredients as strings.

EXCEPTIONS

InvalidKindError Raised if the kind is invalid.

Fig. 5: Summary page created by autosummary

Each of the links in the summary page will take you to the places where you originally used the corresponding autodoc directive, in this case in the usage.rst document.



The generated files are based on Jinja2 templates⁴⁰ that *can be customized*, but that is out of scope for this tutorial.

3.7 Appendix: Deploying a Sphinx project online

When you are ready to show your documentation project to the world, there are many options available to do so. Since the HTML generated by Sphinx is static, you can decouple the process of building your HTML documentation from hosting such files in the platform of your choice. You will not need a sophisticated server running Python: virtually every web hosting service will suffice.

Therefore, the challenge is less how or where to serve the static HTML, but rather how to pick a workflow that automatically updates the deployed documentation every time there is a change in the source files.

The following sections describe some of the available options to deploy your online documentation, and give some background information. If you want to go directly to the practical part, you can skip to *Publishing your documentation sources*.

⁴⁰ https://jinja.palletsprojects.com/

Sphinx-friendly deployment options

There are several possible options you have to host your Sphinx documentation. Some of them are:

Read the Docs

Read the Docs⁴¹ is an online service specialized in hosting technical documentation written in Sphinx, as well as MkDocs. They have a number of extra features, such as versioned documentation, traffic and search analytics, custom domains, user-defined redirects, and more.

GitHub Pages

GitHub Pages⁴² is a simple static web hosting tightly integrated with GitHub⁴³: static HTML is served from one of the branches of a project, and usually sources are stored in another branch so that the output can be updated every time the sources change (for example using GitHub Actions⁴⁴). It is free to use and supports custom domains.

GitLab Pages

GitLab Pages⁴⁵ is a similar concept to GitHub Pages, integrated with $GitLab^{46}$ and usually automated with GitLab CI^{47} instead.

Netlify

Netlify⁴⁸ is a sophisticated hosting for static sites enhanced by client-side web technologies like JavaScript (so-called "Jamstack", They offer support for headless content management systems and serverless computing.

Your own server

You can always use your own web server to host Sphinx HTML documentation. It is the option that gives more flexibility, but also more complexity.

All these options have zero cost, with the option of paying for extra features.

Embracing the "Docs as Code" philosophy

The free offerings of most of the options listed above require your documentation sources to be publicly available. Moreover, these services expect you to use a Version Control System⁵⁰, a technology that tracks the evolution of a collection of files as a series of snapshots ("commits"). The practice of writing documentation in plain text files with the same tools as the ones used for software development is commonly known as "Docs as Code"⁵¹.

The most popular Version Control System nowadays is Git⁵², a free and open source tool that is the backbone of services like GitHub and GitLab. Since both Read the Docs and Netlify have integrations with GitHub and GitLab, and both GitHub and GitLab have an integrated Pages product, the most effective way of automatically build your documentation online is to upload your sources to either of these Git hosting services.

Publishing your documentation sources

GitHub

The quickest way to upload an existing project to GitHub is to:

1. Sign up for a GitHub account⁵³.

- 41 https://readthedocs.org/
- 42 https://pages.github.com/
- 43 https://github.com/
- 44 https://github.com/features/actions
- 45 https://about.gitlab.com/stages-devops-lifecycle/pages/
- 46 https://gitlab.com/
- ⁴⁷ https://about.gitlab.com/stages-devops-lifecycle/continuous-integration/
- 48 https://www.netlify.com/
- 49 https://jamstack.org/
- 50 https://en.wikipedia.org/wiki/Version_control
- 51 https://www.writethedocs.org/guide/docs-as-code/
- 52 https://git-scm.com/
- 53 https://github.com/signup

- 2. Create a new repository⁵⁴.
- 3. Open the "Upload files" page⁵⁵ of your new repository.
- 4. Select the files on your operating system file browser (in your case README.rst, lumache.py, the makefiles under the docs directory, and everything under docs/source) and drag them to the GitHub interface to upload them all.
- 5. Click on the *Commit changes* button.



Make sure you don't upload the docs/build directory, as it contains the output generated by Sphinx and it will change every time you change the sources, complicating your workflow.

These steps do not require access to the command line or installing any additional software. To learn more, read this quickstart tutorial or consult the official GitHub documentation or the command line or installing any additional software.

GitLab

Similarly to GitHub, the fastest way to upload your project to GitLab is using the web interface:

- 1. Sign up for a GitLab account⁵⁸.
- 2. Create a new blank project⁵⁹.
- 3. Upload the project files (in your case README.rst, lumache.py, the makefiles under the docs directory, and everything under docs/source) one by one using the *Upload File* button⁶².

Again, these steps do not require additional software on your computer. To learn more, you can:

- Follow this tutorial⁶⁰ to install Git on your machine.
- Browse the GitLab User documentation⁶¹ to understand the possibilities of the platform.

1 Note

Make sure you don't upload the docs/build directory, as it contains the output generated by Sphinx and it will change every time you change the sources, complicating your workflow.

Publishing your HTML documentation

Read the Docs

Read the Docs⁶⁴ offers integration with both GitHub and GitLab. The quickest way of getting started is to follow the RTD tutorial⁶⁵, which is loosely based on this one. You can publish your sources on GitHub as explained *in the previous*

- 54 https://github.com/new
- 55 https://docs.github.com/en/repositories/working-with-files/managing-files/adding-a-file-to-a-repository
- 56 https://docs.github.com/en/get-started/quickstart
- 57 https://docs.github.com/en/get-started
- 58 https://gitlab.com/users/sign_up
- ⁵⁹ https://gitlab.com/projects/new
- ⁶² At the time of writing, uploading whole directories to GitLab using only the web interface⁶³ is not yet implemented.
- 63 https://gitlab.com/gitlab-org/gitlab/-/issues/228490
- 60 https://docs.gitlab.com/ee/gitlab-basics/start-using-git.html
- 61 https://docs.gitlab.com/ee/user/index.html
- 64 https://readthedocs.org/
- 65 https://docs.readthedocs.io/en/stable/tutorial/index.html

section, then skip directly to Creating a Read the Docs account⁶⁶. If you choose GitLab instead, the process is similar.

GitHub Pages

GitHub Pages⁶⁷ requires you to *publish your sources* on GitHub⁶⁸. After that, you will need an automated process that performs the make html step every time the sources change. That can be achieved using GitHub Actions⁶⁹.

After you have published your sources on GitHub, create a file named .github/workflows/sphinx.yml in your repository with the following contents:

Listing 27: .github/workflows/

```
name: "Sphinx: Render docs"
on: push
jobs:
  build:
   runs-on: ubuntu-latest
   permissions:
      contents: write
   steps:
    uses: actions/checkout@v4
    - name: Build HTML
      uses: ammaraskar/sphinx-action@master
    - name: Upload artifacts
      uses: actions/upload-artifact@v4
      with:
       name: html-docs
       path: docs/build/html/
    - name: Deploy
      uses: peaceiris/actions-gh-pages@v3
      if: github.ref == 'refs/heads/main'
      with:
        github_token: ${{ secrets.GITHUB_TOKEN }}
       publish_dir: docs/build/html
```

This contains a GitHub Actions workflow with a single job of four steps:

- 1. Checkout the code.
- 2. Build the HTML documentation using Sphinx.
- 3. Attach the HTML output the artifacts to the GitHub Actions job, for easier inspection.
- 4. If the change happens on the default branch, take the contents of docs/build/html and push it to the gh-pages branch.

Next, you need to specify the dependencies for the make html step to be successful. For that, create a file docs/requirements.txt and add the following contents:

 $^{^{66}\} https://docs.readthedocs.io/en/stable/tutorial/index.html\#creating-a-read-the-docs-account$

⁶⁷ https://pages.github.com/

⁶⁸ https://github.com/

⁶⁹ https://github.com/features/actions

Listing 28: docs/requirements.txt

```
furo==2021.11.16
```

And finally, you are ready to enable GitHub Pages on your repository⁷⁰. For that, go to *Settings*, then *Pages* on the left sidebar, select the gh-pages branch in the "Source" dropdown menu, and click *Save*. After a few minutes, you should be able to see your HTML at the designated URL.

GitLab Pages

GitLab Pages⁷¹, on the other hand, requires you to *publish your sources* on GitLab⁷². When you are ready, you can automate the process of running make html using GitLab CI⁷³.

After you have published your sources on GitLab, create a file named .gitlab-ci.yml in your repository with these contents:

Listing 29: .gitlab-ci.yml

```
stages:
  deploy
pages:
  stage: deploy
  image: python:3.12-slim
  before_script:
    - apt-get update && apt-get install make --no-install-recommends -y
    - python -m pip install sphinx furo
    - cd docs && make html
  after_script:
    - mv docs/build/html/ ./public/
  artifacts:
   paths:
    public
  rules:
    - if: $CI_COMMIT_REF_NAME == $CI_DEFAULT_BRANCH
```

This contains a GitLab CI workflow with one job of several steps:

- 1. Install the necessary dependencies.
- 2. Build the HTML documentation using Sphinx.
- 3. Move the output to a known artifacts location.

1 Note

You will need to validate your account⁷⁴ by entering a payment method (you will be charged a small amount that will then be reimbursed).

 $^{^{70}\} https://docs.github.com/en/pages/getting-started-with-github-pages/configuring-a-publishing-source-for-your-github-pages-site$

⁷¹ https://about.gitlab.com/stages-devops-lifecycle/pages/

⁷² https://gitlab.com/

⁷³ https://about.gitlab.com/stages-devops-lifecycle/continuous-integration/

⁷⁴ https://about.gitlab.com/blog/2021/05/17/prevent-crypto-mining-abuse/#validating-an-account

After that, if the pipeline is successful, you should be able to see your HTML at the designated URL.

3.8 Where to go from here

This tutorial covered the very first steps to create a documentation project with Sphinx. To continue learning more about Sphinx, check out the rest of the documentation.

CHAPTER

FOUR

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*.

4.1 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 reStructuredText. For the authoritative reference, refer to the documentation 75.

reStructuredText Primer

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

♂ See also

The authoritative reStructuredText User Documentation⁷⁶. The "ref" links in this document link to the description of the individual constructs in the reStructuredText reference.

Paragraphs

The paragraph (ref⁷⁷) is the most basic block in a reStructuredText document. Paragraphs are simply chunks of text separated by one or more blank lines. As in Python, indentation is significant in reStructuredText, so all lines of the same paragraph must be left-aligned to the same level of indentation.

Inline markup

The standard reStructuredText 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.

⁷⁵ https://docutils.sourceforge.io/rst.html

⁷⁶ https://docutils.sourceforge.io/rst.html

⁷⁷ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#paragraphs

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 (ref⁷⁸) 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.
#. 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 (ref⁷⁹) 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 (ref⁸⁰) are created by just indenting them more than the surrounding paragraphs.

⁷⁸ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#bullet-lists

⁷⁹ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#definition-lists

⁸⁰ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#block-quotes

Line blocks (ref⁸¹) 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 (ref⁸², with caveats noted in *Field Lists*)
- option lists (ref⁸³)
- quoted literal blocks (ref⁸⁴)
- doctest blocks (ref⁸⁵)

Literal blocks

Literal code blocks (ref⁸⁶) 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 (ref⁸⁷) 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
```

⁸¹ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#line-blocks

⁸² https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#field-lists

⁸³ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#option-lists

⁸⁴ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#quoted-literal-blocks

⁸⁵ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#doctest-blocks

⁸⁶ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#literal-blocks

⁸⁷ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#doctest-blocks

Tables

For grid tables (ref⁸⁸), you have to "paint" the cell grid yourself. They look like this:

Simple tables (ref⁸⁹) 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:

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 (ref⁹⁰), like this:

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

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

Internal links

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

⁸⁸ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#grid-tables

⁸⁹ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#simple-tables

 $^{^{90}\} https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html\#hyperlink-targets$

Sections

Section headers (ref⁹¹) are created by underlining (and optionally overlining) the section title with a punctuation character, at least as long as the text:

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 Developer's Guide for documenting⁹² which you may follow:

- # with overline, for parts
- * with overline, for chapters
- = for sections
- · for subsections
- ^ for subsubsections
- · " for paragraphs

Of course, you are free to use your own marker characters (see the reStructuredText 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 (ref⁹³) 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^{94}) 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 95 - equivalent of *emphasis*

⁹¹ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#sections

⁹² https://devguide.python.org/documentation/markup/#sections

⁹³ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#field-lists

⁹⁴ https://docutils.sourceforge.io/docs/ref/rst/roles.html

⁹⁵ https://docutils.sourceforge.io/docs/ref/rst/roles.html#emphasis

- strong⁹⁶ equivalent of **strong**
- literal⁹⁷ equivalent of ``literal``
- subscript ⁹⁸ subscript text
- superscript ⁹⁹ superscript text
- title-reference ¹⁰⁰ for titles of books, periodicals, and other materials

Refer to *Roles* for roles added by Sphinx.

Explicit Markup

"Explicit markup" (ref¹⁰¹) is used in reStructuredText 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 (ref¹⁰²) is a generic block of explicit markup. Along with roles, it is one of the extension mechanisms of reStructuredText, and Sphinx makes heavy use of it.

Docutils supports the following directives:

- Admonitions: attention¹⁰³, caution¹⁰⁴, danger¹⁰⁵, error¹⁰⁶, hint¹⁰⁷, important¹⁰⁸, note¹⁰⁹, tip¹¹⁰, warning¹¹¹ and the generic admonition¹¹². (Most themes style only "note" and "warning" specially.)
- Images:
 - image¹¹³ (see also *Images* below)
 - figure 114 (an image with caption and optional legend)
- Additional body elements:
 - contents¹¹⁵ (a local, i.e. for the current file only, table of contents)
 - container 116 (a container with a custom class, useful to generate an outer <div> in HTML)
 - rubric¹¹⁷ (a heading without relation to the document sectioning)

```
96 https://docutils.sourceforge.io/docs/ref/rst/roles.html#strong
97 https://docutils.sourceforge.io/docs/ref/rst/roles.html#literal
98 https://docutils.sourceforge.io/docs/ref/rst/roles.html#subscript
99 https://docutils.sourceforge.io/docs/ref/rst/roles.html#superscript
100 https://docutils.sourceforge.io/docs/ref/rst/roles.html#title-reference
101 https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#explicit-markup-blocks
102 https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#directives
103 https://docutils.sourceforge.io/docs/ref/rst/directives.html#attention
104 https://docutils.sourceforge.io/docs/ref/rst/directives.html#caution
105 https://docutils.sourceforge.io/docs/ref/rst/directives.html#danger
106 https://docutils.sourceforge.io/docs/ref/rst/directives.html#error
107 https://docutils.sourceforge.io/docs/ref/rst/directives.html#hint
108 https://docutils.sourceforge.io/docs/ref/rst/directives.html#important
109 https://docutils.sourceforge.io/docs/ref/rst/directives.html#note
110 https://docutils.sourceforge.io/docs/ref/rst/directives.html#tip
111 https://docutils.sourceforge.io/docs/ref/rst/directives.html#warning
112 https://docutils.sourceforge.io/docs/ref/rst/directives.html#admonitions
113 https://docutils.sourceforge.io/docs/ref/rst/directives.html#image
114 https://docutils.sourceforge.io/docs/ref/rst/directives.html#figure
```

https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents
 https://docutils.sourceforge.io/docs/ref/rst/directives.html#container
 https://docutils.sourceforge.io/docs/ref/rst/directives.html#rubric

- topic¹¹⁸, sidebar¹¹⁹ (special highlighted body elements)
- parsed-literal 120 (literal block that supports inline markup)
- epigraph¹²¹ (a block quote with optional attribution line)
- highlights¹²², pull-quote¹²³ (block quotes with their own class attribute)
- compound 124 (a compound paragraph)

• Special tables:

- table 125 (a table with title)
- csv-table¹²⁶ (a table generated from comma-separated values)
- list-table 127 (a table generated from a list of lists)

• Special directives:

- raw¹²⁸ (include raw target-format markup)
- include ¹²⁹ (include reStructuredText from another file) in Sphinx, when given an absolute include file path, this directive takes it as relative to the source directory
- class¹³⁰ (assign a class attribute to the next element)



When the default domain contains a class directive, this directive will be shadowed. Therefore, Sphinx re-exports it as rst-class.

🗘 Tip

If you want to add a class to a directive, you may consider the :class: option¹³¹ instead, which is supported by most directives and allows for a more compact notation.

• HTML specifics:

- meta¹³² (generation of HTML <meta> tags, see also HTML Metadata below)
- title¹³³ (override document title)
- Influencing markup:

¹¹⁸ https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

¹¹⁹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar

¹²⁰ https://docutils.sourceforge.io/docs/ref/rst/directives.html#parsed-literal

¹²¹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#epigraph

https://docutils.sourceforge.io/docs/ref/rst/directives.html#highlights

¹²³ https://docutils.sourceforge.io/docs/ref/rst/directives.html#pull-quote

¹²⁴ https://docutils.sourceforge.io/docs/ref/rst/directives.html#compound-paragraph

¹²⁵ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table

¹²⁶ https://docutils.sourceforge.io/docs/ref/rst/directives.html#csv-table

¹²⁷ https://docutils.sourceforge.io/docs/ref/rst/directives.html#list-table

¹²⁸ https://docutils.sourceforge.io/docs/ref/rst/directives.html#raw-data-pass-through

¹²⁹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#include

¹³⁰ https://docutils.sourceforge.io/docs/ref/rst/directives.html#class

¹³¹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#common-options

¹³² https://docutils.sourceforge.io/docs/ref/rst/directives.html#meta

https://docutils.sourceforge.io/docs/ref/rst/directives.html#metadata-document-title

- default-role¹³⁴ (set a new default role)
- role¹³⁵ (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 sectnum¹³⁶, header¹³⁷ and footer¹³⁸.

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 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 or if options are present, by the same amount as the options.

Be careful as the indent is not a fixed number of whitespace, e.g. three, but any number whitespace. This can be surprising when a fixed indent is used throughout the document and can make a difference for directives which are sensitive to whitespace. Compare:

```
.. code-block::
    :caption: A cool example

    The output of this line starts with four spaces.
.. code-block::
    The output of this line has no spaces at the beginning.
```

In the first code block, the indent for the content was fixated by the option line to three spaces, consequently the content starts with four spaces. In the latter the indent was fixed by the content itself to seven spaces, thus it does not start with a space.

Images

reStructuredText supports an image directive (ref¹³⁹), used like so:

¹³⁴ https://docutils.sourceforge.io/docs/ref/rst/directives.html#default-role

¹³⁵ https://docutils.sourceforge.io/docs/ref/rst/directives.html#role

¹³⁶ https://docutils.sourceforge.io/docs/ref/rst/directives.html#sectnum

¹³⁷ https://docutils.sourceforge.io/docs/ref/rst/directives.html#header

¹³⁸ https://docutils.sourceforge.io/docs/ref/rst/directives.html#footer

¹³⁹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#image

```
.. 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^{140}) , 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 ([#]_).

Citations

Standard reStructuredText citations (ref^{141}) 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 #.

¹⁴⁰ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#footnotes

¹⁴¹ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#citations

Substitutions

reStructuredText supports "substitutions" (ref^{142}), 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 reStructuredText reference for substitutions¹⁴³ 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 144 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^{145}) . 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.
```

HTML Metadata

The meta¹⁴⁶ directive allows specifying the HTML metadata element¹⁴⁷ 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">
```

 $^{^{142}\} https://docutils.sourceforge.io/docs/ref/rst/restructured text. html \# substitution-definitions$

¹⁴³ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#substitution-definitions

¹⁴⁴ https://docutils.sourceforge.io/docs/ref/rst/directives.html#include

¹⁴⁵ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#comments

¹⁴⁶ https://docutils.sourceforge.io/docs/ref/rst/directives.html#meta

¹⁴⁷ https://developer.mozilla.org/en-US/docs/Web/HTML/Element/meta

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 reStructuredText 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 reStructuredText 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 ¹⁴⁸ for the details.
- No nested inline markup: Something like *see :func:`foo`* is not possible.

Roles

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



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-references

See Cross-references.

Cross-reference roles include:

- any
- doc
- download
- envvar

 $^{^{148}\} https://docutils.sourceforge.io/docs/ref/rst/restructured text.html \# substitution-definitions$

- keyword
- numref
- option (and the deprecated cmdoption)
- ref
- term
- token

Inline code highlighting

:code:

An *inline* code example. When used directly, this role just displays the text *without* syntax highlighting, as a literal.

```
By default, inline code such as :code:`1 + 2` just displays without highlighting.
```

Displays: By default, inline code such as 1 + 2 just displays without highlighting.

Unlike the *code-block* directive, this role does not respect the default language set by the *highlight* directive.

To enable syntax highlighting, you must first use the Docutils role¹⁴⁹ directive to define a custom role associated with a specific language:

```
.. role:: python(code)
   :language: python
In Python, :python:`1 + 2` is equal to :python:`3`.
```

To display a multi-line code example, use the *code-block* directive instead.

Math

:math:

Role for inline math. Use like this:

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

Displays: Since Pythagoras, we know that $a^2 + b^2 = c^2$.

: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.

For example: :abbr:`LIFO (last-in, first-out)` displays LIFO (last-in, first-out).

Added in version 0.6.

¹⁴⁹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#role

:command:

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

For example: rm

:dfn:

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

For example: binary mode

: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/python3.{x}/site-packages` ...
```

Displays: ... is installed in /usr/lib/python3.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 curses¹⁵⁰ 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 (for example: :guilabel:`&Cancel` displays <u>Cancel</u>). 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`, displaying C-x C-f and Control-x Control-f respectively.

: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` displays <code>Content-Type</code>.

:makevar:

The name of a make variable.

For example: help

:manpage:

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

Changed in version 7.3: Allow specifying a target with <>, like hyperlinks. For example, :manpage:`blah <1s(1)>` displays blah.

¹⁵⁰ https://docs.python.org/3/library/curses.html#module-curses

:menuselection:

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

Displays: $Start \rightarrow Programs$

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).

For example: text/plain

:newsgroup:

The name of a Usenet newsgroup. For example: comp.lang.python

: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.

For example: curl

:regexp:

A regular expression. Quotes should not be included.

For example: ([abc])+

: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: print(1+variable)

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

Changed in version 1.8: Allowed to escape curly braces with double backslash. For example, in :samp:`print(f"answer=\\{1+{variable}*2\\}")`, the part variable would be emphasized and the escaped curly braces would be displayed: print(f"answer= $\{1+variable*2\}$ ")

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

The following roles generate external links:

:cve:

A reference to a Common Vulnerabilities and Exposures¹⁵¹ record. This generates appropriate index entries. The text "CVE *number*" is generated; with a link to an online copy of the specified CVE. You can link to a specific section by using :cve: `number#anchor`.

For example: CVE 2020-10735¹⁵²

Added in version 8.1.

¹⁵¹ https://www.cve.org/

¹⁵² https://www.cve.org/CVERecord?id=CVE-2020-10735

:cwe:

A reference to a Common Weakness Enumeration¹⁵³. This generates appropriate index entries. The text "CWE *number*" is generated; in the HTML output, with a link to an online copy of the specified CWE. You can link to a specific section by using :cwe: `number#anchor`.

For example: CWE 787¹⁵⁴

Added in version 8.1.

: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`.

For example: PEP 8¹⁵⁵

: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`.

For example: RFC 2324¹⁵⁶

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

Substitutions

The documentation system provides some 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.

|translation progress|

Replaced by the translation progress of the document. This substitution is intended for use by document translators as a marker for the translation progress of the document.

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.

¹⁵³ https://cwe.mitre.org/

¹⁵⁴ https://cwe.mitre.org/data/definitions/787.html

¹⁵⁵ https://peps.python.org/pep-0008/

¹⁵⁶ https://datatracker.ietf.org/doc/html/rfc2324.html



See also

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

Table of contents

Since reStructuredText 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 157 directive.



Note

To create table of contents for current document (.rst file), use the standard reStructuredText contents directive 158.

.. 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. 194

The representation of "TOC tree" is changed in each output format. The builders that output multiple files (e.g. HTML) treat it as a collection of hyperlinks. On the other hand, the builders that output a single file (e.g. 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):

```
.. 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

¹⁵⁷ https://docutils.sourceforge.io/docs/ref/rst/directives.html#include

¹⁵⁸ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents

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 reStructuredText hyperlinks (and Sphinx's *cross-referencing syntax*). This looks like:

```
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.

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.

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 "root document" (selected by $root_doc$) 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 0.6: Added support for external links and "self" references.

Options

:class: class names (a list of class names, separated by spaces)

Assign class attributes ¹⁵⁹. This is a common option ¹⁶⁰. For example:

```
.. toctree::
:class: custom-toc
```

Added in version 7.4.

:name: label (text)

An implicit target name that can be referenced using ref. This is a common option 161 . For example:

```
.. toctree::
   :name: mastertoc
   foo
```

Added in version 1.3.

:caption: (text)

Add a caption to the toctree. For example:

```
.. toctree::
    :caption: Table of Contents
    foo
```

Added in version 1.3.

:numbered:

:numbered: depth

If you want to have section numbers even in HTML output, add the :numbered: option to the *top-level* toctree. For example:

```
.. toctree::
   :numbered:

foo
bar
```

Section 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.

Added in version 0.6.

Changed in version 1.1: Added the numeric *depth* argument.

:titlesonly:

Only list document titles, not other headings of the same level. For example:

```
.. toctree::
    :titlesonly:
    foo
    bar
```

Added in version 1.0.

:glob:

Parse glob wildcards in toctree entries. All entries are matched against the list of available documents, and matches are inserted into the list alphabetically. For 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.)¹⁹⁵

Added in version 0.3.

:reversed:

Reverse the order of the entries in the list. This is particularly useful when using the :glob: option.

Added in version 1.5.

:hidden:

A hidden toctree only defines the document hierarchy. It will not insert links into the document at the location of the directive.

This makes sense if you have other means of navigation, e.g. through manual links, HTML sidebar navigation, or if you use the :includehidden: option on the top-level toctree.

Added in version 0.6.

:includehidden:

If you want one global table of contents showing the complete document structure, you can add the :includehidden: option to the *top-level* toctree directive. All other toctrees on child pages can then be made invisible with the :hidden: option. The top-level toctree with :includehidden: will then include their entries.

Added in version 1.2.

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

This is used for the general index, which is populated with entries from *index* directives and all index-generating *object descriptions*. For example, see Sphinx's genindex.

modindex

This is used for the Python module index, which contains one entry per *py:module* directive. For example, see Sphinx's py-modindex.

• search

This is used for the search page, which contains a form that uses the generated JSON search index and JavaScript to full-text search the generated documents for search words; it works on every major browser. For example, see Sphinx's search.

• 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.

Paragraph-level markup

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

¹⁹⁴ The LaTeX writer only refers the maxdepth option of first toctree directive in the document.

¹⁵⁹ https://docutils.sourceforge.io/docs/ref/doctree.html#classes

¹⁶⁰ https://docutils.sourceforge.io/docs/ref/rst/directives.html#common-options

¹⁶¹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#common-options

¹⁹⁵ 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.

Admonitions, messages, and warnings

The admonition directives create 'admonition' elements, a standardised system of communicating different types of information, from a helpful tip to matters of paramount danger. These directives can be used anywhere an ordinary body element can, and can contain arbitrary body elements. There are nine specific named admonitions and the generic admonition directive.

.. attention::

Information that requires the reader's attention. The content of the directive should be written in complete sentences and include all appropriate punctuation.

Example:



A Attention

Please may I have your attention.

.. caution::

Information with regard to which the reader should exercise care. The content of the directive should be written in complete sentences and include all appropriate punctuation.

Example:



Caution

Exercise due caution.

.. danger::

Information which may lead to near and present danger if not heeded. The content of the directive should be written in complete sentences and include all appropriate punctuation.

Example:



Danger

Let none think to fly the danger for soon or late love is his own avenger.

.. error::

Information relating to failure modes of some description. The content of the directive should be written in complete sentences and include all appropriate punctuation.

Example:



Error

ERROR 418: I'm a teapot.

.. hint::

Information that is helpful to the reader. The content of the directive should be written in complete sentences and include all appropriate punctuation.

Example:



Look under the flowerpot.

.. important::

Information that is of paramount importance and which the reader must not ignore. The content of the directive should be written in complete sentences and include all appropriate punctuation.

Example:

Important

This is a statement of paramount importance.

.. note::

An especially important bit of information that the reader should know. The content of the directive should be written in complete sentences and include all appropriate punctuation.

Example:



This function is not suitable for sending tins of spam.

.. tip::

Some useful tidbit of information for the reader. The content of the directive should be written in complete sentences and include all appropriate punctuation.

Example:



Remember your sun cream!

.. warning::

An important bit of information that the reader should be very aware of. The content of the directive should be written in complete sentences and include all appropriate punctuation.

Example:



Warning

Beware of the dog.

.. admonition:: title

A generic admonition, with an optional title. The content of the directive should be written in complete sentences and include all appropriate punctuation.

Example:

1 This is a title

This is the content of the admonition.

.. 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. The content of the *seealso* directive should be either a single line or a reStructuredText definition list¹⁶².

Example:

```
.. seealso::
    Python's :py:mod:`zipfile` module
        Documentation of Python's standard :py:mod:`zipfile` module.

`GNU tar manual, Basic Tar Format <a href="https://example.org">https://example.org</a>`_
        Documentation for tar archive files, including GNU tar extensions.
```

See also

Module zipfile¹⁶³

Documentation of the zipfile 164 standard module.

GNU tar manual, Basic Tar Format¹⁶⁵

Documentation for tar archive files, including GNU tar extensions.

Describing changes between versions

.. versionadded:: version [brief explanation]

This directive documents the version of the project which added the described feature. When this applies to an entire module or component, it should be placed at the top of the relevant 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.

A Attention

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

Example:

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

 $^{^{162}\} https://docutils.sourceforge.io/docs/ref/rst/restructured text.html \# definition-lists$

¹⁶³ https://docs.python.org/3/library/zipfile.html#module-zipfile

¹⁶⁴ https://docs.python.org/3/library/zipfile.html#module-zipfile

¹⁶⁵ https://example.org

Added in version 2.5: The *spam* parameter.

.. versionchanged:: version [brief explanation]

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

Example:

```
.. versionchanged:: 2.8
The *spam* parameter is now of type *boson*.
```

Changed in version 2.8: The *spam* parameter is now of type *boson*.

.. deprecated:: version [brief explanation]

Similar to *versionadded*, but describes when the feature was deprecated. A *brief* explanation can also be given, for example to tell the reader what to use instead.

Example:

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

Deprecated since version 3.1: Use spam() instead.

.. versionremoved:: version [brief explanation]

Similar to *versionadded*, but describes when the feature was removed. An explanation may be provided to tell the reader what to use instead, or why the feature was removed.

Added in version 7.3.

Example:

```
.. versionremoved:: 4.0

The :py:func:`spam` function is more flexible, and should be used instead.
```

Removed in version 4.0: The spam() function is more flexible, and should be used instead.

Presentational

.. rubric:: title

A rubric is like an informal heading that doesn't correspond to the document's structure, i.e. it does not create a table of contents node.



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.

Options

```
class: class names (a list of class names, separated by spaces)
```

Assign class attributes 166. This is a common option 167.

:name: label (text)

An implicit target name that can be referenced using ref. This is a common option 168 .

:heading-level: n (number from 1 to 6)

Added in version 7.4.1.

Use this option to specify the heading level of the rubric. In this case the rubric will be rendered as <h1> to <h6> for HTML output, or as the corresponding non-numbered sectioning command for LaTeX (see latex_toplevel_sectioning).

.. centered::

This directive creates a centered boldfaced line of text.

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.

Options

:columns: n (int)

The number of columns; defaults to 2. For example:

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

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

Added in version 0.6.

Showing code examples

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

- using reStructuredText doctest blocks;
- using reStructuredText 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 Pygments¹⁶⁹. 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

 $^{^{166}\} https://docutils.sourceforge.io/docs/ref/doctree.html\#classes$

¹⁶⁷ https://docutils.sourceforge.io/docs/ref/rst/directives.html#common-options

¹⁶⁸ https://docutils.sourceforge.io/docs/ref/rst/directives.html#common-options

¹⁶⁹ https://pygments.org

is used. This defaults to python but can be configured using the <code>highlight_language</code> 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 170

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
```

:force: (no value)

If given, minor errors on highlighting are ignored.

Added in version 2.1.

```
.. code-block:: [language]
.. sourcecode:: [language]
```

¹⁷⁰ https://pygments.org/docs/lexers

.. code:: [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 Pygments¹⁷¹. If it is not given, the setting of *highlight* directive will be used. If not set, *highlight_language* will be used. To display a code example *inline* within other text, rather than as a separate block, you can use the *code* role instead.

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.
```

Added 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.')
```

Added in version 1.1.

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

:force: (no value)

Ignore minor errors on highlighting.

Added in version 2.1.

:caption: caption of code block (text)

Set a caption to the code block.

Added 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.
```

Added in version 1.3.

:class: class names (a list of class names separated by spaces)

Assign class attributes 172 . This is a common option 173 .

Added in version 1.4.

: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()¹⁷⁴. For example:

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

    some ruby code
```

Added in version 1.3.

Changed in version 3.5: Support automatic dedent.

.. 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. For example, to include the Python source file example.py, use:

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

¹⁷¹ https://pygments.org/docs/lexers/

¹⁷² https://docutils.sourceforge.io/docs/ref/doctree.html#classes

¹⁷³ https://docutils.sourceforge.io/docs/ref/rst/directives.html#common-options

¹⁷⁴ https://docs.python.org/3/library/textwrap.html#textwrap.dedent

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.

General options

:class: class names (a list of class names, separated by spaces)

Assign class attributes¹⁷⁵. This is a common option¹⁷⁶.

Added in version 1.4.

:name: label (text)

An implicit target name that can be referenced using ref. This is a common option 177 .

Added in version 1.3.

:caption: caption (text)

Add a caption above the included content. If no argument is given, the filename is used as the caption.

Added in version 1.3.

Options for formatting

:dedent: number (number or no value)

Strip indentation characters from the included content. When a number is given, the leading N characters are removed. When no argument given, all common leading indentation is removed (using textwrap. $dedent()^{178}$).

Added in version 1.3.

Changed in version 3.5: Support automatic dedent.

:tab-width: N (number)

Expand tabs to N spaces.

Added in version 1.0.

:encoding: (text)

Explicitly specify the encoding of the file. This overwrites the default encoding (*source_encoding*). For example:

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

Added in version 0.4.3.

:linenos: (no value)

Show line numbers alongside the included content. By default, line numbers are counted from 1. This can be changed by the options :lineno-start: and :lineno-match:.

:lineno-start: number (number)

Set line number for the first line to show. If given, this automatically activates :linenos:.

:lineno-match:

When including only parts of a file and show the original line numbers. This is only allowed only when the selection consists of contiguous lines.

Added in version 1.3.

:emphasize-lines: line numbers (comma separated numbers)

Emphasise particular lines of the included content. For example:

```
.. literalinclude:: example.txt
:emphasize-lines: 1,2,4-6
```

:language: language (text)

Select a highlighting language (Pygments lexer¹⁷⁹) different from the current file's standard language (set by *highlight* or *highlight_language*).

:force: (no value)

Ignore minor errors in highlighting.

Added in version 2.1.

Options for selecting the content to include

```
:pyobject: object (text)
```

For Python files, only include the specified class, function or method:

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

Added in version 0.6.

:lines: line numbers (comma separated numbers)

Specify exactly which lines to include:

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

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

Added in version 0.6.

:start-at: text
:start-after: text
:end-before: text
:end-at: text

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

Added in version 0.6: Added the start-after and end-before options.

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

4.1. reStructuredText 61

Tip

To only select [second-section] of an INI file such as the following, 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

These options can be useful when working with tag comments. Using :start-after: [initialise] and :end-before: [initialised] keeps the lines between between the two comments below:

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

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

:prepend: line (text)

Prepend a line before the included code. This can be useful for example when highlighting PHP code that doesn't include the <?php or ?> markers.

Added in version 1.0.

:append: line (text)

Append a line after the included code. This can be useful for example when highlighting PHP code that doesn't include the <?php or ?> markers.

Added in version 1.0.

:diff: filename

Show the diff of two files. For example:

```
.. literalinclude:: example.txt
:diff: example.txt.orig
```

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

Added in version 1.3.

Changed in version 0.6: Added support for absolute filenames.

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.

¹⁹⁶ There is a standard . . include directive, but it raises errors if the file is not found. This one only emits a warning.

 $^{^{175}\} https://docutils.sourceforge.io/docs/ref/doctree.html\#classes$

¹⁷⁶ https://docutils.sourceforge.io/docs/ref/rst/directives.html#common-options

¹⁷⁷ https://docutils.sourceforge.io/docs/ref/rst/directives.html#common-options

¹⁷⁸ https://docs.python.org/3/library/textwrap.html#textwrap.dedent

¹⁷⁹ https://pygments.org/docs/lexers/

Glossary

.. glossary::

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

```
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
   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:

```
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.

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*.

Options

:sorted:

Sort the entries alphabetically.

Added in version 0.6.

Changed in version 4.4: In internationalized documentation, sort according to translated terms.

4.1. reStructuredText 63

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:

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 sub-entry by separating the sub-entry text with a semicolon (this notation is also used below to describe what entries are created). Examples:

```
.. index:: single: execution single: execution; context
```

• single: execution creates an index entry labelled execution.

• single: execution; context creates an sub-entry of execution labelled context.

pair

A shortcut to create two index entries. The pair of values must be separated by a semicolon. Example:

```
.. index:: pair: loop; statement
```

This would create two index entries; loop; statement and statement; loop.

triple

A shortcut to create three index entries. All three values must be separated by a semicolon. Example:

```
.. index:: triple: module; search; path
```

This would create three index entries; module; search path, search; path, module, and path; module search.

see

A shortcut to create an index entry that refers to another entry. Example:

```
.. index:: see: entry; other
```

This would create an index entry referring from entry to other (i.e. 'entry': See 'other').

seealso

Like see, but inserts 'see also' instead of 'see'.

module, keyword, operator, object, exception, statement, builtin

These **deprecated** shortcuts all create two index entries. For example, module: hashlib creates the entries module; hashlib and hashlib; module.

Deprecated since version 1.0: These Python-specific entry types are deprecated.

Changed in version 7.1: Removal version set to Sphinx 9.0. Using these entry types will now emit warnings with the index category.

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.

4.1. reStructuredText 65

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
```

Added 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 reStructuredText :index: `paragraph` that contains several
:index: index entries <pair: index; entry> `.
```

Added 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 are true (tags can be defined via the --tag command-line option or within conf.py, see here). Boolean expressions (like (latex or html) and draft) are supported and may use parentheses.

The format and the name of the current builder (html, latex or text) are always set as a tag 197. 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 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.

Added 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

¹⁹⁷ 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.

¹⁸⁰ https://docs.python.org/3/reference/lexical_analysis.html#identifiers

Tables

Use reStructuredText tables, i.e. either

- grid table syntax (ref¹⁸¹),
- simple table syntax (ref¹⁸²),
- csv-table¹⁸³ syntax,
- or list-table 184 syntax.

The table 185 directive serves as optional wrapper of the *grid* and *simple* syntaxes.

They work fine in HTML output, but rendering tables to LaTeX is complex. Check the latex_table_style.

Changed in version 1.6: Merged cells (multi-row, multi-column, both) from grid tables containing complex contents such as multiple paragraphs, blockquotes, lists, literal blocks, will render correctly to LaTeX output.

.. tabularcolumns:: column spec

This directive influences only the LaTeX output for the next table in source. The mandatory argument is a column specification (known as an "alignment preamble" in LaTeX idiom). Please refer to a LaTeX documentation, such as the wiki page ¹⁸⁶, for basics of such a column specification.

Added in version 0.3.



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

Sphinx will render tables with more than 30 rows with longtable. Besides the 1, r, c and p{width} column specifiers, one can also use $X{a}{b}$ (new in version 1.5) which configures the column width to be a fraction a/b of the total line width and $Y{f}$ (new in version 1.6) where f is a decimal: for example $Y{0.2}$ means that the column will occupy 0.2 times the line width.

When this directive is used for a table with at most 30 rows, Sphinx will render it with tabulary. One can then use specific column types L (left), R (right), C (centered) and J (justified). They have the effect of a p{width} (i.e. each cell is a LaTeX \parbox) with the specified internal text alignment and an automatically computed width.

A

Warning

- Cells that contain list-like elements such as object descriptions, blockquotes or any kind of lists are not compatible with the LRCJ column types. The column type must then be some p{width} with an explicit width (or \X{a}{b} or \Y{f}).
- Literal blocks do not work with tabulary at all. Sphinx will fall back to tabular or longtable environments and generate a suitable column specification.

In absence of the *tabularcolumns* directive, and for a table with at most 30 rows and no problematic cells as described in the above warning, Sphinx uses tabulary and the J column-type for every column.

4.1. reStructuredText 67

¹⁸¹ https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#grid-tables

¹⁸² https://docutils.sourceforge.io/docs/ref/rst/restructuredtext.html#simple-tables

¹⁸³ https://docutils.sourceforge.io/docs/ref/rst/directives.html#csv-table

 $^{^{184}\} https://docutils.sourceforge.io/docs/ref/rst/directives.html \# list-table$

¹⁸⁵ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table

¹⁸⁶ https://en.wikibooks.org/wiki/LaTeX/Tables

Changed in version 1.6: Formerly, the L column-type was used (text is flushed-left). To revert to this, include $\$ newcolumntype{T}{L} in the LaTeX preamble, as in fact Sphinx uses T and sets it by default to be an alias of J.



A frequent issue with tabulary is that columns with little contents appear to be "squeezed". One can add to the LaTeX preamble for example \setlength{\tymin}{40pt} to ensure a minimal column width of 40pt, the tabulary default of 10pt being too small.

Hint

To force usage of the LaTeX longtable environment pass longtable as a :class: option to table ¹⁸⁷, csv-table ¹⁸⁸, or list-table ¹⁸⁹. Use *rst-class* for other tables.

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 \\:

```
.. math::

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

For more details, look into the documentation of the AmSMath LaTeX package¹⁹⁰.

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
```

¹⁸⁷ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table

¹⁸⁸ https://docutils.sourceforge.io/docs/ref/rst/directives.html#csv-table

¹⁸⁹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#list-table

Options

:class: class names (a list of class names, separated by spaces)

Assign class attributes¹⁹¹. This is a common option¹⁹².

:name: label (text)

An implicit target name that can be referenced using ref. This is a common option ¹⁹³.

:label: label (text)

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.

:nowrap:

Prevent 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*.

4.1. reStructuredText 69

 $^{^{190}\} https://www.ams.org/publications/authors/tex/amslatex$

¹⁹¹ https://docutils.sourceforge.io/docs/ref/doctree.html#classes

¹⁹² https://docutils.sourceforge.io/docs/ref/rst/directives.html#common-options

¹⁹³ https://docutils.sourceforge.io/docs/ref/rst/directives.html#common-options

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 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 reStructuredText 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:

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.



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.



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



This metadata affects the depth of the local toctree. But it does not affect the depth of the *global* toctree. So this does not change the sidebar of themes that use the global toctree.

Added 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:

Added in version 1.0.

nosearch

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

:nosearch:

1 Note

object search is still available even if nosearch option is set.

Added in version 3.0.

MOVED: Domains

MOVED: Basic Markup

See Domains.

MOVED: Python Domain

See The Python Domain.

4.1. reStructuredText 71

MOVED: C Domain

See The C Domain.

MOVED: C++ Domain

See The C++ Domain.

MOVED: Standard Domain

See The Standard Domain.

MOVED: JavaScript Domain

See The JavaScript Domain.

MOVED: reStructuredText Domain

See The reStructuredText Domain.

MOVED: Math Domain

See The Mathematics Domain.

MOVED: More domains

See Domains.

4.2 Markdown

Markdown¹⁹⁸ 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 MyST-Parser¹⁹⁹. MyST-Parser is a Docutils bridge to markdown-it-py²⁰⁰, a Python package for parsing the CommonMark²⁰¹ Markdown flavor.

Configuration

To configure your Sphinx project for Markdown support, proceed as follows:

1. Install the Markdown parser *MyST-Parser*:

```
pip install --upgrade myst-parser
```

2. Add *myst_parser* to the list of configured extensions:

```
extensions = ['myst_parser']
```



MyST-Parser requires Sphinx 2.1 or newer.

¹⁹⁸ https://daringfireball.net/projects/markdown/

¹⁹⁹ https://myst-parser.readthedocs.io/en/latest/

²⁰⁰ https://github.com/executablebooks/markdown-it-py

²⁰¹ https://commonmark.org/

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:

```
source_suffix = {
    '.rst': 'restructuredtext',
    '.txt': 'markdown',
    '.md': 'markdown',
}
```

4. You can further configure *MyST-Parser* to allow custom syntax that standard *CommonMark* doesn't support. Read more in the MyST-Parser documentation²⁰².

4.3 Cross-references

One of Sphinx's most useful features is creating automatic cross-references through semantic cross-referencing roles. A cross reference to an object description, such as :func:`spam`, will create a link to the place where spam() is documented, appropriate to each output format (HTML, PDF, ePUB, etc.).

Syntax

Sphinx supports various cross-referencing roles to create links to other elements in the documentation. In general, writing :role: `target` creates a link to the object called *target* of the type indicated by *role*. The link's text depends the role but is often the same as or similar to *target*.

The behavior can be modified in the following ways:

- **Custom link text:** You can specify the link text explicitly using the same notation as in reStructuredText *external links*: :role:`custom text <target>` will refer to *target* and display *custom text* as the text of the link.
- **Suppressed link:** Prefixing with an exclamation mark (!) prevents the creation of a link but otherwise keeps the visual output of the role.

For example, writing :py:func:`!target` displays target(), with no link generated.

This is helpful for cases in which the link target does not exist; e.g. changelog entries that describe removed functionality, or third-party libraries that don't support *intersphinx*. Suppressing the link prevents warnings in *nitpicky* mode.

Modified domain reference: When referencing domain objects, a tilde ~ prefix shortens the link text to 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.

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 objects

These roles are described with their respective domains:

- C
- C++
- JavaScript
- reStructuredText
- Python

4.3. Cross-references 73

 $^{^{202}\} https://myst-parser.readthedocs.io/en/latest/using/syntax-optional.html$

Cross-referencing arbitrary locations

:ref:

To support cross-referencing to arbitrary locations in any document, the standard reStructuredText 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 table²⁰³ 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>`.



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

Added in version 0.6.

There is also a way to directly link to documents:

 $^{^{203}\} https://docutils.sourceforge.io/docs/ref/rst/directives.html\#table$

: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

Added in version 0.6.

:download:

This role lets you link to files within your source tree that are not reStructuredText 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>`.
```

Cross-referencing figures by figure number

Added 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 reStructuredText 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:

:confval:

A configuration value or setting. Index entries are generated. Also generates a link to the matching *confval* directive, if it exists.

4.3. Cross-references 75

: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.

Cross-referencing anything

:any:

Added 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.

4.4 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 customise Sphinx input and output behaviour.

An optional file docutils. $conf^{204}$ can be added to the configuration directory to adjust $Docutils^{205}$ configuration if not otherwise overridden or set by Sphinx.

Important points to note:

- If not otherwise documented, values must be strings, and their default is the empty string.
- The term "fully-qualified name" (FQN) refers to a string that names an importable Python object inside a module; for example, the fully-qualified name "sphinx.builders.Builder" means the Builder class in the sphinx.builders module.
- Document names use / as the path separator and do not contain the file name extension.
- Where glob-style patterns are permitted, you can use the standard shell constructs (*, ?, [...], and [!...]) with the feature that none of these will match slashes (/). A double star ** can be used to match any sequence of characters *including* slashes.



The configuration file is executed as Python code at build time (using importlib.import_module() 206 , with the current directory set to the *configuration directory*), and therefore can execute arbitrarily complex code.

Sphinx then reads simple names from the file's namespace as its configuration. In general, configuration values should be simple strings, numbers, or lists or dictionaries of simple values.

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 deleting imported modules is not needed.

Project tags

There is a special object named tags available in the config file, which exposes the *project tags*. Tags are defined either via the --tag command-line option or tags.add('tag'). Note that the builder's name and format tags are not available in conf.py.

It can be used to query and change the defined tags as follows:

- To query whether a tag is set, use 'tag' in tags.
- To add a tag, use tags.add('tag').
- To remove a tag, use tags.remove('tag').

Project information

project

Type

 $^{^{204}\} https://docutils.sourceforge.io/docs/user/config.html$

²⁰⁵ https://docutils.sourceforge.io/

²⁰⁶ https://docs.python.org/3/library/importlib.html#importlib.import_module

Default

```
'Project name not set'
```

The documented project's name. Example:

```
project = 'Thermidor'
```

author

```
Type
str

Default
```

'Author name not set'

```
The project's author(s). Example:
```

```
author = 'Joe Bloggs'
```

copyright

project_copyright

```
Type
str | Sequence[str]

Default
```

A copyright statement. Permitted styles are as follows, where 'YYYY' represents a four-digit year.

```
    copyright = 'YYYY'
    copyright = 'YYYY, Author Name'
    copyright = 'YYYY Author Name'
    copyright = 'YYYY-YYYY, Author Name'
    copyright = 'YYYY-YYYY Author Name'
```

If the string '%Y' appears in a copyright line, it will be replaced with the current four-digit year. For example:

```
copyright = '%Y'
copyright = '%Y, Author Name'
copyright = 'YYYY-%Y, Author Name'
```

Added in version 3.5: The project_copyright alias.

Changed in version 7.1: The value may now be a sequence of copyright statements in the above form, which will be displayed each to their own line.

Changed in version 8.1: Copyright statements support the '%Y' placeholder.

version

```
Type
str
Default
```

The major project version, used as the replacement for the |version| default substitution.

This may be something like version = '4.2'. The full project version is defined in release.

If your project does not draw a meaningful distinction between between a 'full' and 'major' version, set both version and release to the same value.

release

```
Type
str
Default
```

The full project version, used as the replacement for the |release| *default substitution*, and e.g. in the HTML templates.

This may be something like release = '4.2.1b0'. The major (short) project version is defined in *version*.

If your project does not draw a meaningful distinction between between a 'full' and 'major' version, set both version and release to the same value.

General configuration

needs_sphinx

```
Type
str
Default
```

Set a minimum supported version of Sphinx required to build the project. The format should be a 'major. minor' version string like '1.1' Sphinx will compare it with its version and refuse to build the project if the running version of Sphinx is too old. By default, there is no minimum version.

Added in version 1.0.

Changed in version 1.4: Allow a 'major.minor.micro' version string.

extensions

```
Type
list[str]

Default
[]
```

A list of strings that are module names of *Sphinx extensions*. These can be extensions bundled with Sphinx (named sphinx.ext.*) or custom first-party or third-party extensions.

To use a third-party extension, you must ensure that it is installed and include it in the extensions list, like so:

There are two options for first-party extensions. The configuration file itself can be an extension; for that, you only need to provide a setup() function in it. Otherwise, you must ensure that your custom extension is importable, and located in a directory that is in the Python path.

Ensure that absolute paths are used when modifying sys.path 207 . If your custom extensions live in a directory that is relative to the *configuration directory*, use os.path.abspath() 208 like so:

The directory structure illustrated above would look like this:

needs_extensions

```
Type
    dict[str, str]
Default
{}
```

If set, this value must be a dictionary specifying version requirements for extensions in *extensions*. The version strings should be in the 'major.minor' form. Requirements do not have to be specified for all extensions, only for those you want to check. Example:

```
needs_extensions = {
    'sphinxcontrib.something': '1.5',
}
```

This requires that the extension declares its version in the setup() function. See *Sphinx API* for further details.

Added in version 1.3.

manpages_url

```
Type str
Default
```

A URL to cross-reference *manpage* roles. 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.

 $^{^{207}\} https://docs.python.org/3/library/sys.html \#sys.path$

 $^{^{208}\} https://docs.python.org/3/library/os.path.html\#os.path.abspath$

```
# To use manpages.debian.org:
manpages_url = 'https://manpages.debian.org/{path}'
# To use man7.org:
manpages_url = 'https://man7.org/linux/man-pages/man{section}/{page}.{section}.html'
# To use linux.die.net:
manpages_url = 'https://linux.die.net/man/{section}/{page}'
# To use helpmanual.io:
manpages_url = 'https://helpmanual.io/man{section}/{page}'
```

Added in version 1.7.

today

today_fmt

These values determine how to format the current date, used as the replacement for the |today| default substitution.

- If you set today to a non-empty value, it is used.
- Otherwise, the current time is formatted using time.strftime()²⁰⁹ and the format given in today_fmt.

The default for today is '', and the default for today_fmt is '%b %d, %Y' (or, if translation is enabled with language, an equivalent format for the selected locale).

Options for figure numbering

numfig

Type bool

Default

False

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.



The LaTeX builder always assigns numbers whether this option is enabled or not.

Added in version 1.3.

numfig_format

```
Type
    dict[str, str]

Default
    {}
```

A dictionary mapping 'figure', 'table', 'code-block' and 'section' to strings that are used for format of figure numbers. The marker %s will be replaced with the figure number.

The defaults are:

 $^{^{209}\} https://docs.python.org/3/library/time.html\#time.strftime$

```
numfig_format = {
    'code-block': 'Listing %s',
    'figure': 'Fig. %s',
    'section': 'Section',
    'table': 'Table %s',
}
```

Added in version 1.3.

numfig_secnum_depth

```
Type int
Default
```

- If set to 0, figures, tables, and code-blocks are continuously numbered starting at 1.
- If 1, the numbering will be x.1, x.2, ... with x representing the section number. (If there is no top-level section, the prefix will not be added). This naturally applies only if section numbering has been activated via the :numbered: option of the toctree directive.
- If 2, the numbering will be x.y.1, x.y.2, ... with x representing the section number and y the sub-section number. If located directly under a section, there will be no y. prefix, and if there is no top-level section, the prefix will not be added.
- Any other positive integer can be used, following the rules above.

Added in version 1.3.

Changed in version 1.7: The LaTeX builder obeys this setting if *numfig* is set to **True**.

Options for highlighting

highlight_language

```
Type
str

Default
'default'
```

The default language to highlight source code in. The default is 'default', which suppresses warnings if highlighting as Python code fails.

The value should be a valid Pygments lexer name, see *Showing code examples* for more details.

Added in version 0.5.

Changed in version 1.4: The default is now 'default'.

highlight_options

```
Type
    dict[str, dict[str, Any]]
Default
{}
```

A dictionary that maps Pygments lexer names to their options. These are lexer-specific; for the options understood by each, see the Pygments documentation²¹⁰.

Example:

```
highlight_options = {
  'default': {'stripall': True},
  'php': {'startinline': True},
}
```

Added in version 1.3.

Changed in version 3.5: Allow configuring highlight options for multiple lexers.

pygments_style

```
Type
str

Default
'sphinx'
```

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.

Options for HTTP requests

tls_verify

```
Type
bool
Default
```

True

If True, Sphinx verifies server certificates.

Added in version 1.5.

tls_cacerts

```
Type
    str | dict[str, str]
Default
''
```

A path to a certification file of CA or a path to directory which contains the certificates. This also allows a dictionary mapping hostnames to the certificate file path. The certificates are used to verify server certifications.

Added in version 1.5.

```
🗘 Tip
```

Sphinx uses requests²¹¹ as a HTTP library internally. If tls_cacerts is not set, Sphinx falls back to requests' default behaviour. See SSL Cert Verification²¹² for further details.

²¹⁰ https://pygments.org/docs/lexers

user_agent

```
Type
    str

Default
    'Mozilla/5.0 (X11; Linux x86_64; rv:100.0) Gecko/20100101 Firefox/100.0
    Sphinx/X.Y.Z'
```

Set the User-Agent used by Sphinx for HTTP requests.

Added in version 2.3.

Options for internationalisation

These options influence Sphinx's Native Language Support. See the documentation on Internationalization for details.

language

```
Type
str

Default
'en'
```

The code for the language the documents 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.

Added in version 0.5.

Changed in version 1.4: Support figure substitution

Changed in version 5.0: The default is now 'en' (previously **None**).

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 (default)

²¹¹ https://requests.readthedocs.io/

²¹² https://requests.readthedocs.io/en/latest/user/advanced/#verification

- 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
- 1t Lithuanian
- 1v 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
- s1 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

```
Type
list[str]

Default
['locale']
```

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 gettext²¹³ 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 -v option to sphinx-build is useful to check the locale_dirs setting is working as expected. If the message catalog directory not found, debug messages are emitted.

Added in version 0.5.

Changed in version 1.5: Use locales directory as a default value

gettext_allow_fuzzy_translations

```
Type
bool
Default
False
```

If True, "fuzzy" messages in the message catalogs are used for translation.

Added in version 4.3.

gettext_compact

```
Type
bool | str

Default
True
```

 $^{^{213}\} https://docs.python.org/3/library/gettext.html\#module-gettext$

- 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 **False**, a document's text domain is the docname, in full.
- If set to a string, every document's text domain is set to this string, making all documents use single text domain.

With gettext_compact = **True**, the document markup/code.rst ends up in the markup text domain. With this option set to **False**, it is markup/code. With this option set to 'sample', it is sample.

Added in version 1.1.

Changed in version 3.3: Allow string values.

gettext_uuid

Type

bool

Default

False

If True, Sphinx generates UUID information for version tracking in message catalogs. It is used to:

- Add a UUID line for each *msgid* in .pot files.
- Calculate similarity between new msgids and previously saved old msgids. (This calculation can take a long time.)



If you want to accelerate the calculation, you can use a third-party package (Levenshtein²¹⁴) by running **pip** install levenshtein.

Added in version 1.3.

gettext_location

Type

bool

Default

True

If **True**, Sphinx generates location information for messages in message catalogs.

Added in version 1.3.

gettext_auto_build

Type

bool

Default

True

If **True**, Sphinx builds a .mo file for each translation catalog file.

Added in version 1.3.

²¹⁴ https://pypi.org/project/Levenshtein/

gettext_additional_targets

```
Type
    set[str] | Sequence[str]
Default
[]
```

Enable gettext translation for certain element types. Example:

```
gettext_additional_targets = {'literal-block', 'image'}
```

The following element types are supported:

- 'index' index terms
- 'literal-block' literal blocks (:: annotation and code-block directive)
- 'doctest-block' doctest block
- 'raw' raw content
- 'image' image/figure uri

Added in version 1.3.

Changed in version 4.0: The alt text for images is translated by default.

Changed in version 7.4: Permit and prefer a set type.

figure_language_filename

```
Type
    str

Default
    '{root}.{language}{ext}'
```

The filename format for language-specific figures. The available format tokens are:

- {root}: the filename, including any path component, without the file extension. For example: images/filename.
- {path}: the directory path component of the filename, with a trailing slash if non-empty. For example: images/.
- {basename}: the filename without the directory path or file extension components. For example: filename.
- {ext}: the file extension. For example: .png.
- ${language}$: the translation language. For example: en.
- {docpath}: the directory path component for the current document, with a trailing slash if non-empty. For example: dirname/.

By default, an image directive .. image:: images/filename.png, using an image at images/filename.png, will use the language-specific figure filename images/filename.en.png.

If figure_language_filename is set as below, the language-specific figure filename will be images/en/filename.png instead.

```
figure_language_filename = '{path}{language}/{basename}{ext}'
```

Added in version 1.4.

Changed in version 1.5: Added {path} and {basename} tokens.

Changed in version 3.2: Added {docpath} token.

translation_progress_classes

```
Type
    bool | 'translated' | 'untranslated'
Default
    False
```

Control which, if any, classes are added to indicate translation progress. This setting would likely only be used by translators of documentation, in order to quickly indicate translated and untranslated content.

True

Add translated and untranslated classes to all nodes with translatable content.

'translated'

Only add the translated class.

'untranslated'

Only add the untranslated class.

False

Do not add any classes to indicate translation progress.

Added in version 7.1.

Options for markup

default_role

Type

str

Default

None

The name of a reStructuredText 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 role can always be set within individual documents using the standard reStructuredText default-role²¹⁵ directive.

Added in version 0.4.

keep_warnings

Type

bool

Default

False

Keep warnings as "system message" paragraphs in the rendered documents. Warnings are always written to the standard error stream when **sphinx-build** is run, regardless of this setting.

Added in version 0.5.

²¹⁵ https://docutils.sourceforge.io/docs/ref/rst/directives.html#default-role

option_emphasise_placeholders

```
Type
bool
Default
False
```

When enabled, emphasise placeholders in *option* directives. To display literal braces, escape with a backslash ($\$ {). For example, option_emphasise_placeholders=True and .. option:: -foption={TYPE} would render with TYPE emphasised.

Added in version 5.1.

primary_domain

```
Type
str

Default
'py'
```

The name of the default *domain*. Can also be **None** to disable a default domain. The default is 'py', for the Python domain.

Those objects in other domain (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"). Example:

```
primary_domain = 'cpp'
```

Added in version 1.0.

rst_epilog

```
Type
str
Default
```

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*). Example:

```
rst_epilog = """
.. |psf| replace:: Python Software Foundation
"""
```

Added in version 0.6.

rst_prolog

```
Type
str
Default
```

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). Example:

91

```
rst_prolog = """
.. |psf| replace:: Python Software Foundation
"""
```

Added in version 1.0.

show_authors

Type bool

Default

False

A boolean that decides whether *codeauthor* and *sectionauthor* directives produce any output in the built files.

trim_footnote_reference_space

Type
bool
Default

False

Trim spaces before footnote references that are necessary for the reStructuredText parser to recognise the footnote, but do not look too nice in the output.

Added in version 0.6.

Options for Maths

These options control maths markup and notation.

math_eqref_format

```
Type
    str

Default
    '({number})'
```

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.

Added in version 1.7.

math_number_all

Type bool

Default False

Force all displayed equations to be numbered. Example:

```
math_number_all = True
```

Added in version 1.4.

math_numfig

Type bool

Default

True

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.

Added in version 1.7.

math_numsep

```
Type
str
Default
```

A string that defines the separator between section numbers and the equation number when *numfig* is enabled and *numfig_secnum_depth* is positive.

```
Example: '-' gets rendered as 1.2-3.
```

Added in version 7.4.

Options for the nitpicky mode

nitpicky

```
Type
bool
Default
False
```

Enables nitpicky mode if **True**. In nitpicky mode, Sphinx will warn about *all* references where the target cannot be found. This is recommended for new projects as it ensures that all references are to valid targets.

You can activate this mode temporarily using the *--nitpicky* command-line option. See *nitpick_ignore* for a way to mark missing references as "known missing".

```
nitpicky = True
```

Added in version 1.0.

nitpick_ignore

```
Type
    set[tuple[str, str]] | Sequence[tuple[str, str]]
Default
    ()
```

A set or list of (warning_type, target) tuples that should be ignored when generating warnings in "nitpicky mode". Note that warning_type should include the domain name if present. Example:

```
nitpick_ignore = {
    ('py:func', 'int'),
    ('envvar', 'LD_LIBRARY_PATH'),
}
```

Added in version 1.1.

Changed in version 6.2: Changed allowable container types to a set, list, or tuple

nitpick_ignore_regex

```
Type
    set[tuple[str, str]] | Sequence[tuple[str, str]]
Default
    ()
```

An extended version of *nitpick_ignore*, which instead interprets the warning_type and target strings as regular expressions. Note that the regular expression must match the whole string (as if the ^ and \$ markers were inserted).

For example, (r'py:.*', r'foo.*bar\.B.*') will ignore nitpicky warnings for all python entities that start with 'foo' and have 'bar.B' in them, such as ('py:const', 'foo_package.bar.BAZ_VALUE') or ('py:class', 'food.bar.Barman').

Added in version 4.1.

Changed in version 6.2: Changed allowable container types to a set, list, or tuple

Options for object signatures

add_function_parentheses

```
Type
bool
Default
True
```

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.

maximum_signature_line_length

```
Type
int | None

Default
None
```

If a signature's length in characters exceeds the number set, each parameter within the signature will be displayed on an individual logical line.

When None, there is no maximum length and the entire signature will be displayed on a single logical line.

A 'logical line' is similar to a hard line break—builders or themes may choose to 'soft wrap' a single logical line, and this setting does not affect that behaviour.

Domains may provide options to suppress any hard wrapping on an individual object directive, such as seen in the C, C++, and Python domains (e.g. *py:function:single-line-parameter-list*).

Added in version 7.1.

strip_signature_backslash

```
Type
bool

Default
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.

Added in version 3.0.

toc_object_entries

```
Type bool

Default
```

Create table of contents entries for domain objects (e.g. functions, classes, attributes, etc.).

Added in version 5.2.

True

toc_object_entries_show_parents

```
Type
    'domain' | 'hide' | 'all'
Default
    'domain'
```

A string that determines how domain objects (functions, classes, attributes, etc.) are displayed in their table of contents entry.

Use 'domain' to allow the domain to determine the appropriate number of parents to show. For example, the Python domain would show Class.method() and function(), leaving out the module. level of parents.

Use 'hide' to only show the name of the element without any parents (i.e. method()).

Use 'all' to show the fully-qualified name for the object (i.e. module.Class.method()), displaying all parents.

Added in version 5.2.

Options for source files

exclude_patterns

```
Type
Sequence[str]

Default
()
```

A list of *glob-style patterns* that should be excluded when looking for source files. They are matched against the source file names relative to the source directory, using slashes as directory separators on all platforms. *exclude_patterns* has priority over *include_patterns*.

Example patterns:

• 'library/xml.rst' - ignores the library/xml.rst file

```
• 'library/xml' – ignores the library/xml directory
```

- 'library/xml*' ignores all files and directories starting with library/xml
- '**/.git' ignores all .git directories
- 'Thumbs.db' ignores all Thumbs.db files

exclude_patterns is also consulted when looking for static files in html_static_path and html_extra_path.

Added in version 1.0.

include_patterns

```
Type
    Sequence[str]
Default
    ('**',)
```

A list of *glob-style patterns* that are used to find source files. They are matched against the source file names relative to the source directory, using slashes as directory separators on all platforms. By default, all files are recursively included from the source directory. *exclude_patterns* has priority over *include_patterns*.

Example patterns:

- '**' all files in the source directory and subdirectories, recursively
- 'library/xml' just the library/xml directory
- 'library/xml*' all files and directories starting with library/xml
- '**/doc' all doc directories (this might be useful if documentation is co-located with source files)

Added in version 5.1.

```
master_doc
root_doc
Type
str
Default
```

Sphinx builds a tree of documents based on the *toctree* directives contained within the source files. This sets the name of the document containing the master toctree directive, and hence the root of the entire tree. Example:

```
master_doc = 'contents'
```

Changed in version 2.0: Default is 'index' (previously 'contents').

Added in version 4.0: The root_doc alias.

source_encoding

```
Type
str

Default
'utf-8-sig'
```

'index'

The file encoding of all source files. The recommended encoding is 'utf-8-sig'.

Added in version 0.5.

source_suffix

```
Type
    dict[str, str] | Sequence[str] | str

Default
    {'.rst': 'restructuredtext'}
```

A dictionary mapping the file extensions (suffixes) of source files to their file types. Sphinx considers all files files with suffixes in source_suffix to be source files. Example:

```
source_suffix = {
   '.rst': 'restructuredtext',
   '.txt': 'restructuredtext',
   '.md': 'markdown',
}
```

By default, Sphinx only supports the 'restructuredtext' file type. Further file types can be added with extensions that register different source file parsers, such as MyST-Parser²¹⁶. Refer to the extension's documentation to see which file types it supports.

If the value is a string or sequence of strings, Sphinx will consider that they are all 'restructuredtext' files.



File extensions must begin with a dot ('.').

Changed in version 1.3: Support a list of file extensions.

Changed in version 1.8: Change to a map of file extensions to file types.

Options for smart quotes

smartquotes

Type bool

Default

True

If **True**, the Smart Quotes transform²¹⁷ will be used to convert quotation marks and dashes to typographically correct entities.

Added in version 1.6.6: Replaces the now-removed html_use_smartypants. It applies by default to all builders except man and text (see *smartquotes_excludes*.)

1 Note

A docutils.conf²¹⁸ file located in the *configuration directory* (or a global ~/.docutils file) is obeyed unconditionally if it *deactivates* smart quotes via the corresponding Docutils option²¹⁹. But if it *activates* them, then *smartquotes* does prevail.

²¹⁶ https://myst-parser.readthedocs.io/

²¹⁷ https://docutils.sourceforge.io/docs/user/smartquotes.html

²¹⁸ https://docutils.sourceforge.io/docs/user/config.html

²¹⁹ https://docutils.sourceforge.io/docs/user/config.html#smart-quotes

smartquotes_action

```
Type
    str
Default
    'aDe'
```

Customise the Smart Quotes transform. See below for the permitted codes. The default 'qDe' educates normal quote characters ", ', em- and en-Dashes ---, --, and ellipses

```
'a'
     Convert quotation marks
'b'
     Convert backtick quotation marks (``double'' only)
'B'
```

Convert backtick quotation marks (``double'' and `single')

'b'

Convert dashes (convert -- to em-dashes and --- to en-dashes)

'D'

Convert dashes (old school) (convert -- to en-dashes and --- to em-dashes)

'i'

Convert dashes (inverted old school) (convert -- to em-dashes and --- to en-dashes)

161

Convert ellipses ...

'w'

Convert '"' entities to '"'

Added in version 1.6.6.

smartquotes_excludes

```
Type
   dict[str, list[str]]
Default
   {'languages': ['ja'], 'builders': ['man', 'text']}
```

Control when the Smart Quotes transform is disabled. Permitted keys are 'builders' and 'languages', and The values are lists of strings.

Each entry gives a sufficient condition to ignore the *smartquotes* setting and deactivate the Smart Quotes transform. Example:

```
smartquotes_excludes = {
    'languages': ['ja'],
    'builders': ['man', 'text'],
```

1 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 SPHINXOPTS="-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 customise) the smart quotes for a given builder, for example latex, is to use make this way:

```
make latex SPHINXOPTS="-D smartquotes_action="
```

This can follow some make html with no problem, in contrast to the situation from the prior note.

Added in version 1.6.6.

Options for templating

template_bridge

```
Type
str
Default
```

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.) Example:

```
template_bridge = 'module.CustomTemplateBridge'
```

templates_path

```
Type
Sequence[str]

Default
()
```

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*. Example:

```
templates_path = ['.templates']
```

Changed in version 1.3: As these files are not meant to be built, they are automatically excluded when discovering source files.

Options for warning control

show_warning_types

```
Type
bool
```

Default True Add the type of each warning as a suffix to the warning message. For example, WARNING: [...] [index] or WARNING: [...] [toc.circular]. This setting can be useful for determining which warnings types to include in a *suppress_warnings* list.

Added in version 7.3.0.

Changed in version 8.0.0: The default is now **True**.

suppress_warnings

```
Type
Sequence[str]
Default
```

A list of warning codes to suppress arbitrary warning messages.

By default, Sphinx supports the following warning codes:

• app.add_node

()

- app.add_directive
- app.add_role
- app.add_generic_role
- app.add_source_parser
- · config.cache
- docutils
- download.not_readable
- epub.unknown_project_files
- epub.duplicated_toc_entry
- i18n.inconsistent_references
- index
- image.not_readable
- ref.term
- ref.ref
- ref.numref
- ref.keyword
- ref.option
- ref.citation
- ref.footnote
- ref.doc
- ref.python
- misc.copy_overwrite
- misc.highlighting_failure
- toc.circular

- toc.excluded
- toc.no_title
- toc.not_readable
- toc.secnum

Extensions can also define their own warning types. Those defined by the first-party sphinx.ext extensions are:

- autodoc
- autodoc.import_object
- autosectionlabel.<document name>
- autosummary
- autosummary.import_cycle
- intersphinx.external

You can choose from these types. You can also give only the first component to exclude all warnings attached to it.

Added 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. <document name>

Changed in version 3.3.0: Added epub.duplicated_toc_entry

Changed in version 4.3: Added toc.excluded and toc.not_readable

Added in version 4.5: Added i18n.inconsistent_references

Added in version 7.1: Added index.

Added in version 7.3: Added config.cache.

Added in version 7.3: Added toc.no_title.

Added in version 8.0: Added misc.copy_overwrite.

Builder options

Options for HTML output

These options influence HTML output. Various other builders are derived from the HTML output, and also make use of these options.

html_theme

```
Type
str

Default
'alabaster'
```

The theme for HTML output. See the HTML theming section.

Added in version 0.6.

Changed in version 1.3: The default theme is now 'alabaster'.

html_theme_options

```
Type
    dict[str, Any]
Default
    {}
```

A dictionary of options that influence the look and feel of the selected theme. These are theme-specific. The options understood by the *builtin themes* are described *here*.

Added in version 0.6.

html_theme_path

```
Type
list[str]

Default
[]
```

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*.

Added in version 0.6.

html_style

```
Type
    Sequence[str] | str
Default
    ()
```

Stylesheets to use for HTML pages. The stylesheet given by the selected theme is used by default A file of that name must exist either in Sphinx's static/ path or in one of the custom paths given in httml_static_path. If you only want to add or override a few things from the theme, use CSS @import to import the theme's stylesheet.

Changed in version 5.1: The value can be a iterable of strings.

html_title

```
Type
    str

Default
    'project release documentation'
```

The "title" for HTML documentation generated with Sphinx's own templates. This is appended to the <tile>tag of individual pages, and used in the navigation bar as the "topmost" element.

html_short_title

```
Type
str

Default
The value of html_title
```

A shorter "title" for HTML documentation. This is used for links in the header and in the HTML Help documentation.

Added in version 0.4.

html_baseurl

```
Type str
Default
```

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**²²⁰.

Added in version 1.8.

html_codeblock_linenos_style

```
Type
    'inline' | 'table'

Default
    'inline'
```

The style of line numbers for code-blocks.

```
'table'
```

Display line numbers using tag

'inline'

Display line numbers using tag

Added in version 3.2.

Changed in version 4.0: It defaults to 'inline'.

Deprecated since version 4.0.

html_context

```
Type
    dict[str, Any]
Default
    {}
```

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 **sphinx-build**'s --html-define command-line option.

Added in version 0.5.

html_logo

```
Type
str
Default
```

²²⁰ https://datatracker.ietf.org/doc/html/rfc6596.html

If given, this must be the name of an image file (path relative to the *configuration directory*) that is the logo of the documentation, or a 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.

Added in version 0.4.1: The image file will be copied to the _static directory, but only if the file does not already exist there.

Changed in version 4.0: Also accepts a URL.

html_favicon

```
Type
str
Default
```

If given, this must be the name of an image file (path relative to the *configuration directory*) that is the favicon²²¹ of the documentation, or a URL that points an image file for the favicon. Browsers use this as the icon for tabs, windows and bookmarks. It should be a 16-by-16 pixel icon in the PNG, SVG, GIF, or ICO file formats.

Example:

```
html_favicon = 'static/favicon.png'
```

Added in version 0.4: The image file will be copied to the _static, 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

```
Type
    Sequence[str | tuple[str, dict[str, str]]]
Default
    []
```

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 <code>html_static_path</code>, or a full URI with scheme like 'https://example.org/style.css'. The *attributes* dictionary is used for the link> tag's attributes.

Example:

```
html_css_files = [
    'custom.css',
    'https://example.com/css/custom.css',
    ('print.css', {'media': 'print'}),
]
```

The special attribute *priority* can be set as an integer to load the CSS file at an earlier or later step. For more information, refer to *Sphinx.add_css_file()*.

Added in version 1.8.

Changed in version 3.5: Support the *priority* attribute

html_js_files

```
Type
   Sequence[str | tuple[str, dict[str, str]]]
```

 $^{^{221}\} https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes/rel\#icon$

Default

A list of JavaScript 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 'https://example.org/script.js'. The *attributes* dictionary is used for the <script> tag's attributes.

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 JavaScript file at an earlier or later step. For more information, refer to *Sphinx.add_js_file()*.

Added in version 1.8.

Changed in version 3.5: Support the *priority* attribute

html_static_path

```
Type
list[str]

Default
[]
```

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.

1 Note

For security reasons, dotfiles under html_static_path will not be copied. If you would like to copy them intentionally, explicitly add each file to this setting:

```
html_static_path = ['_static', '_static/.htaccess']
```

An alternative approach is to use html_extra_path, which allows copying dotfiles under the directories.

Changed in version 0.4: The paths in <a href="https://

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

```
Type
list[str]

Default
[]
```

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.

Added 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

```
Type
str
Default
```

None

If set, a 'Last updated on:' timestamp is inserted into the page footer using the given $strftime()^{222}$ format. The empty string is equivalent to '%b %d, %Y' (or a locale-dependent equivalent).

html_last_updated_use_utc

```
Type
bool

Default
False
```

Use GMT/UTC (+00:00) instead of the system's local time zone for the time supplied to html_last_updated_fmt. This is most useful when the format used includes the time.

html_permalinks

```
Type
bool
Default
True
```

Add link anchors for each heading and description environment.

Added in version 3.5.

html_permalinks_icon

```
Type
str

Default
'¶' (the paragraph sign)
```

Text for link anchors for each heading and description environment. HTML entities and Unicode are allowed.

Added in version 3.5.

html_sidebars

```
Type
    dict[str, Sequence[str]]
```

²²² https://docs.python.org/3/library/time.html#time.strftime

Default

{}

A dictionary defining custom sidebar templates, mapping document names to template names.

The keys can contain *glob-style patterns*, 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.)

Each value must be a list of strings which 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. The builtin themes use these templates by default: 'localtoc.html', 'relations.html', 'sourcelink.html', and 'searchbox.html'.

The bundled first-party 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': ['windows-sidebar.html', 'searchbox.html'],
}
```

This will render the custom template windows-sidebar.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).

Note that this value only has no effect if the chosen theme does not possess a sidebar, like the builtin **scrolls** and **haiku** themes.

Added in version 1.0: The ability to use globbing keys and to specify multiple sidebars.

Deprecated since version 1.7: A single string value for html_sidebars will be removed.

Changed in version 2.0: html_sidebars must be a list of strings, and no longer accepts a single string value.

html_additional_pages

```
Type
    dict[str, str]
Default
{}
```

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': 'custom-download.html.jinja',
}
```

This will render the template custom-download.html.jinja as the page download.html.

html_domain_indices

```
Type
    bool | Sequence[str]
Default
    True
```

If True, generate domain-specific indices in addition to the general index. For e.g. the Python domain, this is the global module index.

This value can be a Boolean 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'.

Example:

```
html_domain_indices = {
    'py-modindex',
}
```

Added in version 1.0.

Changed in version 7.4: Permit and prefer a set type.

html_use_index

```
Type
bool
Default
True
```

Controls if an index is added to the HTML documents.

Added in version 0.4.

html_split_index

```
Type bool

Default
```

False

Generates two versions of the index: once as a single page with all the entries, and once as one page per starting letter.

Added in version 0.4.

html_copy_source

```
Type
bool
Default
True
```

If True, the reStructuredText sources are included in the HTML build as _sources/docname.

html_show_sourcelink

```
Type
```

bool

Default

True

If True (and html_copy_source is true as well), links to the reStructuredText sources will be added to the sidebar.

Added in version 0.6.

html_sourcelink_suffix

```
Type
str

Default
'.txt'
```

The suffix to append to source links (see html_show_sourcelink), unless files they have this suffix already.

Added in version 1.5.

html_use_opensearch

```
Type
str
Default
```

If nonempty, an OpenSearch 223 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'.

Added in version 0.2.

html_file_suffix

```
Type
str

Default
'.html'
```

The file name suffix (file extension) for generated HTML files.

Added in version 0.4.

html_link_suffix

```
Type
str
```

Default

The value of html_file_suffix

The suffix for generated links to HTML files. Intended to support more esoteric server setups.

Added in version 0.6.

²²³ https://github.com/dewitt/opensearch

html_show_copyright

Type

bool

Default

True

If True, "© Copyright ..." is shown in the HTML footer, with the value or values from copyright.

Added in version 1.0.

html_show_search_summary

Type

bool

Default

True

Show a summary of the search result, i.e., the text around the keyword.

Added in version 4.5.

html_show_sphinx

Type

bool

Default

True

Add "Created using Sphinx²²⁴" to the HTML footer.

Added in version 0.4.

html_output_encoding

Type

str

Default

'utf-8'

Encoding of HTML output files. This encoding name must both be a valid Python encoding name and a valid HTML charset value.

Added in version 1.0.

html_compact_lists

Type

bool

Default

True

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 element for any of its items. This is standard docutils behaviour. Default: **True**.

Added in version 1.0.

²²⁴ https://www.sphinx-doc.org/

html_secnumber_suffix

```
Type
str
Default
```

Suffix for section numbers in HTML output. Set to ' ' to suppress the final dot on section numbers.

Added in version 1.0.

html_search_language

```
Type
str
```

Default

The value of language

Language to be used for generating the HTML full-text search index. This defaults to the global language selected with *language*. English ('en') is used as a fall-back option if there is no support for this language.

Support exists for the following languages:

- da Danish
- nl Dutch
- en English
- fi Finnish
- fr French
- de German
- \bullet hu Hungarian
- it Italian
- ja Japanese
- no Norwegian
- pt Portuguese
- ro Romanian
- ru Russian
- es Spanish
- sv Swedish
- tr Turkish
- zh Chinese

<equation-block> Tip

Accelerating build speed

Each language (except Japanese) provides its own stemming algorithm. Sphinx uses a Python implementation by default. If you want to accelerate building the index file, you can use a third-party package (PyStemmer²²⁵) by running **pip install PyStemmer**.

Added in version 1.1: Support English (en) and Japanese (ja).

Changed in version 1.3: Added additional languages.

html_search_options

```
Type
    dict[str, str]
Default
{}
```

A dictionary with options for the search language support. The meaning of these options depends on the language selected. Currently, only Japanese and Chinese support options.

Japanese

type - the type of the splitter to use.

The other options depend on the splitter used.

```
'sphinx.search.ja.DefaultSplitter'
```

The TinySegmenter algorithm, used by default.

```
'sphinx.search.ja.MecabSplitter':
```

The MeCab binding To use this splitter, the 'mecab' python binding or dynamic link library ('libmecab.so' for Linux, 'libmecab.dll' for Windows) is required.

```
'sphinx.search.ja.JanomeSplitter':
```

The Janome binding. To use this splitter, Janome²²⁶ is required.

Deprecated since version 1.6: 'mecab', 'janome' and 'default' is deprecated. To keep compatibility, 'mecab', 'janome' and 'default' are also acceptable.

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 ${\tt ctypes}$ 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'.

²²⁵ https://pypi.org/project/PyStemmer/

Chinese

dict

The jieba dictionary path for using a custom dictionary.

Added in version 1.1.

Changed in version 1.4: Allow any custom splitter in the *type* setting for Japanese.

html_search_scorer

```
Type
str
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.

The scorer must implement the following interface, and may optionally define the score() function for more granular control.

```
const Scorer = {
   // Implement the following function to further tweak the score for each result
    score: result => {
      const [docName, title, anchor, descr, score, filename] = result
     // ... calculate a new score ...
     return score
    },
    // query matches the full name of an object
   objNameMatch: 11,
    // or matches in the last dotted part of the object name
   objPartialMatch: 6,
    // Additive scores depending on the priority of the object
   objPrio: {
     0: 15, // used to be importantResults
     1: 5, // used to be objectResults
     2: -5, // used to be unimportantResults
    // Used when the priority is not in the mapping.
    objPrioDefault: 0,
    // query found in title
    title: 15.
   partialTitle: 7,
    // query found in terms
    term: 5,
   partialTerm: 2,
};
```

Added in version 1.2.

²²⁶ https://pypi.org/project/Janome/

html_scaled_image_link

Type
bool

Default

True

Link images that have been resized with a scale option (*scale*, *width*, or *height*) to their original full-resolution image. This will not overwrite any link given by the *target* option on the the image²²⁷ directive, if present.

```
Tip

To disable this feature on a per-image basis, add the no-scaled-link class to the image directive:

.. image:: sphinx.png
.:scale: 50%
.:class: no-scaled-link
```

Added in version 1.3.

Changed in version 3.0: Images with the no-scaled-link class will not be linked.

html_math_renderer

```
Type
str

Default
'mathjax'
```

The maths renderer to use for HTML output. The bundled renders are *mathjax* and *imgmath*. You must also load the relevant extension in *extensions*.

Added in version 1.8.

Options for Single HTML output

These options influence Single HTML output. This builder derives from the HTML builder, so the HTML options also apply where appropriate.

singlehtml_sidebars

```
Type
    dict[str, Sequence[str]]
Default
    The value of html_sidebars
```

A dictionary defining custom sidebar templates, mapping document names to template names.

This has the same effect as html_sidebars, but the only permitted key is 'index', and all other keys are ignored.

²²⁷ https://docutils.sourceforge.io/docs/ref/rst/directives.html#image

Options for HTML help output

These options influence HTML help output. This builder derives from the HTML builder, so the HTML options also apply where appropriate.

htmlhelp_basename

```
Type
    str

Default
    '{project}doc'
```

Output file base name for HTML help builder. The default is the *project name* with spaces removed and doc appended.

htmlhelp_file_suffix

```
Type
str

Default
'.html'
```

This is the file name suffix for generated HTML help files.

Added in version 2.0.

htmlhelp_link_suffix

```
Type
str
```

Default

The value of **htmlhelp_file_suffix**

Suffix for generated links to HTML files.

Added in version 2.0.

Options for Apple Help output

Added in version 1.3.

These options influence Apple Help output. This builder derives from the HTML builder, so the HTML options also apply where appropriate.



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 <code>applehelp_disable_external_tools</code>, but the result will not be a valid help book until the indexer is run over the <code>.lproj</code> directories within the bundle.

applehelp_bundle_name

```
Type
str

Default
The value of project
```

The basename for the Apple Help Book. The default is the *project name* with spaces removed.

applehelp_bundle_id

```
Type
    str
Default
```

None

The bundle ID for the help book bundle.

Warning

You must set this value in order to generate Apple Help.

applehelp_bundle_version

```
Type
    str
Default
    '1'
```

The bundle version, as a string.

applehelp_dev_region

```
Type
    str
Default
    'en-us'
```

The development region. Defaults to Apple's recommended setting, 'en-us'.

applehelp_icon

```
Type
   str
Default
   None
```

Path to 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

```
Type
              str
          Default
              'project-release'
     The product tag for use with applehelp_kb_url.
applehelp_kb_url
          Type
              str
```

Default

None

The URL for your knowledgebase server, e.g. https://example.com/kbsearch.py? p='product'&q='query'&l='lang'. At runtime, Help Viewer will replace 'product' with the contents of applehelp_kb_product, 'query' with the text entered by the user in the search box, and 'lang' with the user's system language.

Set this to to **None** to disable remote search.

applehelp_remote_url

Type

str

Default

None

The URL for remote content. You can place a copy of your Help Book's Resources directory at this location and Help Viewer will attempt to use it to fetch updated content.

For example, 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.

Set this to to **None** for no remote content.

applehelp_index_anchors

Type

bool

Default

False

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

Type

str

Default

None

Controls the minimum term length for the help indexer. If **None**, use the default length.

applehelp_stopwords

Type

str

Default

The value of language

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:

- German (de)
- English (en)

```
• Spanish (es)
```

- French (fr)
- Hungarian (hu)
- Italian (it)
- Swedish (sv)

applehelp_locale

```
Type
```

str

Default

The value of language

Specifies the locale to generate help for. This is used to determine the name of the .lproj directory inside the Help Book's Resources, and is passed to the help indexer.

applehelp_title

```
Type
```

str

Default

'project Help'

Specifies the help book title.

applehelp_codesign_identity

```
Type
```

str

Default

The value of CODE_SIGN_IDENTITY

Specifies the identity to use for code signing. Use None if code signing is not to be performed.

Defaults to the value of the CODE_SIGN_IDENTITY environment variable, which is set by Xcode for script build phases, or **None** if that variable is not set.

applehelp_codesign_flags

```
Type
```

list[str]

Default

The value of OTHER_CODE_SIGN_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 OTHER_CODE_SIGN_FLAGS environment variable, which is set by Xcode for script build phases, or the empty list if that variable is not set.

applehelp_codesign_path

```
Type
```

str

Default

'/usr/bin/codesign'

The path to the **codesign** program.

applehelp_indexer_path

```
Type
   str
Default
    '/usr/bin/hiutil'
```

The path to the **hiutil** program.

applehelp_disable_external_tools

```
Type
    bool
Default
   False
```

Do 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-macOS platform and then complete the final steps on a Mac for some reason.

Options for EPUB output

These options influence EPUB output. This builder derives from the HTML builder, so the HTML options also apply where appropriate.



1 Note

The actual value for some of these options is not important, but they are required for the Dublin Core metadata²²⁸.

epub_basename

```
Type
    str
Default
    The value of project
```

The basename for the EPUB file.

epub_theme

```
Type
    str
Default
```

The HTML theme for the EPUB output. Since the default themes are not optimised 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

```
Type
   dict[str, Any]
```

²²⁸ https://dublincore.org/

```
Default
               {}
     A dictionary of options that influence the look and feel of the selected theme. These are theme-specific. The
     options understood by the builtin themes are described here.
     Added in version 1.2.
epub_title
           Type
               str
           Default
               The value of project
     The title of the document.
     Changed in version 2.0: It defaults to the project option (previously html_title).
epub_description
           Type
           Default
               'unknown'
     The description of the document.
     Added in version 1.4.
     Changed in version 1.5: Renamed from epub3_description
epub_author
           Type
               str
           Default
               The value of author
     The author of the document. This is put in the Dublin Core metadata.
epub_contributor
           Type
               str
           Default
               'unknown'
     The name of a person, organisation, etc. that played a secondary role in the creation of the content of an EPUB
     Publication.
     Added in version 1.4.
     Changed in version 1.5: Renamed from epub3_contributor
epub_language
           Type
```

4.4. Configuration 119

str

The value of language

Default

The language of the document. This is put in the Dublin Core metadata.

epub_publisher

```
Type
```

str

Default

The value of author

The publisher of the document. This is put in the Dublin Core metadata. You may use any sensible string, e.g. the project homepage.

epub_copyright

```
Type
```

str

Default

The value of copyright

The copyright of the document. See *copyright* for permitted formats.

epub_identifier

```
Type
```

str

Default

'unknown'

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.

epub_scheme

```
Type
```

str

Default

'unknown'

The publication scheme for the <code>epub_identifier</code>. This is put in the Dublin Core metadata. For published books the scheme is 'ISBN'. If you use the project homepage, 'URL' seems reasonable.

epub_uid

```
Type
```

str

Default

'unknown'

A unique identifier for the document. This is put in the Dublin Core metadata. You may use a XML's Name format²²⁹ string. You can't use hyphen, period, numbers as a first character.

epub_cover

```
Type
    tuple[str, str]
Default
()
```

²²⁹ https://www.w3.org/TR/REC-xml/#NT-NameStartChar

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 = ()
```

Added in version 1.1.

```
epub_css_files
```

```
Type
    Sequence[str | tuple[str, dict[str, str]]]
Default
    []
```

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 'https://example.org/style.css'. The *attributes* dictionary is used for the link> tag's attributes. For more information, see *html_css_files*.

Added in version 1.8.

epub_guide

```
Type
    Sequence[tuple[str, str, str]]
Default
    ()
```

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 documentation²³⁰ 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', 'Cover Page'),
)
```

The default value is ().

epub_pre_files

```
Type
    Sequence[tuple[str, str]]
Default
    ()
```

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:

²³⁰ https://idpf.org/epub

```
epub_pre_files = [
    ('index.html', 'Welcome'),
]
```

epub_post_files

```
Type
    Sequence[tuple[str, str]]
Default
    ()
```

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.

Example:

```
epub_post_files = [
    ('appendix.xhtml', 'Appendix'),
]
```

epub_exclude_files

```
Type
Sequence[str]
Default
[]
```

A sequence of files that are generated/copied in the build directory but should not be included in the EPUB file.

epub_tocdepth

```
Type int

Default
```

The depth of the table of contents in the file toc.ncx. It should be an integer greater than zero.

epub_tocdup

```
Type
bool
Default
True
```

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.

epub_tocscope

```
Type
    'default' | 'includehidden'

Default
    'default'
```

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

'includehidden'

Include all ToC entries

Added in version 1.2.

epub_fix_images

```
Type bool
```

Default

False

Try and fix image formats that are not supported by some EPUB readers. At the moment palette images with a small colour table are upgraded. This is disabled by default because the automatic conversion may lose information. You need the Python Image Library (Pillow²³¹) installed to use this option.

Added in version 1.2.

epub_max_image_width

```
Type int
Default
```

This option specifies the maximum width of images. If it is set to a valuevgreater than zero, images with a width larger than the given value are scaled accordingly. If it is zero, no scaling is performed. You need the Python Image Library (Pillow²³²) installed to use this option.

Added in version 1.2.

epub_show_urls

```
Type
    'footnote' | 'no' | 'inline'
Default
    'footnote'
```

Control how to display URL addresses. This is very useful for readers that have no other means to display the linked URL. The setting can have the following values:

'inline'

Display URLs inline in parentheses.

'footnote'

Display URLs in footnotes.

²³¹ https://pypi.org/project/Pillow/

²³² https://pypi.org/project/Pillow/

```
'no'
```

Do not display URLs.

The display of inline URLs can be customised by adding CSS rules for the class link-target.

Added in version 1.2.

epub_use_index

```
Type bool
```

Default

The value of **html_use_index**

Add an index to the EPUB document.

Added in version 1.2.

epub_writing_mode

```
Type
    'horizontal' | 'vertical'

Default
    'horizontal'
```

It specifies writing direction. It can accept 'horizontal' and 'vertical'

epub_writing_mode	'horizontal'	'vertical'
writing-mode ²³³	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.

Options for LaTeX output

These options influence LaTeX output.

latex_engine

```
Type
    'pdflatex' | 'xelatex' | 'lualatex' | 'platex' | 'uplatex'

Default
    'pdflatex'
```

The LaTeX engine to build the documentation. The setting can have the following values:

- 'pdflatex' PDFLaTeX (default)
- 'xelatex' XeLaTeX (default if language is one of el, zh_CN, or zh_TW)
- 'lualatex' LuaLaTeX
- 'platex' pLaTeX
- 'uplatex' upLaTeX (default if language is 'ja')

²³³ https://developer.mozilla.org/en-US/docs/Web/CSS/writing-mode

Important

'pdflatex''s support for Unicode characters is limited. 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 typeface is GNU FreeFont, which has good coverage of Latin, Cyrillic, and Greek glyphs.

Note

Sphinx 2.0 adds support for occasional Cyrillic and Greek letters or words in documents using a Latin language and 'pdflatex'. To enable this, the 'fontenc' key of latex_elements must be used appropriately.

1 Note

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 'preamble' key of latex_elements). You may prefer r'\usepackage[math-style=literal]{unicode-math}' to keep a Unicode literal such as α (U+03B1) as-is in output, rather than being rendered as α .

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: Use 'uplatex' by default for Japanese documents.

latex_documents

Type

```
Sequence[tuple[str, str, str, str, str, bool]]
```

Default

The default 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.

Added in version 0.3: The 6th item toctree_only. Tuples with 5 items are still accepted.

Added 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.

latex_logo

```
Type str
Default
```

If given, this must be the name of an image file (path relative to the *configuration directory*) that is the logo of the documentation. It is placed at the top of the title page.

latex_toplevel_sectioning

```
Type
    'part' | 'chapter' | 'section'
Default
    None
```

This value determines the topmost sectioning unit. The default setting is 'section' if *latex_theme* is 'howto', and 'chapter' if it is 'manual'. The alternative in both cases is to specify 'part', which means that LaTeX document will use the \part command.

In that case the numbering of sectioning units one level deep gets off-sync with HTML numbering, as by default LaTeX does not reset \chapter numbering (or \section for 'howto' theme) when encountering \part command.

Added in version 1.4.

latex_appendices

```
Type
Sequence[str]
Default
()
```

A list of document names to append as an appendix to all manuals. This is ignored if <code>latex_theme</code> is set to 'howto'.

latex_domain_indices

```
Type
bool | Sequence[str]

Default
True
```

If True, generate domain-specific indices in addition to the general index. For e.g. the Python domain, this is the global module index.

This value can be a Boolean 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'.

Example:

```
latex_domain_indices = {
    'py-modindex',
}
```

Added in version 1.0.

Changed in version 7.4: Permit and prefer a set type.

latex_show_pagerefs

```
Type
bool

Default
False
```

Add page references after internal references. This is very useful for printed copies of the manual.

Added in version 1.0.

latex_show_urls

```
Type
    'no' | 'footnote' | 'inline'
Default
    'no'
```

Control how 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
'footnote'
Display URLs in footnotes
'inline'
Display URLs inline in parentheses
```

Added 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

```
Type
bool

Default
False
```

Use standard LaTeX's \multicolumn for merged cells in tables.

False

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.

True

Use LaTeX's standard \multicolumn; this is incompatible with literal blocks in horizontally merged cells, and also with multiple paragraphs in such cells if the table is rendered using tabulary.

Added in version 1.6.

latex_table_style

```
Type
    list[str]

Default
    ['booktabs', 'colorrows']
```

A list of styling classes (strings). Currently supported:

'booktabs'

No vertical lines, and only 2 or 3 horizontal lines (the latter if there is a header), using the booktabs²³⁴ package.

'borderless'

No lines whatsoever.

'colorrows'

The table rows are rendered with alternating background colours. The interface to customise them is via *dedicated keys* of *The sphinxsetup configuration setting*.

Important

With the 'colorrows' style, the \rowcolors LaTeX command becomes a no-op (this command has limitations and has never correctly supported all types of tables Sphinx produces in LaTeX). Please update your project to use the *latex table color configuration* keys instead.

To customise the styles for a table, use the :class: option if the table is defined using a directive, or otherwise insert a *rst-class* directive before the table (cf. *Tables*).

Currently recognised classes are booktabs, borderless, standard, colorrows, nocolorrows. The latter two can be combined with any of the first three. The standard class produces tables with both horizontal and vertical lines (as has been the default so far with Sphinx).

A single-row multi-column merged cell will obey the row colour, if it is set. See also TableMergeColor{Header,Odd,Even} in the *The sphinxsetup configuration setting* section.

1 Note

• It is hard-coded in LaTeX that a single cell will obey the row colour even if there is a column colour set via \columncolor from a column specification (see tabularcolumns). Sphinx provides \sphinxnorowcolor which can be used in a table column specification like this:

```
>{\columncolor{blue}\sphinxnorowcolor}
```

• Sphinx also provides \sphinxcolorblend, which however requires the xcolor²³⁵ package. Here is an example:

>{\sphinxcolorblend{!95!red}}

It means that in this column, the row colours will be slightly tinted by red; refer to xcolor²³⁶ documentation for more on the syntax of its \blendcolors command (a \blendcolors in place of \sphinxcolorblend would modify colours of the cell contents, not of the cell background colour panel...). You can find an example of usage in the Deprecated APIs section of this document in PDF format.

Hint

If you want to use a special colour for the *contents* of the cells of a given column use >{\noindent\ color{<color>}}, possibly in addition to the above.

- · Multi-row merged cells, whether single column or multi-column currently ignore any set column, row, or cell colour.
- It is possible for a simple cell to set a custom colour via the raw²³⁷ directive and the \cellcolor LaTeX command used anywhere in the cell contents. This currently is without effect in a merged cell, whatever its kind.

Hint

In a document not using 'booktabs' globally, it is possible to style an individual table via the booktabs class, but it will be necessary to add r'\usepackage{booktabs}' to the LaTeX preamble.

On the other hand one can use colorrows class for individual tables with no extra package (as Sphinx since 5.3.0 always loads colortbl²³⁸).

Added in version 5.3.0.

Changed in version 6.0.0: Modify default from [] to ['booktabs', 'colorrows'].

latex_use_xindy

Type

bool

Default

```
True if latex_engine in {'xelatex', 'lualatex'} else False
```

Use Xindy²³⁹ to prepare the index of general terms. By default, the LaTeX builder uses **makeindex** for preparing the index of general terms. Using Xindy²⁴⁰ means that words with UTF-8 characters will be 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 creates .ind files containing invalid bytes for the UTF-8 encoding if any indexed term starts with a non-ASCII character. With 'lualatex' this then breaks the PDF build.

²³⁴ https://ctan.org/pkg/booktabs

²³⁵ https://ctan.org/pkg/xcolor

²³⁶ https://ctan.org/pkg/xcolor

²³⁷ https://docutils.sourceforge.io/docs/ref/rst/directives.html#raw

²³⁸ https://ctan.org/pkg/colortbl

• 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 some dedicated support to the **xindy** base distribution for using 'pdflatex' engine with Cyrillic scripts. With both 'pdflatex' and Unicode engines, Cyrillic documents handle the indexing of Latin names correctly, even those having diacritics.

Added in version 1.8.

latex_elements

```
Type
   dict[str, str]
Default
    {}
```

Added in version 0.5.

See the full documentation for latex_elements.

latex_docclass

```
Type
   dict[str, str]
Default
```

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'.

Added in version 1.0.

Changed in version 1.5: In Japanese documentation (language is 'ja'), by default 'jreport' is used for 'howto' and 'jsbook' for 'manual'.

latex_additional_files

```
Type
   Sequence[str]
Default
    ()
```

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, or to overwrite Sphinx LaTeX support files with custom versions. Image files that are referenced in source files (e.g. via .. image::) are copied automatically and should not be listed there.

A Attention

Filenames with the .tex extension will be automatically handed over to the PDF build process triggered by sphinx-build -M latexpdf or by make latexpdf. If the file was added only to be \input{} from a modified preamble, you must add a further suffix such as .txt to the filename and adjust the \input{} macro accordingly.

Added in version 0.6.

²³⁹ https://xindy.sourceforge.net/

²⁴⁰ https://xindy.sourceforge.net/

Changed in version 1.2: This overrides the files provided from Sphinx such as sphinx.sty.

latex_theme

```
Type
str

Default
'manual'
```

The "theme" that the LaTeX output should use. It is a collection of settings for LaTeX output (e.g. document class, top level sectioning unit and so on).

The bundled first-party LaTeX themes are manual and howto:

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). latex_appendices is only available for this theme.

Added in version 3.0.

latex_theme_options

```
Type
    dict[str, Any]
Default
{}
```

A dictionary of options that influence the look and feel of the selected theme. These are theme-specific.

Added in version 3.1.

latex_theme_path

```
Type
list[str]

Default
[]
```

A list of paths that contain custom LaTeX themes as subdirectories. Relative paths are taken as relative to the *configuration directory*.

Added in version 3.0.

Options for text output

These options influence text output.

text_add_secnumbers

```
Type
bool
Default
True
```

Include section numbers in text output.

Added in version 1.7.

text_newlines

```
Type
    'unix' | 'windows' | 'native'

Default
    'unix'
```

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.

Added in version 1.1.

text_secnumber_suffix

```
Type
str

Default
```

Suffix for section numbers in text output. Set to ' ' to suppress the final dot on section numbers.

Added in version 1.7.

text_sectionchars

```
Type
str

Default

'*=-~"+`'
```

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.

Added in version 1.1.

Options for manual page output

These options influence manual page output.

man_pages

```
Type
    Sequence[tuple[str, str, str, str, str]]
Default
    The default manual 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 page. (If you want to use the default master document for your manual pages build, provide 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. Can be an empty string if you do not want to automatically generate 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.

Added in version 1.0.

man_show_urls

```
Type
```

bool

Default

False

Add URL addresses after links.

Added in version 1.1.

man_make_section_directory

```
Type
```

bool

Default

True

Make a section directory on build man page.

Added in version 3.3.

Changed in version 4.0: The default is now **False** (previously **True**).

Changed in version 4.0.2: Revert the change in the default.

Options for Texinfo output

These options influence Texinfo output.

texinfo_documents

```
Type
```

```
Sequence[tuple[str, str, str, str, str, str, bool]]
```

Default

The default 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 Texinfo file's master document. All documents referenced

4.4. Configuration 133

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.

Added in version 1.1.

texinfo_appendices

```
Type
```

Sequence[str]

Default

Г٦

A list of document names to append as an appendix to all manuals.

Added in version 1.1.

texinfo_cross_references

```
Type
```

bool

Default

True

Generate inline references in a document. Disabling inline references can make an info file more readable with a stand-alone reader (info).

Added in version 4.4.

texinfo_domain_indices

```
Type
```

bool | Sequence[str]

Default

True

If True, generate domain-specific indices in addition to the general index. For e.g. the Python domain, this is the global module index.

This value can be a Boolean 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'.

Example:

```
texinfo_domain_indices = {
    'py-modindex',
}
```

Added in version 1.1.

Changed in version 7.4: Permit and prefer a set type.

texinfo_elements

```
Type
    dict[str, Any]
Default
{}
```

A dictionary that contains Texinfo snippets that override those that 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', and 'title'.

Added in version 1.1.

texinfo_no_detailmenu

```
Type
bool
Default
```

Do not generate a @detailmenu in the "Top" node's menu containing entries for each sub-node in the document.

Added in version 1.2.

texinfo_show_urls

```
Type
   'footnote' | 'no' | 'inline'
```

4.4. Configuration 135

```
Default
               'footnote'
     Control how to display URL addresses. The setting can have the following values:
      'footnote'
          Display URLs in footnotes.
      'no'
          Do not display URLs.
      'inline'
          Display URLs inline in parentheses.
     Added in version 1.1.
Options for QtHelp output
These options influence qthelp output. This builder derives from the HTML builder, so the HTML options also apply
where appropriate.
qthelp_basename
          Type
               str
          Default
               The value of project
     The basename for the qthelp file.
qthelp_namespace
          Type
               str
          Default
               'org.sphinx.{project_name}.{project_version}'
     The namespace for the qthelp file.
qthelp_theme
          Type
               str
          Default
               'nonav'
     The HTML theme for the qthelp output.
qthelp_theme_options
          Type
```

A dictionary of options that influence the look and feel of the selected theme. These are theme-specific. The options understood by the *builtin themes* are described *here*.

dict[str, Any]

Default

Options for XML output

```
xml_pretty
Type
bool
Default
```

Pretty-print the XML.

True

Added in version 1.2.

Options for the linkcheck builder

Filtering

These options control which links the *linkcheck* builder checks, and which failures and redirects it ignores.

linkcheck_allowed_redirects

```
Type
    dict[str, str]
Default
{}
```

A dictionary that maps a pattern of the source URI to a pattern of the canonical URI. The *linkcheck* builder treats the redirected link as "working" when:

- the link in the document matches the source URI pattern, and
- the redirect location matches the canonical URI pattern.

The *linkcheck* builder will emit a warning when it finds redirected links that don't meet the rules above. It can be useful to detect unexpected redirects when using *the fail-on-warnings mode*.

Example:

```
linkcheck_allowed_redirects = {
    # All HTTP redirections from the source URI to
    # the canonical URI will be treated as "working".
    r'https://sphinx-doc\.org/.*': r'https://sphinx-doc\.org/en/master/.*'
}
```

Added in version 4.1.

linkcheck_anchors

```
Type
bool
Default
True
```

Check the validity of #anchors in links. Since this requires downloading the whole document, it is considerably slower when enabled.

Added in version 1.2.

4.4. Configuration 137

linkcheck_anchors_ignore

```
Type
    Sequence[str]
Default
    ["^!"]
```

A list of regular expressions that match anchors that the *linkcheck* builder should skip when checking the validity of anchors in links. For example, this allows skipping anchors added by a website's JavaScript.

```
🗘 Tip
```

Use linkcheck_anchors_ignore_for_url to check a URL, but skip verifying that the anchors exist.

1 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 <code>linkcheck_ignore</code> instead, for example as follows:

```
linkcheck_ignore = [
   'https://www.sphinx-doc.org/en/1.7/intro.html#',
]
```

Added in version 1.5.

linkcheck_anchors_ignore_for_url

```
Type
    Sequence[str]
Default
    ()
```

A list or tuple of regular expressions matching URLs for which the *linkcheck* builder should not check the validity of anchors. This allows skipping anchor checks on a per-page basis while still checking the validity of the page itself.

Added in version 7.1.

linkcheck_exclude_documents

```
Type
Sequence[str]

Default
```

A list of regular expressions that match documents in which the *linkcheck* builder should not check the validity of links. This can be used for permitting link decay in legacy or historical sections of the documentation.

Example:

```
# ignore all links in documents located in a subdirectory named 'legacy'
linkcheck_exclude_documents = [r'.*/legacy/.*']
```

Added in version 4.4.

linkcheck_ignore

```
Type
Sequence[str]

Default
```

A list of regular expressions that match URIs that should not be checked when doing a linkcheck build.

Example:

```
linkcheck_ignore = [r'https://localhost:\d+/']
```

Added in version 1.1.

HTTP Requests

These options control how the *linkcheck* builder makes HTTP requests, including how it handles redirects and authentication, and the number of workers to use.

linkcheck_auth

```
Type
    Sequence[tuple[str, Any]]
Default
    []
```

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 authentication²⁴¹ 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(...)),
]
```

Added in version 2.3.

linkcheck_allow_unauthorized

```
Type
bool

Default
False
```

4.4. Configuration 139

²⁴¹ https://requests.readthedocs.io/en/latest/user/authentication/#authentication

When a webserver responds with an HTTP 401 (unauthorised) response, the current default behaviour of the *linkcheck* builder is to treat the link as "broken". To change that behaviour, set this option to **True**.

Changed in version 8.0: The default value for this option changed to **False**, meaning HTTP 401 responses to checked hyperlinks are treated as "broken" by default.

Added in version 7.3.

linkcheck_rate_limit_timeout

```
Type int

Default
300
```

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 behaviour when servers indicate that requests are rate-limited, by setting the maximum duration (in seconds) that the builder will wait for between each attempt before recording a failure.

The *linkcheck* builder always respects a server's direction of when to retry (using the Retry-After²⁴² header). 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 (in seconds). Custom timeouts should be given as a number of seconds.

Added in version 3.4.

linkcheck_report_timeouts_as_broken

```
Type
bool

Default
False
```

If <code>linkcheck_timeout</code> expires while waiting for a response from a hyperlink, the <code>linkcheck</code> builder will report the link as a <code>timeout</code> by default. To report timeouts as <code>broken</code> instead, you can set <code>linkcheck_report_timeouts_as_broken</code> to <code>True</code>.

Changed in version 8.0: The default value for this option changed to **False**, meaning timeouts that occur while checking hyperlinks will be reported using the new 'timeout' status code.

Added in version 7.3.

linkcheck_request_headers

```
Type
    dict[str, dict[str, str]]
Default
{}
```

A dictionary that maps URL (without paths) to HTTP request headers.

The key is a URL base string like 'https://www.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:

²⁴² https://datatracker.ietf.org/doc/html/rfc7231#section-7.1.3

```
linkcheck_request_headers = {
    "https://www.sphinx-doc.org/": {
        "Accept": "text/html",
        "Accept-Encoding": "utf-8",
    },
    "*": {
        "Accept": "text/html,application/xhtml+xml",
    }
}
```

Added in version 3.1.

linkcheck_retries

```
Type int

Default
```

The number of times the *linkcheck* builder will attempt to check a URL before declaring it broken.

Added in version 1.4.

linkcheck_timeout

```
Type int

Default
30
```

The duration, in seconds, that the *linkcheck* builder will wait for a response after each hyperlink request.

Added in version 1.1.

linkcheck_workers

```
Type int
Default
5
```

The number of worker threads to use when checking links.

Added in version 1.1.

Domain options

Options for the C domain

c_extra_keywords

```
Type
    Set[str] | Sequence[str]

Default
    ['alignas', 'alignof', 'bool', 'complex', 'imaginary', 'noreturn',
    'static_assert', 'thread_local']
```

4.4. Configuration 141

A list of identifiers to be recognised as keywords by the C parser.

Added in version 4.0.3.

Changed in version 7.4: c_extra_keywords can now be a set.

c_id_attributes

```
Type
    Sequence[str]
Default
    ()
```

A sequence of strings that the parser should additionally accept as attributes. For example, this can be used when #define has been used for attributes, for portability.

Example:

```
c_id_attributes = [
    'my_id_attribute',
]
```

Added in version 3.0.

Changed in version 7.4: c_id_attributes can now be a tuple.

c_maximum_signature_line_length

```
Type
int | None

Default
None
```

If a signature's length in characters exceeds the number set, each parameter within the signature will be displayed on an individual logical line.

When None, there is no maximum length and the entire signature will be displayed on a single logical line.

This is a domain-specific setting, overriding maximum_signature_line_length.

Added in version 7.1.

c_paren_attributes

```
Type
    Sequence[str]
Default
    ()
```

A sequence of strings that the parser should additionally 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 {}). For example, this can be used when #define has been used for attributes, for portability.

Example:

```
c_paren_attributes = [
    'my_align_as',
]
```

Added in version 3.0.

Changed in version 7.4: c_paren_attributes can now be a tuple.

Options for the C++ domain

cpp_id_attributes

```
Type
    Sequence[str]
Default
    ()
```

A sequence of strings that the parser should additionally accept as attributes. For example, this can be used when #define has been used for attributes, for portability.

Example:

```
cpp_id_attributes = [
    'my_id_attribute',
]
```

Added in version 1.5.

Changed in version 7.4: cpp_id_attributes can now be a tuple.

cpp_index_common_prefix

```
Type
Sequence[str]

Default
```

A list of prefixes that will be ignored when sorting C++ objects in the global index.

Example:

```
cpp_index_common_prefix = [
    'awesome_lib::',
]
```

Added in version 1.5.

cpp_maximum_signature_line_length

```
Type
int | None

Default
None
```

If a signature's length in characters exceeds the number set, each parameter within the signature will be displayed on an individual logical line.

When None, there is no maximum length and the entire signature will be displayed on a single logical line.

This is a domain-specific setting, overriding maximum_signature_line_length.

Added in version 7.1.

4.4. Configuration 143

cpp_paren_attributes

```
Type
Sequence[str]
Default
()
```

A sequence of strings that the parser should additionally 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 {}). For example, this can be used when #define has been used for attributes, for portability.

Example:

```
cpp_paren_attributes = [
    'my_align_as',
]
```

Added in version 1.5.

Changed in version 7.4: cpp_paren_attributes can now be a tuple.

Options for the Javascript domain

javascript_maximum_signature_line_length

```
Type
int | None

Default
None
```

If a signature's length in characters exceeds the number set, each parameter within the signature will be displayed on an individual logical line.

When None, there is no maximum length and the entire signature will be displayed on a single logical line.

This is a domain-specific setting, overriding maximum_signature_line_length.

Added in version 7.1.

Options for the Python domain

add_module_names

```
Type
bool
Default
```

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.

modindex_common_prefix

```
Type
list[str]

Default
[]
```

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.

```
Caution
```

Works only for the HTML builder currently.

Added in version 0.6.

python_display_short_literal_types

```
Type
bool
Default
```

False

This value controls how Literal²⁴³ types are displayed.

Examples

The examples below use the following *py:function* directive:

```
.. py:function:: serve_food(item: Literal["egg", "spam", "lobster thermidor"]) ->⊔

→None
```

When **False**, Literal²⁴⁴ types display as per standard Python syntax, i.e.:

```
serve_food(item: Literal["egg", "spam", "lobster thermidor"]) -> None
```

When **True**, Literal²⁴⁵ types display with a short, **PEP 604**²⁴⁶-inspired syntax, i.e.:

```
serve_food(item: "egg" | "spam" | "lobster thermidor") -> None
```

Added in version 6.2.

python_maximum_signature_line_length

```
Type
int | None

Default
None
```

If a signature's length in characters exceeds the number set, each parameter within the signature will be displayed on an individual logical line.

When None, there is no maximum length and the entire signature will be displayed on a single logical line.

This is a domain-specific setting, overriding maximum_signature_line_length.

For the Python domain, the signature length depends on whether the type parameters or the list of arguments are being formatted. For the former, the signature length ignores the length of the arguments list; for the latter, the signature length ignores the length of the type parameters list.

4.4. Configuration 145

²⁴³ https://docs.python.org/3/library/typing.html#typing.Literal

²⁴⁴ https://docs.python.org/3/library/typing.html#typing.Literal

²⁴⁵ https://docs.python.org/3/library/typing.html#typing.Literal

²⁴⁶ https://peps.python.org/pep-0604/

For instance, with python_maximum_signature_line_length = 20, only the list of type parameters will be wrapped while the arguments list will be rendered on a single line

```
.. py:function:: add[T: VERY_LONG_SUPER_TYPE, U: VERY_LONG_SUPER_TYPE](a: T, b: U)
```

Added in version 7.1.

python_use_unqualified_type_names

Type

bool

Default

False

Suppress the module name of the python reference if it can be resolved.

Added in version 4.0.

```
Caution
```

This feature is experimental.

trim_doctest_flags

Type

bool

Default

True

Remove doctest flags (comments looking like # doctest: FLAG, ...) at the ends of lines and <BLANKLINE> markers for all code blocks showing interactive Python sessions (i.e. doctests). See the extension doctest for more possibilities of including doctests.

Added in version 1.0.

Changed in version 1.1: Now also removes <BLANKLINE>.

Extension options

Extensions frequently have their own configuration options. Those for Sphinx's first-party extensions are documented in each *extension's page*.

Example configuration file

(continues on next page)

(continued from previous page)

4.5 Builders

These are the built-in Sphinx builders. More builders can be added by extensions.

The builder's "name" must be given to the **-M** or **-b** command-line options of **sphinx-build** to select 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).

4.5. Builders 147

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.

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 reStructuredText 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: str^{247} = 'html'
```

The builder's name. This is the value used to select builders on the command line.

```
format: str^{248} = 'html'
```

The builder's output format, or "if no document output is produced. This is commonly the file extension, e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as *SphinxPostTransform* or extensions to determine their compatibility with the builder.

```
supported_image_types: list<sup>249</sup>[str<sup>250</sup>] = ['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: str<sup>251</sup> = 'dirhtml'
```

The builder's name. This is the value used to select builders on the command line.

```
format: str^{252} = 'html'
```

The builder's output format, or 'if no document output is produced. This is commonly the file extension, e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as *SphinxPostTransform* or extensions to determine their compatibility with the builder.

```
supported_image_types: list^{253}[str^{254}] = ['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.

Added in version 0.6.

 $^{^{247}\} https://docs.python.org/3/library/stdtypes.html\#str$

²⁴⁸ https://docs.python.org/3/library/stdtypes.html#str

²⁴⁹ https://docs.python.org/3/library/stdtypes.html#list

²⁵⁰ https://docs.python.org/3/library/stdtypes.html#str

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 root document. No indices will be generated.

```
name: str<sup>255</sup> = 'singlehtml'
```

The builder's name. This is the value used to select builders on the command line.

```
format: str^{256} = 'html'
```

The builder's output format, or "if no document output is produced. This is commonly the file extension, e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as *SphinxPostTransform* or extensions to determine their compatibility with the builder.

```
supported_image_types: list<sup>257</sup>[str<sup>258</sup>] = ['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.

Added 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: str<sup>259</sup> = 'htmlhelp'
```

The builder's name. This is the value used to select builders on the command line.

```
format: str^{260} = 'html'
```

The builder's output format, or 'if no document output is produced. This is commonly the file extension, e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as *SphinxPostTransform* or extensions to determine their compatibility with the builder.

```
supported_image_types: list<sup>261</sup>[str<sup>262</sup>] = ['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 help²⁶³ 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: str^{264} = 'gthelp'
```

The builder's name. This is the value used to select builders on the command line.

```
format: str^{265} = 'html'
```

The builder's output format, or "if no document output is produced. This is commonly the file extension,

4.5. Builders 149

²⁵¹ https://docs.python.org/3/library/stdtypes.html#str

²⁵² https://docs.python.org/3/library/stdtypes.html#str

²⁵³ https://docs.python.org/3/library/stdtypes.html#list

²⁵⁴ https://docs.python.org/3/library/stdtypes.html#str

²⁵⁵ https://docs.python.org/3/library/stdtypes.html#str

²⁵⁶ https://docs.python.org/3/library/stdtypes.html#str

²⁵⁷ https://docs.python.org/3/library/stdtypes.html#list

²⁵⁸ https://docs.python.org/3/library/stdtypes.html#str

²⁵⁹ https://docs.python.org/3/library/stdtypes.html#str

²⁶⁰ https://docs.python.org/3/library/stdtypes.html#str

²⁶¹ https://docs.python.org/3/library/stdtypes.html#list

²⁶² https://docs.python.org/3/library/stdtypes.html#str

e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as SphinxPostTransform or extensions to determine their compatibility with the builder.

```
supported_image_types: list<sup>266</sup>[str<sup>267</sup>] = ['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 .1proj 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: str<sup>268</sup> = 'applehelp'
```

The builder's name. This is the value used to select builders on the command line.

```
format: str^{269} = 'html'
```

The builder's output format, or "if no document output is produced. This is commonly the file extension, e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as SphinxPostTransform or extensions to determine their compatibility with the builder.

```
supported_image_types: list<sup>270</sup>[str<sup>271</sup>] = ['image/png', 'image/gif', 'image/jpeg',
'image/tiff', 'image/jp2', 'image/svg+xml']
```

The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

Added 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 Devhelp²⁷² support file that allows the GNOME Devhelp reader to view them.

```
name: str^{273} = 'devhelp'
```

The builder's name. This is the value used to select builders on the command line.

```
format: str<sup>274</sup> = 'html'
```

The builder's output format, or "if no document output is produced. This is commonly the file extension, e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as SphinxPostTransform or extensions to determine their compatibility with the builder.

 $^{^{263}\} https://doc.qt.io/qt-4.8/qthelp-framework.html$

²⁶⁴ https://docs.python.org/3/library/stdtypes.html#str

²⁶⁵ https://docs.python.org/3/library/stdtypes.html#str

²⁶⁶ https://docs.python.org/3/library/stdtypes.html#list

²⁶⁷ https://docs.python.org/3/library/stdtypes.html#str ²⁶⁸ https://docs.python.org/3/library/stdtypes.html#str

²⁶⁹ https://docs.python.org/3/library/stdtypes.html#str

²⁷⁰ https://docs.python.org/3/library/stdtypes.html#list

https://docs.python.org/3/library/stdtypes.html#str

```
supported_image_types: list<sup>275</sup>[str<sup>276</sup>] = ['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 https://idpf.org/epub or https://en.wikipedia.org/wiki/EPUB. The builder creates *EPUB 3* files.

```
name: str^{277} = 'epub'
```

The builder's name. This is the value used to select builders on the command line.

```
format: str^{278} = 'html'
```

The builder's output format, or "if no document output is produced. This is commonly the file extension, e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as *SphinxPostTransform* or extensions to determine their compatibility with the builder.

```
supported_image_types: list<sup>279</sup>[str<sup>280</sup>] = ['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.

Added in version 1.4.

Changed in version 1.5: Since Sphinx 1.5, the epub3 builder is used as the default epub builder.

class sphinx.builders.latex.LaTeXBuilder

This builder produces LaTeX source files in the output directory. The actual PDF builds happen inside this output directory and need to be triggered in a second step. This can be done via **make all-pdf** there. To combine the two steps into only one, use sphinx-build - M (i.e. -M latexpdf not -b latexpdf) or **make latexpdf** at the project root.

See latex_documents and the chapter Options for LaTeX output for available options.

PDF builds need a sufficiently complete LaTeX installation. The testing is currently (since 5.3.0) done on Ubuntu 22.04LTS, whose LaTeX distribution matches upstream TeXLive 2021 as of 2022/02/04, but PDF builds can be successfully done on much older LaTeX installations.

At any rate, on Ubuntu for example, following packages must all be present:

- texlive-latex-recommended
- texlive-fonts-recommended
- texlive-fonts-extra (needed for fontawesome5, see the 7.4.0 change notice below)
- tex-gyre (if latex_engine left to default)
- texlive-latex-extra
- latexmk

4.5. Builders 151

²⁷² https://wiki.gnome.org/Apps/Devhelp

²⁷³ https://docs.python.org/3/library/stdtypes.html#str

²⁷⁴ https://docs.python.org/3/library/stdtypes.html#str

²⁷⁵ https://docs.python.org/3/library/stdtypes.html#list

²⁷⁶ https://docs.python.org/3/library/stdtypes.html#str

²⁷⁷ https://docs.python.org/3/library/stdtypes.html#str

https://docs.python.org/3/library/stdtypes.html#str

²⁷⁹ https://docs.python.org/3/library/stdtypes.html#list

²⁸⁰ https://docs.python.org/3/library/stdtypes.html#str

Changed in version 4.0.0: TeX Gyre fonts now required for 'pdflatex' engine (default).

Changed in version 7.4.0: LaTeX package xcolor is now required (it is part of Ubuntu texlive-latex-recommended anyhow). The LaTeX package fontawesome5 is recommended. See the 'sphinxsetup' iconpackage key for more.

Additional packages are needed in some circumstances:

- texlive-lang-cyrillic for Cyrillic (and also then cm-super if using the default fonts),
- texlive-lang-greek for Greek (and also then cm-super if using the default fonts),
- texlive-xetex if latex_engine is 'xelatex',
- texlive-luatex if latex_engine is 'lualatex',
- fonts-freefont-otf if *latex_engine* is either 'xelatex' or 'lualatex'.

1 Note

Since 1.6, make latexpdf uses on GNU/Linux and macOS latexmk, as it makes sure the needed number of runs is automatically executed. On Windows the PDF builds execute a fix number of LaTeX runs (three, then makeindex, then two more).

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: str^{281} = 'latex'
```

The builder's name. This is the value used to select builders on the command line.

```
format: str^{282} = 'latex'
```

The builder's output format, or 'if no document output is produced. This is commonly the file extension, e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as *SphinxPostTransform* or extensions to determine their compatibility with the builder.

```
supported_image_types: list<sup>283</sup>[str<sup>284</sup>] = ['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 rinohtype²⁸⁵. The builder's name is rinoh. Refer to the rinohtype manual²⁸⁶ for details.

²⁸¹ https://docs.python.org/3/library/stdtypes.html#str

²⁸² https://docs.python.org/3/library/stdtypes.html#str

²⁸³ https://docs.python.org/3/library/stdtypes.html#list

²⁸⁴ https://docs.python.org/3/library/stdtypes.html#str

²⁸⁵ https://github.com/brechtm/rinohtype

²⁸⁶ https://www.mos6581.org/rinohtype/master/quickstart.html#sphinx-builder

class sphinx.builders.text.TextBuilder

This builder produces a text file for each reStructuredText file. This is almost the same as the reStructuredText source, but with much of the markup stripped for better readability.

```
name: str^{287} = 'text'
```

The builder's name. This is the value used to select builders on the command line.

```
format: str^{288} = 'text'
```

The builder's output format, or 'if no document output is produced. This is commonly the file extension, e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as *SphinxPostTransform* or extensions to determine their compatibility with the builder.

```
supported_image_types: list<sup>289</sup>[str<sup>290</sup>] = []
```

The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

Added 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: str^{291} = 'man'
```

The builder's name. This is the value used to select builders on the command line.

```
format: str^{292} = 'man'
```

The builder's output format, or 'if no document output is produced. This is commonly the file extension, e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as *SphinxPostTransform* or extensions to determine their compatibility with the builder.

```
supported_image_types: list<sup>293</sup>[str<sup>294</sup>] = []
```

The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

Added 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 <code>texinfo_documents</code> 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: str^{295} = 'texinfo'
```

The builder's name. This is the value used to select builders on the command line.

4.5. Builders 153

 $^{^{287}\} https://docs.python.org/3/library/stdtypes.html\#str$

²⁸⁸ https://docs.python.org/3/library/stdtypes.html#str

²⁸⁹ https://docs.python.org/3/library/stdtypes.html#list

²⁹⁰ https://docs.python.org/3/library/stdtypes.html#str

²⁹¹ https://docs.python.org/3/library/stdtypes.html#str

²⁹² https://docs.python.org/3/library/stdtypes.html#str

²⁹³ https://docs.python.org/3/library/stdtypes.html#list

²⁹⁴ https://docs.python.org/3/library/stdtypes.html#str

```
format: str<sup>296</sup> = 'texinfo'
```

The builder's output format, or "if no document output is produced. This is commonly the file extension, e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as *SphinxPostTransform* or extensions to determine their compatibility with the builder.

```
supported_image_types: list<sup>297</sup>[str<sup>298</sup>] = ['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.

Added 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 serialization²⁹⁹ 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, physerialize, 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.

Added in version 0.5.

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.

²⁹⁵ https://docs.python.org/3/library/stdtypes.html#str

²⁹⁶ https://docs.python.org/3/library/stdtypes.html#str

²⁹⁷ https://docs.python.org/3/library/stdtypes.html#list

²⁹⁸ https://docs.python.org/3/library/stdtypes.html#str

²⁹⁹ https://pypi.org/project/phpserialize/

```
name: str<sup>300</sup> = 'pickle'
```

The builder's name. This is the value used to select builders on the command line.

The old name web still works as well.

```
format: str<sup>301</sup> = 'html'
```

The builder's output format, or "if no document output is produced. This is commonly the file extension, e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as *SphinxPostTransform* or extensions to determine their compatibility with the builder.

```
supported_image_types: list<sup>302</sup>[str<sup>303</sup>] = ['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: str^{304} = 'json'
```

The builder's name. This is the value used to select builders on the command line.

```
format: str<sup>305</sup> = 'html'
```

The builder's output format, or 'if no document output is produced. This is commonly the file extension, e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as *SphinxPostTransform* or extensions to determine their compatibility with the builder.

```
supported\_image\_types: list^{306}[str^{307}] = ['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.

Added 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: str^{308} = 'gettext'
```

The builder's name. This is the value used to select builders on the command line.

4.5. Builders 155

 $^{^{300}\} https://docs.python.org/3/library/stdtypes.html\#str$

³⁰¹ https://docs.python.org/3/library/stdtypes.html#str

³⁰² https://docs.python.org/3/library/stdtypes.html#list

³⁰³ https://docs.python.org/3/library/stdtypes.html#str

³⁰⁴ https://docs.python.org/3/library/stdtypes.html#str

³⁰⁵ https://docs.python.org/3/library/stdtypes.html#str

³⁰⁶ https://docs.python.org/3/library/stdtypes.html#list

³⁰⁷ https://docs.python.org/3/library/stdtypes.html#str

```
format: str^{309} = "
```

The builder's output format, or "if no document output is produced. This is commonly the file extension, e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as *SphinxPostTransform* or extensions to determine their compatibility with the builder.

```
supported_image_types: list<sup>310</sup>[str<sup>311</sup>] = []
```

The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

Added in version 1.1.

class sphinx.builders.changes.ChangesBuilder

This builder produces an HTML overview of all *versionadded*, *versionchanged*, *deprecated* and *versionremoved* directives for the current *version*. This is useful to generate a changelog file, for example.

```
name: str^{312} = 'changes'
```

The builder's name. This is the value used to select builders on the command line.

```
format: str^{313} = ''
```

The builder's output format, or 'if no document output is produced. This is commonly the file extension, e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as *SphinxPostTransform* or extensions to determine their compatibility with the builder.

```
supported_image_types: list<sup>314</sup>[str<sup>315</sup>] = []
```

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: str^{316} = 'dummy'
```

The builder's name. This is the value used to select builders on the command line.

```
supported_image_types: list<sup>317</sup>[str<sup>318</sup>] = []
```

The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

Added 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: str<sup>319</sup> = 'linkcheck'
```

The builder's name. This is the value used to select builders on the command line.

³⁰⁸ https://docs.python.org/3/library/stdtypes.html#str

³⁰⁹ https://docs.python.org/3/library/stdtypes.html#str

³¹⁰ https://docs.python.org/3/library/stdtypes.html#list

³¹¹ https://docs.python.org/3/library/stdtypes.html#str

https://docs.python.org/3/library/stdtypes.html#str

³¹³ https://docs.python.org/3/library/stdtypes.html#str

³¹⁴ https://docs.python.org/3/library/stdtypes.html#list

³¹⁵ https://docs.python.org/3/library/stdtypes.html#str

³¹⁶ https://docs.python.org/3/library/stdtypes.html#str

³¹⁷ https://docs.python.org/3/library/stdtypes.html#list

³¹⁸ https://docs.python.org/3/library/stdtypes.html#str

```
format: str^{320} = ''
```

The builder's output format, or 'if no document output is produced. This is commonly the file extension, e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as *SphinxPostTransform* or extensions to determine their compatibility with the builder.

```
supported_image_types: list<sup>321</sup>[str<sup>322</sup>] = []
```

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 uses the 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: str^{323} = 'xml'
```

The builder's name. This is the value used to select builders on the command line.

```
format: str^{324} = 'xml'
```

The builder's output format, or 'if no document output is produced. This is commonly the file extension, e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as *SphinxPostTransform* or extensions to determine their compatibility with the builder.

```
supported_image_types: list<sup>325</sup>[str<sup>326</sup>] = []
```

The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

Added 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: str<sup>327</sup> = 'pseudoxml'
```

The builder's name. This is the value used to select builders on the command line.

```
format: str<sup>328</sup> = 'pseudoxml'
```

The builder's output format, or "if no document output is produced. This is commonly the file extension, e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as *SphinxPostTransform* or extensions to determine their compatibility with the builder.

```
supported_image_types: list<sup>329</sup>[str<sup>330</sup>] = []
```

The list of MIME types of image formats supported by the builder. Image files are searched in the order in which they appear here.

Added in version 1.2.

4.5. Builders 157

³¹⁹ https://docs.python.org/3/library/stdtypes.html#str

³²⁰ https://docs.python.org/3/library/stdtypes.html#str

³²¹ https://docs.python.org/3/library/stdtypes.html#list

³²² https://docs.python.org/3/library/stdtypes.html#str

³²³ https://docs.python.org/3/library/stdtypes.html#str

³²⁴ https://docs.python.org/3/library/stdtypes.html#str

³²⁵ https://docs.python.org/3/library/stdtypes.html#list

³²⁶ https://docs.python.org/3/library/stdtypes.html#str

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 reStructuredText source files to the _sources directory 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 .

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.

³²⁷ https://docs.python.org/3/library/stdtypes.html#str

³²⁸ https://docs.python.org/3/library/stdtypes.html#str

³²⁹ https://docs.python.org/3/library/stdtypes.html#list

³³⁰ https://docs.python.org/3/library/stdtypes.html#str

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.

Unlike the other pickle files this pickle file requires that the sphinx package is available on unpickling.

4.6 Domains

Added 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 *directives* and *roles*) to describe and link to *objects* 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 entities 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 :no-index-entry:. If you want to exclude the object description from the table of contents, you can give the directive option flag :no-contents-entry:. If you want to typeset an object description, without even making it available for cross-referencing, you can give the directive option flag :no-index: (which implies :no-index-entry:). If you do not want to typeset anything, you can give the directive option flag :no-typesetting:. This can for example be used to create only a target and index entry for later reference. Though, note that not every directive in every domain may support these options.

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

Added in version 5.2.3: The directive option :nocontentsentry: in the Python, C, C++, Javascript, and reStructuredText domains.

4.6. Domains 159

Added in version 7.2: The directive option no-typesetting in the Python, C, C++, Javascript, and reStructuredText domains.

Changed in version 7.2:

- The directive option :noindex: was renamed to :no-index:.
- The directive option :noindexentry: was renamed to :no-index-entry:.
- The directive option :nocontentsentry: was renamed to :no-contents-entry:.

The previous names are retained as aliases, but will be deprecated and removed in a future version of Sphinx.

An example using a Python domain directive:

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:

(This example also shows how to use the :no-index: 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.

The directive option :no-typesetting: can be used to create a target (and index entry) which can later be referenced by the roles provided by the domain. This is particularly useful for literate programming:

```
.. py:function:: spam(eggs)
    :no-typesetting:
.. code:: python
    def spam(eggs):
        pass
The function :py:func:`spam` does nothing.
```

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 *cross-referencing modifiers* exist as for general cross-references. In short:

- You may supply an explicit title and reference target: :py:mod:`mathematical functions <math>` will refer to the math module, but the link text will be "mathematical functions".
- If you prefix the content with an exclamation mark (!), 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.get` will refer to queue.Queue.get but only display get as the link text.

Built-in domains

The following domains are included within Sphinx:

The Standard Domain

Added in version 1.0.

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:`-m`, or :option:`-module`).

Changed in version 5.3: One can cross-reference including an option value: :option:`--module=foobar`, ``:option:-module[=foobar]``or:option:`--module foobar`.

Use *option_emphasise_placeholders* for parsing of "variable part" of a literal text (similarly to the *samp* role).

4.6. Domains 161

cmdoption directive is a deprecated alias for the option directive.

.. confval:: name

Describes a configuration value or setting that the documented code or program uses or defines. Referenceable by *confval*.

:type: (text)

Describes the type of the configuration value. This is optional, and if specified will be interpreted as reStructuredText.

:default: (text)

Describes the default value of the configuration value. This is optional, and if specified will be interpreted as reStructuredText.

Example:

```
.. confval:: the_answer
   :type: ``int`` (a *number*)
   :default: **42**

This is a setting that controls the value of the answer.
```

will be rendered as follows:

the_answer

```
Type
int (a number)

Default
42
```

This is a setting that controls the value of the answer.

.. 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.

If None is passed to the argument, the directive will reset the current program name.

The program name may contain spaces (in case you want to document subcommands like svn add and svn commit separately).

Added 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 C Domain

Added in version 1.0.

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

- \dots **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 reStructuredText inliner.

In the description of a function you can use the following info fields (see also *Info field lists*).

- param, parameter, arg, argument, Description of a parameter.
- type: Type of a parameter, written as if passed to the *c:expr* role.
- returns, return: Description of the return value.
- rtype: Return type, written as if passed to the *c:expr* role.
- retval, retvals: An alternative to returns for describing the result of the function.

Added in version 4.3: The retval field type.

For example:

```
.. c:function:: PyObject *PyType_GenericAlloc(PyTypeObject *type, Py_ssize_t nitems)
:param type: description of the first parameter.
:param nitems: description of the second parameter.
```

(continues on next page)

4.6. Domains 163

(continued from previous page)

```
:returns: a result.
:retval NULL: under some conditions.
:retval NULL: under some other conditions as well.
```

which renders as

```
PyObject<sup>331</sup> *PyType_GenericAlloc(PyTypeObject<sup>332</sup> *type, Py_ssize_t<sup>333</sup> nitems)
```

No-contents-entry

No-index-entry

Parameters

- **type** description of the first parameter.
- **nitems** description of the second parameter.

Returns

a result.

Return values

- **NULL** under some conditions.
- NULL under some other conditions as well.

:single-line-parameter-list: (no value)

Ensures that the function's parameters will be emitted on a single logical line, overriding c_maximum_signature_line_length and maximum_signature_line_length.

Added in version 7.1.

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

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

In the description of a macro you can use the same info fields as for the c:function directive.

Added in version 3.0: The function style variant.

:single-line-parameter-list: (no value)

Ensures that the macro's parameters will be emitted on a single logical line, overriding c_maximum_signature_line_length and maximum_signature_line_length.

Added in version 7.1.

.. c:struct:: name

Describes a C struct.

Added in version 3.0.

.. c:union:: name

Describes a C union.

Added in version 3.0.

³³¹ https://docs.python.org/3/c-api/structures.html#c.PyObject

³³¹ https://docs.python.org/3/c-api/structures.html#c.PyObject

³³² https://docs.python.org/3/c-api/type.html#c.PyTypeObject

³³³ https://docs.python.org/3/c-api/intro.html#c.Py_ssize_t

```
.. c:enum:: name
     Describes a C enum.
     Added in version 3.0.
.. c:enumerator:: name
     Describes a C enumerator.
     Added 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.
```

Added 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

4.6. Domains 165

```
union [anonymous]
int a
double b

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

becomes

int data

int **f**(double k)

int data

int f (double k)

Added 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.

Added in version 3.3.

:noroot:

Skip the mentioned declarations and only render nested declarations. Requires \max depth either 0 or at least 2.

Added 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*).
Added in version 3.0.
```

Namespacing

Added 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:

4.6. Domains 167

```
.. 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

Added in version 1.0.

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:

Full and partial template specialisations can be declared:

Added 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.
.. 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:

and function template specialisations:

:single-line-parameter-list: (no value)

Ensures that the function's parameters will be emitted on a single logical line, overriding cpp_maximum_signature_line_length and maximum_signature_line_length.

Added in version 7.1.

- .. 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: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:

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: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.

Added in version 1.8.

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

A 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**
```

(continues on next page)

(continued from previous page)

```
- :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

 $It \mathbf{r}$

An Ivalue.

Valid Expressions

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

Added in version 1.5.

Options

Some directives support options:

- :no-index-entry: and :no-contents-entry:, see *Basic Markup*.
- :tparam-line-spec:, for templated declarations. If specified, each template parameter will be rendered on a separate line.

Added 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.
Added 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
```

 $\operatorname{void} f(\operatorname{double} d) \operatorname{const}$

void *f* (double d)

void *f* (int i)

 $\operatorname{void} f()$

whereas:

becomes

```
\operatorname{void} f(\operatorname{double} d) \operatorname{const}
```

void *f* (double d)

Added 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.

Added in version 3.5.

:noroot:

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

Added 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.



1 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>&).
Added in version 1.7: The cpp:expr role.
Added 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.,:

declares size as a member function of the class template std::vector. Equivalently this could have been declared using:

or:

.. 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.

Added 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
```

Added in version 1.4.

Info field lists

All the C++ directives for declaring entities support the following info fields (see also *Info field lists*):

• tparam: Description of a template parameter.

The *cpp:function* directive additionally supports the following fields:

- param, parameter, arg, argument: Description of a parameter.
- returns, return: Description of a return value.
- retval, retvals: An alternative to returns for describing the result of the function.
- throws, throw, exception: Description of a possibly thrown exception.

Added in version 4.3: The retval field type.

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.

Added in version 2.0: The *cpp:struct* role as alias for the *cpp:class* role.

1 Note on References with Templates Parameters/Arguments

These roles follow the Sphinx *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>). As a shorthand the empty template parameter list can be omitted, e.g., Outer\<int>:(Outer<int>) and 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*>> Currently the lookup only succeed if the template parameter identifiers are equal strings.

Configuration Variables

See *Options for the C++ domain*.

The JavaScript Domain

Added in version 1.0.

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 no-index option is specified. If this option is specified, the directive will only update the current module name

Added in version 1.6.

Changed in version 5.2: Module directives support body content.

.. 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.

:single-line-parameter-list: (no value)

Ensures that the function's parameters will be emitted on a single logical line, overriding <code>javascript_maximum_signature_line_length</code> and <code>maximum_signature_line_length</code>.

Added in version 7.1.

.. 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.

Added in version 1.6.

:single-line-parameter-list: (no value)

Ensures that the function's parameters will be emitted on a single logical line, overriding <code>javascript_maximum_signature_line_length</code> and <code>maximum_signature_line_length</code>.

Added in version 7.1.

.. 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

:single-line-parameter-list: (no value)

Ensures that the function's parameters will be emitted on a single logical line, overriding <code>javascript_maximum_signature_line_length</code> and <code>maximum_signature_line_length</code>.

Added in version 7.1.

```
.. 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 Mathematics Domain

Added in version 1.8.

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.
```

Added in version 1.8.

The Python Domain

Added in version 1.0.

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). A description of the module such as the docstring can be placed in the body of the directive.

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

Changed in version 5.2: Module directives support body content.

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 <code>py:module</code>), but it will not create index entries, an entry in the Global Module Index, or a link target for <code>py:mod</code>. This is helpful in situations where documentation for things in a module is spread over multiple files or sections – one location has the <code>py:module</code> directive, the others only <code>py:currentmodule</code>.

The following directives are provided for module and class contents:

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

.. py:function:: name[type parameters](parameters)

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

```
.. py:function:: Timer.repeat(repeat=3, number=1_000_000)
.. py:function:: add[T](a: T, b: T) -> T
```

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.

Added 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

Added in version 4.0.

:single-line-parameter-list: (no value)

Ensures that the function's arguments will be emitted on a single logical line, overriding python_maximum_signature_line_length and maximum_signature_line_length.

Added in version 7.1.

:single-line-type-parameter-list: (no value)

Ensure that the function's type parameters are emitted on a single logical line, overriding python_maximum_signature_line_length and maximum_signature_line_length.

Added in version 7.1.

.. py:data:: name

Describes global data in a module, including both variables and values used as "defined constants." Consider using *py:type* for type aliases instead and *py:attribute* for class variables and instance attributes.

options

:type: type of the variable (text)

Added in version 2.4.

```
Added 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
          Added in version 4.0.
.. py:exception:: name
.. py:exception:: name(parameters)
.. py:exception:: name[type parameters](parameters)
     Describes an exception class. The signature can, but need not include parentheses with constructor arguments,
     or may optionally include type parameters (see PEP 695<sup>334</sup>).
     options
     :final: (no value)
          Indicate the class is a final class.
          Added in version 3.1.
     :single-line-parameter-list: (no value)
          See py:class:single-line-parameter-list.
          Added in version 7.1.
     :single-line-type-parameter-list: (no value)
          See py:class:single-line-type-parameter-list.
          Added in version 7.1.
.. py:class:: name
.. py:class:: name(parameters)
.. py:class:: name[type parameters](parameters)
     Describes a class. The signature can optionally include type parameters (see PEP 695<sup>335</sup>) or 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()
```

:value: initial value of the variable (text)

The first way is the preferred one.

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

.. py:class:: Bar

-- or --

³³⁴ https://peps.python.org/pep-0695/

options

:canonical: (full qualified name including module name)

Describe the location where the object is defined if the object is imported from other modules

Added in version 4.0.

:final: (no value)

Indicate the class is a final class.

Added in version 3.1.

:single-line-parameter-list: (no value)

Ensures that the class constructor's arguments will be emitted on a single logical line, overriding python_maximum_signature_line_length and maximum_signature_line_length.

Added in version 7.1.

:single-line-type-parameter-list: (no value)

Ensure that the class type parameters are emitted on a single logical line, overriding python_maximum_signature_line_length and maximum_signature_line_length.

.. 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. Type aliases should be documented with *py:type*.

options

:type: type of the attribute (text)

Added in version 2.4.

:value: initial value of the attribute (text)

Added 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

Added in version 4.0.

.. py:property:: name

Describes an object property.

Added in version 4.0.

options

:abstractmethod: (no value)

Indicate the property is abstract.

:classmethod: (no value)

Indicate the property is a classmethod.

Added in version 4.2.

:type: type of the property (text)

³³⁵ https://peps.python.org/pep-0695/

.. py:type:: name

Describe a type alias³³⁶.

The type that the alias represents should be described with the canonical option. This directive supports an optional description body.

For example:

```
.. py:type:: UInt64

Represent a 64-bit positive integer.
```

will be rendered as follows:

type UInt64

Represent a 64-bit positive integer.

options

:canonical: (text)

The canonical type represented by this alias, for example:

```
.. py:type:: StrPattern
:canonical: str | re.Pattern[str]

Represent a regular expression or a compiled pattern.
```

This is rendered as:

```
type StrPattern = str<sup>337</sup> | re.Pattern<sup>338</sup>[str<sup>339</sup>]
```

Represent a regular expression or a compiled pattern.

Added in version 7.4.

- .. py:method:: name(parameters)
- .. py:method:: name[type parameters](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.

Added in version 2.1.

:async: (no value)

Indicate the method is an async method.

Added 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

Added in version 4.0.

 $^{^{336}\} https://docs.python.org/3/library/typing.html\#type-aliases$

³³⁷ https://docs.python.org/3/library/stdtypes.html#str

³³⁸ https://docs.python.org/3/library/re.html#re.Pattern

³³⁹ https://docs.python.org/3/library/stdtypes.html#str

:classmethod: (no value)

Indicate the method is a class method.

Added in version 2.1.

:final: (no value)

Indicate the method is a final method.

Added in version 3.1.

:single-line-parameter-list: (no value)

Ensures that the method's arguments will be emitted on a single logical line, overriding python_maximum_signature_line_length and maximum_signature_line_length.

Added in version 7.1.

:single-line-type-parameter-list: (no value)

Ensure that the method's type parameters are emitted on a single logical line, overriding python_maximum_signature_line_length and maximum_signature_line_length.

Added in version 7.2.

:staticmethod: (no value)

Indicate the method is a static method.

Added in version 2.1.

- .. py:staticmethod:: name(parameters)
- .. py:staticmethod:: name[type parameters](parameters)

Like *py:method*, but indicates that the method is a static method.

Added in version 0.4.

- .. py:classmethod:: name(parameters)
- .. py:classmethod:: name[type parameters](parameters)

Like *py:method*, but indicates that the method is a class method.

Added in version 0.6.

- .. py:decorator:: name
- .. py:decorator:: name(parameters)
- .. py:decorator:: name[type parameters](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.

:single-line-parameter-list: (no value)

Ensures that the decorator's arguments will be emitted on a single logical line, overriding python_maximum_signature_line_length and maximum_signature_line_length.

Added in version 7.1.

:single-line-type-parameter-list: (no value)

Ensure that the decorator's type parameters are emitted on a single logical line, overriding python_maximum_signature_line_length and maximum_signature_line_length.

Added in version 7.2.

```
.. py:decoratormethod:: name
```

- .. py:decoratormethod:: name(signature)
- .. py:decoratormethod:: name[type parameters](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.

Python 3.12 introduced *type parameters*, which are type variables declared directly within the class or function definition:

```
class AnimalList[AnimalT](list[AnimalT]):
    ...
def add[T](a: T, b: T) -> T:
    return a + b
```

The corresponding reStructuredText documentation would be:

```
.. py:class:: AnimalList[AnimalT]
.. py:function:: add[T](a: T, b: T) -> T
```

See PEP 695³⁴⁰ and PEP 696³⁴¹ for details and the full specification.

Info field lists

Added in version 0.4.

Changed in version 3.0: meta fields are added.

Inside Python object description directives, reStructuredText 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.

1 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:

³⁴⁰ https://peps.python.org/pep-0695/

³⁴¹ https://peps.python.org/pep-0696/

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

Send a message to a recipient

Parameters

- **sender** (str^{342}) The person sending the message
- **recipient** (str^{343}) The recipient of the message
- **message_body** (str^{344}) The body of the message
- priority (int³⁴⁵ or None) The priority of the message, can be a number 1-5

Returns

the message id

Return type int³⁴⁶

Raises

- **ValueError**³⁴⁷ if the message_body exceeds 160 characters
- **TypeError**³⁴⁸ 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
```

Added 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
```

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.

```
342 https://docs.python.org/3/library/stdtypes.html#str
```

³⁴³ https://docs.python.org/3/library/stdtypes.html#str

³⁴⁴ https://docs.python.org/3/library/stdtypes.html#str

³⁴⁵ https://docs.python.org/3/library/functions.html#int

³⁴⁶ https://docs.python.org/3/library/functions.html#int

³⁴⁷ https://docs.python.org/3/library/exceptions.html#ValueError

³⁴⁸ https://docs.python.org/3/library/exceptions.html#TypeError

: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.

:pv: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.



1 Note

The role is also able to refer to property.

:py:type:

Reference a type alias.

: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*.

Added in version 0.4.

Target specification

The target can be specified as a fully qualified name (e.g. :py:meth:`my_module.MyClass.my_method`) or any shortened version (e.g. :py:meth: `MyClass.my_method` or :py:meth: `my_method`). See target resolution for details on the resolution of shortened names.

Cross-referencing modifiers can be applied. In short:

- You may supply an explicit title and reference target: :py:mod:`mathematical functions <math>` will refer to the math module, but the link text will be "mathematical functions".
- If you prefix the content with an exclamation mark (!), 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.

Target resolution

A given link target name is resolved to an object using the following strategy:

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 codecs³⁴⁹ module, :py:func:`open` always refers to the built-in function, while :py:func:`.open` refers to codecs.open()³⁵⁰.

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().

The reStructuredText Domain

Added in version 1.0.

The reStructuredText domain (name **rst**) provides the following directives:

.. rst:directive:: name

Describes a reStructuredText 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 reStructuredText 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:

³⁴⁹ https://docs.python.org/3/library/codecs.html#module-codecs

³⁵⁰ https://docs.python.org/3/library/codecs.html#codecs.open

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
```

Added in version 2.1.

.. rst:role:: name

Describes a reStructuredText 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:

Reference directives and directive options. Examples:

- Use :rst:dir:`my-directive` to reference a directive.
- Use :rst:dir:`my-directive:my-option` to reference a directive option.

:rst:role:

Reference a role. Example: :rst:role:`my-role`.

More domains

There are several third-party domains available as extensions, including:

- Ada³⁵¹
- Chapel³⁵²
- CoffeeScript³⁵³
- Common Lisp³⁵⁴
- dqn³⁵⁵
- Erlang³⁵⁶
- Go³⁵⁷

³⁵¹ https://pypi.org/project/sphinxcontrib-adadomain/

³⁵² https://pypi.org/project/sphinxcontrib-chapeldomain/

³⁵³ https://pypi.org/project/sphinxcontrib-coffee/

³⁵⁴ https://pypi.org/project/sphinxcontrib-cldomain/

³⁵⁵ https://pypi.org/project/sphinxcontrib-dqndomain/

³⁵⁶ https://pypi.org/project/sphinxcontrib-erlangdomain/

³⁵⁷ https://pypi.org/project/sphinxcontrib-golangdomain/

- HTTP³⁵⁸
- Jinia³⁵⁹
- Lasso³⁶⁰
- MATLAB361
- Operation³⁶²
- PHP³⁶³
- Ruby³⁶⁴
- Scala³⁶⁵

4.7 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 Sphinx API.

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 semiautomatic way.



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 that the main routine is protected by an 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 NumPy³⁶⁶ or Google³⁶⁷ style docstrings over reStructuredText, you can also enable the napoleon extension. napoleon is a preprocessor that converts docstrings to correct reStructuredText before autodoc processes them.

4.7. Extensions 193

³⁵⁸ https://pypi.org/project/sphinxcontrib-httpdomain/

³⁵⁹ https://pypi.org/project/sphinxcontrib-jinjadomain/

³⁶⁰ https://pypi.org/project/sphinxcontrib-lassodomain/

³⁶¹ https://pypi.org/project/sphinxcontrib-matlabdomain/

³⁶² https://pypi.org/project/sphinxcontrib-operationdomain/

³⁶³ https://pypi.org/project/sphinxcontrib-phpdomain/

³⁶⁴ https://pypi.org/project/sphinxcontrib-rubydomain/

³⁶⁵ https://pypi.org/project/sphinxcontrib-scaladomain/

³⁶⁶ https://numpydoc.readthedocs.io/en/latest/format.html#docstring-standard

³⁶⁷ https://google.github.io/styleguide/pyguide.html#38-comments-and-docstrings

Getting started

Setup

Activate the plugin by adding 'sphinx.ext.autodoc' to the extensions list in conf.py:

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

Ensuring the code can be imported

autodoc analyses the code and docstrings by introspection after **importing the modules**. For importing to work, you have to make sure that your modules can be found by Sphinx and that dependencies can be resolved (if your module does **import** foo, but foo is not available in the python environment that Sphinx runs in, your module import will fail).

There are two ways to ensure this:

- 1. Use an environment that contains your package and Sphinx. This can e.g. be your local development environment (with an editable install), or an environment in CI in which you install Sphinx and your package. The regular installation process ensures that your package can be found by Sphinx and that all dependencies are available.
- 2. It is alternatively possible to patch the Sphinx run so that it can operate directly on the sources; e.g. if you want to be able to do a Sphinx build from a source checkout.
 - Patch sys.path³⁶⁸ in conf.py to include your source path. For example if you have a repository structure with doc/conf.py and your package is at src/my_package, then you should add the following to your conf.py.

```
import sys
from pathlib import Path

sys.path.insert(0, str(Path('...', 'src').resolve()))
```

To cope with missing dependencies, specify the missing modules in the autodoc_mock_imports setting.

Usage

You can now use the *Directives* to add formatted documentation for Python code elements like functions, classes, modules, etc. For example, to document the function io.open(), reading its signature and docstring from the source file, you'd write:

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

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

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

³⁶⁸ https://docs.python.org/3/library/sys.html#sys.path

```
    Ţip
```

As a hint to autodoc extension, you can put a :: separator between the module name and the object name to let autodoc know the correct module, if it is ambiguous:

```
.. autoclass:: module.name::Noodle
```

Marking objects as public or private

• 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:
    """
```

Added 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:
    """
```

Added 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:
```

Added in version 3.5.

Doc comments and docstrings

Python has no built-in support for docstrings for module data members or class attributes. To allow documenting these, autodoc recognises a special format of comment³⁶⁹ called a 'doc comment' or 'documentation comment'.

These comments start with a colon and an optional space character, '#:' or '#: '. To be recognised, the comments must appear either on the same line as the variable or on one or more lines before the variable. Multi-line doc-comments must always appear on the lines before the variable's definition.

For example, all three of the following variables have valid doc-comments:

```
egg_and_spam = 1.50 #: A simple meal

#: Spam! Lovely spam! Lovely spam!
egg_bacon_sausage_and_spam = 2.49

#: Truly gourmet cuisine for madam; Lobster Thermidor

(continues on next page)
```

4.7. Extensions

³⁶⁹ https://docs.python.org/3/reference/lexical_analysis.html#comments

(continued from previous page)

```
#: aux Crevettes with a mornay sauce garnished with
#: truffle pate, brandy and a fried egg on top and spam.
lobster_thermidor = 35.00
```

Alternatively, autodoc can recognise a docstring on the line immediately following the definition.

In the the following class definition, all attributes have documentation recognised by autodoc:

```
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."""
```

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.



Just as py:class respects the current py:module, autoclass will also do so. Likewise, automethod will respect the current py:class.

Default directive options

To make any of the options described below the default, use the autodoc_default_options dictionary in conf.py.

If using defaults for the :members:, :exclude-members:, :private-members:, or :special-members: options, setting the option on a directive will override the default. Instead, to extend the default list with the per-directive option, the list may be prepended with a plus sign (+), as follows:

```
.. autoclass:: Noodle
:members: eat
:private-members: +_spicy, _garlickly
```

🗘 Tip

If using <code>autodoc_default_options</code>, the defaults can be disabled per-directive with the negated form, <code>:no-option:</code> as an option of the directive For example:

```
.. automodule:: foo :no-undoc-members:
```

Automatically document modules

.. automodule::

Document a module. By default, the directive only inserts the docstring of the module itself:

```
.. automodule:: noodles
```

will produce source like this:

```
.. py:module:: noodles

The noodles module.
```

The directive can also contain content of its own, which 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, as follows:

```
.. automodule:: noodles
   :members: Noodle
   .. py:function:: boiled_noodle(time=10)

   Create a noodle that has been boiled for *time* minutes.
```

Options

:no-index:

Do not generate an index entry for the documented module or any auto-documented members.

Added in version 0.4.

:platform: platforms (comma separated list)

Indicate platforms on which the module is available. This is identical to py:module's :platform: option.

:synopsis: purpose (text)

A sentence describing the module's purpose. This is identical to *py:module*'s :synopsis: option.

Added in version 0.5.

:deprecated:

Mark a module as deprecated. This is identical to py:module's :deprecated: option.

Added in version 0.5.

:ignore-module-all: (no value)

Do not use __all__ when analysing the module to document.

Added in version 1.7.

4.7. Extensions

Options for selecting members to document

:members: (no value or comma separated list)

Generate automatic documentation for all members of the target module:

```
.. automodule:: noodles
:members:
```

By default, autodoc only includes public members with a docstring or *doc-comment* (#:). If __all__ exists, it will be used to define which members are public, unless the <code>:ignore-module-all:</code> option is set.

To only document certain members, an explicit comma-separated list may be used as the argument to :members::

```
.. automodule:: noodles
:members: Noodle
```

:exclude-members: (comma separated list)

Exclude the given names from the members to document. For example:

```
.. automodule:: noodles
:members:
:exclude-members: NoodleBase
```

Added in version 0.6.

:imported-members: (no value)

To prevent documentation of imported classes or functions, in an automodule directive with the members option set, only module members where the __module__ attribute is equal to the module name given to automodule will be documented.

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.

Added in version 1.2.

:undoc-members:

Generate automatic documentation for members of the target module that don't have a docstring or doccomment. For example:

```
.. automodule:: noodles
:members:
:undoc-members:
```

:private-members: (no value or comma separated list)

Generate automatic documentation for private members of the target module. This includes names with a leading underscore (e.g. _private) and those members explicitly marked as private with :meta private:.

```
.. automodule:: noodles
:members:
:private-members:
```

To only document certain private members, an explicit comma-separated list may be used as the argument to :private-members::

```
.. automodule:: noodles
:members:
:private-members: _spicy, _garlickly
```

Added in version 1.1.

Changed in version 3.2: The option can now take a comma-separated list of arguments.

:special-members: (no value or comma separated list)

Generate automatic documentation for special members of the target module, also known as 'dunder' names³⁷⁰. This includes all names enclosed with a double-underscore, e.g. __special__:

```
.. automodule:: my.Class
:members:
:special-members:
```

To only document certain special members, an explicit comma-separated list may be used as the argument to :special-members::

```
.. automodule:: noodles
:members:
:special-members: __version__
```

Added in version 1.1.

Changed in version 1.2: The option can now take a comma-separated list of arguments.

Options for documented members

:member-order: (alphabetical, bysource, or groupwise)

Choose the ordering of automatically documented members (default: alphabetical). This overrides the autodoc_member_order setting.

- alphabetical: Use simple alphabetical order.
- groupwise: Group by object type (class, function, etc), use alphabetical order within groups.
- bysource: Use the order of objects in the module's source. The __all__ variable can be used to override this order.

Note that for source order, the module must be a Python module with the source code available.

Added in version 0.6.

Changed in version 1.0: Support the 'bysource' option.

:show-inheritance: (no value)

Enable the :show-inheritance: option for all members of the module, if :members: is enabled.

Added in version 0.4.

4.7. Extensions

³⁷⁰ https://docs.python.org/3/reference/datamodel.html#specialnames

Automatically document classes or exceptions

.. autoclass::

.. autoexception::

Document a class. For exception classes, prefer .. autoexception::. By default, the directive only inserts the docstring of the class itself:

```
.. autoclass:: noodles.Noodle
```

will produce source like this:

```
.. py:class:: Noodle

The Noodle class's docstring.
```

The directive can also contain content of its own, which 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, as follows:

```
.. autoclass:: noodles.Noodle
  :members: eat, slurp

.. py:method:: boil(time=10)

Boil the noodle for *time* minutes.
```

Advanced usage

• It is 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:: noodles.Noodle(type)
.. automethod:: eat(persona)
```

This is useful if the signature from the method is hidden by a decorator.

Added in version 0.4.

Options

:no-index:

Do not generate an index entry for the documented class or any auto-documented members.

Added in version 0.4.

```
:class-doc-from: (class, init, or both)
```

Select which docstring will be used for the main body of the directive. This overrides the global value of *autoclass_content*. The possible values are:

- class: Only use the class's docstring. The __init__() method can be separately documented using the :members: option or automethod.
- init: Only use the docstring of the __init__() method.
- both: Use both, appending the docstring of the __init__() method to the class's docstring.

If the __init__() method doesn't exist or has a blank docstring, autodoc will attempt to use the __new__() method's docstring, if it exists and is not blank.

Added in version 4.1.

Options for selecting members to document

:members: (no value or comma separated list)

Generate automatic documentation for all members of the target class:

```
.. autoclass:: noodles.Noodle :members:
```

By default, autodoc only includes public members with a docstring or *doc-comment* (#:) that are attributes of the target class (i.e. not inherited).

To only document certain members, an explicit comma-separated list may be used as the argument to :members::

```
.. autoclass:: noodles.Noodle
:members: eat, slurp
```

:exclude-members: (comma separated list)

Exclude the given names from the members to document. For example:

```
.. autoclass:: noodles.Noodle
:members:
:exclude-members: prepare
```

Added in version 0.6.

:inherited-members: (comma separated list)

To generate automatic documentation for members inherited from base classes, use the :inherited-members: option:

```
.. autoclass:: noodles.Noodle
:members:
:inherited-members:
```

This can be combined with the :undoc-members: option to generate automatic documentation for *all* available members of the class.

The members of classes listed in the argument to :inherited-members: are excluded from the automatic documentation. This defaults to object³⁷¹ if no argument is provided, meaning that members of the object class are not documented. To include these, use None as the argument.

For example; If your class MyList 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.

1 Note

Should any of the inherited members use a format other than reStructuredText for their docstrings, there may be markup warnings or errors.

Added in version 0.3.

4.7. Extensions 201

Changed in version 3.0: :inherited-members: now takes the name of a base class to exclude as an argument.

Changed in version 5.0: A comma separated list of base class names can be used.

:undoc-members: (no value)

Generate automatic documentation for members of the target class that don't have a docstring or doccomment. For example:

```
.. autoclass:: noodles.Noodle
   :members:
   :undoc-members:
```

:private-members: (no value or comma separated list)

Generate automatic documentation for private members of the target class. This includes names with a leading underscore (e.g. _private) and those members explicitly marked as private with :meta private:.

```
.. autoclass:: noodles.Noodle
   :members:
   :private-members:
```

To only document certain private members, an explicit comma-separated list may be used as the argument to :private-members::

```
.. autoclass:: noodles.Noodle
:members:
:private-members: _spicy, _garlickly
```

Added in version 1.1.

Changed in version 3.2: The option can now take arguments.

:special-members: (no value or comma separated list)

Generate automatic documentation for special members of the target class, also known as 'dunder' names³⁷². This includes all names enclosed with a double-underscore, e.g. __special__:

```
.. autoclass:: noodles.Noodle
   :members:
   :special-members:
```

To only document certain special members, an explicit comma-separated list may be used as the argument to :special-members::

```
.. autoclass:: noodles.Noodle
   :members:
   :special-members: __init__, __name__
```

Added in version 1.1.

Changed in version 1.2: The option can now take a comma-separated list of arguments.

Options for documented members

:member-order: (alphabetical, bysource, or groupwise)

Choose the ordering of automatically documented members (default: alphabetical). This overrides the <code>autodoc_member_order</code> setting.

- 'alphabetical': Use simple alphabetical order.
- 'groupwise': Group by object type (class, function, etc), use alphabetical order within groups.
- 'bysource': Use the order of objects in the module's source. The __all__ variable can be used to override this order.

Note that for source order, the module must be a Python module with the source code available.

Added in version 0.6.

Changed in version 1.0: Support the 'bysource' option.

:show-inheritance: (no value)

Insert the class's base classes after the class signature.

Added in version 0.4.

Automatically document function-like objects

```
.. autofunction::
```

- .. automethod::
- .. autoproperty::
- .. autodecorator::

Document a function, method, property, or decorator. By default, the directive only inserts the docstring of the function itself:

```
.. autofunction:: noodles.average_noodle
```

will produce source like this:

```
.. py:function:: noodles.average_noodle
The average_noodle function's docstring.
```

The directive can also contain content of its own, which will be inserted into the resulting non-auto directive source after the docstring.

Therefore, you can also mix automatic and non-automatic documentation, as follows:

```
.. autofunction:: noodles.average_noodle
.. note:: For more flexibility, use the :py:class:`!Noodle` class.
```

Added in version 2.0: autodecorator.

Added in version 2.1: autoproperty.



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.

4.7. Extensions 203

 $^{^{371}\} https://docs.python.org/3/library/functions.html\#object$

³⁷² https://docs.python.org/3/reference/datamodel.html#specialnames

Advanced usage

• It is 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.

Added in version 0.4.

Options

:no-index:

Do not generate an index entry for the documented function.

Added in version 0.4.

Automatically document attributes or data

- .. autodata::
- .. autoattribute::

Document a module level variable or constant ('data') or class attribute. By default, the directive only inserts the docstring of the variable itself:

```
.. autodata:: noodles.FLOUR_TYPE
```

will produce source like this:

```
.. py:data:: noodles.FLOUR_TYPE

The FLOUR_TYPE constant's docstring.
```

The directive can also contain content of its own, which will be inserted into the resulting non-auto directive source after the docstring.

Therefore, you can also mix automatic and non-automatic member documentation, as follows:

```
.. autodata:: noodles.FLOUR_TYPE
.. hint:: Cooking time can vary by which flour type is used.
```

Changed in version 0.6: autodata and autoattribute can now extract docstrings.

Changed in version 1.1: Doc-comments are now allowed on the same line of an assignment.

Options

:no-index:

Do not generate an index entry for the documented class or any auto-documented members.

Added in version 0.4.

:annotation: value (string)

Added in version 1.2.

By default, autodoc attempts to obtain the type annotation and value of the variable by introspection, displaying it after the variable's name. To override this, a custom string for the variable's value may be used as the argument to annotation.

For example, if the runtime value of FILE_MODE is 0o755, the displayed value will be 493 (as oct(493) == '0o755'). This can be fixed by setting :annotation: = 0o755.

If :annotation: is used without arguments, no value or type hint will be shown for the variable.

:no-value:

Added in version 3.4.

To display the type hint of the variable without a value, use the :no-value: option. If both the :annotation: and :no-value: options are used, :no-value: has no effect.

Configuration

There are also config values that you can set:

autoclass_content

```
Type
str

Default
'class'
```

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. 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.

Added 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.

Added in version 1.4.

autodoc_class_signature

```
Type
str

Default
'mixed'
```

This value selects how the signature will be displayed for the class defined by *autoclass* directive. The possible values are:

'mixed'

Display the signature with the class name.

'separated'

Display the signature as a method.

Added in version 4.1.

autodoc_member_order

```
Type
str

Default
'alphabetical'
```

Define the order in which automodule and autoclass members are listed. Supported values are:

- 'alphabetical': Use alphabetical order.
- 'groupwise': order by member type. The order is:
 - for modules, exceptions, classes, functions, data
 - for classes: methods, then properties and attributes

Members are ordered alphabetically within groups.

• 'bysource': Use the order in which the members appear in the source code. This requires that the module must be a Python module with the source code available.

Added in version 0.6.

Changed in version 1.0: Support for 'bysource'.

autodoc_default_options

```
Type
    dict[str, str | bool]
Default
{}
```

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': See automodule: members.
- 'undoc-members' See automodule:undoc-members.
- 'private-members': See automodule:private-members.
- 'special-members': See automodule:special-members.
- 'inherited-members': See autoclass:inherited-members.
- 'imported-members': See automodule:imported-members.

```
• 'exclude-members': See automodule:exclude-members.
```

- 'ignore-module-all': See automodule:ignore-module-all.
- 'member-order': See automodule:member-order.
- 'show-inheritance': See autoclass:show-inheritance.
- 'class-doc-from': See autoclass:class-doc-from.
- 'no-value': See autodata:no-value.

Added in version 1.8.

Changed in version 2.0: Accepts True as a value.

Changed in version 2.1: Added 'imported-members'.

Changed in version 4.1: Added 'class-doc-from'.

Changed in version 4.5: Added 'no-value'.

autodoc_docstring_signature

```
Type bool
```

Default

True

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.

autodoc will continue to look for multiple signature lines, stopping at the first line that does not look like a signature. This is useful for declaring overloaded function signatures.

Added in version 1.1.

Changed in version 3.1: Support overloaded signatures

Changed in version 4.0: Overloaded signatures do not need to be separated by a backslash

autodoc_mock_imports

```
Type
list[str]

Default
```

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.

Added 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

```
Type
str

Default
'signature'
```

This value controls how to represent typehints. The setting takes the following values:

- 'signature' Show typehints in the signature
- 'description' Show typehints as content of the function or method The typehints of overloaded functions or methods will still be represented in the signature.
- 'none' Do not show typehints
- 'both' Show typehints in the signature and as content of the function or method

Overloaded functions or methods will not have typehints included in the description because it is impossible to accurately represent all possible overloads as a list of parameters.

Added in version 2.1.

Added in version 3.0: New option 'description' is added.

Added in version 4.1: New option 'both' is added.

autodoc_typehints_description_target

```
Type
str

Default
'all'
```

This value controls whether the types of undocumented parameters and return values are documented when <code>autodoc_typehints</code> is set to 'description'. Supported values:

- 'all': Types are documented for all parameters and return values, whether they are documented or not.
- 'documented': Types will only be documented for a parameter or a return value that is already documented by the docstring.
- 'documented_params': Parameter types will only be annotated if the parameter is documented in the docstring. The return type is always annotated (except if it is None).

Added in version 4.0.

Added in version 5.0: New option 'documented_params' is added.

autodoc_type_aliases

```
Type
    dict[str, str]
Default
    {}
```

A dictionary for users defined type aliases³⁷³ that maps a type name to the full-qualified object name. It is used to keep type aliases not evaluated in the document.

The type aliases are only available if your program enables **Postponed Evaluation of Annotations** (**PEP 563**)³⁷⁴ 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:
...
```

Added in version 3.3.

autodoc_typehints_format

```
Type
str

Default
'short'
```

This value controls the format of typehints. The setting takes the following values:

- 'fully-qualified' Show the module name and its name of typehints
- 'short' Suppress the leading module names of the typehints (e.g. io.StringIO -> StringIO)

Added in version 4.4.

Changed in version 5.0: The default setting was changed to 'short'

autodoc_preserve_defaults

```
Type
bool
Default
```

False

If True, the default argument values of functions will be not evaluated on generating document. It preserves them as is in the source code.

Added in version 4.0: Added as an experimental feature. This will be integrated into autodoc core in the future.

autodoc_warningiserror

```
Type bool
```

 $^{^{373}\} https://mypy.readthedocs.io/en/latest/kinds_of_types.html\#type-aliases$

³⁷⁴ https://peps.python.org/pep-0563/

Default

True

This value controls the behavior of *sphinx-build --fail-on-warning* during importing modules. If False is given, autodoc forcedly suppresses the error if the imported module emits warnings.

Changed in version 8.1: This option now has no effect as --fail-on-warning no longer exits early.

autodoc_inherit_docstrings

```
Type
```

bool

Default

True

This value controls the docstrings inheritance. If set to True the docstring for classes or methods, if not explicitly set, is inherited from parents.

Added in version 1.7.

suppress_warnings

```
Type
```

Sequence[str]

Default

()

autodoc supports suppressing warning messages via *suppress_warnings*. It defines the following additional warnings types:

- autodoc
- autodoc.import_object

Docstring preprocessing

autodoc provides the following additional events:

```
autodoc-process-docstring(app, what, name, obj, options, lines)
```

Added 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 no-index 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)

Added in version 2.4.

Emitted before autodoc formats a signature for an object. The event handler can modify an object to change its signature.

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)

Added 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 no-index 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^{375}, post: int^{376} = 0, what: str^{377} \mid list^{378}[str^{379}] \mid None^{380} = None) \rightarrow AutodocProcessDocstringListener
```

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.

³⁷⁵ https://docs.python.org/3/library/functions.html#int

https://docs.python.org/3/library/functions.html#int

³⁷⁷ https://docs.python.org/3/library/stdtypes.html#str

³⁷⁸ https://docs.python.org/3/library/stdtypes.html#list

³⁷⁹ https://docs.python.org/3/library/stdtypes.html#str

³⁸⁰ https://docs.python.org/3/library/constants.html#None

```
sphinx.ext.autodoc.between(marker: str^{381}, what: Sequence[str^{382}] \mid None^{383} = None, keepempty: bool^{384} = False, exclude: bool^{385} = False) \rightarrow AutodocProcessDocstringListener
```

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.

autodoc-process-bases(app, name, obj, options, bases)

Emitted when autodoc has read and processed a class to determine the base-classes. *bases* is a list of classes that the event handler can modify **in place** to change what Sphinx puts into the output. It's emitted only if show-inheritance option given.

Parameters

- app the Sphinx application object
- name the fully qualified name of the object
- **obj** the object itself
- options the options given to the class directive
- bases the list of base classes signature. see above.

Added in version 4.1.

Changed in version 4.3: bases can contain a string as a base class name. It will be processed as reStructuredText.

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)
```

Added 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

³⁸¹ https://docs.python.org/3/library/stdtypes.html#str

³⁸² https://docs.python.org/3/library/stdtypes.html#str

³⁸³ https://docs.python.org/3/library/constants.html#None

³⁸⁴ https://docs.python.org/3/library/functions.html#bool

³⁸⁵ https://docs.python.org/3/library/functions.html#bool

• **options** – the options given to the directive: an object with attributes inherited_members, undoc_members, show_inheritance and no-index that are true if the flag option of same name was given to the auto directive

sphinx.ext.autosectionlabel - Allow referencing sections by their title

Added in version 1.4.

By default, cross-references to sections use labels (see *ref*). This extension allows you to instead refer to sections by their title.

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

```
Type
bool

Default
False
```

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

```
Type
int | None

Default
None
```

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 (i.e. all sections are labeled).

Debugging

The WARNING: undefined label indicates that your reference in ref is mis-spelled. Invoking **sphinx-build** with -vvv (see -v³⁸⁶) will print all section names and the labels that have been generated for them. This output can help finding the right reference label.

³⁸⁶ https://docs.python.org/3/using/cmdline.html#cmdoption-v

sphinx.ext.autosummary - Generate autodoc summaries

Added 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. A *autosummary* directive also generates short "stub" files for the entries listed in its content. These files by default contain only the corresponding *sphinx.ext.autodoc* directive, but can be customized with templates.

The **sphinx-autogen** script is also able to generate "stub" files from command line.

.. 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:

<pre>environment.BuildEnvironment(app)</pre>	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.

Added 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.

Added 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
```

Added 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:

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.

Added 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 enabled 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.

Changed in version 4.0: Enabled by default.

autosummary_generate_overwrite

If true, autosummary overwrites existing files by generated stub pages. Defaults to true (enabled).

Added 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.

Added in version 2.0.

autosummary_imported_members

A boolean flag indicating whether to document classes and functions imported in modules. Default is False

Added in version 2.1.

Changed in version 4.4: If autosummary_ignore_module_all is False, this configuration value is ignored for members listed in __all__.

autosummary_ignore_module_all

If False and a module has the __all__ attribute set, autosummary documents every member listed in __all__ and no others. Default is True

Note that if an imported member is listed in __all__, it will be documented regardless of the value of autosummary_imported_members. To match the behaviour of from module import *, set autosummary_ignore_module_all to False and autosummary_imported_members to True.

Added in version 4.4.

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.

Added in version 3.2.

Customizing templates

Added 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.)

1 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 are 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.

objtype

```
Type of the documented object, one of "module", "function", "class", "method", "attribute", "data",
"object", "exception", "newvarattribute", "newtypedata", "property".
```

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.

Added in version 1.8.0.

functions

List containing names of "public" functions in the module. Here, "public" 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.

Added 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 filter³⁸⁷ 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.

1 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.

Note

The *sphinx-apidoc* command can be used to automatically generate API documentation for all code in a project, avoiding the need to manually author these documents and keep them up-to-date.

Marning

coverage **imports** the modules to be documented. If any modules have side effects on import, these will be executed by the coverage builder 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.

1 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.path³⁸⁸ – adapt your sys.path³⁸⁹ in the configuration file accordingly.

³⁸⁷ https://jinja.palletsprojects.com/en/3.0.x/templates/#jinja-filters.escape

To use this builder, activate the coverage extension in your configuration file and run sphinx-build -M coverage on the command line.

Builder

class sphinx.ext.coverage.CoverageBuilder

Configuration

Several configuration values can be used to specify what the builder should check:

coverage_modules

```
Type
list[str]

Default
```

List of Python packages or modules to test coverage for. When this is provided, Sphinx will introspect each package or module provided in this list as well as all sub-packages and sub-modules found in each. When this is not provided, Sphinx will only provide coverage for Python packages and modules that it is aware of: that is, any modules documented using the <code>py:module</code> directive provided in the <code>Python domain</code> or the <code>automodule</code> directive provided by the <code>autodoc</code> extension.

Added in version 7.4.

```
coverage_ignore_modules

coverage_ignore_functions

coverage_ignore_classes

coverage_ignore_pyobjects
```

List of Python regular expressions³⁹⁰.

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.

Added in version 2.1.

```
coverage_c_path
coverage_c_regexes
coverage_ignore_c_items
coverage_write_headline
    Set to False to not write headlines.
    Added in version 1.1.
```

coverage_skip_undoc_in_source

Skip objects that are not documented in the source with a docstring. False by default.

Added in version 1.1.

³⁸⁸ https://docs.python.org/3/library/sys.html#sys.path

 $^{^{389}\} https://docs.python.org/3/library/sys.html#sys.path$

³⁹⁰ https://docs.python.org/library/re

coverage_show_missing_items

Print objects that are missing to standard output also. False by default.

Added in version 3.1.

coverage_statistics_to_report

Print a tabular report of the coverage statistics to the coverage report. True by default.

Example output:

++ Module
+=====+===++====++=====+ package.foo_module
package.bar_module

Added in version 7.2.

coverage_statistics_to_stdout

Print a tabular report of the coverage statistics to standard output. False by default.

Example output:

Added in version 7.2.

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.

Added in version 1.1.

.. doctest:: [group]

A doctest-style code block. You can use standard doctest³⁹¹ 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 3.12:

```
.. doctest::
   :pyversion: > 3.12
```

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 Specifiers³⁹².

Added 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.

1 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:

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.

.. 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:

³⁹¹ https://docs.python.org/3/library/doctest.html#module-doctest

³⁹² https://peps.python.org/pep-0440/#version-specifiers

(continued from previous page)

```
.. 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::
        :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.

Added in version 1.5.

doctest_show_successes

Defaults to True. Controls whether successes are reported.

For a project with many doctests, it may be useful to set this to False to only highlight failures.

Added in version 7.2.

doctest_path

A list of directories that will be added to $sys.path^{393}$ 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.

Added 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.

Added in version 1.1.

doctest_test_doctest_blocks

If this is a nonempty string (the default is 'default'), standard reStructuredText doctest blocks will be tested too. They will be assigned to the group name given.

reStructuredText 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.)

If this value is left at its default value, the above snippet is interpreted by the doctest builder exactly like the following:

 $^{^{393}\} https://docs.python.org/3/library/sys.html#sys.path$

```
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 reStructuredText 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

Added 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

Added 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 *caption*. For example, to create an alias for the above mentioned issues, you would add

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 *caption*:

- If *caption* is None, the link caption is the full URL.
- If *caption* is a string, then it must contain %s exactly once. In this case the link caption is *caption* with the partial URL substituted for %s in the above example, the link caption would be issue 123.

To produce a literal % in either base URL or caption, use %%:

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 *caption* is not relevant.

Changed in version 4.0: Support to substitute by '%s' in the caption.

1 Note

Since links are generated from the role in the reading stage, they appear as ordinary links to e.g. the linkcheck builder.

extlinks_detect_hardcoded_links

If enabled, extlinks emits a warning if a hardcoded link is replaceable by an extlink, and suggests a replacement via warning. It defaults to False.

Added in version 4.5.

sphinx.ext.githubpages - Publish HTML docs in GitHub Pages

Added 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

Added in version 0.6.

This extension allows you to embed Graphviz³⁹⁴ 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.

³⁹⁴ 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.

Added in version 1.0.

:align: alignment of the graph (left, center or right)

The horizontal alignment of the graph.

Added in version 1.5.

:caption: caption of the graph (text)

The caption of the graph.

Added in version 1.1.

:layout: layout type of the graph (text)

The layout of the graph (e.g. dot, neato and so on). A path to the graphviz commands are also allowed. By default, graphviz_dot is used.

Added in version 1.4.

Changed in version 2.2: Renamed from graphviz_dot

:name: label (text)

The label of the graph.

Added in version 1.6.

:class: class names (a list of class names separated by spaces)

The class name of the graph.

Added 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";
```

1 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)

Added in version 1.0.

```
:align: alignment of the graph (left, center or right)
         Added in version 1.5.
     :caption: caption of the graph (text)
         Added in version 1.1.
     :layout: layout type of the graph (text)
         Added in version 1.4.
         Changed in version 2.2: Renamed from graphviz_dot
     :name: label (text)
         Added in version 1.6.
     :class: class names (a list of class names separated by spaces)
         The class name of the graph.
         Added 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)
         Added in version 1.0.
     :align: alignment of the graph (left, center or right)
         Added in version 1.5.
     :caption: caption of the graph (text)
         Added in version 1.1.
     :layout: layout type of the graph (text)
         Added in version 1.4.
         Changed in version 2.2: Renamed from graphviz_dot
     :name: label (text)
         Added in version 1.6.
     :class: class names (a list of class names separated by spaces)
         The class name of the graph.
         Added in version 2.4.
```

4.7. Extensions 229

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 -M html -D graphviz_dot=C:\graphviz\bin\dot.exe . _build
```

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:

```
digraph example {
    a [label="sphinx", href="https://www.sphinx-doc.org/", target="_top"];
    b [label="other"];
    a -> b;
}
```

Added 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:

A 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

Added 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.

By default the extension uses ImageMagick³⁹⁵ to perform conversions, and will not work if ImageMagick is not installed.



1 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-svg2pdfconverter³⁹⁶ (which uses Inkscape or rsvq-convert).

Configuration

image_converter

A path to a conversion command. By default, the imgconverter finds the command from search paths.

On Unix platforms, the command **convert** is used by default.

On Windows, the command **magick** 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, it defaults to ["convert"].

Changed in version 3.1: Use ["convert"] by default on Windows

sphinx.ext.inheritance_diagram - Include inheritance diagrams

Added 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.

³⁹⁵ https://www.imagemagick.org

³⁹⁶ https://github.com/missinglinkelectronics/sphinxcontrib-svg2pdfconverter

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:

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:

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
```

class sphinx.ext.inheritance_diagram.InheritanceDiagram

The internal class that implements the inheritance-diagram directive.

Configuration

inheritance_graph_attrs

A dictionary of graphviz graph attributes for inheritance diagrams.

For example:

inheritance_node_attrs

A dictionary of graphviz node attributes for inheritance diagrams.

For example:

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

Added in version 0.5.

This extension can generate links to the documentation of objects in external projects, either explicitly through the *external* role, or as a fallback resolution for any other cross-reference.

Usage for fallback resolution is simple: whenever Sphinx encounters a cross-reference that has no matching target in the current documentation set, it looks for targets in the external documentation sets configured in <code>intersphinx_mapping</code>. A reference like <code>:py:class:`zipfile.ZipFile`</code> can then link to the Python documentation for the ZipFile class, without you having to specify where it is located exactly.

When using the *external* role, you can force lookup to any external projects, and optionally to a specific external project. A link like :external:ref:`comparison manual <comparisons>` will then link to the label "comparisons" in whichever configured external project, if it exists, and a link like :external+python:ref:`comparison manual <comparisons>` will link to the label "comparisons" only 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 both *external* references, and also 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.

Format

Added 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 in the *external* role, so that it is clear which intersphinx set the target belongs to. A link like :external+python:ref:`comparison manual <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:

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

Added 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:

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_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.



1 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.

intersphinx_disabled_reftypes

Added in version 4.3.

Changed in version 5.0: Changed default value from an empty list to ['std:doc'].

A list of strings being either:

- the name of a specific reference type in a domain, e.g., std:doc, py:func, or cpp:class,
- the name of a domain, and a wildcard, e.g., std:*, py:*, or cpp:*, or
- simply a wildcard *.

The default value is ['std:doc'].

When a non-external cross-reference is being resolved by intersphinx, skip resolution if it matches one of the specifications in this list.

intersphinx_disabled_reftypes = ['std:doc'] For example, with cross-reference :doc:`installation` not be attempted to be resolved intersphinx, :external+otherbook:doc: installation will be attempted to be resolved in the inventory named otherbook in intersphinx_mapping. At the same time, all cross-references generated in, e.g., Python, declarations will still be attempted to be resolved by intersphinx.

If * is in the list of domains, then no non-external references will be resolved by intersphinx.

Explicitly Reference External Objects

The Intersphinx extension provides the following role.

:external:

Added in version 4.4.

Use Intersphinx to perform lookup only in external projects, and not the current project. Intersphinx still needs to know the type of object you would like to find, so the general form of this role is to write the cross-reference as if the object is in the current project, but then prefix it with :external. The two forms are then

- :external:domain:reftype:`target`,e.g.,:external:py:class:`zipfile.ZipFile`,or
- :external:reftype:`target`,e.g.,:external:doc:`installation`. With this shorthand, the domain is assumed to be std.

If you would like to constrain the lookup to a specific external project, then the key of the project, as specified in intersphinx_mapping, is added as well to get the two forms

- :external+invname:domain:reftype:`target`,e.g.,:external+python:py:class:`zipfile. ZipFile`, or
- :external+invname:reftype:`target`,e.g.,:external+python:doc:`installation`.

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 -m sphinx.ext.intersphinx https://docs.python.org/3/objects.inv
```

Using Intersphinx with inventory file under Basic Authorization

Intersphinx supports Basic Authorization like this:

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

Added 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:

```
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

Added 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

Added in version 1.4.

This extension renders math via LaTeX and dvipng³⁹⁷ or dvisvgm³⁹⁸ 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 dvipng³⁹⁹ or dvisvgm⁴⁰⁰.

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 package⁴⁰¹ (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.

imamath 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:

```
sphinx-build -M html -D imgmath_latex=C:\tex\latex.exe . _build
```

This value should only contain the path to the latex executable, not further arguments; use *imgmath_latex_args* for that purpose.

³⁹⁷ https://savannah.nongnu.org/projects/dvipng/

³⁹⁸ https://dvisvgm.de/

³⁹⁹ https://savannah.nongnu.org/projects/dvipng/

⁴⁰⁰ https://dvisvgm.de/

⁴⁰¹ https://www.gnu.org/software/auctex/preview-latex.html

Hint

To use OpenType Math fonts⁴⁰² with unicode-math, via a custom <code>imgmath_latex_preamble</code>, you can set <code>imgmath_latex</code> to 'dvilualatex', but must then set <code>imgmath_image_format</code> to 'svg'. Note: this has only been tested with <code>dvisvgm 3.0.3</code>. It significantly increases image production duration compared to using standard 'latex' with traditional TeX math fonts.

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.

Changed in version 6.2.0: Using 'xelatex' (or a full path to it) is now supported. But you must then add '-no-pdf' to the <code>imgmath_latex_args</code> list of the command options. The 'svg' <code>imgmath_image_format</code> is required. Also, you may need the <code>dvisvgm</code> binary to be relatively recent (testing was done only with its 3.0.3 release).

1 Note

Regarding the previous note, it is currently not supported to use latexmk with option -xelatex.

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'.

 $^{^{402}\} https://tex.stackexchange.com/questions/425098/which-opentype-math-fonts-are-available$

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 FAQ⁴⁰³). This option is used only when imgmath_image_format is 'svg'.

imamath embed

Default: False. If true, encode LaTeX output images within HTML files (base64 encoded) and do not save separate png/svg files to disk.

Added in version 5.2.

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 (see mathjax3_config).

Added in version 1.1.

This extension puts math as-is into the HTML files. The JavaScript package MathJax⁴⁰⁴ 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.



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 jsdelivr⁴⁰⁵ Content Delivery Network. See the MathJax Getting Started page 406 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:

⁴⁰³ https://dvisvgm.de/FAQ

⁴⁰⁴ https://www.mathjax.org/

⁴⁰⁵ https://www.jsdelivr.com/

⁴⁰⁶ https://www.mathjax.org/#gettingstarted

```
mathjax_options = {
    'integrity': 'sha384-....',
}
```

The default is empty $(\{\})$.

Added in version 1.8.

Changed in version 4.4.1: Allow to change the loading method (async or defer) of MathJax if "async" or "defer" key is set.

mathjax3_config

The configuration options for MathJax v3 (which is used by default). The given dictionary is assigned to the JavaScript variable window.MathJax. For more information, please read Configuring MathJax⁴⁰⁷.

The default is empty (not configured).

Added in version 4.0.

mathjax2_config

The configuration options for MathJax v2 (which can be loaded via mathjax_path). The value is used as a parameter of MathJax.Hub.Config(). For more information, please read Using in-line configuration options⁴⁰⁸.

For example:

```
mathjax2_config = {
    'extensions': ['tex2jax.js'],
    'jax': ['input/TeX', 'output/HTML-CSS'],
}
```

The default is empty (not configured).

Added in version 4.0: mathjax_config has been renamed to mathjax2_config.

mathjax_config

Former name of mathjax2_config.

For help converting your old MathJax configuration to to the new *mathjax3_config*, see Converting Your v2 Configuration to v3⁴⁰⁹.

Added in version 1.8.

Changed in version 4.0: This has been renamed to mathjax2_config. mathjax_config is still supported for backwards compatibility.

sphinx.ext.jsmath - Render math via JavaScript

This extension works just as the MathJax extension does, but uses the older package jsMath⁴¹⁰. 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.

4.7. Extensions 241

⁴⁰⁷ https://docs.mathjax.org/en/latest/web/configuration.html#configuration

 $^{^{408}\} https://docs.mathjax.org/en/v2.7-latest/configuration.html \#using-in-line-configuration-options$

⁴⁰⁹ https://docs.mathjax.org/en/latest/web/configuration.html#converting-your-v2-configuration-to-v3

⁴¹⁰ https://www.math.union.edu/~dpvc/jsmath/

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.

sphinx.ext.napoleon - Support for NumPy and Google style docstrings

Module author: Rob Ruana
Added 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
```

reStructuredText⁴¹¹ is great, but it creates visually dense, hard to read **docstrings**⁴¹². Compare the jumble above to the same thing rewritten according to the Google Python Style Guide⁴¹³:

```
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 NumPy⁴¹⁴ and Google⁴¹⁵ style docstrings - the style recommended by Khan Academy⁴¹⁶.

Napoleon is a pre-processor that parses NumPy⁴¹⁷ and Google⁴¹⁸ 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.

```
411 https://docutils.sourceforge.io/rst.html
```

⁴¹² https://peps.python.org/pep-0287/

⁴¹³ https://google.github.io/styleguide/pyguide.html

 $^{^{414}\} https://numpydoc.readthedocs.io/en/latest/format.html\#docstring-standard$

⁴¹⁵ https://google.github.io/styleguide/pyguide.html#38-comments-and-docstrings

⁴¹⁶ https://github.com/Khan/style-guides/blob/master/style/python.md#docstrings

⁴¹⁷ https://numpydoc.readthedocs.io/en/latest/format.html#docstring-standard

⁴¹⁸ https://google.github.io/styleguide/pyguide.html#38-comments-and-docstrings

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

4.7. Extensions 243

- 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: Google⁴¹⁹ and NumPy⁴²⁰. 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)

 $^{^{419}\} https://google.github.io/styleguide/pyguide.html \#38-comments-and-docstrings$

⁴²⁰ 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 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 484⁴²¹ 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**⁴²² is that type checkers and IDEs can take advantage of them for static code analysis. **PEP 484**⁴²³ was then extended by **PEP 526**⁴²⁴ 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)
```

421 https://peps.python.org/pep-0484/

4.7. Extensions 245

⁴²² https://peps.python.org/pep-0484/

⁴²³ https://peps.python.org/pep-0484/

⁴²⁴ https://peps.python.org/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
    """"
```

1 Note

Python 2/3 compatible annotations⁴²⁵ aren't currently supported by Sphinx and won't show up in the docs.

 $^{^{425}\} https://peps.python.org/pep-0484/\# suggested-syntax-for-python-2-7-and-straddling-code$

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_preprocess_types = False
napoleon_type_aliases = None
napoleon_attr_annotations = True
```

napoleon_google_docstring

True to parse Google style 426 docstrings. False to disable support for Google style docstrings. Defaults to True.

napoleon_numpy_docstring

True to parse NumPy style 427 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:

4.7. Extensions 247

 $^{^{426}\} https://google.github.io/styleguide/pyguide.html \#38-comments-and-docstrings$

⁴²⁷ https://numpydoc.readthedocs.io/en/latest/format.html#docstring-standard

```
def _included(self):
    """
    This will be included in the docs because it has a docstring
    """
    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 style⁴²⁸ 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*.

⁴²⁸ https://numpydoc.readthedocs.io/en/latest/format.html#docstring-standard



The singular **Note** section will always be converted to a .. note:: directive.

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*.

```
Process

Read a section of the section of the
```

napoleon_use_ivar

True to use the :ivar: role for instance variables. False to use the .. attribute:: directive instead. *Defaults to False*.

This NumPy style⁴²⁹ 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 style⁴³⁰ snippet will be converted as follows:

```
Parameters
-----
arg1: str

(continues on next page)
```

429 https://numpydoc.readthedocs.io/en/latest/format.html#docstring-standard

4.7. Extensions 249

(continued from previous page)

```
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:

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 <code>napoleon_use_param</code>. 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 style⁴³¹ 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
```

⁴³⁰ https://numpydoc.readthedocs.io/en/latest/format.html#docstring-standard

⁴³¹ https://numpydoc.readthedocs.io/en/latest/format.html#docstring-standard

napoleon_preprocess_types

True to convert the type definitions in the docstrings as references. Defaults to False.

Added in version 3.2.1.

Changed in version 3.5: Do preprocess the Google style docstrings also.

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 style⁴³² snippet:

```
Parameters
------
arg1 : CustomType
   Description of `arg1`
arg2 : dict-like
   Description of `arg2`
```

becomes:

```
:param arg1: Description of `arg1`
:type arg1: mypackage.CustomType
:param arg2: Description of `arg2`
:type arg2: :term:`dict-like <mapping>`
```

Added in version 3.2.

napoleon_attr_annotations

True to allow using PEP 526^{433} 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.

Added 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

4.7. Extensions 251

⁴³² https://numpydoc.readthedocs.io/en/latest/format.html#docstring-standard

⁴³³ https://peps.python.org/pep-0526/

value is "params_style" or "returns_style", the custom section will be displayed like the parameters section or returns section.

Added 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

Added 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.

Added 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.

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.

Added in version 1.5.

todo_link_only

If this is True, todolist produce output without file path and line, The default is False.

Added in version 1.4.

autodoc provides the following an additional event:

todo-defined(app, node)

Added 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

Added 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.

Added 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.

Added 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 epubcheck⁴³⁴'s score becomes worse even if the reader supports.

viewcode_line_numbers

Default: False.

If set to True, inline line numbers will be added to the highlighted code.

Added in version 7.2.

viewcode-find-source(app, modname)

Added in version 1.8.

4.7. Extensions 253

⁴³⁴ https://github.com/IDPF/epubcheck

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)

Added 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-contrib⁴³⁵ organization. If you wish to include your extension in this organization, simply follow the instructions provided in the github-administration⁴³⁶ project. This is optional and there are several extensions hosted elsewhere. The awesome-sphinxdoc⁴³⁷ and sphinx-extensions⁴³⁸ projects are both curated lists of Sphinx packages, and many packages use the Framework:: Sphinx:: Extension⁴³⁹ and Framework:: Sphinx:: Theme⁴⁴⁰ trove classifiers for Sphinx extensions and themes, respectively.

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.

4.8 HTML theming

Sphinx provides a number of builders for HTML and HTML-based formats.

```
435 https://github.com/sphinx-contrib/
```

⁴³⁶ https://github.com/sphinx-contrib/github-administration

⁴³⁷ https://github.com/yoloseem/awesome-sphinxdoc

⁴³⁸ https://sphinx-extensions.readthedocs.io/en/latest/

⁴³⁹ https://pypi.org/search/?c=Framework+%3A%3A+Sphinx+%3A%3A+Extension

⁴⁴⁰ https://pypi.org/search/?c=Framework+%3A%3A+Sphinx+%3A%3A+Theme

Builders

Themes

Added in version 0.6.



1 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.toml 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
```

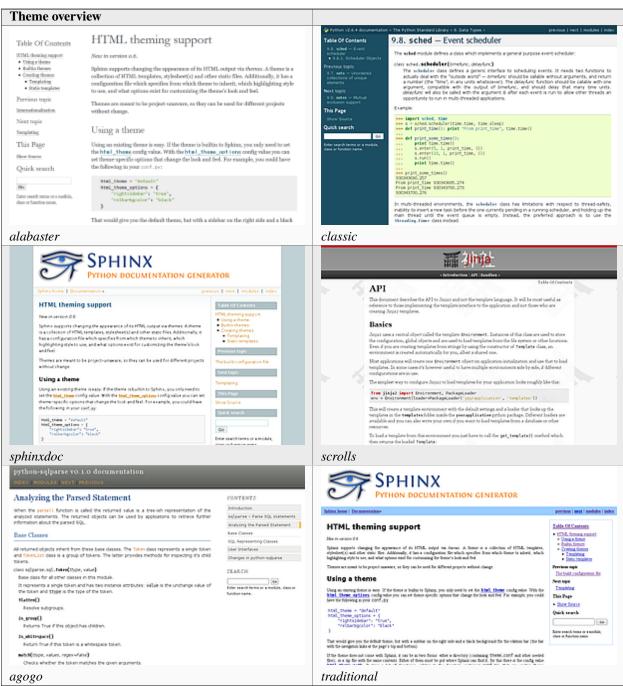
Once installed, this can be used in the same manner as a directory or zipfile-based theme:

```
html_theme = "dotted"
```

For more information on the design of themes, including information about writing your own themes, refer to HTML theme development.

4.8. HTML theming 255

Builtin themes



continues on next page



Table 1 - continued from previous page

Sphinx comes with a selection of themes to choose from.

Note that from these themes only the Alabaster and Scrolls themes are mobile-optimated, the other themes resort to horizontal scrolling if the screen is too narrow.

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 with the following keyboard shortcuts:

- Left arrow: previous page

4.8. HTML theming 257

- Right arrow: next page

Defaults to False.

• enable_search_shortcuts (true or false): Allow jumping to the search box with / and allow removal of search highlighting with Esc.

Defaults to True.

• **globaltoc_collapse** (true or false): Only expand subsections of the current document in **globaltoc.html** (see html_sidebars). Defaults to True.

Added 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.

Added 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.

Added in version 3.2.

alabaster

Alabaster theme⁴⁴¹ 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 page⁴⁴² for information on how to configure *html_sidebars* for its use.

classic

This is the classic theme, which looks like the Python 2 documentation⁴⁴³. 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 *collapsi-blesidebar* 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.

⁴⁴¹ https://pypi.org/project/alabaster/

⁴⁴² https://alabaster.readthedocs.io/en/latest/installation.html

⁴⁴³ https://docs.python.org/2/

- 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*.



The Sphinx documentation now uses an adjusted version of the sphinxdoc theme⁴⁴⁴.

scrolls

A more lightweight theme, based on the Jinja documentation 445. The following color options are available:

- headerbordercolor
- 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.

4.8. HTML theming 259

 $^{^{444}\} https://github.com/sphinx-doc/sphinx/tree/master/doc/_themes/sphinx 13$

⁴⁴⁵ https://jinja.palletsprojects.com/

- 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 guide⁴⁴⁶. 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.

Added 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.

Third Party Themes

There are many third-party themes created for Sphinx. Some of these are for general use, while others are specific to an individual project.

sphinx-themes.org⁴⁴⁷ is a gallery that showcases various themes for Sphinx, with demo documentation rendered under each theme. Themes can also be found on $PyPI^{448}$ (using the classifier Framework :: Sphinx :: Theme),

⁴⁴⁶ https://www.haiku-os.org/docs/userguide/en/contents.html

⁴⁴⁷ https://sphinx-themes.org/

⁴⁴⁸ https://pypi.org/search/?q=&o=&c=Framework+%3A%3A+Sphinx+%3A%3A+Theme

GitHub⁴⁴⁹ and GitLab⁴⁵⁰.

4.9 Internationalization

Added 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 internationalisation* for details on configuration.

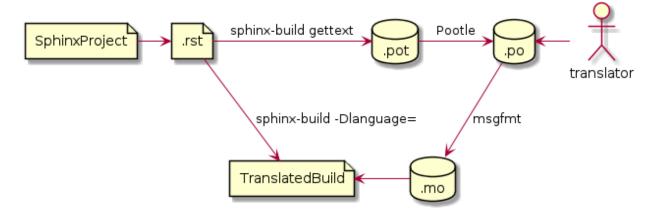


Fig. 1: Workflow visualization of translations in Sphinx. (The figure is created by plantuml⁴⁵¹.)

- Sphinx internationalization details
- Translating with sphinx-intl
 - Quick guide
 - Translating
 - Update your po files by new pot files
- Using Transifex service for team translation
- Using Weblate service for team translation
- Contributing to Sphinx reference translation
- Translation progress and statistics

Sphinx internationalization details

gettext¹ 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 -M gettext**.

 $^{^{449}}$ https://github.com/search?utf8=%E2%9C%93&q=sphinx+theme

⁴⁵⁰ https://gitlab.com/explore?name=sphinx+theme

⁴⁵¹ https://plantuml.com

¹ See the GNU gettext utilities⁴⁶⁴ for details on that software suite.

⁴⁶⁴ https://www.gnu.org/software/gettext/manual/gettext.html#Introduction

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 <code>MessageCatalogBuilder</code> you will find a collection of .pot files in your output directory. These are <code>catalog templates</code> and contain messages in your original language <code>only</code>.

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 translations² 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.

In order to protect against mistakes, a warning is emitted if cross-references in the translated paragraph do not match those from the original. This can be turned off globally using the *suppress_warnings* configuration variable. Alternatively, to turn it off for one message only, end the message with #noqa like this:

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse risus tortor, luctus id ultrices at. #noqa
```

(Write \#noqa in case you want to have "#noqa" literally in the text. This does not apply to code blocks, where #noqa is ignored because code blocks do not contain references anyway.)

Added in version 4.5: The #noqa mechanism.

Translating with sphinx-intl

Quick guide

sphinx-intl⁴⁵² is a useful tool to work with Sphinx translation flow. This section describe an easy way to translate with *sphinx-intl*.

1. Install sphinx-intl⁴⁵³.

```
$ pip install sphinx-intl
```

2. Add configurations to conf.py.

```
locale_dirs = ['locale/']  # path is example but recommended.
gettext_compact = False  # optional.
```

² Because nobody expects the Spanish Inquisition!

⁴⁵² https://pypi.org/project/sphinx-intl/

⁴⁵³ https://pypi.org/project/sphinx-intl/

This case-study assumes that BUILDDIR is set to _build, <code>locale_dirs</code> is set to <code>locale/</code> and <code>gettext_compact</code> is set to <code>False</code> (the Sphinx document is already configured as such).

3. Extract translatable messages into pot files.

```
$ make gettext
```

The generated pot files will be placed in the _build/gettext directory. If you want to customize the output beyond what can be done via the *Options for internationalisation*, the default pot file template can be replaced by a custom message.pot.jinja file placed in any directory listed in templates_path.

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/<lamp>/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 reStructuredText 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 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:

```
PS> Set-Item env:SPHINXOPTS "-D language=de"
PS> .\make.bat html
```

Congratulations! You got the translated documentation in the _build/html directory.

Added 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/gettext
```

Using Transifex service for team translation

Transifex⁴⁵⁴ is one of several services that allow collaborative translation via a web interface. It has a nifty Go-based command line client that makes it easy to fetch and push translations.

1. Install the Transifex CLI tool⁴⁵⁵.

You need the **tx** command line tool for uploading resources (pot files). The official installation process place the **tx** binary file in the current directory along with a README and a LICENSE file, and adds the current directory to \$PATH.

\$ curl -o- https://raw.githubusercontent.com/transifex/cli/master/install.sh | bash

```
★ See also
Transifex Client documentation<sup>456</sup>
```

2. Create your Transifex⁴⁵⁷ account and create a new project and an organization 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:

```
Organization ID
    sphinx-document

Project ID
    sphinx-document-test_1_0

Project URL
    https://www.transifex.com/projects/p/sphinx-document-test_1_0/
```

3. Create an API token to be used in the command-line.

Go to your Transifex API token⁴⁵⁸ page and generate a token. Copy the generated token now, as you will not be

⁴⁵⁴ https://app.transifex.com/

⁴⁵⁵ https://github.com/transifex/cli/

⁴⁵⁶ https://developers.transifex.com/docs/using-the-client

⁴⁵⁷ https://app.transifex.com/

⁴⁵⁸ https://app.transifex.com/user/settings/api/

able to see it again later.

4. Set your Transifex API token in the user configuration file \$HOME/.transifexrc.

```
[https://app.transifex.com]
rest_hostname = https://rest.api.transifex.com
token = paste_your_api_token_here
```

5. Alternatively, you can store your Transifex API token as the environment variable TX_TOKEN, which is recognized and used by tx.

```
$ export TX_TOKEN=paste_your_api_token_here
```

6. Create the project's config file for **tx** command.

This process will create .tx/config in the current directory.

```
$ cd /your/document/root
$ tx init
Successful creation of '.tx/config' file
```

7. Upload pot files to Transifex service.

Register pot files to .tx/config file using **sphinx-intl update-txconfig-resources**, adjusting --pot-dir value to your project's pot files' directory:

```
$ cd /your/document/root
$ sphinx-intl update-txconfig-resources --pot-dir _build/locale \
    --transifex-organization-name=sphinx-document \
    --transifex-project-name=sphinx-document-test_1_0
```

You can use the SPHINXINTL_TRANSIFEX_ORGANIZATION_NAME and SPHINXINTL_TRANSIFEX_PROJECT_NAME environment variables instead of the respective command line arguments.

```
See also
sphinx-intl update-txconfig-resources documentation<sup>459</sup>
```

and upload pot files:

```
$ tx push -s
# Getting info about resources

sphinx-document-test_1_0.builders - Getting info
sphinx-document-test_1_0.builders - Done

# Pushing source files

sphinx-document-test_1_0.builders - Uploading file
sphinx-document-test_1_0.builders - Done
```

8. Forward the translation on Transifex.

⁴⁵⁹ https://sphinx-intl.readthedocs.io/en/master/refs.html#sphinx-intl-update-txconfig-resources

9. 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
# Getting info about resources

sphinx-document-test_1_0.builders - Getting info
sphinx-document-test_1_0.builders - Done

# Pulling files

sphinx-document-test_1_0.builders [de] - Pulling file
sphinx-document-test_1_0.builders [de] - Creating download job
sphinx-document-test_1_0.builders [de] - Done
```

Invoke **make** html (for BSD/GNU make) passing the language code:

```
$ make -e SPHINXOPTS="-D language='de'" html
```

That's all!

Ţip

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.

Using Weblate service for team translation

Read more in Weblate's documentation⁴⁶⁰.

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 page⁴⁶¹ for Sphinx (master) documentation.

- 1. Login to Transifex⁴⁶² service.
- 2. Go to sphinx translation page⁴⁶³.
- 3. Click Request language and fill form.
- 4. Wait acceptance by Transifex sphinx translation maintainers.
- 5. (After acceptance) Translate on Transifex.

Detail is here: https://help.transifex.com/en/articles/6248698-getting-started-as-a-translator

⁴⁶⁰ https://docs.weblate.org/en/latest/devel/sphinx.html

⁴⁶¹ https://app.transifex.com/sphinx-doc/sphinx-doc/

⁴⁶² https://app.transifex.com/

⁴⁶³ https://app.transifex.com/sphinx-doc/sphinx-doc/

Translation progress and statistics

Added in version 7.1.0.

During the rendering process, Sphinx marks each translatable node with a translated attribute, indicating if a translation was found for the text in that node.

The translation_progress_classes configuration value can be used to add a class to each element, depending on the value of the translated attribute.

The |translation progress| substitution can be used to display the percentage of nodes that have been translated on a per-document basis.

4.10 Sphinx Web Support

Added 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 <code>WebSupport</code> class and call its <code>build()</code> method:

This will read reStructuredText sources from srcdir and place the necessary data in builddir. The builddir will contain two subdirectories: 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".



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 web-app

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

(continues on next page)
```

(continued from previous page)

You'll only need one of these for each set of documentation you will be working with. You can then call its <code>get_document()</code> 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 $Jinja2^{465}$ is:

```
{%- extends "layout.html" %}
{%- block title %}
   {{ document.title }}
{%- endblock %}
{% block css %}
   {{ super() }}
   {{ 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 %}
```

⁴⁶⁵ https://jinja.palletsprojects.com/

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 <code>WebSupport</code> methods using the <code>username</code> and <code>moderator</code> 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 Flask⁴⁶⁶ 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 docname to $get_document()$. The web support package will then add this data to the COMMENT_OPTIONS that are used in the template.

1 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 <code>get_search_results()</code> method to retrieve search results. In Flask⁴⁶⁷ that would be like this:

⁴⁶⁶ https://flask.palletsprojects.com/

⁴⁶⁷ https://flask.palletsprojects.com/

(continued from previous page)

```
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():

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
    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:

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 <code>WebSupport.add_comment()</code>.

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').



1 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.
username	The username that was stored with the comment.
id	The comment's unique identifier.
rating	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. <i>iso</i> is the time formatted in ISO 8601 format. <i>delta</i> is a printable form of how old the comment is (e.g. "3 hours ago").
vote	If <i>user_id</i> 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.
parent	The id of the comment that this comment is attached to if it is not attached to a node.
children	A list of all children, in this format.
proposal_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:

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:

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: Sequence<sup>468</sup>[str^{469}] = ()) \rightarrow None<sup>470</sup>
```

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

 $^{^{468}\} https://docs.python.org/3/library/collections.abc.html \# collections.abc. Sequence$

⁴⁶⁹ https://docs.python.org/3/library/stdtypes.html#str

⁴⁷⁰ https://docs.python.org/3/library/constants.html#None

- **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:

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

FIVE

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 *Sphinx API*.

5.1 Tutorials

Extending syntax with roles and directives

Overview

The syntax of both reStructuredText and MyST can be extended by creating new **directives** - for block-level elements - and **roles** - for inline elements.

In this tutorial we shall extend Sphinx to add:

- A hello role, that will simply output the text Hello {text}!.
- A hello directive, that will simply output the text Hello {text}!, as a paragraph.

For this extension, you will need some basic understanding of Python, and we shall also introduce aspects of the docutils⁴⁷¹ API.

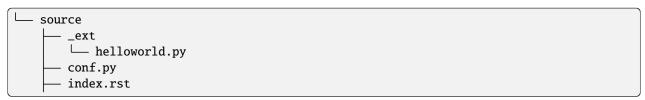
Setting up the project

You can either use an existing Sphinx project or create a new one using sphinx-quickstart.

With this we will add the extension to the project, within the source folder:

- 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:



⁴⁷¹ https://docutils.sourceforge.io/

Writing the extension

Open helloworld.py and paste the following code in it:

```
from __future__ import annotations
   from docutils import nodes
   from sphinx.application import Sphinx
   from sphinx.util.docutils import SphinxDirective, SphinxRole
   from sphinx.util.typing import ExtensionMetadata
   class HelloRole(SphinxRole):
10
       """A role to say hello!"""
11
       def run(self) -> tuple[list[nodes.Node], list[nodes.system_message]]:
13
           node = nodes.inline(text=f'Hello {self.text}!')
           return [node], []
15
16
   class HelloDirective(SphinxDirective):
18
       """A directive to say hello!"""
19
20
       required_arguments = 1
21
22
       def run(self) -> list[nodes.Node]:
23
           paragraph_node = nodes.paragraph(text=f'hello {self.arguments[0]}!')
24
           return [paragraph_node]
26
27
   def setup(app: Sphinx) -> ExtensionMetadata:
28
       app.add_role('hello', HelloRole())
29
       app.add_directive('hello', HelloDirective)
30
       return {
32
            'version': '0.1',
33
            'parallel_read_safe': True,
34
            'parallel_write_safe': True,
35
       }
```

Some essential things are happening in this example:

The role class

Our new role is declared in the HelloRole class.

```
class HelloRole(SphinxRole):
    """A role to say hello!"""

def run(self) -> tuple[list[nodes.Node], list[nodes.system_message]]:
    node = nodes.inline(text=f'Hello {self.text}!')
    return [node], []
```

This class extends the SphinxRole class. The class contains a run method, which is a requirement for every role. It

contains the main logic of the role and it returns a tuple containing:

- a list of inline-level docutils nodes to be processed by Sphinx.
- an (optional) list of system message nodes

The directive class

Our new directive is declared in the HelloDirective class.

```
class HelloDirective(SphinxDirective):
    """A directive to say hello!"""

required_arguments = 1

def run(self) -> list[nodes.Node]:
    paragraph_node = nodes.paragraph(text=f'hello {self.arguments[0]}!')
    return [paragraph_node]
```

This class extends the *SphinxDirective* class. The class contains a run method, which is a requirement for every directive. It contains the main logic of the directive and it returns a list of block-level docutils nodes to be processed by Sphinx. It also contains a required_arguments attribute, which tells Sphinx how many arguments are required for the directive.

What are docutils nodes?

When Sphinx parses a document, it creates an "Abstract Syntax Tree" (AST) of nodes that represent the content of the document in a structured way, that is generally independent of any one input (rST, MyST, etc) or output (HTML, LaTeX, etc) format. It is a tree because each node can have children nodes, and so on:

```
<document>
  <paragraph>
  <text>
        Hello world!
```

The docutils⁴⁷² package provides many built-in nodes⁴⁷³, to represent different types of content such as text, paragraphs, references, tables, etc.

Each node type generally only accepts a specific set of direct child nodes, for example the document node should only contain "block-level" nodes, such as paragraph, section, table, etc, whilst the paragraph node should only contain "inline-level" nodes, such as text, emphasis, strong, etc.

```
    ✓ See also
    The docutils documentation on creating directives<sup>474</sup>, and creating roles<sup>475</sup>.
```

⁴⁷² https://docutils.sourceforge.io/

⁴⁷³ https://docutils.sourceforge.io/docs/ref/doctree.html

⁴⁷⁴ https://docutils.sourceforge.io/docs/howto/rst-directives.html

⁴⁷⁵ https://docutils.sourceforge.io/docs/howto/rst-roles.html

The setup function

This function is a requirement. We use it to plug our new directive into Sphinx.

```
def setup(app: Sphinx) -> ExtensionMetadata:
    app.add_role('hello', HelloRole())
    app.add_directive('hello', HelloDirective)

return {
        'version': '0.1',
        'parallel_read_safe': True,
        'parallel_write_safe': True,
    }
}
```

The simplest thing you can do is to call the <code>Sphinx.add_role()</code> and <code>Sphinx.add_directive()</code> methods, which is what we've done here. For this particular call, the first argument is the name of the role/directive itself as used in a reStructuredText file. In this case, we would use hello. For example:

```
Some intro text here...
.. hello:: world
Some text with a :hello:`world` role.
```

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 path⁴⁷⁶ 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']
```

Ţip

Because we haven't installed our extension as a Python package⁴⁷⁷, we need to modify the Python path⁴⁷⁸ 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:

 $^{^{476}\} https://docs.python.org/3/using/cmdline.html\#envvar-PYTHONPATH$

⁴⁷⁷ https://packaging.python.org/

⁴⁷⁸ https://docs.python.org/3/using/cmdline.html#envvar-PYTHONPATH

```
Some intro text here...
.. hello:: world
Some text with a :hello:`world` role.
```

The sample above would generate:

```
Some intro text here...
Hello world!
Some text with a hello world! role.
```

Further reading

This is the very basic principle of an extension that creates a new role and directive.

For a more advanced example, refer to Extending the build process.

If you wish to share your extension across multiple projects or with others, check out the *Third-party extensions* section.

Extending the build process

The objective of this tutorial is to create a more comprehensive extension than that created in Extending syntax with roles and directives. Whereas that guide just covered writing a custom role and directive, this guide covers a more complex extension to the Sphinx build process; adding 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 to the *sphinx.ext.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 *Extending syntax with roles and directives*, 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:

Writing the extension

Open todo.py and paste the following code in it, all of which we will explain in detail shortly:

```
from docutils import nodes
   from docutils.parsers.rst import Directive
   from sphinx.application import Sphinx
   from sphinx.locale import _
   from sphinx.util.docutils import SphinxDirective
   from sphinx.util.typing import ExtensionMetadata
   class todo(nodes.Admonition, nodes.Element):
10
       pass
11
12
   class todolist(nodes.General, nodes.Element):
14
       pass
16
17
   def visit_todo_node(self, node):
18
       self.visit_admonition(node)
19
20
21
   def depart_todo_node(self, node):
22
       self.depart_admonition(node)
23
24
25
   class TodolistDirective(Directive):
26
       def run(self):
27
           return [todolist('')]
```

```
29
   class TodoDirective(SphinxDirective):
31
       # this enables content in the directive
32
       has content = True
33
       def run(self):
35
            targetid = 'todo-%d' % self.env.new_serialno('todo')
36
            targetnode = nodes.target('', '', ids=[targetid])
38
            todo_node = todo('\n'.join(self.content))
39
            todo_node += nodes.title(_('Todo'), _('Todo'))
40
            todo_node += self.parse_content_to_nodes()
42
           if not hasattr(self.env, 'todo_all_todos'):
43
                self.env.todo_all_todos = []
44
            self.env.todo_all_todos.append({
46
                'docname': self.env.docname,
                'lineno': self.lineno,
48
                'todo': todo_node.deepcopy(),
                'target': targetnode,
50
           })
51
52
           return [targetnode, todo_node]
53
54
55
   def purge_todos(app, env, docname):
56
       if not hasattr(env, 'todo_all_todos'):
57
           return
58
       env.todo_all_todos = [
            todo for todo in env.todo_all_todos if todo['docname'] != docname
61
       ]
63
   def merge_todos(app, env, docnames, other):
65
       if not hasattr(env, 'todo_all_todos'):
            env.todo_all_todos = []
67
       if hasattr(other, 'todo_all_todos'):
            env.todo_all_todos.extend(other.todo_all_todos)
69
71
   def process_todo_nodes(app, doctree, fromdocname):
72
       if not app.config.todo_include_todos:
73
            for node in doctree.findall(todo):
74
                node.parent.remove(node)
76
       # Replace all todolist nodes with a list of the collected todos.
       # Augment each todo with a backlink to the original location.
78
       env = app.builder.env
                                                                                   (continues on next page)
```

```
if not hasattr(env, 'todo_all_todos'):
81
            env.todo_all_todos = []
83
        for node in doctree.findall(todolist):
            if not app.config.todo_include_todos:
                node.replace_self([])
                continue
87
88
            content = []
            for todo_info in env.todo_all_todos:
91
                para = nodes.paragraph()
92
                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)
                # Create a reference
                newnode = nodes.reference('', '')
100
                innernode = nodes.emphasis(_('here'), _('here'))
10
                newnode['refdocname'] = todo_info['docname']
102
                newnode['refuri'] = app.builder.get_relative_uri(
103
                     fromdocname, todo_info['docname']
105
                newnode['refuri'] += '#' + todo_info['target']['refid']
106
                newnode.append(innernode)
107
                para += newnode
108
                para += nodes.Text('.)')
109
110
                # Insert into the todolist
111
                content.extend((
                     todo_info['todo'],
113
                     para,
                ))
115
            node.replace_self(content)
117
118
119
   def setup(app: Sphinx) -> ExtensionMetadata:
120
        app.add_config_value('todo_include_todos', False, 'html')
121
122
        app.add_node(todolist)
123
        app.add_node(
124
            todo.
125
            html=(visit_todo_node, depart_todo_node),
126
            latex=(visit_todo_node, depart_todo_node),
            text=(visit_todo_node, depart_todo_node),
128
        )
130
        app.add_directive('todo', TodoDirective)
13
        app.add_directive('todolist', TodolistDirective)
132
```

```
app.connect('doctree-resolved', process_todo_nodes)
133
        app.connect('env-purge-doc', purge_todos)
        app.connect('env-merge-info', merge_todos)
135
        return {
137
             'version': '0.1'.
138
             'env_version': 1,
139
             'parallel_read_safe': True,
140
             'parallel_write_safe': True,
141
        }
142
```

This is far more extensive extension than the one detailed in *Extending syntax with roles and directives*, 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)
```

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.

1 Note

Many extensions will not have to create their own node classes and work fine with the nodes already provided by docutils 479 and Sphinx.

A 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 PickleError⁴⁸⁰. So if something goes wrong, please make sure that you put inherited nodes into a separate Python module.

For more details, see:

⁴⁷⁹ https://docutils.sourceforge.io/docs/ref/doctree.html

- 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 documentation⁴⁸¹; 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):
```

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:

```
class TodoDirective(SphinxDirective):
       # this enables content in the directive
       has content = True
       def run(self):
6
           targetid = 'todo-%d' % self.env.new_serialno('todo')
           targetnode = nodes.target('', '', ids=[targetid])
           todo_node = todo('\n'.join(self.content))
10
           todo_node += nodes.title(_('Todo'), _('Todo'))
           todo_node += self.parse_content_to_nodes()
12
13
           if not hasattr(self.env, 'todo_all_todos'):
14
                self.env.todo_all_todos = []
15
           self.env.todo_all_todos.append({
17
                'docname': self.env.docname,
                'lineno': self.lineno,
19
                'todo': todo_node.deepcopy(),
                'target': targetnode,
21
           })
23
           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

 $^{^{480}\} https://docs.python.org/3/library/pickle.html\#pickle.PickleError$

⁴⁸¹ https://docutils.sourceforge.io/docs/ref/rst/directives.html

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:

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:

```
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 <code>env-purge-doc</code> 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:

```
def merge_todos(app, env, docnames, other):

(continues on next page)
```

```
if not hasattr(env, 'todo_all_todos'):
    env.todo_all_todos = []
if hasattr(other, 'todo_all_todos'):
```

The other handler belongs to the *doctree-resolved* event:

```
def process_todo_nodes(app, doctree, fromdocname):
2
       if not app.config.todo_include_todos:
            for node in doctree.findall(todo):
                node.parent.remove(node)
6
       # 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
10
       if not hasattr(env, 'todo_all_todos'):
11
            env.todo_all_todos = []
12
13
       for node in doctree.findall(todolist):
14
            if not app.config.todo_include_todos:
15
                node.replace_self([])
                continue
17
18
            content = []
19
            for todo_info in env.todo_all_todos:
21
                para = nodes.paragraph()
                filename = env.doc2path(todo_info['docname'], base=None)
23
                description = _(
                     '(The original entry is located in {\it \%s}, line {\it \%d} and can be found '
25
                ) % (filename, todo_info['lineno'])
                para += nodes.Text(description)
27
28
                # Create a reference
29
                newnode = nodes.reference('', '')
                innernode = nodes.emphasis(_('here'), _('here'))
31
                newnode['refdocname'] = todo_info['docname']
32
                newnode['refuri'] = app.builder.get_relative_uri(
33
                    fromdocname, todo_info['docname']
34
                newnode['refuri'] += '#' + todo_info['target']['refid']
36
                newnode.append(innernode)
37
                para += newnode
38
                para += nodes.Text('.)')
40
                # Insert into the todolist
41
                content.extend((
42.
                    todo_info['todo'],
```

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 <code>sphinx.builders.Builder.get_relative_uri()</code> 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:

```
node.replace_self(content)
2
   def setup(app: Sphinx) -> ExtensionMetadata:
       app.add_config_value('todo_include_todos', False, 'html')
6
       app.add_node(todolist)
       app.add_node(
           todo,
10
           html=(visit_todo_node, depart_todo_node),
11
           latex=(visit_todo_node, depart_todo_node),
12
           text=(visit_todo_node, depart_todo_node),
       )
14
       app.add_directive('todo', TodoDirective)
16
       app.add_directive('todolist', TodolistDirective)
       app.connect('doctree-resolved', process_todo_nodes)
18
       app.connect('env-purge-doc', purge_todos)
       app.connect('env-merge-info', merge_todos)
20
21
       return {
22
            'version': '0.1',
23
            'env_version': 1,
24
            'parallel_read_safe': True,
25
            'parallel_write_safe': True,
26
       }
```

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 path⁴⁸² 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
```

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
```

 $^{^{482}\} https://docs.python.org/3/using/cmdline.html#envvar-PYTHONPATH$

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 docutils⁴⁸³ documentation and *Sphinx API*.

If you wish to share your extension across multiple projects or with others, check out the *Third-party extensions* section.

Adding a reference domain

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.



1 Note

This tutorial is based on a guide first published on opensource.com⁴⁸⁴ and is provided here with the original author's permission.

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
```

⁴⁸³ https://docutils.sourceforge.io/docs/

⁴⁸⁴ https://opensource.com/article/18/11/building-custom-workflows-sphinx

Writing the extension

Open recipe.py and paste the following code in it, all of which we will explain in detail shortly:

```
from collections import defaultdict
   from docutils.parsers.rst import directives
   from sphinx import addnodes
   from sphinx.application import Sphinx
   from sphinx.directives import ObjectDescription
   from sphinx.domains import Domain, Index
   from sphinx.roles import XRefRole
   from sphinx.util.nodes import make_refnode
   from sphinx.util.typing import ExtensionMetadata
11
12
13
   class RecipeDirective(ObjectDescription):
       """A custom directive that describes a recipe."""
15
       has_content = True
17
       required_arguments = 1
       option_spec = {
19
            'contains': directives.unchanged_required,
21
22
       def handle_signature(self, sig, signode):
23
           signode += addnodes.desc_name(text=sig)
24
           return sig
25
26
       def add_target_and_index(self, name_cls, sig, signode):
27
           signode['ids'].append('recipe' + '-' + sig)
28
           if 'contains' in self.options:
29
                ingredients = [x.strip() for x in self.options.get('contains').split(',')]
30
31
                recipes = self.env.get_domain('recipe')
32
                recipes add_recipe(sig, ingredients)
34
   class IngredientIndex(Index):
36
       """A custom index that creates an ingredient matrix."""
37
38
       name = 'ingredient'
       localname = 'Ingredient Index'
40
       shortname = 'Ingredient'
41
42
       def generate(self, docnames=None):
43
           content = defaultdict(list)
44
45
           recipes = {
               name: (dispname, typ, docname, anchor)
47
                for name, dispname, typ, docname, anchor, _ in self.domain.get_objects()
49
           recipe_ingredients = self.domain.data['recipe_ingredients']
```

```
ingredient_recipes = defaultdict(list)
51
52
            # flip from recipe_ingredients to ingredient_recipes
53
            for recipe_name, ingredients in recipe_ingredients.items():
                for ingredient in ingredients:
55
                    ingredient_recipes[ingredient].append(recipe_name)
57
            # convert the mapping of ingredient to recipes to produce the expected
58
            # output, shown below, using the ingredient name as a key to group
60
            # name, subtype, docname, anchor, extra, qualifier, description
61
            for ingredient, recipe_names in ingredient_recipes.items():
62
                for recipe_name in recipe_names:
                    dispname, typ, docname, anchor = recipes[recipe_name]
64
                    content[ingredient].append((
                        dispname.
66
                        docname,
68
                        anchor,
                        docname,
70
                        typ,
72
                    ))
74
            # convert the dict to the sorted list of tuples expected
75
            content = sorted(content.items())
76
            return content, True
78
   class RecipeIndex(Index):
81
        """A custom index that creates an recipe matrix."""
83
       name = 'recipe'
        localname = 'Recipe Index'
85
        shortname = 'Recipe'
87
        def generate(self, docnames=None):
            content = defaultdict(list)
89
            # sort the list of recipes in alphabetical order
91
            recipes = self.domain.get_objects()
92
            recipes = sorted(recipes, key=lambda recipe: recipe[0])
93
            # generate the expected output, shown below, from the above using the
95
            # first letter of the recipe as a key to group thing
            # name, subtype, docname, anchor, extra, qualifier, description
            for _name, dispname, typ, docname, anchor, _priority in recipes:
                content[dispname[0].lower()].append((
100
                    dispname,
10
                    0,
102
```

(continues on next page)

```
docname.
103
                      anchor,
                      docname,
105
                      typ,
107
                 ))
108
109
             # convert the dict to the sorted list of tuples expected
110
             content = sorted(content.items())
111
112
             return content, True
113
114
115
    class RecipeDomain(Domain):
116
        name = 'recipe'
117
        label = 'Recipe Sample'
118
        roles = {
119
             'ref': XRefRole(),
120
121
        directives = {
122
             'recipe': RecipeDirective,
        }
124
        indices = {
125
             RecipeIndex,
126
             IngredientIndex,
127
128
        initial_data = {
129
             'recipes': [], # object list
130
             'recipe_ingredients': {}, # name -> object
131
132
        data_version = 0
133
        def get_full_qualified_name(self, node):
135
             return f'recipe.{node.arguments[0]}'
136
137
        def get_objects(self):
             yield from self.data['recipes']
139
140
        def resolve_xref(self, env, fromdocname, builder, typ, target, node, contnode):
141
             match = [
142
                 (docname, anchor)
143
                 for name, sig, typ, docname, anchor, prio in self.get_objects()
144
                 if sig == target
145
             ]
146
147
             if len(match) > 0:
148
                 todocname = match[0][0]
                 targ = match[0][1]
150
151
                 return make_refnode(builder, fromdocname, todocname, targ, contnode, targ)
152
             else:
153
                 print('Awww, found nothing')
154
```

```
return None
155
        def add_recipe(self, signature, ingredients):
157
             """Add a new recipe to the domain."""
             name = f'recipe.{signature}'
159
             anchor = f'recipe-{signature}'
160
161
             self.data['recipe_ingredients'][name] = ingredients
162
             # name, dispname, type, docname, anchor, priority
             self.data['recipes'].append((
164
165
                 name.
                 signature,
166
                 'Recipe',
                 self.env.docname,
168
                 anchor,
                 0.
170
             ))
171
172
173
    def setup(app: Sphinx) -> ExtensionMetadata:
174
        app.add_domain(RecipeDomain)
175
176
        return {
177
             'version': '0.1',
178
             'parallel_read_safe': True,
179
             'parallel_write_safe': True,
180
        }
181
```

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:

```
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):
           signode += addnodes.desc_name(text=sig)
11
           return sig
13
       def add_target_and_index(self, name_cls, sig, signode):
14
           signode['ids'].append('recipe' + '-' + sig)
15
           if 'contains' in self.options:
16
               ingredients = [x.strip() for x in self.options.get('contains').split(',')]
18
```

5.1. Tutorials 297

```
recipes = self.env.get_domain('recipe')
recipes.add_recipe(sig, ingredients)
```

Unlike Extending syntax with roles and directives and Extending the build process, 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_target_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

```
class IngredientIndex(Index):
       """A custom index that creates an ingredient matrix."""
2
       name = 'ingredient'
       localname = 'Ingredient Index'
       shortname = 'Ingredient'
       def generate(self, docnames=None):
           content = defaultdict(list)
           recipes = {
11
                name: (dispname, typ, docname, anchor)
12
                for name, dispname, typ, docname, anchor, _ in self.domain.get_objects()
13
           }
           recipe_ingredients = self.domain.data['recipe_ingredients']
           ingredient_recipes = defaultdict(list)
16
17
           # flip from recipe_ingredients to ingredient_recipes
18
           for recipe_name, ingredients in recipe_ingredients.items():
                for ingredient in ingredients:
20
                    ingredient_recipes[ingredient].append(recipe_name)
21
22
           # convert the mapping of ingredient to recipes to produce the expected
23
           # output, shown below, using the ingredient name as a key to group
24
           #
25
           # name, subtype, docname, anchor, extra, qualifier, description
26
           for ingredient, recipe_names in ingredient_recipes.items():
                for recipe_name in recipe_names:
28
                    dispname, typ, docname, anchor = recipes[recipe_name]
                    content[ingredient].append((
                        dispname,
31
                        0,
32
                        docname,
33
34
                        anchor,
                        docname,
35
```

```
typ,
typ,

typ,

return content, True

'',
typ,

typ,

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."""
2
       name = 'recipe'
       localname = 'Recipe Index'
       shortname = 'Recipe'
6
       def generate(self, docnames=None):
           content = defaultdict(list)
10
           # sort the list of recipes in alphabetical order
11
           recipes = self.domain.get_objects()
12
           recipes = sorted(recipes, key=lambda recipe: recipe[0])
14
            # generate the expected output, shown below, from the above using the
            # first letter of the recipe as a key to group thing
16
17
           # name, subtype, docname, anchor, extra, qualifier, description
18
           for _name, dispname, typ, docname, anchor, _priority in recipes:
                content[dispname[0].lower()].append((
20
                    dispname,
21
                    0,
22
                    docname,
23
                    anchor,
                    docname,
25
                    ٠٠,
                    typ,
27
                ))
29
            # convert the dict to the sorted list of tuples expected
           content = sorted(content.items())
31
32
           return content, True
33
```

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 <code>Index.generate()</code> 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 <code>domain API guide</code> for more information on this API.

These index pages can be referenced with the ref role by combining the domain name and the index name value. For

example, RecipeIndex can be referenced with :ref:`recipe-recipe` and IngredientIndex can be referenced with :ref:`recipe-ingredient`.

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.

```
class RecipeDomain(Domain):
       name = 'recipe'
       label = 'Recipe Sample'
       roles = {
            'ref': XRefRole(),
6
       directives = {
            'recipe': RecipeDirective,
       }
       indices = {
10
           RecipeIndex,
11
            IngredientIndex,
12
13
       initial_data = {
14
            'recipes': [], # object list
15
            'recipe_ingredients': {}, # name -> object
16
       }
       data_version = 0
18
       def get_full_qualified_name(self, node):
20
           return f'recipe. {node.arguments[0]}'
21
22
       def get_objects(self):
23
           yield from self.data['recipes']
24
25
       def resolve_xref(self, env, fromdocname, builder, typ, target, node, contnode):
26
           match = [
27
                (docname, anchor)
28
                for name, sig, typ, docname, anchor, prio in self.get_objects()
29
                if sig == target
30
           ]
31
            if len(match) > 0:
33
                todocname = match[0][0]
                targ = match[0][1]
35
                return make_refnode(builder, fromdocname, todocname, targ, contnode, targ)
37
            else:
                print('Awww, found nothing')
                return None
41
       def add_recipe(self, signature, ingredients):
42
            """Add a new recipe to the domain."""
43
           name = f'recipe.{signature}'
44
            anchor = f'recipe-{signature}'
45
```

```
self.data['recipe_ingredients'][name] = ingredients
47
            # name, dispname, type, docname, anchor, priority
            self.data['recipes'].append((
49
                name,
                signature,
51
                'Recipe',
52
                self.env.docname,
53
                anchor,
                0,
55
            ))
```

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 recipe_ingredients. Each contains 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: Sphinx) -> ExtensionMetadata:
    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

```
Below are a collection of my favourite recipes. I highly recommend the
:recipe:ref:`TomatoSoup` recipe in particular!
.. toctree::
   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 docutils⁴⁸⁵ documentation and *Sphinx API*.

If you wish to share your extension across multiple projects or with others, check out the *Third-party extensions* section.

Developing autodoc extensions

The objective of this tutorial is to create an extension that adds support for new type for autodoc. This autodoc extension will format the IntEnum class from Python standard library. (module enum)

Overview

We want the extension that will create auto-documentation for IntEnum. IntEnum is the integer enum class from standard library enum module.

Currently this class has no special auto documentation behavior.

We want to add following to autodoc:

- A new autointenum directive that will document the IntEnum class.
- The generated documentation will have all the enum possible values with names.
- The autointenum directive will have an option :hex: which will cause the integers be printed in hexadecimal form.

Prerequisites

We need the same setup as in *the previous extensions*. This time, we will be putting out extension in a file called autodoc_intenum.py. The my_enums.py will contain the sample enums we will document.

Here is an example of the folder structure you might obtain:

⁴⁸⁵ https://docutils.sourceforge.io/docs/

Writing the extension

Start with setup function for the extension.

```
def setup(app: Sphinx) -> ExtensionMetadata:
    app.setup_extension('sphinx.ext.autodoc') # Require autodoc extension
    app.add_autodocumenter(IntEnumDocumenter)
    return {
        'version': '1',
        'parallel_read_safe': True,
}
```

The <code>setup_extension()</code> method will pull the autodoc extension because our new extension depends on autodoc. <code>add_autodocumenter()</code> is the method that registers our new auto documenter class.

We want to import certain objects from the autodoc extension:

```
from __future__ import annotations

from enum import IntEnum
from typing import TYPE_CHECKING, Any

from sphinx.ext.autodoc import ClassDocumenter, bool_option
```

There are several different documenter classes such as MethodDocumenter or AttributeDocumenter available in the autodoc extension but our new class is the subclass of ClassDocumenter which a documenter class used by autodoc to document classes.

This is the definition of our new the auto-documenter class:

```
class IntEnumDocumenter(ClassDocumenter):
       objtype = 'intenum'
2
       directivetype = ClassDocumenter.objtype
       priority = 10 + ClassDocumenter.priority
       option_spec = dict(ClassDocumenter.option_spec)
       option_spec['hex'] = bool_option
       @classmethod
       def can_document_member(
           cls, member: Any, membername: str, isattr: bool, parent: Any
10
       ) -> bool:
           try:
12
               return issubclass(member, IntEnum)
           except TypeError:
14
               return False
```

(continues on next page)

```
def add_directive_header(self, sig: str) -> None:
17
           super().add_directive_header(sig)
           self.add_line('
                              :final:', self.get_sourcename())
19
       def add_content(
21
22
           self,
           more_content: StringList | None,
23
       ) -> None:
24
           super().add_content(more_content)
25
26
           source_name = self.get_sourcename()
27
           enum_object: IntEnum = self.object
28
           use_hex = self.options.hex
            self.add_line('', source_name)
30
31
           for the_member_name, enum_member in enum_object.__members__.items(): # type:_
32
   →ignore[attr-defined]
                the_member_value = enum_member.value
33
                if use_hex:
                    the_member_value = hex(the_member_value)
35
                self.add_line(f'**{the_member_name}**: {the_member_value}', source_name)
37
                self.add_line('', source_name)
```

Important attributes of the new class:

objtype

This attribute determines the auto directive name. In this case the auto directive will be autointenum.

directivetype

This attribute sets the generated directive name. In this example the generated directive will be ..:py:class::.

priority

the larger the number the higher is the priority. We want our documenter be higher priority than the parent.

option spec

option specifications. We copy the parent class options and add a new option hex.

Overridden members:

can_document_member

This member is important to override. It should return *True* when the passed object can be documented by this class.

add_directive_header

This method generates the directive header. We add **:final:** directive option. Remember to call **super** or no directive will be generated.

add content

This method generates the body of the class documentation. After calling the super method we generate lines for enum description.

Using the extension

You can now use the new autodoc directive to document any IntEnum.

For example, you have the following IntEnum:

Listing 6: my_enums.py

```
class Colors(IntEnum):
    """Colors enumerator"""
    NONE = 0
    RED = 1
    GREEN = 2
    BLUE = 3
```

This will be the documentation file with auto-documentation directive:

Listing 7: index.rst

```
.. autointenum:: my_enums.Colors
```

Further reading

If you wish to share your extension across multiple projects or with others, check out the *Third-party extensions* section.

5.2 How-tos

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.application.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 *sphinx.ext.autodoc* extension:

```
def setup(app):
    app.setup_extension('sphinx.ext.autodoc')
```



Since your extension will depend on another, make sure to include it as a part of your extension's installation requirements.

Configuring builders

Discover builders by entry point

Added in version 1.6.

builder extensions can be discovered by means of entry points⁴⁸⁶ so that they do not have to be listed in the *extensions* configuration value.

5.2. How-tos 305

⁴⁸⁶ https://setuptools.pypa.io/en/latest/userguide/entry_point.html

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 --builder 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 pyproject.toml

```
[project.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.

5.3 HTML theme development

Added in version 0.6.



1 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.



1 Note

See *Sphinx API* 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:

- Either a theme.toml file (preferred) or 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.

Theme configuration (theme.toml)

The theme.toml file is a TOML⁴⁸⁷ document, containing two tables: [theme] and [options].

The [theme] table defines the theme's settings:

• inherit (string): The name of the base theme from which to inherit settings, options, templates, and static files. All static files from theme 'ancestors' will be used. The theme will use all options defined in inherited themes. Finally, inherited themes will be used to locate missing templates (for example, if "basic" is used as the base theme, most templates will already be defined).

If set to "none", the theme will not inherit from any other theme. Inheritance is recursive, forming a chain of inherited themes (e.g. default -> classic -> basic -> none).

⁴⁸⁷ https://toml.io/en/

• **stylesheets** (list of strings): A list of CSS filenames which will be included in generated HTML header. Setting the *html_style* config value will override this setting.

Other mechanisms for including multiple stylesheets include @import in CSS or using a custom HTML template with appropriate link rel="stylesheet"> tags.

- **sidebars** (list of strings): A list of sidebar templates. This can be overridden by the user via the *html_sidebars* config value.
- pygments_style (table): A TOML table defining the names of Pygments styles to use for highlighting syntax. The table has two recognised keys: default and dark. The style defined in the dark key will be used when the CSS media query (prefers-color-scheme: dark) evaluates to true.

[theme.pygments_style.default] can be overridden by the user via the pygments_style config value.

The [options] table defines the options for the theme. It is structured such that each key-value pair corresponds to a variable name and the corresponding default value. These options can be overridden by the user in httml_theme_options and are accessible from all templates as theme_<name>.

Added in version 7.3: theme.toml support.

Exemplar theme.toml file:

```
[theme]
inherit = "basic"
stylesheets = [
    "main-CSS-stylesheet.css",
]
sidebars = [
    "localtoc.html",
    "relations.html",
    "sourcelink.html",
    "searchbox.html",
]
# Style names from https://pygments.org/styles/
pygments_style = { default = "style_name", dark = "dark_style" }

[options]
variable = "default value"
```

Theme configuration (theme.conf)

The theme.conf file is in INI format¹ (readable by the standard Python configparser⁴⁸⁸ 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
```

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.

⁴⁸⁸ https://docs.python.org/3/library/configparser.html#module-configparser

- 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 a list of CSS filenames separated commas which will be referenced in the HTML header. You can also use CSS' @import technique to include one from the other, 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().
- 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 httml_theme_options and are accessible from all templates as theme_ name>.

Added in version 1.7: sidebar settings

Changed in version 5.1: The stylesheet setting accepts multiple CSS filenames

Convert theme.conf to theme.toml

INI-style theme configuration files (theme.conf) can be converted to TOML via a helper programme distributed with Sphinx. This is intended for one-time use, and may be removed without notice in a future version of Sphinx.

```
$ python -m sphinx.theming conf_to_toml [THEME DIRECTORY PATH]
```

The required argument is a path to a directory containing a theme.conf file. The programme will write a theme.toml file in the same directory, and will not modify the original theme.conf file.

Added in version 7.3.

Distribute your theme as a Python package

As a way to distribute your theme, you can use a Python package. This makes it easier for users to set up your theme.

To distribute your theme as a Python package, please define an entry point called sphinx.html_themes in your pyproject.toml file, and write a setup() function to register your theme using the add_html_theme() API:

```
# pyproject.toml

[project.entry-points."sphinx.html_themes"]
name_of_theme = "your_theme_package"
```

```
# your_theme_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.

Added in version 1.2: 'sphinx_themes' entry_points feature.

Deprecated since version 1.6: sphinx_themes entry_points has been deprecated.

Added in version 1.6: sphinx.html_themes entry_points feature.

Styling with CSS

The stylesheets setting can be used to add custom CSS files to a theme.

Caution

The structure of the HTML elements and their classes are currently not a well-defined public API. Please infer them from inspecting the built HTML pages. While we cannot guarantee full stability, they tend to be fairly stable.

Styling search result entries by category

Added in version 8.0.

1 Note

The CSS classes named below are generated by Sphinx's standalone search code. If you are using a third-party search provider, such as ReadTheDocs⁴⁸⁹, to provide search results, then the theming options available may vary.

The search result items have classes indicating the context in which the search term was found. You can use the CSS selectors:

- ul.search li.kind-index: For results in an index, such as the glossary
- ul.search li.kind-object: For results in source code, like Python function definitions
- ul.search li.kind-title: For results found in section headings
- ul.search li.kind-text: For results found anywhere else in the documentation text

As a base for inheritance by other themes, the basic theme is intentionally minimal and does not define CSS rules using these. Derived themes are encouraged to use these selectors as they see fit. For example, the following stylesheet adds contextual icons to the search result list:

```
ul.search {
    padding-left: 30px;
}
ul.search li {
    padding: 5px 0 5px 10px;
    list-style-type: "\25A1"; /* Unicode: White Square */
}
ul.search li.kind-index {
    list-style-type: "\1F4D1"; /* Unicode: Bookmark Tabs */
}
ul.search li.kind-object {
    list-style-type: "\1F4E6"; /* Unicode: Package */
}
ul.search li.kind-title {
    list-style-type: "\1F4C4"; /* Unicode: Page Facing Up */
}
```

⁴⁸⁹ https://docs.readthedocs.io/

```
ul.search li.kind-text {
    list-style-type: "\1F4C4"; /* Unicode: Page Facing Up */
}
```

Templating

Templating

Sphinx uses the Jinja⁴⁹⁰ 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.

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.

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:

```
{% extends "!layout.html" %}
{% block rootrellink %}
     <a href="https://project.invalid/">Project Homepage</a> &raquo;
     {{ super() }}
{% endblock %}
```

⁴⁹⁰ https://jinja.palletsprojects.com/

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.

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 k> 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 root 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.

1 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 ' »' 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 = ' >' %}
```

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_url

The relative path to the HTML favicon image from the current document, or URL to the favicon, or ''.

Added 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 reStructuredText document sources are copied (if html_copy_source is True).

language

The value of language.

last_updated

The build date.

logo_url

The relative path to the HTML logo image from the current document, or URL to the logo, or ''.

Added in version 4.0.

master_doc

root_doc

The value of master_doc or root_doc (aliases), for usage with pathto().

Added in version 4.0: The root_doc template variable.

pagename

The "page name" of the current file, i.e. either the document name if the file is generated from a re-StructuredText 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 represented as a string for example "3.5.1".

sphinx_version_tuple

The version of Sphinx used to build represented as a tuple of five elements. For Sphinx version 3.5.1 beta 3 this would be (3, 5, 1, 'beta', 3). The fourth element can be one of: alpha, beta, rc, final. final always has 0 as the last element.

Added in version 4.2.

docutils_version_info

The version of Docutils used to build represented as a tuple of five elements. For Docutils version 0.16.1 beta 2 this would be (0, 16, 1, 'beta', 2). The fourth element can be one of: alpha, beta, candidate, final. final always has 0 as the last element.

Added in version 5.0.2.

styles

A list of the names of the main stylesheets as given by the theme or html_style.

Added in version 5.1.

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 https://doi.org/10.2016/journal.com/

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 meta⁴⁹¹ 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.

⁴⁹¹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#meta

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.

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) ends with .jinja or _t, it will be processed by the template engine. The suffix will be removed from the final file name.

For example, a theme with a static/theme_styles.css.jinja file could use templating to put options into the stylesheet. When a documentation project is built with that theme, the output directory will contain a _static/theme_styles.css file where all template tags have been processed.

Changed in version 7.4:

The preferred suffix for static templates is now .jinja, in line with the Jinja project's recommended file extension⁴⁹².

The _t file suffix for static templates is now considered 'legacy', and support may eventually be removed.

If a static template with either a _t suffix or a .jinja suffix is detected, it will be processed by the template engine, with the suffix removed from the final file name.

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:

⁴⁹² https://jinja.palletsprojects.com/en/latest/templates/#template-file-extension

```
imykey: 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



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

By default, Sphinx copies static files on the static/ directory of the template directory. However, if your package needs to place static files outside of the static/ directory for some reasons, you need to copy them to the _static/ directory of HTML outputs manually at the build via an event hook. Here is an example of code to accomplish this:

```
import shutil

def copy_custom_files(app, exc):
    if app.builder.format == 'html' and not exc:
        static_dir = app.outdir / '_static'
        shutil.copyfile('path/to/myextension/_static/myjsfile.js', static_dir)

def setup(app):
    app.connect('build-finished', 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:

```
# 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', 0, 'html')
    # 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.

SPHINX API

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 builders 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 Event callbacks API.

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.



1 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 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 the 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 documentation⁴⁹³. 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.

⁴⁹³ https://docutils.sourceforge.io/docs/ref/doctree.html



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 Extending the build process.

6.3 Extension metadata

Added 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.

1 Note

The *parallel-read-safe* extension must satisfy the following conditions:

- The core logic of the extension is parallelly executable during the reading phase.
- It has event handlers for *env-merge-info* and *env-purge-doc* events if it stores data to the build environment object (env) during the reading phase.
- '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.

1 Note

The parallel-write-safe extension must satisfy the following conditions:

- The core logic of the extension is parallelly executable during the writing phase.

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^{494}) \rightarrow None⁴⁹⁵

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.

static Sphinx.require_sphinx(version: $tuple^{496}[int^{497}, int^{498}] \mid str^{499}) \rightarrow None^{500}$

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 or (major, minor).

Added in version 1.0.

Changed in version 7.1: Type of version now allows (major, minor) form.

- Sphinx.connect(event: Literal['config-inited'], callback: Callable[[Sphinx, Config], None⁵⁰¹], priority: $int^{502} = 500$) $\rightarrow int^{503}$
- Sphinx.connect(event: Literal['builder-inited'], callback: Callable[[Sphinx], None⁵⁰⁴], priority: $int^{505} = 500$) $\rightarrow int^{506}$
- Sphinx.connect(event: Literal['env-get-outdated'], callback: Callable[[Sphinx, BuildEnvironment, Set[str⁵⁰⁷], $Set[str^{508}]$, $Set[str^{509}]$], $Sequence[str^{510}]$], $priority: int^{511} = 500$) $\rightarrow int^{512}$
- Sphinx.connect(event: Literal['env-before-read-docs'], callback: Callable[[Sphinx, BuildEnvironment, $list^{513}[str^{514}]]$, None⁵¹⁵], priority: $int^{516} = 500$) $\rightarrow int^{517}$
- Sphinx.connect(event: Literal['env-purge-doc'], callback: Callable[[Sphinx, BuildEnvironment, str^{518}], None⁵¹⁹], priority: $int^{520} = 500$) $\rightarrow int^{521}$
- Sphinx.connect(event: Literal['source-read'], callback: Callable[[Sphinx, str^{522} , $list^{523}[str^{524}]$], None⁵²⁵], priority: $int^{526} = 500$) $\rightarrow int^{527}$
- Sphinx.connect(event: Literal['include-read'], callback: Callable[[Sphinx, Path, str^{528} , $list^{529}[str^{530}]$], $None^{531}$], $priority: int^{532} = 500$) $\rightarrow int^{533}$
- Sphinx.connect(event: Literal['doctree-read'], callback: Callable[[Sphinx, nodes.document], None⁵³⁴], priority: $int^{535} = 500$) $\rightarrow int^{536}$
- Sphinx.connect(event: Literal['env-merge-info'], callback: Callable[[Sphinx, BuildEnvironment, list⁵³⁷[str⁵³⁸], BuildEnvironment], None⁵³⁹], priority: $int^{540} = 500$) $\rightarrow int^{541}$

⁴⁹⁴ https://docs.python.org/3/library/stdtypes.html#str

⁴⁹⁵ https://docs.python.org/3/library/constants.html#None

⁴⁹⁶ https://docs.python.org/3/library/stdtypes.html#tuple

⁴⁹⁷ https://docs.python.org/3/library/functions.html#int

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/constants.html#None

- Sphinx.connect(event: Literal['env-updated'], callback: Callable[[Sphinx, BuildEnvironment], str^{542}], priority: $int^{543} = 500$) $\rightarrow int^{544}$
- Sphinx.connect(event: Literal['env-get-updated'], callback: Callable[[Sphinx, BuildEnvironment], $Iterable[str^{545}]]$, priority: $int^{546} = 500$) $\rightarrow int^{547}$
- Sphinx.connect(event: Literal['env-check-consistency'], callback: Callable[[Sphinx, BuildEnvironment], $None^{548}$], priority: $int^{549} = 500$) $\rightarrow int^{550}$
- Sphinx.connect(event: Literal['write-started'], callback: Callable[[Sphinx, Builder], None⁵⁵¹], priority: $int^{552} = 500$) $\rightarrow int^{553}$
- Sphinx.connect(event: Literal['doctree-resolved'], callback: Callable[[Sphinx, nodes.document, str^{554}], None⁵⁵⁵], priority: $int^{556} = 500$) $\rightarrow int^{557}$
- Sphinx.connect(event: Literal['missing-reference'], callback: Callable[[Sphinx, BuildEnvironment, addnodes.pending_xref, nodes.TextElement], nodes.reference | None⁵⁵⁸], priority: int⁵⁵⁹ = 500) \rightarrow int⁵⁶⁰
- Sphinx.connect(event: Literal['warn-missing-reference'], callback: Callable[[Sphinx, Domain, addnodes.pending xref], $bool^{561} \mid None^{562} \mid, priority: int^{563} = 500) \rightarrow int^{564}$
- Sphinx.connect(event: Literal['build-finished'], callback: Callable[[Sphinx, Exception⁵⁶⁵ | None⁵⁶⁶], None⁵⁶⁷], priority: $int^{568} = 500$) $\rightarrow int^{569}$
- Sphinx.connect(event: Literal['html-collect-pages'], callback: Callable[[Sphinx], Iterable[tuple⁵⁷⁰[str⁵⁷¹, $dict^{572}[str^{573}, Any], str^{574}]]]$, priority: $int^{575} = 500$) $\rightarrow int^{576}$
- Sphinx.connect(event: Literal['html-page-context'], callback: Callable[[Sphinx, str^{577} , str^{578} , $dict^{579}[str^{580}, Any]$, nodes.document], $str^{581} \mid None^{582}$], priority: $int^{583} = 500$) $\rightarrow int^{584}$
- Sphinx.connect(event: Literal['linkcheck-process-uri'], callback: Callable[[Sphinx, str^{585}], $str^{586} \mid None^{587}$], priority: $int^{588} = 500$) $\rightarrow int^{589}$
- Sphinx.connect(event: Literal['object-description-transform'], callback: Callable[[Sphinx, str^{590} , str^{591} , addnodes.desc content], $None^{592}$], priority: $int^{593} = 500$) $\rightarrow int^{594}$
- Sphinx.connect(event: Literal['autodoc-process-docstring'], callback: _AutodocProcessDocstringListener, priority: $int^{595} = 500$) $\rightarrow int^{596}$
- Sphinx.connect(event: Literal['autodoc-before-process-signature'], callback: Callable[[Sphinx, Any, bool⁵⁹⁷], None⁵⁹⁸], priority: $int^{599} = 500$) $\rightarrow int^{600}$
- $\begin{array}{l} \textbf{Sphinx.connect}(\textit{event: Literal['autodoc-process-signature'], callback: Callable[[Sphinx, Literal['module', 'class', 'exception', 'function', 'method', 'attribute'], \textit{str}^{601}, \textit{Any, dict}^{602}[\textit{str}^{603}, \textit{bool}^{604}], \textit{str}^{605} \mid \textit{None}^{606}, \\ \textit{str}^{607} \mid \textit{None}^{608}], \textit{tuple}^{609}[\textit{str}^{610} \mid \textit{None}^{611}, \textit{str}^{612} \mid \textit{None}^{613}] \mid \textit{None}^{614}], \textit{priority: int}^{615} = 500) \rightarrow \\ & \text{int}^{616} \end{array}$
- Sphinx.connect(event: Literal['autodoc-process-bases'], callback: Callable[[Sphinx, str^{617} , Any, $dict^{618}[str^{619}, bool^{620}]$, $list^{621}[str^{622}]$], $None^{623}$], priority: $int^{624} = 500$) $\rightarrow int^{625}$
- Sphinx.connect(event: Literal['autodoc-skip-member'], callback: Callable[[Sphinx, Literal['module', 'class', 'exception', 'function', 'method', 'attribute'], str^{626} , Any, $bool^{627}$, $dict^{628}[str^{629}, bool^{630}]]$, $bool^{631}]$, priority: $int^{632} = 500$) $\rightarrow int^{633}$
- Sphinx.connect(event: Literal['todo-defined'], callback: Callable[[Sphinx, todo_node], None⁶³⁴], priority: $int^{635} = 500$) $\rightarrow int^{636}$
- Sphinx.connect(event: Literal['viewcode-find-source'], callback: Callable[[Sphinx, str^{637}], tuple 638 [str 639 , $dict^{640}$ [str 641 , tuple 642 [Literal['class', 'def', 'other'], int 643 , int 644]]]], priority: int 645 = 500) \rightarrow int 646
- Sphinx.connect(event: Literal['viewcode-follow-imported'], callback: Callable[[Sphinx, str^{647} , str^{648}], str^{649} | None⁶⁵⁰], priority: $int^{651} = 500$) $\rightarrow int^{652}$
- Sphinx.connect(event: str^{653} , callback: Callable[..., Any], priority: $int^{654} = 500$) $\rightarrow int^{655}$ Register callback to be called when event is emitted.
 - For details on available core events and the arguments of callback functions, please see *Event callbacks API*.

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 order of *priority* (ascending).

Returns

A listener ID. It can be used for disconnect().

Changed in version 3.0: Support *priority*

- $^{501}\ https://docs.python.org/3/library/constants.html#None$ 502 https://docs.python.org/3/library/functions.html#int 503 https://docs.python.org/3/library/functions.html#int 504 https://docs.python.org/3/library/constants.html#None 505 https://docs.python.org/3/library/functions.html#int 506 https://docs.python.org/3/library/functions.html#int 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/stdtypes.html#str 510 https://docs.python.org/3/library/stdtypes.html#str 511 https://docs.python.org/3/library/functions.html#int 512 https://docs.python.org/3/library/functions.html#int 513 https://docs.python.org/3/library/stdtypes.html#list 514 https://docs.python.org/3/library/stdtypes.html#str 515 https://docs.python.org/3/library/constants.html#None 516 https://docs.python.org/3/library/functions.html#int 517 https://docs.python.org/3/library/functions.html#int 518 https://docs.python.org/3/library/stdtypes.html#str 519 https://docs.python.org/3/library/constants.html#None 520 https://docs.python.org/3/library/functions.html#int 521 https://docs.python.org/3/library/functions.html#int 522 https://docs.python.org/3/library/stdtypes.html#str 523 https://docs.python.org/3/library/stdtypes.html#list 524 https://docs.python.org/3/library/stdtypes.html#str 525 https://docs.python.org/3/library/constants.html#None 526 https://docs.python.org/3/library/functions.html#int 527 https://docs.python.org/3/library/functions.html#int 528 https://docs.python.org/3/library/stdtypes.html#str 529 https://docs.python.org/3/library/stdtypes.html#list 530 https://docs.python.org/3/library/stdtypes.html#str 531 https://docs.python.org/3/library/constants.html#None 532 https://docs.python.org/3/library/functions.html#int 533 https://docs.python.org/3/library/functions.html#int https://docs.python.org/3/library/constants.html#None https://docs.python.org/3/library/functions.html#int 536 https://docs.python.org/3/library/functions.html#int 537 https://docs.python.org/3/library/stdtypes.html#list 538 https://docs.python.org/3/library/stdtypes.html#str 539 https://docs.python.org/3/library/constants.html#None 540 https://docs.python.org/3/library/functions.html#int 541 https://docs.python.org/3/library/functions.html#int 542 https://docs.python.org/3/library/stdtypes.html#str 543 https://docs.python.org/3/library/functions.html#int 544 https://docs.python.org/3/library/functions.html#int 545 https://docs.python.org/3/library/stdtypes.html#str 546 https://docs.python.org/3/library/functions.html#int 547 https://docs.python.org/3/library/functions.html#int 548 https://docs.python.org/3/library/constants.html#None 549 https://docs.python.org/3/library/functions.html#int 550 https://docs.python.org/3/library/functions.html#int 551 https://docs.python.org/3/library/constants.html#None 552 https://docs.python.org/3/library/functions.html#int 553 https://docs.python.org/3/library/functions.html#int 554 https://docs.python.org/3/library/stdtypes.html#str 555 https://docs.python.org/3/library/constants.html#None 556 https://docs.python.org/3/library/functions.html#int 557 https://docs.python.org/3/library/functions.html#int 558 https://docs.python.org/3/library/constants.html#None $^{559}\ https://docs.python.org/3/library/functions.html\#int$ 560 https://docs.python.org/3/library/functions.html#int 561 https://docs.python.org/3/library/functions.html#bool 562 https://docs.python.org/3/library/constants.html#None 563 https://docs.python.org/3/library/functions.html#int 564 https://docs.python.org/3/library/functions.html#int 565 https://docs.python.org/3/library/exceptions.html#Exception 566 https://docs.python.org/3/library/constants.html#None 567 https://docs.python.org/3/library/constants.html#None 568 https://docs.python.org/3/library/functions.html#int
- 6,49 https://docs.python.org/3/library/functions.html#int https://docs.python.org/3/library/stdtypes.html#uple
- 571 https://docs.python.org/3/library/stdtypes.html#str
- 572 https://docs.python.org/3/library/stdtypes.html#dict
- 573 https://docs.python.org/3/library/stdtypes.html#str python org/3/library/etdtype

```
Sphinx.disconnect(listener id: int^{656}) \rightarrow None<sup>657</sup>
```

Unregister callback by listener_id.

Parameters

listener_id – A listener_id that *connect()* returns

```
Sphinx.add_builder(builder: type^{658}[Builder], override: bool^{659} = False) \rightarrow None<sup>660</sup>
```

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^{661}, default: Any, rebuild: _ConfigRebuild, types: type^{662} \mid Collection[type^{663}] \mid ENUM = (), description: str^{664} = ") \rightarrow None^{665}
```

Register a configuration value.

This is necessary for Sphinx to recognize new values and set default values accordingly.

Parameters

- **name** The name of the 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.
- **description** A short description of the configuration value.

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.

Added in version 7.4: The *description* parameter.

⁶⁵⁶ https://docs.python.org/3/library/functions.html#int

⁶⁵⁷ https://docs.python.org/3/library/constants.html#None

⁶⁵⁸ https://docs.python.org/3/library/functions.html#type

⁶⁵⁹ https://docs.python.org/3/library/functions.html#bool

⁶⁶⁰ https://docs.python.org/3/library/constants.html#None

⁶⁶¹ https://docs.python.org/3/library/stdtypes.html#str

⁶⁶² https://docs.python.org/3/library/functions.html#type

⁶⁶³ https://docs.python.org/3/library/functions.html#type

⁶⁶⁴ https://docs.python.org/3/library/stdtypes.html#str

⁶⁶⁵ https://docs.python.org/3/library/constants.html#None

```
Sphinx.add_event(name: str^{666}) \rightarrow None<sup>667</sup>
```

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^{668}, translator\_class: type^{669}[nodes.NodeVisitor], override: bool^{670} = False)
\rightarrow None^{671}
```

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 a custom translator and define custom nodes for the translator (see add_node()).

Parameters

- name The name of the 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

Added in version 1.3.

Changed in version 1.8: Add override keyword.

```
Sphinx.add_node(node: type<sup>672</sup>[Element], override: bool<sup>673</sup> = False, **kwargs: tuple<sup>674</sup>[Callable, Callable | None<sup>675</sup>]) \rightarrow None<sup>676</sup>
```

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:

```
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>')
```

(continues on next page)

 $^{^{666}\} https://docs.python.org/3/library/stdtypes.html\#str$

⁶⁶⁷ https://docs.python.org/3/library/constants.html#None

 $^{^{668}\} https://docs.python.org/3/library/stdtypes.html\#str$

⁶⁶⁹ https://docs.python.org/3/library/functions.html#type

⁶⁷⁰ https://docs.python.org/3/library/functions.html#bool

⁶⁷¹ https://docs.python.org/3/library/constants.html#None

(continued from previous page)

```
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^{677}[Element], figtype: str^{678}, title\_getter: TitleGetter | None^{679} = None, override: bool^{680} = False, **kwargs: tuple^{681}[Callable, Callable]) \rightarrow None<sup>682</sup>
```

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 figtype has individual numbering sequences. As system figtypes, figure, table and code-block are defined. It is possible to add custom nodes to these default figtypes. It is also possible to define new custom figtype if a 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

Added in version 1.4.

Sphinx.add_directive(name: str^{683} , cls: $type^{684}$ [Directive], override: $bool^{685} = False$) \rightarrow None⁶⁸⁶ Register a Docutils directive.

Parameters

- name The name of the directive
- cls A directive class
- **override** If false, do not install it if another directive is already installed as the same name If true, unconditionally install the directive.

For example, a custom directive named my-directive would be added like this:

⁶⁷² https://docs.python.org/3/library/functions.html#type

⁶⁷³ https://docs.python.org/3/library/functions.html#bool

⁶⁷⁴ https://docs.python.org/3/library/stdtypes.html#tuple

⁶⁷⁵ https://docs.python.org/3/library/constants.html#None

⁶⁷⁶ https://docs.python.org/3/library/constants.html#None

⁶⁷⁷ https://docs.python.org/3/library/constants.html#type

⁶⁷⁸ https://docs.python.org/3/library/stdtypes.html#str

⁶⁷⁹ https://docs.python.org/3/library/constants.html#None

⁶⁸⁰ https://docs.python.org/3/library/functions.html#bool

⁶⁸¹ https://docs.pvthon.org/3/library/stdtvpes.html#tuple

⁶⁸² https://docs.python.org/3/library/constants.html#None

```
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):
    app.add_directive('my-directive', MyDirective)
```

For more details, see the Docutils docs⁶⁸⁷.

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^{688}, role: Any, override: bool^{689} = False) \rightarrow None<sup>690</sup> Register a Docutils role.
```

Parameters

- name The name of role
- role A role function
- **override** If false, do not install it if another role is already installed as the same name If true, unconditionally install the role.

For more details about role functions, see the Docutils docs⁶⁹¹.

Changed in version 1.8: Add override keyword.

```
Sphinx.add_generic_role(name: str^{692}, nodeclass: type^{693}[Node], override: bool^{694} = False) \rightarrow None<sup>695</sup> Register a generic Docutils role.
```

Register a Docutils role that does nothing but wrap its contents in the node given by nodeclass.

Parameters

override – If false, do not install it if another role is already installed as the same name If true, unconditionally install the role.

```
683 https://docs.python.org/3/library/stdtypes.html#str
684 https://docs.python.org/3/library/functions.html#type
685 https://docs.python.org/3/library/functions.html#bool
686 https://docs.python.org/3/library/constants.html#None
687 https://docs.python.org/3/library/stdtypes.html#str
688 https://docs.python.org/3/library/stdtypes.html#str
689 https://docs.python.org/3/library/functions.html#bool
690 https://docs.python.org/3/library/constants.html#None
```

691 https://docutils.sourceforge.io/docs/howto/rst-roles.html

Added in version 0.6.

Changed in version 1.8: Add override keyword.

Sphinx.add_domain(domain: $type^{696}$ [Domain], $override: bool^{697} = False$) \rightarrow None⁶⁹⁸ Register a domain.

Parameters

- domain A domain class
- **override** If false, do not install it if another domain is already installed as the same name If true, unconditionally install the domain.

Added in version 1.0.

Changed in version 1.8: Add override keyword.

```
Sphinx.add_directive_to_domain(domain: str^{699}, name: str^{700}, cls: type^{701}[Directive], override: bool^{702} = False) \rightarrow None^{703}
```

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 false, do not install it if another directive is already installed as the same name If true, unconditionally install the directive.

Added in version 1.0.

Changed in version 1.8: Add *override* keyword.

```
Sphinx.add_role_to_domain(domain: str^{704}, name: str^{705}, role: RoleFunction | XRefRole, override: bool^{706} = False) \rightarrow None^{707}
```

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 the target domain
- name The name of the role
- role The role function

⁶⁹² https://docs.python.org/3/library/stdtypes.html#str

⁶⁹³ https://docs.python.org/3/library/functions.html#type

⁶⁹⁴ https://docs.python.org/3/library/functions.html#bool

⁶⁹⁵ https://docs.python.org/3/library/constants.html#None

⁶⁹⁶ https://docs.python.org/3/library/functions.html#type

⁶⁹⁷ https://docs.python.org/3/library/functions.html#bool

⁶⁹⁸ https://docs.python.org/3/library/constants.html#None

⁶⁹⁹ https://docs.python.org/3/library/stdtypes.html#str

⁷⁰⁰ https://docs.python.org/3/library/stdtypes.html#str

⁷⁰¹ https://docs.python.org/3/library/functions.html#type

⁷⁰² https://docs.python.org/3/library/functions.html#bool

⁷⁰³ https://docs.python.org/3/library/constants.html#None

• **override** – If false, do not install it if another role is already installed as the same name If true, unconditionally install the role.

Added in version 1.0.

Changed in version 1.8: Add override keyword.

```
Sphinx.add_index_to_domain(domain: str^{708}, index: type^{709}[Index], \_override: bool^{710} = False) \rightarrow None<sup>711</sup> Register a custom index for a domain.
```

Add a custom *index* class to the domain named *domain*.

Parameters

- domain The name of the target domain
- index The index class
- **override** If false, do not install it if another index is already installed as the same name If true, unconditionally install the index.

Added in version 1.0.

Changed in version 1.8: Add override keyword.

```
Sphinx.add_object_type(directivename: str^{712}, rolename: str^{713}, indextemplate: str^{714} = ", parse_node: Callable | None<sup>715</sup> = None, ref_nodeclass: type^{716}[nodes.TextElement] | None<sup>717</sup> = None, objname: str^{718} = ", doc_field_types: Sequence = (), override: bool^{719} = False) \rightarrow None^{720}
```

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:

⁷⁰⁴ https://docs.python.org/3/library/stdtypes.html#str

 $^{^{705}\} https://docs.python.org/3/library/stdtypes.html#str$

⁷⁰⁶ https://docs.python.org/3/library/functions.html#bool

https://docs.python.org/3/library/constants.html#None

⁷⁰⁸ https://docs.python.org/3/library/stdtypes.html#str

⁷⁰⁹ https://docs.python.org/3/library/functions.html#type

⁷¹⁰ https://docs.python.org/3/library/functions.html#bool

⁷¹¹ https://docs.python.org/3/library/constants.html#None

```
.. rst:directive:: function

Document a function.
<...>
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 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^{721}, rolename: str^{722}, indextemplate: str^{723} = ", ref_nodeclass: type^{724}[nodes.TextElement] \mid None^{725} = None, objname: str^{726} = ", override: bool^{727} = False) \rightarrow None^{728}
```

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:

Example usage:

```
.. topic:: application API
The application API
------
Some random text here.
```

(continues on next page)

⁷¹² https://docs.python.org/3/library/stdtypes.html#str

 $^{^{713}\} https://docs.python.org/3/library/stdtypes.html\#str$

⁷¹⁴ https://docs.python.org/3/library/stdtypes.html#str

⁷¹⁵ https://docs.python.org/3/library/constants.html#None

⁷¹⁶ https://docs.python.org/3/library/functions.html#type

⁷¹⁷ https://docs.python.org/3/library/constants.html#None

⁷¹⁸ https://docs.python.org/3/library/stdtypes.html#str

⁷¹⁹ https://docs.python.org/3/library/functions.html#bool

⁷²⁰ https://docs.python.org/3/library/constants.html#None

(continued from previous page)

See also :topic:`this section <application API>`.

(Of course, the element following the topic directive needn't be a section.)

Parameters

override – If false, do not install it if another cross-reference type is already installed as the same name If true, unconditionally install the cross-reference type.

Changed in version 1.8: Add override keyword.

Sphinx.add_transform(transform: $type^{729}[Transform]$) \rightarrow None⁷³⁰

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 Categories⁷³¹

Sphinx.add_post_transform(transform: $type^{732}[Transform]$) \rightarrow None⁷³³

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: $str^{734} \mid None^{735}$, priority: $int^{736} = 500$, loading_method: $str^{737} \mid None^{738} = None$, **kwargs: Any) $\rightarrow None^{739}$

Register a JavaScript file to include in the HTML output.

⁷²¹ https://docs.python.org/3/library/stdtypes.html#str

⁷²² https://docs.python.org/3/library/stdtypes.html#str

⁷²³ https://docs.python.org/3/library/stdtypes.html#str

⁷²⁴ https://docs.python.org/3/library/functions.html#type

⁷²⁵ https://docs.python.org/3/library/constants.html#None

https://docs.python.org/3/library/stdtypes.html#str

⁷²⁷ https://docs.python.org/3/library/functions.html#bool

⁷²⁸ https://docs.python.org/3/library/constants.html#None

⁷²⁹ https://docs.python.org/3/library/functions.html#type

⁷³⁰ https://docs.python.org/3/library/constants.html#None

⁷³¹ https://docutils.sourceforge.io/docs/ref/transforms.html#transform-priority-range-categories

⁷³² https://docs.python.org/3/library/functions.html#type

⁷³³ https://docs.python.org/3/library/constants.html#None

Parameters

- **filename** The name of a JavaScript file that the default HTML template will include. It must be relative to the HTML static path, or a full URI with scheme, or None . The None value is used to create an inline <script> tag. See the description of *kwargs* below.
- **priority** Files are included in ascending order of priority. If multiple JavaScript files have the same priority, those files will be included in order of registration. See list of "priority range for JavaScript files" below.
- loading_method The loading method for the JavaScript file. Either 'async' or 'defer' are allowed.
- **kwargs** Extra keyword arguments are included as attributes of the <script> tag. If the special keyword argument body is given, its value will be added as the content of the <script> tag.

Example:

```
app.add_js_file('example.js')
# => <script src="_static/example.js"></script>

app.add_js_file('example.js', loading_method="async")
# => <script src="_static/example.js" async="async"></script>

app.add_js_file(None, body="var myVariable = 'foo';")
# => <script>var myVariable = 'foo';</script>
```

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 an extension calls this method on httml-page-context event.

Added 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.

Changed in version 4.4: Take loading_method argument. Allow to change the loading method of the JavaScript file.

```
Sphinx.add_css_file(filename: str^{740}, priority: int^{741} = 500, **kwargs: Any) \rightarrow None<sup>742</sup>
```

Register a stylesheet to include in the HTML output.

Parameters

⁷³⁴ https://docs.python.org/3/library/stdtypes.html#str

⁷³⁵ https://docs.python.org/3/library/constants.html#None

⁷³⁶ https://docs.python.org/3/library/functions.html#int

⁷³⁷ https://docs.python.org/3/library/stdtypes.html#str

⁷³⁸ https://docs.python.org/3/library/constants.html#None

⁷³⁹ https://docs.python.org/3/library/constants.html#None

- **filename** The name of a CSS file that the default HTML template will include. It must be relative to the HTML static path, or a full URI with scheme.
- **priority** Files are included in ascending order of priority. If multiple CSS files have the same priority, those files will be included in order of registration. See list of "priority range for CSS files" below.
- **kwargs** Extra keyword arguments are included as attributes of the 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 an extension calls this method on https://html.page-context event.

Added in version 1.0.

Changed in version 1.6: Optional alternate and/or title attributes can be supplied with the arguments *alternate* (a Boolean) and *title* (a string). The default is no title and *alternate* = False. For more information, refer to the documentation⁷⁴³.

Changed in version 1.8: Renamed from app.add_stylesheet(). And it allows keyword arguments as attributes of link tag.

Changed in version 3.5: Take priority argument. Allow to add a CSS file to the specific page.

```
Sphinx.add_latex_package(packagename: str^{744}, options: str^{745} \mid None^{746} = None, after_hyperref: bool^{747} = False) \rightarrow None^{748}
```

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 the *usepackage* declaration. If you set *after_hyperref* truthy, the package will be loaded after hyperref package.

```
app.add_latex_package('mypackage')
# => \usepackage{mypackage}

(continues on next page)
```

⁷⁴⁰ https://docs.python.org/3/library/stdtypes.html#str

⁷⁴¹ https://docs.python.org/3/library/functions.html#int

⁷⁴² https://docs.python.org/3/library/constants.html#None

⁷⁴³ https://mdn.io/Web/CSS/Alternative_style_sheets

(continued from previous page)

```
app.add_latex_package('mypackage', 'foo,bar')
# => \usepackage[foo,bar]{mypackage}
```

Added in version 1.3.

Added in version 3.1: after_hyperref option.

```
Sphinx.add_lexer(alias: str^{749}, lexer: type^{750}[Lexer]) \rightarrow None<sup>751</sup>
```

Register a new lexer for source code.

Use *lexer* to highlight code blocks with the given language *alias*.

Added in version 0.6.

Changed in version 2.1: Take a lexer class as an argument.

Changed in version 4.0: Removed support for lexer instances as an argument.

```
Sphinx.add_autodocumenter(cls: type<sup>752</sup>[Documenter], override: bool^{753} = False) \rightarrow None<sup>754</sup>
```

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 auto-documenting 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.

See Developing autodoc extensions.

Added in version 0.6.

Changed in version 2.2: Add override keyword.

```
Sphinx.add_autodoc_attrgetter(typ: type^{755}, getter: Callable[[Any, str^{756}, Any], Any]) \rightarrow None^{757}
```

Register a new getattr-like function for the autodoc extension.

Add *getter*, which must be a function with an interface compatible to the $getattr()^{758}$ 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 $getattr()^{759}$.

Added in version 0.6.

Sphinx.add_search_language(cls: type⁷⁶⁰[SearchLanguage]) \rightarrow None⁷⁶¹

Register a new language for the HTML search index.

```
744 https://docs.python.org/3/library/stdtypes.html#str
```

⁷⁴⁵ https://docs.python.org/3/library/stdtypes.html#str

⁷⁴⁶ https://docs.python.org/3/library/constants.html#None

⁷⁴⁷ https://docs.python.org/3/library/functions.html#bool

⁷⁴⁸ https://docs.python.org/3/library/constants.html#None

⁷⁴⁹ https://docs.python.org/3/library/stdtypes.html#str

⁷⁵⁰ https://docs.python.org/3/library/functions.html#type

⁷⁵¹ https://docs.python.org/3/library/constants.html#None

⁷⁵² https://docs.python.org/3/library/functions.html#type

⁷⁵³ https://docs.python.org/3/library/functions.html#bool

⁷⁵⁴ https://docs.python.org/3/library/constants.html#None

https://docs.python.org/3/library/functions.html#type

⁷⁵⁶ https://docs.python.org/3/library/stdtypes.html#str

⁷⁵⁷ https://docs.python.org/3/library/constants.html#None

⁷⁵⁸ https://docs.python.org/3/library/functions.html#getattr

⁷⁵⁹ https://docs.python.org/3/library/functions.html#getattr

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 https://html.search_language.

Added in version 1.1.

```
Sphinx.add_source_suffix(suffix: str^{762}, filetype: str^{763}, override: bool^{764} = False) \rightarrow None Register a suffix of source files.
```

Same as *source_suffix*. The users can override this using the config setting.

Parameters

override – If false, do not install it the same suffix is already installed. If true, unconditionally install the suffix.

Added in version 1.8.

```
Sphinx.add_source_parser(parser: type^{766}[Parser], override: bool^{767} = False) \rightarrow None<sup>768</sup> Register a parser class.
```

Parameters

override – If false, do not install it if another parser is already installed for the same suffix. If true, unconditionally install the parser.

Added 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<sup>769</sup>[EnvironmentCollector]) \rightarrow None<sup>770</sup>
```

Register an environment collector class.

Refer to Environment Collector API.

Added in version 1.6.

```
Sphinx.add_html_theme(name: str^{771}, theme_path: str^{772}) \rightarrow None<sup>773</sup>
```

Register a HTML Theme.

The *name* is a name of theme, and *theme_path* is a full path to the theme (refs: *Distribute your theme as a Python package*).

Added in version 1.6.

```
Sphinx.add_html_math_renderer(name: str^{774}, inline_renderers: tuple^{775}[Callable, Callable | None^{776}] | None^{777} = None, block_renderers: tuple^{778}[Callable, Callable | None^{779}] | None^{780} = None) \rightarrow None^{781}
```

⁷⁶⁰ https://docs.python.org/3/library/functions.html#type

⁷⁶¹ https://docs.python.org/3/library/constants.html#None

⁷⁶² https://docs.python.org/3/library/stdtypes.html#str

⁷⁶³ https://docs.python.org/3/library/stdtypes.html#str

⁷⁶⁴ https://docs.python.org/3/library/functions.html#bool

⁷⁶⁵ https://docs.python.org/3/library/constants.html#None

⁷⁶⁶ https://docs.python.org/3/library/functions.html#type

⁷⁶⁷ https://docs.python.org/3/library/functions.html#bool

⁷⁶⁸ https://docs.python.org/3/library/constants.html#None

⁷⁶⁹ https://docs.python.org/3/library/functions.html#type

https://docs.python.org/3/library/constants.html#None

⁷⁷¹ https://docs.python.org/3/library/stdtypes.html#str

https://docs.python.org/3/library/stdtypes.html#str

https://docs.python.org/3/library/constants.html#None

Register a math renderer for HTML.

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.

Added in version 1.8.

```
Sphinx.add_message_catalog(catalog: str^{782}, locale\_dir: str^{783}) \rightarrow None<sup>784</sup>
```

Register a message catalog.

Parameters

- catalog The name of the catalog
- locale_dir The base path of the message catalog

For more details, see *sphinx.locale.get_translation()*.

Added in version 1.8.

```
Sphinx.is_parallel_allowed(typ: str^{785}) \rightarrow bool^{786}
```

Check whether 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^{787}, *args: Any, allowed_exceptions: tuple^{788}[type^{789}[Exception^{790}], ...] = ()) <math>\rightarrow list^{791} 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

 $^{^{774}\} https://docs.python.org/3/library/stdtypes.html\#str$

⁷⁷⁵ https://docs.python.org/3/library/stdtypes.html#tuple

https://docs.python.org/3/library/constants.html#None

https://docs.python.org/3/library/constants.html#None

https://docs.python.org/3/library/stdtypes.html#tuple

⁷⁷⁹ https://docs.python.org/3/library/constants.html#None

https://docs.python.org/3/library/constants.html#None
 https://docs.python.org/3/library/constants.html#None

⁷⁸² https://docs.python.org/3/library/stdtypes.html#str

⁷⁸³ https://docs.python.org/3/library/stdtypes.html#str

https://docs.python.org/3/library/stdtypes.html#str

⁷⁸⁴ https://docs.python.org/3/library/constants.html#None

⁷⁸⁵ https://docs.python.org/3/library/stdtypes.html#str 786 https://docs.python.org/3/library/functions.html#bool

```
emit_firstresult(event: str^{792}, *args: Any, allowed_exceptions: tuple^{793}[type^{794}[Exception^{795}], ...] = ())
                         \rightarrow 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

Added 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.fresh_env_used

True/False as to whether a new environment was created for this build, or None if the environment has not been initialised yet.

Sphinx core events



1 Note

Moved to Event callbacks API.

Checking the Sphinx version

Use this to adapt your extension to API changes in Sphinx.

⁷⁸⁷ https://docs.python.org/3/library/stdtypes.html#str

⁷⁸⁸ https://docs.python.org/3/library/stdtypes.html#tuple

⁷⁸⁹ https://docs.python.org/3/library/functions.html#type

⁷⁹⁰ https://docs.python.org/3/library/exceptions.html#Exception

⁷⁹¹ https://docs.python.org/3/library/stdtypes.html#list

⁷⁹² https://docs.python.org/3/library/stdtypes.html#str

⁷⁹³ https://docs.python.org/3/library/stdtypes.html#tuple

⁷⁹⁴ https://docs.python.org/3/library/functions.html#type

⁷⁹⁵ https://docs.python.org/3/library/exceptions.html#Exception

```
sphinx.version_info = (8, 1, 2, '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.

Added in version 1.2: Before version 1.2, check the string sphinx.__version__.

The Config object

```
class sphinx.config.Config: dict^{796}[str^{797}, Any^{798}] \mid None^{799} = None, overrides: <math>dict^{800}[str^{801}, Any^{802}] \mid None^{803} = None)
```

Configuration file abstraction.

The Config object makes the values of all config options available as attributes.

It is exposed via the *Sphinx*.config and *sphinx*.environment.BuildEnvironment.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: Theme | None<sup>804</sup> = None, dirs: list^{805}[str^{806}] | None<sup>807</sup> = None<sup>808</sup> 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() \rightarrow float^{809}
```

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^{810}, context: dict^{811}[str^{812}, Any]) \rightarrow None<sup>813</sup>
```

Called by the builder to render a template given as a filename with a specified context (a Python dictionary).

```
render_string(template: str^{814}, context: dict^{815}) \rightarrow str^{816}
```

Called by the builder to render a template given as a string with a specified context (a Python dictionary).

⁷⁹⁶ https://docs.python.org/3/library/stdtypes.html#dict

⁷⁹⁷ https://docs.python.org/3/library/stdtypes.html#str

⁷⁹⁸ https://docs.python.org/3/library/typing.html#typing.Any

⁷⁹⁹ https://docs.python.org/3/library/constants.html#None

⁸⁰⁰ https://docs.python.org/3/library/stdtypes.html#dict

⁸⁰¹ https://docs.python.org/3/library/stdtypes.html#str

⁸⁰² https://docs.python.org/3/library/typing.html#typing.Any

⁸⁰³ https://docs.python.org/3/library/constants.html#None

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^{817}, orig\_exc: Exception^{818} | None^{819} = None, modname: <math>str^{820} | None^{821} = None)
```

Extension error.

```
exception sphinx.errors.ThemeError
```

Theme error.

exception sphinx.errors.VersionRequirementError

Incompatible Sphinx version error.

Event callbacks API

Connecting callback functions to events is a simple way to extend Sphinx, by hooking into the build process at various points.

Use *Sphinx.connect()* in an extension's setup function, or a setup function in your projects conf.py, to connect functions to the events:

⁸⁰⁴ https://docs.python.org/3/library/constants.html#None

⁸⁰⁵ https://docs.python.org/3/library/stdtypes.html#list

⁸⁰⁶ https://docs.python.org/3/library/stdtypes.html#str

⁸⁰⁷ https://docs.python.org/3/library/constants.html#None

https://docs.python.org/3/library/constants.html#None

https://docs.python.org/3/library/functions.html#float

⁸¹⁰ https://docs.python.org/3/library/stdtypes.html#str

⁸¹¹ https://docs.python.org/3/library/stdtypes.html#dict

⁸¹² https://docs.python.org/3/library/stdtypes.html#str

⁸¹³ https://docs.python.org/3/library/constants.html#None

⁸¹⁴ https://docs.python.org/3/library/stdtypes.html#str

⁸¹⁵ https://docs.python.org/3/library/stdtypes.html#dict

⁸¹⁶ https://docs.python.org/3/library/stdtypes.html#str

⁸¹⁷ https://docs.python.org/3/library/stdtypes.html#str

⁸¹⁸ https://docs.python.org/3/library/exceptions.html#Exception

https://docs.python.org/3/library/constants.html#None

⁸²⁰ https://docs.python.org/3/library/stdtypes.html#str

⁸²¹ https://docs.python.org/3/library/constants.html#None

(continued from previous page)

```
def setup(app):
    app.connect('source-read', source_read_handler)
```

→ See also

Extensions can add their own events by using Sphinx.add_event(), and calling them them with Sphinx.emit() or Sphinx.emit_firstresult().

Core events overview

Below is an overview of the core event that happens during a build.

```
    event.config-inited(app,config)

event.builder-inited(app)
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:
      source-read(app, docname, source)
      7. run source parsers: text -> docutils.document
         - parsers can be added with the app.add_source_parser() API
         - event.include-read(app, relative_path, parent_docname, content)
           is called for each include directive
      8. apply transforms based on priority: docutils.document -> docutils.document
         - event.doctree-read(app, doctree) is called in the middle of transforms,
           transforms come before/after this event depending on their priority.
9. event.env-merge-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)
if environment is written to disk:
   12. event.env-check-consistency(app, env)
13. event.write-started(app, builder)
    - This is called after ``app.parallel_ok`` has been set,
      which must not be altered by any event handler.
# 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:
   14. apply post-transforms (by priority): docutils.document -> docutils.document
```

(continues on next page)

(continued from previous page)

- 15. 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)
- 16. Generate output files
- 17. event.build-finished(app, exception)

Here is also a flow diagram of the events, within the context of the Sphinx build process:

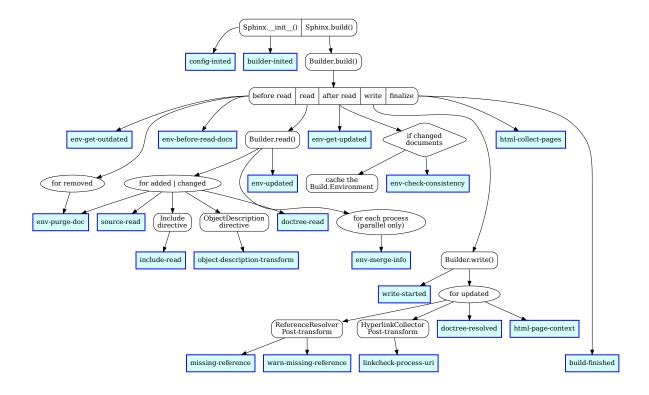


Fig. 1: Sphinx core events flow

Core event details

Here is a more detailed list of these events.

config-inited(app, config)

Parameters

- app Sphinx
- config Config

Emitted when the config object has been initialized.

Added in version 1.8.

builder-inited(app)

Parameters

```
app - Sphinx
```

Emitted when the builder object has been created (available as app.builder).

env-get-outdated(app, env, added, changed, removed)

Parameters

- app Sphinx
- env BuildEnvironment
- added Set[str]
- changed Set[str]
- removed Set[str]

Returns

Sequence[str] of additional docnames to re-read

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.

Added in version 1.1.

env-purge-doc(app, env, docname)

Parameters

- app Sphinx
- env BuildEnvironment
- **docname** str

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.

Added in version 0.5.

env-before-read-docs(app, env, docnames)

Parameters

- app Sphinx
- env BuildEnvironment
- docnames list[str]

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 *found_docs*).

You can also remove document names; do this with caution since it will make Sphinx treat changed files as unchanged.

Added in version 1.3.

source-read(app, docname, content)

Parameters

- app Sphinx
- docname str
- content list[str] with a single element, representing the content of the included file.

Emitted when a source file has been read.

You can process the content 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:`...`.

Added in version 0.5.

include-read(app, relative_path, parent_docname, content)

Parameters

- app Sphinx
- **relative_path** Path⁸²² representing the included file relative to the *source directory*.
- parent_docname str of the document name that contains the include 823 directive.
- **content** list[str] with a single element, representing the content of the included file.

Emitted when a file has been read with the include 824 directive.

You can process the content and replace this item to transform the included content, as with the *source-read* event.

Added in version 7.2.5.

→ See also

The include⁸²⁵ directive and the *source-read* event.

 ${\bf object-description-transform} (app, domain, objtype, content node)$

Parameters

- app Sphinx
- domain str
- **objtype** str
- contentnode desc_content

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.

Added in version 2.4.

⁸²² https://docs.python.org/3/library/pathlib.html#pathlib.Path

⁸²³ https://docutils.sourceforge.io/docs/ref/rst/directives.html#include

⁸²⁴ https://docutils.sourceforge.io/docs/ref/rst/directives.html#include

⁸²⁵ https://docutils.sourceforge.io/docs/ref/rst/directives.html#include

doctree-read(app, doctree)

Parameters

- app Sphinx
- doctree docutils.nodes.document

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)

Parameters

- app Sphinx
- env BuildEnvironment
- **node** The *pending_xref* node to be resolved. Its 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.

Returns

A new node to be inserted in the document tree in place of the node, or None to let other handlers try.

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.

Added in version 0.5.

warn-missing-reference(app, domain, node)

Parameters

- app Sphinx
- **domain** The *Domain* of the missing reference.
- **node** The *pending_xref* node that could not be resolved.

Returns

True if a warning was emitted, else None

Emitted when a cross-reference to an object cannot be resolved even after <code>missing-reference</code>. If the event handler can emit warnings for the missing reference, it should return <code>True</code>. The configuration variables <code>nitpick_ignore</code> and <code>nitpick_ignore_regex</code> prevent the event from being emitted for the corresponding nodes.

Added in version 3.4.

${\tt doctree-resolved}(\textit{app}, \textit{doctree}, \textit{docname})$

Parameters

- app Sphinx
- doctree docutils.nodes.document
- docname str

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)

Parameters

- app Sphinx
- env BuildEnvironment
- docnames list[str]
- other BuildEnvironment

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.

Added in version 1.3.

env-updated(app, env)

Parameters

- app Sphinx
- env BuildEnvironment

Returns

iterable of str

Emitted after reading all documents, when 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.

Added in version 0.5.

Changed in version 1.3: The handlers' return value is now used.

env-get-updated(app, env)

Parameters

- app Sphinx
- env BuildEnvironment

Returns

iterable of str

Emitted when the environment determines which source files have changed and should be re-read. You can return an iterable of docnames to re-read.

env-check-consistency(app, env)

Parameters

- app Sphinx
- env BuildEnvironment

Emitted when Consistency checks phase. You can check consistency of metadata for whole of documents.

Added in version 1.6.

write-started(app, builder)

Parameters

- app Sphinx
- builder Builder

Emitted before the builder starts to resolve and write documents.

Added in version 7.4.

build-finished(app, exception)

Parameters

- app Sphinx
- exception Exception or None

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.

Added in version 0.5.

Builder specific events

These events are emitted by specific builders.

html-collect-pages(app)

Parameters

```
app - Sphinx
```

Returns

iterable of (pagename, context, templatename) where *pagename* and *templatename* are strings and *context* is a dict[str, Any].

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.

Added in version 1.0.

html-page-context(app, pagename, templatename, context, doctree)

Parameters

- app Sphinx
- pagename str
- templatename str
- context dict[str, Any]
- doctree docutils.nodes.document or None

Returns

str or None

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 reStructuredText 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.

The *doctree* argument will be a doctree when the page is created from a reStructuredText 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.

Ţip

You can install JS/CSS files for the specific page via $Sphinx.add_js_file()$ and $Sphinx.add_css_file()$ (since v3.5.0).

Added in version 0.4.

Changed in version 1.3: The return value can now specify a template name.

linkcheck-process-uri(app, uri)

Parameters

- app Sphinx
- uri str of the collected URI

Returns

str or None

Emitted when the linkcheck builder collects hyperlinks from document.

The event handlers can modify the URI by returning a string.

Added in version 4.1.

Project API

```
class sphinx.project.Project(srcdir: str<sup>826</sup> | os.PathLike<sup>827</sup>[str<sup>828</sup>], source_suffix: Iterable[str<sup>829</sup>])
```

A project is the source code set of the Sphinx document(s).

```
\mathbf{discover}(exclude\_paths: Iterable[str^{830}] = (), include\_paths: Iterable[str^{831}] = ('**',)) \rightarrow \operatorname{set}^{832}[\operatorname{str}^{833}]
```

Find all document files in the source directory and put them in docnames.

```
doc2path(docname: str^{834}, absolute: bool^{835}) \rightarrow StrPath
```

Return the filename for the document name.

If absolute is True, return as an absolute path. Else, return as a relative path to the source directory.

```
path2doc(filename: str^{836} | PathLike^{837}[str^{838}]) \rightarrow str^{839} | None^{840}
```

Return the docname for the filename if the file is a document.

filename should be absolute or relative to the source directory.

```
restore(other: Project) → None<sup>841</sup>
Take over a result of last build.

docnames: set<sup>842</sup>[str<sup>843</sup>]
The name of documents belonging to this project.

source_suffix
source_suffix. Same as source_suffix.

srcdir
Source directory.
```

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).
```

```
827 https://docs.python.org/3/library/os.html#os.PathLike
828 https://docs.python.org/3/library/stdtypes.html#str
https://docs.python.org/3/library/stdtypes.html#str
830 https://docs.python.org/3/library/stdtypes.html#str
831 https://docs.python.org/3/library/stdtypes.html#str
832 https://docs.python.org/3/library/stdtypes.html#set
833 https://docs.python.org/3/library/stdtypes.html#str
834 https://docs.python.org/3/library/stdtypes.html#str
835 https://docs.python.org/3/library/functions.html#bool
836 https://docs.python.org/3/library/stdtypes.html#str
https://docs.python.org/3/library/os.html#os.PathLike
838 https://docs.python.org/3/library/stdtypes.html#str
839 https://docs.python.org/3/library/stdtypes.html#str
840 https://docs.python.org/3/library/constants.html#None
https://docs.python.org/3/library/constants.html#None
842 https://docs.python.org/3/library/stdtypes.html#set
843 https://docs.python.org/3/library/stdtypes.html#str
```

826 https://docs.python.org/3/library/stdtypes.html#str

titles

Dictionary mapping docnames to the docutils node for their main title.

docname

Returns the documen of the document currently being parsed.

parser

Returns the parser being used for to parse the current document.

Utility methods

```
doc2path(docname: str^{844}, base: bool^{845} = True) \rightarrow StrPath
```

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:
$$str^{846}$$
, docname: $str^{847} \mid None^{848} = None$) $\rightarrow tuple^{849} [str^{850}, str^{851}]$

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^{852}) \rightarrow None<sup>853</sup>
```

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^{854} = ") \rightarrow int^{855}
```

Return a serial number, e.g. for index entry targets.

The number is guaranteed to be unique in the current document.

```
note\_reread() \rightarrow None^{856}
```

Add the current document to the list of documents that will automatically be re-read at the next build.

Builder API

class sphinx.builders.Builder

This is the base class for all builders.

It follows this basic workflow:

 $^{^{844}\} https://docs.python.org/3/library/stdtypes.html\#str$

⁸⁴⁵ https://docs.python.org/3/library/functions.html#bool

⁸⁴⁶ https://docs.python.org/3/library/stdtypes.html#str

⁸⁴⁷ https://docs.python.org/3/library/stdtypes.html#str

https://docs.python.org/3/library/constants.html#None

https://docs.python.org/3/library/stdtypes.html#tuple

https://docs.python.org/3/library/stdtypes.html#str

⁸⁵¹ https://docs.python.org/3/library/stdtypes.html#str

⁸⁵² https://docs.python.org/3/library/stdtypes.html#str

⁸⁵³ https://docs.python.org/3/library/constants.html#None

https://docs.python.org/3/library/stdtypes.html#str

⁸⁵⁵ https://docs.python.org/3/library/functions.html#int

⁸⁵⁶ https://docs.python.org/3/library/constants.html#None



Fig. 2: Call graph for the standard Sphinx build workflow

Overridable Attributes

These class attributes should be set on builder sub-classes:

name: $str^{857} = "$

The builder's name. This is the value used to select builders on the command line.

format: $str^{858} = "$

The builder's output format, or "if no document output is produced. This is commonly the file extension, e.g. "html", though any string value is accepted. The builder's format string can be used by various components such as *SphinxPostTransform* or extensions to determine their compatibility with the builder.

epilog: $str^{859} = "$

The message emitted upon successful build completion. This can be a printf-style template string with the following keys: outdir, project

allow_parallel: bool⁸⁶⁰ = False

Whether it is safe to make parallel write_doc() calls.

 $supported_image_types: list^{861}[str^{862}] = []$

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: bool⁸⁶³ = False

The builder can produce output documents that may fetch external images when opened.

supported_data_uri_images: bool⁸⁶⁴ = False

The file format produced by the builder allows images to be embedded using data-URIs.

default_translator_class: type⁸⁶⁵[nodes.NodeVisitor]

default translator class for the builder. This can be overridden by set_translator().

Core Methods

These methods define the core build workflow and must not be overridden:

final build_all() \rightarrow None⁸⁶⁶

Build all source files.

final build_specific(filenames: $list^{867}[str^{868}]) \rightarrow None^{869}$

Only rebuild as much as needed for changes in the *filenames*.

final build_update() \rightarrow None⁸⁷⁰

Only rebuild what was changed or added since last build.

final build(docnames: $Iterable[str^{871}] \mid None^{872}$, summary: $str^{873} \mid None^{874} = None$, method: $Literal['all', 'specific', 'update'] = 'update') \rightarrow None^{875}$

Main build method, usually called by a specific build_* method.

First updates the environment, and then calls write().

final read()
$$\rightarrow$$
 list⁸⁷⁶[str⁸⁷⁷]

(Re-)read all files new or changed since last update.

Store all environment docnames in the canonical format (ie using SEP as a separator in place of os.path.sep).

```
final read_doc(docname: str^{878}, *, cache: bool^{879} = True) \rightarrow None<sup>880</sup>
```

Parse a file and add/update inventory entries for the doctree.

final write_doctree($docname: str^{881}, doctree: document, *, _cache: <math>bool^{882} = True$) \rightarrow None⁸⁸³ Write the doctree to a file, to be used as a cache by re-builds.

final write(build_docnames: Iterable[str⁸⁸⁴] | None⁸⁸⁵, updated_docnames: Sequence[str⁸⁸⁶], method: Literal['all', 'specific', 'update'] = 'update') \rightarrow None⁸⁸⁷

Write builder specific output files.

Abstract Methods

These must be implemented in builder sub-classes:

```
get\_outdated\_docs() \rightarrow str^{888} | Iterable[str^{889}]
```

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.

write_doc(docname: str^{890} , doctree: document) \rightarrow None⁸⁹¹

Write the output file for the document

Parameters

- **docname** the *docname*.
- **doctree** defines the content to be written.

The output filename must be determined within this method, typically by calling <code>get_target_uri()</code> or <code>get_relative_uri()</code>.

```
get_target_uri(docname: str^{892}, typ: str^{893} \mid None^{894} = None) \rightarrow str^{895}
```

Return the target URI for a document name.

typ can be used to qualify the link characteristic for individual builders.

Overridable Methods

These methods can be overridden in builder sub-classes:

```
init() \rightarrow None^{896}
```

Load necessary templates and perform initialization. The default implementation does nothing.

```
write_documents(docnames: Set[str^{897}]) \rightarrow None^{898}
```

Write all documents in docnames.

This method can be overridden if a builder does not create output files for each document.

```
prepare_writing(docnames: Set[str^{899}]) \rightarrow None^{900}
```

A place where you can add logic before write_doc() is run

```
copy_assets() \rightarrow None^{901}
```

Where assets (images, static files, etc) are copied before writing

```
get_relative_uri(from_{\cdot}: str^{902}, to: str^{903}, typ: str^{904} \mid None^{905} = None) \rightarrow str^{906}
```

Return a relative URI between two source filenames.

Raises

NoUri if there's no way to return a sensible URI.

```
finish() \rightarrow None^{907}
```

Finish the building process.

The default implementation does nothing.

Attributes

Attributes that are callable from the builder instance:

events

An EventManager object.

Overridable Attributes (extensions)

Builder sub-classes can set these attributes to support built-in extensions:

```
supported_linkcode: str<sup>908</sup>
```

By default, the *linkcode* extension will only inject references for an html builder. The supported_linkcode class attribute can be defined in a non-HTML builder to support managing references generated by linkcode. The expected value for this attribute is an expression which is compatible with *only*.

For example, if a builder was named custom-builder, the following can be used:

```
class CustomBuilder(Builder):
    supported_linkcode = 'custom-builder'
    ...
```

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.



This class essentially wraps a sub-set of *Sphinx event callbacks*.

```
857 https://docs.python.org/3/library/stdtypes.html#str
858 https://docs.python.org/3/library/stdtypes.html#str
859 https://docs.python.org/3/library/stdtypes.html#str
860 https://docs.python.org/3/library/functions.html#bool
861 https://docs.python.org/3/library/stdtypes.html#list
862 https://docs.python.org/3/library/stdtypes.html#str
863 https://docs.python.org/3/library/functions.html#bool
https://docs.python.org/3/library/functions.html#bool
865 https://docs.python.org/3/library/functions.html#type
https://docs.python.org/3/library/constants.html#None
867 https://docs.python.org/3/library/stdtypes.html#list
868 https://docs.python.org/3/library/stdtypes.html#str
869 https://docs.python.org/3/library/constants.html#None
870 https://docs.python.org/3/library/constants.html#None
871 https://docs.python.org/3/library/stdtypes.html#str
https://docs.python.org/3/library/constants.html#None
873 https://docs.python.org/3/library/stdtypes.html#str
874 https://docs.python.org/3/library/constants.html#None
875 https://docs.python.org/3/library/constants.html#None
876 https://docs.python.org/3/library/stdtypes.html#list
877 https://docs.python.org/3/library/stdtypes.html#str
878 https://docs.python.org/3/library/stdtypes.html#str
879 https://docs.python.org/3/library/functions.html#bool
880 https://docs.python.org/3/library/constants.html#None
881 https://docs.python.org/3/library/stdtypes.html#str
882 https://docs.python.org/3/library/functions.html#bool
883 https://docs.python.org/3/library/constants.html#None
https://docs.python.org/3/library/stdtypes.html#str
https://docs.python.org/3/library/constants.html#None
886 https://docs.python.org/3/library/stdtypes.html#str
887 https://docs.python.org/3/library/constants.html#None
888 https://docs.python.org/3/library/stdtypes.html#str
889 https://docs.python.org/3/library/stdtypes.html#str
890 https://docs.python.org/3/library/stdtypes.html#str
891 https://docs.python.org/3/library/constants.html#None
892 https://docs.python.org/3/library/stdtypes.html#str
893 https://docs.python.org/3/library/stdtypes.html#str
894 https://docs.python.org/3/library/constants.html#None
895 https://docs.python.org/3/library/stdtypes.html#str
896 https://docs.python.org/3/library/constants.html#None
897 https://docs.python.org/3/library/stdtypes.html#str
898 https://docs.python.org/3/library/constants.html#None
https://docs.python.org/3/library/stdtypes.html#str
900 https://docs.python.org/3/library/constants.html#None
901 https://docs.python.org/3/library/constants.html#None
902 https://docs.python.org/3/library/stdtypes.html#str
903 https://docs.python.org/3/library/stdtypes.html#str
904 https://docs.python.org/3/library/stdtypes.html#str
905 https://docs.python.org/3/library/constants.html#None
906 https://docs.python.org/3/library/stdtypes.html#str
907 https://docs.python.org/3/library/constants.html#None
```

908 https://docs.python.org/3/library/stdtypes.html#str

clear_doc(app: Sphinx, env: BuildEnvironment, docname: str^{909}) \rightarrow None⁹¹⁰

Remove specified data of a document.

This method is called on the removal of the document.

```
See also
env-purge-doc
```

get_outdated_docs(app: Sphinx, env: BuildEnvironment, added: $set^{911}[str^{912}]$, changed: $set^{913}[str^{914}]$, removed: $set^{915}[str^{916}]$) \rightarrow list⁹¹⁷[str^{918}]

Return a list of docnames to re-read.

This method is called before reading the documents.

```
★ See also
env-get-outdated
```

 $get_updated_docs(app: Sphinx, env: BuildEnvironment) \rightarrow list^{919}[str^{920}]$

Return a list of docnames to re-read.

This method is called after reading the whole of documents.

```
★ See also
env-get-updated
```

merge_other(app: Sphinx, env: BuildEnvironment, docnames: $set^{921}[str^{922}]$, other: BuildEnvironment) \rightarrow None⁹²³

Merge in specified data regarding docnames from a different *BuildEnvironment* object which coming from a subprocess in parallel builds.

```
→ See also
env-merge-info
```

process_doc(app: Sphinx, doctree: nodes.document) \rightarrow None⁹²⁴

Process a document and gather specific data from it.

This method is called after the document is read.

```
★ See also
doctree-read
```

Docutils markup API

This section describes the API for adding reStructuredText markup elements (roles and directives).

Roles

Roles follow the interface described below. They have to be registered by an extension using *Sphinx.add_role()* or *Sphinx.add_role_to_domain()*.

```
def role_function(
    role_name: str, raw_source: str, text: str,
    lineno: int, inliner: Inliner,
    options: dict = {}, content: list = [],
) -> tuple[list[Node], list[system_message]]:
    elements = []
    messages = []
    return elements, messages
```

The *options* and *content* parameters are only used for custom roles created via the role⁹²⁵ directive. The return value is a tuple of two lists, the first containing the text nodes and elements from the role, and the second containing any system messages generated. For more information, see the custom role overview⁹²⁶ from Docutils.

Creating custom roles

Sphinx provides two base classes for creating custom roles, SphinxRole and ReferenceRole.

These provide a class-based interface for creating roles, where the main logic must be implemented in your run() method. The classes provide a number of useful methods and attributes, such as self.text, self.config, and self.env. The ReferenceRole class implements Sphinx's title <target> logic, exposing self.target and self.title attributes. This is useful for creating cross-reference 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.

```
909 https://docs.python.org/3/library/stdtypes.html#str
910 https://docs.python.org/3/library/constants.html#None
911 https://docs.python.org/3/library/stdtypes.html#set
912 https://docs.python.org/3/library/stdtypes.html#str
913 https://docs.python.org/3/library/stdtypes.html#set
914 https://docs.python.org/3/library/stdtypes.html#str
915 https://docs.python.org/3/library/stdtypes.html#set
916 https://docs.python.org/3/library/stdtypes.html#str
917 https://docs.python.org/3/library/stdtypes.html#list
918 https://docs.python.org/3/library/stdtypes.html#str
919 https://docs.python.org/3/library/stdtypes.html#list
920 https://docs.python.org/3/library/stdtypes.html#str
921 https://docs.python.org/3/library/stdtypes.html#set
922 https://docs.python.org/3/library/stdtypes.html#str
923 https://docs.python.org/3/library/constants.html#None
924 https://docs.python.org/3/library/constants.html#None
925 https://docutils.sourceforge.io/docs/ref/rst/directives.html#role
^{926}\ \mathrm{https://docutils.sourceforge.io/docs/howto/rst-roles.html}
```

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 ValueError⁹²⁷ or TypeError⁹²⁸ 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.

See also

Creating directives⁹²⁹ HOWTO of the Docutils documentation

⁹²⁷ https://docs.python.org/3/library/exceptions.html#ValueError

⁹²⁸ https://docs.python.org/3/library/exceptions.html#TypeError

⁹²⁹ https://docutils.sourceforge.io/docs/howto/rst-directives.html

Parsing directive content as reStructuredText

Many directives will contain more markup that must be parsed. To do this, use one of the following APIs from the *run()* method:

- SphinxDirective.parse_content_to_nodes()
- SphinxDirective.parse_text_to_nodes()

The first method parses all the directive's content as markup, whilst the second only parses the given *text* string. Both methods return the parsed Docutils nodes in a list.

The methods are used as follows:

```
def run(self) -> list[Node]:
    # either
    parsed = self.parse_content_to_nodes()
    # or
    parsed = self.parse_text_to_nodes('spam spam spam')
    return parsed
```

Note

The above utility methods were added in Sphinx 7.4. Prior to Sphinx 7.4, the following methods should be used to parse content:

- self.state.nested_parse
- sphinx.util.nodes.nested_parse_with_titles() this allows titles in the parsed content.

```
def run(self) -> list[Node]:
    container = docutils.nodes.Element()
    # either
    nested_parse_with_titles(self.state, self.result, container)
    # or
    self.state.nested_parse(self.result, 0, container)
    parsed = container.children
    return parsed
```

To parse inline markup, use *parse_inline()*. This must only be used for text which is a single line or paragraph, and does not contain any structural elements (headings, transitions, directives, etc).

1 Note

sphinx.util.docutils.switch_source_input() allows changing the source (input) file during parsing content in a directive. It is useful to parse mixed content, such as in sphinx.ext.autodoc, where it is used to parse docstrings.

```
from sphinx.util.docutils import switch_source_input
from sphinx.util.parsing import nested_parse_to_nodes

# 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):
    parsed = nested_parse_to_nodes(self.state, self.result)
```

Deprecated since version 1.7: Until Sphinx 1.6, sphinx.ext.autodoc.AutodocReporter was used for this purpose. It is replaced by switch_source_input().

ViewLists and StringLists

Docutils represents document source lines in a StringList class, which inherits from ViewList, both in the docutils.statemachine module. This is a list with extended functionality, including that slicing creates views of the original list and that the list contains information about source line numbers.

The *Directive.content* attribute is a StringList. If you generate content to be parsed as reStructuredText, you have to create a StringList for the Docutils APIs. The utility functions provided by Sphinx handle this automatically. Important for content generation are the following points:

- The ViewList constructor takes a list of strings (lines) and a source (document) name.
- The ViewList.append() method takes a line and a source name as well.

Domain API

class sphinx.domains.**Domain**(env: BuildEnvironment)

Object descriptions are tuples with six items:

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<sup>930</sup>, objtype: ObjType) → None<sup>931</sup>
Add an object type.

check_consistency() → None<sup>932</sup>
Do consistency checks (experimental).

clear_doc(docname: str<sup>933</sup>) → None<sup>934</sup>
Remove traces of a document in the domain-specific inventories.

directive(name: str<sup>935</sup>) → Callable | None<sup>936</sup>
Return a directive adapter class that always gives the registered directive its full name ('domain:name') as self.name.

get_enumerable_node_type(node: Node) → str<sup>937</sup> | None<sup>938</sup>
Get type of enumerable nodes (experimental).

get_full_qualified_name(node: Element) → str<sup>939</sup> | None<sup>940</sup>
Return full qualified name for given node.

get_objects() → Iterable[tuple<sup>941</sup>[str<sup>942</sup>, str<sup>943</sup>, str<sup>944</sup>, str<sup>945</sup>, str<sup>946</sup>, int<sup>947</sup>]]
Return an iterable of "object descriptions".
```

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:

Default priority (placed before full-text matches).

Object is important (placed before default-priority objects).

2 Object is unimportant (placed after full-text matches).

Object should not show up in search at all.

```
get_type_name(type: ObjType, primary: bool^{948} = False) \rightarrow str<sup>949</sup>
```

Return full name for given ObjType.

```
merge_domaindata(docnames: Set[str<sup>950</sup>], otherdata: dict<sup>951</sup>[str<sup>952</sup>, Any]) \rightarrow None<sup>953</sup>
```

Merge in data regarding *docnames* from a different domaindata inventory (coming from a subprocess in parallel builds).

```
process_doc(env: BuildEnvironment, docname: str^{954}, document: nodes.document) \rightarrow None ^{955}
```

Process a document after it is read by the environment.

```
process_field_xref(pnode: pending xref) \rightarrow None<sup>956</sup>
```

Process a pending xref created in a doc field. For example, attach information about the current scope.

```
resolve_any_xref(env: BuildEnvironment, fromdocname: str^{957}, builder: Builder, target: str^{958}, node: pending_xref, contnode: Element) \rightarrow list<sup>959</sup>[tuple<sup>960</sup>[str<sup>961</sup>, 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.

Added in version 1.3.

```
resolve_xref(env: BuildEnvironment, fromdocname: str^{962}, builder: Builder, typ: str^{963}, target: str^{964}, node: pending_xref, contnode: Element| None<sup>965</sup>
```

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.

role name -> role callable

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^{966}) \rightarrow RoleFunction | None<sup>967</sup>
      Return a role adapter function that always gives the registered role its full name ('domain:name') as the
      first argument.
setup() \rightarrow None^{968}
      Set up domain object.
dangling_warnings: dict^{969}[str^{970}, str^{971}] = \{\}
      role name -> a warning message if reference is missing
data: dict^{972}[str^{973}, Any]
      data value
data_version = 0
      data version, bump this when the format of self.data changes
directives: dict<sup>974</sup>[str<sup>975</sup>, type<sup>976</sup>[Directive</sup>]] = {}
      directive name -> directive class
enumerable_nodes: dict<sup>977</sup>[type<sup>978</sup>[Node], tuple<sup>979</sup>[str<sup>980</sup>, TitleGetter | None<sup>981</sup>]] = {}
      node_class -> (enum_node_type, title_getter)
indices: list^{982}[type^{983}[Index]] = []
      a list of Index subclasses
initial_data: dict<sup>984</sup> = {}
      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: dict<sup>985</sup>[str<sup>986</sup>, ObjType] = {}
      type (usually directive) name -> ObjType instance
roles: dict<sup>987</sup>[str<sup>988</sup>, RoleFunction | XRefRole] = {}
```

class sphinx.domains.ObjType(Iname: str⁹⁸⁹,/, *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:

```
930 https://docs.python.org/3/library/stdtypes.html#str
931 https://docs.python.org/3/library/constants.html#None
932 https://docs.python.org/3/library/constants.html#None
933 https://docs.python.org/3/library/stdtypes.html#str
934 https://docs.python.org/3/library/constants.html#None
935 https://docs.python.org/3/library/stdtypes.html#str
936 https://docs.python.org/3/library/constants.html#None
937 https://docs.python.org/3/library/stdtypes.html#str
938 https://docs.python.org/3/library/constants.html#None
939 https://docs.python.org/3/library/stdtypes.html#str
940 https://docs.python.org/3/library/constants.html#None
941 https://docs.python.org/3/library/stdtypes.html#tuple
942 https://docs.python.org/3/library/stdtypes.html#str
943 https://docs.python.org/3/library/stdtypes.html#str
944 https://docs.python.org/3/library/stdtypes.html#str
945 https://docs.python.org/3/library/stdtypes.html#str
946 https://docs.python.org/3/library/stdtypes.html#str
947 https://docs.python.org/3/library/functions.html#int
948 https://docs.python.org/3/library/functions.html#bool
949 https://docs.python.org/3/library/stdtypes.html#str
950 https://docs.python.org/3/library/stdtypes.html#str
951 https://docs.python.org/3/library/stdtypes.html#dict
952 https://docs.python.org/3/library/stdtypes.html#str
953 https://docs.python.org/3/library/constants.html#None
954 https://docs.python.org/3/library/stdtypes.html#str
955 https://docs.python.org/3/library/constants.html#None
956 https://docs.python.org/3/library/constants.html#None
957 https://docs.python.org/3/library/stdtypes.html#str
958 https://docs.python.org/3/library/stdtypes.html#str
959 https://docs.python.org/3/library/stdtypes.html#list
960 https://docs.python.org/3/library/stdtypes.html#tuple
961 https://docs.python.org/3/library/stdtypes.html#str
962 https://docs.python.org/3/library/stdtypes.html#str
963 https://docs.python.org/3/library/stdtypes.html#str
964 https://docs.python.org/3/library/stdtypes.html#str
965 https://docs.python.org/3/library/constants.html#None
966 https://docs.python.org/3/library/stdtypes.html#str
967 https://docs.python.org/3/library/constants.html#None
968 https://docs.python.org/3/library/constants.html#None
969 https://docs.python.org/3/library/stdtypes.html#dict
970 https://docs.python.org/3/library/stdtypes.html#str
971 https://docs.python.org/3/library/stdtypes.html#str
972 https://docs.python.org/3/library/stdtypes.html#dict
973 https://docs.python.org/3/library/stdtypes.html#str
974 https://docs.python.org/3/library/stdtypes.html#dict
975 https://docs.python.org/3/library/stdtypes.html#str
976 https://docs.python.org/3/library/functions.html#type
977 https://docs.python.org/3/library/stdtypes.html#dict
978 https://docs.python.org/3/library/functions.html#type
979 https://docs.python.org/3/library/stdtypes.html#tuple
980 https://docs.python.org/3/library/stdtypes.html#str
981 https://docs.python.org/3/library/constants.html#None
^{982}\ https://docs.python.org/3/library/stdtypes.html#list
983 https://docs.python.org/3/library/functions.html#type
984 https://docs.python.org/3/library/stdtypes.html#dict
985 https://docs.python.org/3/library/stdtypes.html#dict
986 https://docs.python.org/3/library/stdtypes.html#str
987 https://docs.python.org/3/library/stdtypes.html#dict
988 https://docs.python.org/3/library/stdtypes.html#str
```

- *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: 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`).
- *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: Iterable[str^{990}] \mid None^{991} = None) \rightarrow tuple^{992}[list^{993}[tuple^{994}[str^{995}]]
                                list<sup>996</sup>[IndexEntry]]], bool<sup>997</sup>]
```

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 an *IndexEntry*.

```
class sphinx.domains.IndexEntry(name: str<sup>998</sup>, subtype: int<sup>999</sup>, docname: str<sup>1000</sup>, anchor: str<sup>1001</sup>, extra:
                                                       str^{1002}, qualifier: str^{1003}, descr: str^{1004})
```

An index entry.



1 Note

The qualifier and description are not rendered for some output formats, such as LaTeX.

⁹⁸⁹ https://docs.python.org/3/library/stdtypes.html#str

⁹⁹⁰ https://docs.python.org/3/library/stdtypes.html#str

⁹⁹¹ https://docs.python.org/3/library/constants.html#None

⁹⁹² https://docs.python.org/3/library/stdtypes.html#tuple

⁹⁹³ https://docs.python.org/3/library/stdtypes.html#list

⁹⁹⁴ https://docs.python.org/3/library/stdtypes.html#tuple

⁹⁹⁵ https://docs.python.org/3/library/stdtypes.html#str

⁹⁹⁶ https://docs.python.org/3/library/stdtypes.html#list

⁹⁹⁷ https://docs.python.org/3/library/functions.html#bool

```
name: str^{1005}
            The name of the index entry to be displayed.
      subtype: int<sup>1006</sup>
            The sub-entry related type. One of:
                 A normal entry.
            1
                An entry with sub-entries.
            2
                A sub-entry.
      docname: str<sup>1007</sup>
            docname where the entry is located.
      anchor: str<sup>1008</sup>
            Anchor for the entry within docname
      extra: str<sup>1009</sup>
            Extra info for the entry.
      qualifier: str<sup>1010</sup>
            Qualifier for the description.
      descr: str<sup>1011</sup>
            Description for the entry.
class sphinx.directives.ObjectDescription(name, arguments, options, content, lineno, content_offset,
                                                        block_text, state, state_machine)
      Directive to describe a class, function or similar object. Not used directly, but subclassed (in domain-specific
      directives) to add custom behavior.
      _object_hierarchy_parts(sig\_node: desc_signature) \rightarrow tuple<sup>1012</sup>[str^{1013}, ...]
            Returns a tuple of strings, one entry for each part of the object's hierarchy (e.g.
                                                                                                           ('module',
            'submodule', 'Class', 'method')). The returned tuple is used to properly nest children within par-
            ents in the table of contents, and can also be used within the _toc_entry_name() method.
```

```
_toc_entry_name(sig\_node: desc_signature) \rightarrow str<sup>1014</sup>
```

Returns the text of the table of contents entry for the object.

This method must not be used outwith table of contents generation.

```
998 https://docs.python.org/3/library/stdtypes.html#str
999 https://docs.python.org/3/library/functions.html#int
1000 https://docs.python.org/3/library/stdtypes.html#str
1001 https://docs.python.org/3/library/stdtypes.html#str
1002 https://docs.python.org/3/library/stdtypes.html#str
1003 https://docs.python.org/3/library/stdtypes.html#str
1004 https://docs.python.org/3/library/stdtypes.html#str
1005 https://docs.python.org/3/library/stdtypes.html#str
1006 https://docs.python.org/3/library/stdtypes.html#str
1008 https://docs.python.org/3/library/stdtypes.html#str
1009 https://docs.python.org/3/library/stdtypes.html#str
1010 https://docs.python.org/3/library/stdtypes.html#str
1011 https://docs.python.org/3/library/stdtypes.html#str
1011 https://docs.python.org/3/library/stdtypes.html#str
```

This function is called once, in run(), to set the name for the table of contents entry (a special attribute _toc_name is set on the object node, later used in environment.collectors.toctree. TocTreeCollector.process_doc().build_toc() when the table of contents entries are collected).

To support table of contents entries for their objects, domains must override this method, also respecting the configuration setting toc_object_entries_show_parents. Domains must also override _object_hierarchy_parts(), with one (string) entry for each part of the object's hierarchy. The result of this method is set on the signature node, and can be accessed as sig_node['_toc_parts'] for use within this method. The resulting tuple is also used to properly nest children within parents in the table of contents.

An example implementations of this method is within the python domain (PyObject._toc_entry_name()). The python domain sets the _toc_parts attribute within the handle_signature() method.

```
add_target_and_index(name: ObjDescT, sig: str^{1015}, signode: desc_signature) \rightarrow None^{1016}
```

Add cross-reference IDs and entries to self.indexnode, if applicable.

name is whatever handle_signature() returned.

```
after\_content() \rightarrow None^{1017}
```

Called after parsing content. Used to reset information about the current directive context on the build environment.

```
before_content() \rightarrow None<sup>1018</sup>
```

Called before parsing content. Used to set information about the current directive context on the build environment.

```
get_signatures() \rightarrow list^{1019}[str^{1020}]
```

Retrieve the signatures to document from the directive arguments. By default, signatures are given as arguments, one per line.

```
handle_signature(sig: str^{1021}, signode: desc_signature) \rightarrow ObjDescT
```

Parse the signature *sig* into individual nodes and append them to *signode*. If ValueError is raised, parsing is aborted and the whole *sig* is put into a single desc_name node.

The return value should be a value that identifies the object. It is passed to add_target_and_index() unchanged, and otherwise only used to skip duplicates.

```
run() \rightarrow list^{1022}[Node]
```

Main directive entry function, called by docutils upon encountering the directive.

This directive is meant to be quite easily subclassable, so it delegates to several additional methods. What it does:

- find out if called as a domain-specific directive, set self.domain
- create a *desc* node to fit all description inside
- parse standard options, currently no-index
- · create an index node if needed as self.indexnode
- parse all given signatures (as returned by self.get_signatures()) using self.handle_signature(), which should either return a name or raise ValueError
- add index entries using self.add_target_and_index()
- · parse the content and handle doc fields in it

```
Called after creating the content through nested parsing, but before the object-description-transform
                      event is emitted, and before the info-fields are transformed. Can be used to manipulate the content.
           final_argument_whitespace = True
                      May the final argument contain whitespace?
           has content = True
                      May the directive have content?
           option_spec: ClassVar^{1024}[dict^{1025}[str^{1026}, Callable^{1027}[[str^{1028}], Any^{1029}]]] =
            {'no-contents-entry': <function flag>, 'no-index': <function flag>,
            'no-index-entry': <function flag>, 'no-typesetting': <function flag>,
            'nocontentsentry': <function flag>, 'noindex': <function flag>, 'noindexentry':
           <function flag>}
                      Mapping of option names to validator functions.
           optional_arguments = 0
                      Number of optional arguments after the required arguments.
           required_arguments = 1
                      Number of required directive arguments.
Python Domain
class sphinx.domains.python.PythonDomain(env: BuildEnvironment)
           Python language domain.
           objects
           modules
           note\_object(name: str^{1030}, objtype: str^{1031}, node\_id: str^{1032}, aliased: bool^{1033} = False, location: Any^{1034} = False, location = False, loca
                                          None) \rightarrow None<sup>1035</sup>
                      Note a python object for cross reference.
                      Added in version 2.1.
           note\_module(name: str^{1036}, node\_id: str^{1037}, synopsis: str^{1038}, platform: str^{1039}, deprecated: bool^{1040}) \rightarrow
                                         None<sup>1041</sup>
1012 https://docs.python.org/3/library/stdtypes.html#tuple
 1013 https://docs.python.org/3/library/stdtypes.html#str
 1014 https://docs.python.org/3/library/stdtypes.html#str
 1015 https://docs.python.org/3/library/stdtypes.html#str
 1016 https://docs.python.org/3/library/constants.html#None
 1017 https://docs.python.org/3/library/constants.html#None
 1018 https://docs.python.org/3/library/constants.html#None
 1019 https://docs.python.org/3/library/stdtypes.html#list
 1020 https://docs.python.org/3/library/stdtypes.html#str
 1021 https://docs.python.org/3/library/stdtypes.html#str
 1022 https://docs.python.org/3/library/stdtypes.html#list
 1023 https://docs.python.org/3/library/constants.html#None
 1024 https://docs.python.org/3/library/typing.html#typing.ClassVar
 1025 https://docs.python.org/3/library/stdtypes.html#dict
 1026 https://docs.python.org/3/library/stdtypes.html#str
 1027 https://docs.python.org/3/library/collections.abc.html#collections.abc.Callable
 1028 https://docs.python.org/3/library/stdtypes.html#str
 1029 https://docs.python.org/3/library/typing.html#typing.Any
```

transform_content(content node: desc content) \rightarrow None¹⁰²³

Note a python module for cross reference.

Added in version 2.1.

Parser API

The docutils documentation describes 1042 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 sphinx core runtime objects (app, config and env).

```
set_application(app: Sphinx) \rightarrow None^{1043}
```

set_application will be called from Sphinx to set app and other instance variables

Parameters

```
app (sphinx.application.Sphinx) - Sphinx application object
```

config: Config

The config object

env: BuildEnvironment

The environment object

```
1030 https://docs.python.org/3/library/stdtypes.html#str
1031 https://docs.python.org/3/library/stdtypes.html#str
1032 https://docs.python.org/3/library/stdtypes.html#str
1033 https://docs.python.org/3/library/functions.html#bool
1034 https://docs.python.org/3/library/typing.html#typing.Any
1035 https://docs.python.org/3/library/constants.html#None
1036 https://docs.python.org/3/library/stdtypes.html#str
1037 https://docs.python.org/3/library/stdtypes.html#str
1038 https://docs.python.org/3/library/stdtypes.html#str
1039 https://docs.python.org/3/library/stdtypes.html#str
1040 https://docs.python.org/3/library/functions.html#bool
1041 https://docs.python.org/3/library/constants.html#None
1042 https://docs.python.org/3/library/constants.html#parsing-the-document
1043 https://docs.python.org/3/library/constants.html#None
```

Doctree node classes added by Sphinx

Nodes for domain-specific object descriptions

Top-level nodes

These nodes form the top-most levels of object descriptions.

```
class sphinx.addnodes.desc(rawsource=", *children, **attributes)
```

Node for a list of object signatures and a common description of them.

Contains one or more desc_signature nodes and then a single desc_content node.

This node always has two classes:

- The name of the domain it belongs to, e.g., py or cpp.
- The name of the object type in the domain, e.g., function.

```
class sphinx.addnodes.desc_signature(*args: Any1044, **kwargs: Any1045)
```

Node for a single object signature.

As default the signature is a single-line signature. Set is_multiline = True to describe a multi-line signature. In that case all child nodes must be <code>desc_signature_line</code> nodes.

This node always has the classes sig, sig-object, and the domain it belongs to.

```
class sphinx.addnodes.desc_signature_line(rawsource=", text=", *children, **attributes)
```

Node for a line in a multi-line object signature.

It should only be used as a child of a *desc_signature* with is_multiline set to True. Set add_permalink = True for the line that should get the permalink.

```
class sphinx.addnodes.desc_content(rawsource=", *children, **attributes)
```

Node for object description content.

Must be the last child node in a *desc* node.

```
class sphinx.addnodes.desc_inline(domain: str<sup>1046</sup>, *args: Any<sup>1047</sup>, **kwargs: Any<sup>1048</sup>)
```

Node for a signature fragment in inline text.

This is for example used for roles like *cpp:expr*.

This node always has the classes sig, sig-inline, and the name of the domain it belongs to.

Nodes for high-level structure in signatures

These nodes occur in in non-multiline desc_signature nodes and in desc_signature_line nodes.

```
class sphinx.addnodes.desc_name(*args: Any<sup>1049</sup>, **kwargs: Any<sup>1050</sup>)
```

Node for the main object name.

For example, in the declaration of a Python class MyModule.MyClass, the main name is MyClass.

This node always has the class sig-name.

¹⁰⁴⁴ https://docs.python.org/3/library/typing.html#typing.Any

¹⁰⁴⁵ https://docs.python.org/3/library/typing.html#typing.Any

¹⁰⁴⁶ https://docs.python.org/3/library/stdtypes.html#str

¹⁰⁴⁷ https://docs.python.org/3/library/typing.html#typing.Any

https://docs.python.org/3/library/typing.html#typing.Any

¹⁰⁴⁹ https://docs.python.org/3/library/typing.html#typing.Any

¹⁰⁵⁰ https://docs.python.org/3/library/typing.html#typing.Any

```
class sphinx.addnodes.desc_addname(*args: Any<sup>1051</sup>, **kwargs: Any<sup>1052</sup>)
```

Node for additional name parts for an object.

For example, in the declaration of a Python class MyModule. MyClass, the additional name part is MyModule..

This node always has the class sig-prename.

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_parameterlist(rawsource=", text=", *children, **attributes)

Node for a general parameter list.

As default the parameter list is written in line with the rest of the signature. Set multi_line_parameter_list = True to describe a multi-line parameter list. In that case each parameter will then be written on its own, indented line.

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).

Nodes for signature text elements

These nodes inherit *desc_sig_element* and are generally translated to docutils.nodes.inline by SigElementFallbackTransform.

Extensions may create additional desc_sig_*-like nodes but in order for SigElementFallbackTransform to translate them to inline nodes automatically, they must be added to SIG_ELEMENTS via the class keyword argument _sig_element=True of desc_sig_element, e.g.:

```
class desc_custom_sig_node(desc_sig_element, _sig_element=True): ...
```

For backwards compatibility, it is still possible to add the nodes directly using SIG_ELEMENTS. add(desc_custom_sig_node).

```
sphinx.addnodes.SIG_ELEMENTS: set<sup>1053</sup>[type<sup>1054</sup>[desc_sig_element]]
```

A set of classes inheriting *desc_sig_element*. Each node class is expected to be handled by the builder's translator class if the latter does not inherit from SphinxTranslator.

This set can be extended manually by third-party extensions or by subclassing $desc_sig_element$ and using the class keyword argument $_sig_element$ =True.

class sphinx.addnodes.desc_sig_element(rawsource: $str^{1055} = "$, text: $str^{1056} = "$, *children: Element, **attributes: Any)

Common parent class of nodes for inline text of a signature.

¹⁰⁵¹ https://docs.python.org/3/library/typing.html#typing.Any

¹⁰⁵² https://docs.python.org/3/library/typing.html#typing.Any

¹⁰⁵³ https://docs.python.org/3/library/stdtypes.html#set

¹⁰⁵⁴ https://docs.python.org/3/library/functions.html#type

¹⁰⁵⁵ https://docs.python.org/3/library/stdtypes.html#str

https://docs.python.org/3/library/stdtypes.html#str

```
class sphinx.addnodes.desc_sig_space(rawsource: str<sup>1057</sup> = ", text: str<sup>1058</sup> = '', *children: Element,
                                                    **attributes: Anv)
      Node for a space in a signature.
class sphinx.addnodes.desc_sig_name(rawsource: str<sup>1059</sup> = ", text: str<sup>1060</sup> = ", *children: Element,
                                                   **attributes: Any)
      Node for an identifier in a signature.
class sphinx.addnodes.desc_sig_operator(rawsource: str<sup>1061</sup> = ", text: str<sup>1062</sup> = ", *children: Element,
                                                        **attributes: Any)
      Node for an operator in a signature.
class sphinx.addnodes.desc_sig_punctuation(rawsource: str^{1063} = ", text: str^{1064} = ", *children: Element,
                                                             **attributes: Any)
      Node for punctuation in a signature.
class sphinx.addnodes.desc_sig_keyword(rawsource: str<sup>1065</sup> = ", text: str<sup>1066</sup> = ", *children: Element,
                                                       **attributes: Any)
      Node for a general keyword in a signature.
class sphinx.addnodes.desc_sig_keyword_type(rawsource: str<sup>1067</sup> = ", text: str<sup>1068</sup> = ", *children:
                                                              Element, **attributes: Any)
      Node for a keyword which is a built-in type in a signature.
class sphinx.addnodes.desc_sig_literal_number(rawsource: str^{1069} = ", text: str^{1070} = ", *children:
                                                                 Element, **attributes: Any)
      Node for a numeric literal in a signature.
class sphinx.addnodes.desc_sig_literal_string(rawsource: str<sup>1071</sup> = ", text: str<sup>1072</sup> = ", *children:
                                                                 Element, **attributes: Any)
      Node for a string literal in a signature.
class sphinx.addnodes.desc_sig_literal_char(rawsource: str^{1073} = ", text: str^{1074} = ", *children:
                                                              Element, **attributes: Any)
      Node for a character literal in a signature.
1057 https://docs.python.org/3/library/stdtypes.html#str
1058 https://docs.python.org/3/library/stdtypes.html#str
1059 https://docs.python.org/3/library/stdtypes.html#str
1060 https://docs.python.org/3/library/stdtypes.html#str
1061 https://docs.python.org/3/library/stdtypes.html#str
1062 https://docs.python.org/3/library/stdtypes.html#str
1063 https://docs.python.org/3/library/stdtypes.html#str
```

¹⁰⁵⁸ https://docs.python.org/3/library/stdtypes.html#str
1059 https://docs.python.org/3/library/stdtypes.html#str
1060 https://docs.python.org/3/library/stdtypes.html#str
1061 https://docs.python.org/3/library/stdtypes.html#str
1062 https://docs.python.org/3/library/stdtypes.html#str
1063 https://docs.python.org/3/library/stdtypes.html#str
1064 https://docs.python.org/3/library/stdtypes.html#str
1065 https://docs.python.org/3/library/stdtypes.html#str
1066 https://docs.python.org/3/library/stdtypes.html#str
1067 https://docs.python.org/3/library/stdtypes.html#str
1068 https://docs.python.org/3/library/stdtypes.html#str
1069 https://docs.python.org/3/library/stdtypes.html#str
1070 https://docs.python.org/3/library/stdtypes.html#str
1071 https://docs.python.org/3/library/stdtypes.html#str
1072 https://docs.python.org/3/library/stdtypes.html#str
1073 https://docs.python.org/3/library/stdtypes.html#str
1074 https://docs.python.org/3/library/stdtypes.html#str

New admonition-like constructs

```
class sphinx.addnodes.versionmodified(rawsource=", text=", *children, **attributes)
```

Node for version change entries.

Currently used for "versionadded", "versionchanged", "deprecated" and "versionremoved" 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 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.pending_xref_condition(rawsource=", text=", *children, **attributes)
```

Node representing a potential way to create a cross-reference and the condition in which this way should be used.

This node is only allowed to be placed under a *pending_xref* node. A **pending_xref** node must contain either no **pending_xref_condition** nodes or it must only contains **pending_xref_condition** nodes.

The cross-reference resolver will replace a *pending_xref* which contains **pending_xref_condition** nodes by the content of exactly one of those **pending_xref_condition** nodes' content. It uses the **condition** attribute to decide which **pending_xref_condition** node's content to use. For example, let us consider how the cross-reference resolver acts on:

If the cross-reference resolver successfully resolves the cross-reference, then it rewrites the **pending_xref** as:

```
<reference>
     teral>
          StringIO
```

Otherwise, if the cross-reference resolution failed, it rewrites the **pending_xref** as:

```
<reference>
     teral>
        io.StringIO
```

The **pending_xref_condition** node should have **condition** attribute. Domains can be store their individual conditions into the attribute to filter contents on resolving phase. As a reserved condition name, **condition="*"** is used for the fallback of resolution failure. Additionally, as a recommended condition name, **condition="resolved"** represents a resolution success in the intersphinx module.

Added in version 4.0.

```
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.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()

Context manager to postpone logging all logs temporarily.

For example:

```
>>> with pending_logging():
>>> logger.warning('Warning message!') # not flushed yet
>>> some_long_process()
```

(continues on next page)

(continued from previous page)

```
>>> Warning message! # the warning is flushed here
```

sphinx.util.logging.pending_warnings()

Context manager to postpone logging warnings temporarily.

Similar to pending_logging().

```
sphinx.util.logging.prefixed_warnings()
```

Context manager to prepend prefix to all warning log records temporarily.

For example:

```
>>> with prefixed_warnings("prefix:"):
>>> logger.warning('Warning message!') # => prefix: Warning message!
```

Added in version 2.0.

i18n API

```
sphinx.locale.init(locale_dirs: Iterable[str^{1075} | os.PathLike^{1076}[str^{1077}] | None^{1078}], language: str^{1079} | None^{1080}, catalog: str^{1081} = 'sphinx', namespace: str^{1082} = 'general') \rightarrow tuple 1083 [NullTranslations, bool 1084]
```

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^{1085} \mid None^{1086} = None, catalog: str^{1087} = 'sphinx') \rightarrow tuple^{1088}[NullTranslations^{1089}, bool^{1090}]
```

Initialize locale for console.

Added in version 1.8.

sphinx.locale.get_translation($catalog: str^{1091}, namespace: str^{1092} = 'general'$) \rightarrow Callable[[str^{1093}], str^{1094}] Get a translation function based on the catalog and namespace.

The extension can use this API to translate the messages on the extension:

```
import os
from sphinx.locale import get_translation
```

(continues on next page)

```
^{1075}\ https://docs.python.org/3/library/stdtypes.html#str
```

¹⁰⁷⁶ https://docs.python.org/3/library/os.html#os.PathLike

¹⁰⁷⁷ https://docs.python.org/3/library/stdtypes.html#str

¹⁰⁷⁸ https://docs.python.org/3/library/constants.html#None

¹⁰⁷⁹ https://docs.python.org/3/library/stdtypes.html#str

¹⁰⁸⁰ https://docs.python.org/3/library/constants.html#None

¹⁰⁸¹ https://docs.python.org/3/library/stdtypes.html#str

¹⁰⁸² https://docs.python.org/3/library/stdtypes.html#str

¹⁰⁸³ https://docs.python.org/3/library/stdtypes.html#tuple

¹⁰⁸⁴ https://docs.python.org/3/library/functions.html#bool

¹⁰⁸⁵ https://docs.python.org/3/library/stdtypes.html#str

¹⁰⁸⁶ https://docs.python.org/3/library/constants.html#None

¹⁰⁸⁷ https://docs.python.org/3/library/stdtypes.html#str

https://docs.python.org/3/library/stdtypes.html#tuple

 $^{^{1089}\} https://docs.python.org/3/library/gettext.html\#gettext.NullTranslations$

¹⁰⁹⁰ https://docs.python.org/3/library/functions.html#bool

(continued from previous page)

```
MESSAGE_CATALOG_NAME = 'myextension' # name of *.pot, *.po and *.mo files
_ = get_translation(MESSAGE_CATALOG_NAME)
text = _('Hello Sphinx!')

def setup(app):
    package_dir = os.path.abspath(os.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.

Added in version 1.8.

```
sphinx.locale.\_(message: str^{1095}) \rightarrow str^{1096}
```

Translation function for messages on documentation (menu, labels, themes and so on). This function follows *language* setting.

```
sphinx.locale.\__(message: str^{1097}) \rightarrow str^{1098}
```

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

Added 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.:

3. Set up your extension to be aware of its dedicated translations:

```
1091 https://docs.python.org/3/library/stdtypes.html#str
1092 https://docs.python.org/3/library/stdtypes.html#str
1093 https://docs.python.org/3/library/stdtypes.html#str
1094 https://docs.python.org/3/library/stdtypes.html#str
1095 https://docs.python.org/3/library/stdtypes.html#str
1096 https://docs.python.org/3/library/stdtypes.html#str
1097 https://docs.python.org/3/library/stdtypes.html#str
1098 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 1099:

```
$ 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 Babel¹¹⁰⁰:

- 6. Translate message catalogs for each language manually
- 7. Compile message catalogs into *.mo files, for example via Babel¹¹⁰¹:

```
$ 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:

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 Babel¹¹⁰²:

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.

1 Note

1102 https://babel.pocoo.org/

The subclasses of them might not work with bare docutils because they are strongly coupled with Sphinx.

```
1099 https://babel.pocoo.org/
1100 https://babel.pocoo.org/
1101 https://babel.pocoo.org/
```

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: Sphinx

Reference to the Sphinx object.

property config: Config

Reference to the *Config* object.

property env: BuildEnvironment

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 resolve references, convert images, do special transformation for each output formats and so on. This class helps to implement these post transforms.

apply(** $kwargs: Any^{1103}$) \rightarrow None¹¹⁰⁴

Override to apply the transform to the document tree.

 $is_supported() \rightarrow bool^{1105}$

Check this transform working for current builder.

run (** $kwargs: Any^{1106}$) \rightarrow None¹¹⁰⁷

Main method of post transforms.

Subclasses should override this method instead of apply().

A base class for Sphinx directives.

This class provides helper methods for Sphinx directives.

Added in version 1.8.

1 Note

The subclasses of this class might not work with docutils. This class is strongly coupled with Sphinx.

 $get_location() \rightarrow str^{1108}$

Get current location info for logging.

Added in version 4.2.

get_source_info() \rightarrow tuple¹¹⁰⁹[str¹¹¹⁰, int¹¹¹¹]

Get source and line number.

Added in version 3.0.

¹¹⁰³ https://docs.python.org/3/library/typing.html#typing.Any

¹¹⁰⁴ https://docs.python.org/3/library/constants.html#None

¹¹⁰⁵ https://docs.python.org/3/library/functions.html#bool

¹¹⁰⁶ https://docs.python.org/3/library/typing.html#typing.Any

¹¹⁰⁷ https://docs.python.org/3/library/constants.html#None

```
parse\_content\_to\_nodes(allow\_section\_headings: bool^{1112} = False) \rightarrow list^{1113}[Node]
```

Parse the directive's content into nodes.

Parameters

allow_section_headings — Are titles (sections) allowed in the directive's content? Note that this option bypasses Docutils' usual checks on doctree structure, and misuse of this option can lead to an incoherent doctree. In Docutils, section nodes should only be children of Structural nodes, which includes document, section, and sidebar nodes.

Added in version 7.4.

```
parse_inline(text: str^{1114}, *, lineno: int^{1115} = -I) \rightarrow tuple^{1116}[list^{1117}[Node], list^{1118}[system_message]]
Parse text as inline elements.
```

Parameters

- **text** The text to parse, which should be a single line or paragraph. This cannot contain any structural elements (headings, transitions, directives, etc).
- **lineno** The line number where the interpreted text begins.

Returns

A list of nodes (text and inline elements) and a list of system_messages.

Added in version 7.4.

```
parse_text_to_nodes(text: str^{1119} = ", /, *, offset: int^{1120} = -1, allow_section_headings: bool^{1121} = False) \rightarrow list^{1122}[Node]
```

Parse text into nodes.

Parameters

- text Text, in string form. StringList is also accepted.
- allow_section_headings Are titles (sections) allowed in *text*? Note that this option bypasses Docutils' usual checks on doctree structure, and misuse of this option can lead to an incoherent doctree. In Docutils, section nodes should only be children of Structural nodes, which includes document, section, and sidebar nodes.
- **offset** The offset of the content.

Added in version 7.4.

```
set\_source\_info(node: Node) \rightarrow None^{1123}
```

Set source and line number to the node.

Added in version 2.1.

property config: Config

Reference to the *Config* object.

Added in version 1.8.

property env: BuildEnvironment

Reference to the BuildEnvironment object.

Added in version 1.8.

class sphinx.util.docutils.SphinxRole

A base class for Sphinx roles.

This class provides helper methods for Sphinx roles.

Added in version 2.0.

1 Note

The subclasses of this class might not work with docutils. This class is strongly coupled with Sphinx.

```
get_location() \rightarrow str^{1124}
```

Get current location info for logging.

Added in version 4.2.

property config: Config

Reference to the *Config* object.

Added in version 2.0.

content: Sequence¹¹²⁵[str¹¹²⁶]

A list of strings, the directive content for customisation (from the "role" directive).

property env: BuildEnvironment

Reference to the BuildEnvironment object.

Added in version 2.0.

inliner: Inliner

The docutils.parsers.rst.states.Inliner object.

lineno: int¹¹²⁷

The line number where the interpreted text begins.

name: str¹¹²⁸

The role name actually used in the document.

```
options: dict^{1129}[str^{1130}, Any^{1131}]
```

A dictionary of directive options for customisation (from the "role" directive).

rawtext: str¹¹³²

A string containing the entire interpreted text input.

¹¹⁰⁸ https://docs.python.org/3/library/stdtypes.html#str

¹¹⁰⁹ https://docs.python.org/3/library/stdtypes.html#tuple

¹¹¹⁰ https://docs.python.org/3/library/stdtypes.html#str

¹¹¹¹ https://docs.python.org/3/library/functions.html#int

 $^{^{1112}\} https://docs.python.org/3/library/functions.html\#bool$

¹¹¹³ https://docs.python.org/3/library/stdtypes.html#list

¹¹¹⁴ https://docs.python.org/3/library/stdtypes.html#str

¹¹¹⁵ https://docs.python.org/3/library/functions.html#int

¹¹¹⁶ https://docs.python.org/3/library/stdtypes.html#tuple

¹¹¹⁷ https://docs.python.org/3/library/stdtypes.html#list

¹¹¹⁸ https://docs.python.org/3/library/stdtypes.html#list

¹¹¹⁹ https://docs.python.org/3/library/stdtypes.html#str

¹¹²⁰ https://docs.python.org/3/library/functions.html#int

¹¹²¹ https://docs.python.org/3/library/functions.html#bool

¹¹²² https://docs.python.org/3/library/stdtypes.html#list

¹¹²³ https://docs.python.org/3/library/constants.html#None

```
text: str^{1133}
```

The interpreted text content.

class sphinx.util.docutils.ReferenceRole

A base class for reference roles.

The reference roles can accept 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.

Added in version 2.0.

disabled: bool 1134

A boolean indicates the reference is disabled.

has_explicit_title: bool 1135

A boolean indicates the role has explicit title or not.

target: str¹¹³⁶

The link target for the interpreted text.

title: str¹¹³⁷

The link title for the interpreted text.

class sphinx.transforms.post_transforms.images.ImageConverter(document, startnode=None)

A base class for image converters.

An image converter is kind of Docutils transform module. It is used to convert image files which are not supported by a builder to the 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, using image converters allows to embed these unsupported images into the document. One of the 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^{1138}, to: str^{1139}) \rightarrow bool<sup>1140</sup>
```

Convert an image file to the expected format.

_from is a path of the source image file, and _to is a path of the destination file.

¹¹²⁴ https://docs.python.org/3/library/stdtypes.html#str

https://docs.python.org/3/library/collections.abc.html#collections.abc.Sequence

https://docs.python.org/3/library/stdtypes.html#str

¹¹²⁷ https://docs.python.org/3/library/functions.html#int

 $^{^{1128}\} https://docs.python.org/3/library/stdtypes.html#str$

¹¹²⁹ https://docs.python.org/3/library/stdtypes.html#dict

https://docs.python.org/3/library/stdtypes.html#str

https://docs.python.org/3/library/typing.html#typing.Any

¹¹³² https://docs.python.org/3/library/stdtypes.html#str

¹¹³³ https://docs.python.org/3/library/stdtypes.html#str

¹¹³⁴ https://docs.python.org/3/library/functions.html#bool

https://docs.python.org/3/library/functions.html#bool

https://docs.python.org/3/library/stdtypes.html#str

¹¹³⁷ https://docs.python.org/3/library/stdtypes.html#str

```
is_available() \rightarrow bool^{1141}
```

Return the image converter is available or not.

```
available: bool^{1142} | None<sup>1143</sup> = 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.

```
conversion rules: list^{1144}[tuple^{1145}[str^{1146}. str^{1147}]] = []
```

A conversion rules the image converter supports. It is represented as a list of pair of source image format (mimetype) and destination one:

```
conversion_rules = [
    ('image/svg+xml', 'image/png'),
    ('image/gif', 'image/png'),
    ('application/pdf', 'image/png'),
]
```

default_priority = 200

Numerical priority of this transform, 0 through 999 (override).

Utility components

```
class sphinx.events.EventManager(app: Sphinx)
       Event manager for Sphinx.
       add(name: str^{1148}) \rightarrow None<sup>1149</sup>
               Register a custom Sphinx event.
       connect(name: Literal['config-inited'], callback: Callable[[Sphinx, Config], None^{1150}], priority: int^{1151}) \rightarrow
       \textbf{connect}(\textit{name: Literal['builder-inited'], callback: Callable[[Sphinx], None^{1153}], \textit{priority: int}^{1154})} \rightarrow \text{int}^{1155}
       connect(name: Literal['env-get-outdated'], callback: Callable[[Sphinx, BuildEnvironment, Set[str<sup>1156</sup>],
                     Set[str<sup>1157</sup>], Set[str<sup>1158</sup>]], Sequence[str<sup>1159</sup>]], priority: int^{1160}) \rightarrow int^{1161}
       connect(name: Literal['env-before-read-docs'], callback: Callable[[Sphinx, BuildEnvironment,
                     list^{1162}[str^{1163}]], None^{1164}], priority: int^{1165}) \rightarrow int^{1166}
       connect(name: Literal['env-purge-doc'], callback: Callable[[Sphinx, BuildEnvironment, str<sup>1167</sup>], None<sup>1168</sup>],
                     priority: int^{1169}) \rightarrow int^{1170}
       connect(name: Literal['source-read'], callback: Callable[[Sphinx, str<sup>1171</sup>, list<sup>1172</sup>[str<sup>1173</sup>]], None<sup>1174</sup>],
                     priority: int^{1175}) \rightarrow int<sup>1176</sup>
       connect(name: Literal['include-read'], callback: Callable[[Sphinx, Path, str<sup>1177</sup>, list<sup>1178</sup>[str<sup>1179</sup>]], None<sup>1180</sup>],
                     priority: int^{1181}) \rightarrow int<sup>1182</sup>
       connect(name: Literal['doctree-read'], callback: Callable[[Sphinx, nodes.document], None<sup>1183</sup>], priority:
                     int^{1184}) \to int^{1185}
```

¹¹³⁸ https://docs.python.org/3/library/stdtypes.html#str

¹¹³⁹ https://docs.python.org/3/library/stdtypes.html#str

¹¹⁴⁰ https://docs.python.org/3/library/functions.html#bool

¹¹⁴¹ https://docs.python.org/3/library/functions.html#bool

¹¹⁴² https://docs.python.org/3/library/functions.html#bool

¹¹⁴³ https://docs.python.org/3/library/constants.html#None

¹¹⁴⁴ https://docs.python.org/3/library/stdtypes.html#list

¹¹⁴⁵ https://docs.python.org/3/library/stdtypes.html#tuple

¹¹⁴⁶ https://docs.python.org/3/library/stdtypes.html#str

¹¹⁴⁷ https://docs.python.org/3/library/stdtypes.html#str

- **connect**(*name: Literal*['env-merge-info'], callback: Callable[[Sphinx, BuildEnvironment, list¹¹⁸⁶[str¹¹⁸⁷], BuildEnvironment], None¹¹⁸⁸], priority: int¹¹⁸⁹) → int¹¹⁹⁰
- **connect**(name: Literal['env-updated'], callback: Callable[[Sphinx, BuildEnvironment], str^{1191}], priority: int^{1192}) \rightarrow int¹¹⁹³
- **connect**(*name*: Literal['env-get-updated'], callback: Callable[[Sphinx, BuildEnvironment], Iterable[str¹¹⁹⁴]], priority: int^{1195}) $\rightarrow int^{1196}$
- **connect**(name: Literal['env-check-consistency'], callback: Callable[[Sphinx, BuildEnvironment], None¹¹⁹⁷], priority: int^{1198}) $\rightarrow int^{1199}$
- $\textbf{connect}(\textit{name: Literal['write-started'], callback: Callable[[Sphinx, Builder], None^{1200}], priority: int^{1201})} \rightarrow \inf^{1202} (\textit{name: Literal['write-started'], callback: Callable[[Sphinx, Builder], None^{1200}], priority: int^{1201})} \rightarrow \inf^{1202} (\textit{name: Literal['write-started'], callback: Callable[[Sphinx, Builder], None^{1200}], priority: int^{1201})}$
- **connect**(name: Literal['doctree-resolved'], callback: Callable[[Sphinx, nodes.document, str^{1203}], None¹²⁰⁴], priority: int^{1205}) $\rightarrow int^{1206}$
- **connect**(*name: Literal*['missing-reference'], callback: Callable[[Sphinx, BuildEnvironment, addnodes.pending_xref, nodes.TextElement], nodes.reference | None 1207], priority: int 1208) \rightarrow int 1209
- **connect**(name: Literal['warn-missing-reference'], callback: Callable[[Sphinx, Domain, addnodes.pending_xref], bool¹²¹⁰ | None¹²¹¹], priority: int^{1212}) \rightarrow int¹²¹³
- **connect**(name: Literal['build-finished'], callback: Callable[[Sphinx, Exception¹²¹⁴ | None¹²¹⁵], None¹²¹⁶], priority: int^{1217}) $\rightarrow int^{1218}$
- **connect** (name: Literal['html-collect-pages'], callback: Callable[[Sphinx], Iterable[tuple¹²¹⁹[str¹²²⁰, $dict^{1221}[str^{1222}, Any], str^{1223}]]]$, priority: int^{1224}) $\rightarrow int^{1225}$
- **connect**(name: Literal['html-page-context'], callback: Callable[[Sphinx, str^{1226} , str^{1227} , $dict^{1228}[str^{1229}$, Any], nodes.document], $str^{1230} \mid None^{1231}$], priority: int^{1232}) $\rightarrow int^{1233}$
- **connect**(name: Literal['linkcheck-process-uri'], callback: Callable[[Sphinx, str^{1234}], str^{1235} | None¹²³⁶], priority: int^{1237}) \rightarrow int¹²³⁸
- **connect**(*name*: Literal['object-description-transform'], callback: Callable[[Sphinx, str^{1239} , str^{1240} , addnodes.desc_content], None¹²⁴¹], priority: int^{1242}) $\rightarrow int^{1243}$
- **connect**(name: Literal['autodoc-process-docstring'], callback: _AutodocProcessDocstringListener, priority: int^{1244}) \rightarrow int¹²⁴⁵
- **connect** (name: Literal['autodoc-before-process-signature'], callback: Callable[[Sphinx, Any, bool 1246], None 1247], priority: int 1248) \rightarrow int 1249
- **connect**(name: Literal['autodoc-process-signature'], callback: Callable[[Sphinx, Literal['module', 'class', 'exception', 'function', 'method', 'attribute'], str^{1250} , Any, $dict^{1251}[str^{1252}, bool^{1253}]$, $str^{1254} \mid None^{1255}$, $str^{1256} \mid None^{1257}$], $tuple^{1258}[str^{1259} \mid None^{1260}, str^{1261} \mid None^{1262}] \mid None^{1263}$], $priority: int^{1264}$) $\rightarrow int^{1265}$
- **connect**(name: Literal['autodoc-process-bases'], callback: Callable[[Sphinx, str^{1266} , Any, $dict^{1267}[str^{1268}$, $bool^{1269}$], $list^{1270}[str^{1271}]$], $None^{1272}$], $priority: int^{1273}$) $\rightarrow int^{1274}$
- **connect**(name: Literal['autodoc-skip-member'], callback: Callable[[Sphinx, Literal['module', 'class', 'exception', 'function', 'method', 'attribute'], str^{1275} , Any, $bool^{1276}$, $dict^{1277}[str^{1278}, bool^{1279}]$], $bool^{1280}$], $priority: int^{1281}$) $\rightarrow int^{1282}$
- **connect**(name: Literal['todo-defined'], callback: Callable[[Sphinx, todo_node], None 1283], priority: int 1284) \rightarrow int 1285
- **connect**(name: Literal['viewcode-find-source'], callback: Callable[[Sphinx, str^{1286}], tuple ¹²⁸⁷[str^{1288} , $dict^{1289}[str^{1290}, tuple^{1291}[Literal['class', 'def', 'other'], int^{1292}, int^{1293}]]]], priority: int^{1294}) \rightarrow int^{1295}$
- **connect**(name: Literal['viewcode-follow-imported'], callback: Callable[[Sphinx, str^{1296} , str^{1297}], str^{1298} | None¹²⁹⁹], priority: int^{1300}) \rightarrow int¹³⁰¹
- **connect**(*name*: str^{1302} , callback: Callable[..., Any], priority: int^{1303}) \rightarrow int^{1304} Connect a handler to specific event.

```
disconnect(listener\_id: int^{1305}) \rightarrow None^{1306}
```

Disconnect a handler.

emit(name: str^{1307} , *args: Any, allowed_exceptions: $tuple^{1308}[type^{1309}[Exception^{1310}], ...] = ()) \rightarrow list^{1311}$ Emit a Sphinx event.

emit_firstresult(name:
$$str^{1312}$$
, *args: Any, allowed_exceptions: $tuple^{1313}[type^{1314}[Exception^{1315}], ...] = ()) \rightarrow Any$

Emit a Sphinx event and returns first result.

This returns the result of the first handler that doesn't return None.

- 1148 https://docs.python.org/3/library/stdtypes.html#str 1149 https://docs.python.org/3/library/constants.html#None 1150 https://docs.python.org/3/library/constants.html#None 1151 https://docs.python.org/3/library/functions.html#int 1152 https://docs.python.org/3/library/functions.html#int 1153 https://docs.python.org/3/library/constants.html#None 1154 https://docs.python.org/3/library/functions.html#int 1155 https://docs.python.org/3/library/functions.html#int 1156 https://docs.python.org/3/library/stdtypes.html#str 1157 https://docs.python.org/3/library/stdtypes.html#str 1158 https://docs.python.org/3/library/stdtypes.html#str 1159 https://docs.python.org/3/library/stdtypes.html#str 1160 https://docs.python.org/3/library/functions.html#int 1161 https://docs.python.org/3/library/functions.html#int 1162 https://docs.python.org/3/library/stdtypes.html#list 1163 https://docs.python.org/3/library/stdtypes.html#str 1164 https://docs.python.org/3/library/constants.html#None 1165 https://docs.python.org/3/library/functions.html#int 1166 https://docs.python.org/3/library/functions.html#int 1167 https://docs.python.org/3/library/stdtypes.html#str 1168 https://docs.python.org/3/library/constants.html#None 1169 https://docs.python.org/3/library/functions.html#int 1170 https://docs.python.org/3/library/functions.html#int 1171 https://docs.python.org/3/library/stdtypes.html#str 1172 https://docs.python.org/3/library/stdtypes.html#list 1173 https://docs.python.org/3/library/stdtypes.html#str 1174 https://docs.python.org/3/library/constants.html#None 1175 https://docs.python.org/3/library/functions.html#int 1176 https://docs.python.org/3/library/functions.html#int 1177 https://docs.python.org/3/library/stdtypes.html#str 1178 https://docs.python.org/3/library/stdtypes.html#list 1179 https://docs.python.org/3/library/stdtypes.html#str 1180 https://docs.python.org/3/library/constants.html#None 1181 https://docs.python.org/3/library/functions.html#int 1182 https://docs.python.org/3/library/functions.html#int 1183 https://docs.python.org/3/library/constants.html#None 1184 https://docs.python.org/3/library/functions.html#int 1185 https://docs.python.org/3/library/functions.html#int 1186 https://docs.python.org/3/library/stdtypes.html#list 1187 https://docs.python.org/3/library/stdtypes.html#str 1188 https://docs.python.org/3/library/constants.html#None 1189 https://docs.python.org/3/library/functions.html#int 1190 https://docs.python.org/3/library/functions.html#int 1191 https://docs.python.org/3/library/stdtypes.html#str 1192 https://docs.python.org/3/library/functions.html#int 1193 https://docs.python.org/3/library/functions.html#int 1194 https://docs.python.org/3/library/stdtypes.html#str 1195 https://docs.python.org/3/library/functions.html#int 1196 https://docs.python.org/3/library/functions.html#int 1197 https://docs.python.org/3/library/constants.html#None 1198 https://docs.python.org/3/library/functions.html#int 1199 https://docs.python.org/3/library/functions.html#int 1200 https://docs.python.org/3/library/constants.html#None 1201 https://docs.python.org/3/library/functions.html#int 1202 https://docs.python.org/3/library/functions.html#int 1203 https://docs.python.org/3/library/stdtypes.html#str 1204 https://docs.python.org/3/library/constants.html#None $^{1205}\ https://docs.python.org/3/library/functions.html\#int$ $^{1206}\ https://docs.python.org/3/library/functions.html\#int$ 1207 https://docs.python.org/3/library/constants.html#None 1208 https://docs.python.org/3/library/functions.html#int 1209 https://docs.python.org/3/library/functions.html#int 1210 https://docs.python.org/3/library/functions.html#bool 1211 https://docs.python.org/3/library/constants.html#None 1212 https://docs.python.org/3/library/functions.html#int 1213 https://docs.python.org/3/library/functions.html#int 1214 https://docs.python.org/3/library/exceptions.html#Exception 1215 https://docs.python.org/3/library/constants.html#None
- 1216 https://docs.pythou.org/3/library/constants.html#None
 621 https://docs.python.org/3/library/functions.html#int
 1218 https://docs.python.org/3/library/functions.html#int

1219 https://docs.python.org/3/library/stdtypes.html#tuple
1220 https://docs.python.org/3/library/stdtypes.html#str
1221 https://docs.python.org/3/library/stdtypes.html#dict

Utility functions

```
sphinx.util.parsing.nested_parse_to_nodes(state: RSTState, text: str^{1316} \mid StringList, *, source: str^{1317} =  '<generated text>', offset: int^{1318} = 0, allow_section_headings: bool^{1319} = True, keep_title_context: bool^{1320} = False) \rightarrow list^{1321}[Node]
```

Parse *text* into nodes.

Parameters

- **state** The state machine state. Must be a subclass of RSTState.
- text Text, in string form. StringList is also accepted.
- **source** The text's source, used when creating a new StringList.
- **offset** The offset of the content.
- allow_section_headings Are titles (sections) allowed in *text*? Note that this option bypasses Docutils' usual checks on doctree structure, and misuse of this option can lead to an incoherent doctree. In Docutils, section nodes should only be children of Structural nodes, which includes document, section, and sidebar nodes.
- **keep_title_context** If this is False (the default), then *content* is parsed as if it were an independent document, meaning that title decorations (e.g. underlines) do not need to match the surrounding document. This is useful when the parsed content comes from a completely different context, such as docstrings. If this is True, then title underlines must match those in the surrounding document, otherwise the behaviour is undefined.

Added in version 7.4.

Utility types

class sphinx.util.typing.ExtensionMetadata

The metadata returned by an extension's setup() function.

See Extension metadata.

```
env_version: int<sup>1322</sup>
```

An integer that identifies the version of env data added by the extension.

```
parallel_read_safe: bool<sup>1323</sup>
```

Indicate whether parallel reading of source files is supported by the extension.

```
parallel write safe: bool 1324
```

Indicate whether parallel writing of output files is supported by the extension (default: True).

```
version: str^{1325}
```

The extension version (default: 'unknown version').

¹³¹⁶ https://docs.python.org/3/library/stdtypes.html#str

¹³¹⁷ https://docs.python.org/3/library/stdtypes.html#str

¹³¹⁸ https://docs.python.org/3/library/functions.html#int

¹³¹⁹ https://docs.python.org/3/library/functions.html#bool

¹³²⁰ https://docs.python.org/3/library/functions.html#bool

¹³²¹ https://docs.python.org/3/library/stdtypes.html#list

¹³²² https://docs.python.org/3/library/functions.html#int

¹³²³ https://docs.python.org/3/library/functions.html#bool

¹³²⁴ https://docs.python.org/3/library/functions.html#bool

¹³²⁵ https://docs.python.org/3/library/stdtypes.html#str

Testing API

Added in version 1.6.

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 pytest 1326.

pytest configuration

To use pytest fixtures that are provided by sphinx.testing, add the 'sphinx.testing.fixtures' plugin to your test modules or conftest.py files as follows:

```
pytest_plugins = ('sphinx.testing.fixtures',)
```

Usage

If you want to know more detailed usage, please refer to tests/conftest.py and other test_*.py files under the tests/directory.

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	Depre- cated	Re- moved	Alternatives
sphinx.util.FilenameUniqDict	8.1	10.0	N/A
sphinx.util.DownloadFiles	8.1	10.0	N/A
<pre>sphinx.util.import_object</pre>	8.1	10.0	<pre>importlib.import_module</pre>
<pre>sphinx.ext.intersphinx. normalize_intersphinx_mapping</pre>	8.0	10.0	<pre>sphinx.ext.intersphinx. validate_intersphinx_mapping</pre>
<pre>sphinx.testing.util.strip_escseq</pre>	7.3	9.0	<pre>sphinx.util.console. strip_colors</pre>
Old-style Makefiles in sphinx-quickstart and the -M, -m,no-use-make-mode, anduse-make-mode options	7.3	9.0	Vendoring the old style Makefile templates 1327
<pre>sphinx.ext.autodoc. preserve_defaults. get_function_def()</pre>	7.2	9.0	N/A (replacement is private)
<pre>sphinx.builders.html. StandaloneHTMLBuilder.css_files</pre>	7.2	9.0	N/A
<pre>sphinx.builders.html. StandaloneHTMLBuilder. script_files</pre>	7.2	9.0	N/A
sphinx.builders.html.Stylesheet	7.2	9.0	<pre>sphinx.application.Sphinx. add_css_file()</pre>

¹³²⁶ https://docs.pytest.org/en/latest/

Table 4 – continued from previous page

Table 4	continued	non previo	us page
Target	Depre- cated	Re- moved	Alternatives
sphinx.builders.html.JavaScript	7.2	9.0	<pre>sphinx.application.Sphinx. add_js_file()</pre>
<pre>sphinx.util.split_into</pre>	7.2	9.0	N/A
<pre>sphinx.util.split_index_msg</pre>	7.2	9.0	<pre>sphinx.util.index_entries. split_index_msg</pre>
sphinx.testing.path	7.2	9.0	os.path or pathlib
sphinx.util.md5	7.2	9.0	hashlib.md5
sphinx.util.sha1	7.2	9.0	hashlib.sha1
sphinx.util.osutil.cd	6.2	8.0	contextlib.chdir
sphinx.util.save_traceback	6.1	8.0	<pre>sphinx.util.exceptions. save_traceback</pre>
<pre>sphinx.util. format_exception_cut_frames</pre>	6.1	8.0	<pre>sphinx.util.exceptions. format_exception_cut_frames</pre>
sphinx.util.epoch_to_rfc1123	6.1	8.0	<pre>sphinx.util.http_date. epoch_to_rfc1123</pre>
sphinx.util.rfc1123_to_epoch	6.1	8.0	<pre>sphinx.util.http_date. rfc1123_to_epoch</pre>
sphinx.util.status_iterator	6.1	8.0	<pre>sphinx.util.display. status_iterator</pre>
sphinx.util.display_chunk	6.1	8.0	sphinx.util.display. display_chunk
sphinx.util.SkipProgressMessage	6.1	8.0	sphinx.util.display. SkipProgressMessage
sphinx.util.progress_message	6.1	8.0	sphinx.util.display. progress_message
sphinx.util.typing.stringify	6.1	8.0	sphinx.util.typing. stringify_annotation
HTML 4 support	5.2	7.0	N/A
sphinx.util.path_stabilize	5.1	7.0	sphinx.util.osutil.
			path_stabilize
<pre>sphinx.util.get_matching_files</pre>	5.1	7.0	<pre>sphinx.util.matching. get_matching_files</pre>
sphinx.ext.napoleon.iterators	5.1	7.0	pockets.iterators
sphinx.util.stemmer	5.1	7.0	snowballstemmer
sphinx.util.jsdump	5.0	7.0	The standard library json module.
The Setuptools integration (setup.py build_sphinx)	5.0	7.0	N/A
The locale argument of sphinx.util. i18n:babel_format_date()	5.0	7.0	N/A
The language argument of sphinx.util.i18n:format_date()	5.0	7.0	N/A
sphinx.builders.html.html5_ready	5.0	7.0	N/A
sphinx.io.read_doc()	5.0	7.0	<pre>sphinx.builders.Builder. read_doc()</pre>
<pre>sphinx.util.docutilsversion_info</pre>	5.0	7.0	docutilsversion_info
<pre>sphinx.util.docutils. is_html5_writer_available()</pre>	5.0	7.0	N/A

Table 4 – continued from previous page

Target	Depre-	Re-	Alternatives
	cated	moved	
<pre>sphinx.writers.latex.LaTeXWriter.</pre>	5.0	7.0	N/A
docclasses			
sphinx.ext.napoleon.docstring.	4.5	6.0	N/A
GoogleDocstringqualify_name()	4.3	6.0	N/A
<pre>sphinx.ext.autodoc. AttributeDocumenter.</pre>	4.3	0.0	IV/A
_datadescriptor			
sphinx.writers.html.	4.3	6.0	sphinx.writers.html.
HTMLTranslator.			HTMLTranslator.
_fieldlist_row_index			_fieldlist_row_indices
sphinx.writers.html.	4.3	6.0	sphinx.writers.html.
<pre>HTMLTranslatortable_row_index</pre>			HTMLTranslator.
		- 0	_table_row_indices
sphinx.writers.html5.	4.3	6.0	sphinx.writers.html5.
HTML5Translator.			HTML5Translator.
_fieldlist_row_index sphinx.writers.html5.	4.3	6.0	_fieldlist_row_indices sphinx.writers.html5.
HTML5Translatortable_row_index	4.3	0.0	HTML5Translator.
IIIIEJITatistatortable_row_ritaex			_table_row_indices
The optional argument app for sphinx.	4.1	6.0	The required argument
environment.BuildEnvironment			
sphinx.application.Sphinx.	4.1	6.0	sphinx.registry.
html_theme			SphinxComponentRegistry.
			html_themes
sphinx.ext.autosummaryapp	4.1	6.0	N/A
sphinx.util.docstrings.	4.1	6.0	sphinx.util.docstrings.
extract_metadata()	4.0	6.0	<pre>separate_metadata() favicon_url</pre>
favicon variable in HTML templates logo variable in HTML templates	4.0	6.0	logo_url
sphinx.directives.patches.	4.0	6.0	docutils.parsers.rst.
ListTable			directives.tables.ListSVTable
sphinx.directives.patches.	4.0	6.0	docutils.parsers.rst.
RSTTable			directives.tables.RSTTable
sphinx.ext.autodoc.directive.	4.0	6.0	sphinx.ext.autodoc.directive.
DocumenterBridge.filename_set			DocumenterBridge.
			record_dependencies
sphinx.ext.autodoc.directive.	4.0	6.0	Logging API
DocumenterBridge.warn() sphinx.registry.	4.0	6.0	N/A
SphinxComponentRegistry.	4.0	0.0	IVA
<pre>get_source_input()</pre>			
sphinx.registry.	4.0	6.0	N/A
SphinxComponentRegistry.			
source_inputs			
sphinx.transforms.FigureAligner	4.0	6.0	N/A
sphinx.util.pycompat.	4.0	6.0	N/A
convert_with_2to3()			
<pre>sphinx.util.pycompat.execfile_()</pre>	4.0	6.0	N/A
sphinx.util.smartypants	4.0	6.0	docutils.utils.smartquotes

Table 4 – continued from previous page

Target	Depre- cated	Re- moved	Alternatives
			NYA
<pre>sphinx.util.typing. DirectiveOption</pre>	4.0	6.0	N/A
pending_xref node for viewcode extension	3.5	5.0	<pre>sphinx.ext.viewcode. viewcode_anchor</pre>
<pre>sphinx.builders.linkcheck. CheckExternalLinksBuilder. anchors_ignore</pre>	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
<pre>sphinx.builders.linkcheck. CheckExternalLinksBuilder. to_ignore</pre>	3.5	5.0	N/A
<pre>sphinx.builders.linkcheck. CheckExternalLinksBuilder.workers</pre>	3.5	5.0	N/A
sphinx.builders.linkcheck. CheckExternalLinksBuilder.wqueue	3.5	5.0	N/A
<pre>sphinx.builders.linkcheck. node_line_or_0()</pre>	3.5	5.0	<pre>sphinx.util.nodes. get_node_line()</pre>
<pre>sphinx.ext.autodoc. AttributeDocumenter. isinstanceattribute()</pre>	3.5	5.0	N/A
<pre>sphinx.ext.autodoc.importer. get_module_members()</pre>	3.5	5.0	<pre>sphinx.ext.autodoc. ModuleDocumenter. get_module_members()</pre>
<pre>sphinx.ext.autosummary.generatesimple_info()</pre>	3.5	5.0	Logging API
<pre>sphinx.ext.autosummary.generatesimple_warn()</pre>	3.5	5.0	Logging API
sphinx.writers.html. HTMLTranslator.permalink_text	3.5	5.0	html_permalinks_icon
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	<pre>sphinx.ext.autodoc.Documenter. get_doc()</pre>
<pre>sphinx.ext.autodoc.Documenter. get_object_members()</pre>	3.4	6.0	sphinx.ext.autodoc. ClassDocumenter. get_object_members()

Table 4 – continued from previous page

		Re-	Alternatives
Target	Depre- cated	moved	Alternatives
sphinx.ext.autodoc.	3.4	5.0	sphinx.ext.autodoc.
DataDeclarationDocumenter			DataDocumenter
sphinx.ext.autodoc.	3.4	5.0	sphinx.ext.autodoc.
GenericAliasDocumenter			DataDocumenter
sphinx.ext.autodoc.	3.4	5.0	sphinx.ext.autodoc.
InstanceAttributeDocumenter			AttributeDocumenter
sphinx.ext.autodoc.	3.4	5.0	sphinx.ext.autodoc.
SlotsAttributeDocumenter			AttributeDocumenter
sphinx.ext.autodoc.	3.4	5.0	sphinx.ext.autodoc.
TypeVarDocumenter			DataDocumenter
sphinx.ext.autodoc.directive.	3.5	5.0	sphinx.util.logging
DocumenterBridge.reporter			
sphinx.ext.autodoc.importer.	3.4	4.0	sphinx.util.inspect.
_getannotations()			<pre>getannotations()</pre>
sphinx.ext.autodoc.importer.	3.4	4.0	<pre>sphinx.util.inspect.getmro()</pre>
_getmro()			
sphinx.pycode.ModuleAnalyzer.	3.4	5.0	sphinx.pycode.ModuleAnalyzer.
parse()			analyze()
<pre>sphinx.util.osutil.movefile()</pre>	3.4	5.0	os.replace()
sphinx.util.requests.	3.4	5.0	N/A
is_ssl_error()			
sphinx.builders.latex.	3.3	5.0	N/A
LaTeXBuilder.usepackages			
sphinx.builders.latex.	3.3	5.0	N/A
LaTeXBuilder.			
usepackages_afger_hyperref			
sphinx.ext.autodoc.	3.3	5.0	sphinx.ext.autodoc.
SingledispatchFunctionDocumenter	2.2	= 0	FunctionDocumenter
sphinx.ext.autodoc.	3.3	5.0	sphinx.ext.autodoc.
SingledispatchMethodDocumenter	2.0	7.0	MethodDocumenter
sphinx.ext.autodoc.	3.2	5.0	N/A
members_set_option()	2.0	7 0	1
sphinx.ext.autodoc.	3.2	5.0	sphinx.ext.autodoc.
<pre>merge_special_members_option()</pre>	2.0	5 0	merge_members_option()
sphinx.writers.texinfo.	3.2	5.0	sphinx.writers.texinfo.
TexinfoWriter.desc	2.1	5 0	TexinfoWriter.descs
The first argument for	3.1	5.0	N/A
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	3.1	3.0	11/12
type as an argument			
The ignore argument of sphinx.ext.	3.1	5.0	N/A
autodoc.Documenter.get_doc()	3.1	3.0	10/1
The template_dir argument of	3.1	5.0	N/A
sphinx.ext.autosummary.generate.	J.1	5.0	1 1/1 1
AutosummaryRenderer			
The Coommittee of the Cooperation of the Cooperatio			continues on poyt nage

Table 4 – continued from previous page

Target	Depre-	Re-	Alternatives
raigot	cated	moved	7 illomativos
The module argument of	3.0	5.0	N/A
sphinx.ext.autosummary.generate.			
<pre>find_autosummary_in_docstring()</pre>			
The builder argument of	3.1	5.0	N/A
sphinx.ext.autosummary.generate.			
<pre>generate_autosummary_docs()</pre>			
The template_dir argument of	3.1	5.0	N/A
sphinx.ext.autosummary.generate.			
<pre>generate_autosummary_docs()</pre>			
sphinx.ext.autosummary.generate.	3.1	5.0	N/A
<pre>AutosummaryRenderer.exists()</pre>			
The ignore argument of sphinx.util.	3.1	5.0	N/A
<pre>docstring.prepare_docstring()</pre>			
<pre>sphinx.util.rpartition()</pre>	3.1	5.0	str.rpartition()
<pre>desc_signature['first']</pre>		3.0	N/A
sphinx.directives.DescDirective	3.0	5.0	sphinx.directives.
			ObjectDescription
sphinx.domains.std.	3.0	5.0	sphinx.domains.std.
<pre>StandardDomain.add_object()</pre>			<pre>StandardDomain.note_object()</pre>
sphinx.domains.python.	3.0	5.0	N/A
PyDecoratorMixin			
sphinx.ext.autodoc.	3.0	5.0	sphinx.registry.documenters
<pre>get_documenters()</pre>			
sphinx.ext.autosummary.	3.0	5.0	N/A
<pre>process_autosummary_toc()</pre>			
sphinx.parsers.Parser.app	3.0	5.0	N/A
<pre>sphinx.testing.path.Path.text()</pre>	3.0	5.0	<pre>sphinx.testing.path.Path. read_text()</pre>
<pre>sphinx.testing.path.Path.bytes()</pre>	3.0	5.0	<pre>sphinx.testing.path.Path. read_bytes()</pre>
<pre>sphinx.util.inspect.getargspec()</pre>	3.0	5.0	<pre>inspect.getargspec()</pre>
sphinx.writers.latex.LaTeXWriter.	3.0	5.0	LaTeX Themes
format_docclass()	0	- • •	
decode argument of	2.4	4.0	N/A
sphinx.pycode.ModuleAnalyzer()			
sphinx.directives.other.Index	2.4	4.0	<pre>sphinx.domains.index. IndexDirective</pre>
sphinx.environment.	2.4	4.0	documents.nameids
temp_data['gloss_entries']	۷.٦	7.0	documents mamerus
sphinx.environment.	2.4	4.0	sphinx.domains.index.
BuildEnvironment.indexentries	۷.٦	7.0	IndexDomain
sphinx.environment.collectors.	2.4	4.0	sphinx.domains.index.
indexentries.	2.7	4.0	IndexDomain
IndexEntriesCollector			THE CAP ORIGITI
sphinx.io.FiletypeNotFoundError	2.4	4.0	sphinx.errors.
opmination free penotional databases	۵,٦	1.0	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
			continues on next nage

Table 4 – continued from previous page

		i irom previo	
Target	Depre- cated	Re- moved	Alternatives
<pre>sphinx.io.get_filetype()</pre>	2.4	4.0	<pre>sphinx.util.get_filetype()</pre>
<pre>sphinx.pycode.ModuleAnalyzer. encoding</pre>	2.4	4.0	N/A
sphinx.roles.Index	2.4	4.0	sphinx.domains.index.IndexRole
<pre>sphinx.util.detect_encoding()</pre>	2.4	4.0	<pre>tokenize.detect_encoding()</pre>
<pre>sphinx.util.get_module_source()</pre>	2.4	4.0	N/A
sphinx.util.inspect.Signature	2.4	4.0	<pre>sphinx.util.inspect.signature and sphinx.util.inspect. stringify_signature()</pre>
<pre>sphinx.util.inspect. safe_getmembers()</pre>	2.4	4.0	<pre>inspect.getmembers()</pre>
sphinx.writers.latex.	2.4	4.0	N/A
LaTeXTranslator.settings.author			
<pre>sphinx.writers.latex. LaTeXTranslator.settings. contentsname</pre>	2.4	4.0	document['contentsname']
<pre>sphinx.writers.latex. LaTeXTranslator.settings.docclass</pre>	2.4	4.0	document['docclass']
<pre>sphinx.writers.latex. LaTeXTranslator.settings.docname</pre>	2.4	4.0	N/A
<pre>sphinx.writers.latex. LaTeXTranslator.settings.title</pre>	2.4	4.0	N/A
sphinx.writers.latex. ADDITIONAL_SETTINGS	2.4	4.0	<pre>sphinx.builders.latex. constants.ADDITIONAL_SETTINGS</pre>
<pre>sphinx.writers.latex. DEFAULT_SETTINGS</pre>	2.4	4.0	<pre>sphinx.builders.latex. constants.DEFAULT_SETTINGS</pre>
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	<pre>sphinx.builders.latex. constants. PDFLATEX_DEFAULT_FONTPKG</pre>
sphinx.writers.latex. XELATEX_DEFAULT_FONTPKG	2.4	4.0	<pre>sphinx.builders.latex. constants. XELATEX_DEFAULT_FONTPKG</pre>
<pre>sphinx.writers.latex. XELATEX_GREEK_DEFAULT_FONTPKG</pre>	2.4	4.0	<pre>sphinx.builders.latex. constants. XELATEX_GREEK_DEFAULT_FONTPKG</pre>
sphinx.builders.gettext.POHEADER	2.3	4.0	<pre>sphinx/templates/gettext/ message.pot_t (template file)</pre>
<pre>sphinx.io.SphinxStandaloneReader. app</pre>	2.3	4.0	<pre>sphinx.io. SphinxStandaloneReader.setup()</pre>
<pre>sphinx.io.SphinxStandaloneReader. env</pre>	2.3	4.0	sphinx.io. SphinxStandaloneReader.setup()
<pre>sphinx.util.texescape. tex_escape_map</pre>	2.3	4.0	sphinx.util.texescape.escape()
<pre>sphinx.util.texescape. tex_hl_escape_map_new</pre>	2.3	4.0	<pre>sphinx.util.texescape. hlescape()</pre>

Table 4 – continued from previous page

		from previo	, ,
Target	Depre- cated	Re- moved	Alternatives
sphinx.writers.latex.	2.3	4.0	N/A
LaTeXTranslator.no_contractions			
sphinx.domains.math.MathDomain.	2.2	4.0	sphinx.domains.math.MathDomain.
add_equation()			<pre>note_equation()</pre>
sphinx.domains.math.MathDomain.	2.2	4.0	sphinx.domains.math.MathDomain.
<pre>get_next_equation_number()</pre>			<pre>note_equation()</pre>
The info and warn arguments of	2.2	4.0	logging.info() and
sphinx.ext.autosummary.generate.			logging.warning()
<pre>generate_autosummary_docs()</pre>			
sphinx.ext.autosummary.generate.	2.2	4.0	logging.info()
_simple_info()			
sphinx.ext.autosummary.generate.	2.2	4.0	logging.warning()
_simple_warn()			
<pre>sphinx.ext.todo.merge_info()</pre>	2.2	4.0	sphinx.ext.todo.TodoDomain
sphinx.ext.todo.	2.2	4.0	sphinx.ext.todo.TodoDomain
<pre>process_todo_nodes()</pre>			
<pre>sphinx.ext.todo.process_todos()</pre>	2.2	4.0	sphinx.ext.todo.TodoDomain
<pre>sphinx.ext.todo.purge_todos()</pre>	2.2	4.0	sphinx.ext.todo.TodoDomain
sphinx.builders.latex.	2.1	4.0	N/A
<pre>LaTeXBuilder.apply_transforms()</pre>			
sphinx.buildersepub_base.	2.1	4.0	html.escape()
<pre>EpubBuilder.esc()</pre>			
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
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.domains.changeset.
			VersionChange
			continues on post page

Table 4 – continued from previous page

		Po	. •
Target	Depre- cated	Re- moved	Alternatives
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
<pre>sphinx.domains.python. PyModulelevel</pre>	2.1	4.0	sphinx.domains.python. PyFunction, sphinx.domains.python.PyObject and sphinx.domains.python. PyVariable
<pre>sphinx.domains.std. StandardDomainresolve_citation_xref()</pre>	2.1	4.0	<pre>sphinx.domains.citation. CitationDomain.resolve_xref()</pre>
<pre>sphinx.domains.std. StandardDomain.note_citations()</pre>	2.1	4.0	<pre>sphinx.domains.citation. CitationDomain.note_citation()</pre>
<pre>sphinx.domains.std. StandardDomain. note_citation_refs()</pre>	2.1	4.0	<pre>sphinx.domains.citation. CitationDomain. note_citation_reference()</pre>
<pre>sphinx.domains.std. StandardDomain.note_labels()</pre>	2.1	4.0	<pre>sphinx.domains.std. StandardDomain.process_doc()</pre>
<pre>sphinx.domains.js.JSObject. display_prefix</pre>		4.3	<pre>sphinx.domains.js.JSObject. get_display_prefix()</pre>
<pre>sphinx.environment.NoUri sphinx.ext.apidoc. format_directive()</pre>	2.1 2.1	3.0 4.0	sphinx.errors.NoUri N/A
<pre>sphinx.ext.apidoc. format_heading()</pre>	2.1	4.0	N/A
<pre>sphinx.ext.apidoc.makename()</pre>	2.1	4.0	<pre>sphinx.ext.apidoc.module_join()</pre>
<pre>sphinx.ext.autodoc.importer. MockFinder</pre>	2.1	4.0	<pre>sphinx.ext.autodoc.mock. MockFinder</pre>
<pre>sphinx.ext.autodoc.importer. MockLoader</pre>	2.1	4.0	sphinx.ext.autodoc.mock. MockLoader
<pre>sphinx.ext.autodoc.importer. mock()</pre>	2.1	4.0	<pre>sphinx.ext.autodoc.mock.mock()</pre>
<pre>sphinx.ext.autosummary. autolink_role()</pre>	2.1	4.0	sphinx.ext.autosummary.AutoLink
sphinx.ext.imgmath.DOC_BODY	2.1	4.0	N/A
<pre>sphinx.ext.imgmath. DOC_BODY_PREVIEW</pre>	2.1	4.0	N/A
sphinx.ext.imgmath.DOC_HEAD	2.1	4.0	N/A
<pre>sphinx.transforms. CitationReferences</pre>	2.1	4.0	sphinx.domains.citation. CitationReferenceTransform
sphinx.transforms. SmartQuotesSkipper	2.1	4.0	sphinx.domains.citation. CitationDefinitionTransform

Table 4 – continued from previous page

Table 4 – continued from previous page				
Target	Depre- cated	Re- moved	Alternatives	
<pre>sphinx.util.docfields. DocFieldTransformer. preprocess_fieldtypes()</pre>	2.1	4.0	<pre>sphinx.directives. ObjectDescription. get_field_type_map()</pre>	
<pre>sphinx.util.node. find_source_node()</pre>	2.1	4.0	<pre>sphinx.util.node. get_node_source()</pre>	
<pre>sphinx.util.i18n.find_catalog()</pre>	2.1	4.0	<pre>sphinx.util.i18n. docname_to_domain()</pre>	
<pre>sphinx.util.i18n. find_catalog_files()</pre>	2.1	4.0	sphinx.util.i18n. CatalogRepository	
<pre>sphinx.util.i18n. find_catalog_source_files()</pre>	2.1	4.0	sphinx.util.i18n. CatalogRepository	
<pre>encoding argument of autodoc.Documenter.get_doc(), autodoc.DocstringSignatureMixin. get_doc(), autodoc.DocstringSignatureMixinfind_signature(), and autodoc.ClassDocumenter.get_doc()</pre>	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	
<pre>nodetype argument of sphinx.search. WordCollector.is_meta_keywords()</pre>	2.0	4.0	N/A	
<pre>suffix argument of BuildEnvironment.doc2path()</pre>	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	<pre>sphinx.builders.epub3. validate_config_values()</pre>	
sphinx.builders.html. JSONHTMLBuilder	2.0	4.0	sphinx.builders. serializinghtml.JSONHTMLBuilder	
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	

Table 4 – continued from previous page

lable 4 –	Table 4 – continued from previous page					
Target	Depre- cated	Re- moved	Alternatives			
sphinx.builders.html. WebHTMLBuilder	2.0	4.0	sphinx.builders. serializinghtml. PickleHTMLBuilder			
sphinx.builders.htmlhelp	2.0	4.0	sphinxcontrib.htmlhelp			
sphinx.builders.htmlhelp.	2.0	4.0	open()			
<pre>HTMLHelpBuilder.open_file()</pre>						
sphinx.builders.qthelp	2.0	4.0	sphinxcontrib.qthelp			
<pre>sphinx.cmd.quickstart. term_decode()</pre>	2.0	4.0	N/A			
<pre>sphinx.cmd.quickstart. TERM_ENCODING</pre>	2.0	4.0	sys.stdin.encoding			
<pre>sphinx.config.check_unicode()</pre>	2.0	4.0	N/A			
sphinx.config.string_classes	2.0	4.0	[str]			
sphinx.domains.cpp.	2.0	4.0	str(exc)			
DefinitionError.description						
<pre>sphinx.domains.cpp.NoOldIdError. description</pre>	2.0	4.0	str(exc)			
sphinx.domains.cpp. UnsupportedMultiCharacterCharLiteradecoded	2.0	4.0	str(exc)			
<pre>sphinx.ext.autosummary. Autosummary.warn()</pre>	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			
<pre>sphinx.ext.doctest. doctest_encode()</pre>	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			
<pre>sphinx.roles.emph_literal_role()</pre>	2.0	4.0	sphinx.roles.EmphasizedLiteral			
sphinx.roles.menusel_role()	2.0	4.0	sphinx.roles.GUILabel or sphinx.roles.MenuSelection			
<pre>sphinx.roles.index_role()</pre>	2.0	4.0	sphinx.roles.Index			
sphinx.roles.indexmarkup_role()	2.0	4.0	sphinx.roles.PEP or sphinx.roles.RFC			
<pre>sphinx.testing.util. remove_unicode_literal()</pre>	2.0	4.0	N/A			
sphinx.util.attrdict	2.0	4.0	N/A			
sphinx.util.force_decode()	2.0	5.0	N/A N/A			
sphinx.util.force_decode() 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	<pre>sphinxcontrib.serializinghtml. jsonimpl</pre>			
sphinx.util.osutil.EEXIST	2.0	4.0	errno.EEXIST or FileExistsError			
sphinx.util.osutil.EINVAL	2.0	4.0	errno.EINVAL			

Table 4 – continued from previous page

Target	Depre-	Re-	Alternatives
raiget	cated	moved	Allematives
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.	2.0	4.0	io.TextIOWrapper
TextIOWrapper			
<pre>sphinx.util.pycompat.UnicodeMixin</pre>	2.0	4.0	N/A
<pre>sphinx.util.pycompat.htmlescape()</pre>	2.0	4.0	html.escape()
<pre>sphinx.util.pycompat.indent()</pre>	2.0	4.0	textwrap.indent()
<pre>sphinx.util.pycompat.sys_encoding</pre>	2.0	4.0	<pre>sys.getdefaultencoding()</pre>
sphinx.util.pycompat.	2.0	4.0	sphinx.util.console.
terminal_safe()			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	2.0	4.0	<pre>IndexBuilder.feed(docname,</pre>
overriddent IndexBuilder.feed()			filename, title, doctree)
method.			
sphinx.writers.latex.ExtBabel	2.0	4.0	sphinx.builders.latex.util.
			ExtBabel
sphinx.writers.latex.	2.0	4.0	N/A
LaTeXTranslator.babel_defmacro()			
sphinx.application.Sphinx.	2.0	3.0	N/A
_setting_up_extension			
The importer argument of sphinx.ext.	2.0	3.0	N/A
autodoc.importerMockModule	• 0	• •	
sphinx.ext.autodoc.importer.	2.0	3.0	N/A
_MockImporter	- 0	•	
sphinx.io.SphinxBaseFileInput	2.0	3.0	N/A
sphinx.io.SphinxFileInput.	2.0	3.0	N/A
supported	2.0	2.0	N/A
sphinx.io.SphinxRSTFileInput	2.0	3.0	N/A N/A
sphinx.registry.	2.0	3.0	IV/A
<pre>SphinxComponentRegistry. add_source_input()</pre>			
sphinx.writers.latex.	2.0	3.0	N/A
LaTeXTranslator.	2.0	3.0	IVA
_make_visit_admonition()			
sphinx.writers.latex.	2.0	4.0	N/A
LaTeXTranslator.	2.0	4.0	IVA
collect_footnotes()			
sphinx.writers.texinfo.	2.0	3.0	N/A
TexinfoTranslator.	2.0	3.0	IVA
_make_visit_admonition()			
sphinx.writers.text.	2.0	3.0	N/A
TextTranslator.	2.0	5.0	1.07.1
_make_depart_admonition()			
sphinx.writers.latex.	2.0	4.0	N/A
LaTeXTranslator.	2.0	1.0	1 1/1 1
<pre>generate_numfig_format()</pre>			
3-1-0- a cc_1tam±19_101 mac()			

Table 4 – continued from previous page

Target	Depre-	Re-	Alternatives
·	cated	moved	
highlightlang	1.8	4.0	highlight
add_stylesheet()	1.8	6.0	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.	1.8	3.0	N/A
<pre>image.guess_mimetype()</pre>			
gettext_compact arguments of	1.8	3.0	N/A
sphinx.util.i18n.			
find_catalog_source_files()			
sphinx.io.SphinxI18nReader.	1.8	3.0	N/A
set_lineno_for_reporter()			
sphinx.io.SphinxI18nReader.line	1.8	3.0	N/A
sphinx.directives.other.	1.8	3.0	sphinx.domains.changeset.
VersionChange			VersionChange
sphinx.highlighting.	1.8	3.0	N/A
PygmentsBridge.unhighlight()	1.0	2.0	1,411
trim_doctest_flags arguments of	1.8	3.0	N/A
sphinx.highlighting.	1.0	2.0	1,711
PygmentsBridge			
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.
Spiritiv. exc. macribase. Hacribit eccive	1.0	3.0	MathDirective
<pre>sphinx.ext.mathbase.math_role()</pre>	1.8	3.0	docutils.parsers.rst.roles.
Spiritive exermination and en_rote()	1.0	2.0	math_role()
<pre>sphinx.ext.mathbase.setup_math()</pre>	1.8	3.0	add_html_math_renderer()
sphinx.ext.mathbase.	1.8	3.0	N/A
is_in_section_title()	1.0	2.0	1,411
sphinx.ext.mathbase.	1.8	3.0	sphinx.util.math.
<pre>get_node_equation_number()</pre>	1.0	2.0	<pre>get_node_equation_number()</pre>
sphinx.ext.mathbase.	1.8	3.0	sphinx.util.math.
wrap_displaymath()			wrap_displaymath()
sphinx.ext.mathbase.math(node)	1.8	3.0	docutils.nodes.math
sphinx.ext.mathbase.displaymath	1.8	3.0	docutils.nodes.math_block
(node)	1.0	2.0	ao da e225 i 110 a e5 i 111 a e11 <u>-</u> 2 2 5 c 11
sphinx.ext.mathbase.eqref(node)	1.8	3.0	sphinx.builders.latex.nodes.
op	1.0	2.0	math reference
<pre>viewcode_import (config value)</pre>	1.8	3.0	<pre>viewcode_follow_imported_members</pre>
sphinx.writers.latex.Table.	1.8	3.0	N/A
caption_footnotetexts			
sphinx.writers.latex.Table.	1.8	3.0	N/A
header_footnotetexts			
sphinx.writers.latex.	1.8	3.0	N/A
LaTeXTranslator.footnotestack			
sphinx.writers.latex.	1.8	3.0	N/A
LaTeXTranslator.			
in_container_literal_block			
sphinx.writers.latex.	1.8	3.0	N/A
•	-		

Table 4 – continued from previous page

Target	Depre- cated	Re- moved	Alternatives
sphinx.writers.latex.	1.8	3.0	N/A
LaTeXTranslator.			
next_hyperlink_ids			
sphinx.writers.latex.	1.8	3.0	N/A
LaTeXTranslator.			
restrict_footnote()			
sphinx.writers.latex.	1.8	3.0	N/A
LaTeXTranslator.			
unrestrict_footnote()	1.0	2.0	27/4
sphinx.writers.latex.	1.8	3.0	N/A
LaTeXTranslator.			
<pre>push_hyperlink_ids()</pre>	1.0	2.0	NI/A
<pre>sphinx.writers.latex. LaTeXTranslator.</pre>	1.8	3.0	N/A
pop_hyperlink_ids()			
sphinx.writers.latex.	1.8	3.0	N/A
LaTeXTranslator.bibitems	1.0	3.0	IVA
sphinx.writers.latex.	1.8	3.0	N/A
LaTeXTranslator.hlsettingstack	1.0	2.0	17/21
sphinx.writers.latex.ExtBabel.	1.8	3.0	N/A
<pre>get_shorthandoff()</pre>	1.0	2.0	- 111
sphinx.writers.html.	1.8	3.0	N/A
HTMLTranslator.highlightlang()			
sphinx.writers.html.	1.8	3.0	N/A
HTMLTranslator.			
highlightlang_base()			
sphinx.writers.html.	1.8	3.0	N/A
HTMLTranslator.			
highlightlangopts()			
sphinx.writers.html.	1.8	3.0	N/A
HTMLTranslator.			
highlightlinenothreshold()			
sphinx.writers.html5.	1.8	3.0	N/A
HTMLTranslator.highlightlang()	1.0	2.0	27/4
sphinx.writers.html5.	1.8	3.0	N/A
HTMLTranslator.			
<pre>highlightlang_base() sphinx.writers.html5.</pre>	1.8	3.0	N/A
HTMLTranslator.	1.0	3.0	N/A
highlightlangopts()			
sphinx.writers.html5.	1.8	3.0	N/A
HTMLTranslator.	1.0	5.0	14/11
highlightlinenothreshold()			
sphinx.writers.latex.	1.8	3.0	Nothing
LaTeXTranslator.			6
<pre>check_latex_elements()</pre>			
sphinx.application.	1.8	3.0	sphinx.config.CONFIG_FILENAME
CONFIG_FILENAME			
Config.check_unicode()	1.8	3.0	<pre>sphinx.config.check_unicode()</pre>

Table 4 – continued from previous page

Target	Depre-	Re-	Alternatives
	cated	moved	
Config.check_types()	1.8	3.0	sphinx.config.
,			check_confval_types()
dirname, filename and tags arguments	1.8	3.0	Config.read()
of Configinit()			·
The value of html_search_options	1.8	3.0	see html_search_options
<pre>sphinx.versioning.prepare()</pre>	1.8	3.0	sphinx.versioning.UIDTransform
<pre>Sphinx.override_domain()</pre>	1.8	3.0	add_domain()
Sphinx.import_object()	1.8	3.0	<pre>sphinx.util.import_object()</pre>
suffix argument of	1.8	3.0	add_source_suffix()
add_source_parser()			
BuildEnvironment.load()	1.8	3.0	pickle.load()
<pre>BuildEnvironment.loads()</pre>	1.8	3.0	pickle.loads()
<pre>BuildEnvironment.frompickle()</pre>	1.8	3.0	pickle.load()
<pre>BuildEnvironment.dump()</pre>	1.8	3.0	pickle.dump()
<pre>BuildEnvironment.dumps()</pre>	1.8	3.0	pickle.dumps()
<pre>BuildEnvironment.topickle()</pre>	1.8	3.0	pickle.dump()
BuildEnvironmentnitpick_ignore	1.8	3.0	nitpick_ignore
BuildEnvironment.versionchanges	1.8	3.0	N/A
<pre>BuildEnvironment.update()</pre>	1.8	3.0	Builder.read()
<pre>BuildEnvironment.read_doc()</pre>	1.8	3.0	Builder.read_doc()
<pre>BuildEnvironmentread_serial()</pre>	1.8	3.0	Builder.read()
<pre>BuildEnvironmentread_parallel()</pre>	1.8	3.0	Builder.read()
<pre>BuildEnvironment.write_doctree()</pre>	1.8	3.0	Builder.write_doctree()
BuildEnvironment.	1.8	3.0	ChangesDomain.note_changeset()
<pre>note_versionchange()</pre>			
warn() (template helper function)	1.8	3.0	warning()
source_parsers	1.8	3.0	add_source_parser()
sphinx.util.docutils.	1.8	3.0	Directive class of docutils
directive_helper()			
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()
<pre>sphinx.util.copy_static_entry()</pre>	1.5	3.0	sphinx.util.fileutil.
spirimi ucri (copy_scucre_circly ()	1.5		copy_asset()
sphinx.build_main()	1.7	2.0	<pre>sphinx.cmd.build.build_main()</pre>
sphinx.ext.intersphinx.debug()	1.7	2.0	sphinx.ext.intersphinx.
Spirita. Cac. Theer Spirita. acsag()	1.,	2.0	<pre>inspect_main()</pre>
sphinx.ext.autodoc.	1.7	2.0	sphinx.util.inspect.Signature
format_annotation()	1.,		-p
sphinx.ext.autodoc.	1.7	2.0	sphinx.util.inspect.Signature
formatargspec()	1.7		op
sphinx.ext.autodoc.	1.7	2.0	sphinx.util.docutils.
AutodocReporter	1.7	2.0	switch_source_input()
sphinx.ext.autodoc.	1.7	2.0	add_autodocumenter()
add_documenter()	1./	2.0	aaa_aacoaocamenteet ()
sphinx.ext.autodoc.AutoDirective.	1.7	2.0	add_autodocumenter()
_register	1.7	2.0	ada_aatoaocamenter()
			continues on next page

Table 4 – continued from previous page

Target	Depre- cated	Re- moved	Alternatives
AutoDirectivespecial_attrgetters	1.7	2.0	add_autodoc_attrgetter()
<pre>Sphinx.warn(), Sphinx.info()</pre>	1.6	2.0	Logging API
<pre>BuildEnvironment.set_warnfunc()</pre>	1.6	2.0	Logging API
<pre>BuildEnvironment.note_toctree()</pre>	1.6	2.0	Toctree.note() (in sphinx. environment.adapters.toctree)
<pre>BuildEnvironment.get_toc_for()</pre>	1.6	2.0	<pre>Toctree.get_toc_for() (in sphinx. environment.adapters.toctree)</pre>
<pre>BuildEnvironment. get_toctree_for()</pre>	1.6	2.0	<pre>Toctree.get_toctree_for() (in sphinx.environment.adapters. toctree)</pre>
<pre>BuildEnvironment.create_index()</pre>	1.6	2.0	<pre>IndexEntries.create_index() (in sphinx.environment.adapters. indexentries)</pre>
sphinx.websupport	1.6	2.0	sphinxcontrib-websupport ¹³²⁸
StandaloneHTMLBuilder.css_files	1.6	2.0	add_stylesheet()
<pre>document.settings.gettext_compact</pre>	1.8	1.8	<pre>gettext_compact</pre>
<pre>Sphinx.status_iterator()</pre>	1.6	1.7	<pre>sphinx.util.status_iterator()</pre>
<pre>Sphinx.old_status_iterator()</pre>	1.6	1.7	<pre>sphinx.util. old_status_iterator()</pre>
<pre>Sphinxdirective_helper()</pre>	1.6	1.7	<pre>sphinx.util.docutils. directive_helper()</pre>
sphinx.util.compat.Directive	1.6	1.7	docutils.parsers.rst.Directive
sphinx.util.compat.	1.6	1.7	sphinx.util.docutils.
docutils_version			version_info

1 Note

On deprecating on public APIs (internal functions and classes), we also follow the policy as much as possible.

 $[\]frac{1327}{1328}\ https://github.com/sphinx-doc/sphinx/blob/v7.0.0/sphinx/templates/quickstart/Makefile_t$

CHAPTER

SEVEN

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 = {
    'passoptionstopackages': r'''
\PassOptionsToPackage{svgnames}{xcolor}
    '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}
    'sphinxsetup': 'TitleColor=DarkGoldenrod',
    'fncychap': r'\usepackage[Bjornstrup]{fncychap}',
    'printindex': r'\footnotesize\raggedright\printindex',
latex_show_urls = 'footnote'
```

1 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.

7.1 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*. It allows also local configurations inserted in generated files, via raw ¹³²⁹ directives. For example, in the PDF documentation this chapter is styled especially, as will be described later.

¹³²⁹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#raw

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'

Added 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: ''

Added in version 1.4.

'babel'

"babel" package inclusion, default r'\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 polyglossia¹³³⁰, but one should be aware that current babel¹³³¹ 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 modifying 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: r'\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 with 'xelatex' (not 'lualatex') the default is to use babel, not polyglossia.

¹³³⁰ https://ctan.org/pkg/polyglossia

¹³³¹ https://ctan.org/pkg/babel

Changed in version 7.4.0: For French with 'lualatex' the default is to use babel.

'fontpkg'

Font package inclusion. The default with 'pdflatex' is:

```
r"""\usepackage{tgtermes}
\usepackage{tgheros}
\renewcommand\ttdefault{txtt}
"""
```

For 'xelatex' and 'lualatex' on the other hand the \setmainfont, \setsansfont and \setmonofont commands of LaTeX package fontspec (included via 'fontenc') are used to set up the OpenType fonts GNU FreeSerif, FreeSans, and FreeMono (scaled with ratio 0.9) as document fonts.

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. At 4.0.0 these commands were moved to the 'fontsubstitution' key.

Changed in version 4.0.0: The default font setting was changed. As shown above 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.

Changed in version 8.1.0: The monospace font FreeMono used with Unicode engines is loaded at scale 0.9. This replaces the former mechanism via 'fvset' which configured code-blocks to use \small. Inline literals now fit better in their surrounding text, and it is easier to set up custom fonts, as 'fvset' does not intervene anymore by default.

'fncychap'

Inclusion of the "fncychap" package (which makes fancy chapter titles), default r'\usepackage[Bjarne]{fncychap}' for English documentation (this option is slightly customized by Sphinx), r'\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: $r'\usepackage[Bjarne]{fncychap}'$ for English documents, $r'\usepackage[Sonny]{fncychap}'$ 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 <code>latex_additional_files</code>, for example:

```
latex_additional_files = ["mystyle.sty"]
```

Do not use .tex as suffix, else the file is submitted itself to the PDF build process, use .tex.txt or .sty as in the examples above.

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)
```

Added in version 1.3.

'atendofbody'

Additional document content (right before the indices).

Default: ''

Added in version 1.5.

'extrapackages'

Additional LaTeX packages. For example:

```
latex_elements = {
    'extrapackages': r'\usepackage{isodate}'
}
```

The specified LaTeX packages will be loaded before hyperref package and packages loaded from Sphinx extensions.

If you'd like to load additional LaTeX packages after hyperref, use 'preamble' key instead.

Default: ''

Added 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: ''

Added 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.

Marning

- 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

Added in version 1.5.

'inputenc'

"inputenc" package inclusion.

Default: r'\usepackage[utf8]{inputenc}' when using pdflatex, else ''.



1 Note

If using utf8x in place of utf8 it is mandatory to extend the LaTeX preamble with suitable \ PreloadUnicodePage{<number>} commands, as per the utf8x documentation (texdoc ucs on a TeXLive based TeX installation). Else, unexpected and possibly hard-to-spot problems (i.e. not causing a build crash) may arise in the PDF, in particular regarding hyperlinks.

Even if these precautions are taken, PDF build via pdflatex engine may crash due to upstream LaTeX not being fully compatible with utf8x. For example, in certain circumstances related to code-blocks, or attempting to include images whose filenames contain Unicode characters. Indeed, starting in 2015, upstream LaTeX with pdflatex engine has somewhat enhanced native support for Unicode and is becoming more and more incompatible with utf8x. In particular, since the October 2019 LaTeX release, filenames can use Unicode characters, and even spaces. At Sphinx level this means e.g. that the image¹³³² and figure¹³³³ directives are now compatible with such filenames for PDF via LaTeX output. But this is broken if utf8x is in use.

Changed in version 1.4.3: Previously r'\usepackage[utf8] {inputenc}' was used for all compilers.

'cmappkg'

"cmap" package inclusion.

Default: r'\usepackage{cmap}'

Added in version 1.2.

'fontenc'

Its default, for 'pdflatex' as latex_engine, is r'\usepackage[T1]{fontenc}'. Replace it (if using 'pdflatex') with:

- r'\usepackage [X2,T1] {fontenc} ' if you need occasional Cyrillic letters (физика частиц),
- r'\usepackage [LGR, T1] {fontenc}' if you need occasional Greek letters (Σωματιδιαχή φυσιχή),
- r'\usepackage[LGR, X2, T1] {fontenc}' if you need both.

The TeX installation may need some extra packages. For example, on Ubuntu xenial:

• texlive-lang-greek and cm-super are needed for Greek (LGR),

 $^{^{1332}\} https://docutils.sourceforge.io/docs/ref/rst/directives.html\#image$

¹³³³ https://docutils.sourceforge.io/docs/ref/rst/directives.html#figure

• texlive-lang-cyrillic and cm-super are needed for Cyrillic (X2).

With 'xelatex' and 'lualatex', support for Greek and Cyrillic is out-of-the-box: this 'fontenc' key defaults to including the LaTeX package fontspec (with some extras described below) and selects the GNU FreeSerif font as body font. See 'fontpkg'.

Changed in version 1.5: Defaults to r'\usepackage{fontspec}' if latex_engine is set to 'xelatex'.

Changed in version 1.6: Defaults to r'\usepackage{fontspec}' if latex_engine is set to 'lualatex'.

Changed in version 2.0: 'lualatex' executes additionally \defaultfontfeatures[\rmfamily, \sffamily]{} to disable TeX ligatures for << and >>.

Changed in version 2.0: Extra LaTeX configuration is automatically executed if LGR, T2A, or X2 are detected in this key, in order to support occasional Greek or Cyrillic with 'pdflatex'.

Changed in version 2.2.1: Documents having Greek as main language default to 'xelatex' and should not set the 'fontenc' key, which will load fontspec.

Changed in version 2.3.0: 'xelatex' executes \defaultfontfeatures[\rmfamily,\sffamily]{} in order to avoid contractions of -- into en-dash and also transforms of straight quotes into curly quotes (which otherwise would happen even with *smartquotes* 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'.

Added in version 4.0.0.

'textgreek'

For the support of occasional Greek letters.

It is ignored with 'platex', 'xelatex' or 'lualatex' as <code>latex_engine</code> and defaults to either the empty string or to <code>r'\usepackage{textalpha}'</code> 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: r'\usepackage{textalpha}' or '' if fontenc does not include the LGR option.

Added in version 2.0.

'geometry'

"geometry" package inclusion, defaults to r'\usepackage{geometry}' or r'\usepackage[dvipdfm]{geometry}' for Japanese documents. The Sphinx LaTeX style file executes additionally:

\PassOptionsToPackage{hmargin=1in,vmargin=1in,marginpar=0.5in}{geometry}

which can be customized via corresponding 'sphinxsetup' options.

Added in version 1.5.

Changed in version 1.5.2: dvipdfm option if latex_engine is 'platex'.

Added 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 "hyperap" 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.

Added 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 r'\newcommandsphinxbackoftitlepage{<Extra the key value set to material>}\sphinxmaketitle', then <Extra material> will be typeset on back of title page ('manual' docclass only).

Default: r'\sphinxmaketitle'

Changed in version 1.8.3: Original \maketitle from document class is not overwritten, hence is reusable as part of some custom setting for this key.

Added 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 r'\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: r'\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'

Added in version 1.2.

Changed in version 1.6: Remove unneeded {} formerly located after \hrule.

'makeindex'

"makeindex" \begin{document}. With call, the last thing before

r'\usepackage[columns=1]{idxlayout}\makeindex' the index will use only one column. You may have to install idxlayout LaTeX package.

Default: r'\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 r'\footnotesize\raggedright\printindex'. Or, to obtain a one-column index, use r'\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: r'\printindex'

'fvset'

Customization of fancyvrb LaTeX package.

The default value is $r'\fvset\{fontsize=auto\}'$ which means that the font size will adjust correctly if a code-block ends up in a footnote. You may need to modify this when using a custom monospace font, for example set it to $r'\fvset\{fontsize=\small\}'$ if it is Courier-like (for Unicode engines, it is recommended to use rather the Scale interface of \small command from fontspec).

Default: r'\fvset{fontsize=auto}'

Added in version 1.8.

Changed in version 2.0: For 'xelatex' and 'lualatex' defaults to $r'\setminus fvset\{fontsize=\setminus small\}'$ 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 $r'\fvset{fontsize=\small}'$.

Changed in version 4.1.0: Changed default for Chinese documents to $r'\fvset\{fontsize=\small, formatcom=\xeCJKVerbAddon\}'$

Changed in version 8.1.0: Changed default for 'xelatex' and 'lualatex' to be also r'\fvset{fontsize=auto}'. The rescaling for default monospace font FreeMono is now set via the LaTeX package fontspec interface rather. See 'fontpkg'.

Keys that are set by other options and therefore should not be overridden are:

'docclass' 'classoptions' 'title' 'release' 'author'

7.2 The sphinxsetup configuration setting

Added in version 1.5.

The 'sphinxsetup' key of *latex_elements* provides a LaTeX-type customization interface:

```
latex_elements = {
    'sphinxsetup': 'key1=value1, key2=value2, ...',
}
```

LaTeX syntax for boolean keys requires *lowercase* true or false e.g 'sphinxsetup': "verbatimwrapslines=false". If setting a boolean key to true, =true is optional. Spaces around the commas and equal signs are ignored, spaces inside LaTeX macros may be significant. Do not use ticks/quotes to enclose string or numerical values.

The 'sphinxsetup' 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,...}
```

It is possible to insert uses of the \sphinxsetup LaTeX macro directly into the body of the document, via the raw directive:

```
.. raw:: latex
   \begingroup
   \sphinxsetup{%
        TitleColor=DarkGoldenrod,
        ... more comma separated key=value using LaTeX syntax ...
}
All elements here will be under the influence of the raw ``\sphinxsetup``
settings.
.. raw:: latex
   \endgroup
From here on, the raw ``\sphinxsetup`` has no effect anymore.
```

This is the technique which has been used to style especially the present part of the documentation for the PDF output. The actually used options will be found at top of doc/latex.rst at the development repository 1334.

The color used in the above example is available from having passed the sygnames option to the "xcolor" package:

```
latex_elements = {
    'passoptionstopackages': r'\PassOptionsToPackage{svgnames}{xcolor}',
}
```

bookmarksdepth

Controls the depth of the collapsible bookmarks panel in the PDF. May be either a number (e.g. 3) or a LaTeX sectioning name (e.g. subsubsection, i.e. without backslash). For details, refer to the hyperref LaTeX docs.

Default: 5

Added in version 4.0.0.

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})

¹³³⁴ https://github.com/sphinx-doc/sphinx

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',

Added 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

Added in version 1.5.3.

mathnumsep

This defaults to the string (without quotes!) as set by *math_numsep* (which itself defaults to '.'). Use it if a different setting is needed for LaTeX output.

Added in version 8.1.0.

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 color.

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.

1 Note

It is assumed that the Pygments¹³³⁵ 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

Added 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,

¹³³⁵ https://pygments.org/

and if verbatimforcewraps mode is on, the input line will be reset using the forceful algorithm which applies breakpoints at each character.

Default: 3

Added 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.

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 available 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

Added 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 page breaks.

Default: true

Added 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 1 (left aligned), r (right aligned) or c (centered).

Default: r

Added in version 1.7.

parsedliteralwraps

Boolean to specify if long lines in parsed-literal 1336's contents should wrap.

Default: true

Added in version 1.5.2: set this option value to false to recover former behavior.

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

Added in version 1.5: set this option value to false to recover former behavior.

Changed in version 2.3.0: added potential breakpoint at \ characters.

¹³³⁶ https://docutils.sourceforge.io/docs/ref/rst/directives.html#parsed-literal

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:

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 accommodate various font sizes (e.g. code-blocks can be in footnotes).

1 Note

Values for color keys must either:

- obey the syntax of the **\definecolor** LaTeX command, e.g. something such as VerbatimColor= $\{rgb\}\{0.2,0.3,0.5\}$ or $\{RGB\}\{37,23,255\}$ or $\{gray\}\{0.75\}$ or $\{HTML\}\{808080\}$ or ...
- or obey the syntax of the \colorlet command from package xcolor e.g. VerbatimColor=red!10 or red!50!green or -red!75 or MyPreviouslyDefinedColor or... Refer to xcolor¹³³⁷ documentation for this syntax.

Changed in version 5.3.0: Formerly only the \definecolor syntax was accepted.

TitleColor

The color for titles (as configured via use of package "titlesec".)

Default: {rgb}{0.126,0.263,0.361}

InnerLinkColor

A color passed to hyperref as value of linkcolor and citecolor.

Default: {rgb}{0.208,0.374,0.486}.

OuterLinkColor

A color passed to hyperref as value of filecolor, menucolor, and urlcolor.

Default: {rgb}{0.216,0.439,0.388}

VerbatimColor

The background color for *code-blocks*.

Default: $\{RGB\}\{242,242,242\}$ (same as $\{gray\}\{0.95\}$).

Changed in version 6.0.0: Formerly, it was {rgb}{1,1,1} (white).

VerbatimBorderColor

The frame color.

Default: {RGB}{32,32,32}

Changed in version 6.0.0: Formerly it was {rgb}{0,0,0} (black).

¹³³⁷ https://ctan.org/pkg/xcolor

VerbatimHighlightColor

The color for highlighted lines.

Default: {rgb}{0.878,1,1}

Added in version 1.6.6.

TableRowColorHeader

Sets the background color for (all) the header rows of tables.

It will have an effect only if either the <code>latex_table_style</code> contains 'colorrows' or if the table is assigned the colorrows class. It is ignored for tables with nocolorrows class.

As for the other 'sphinxsetup' keys, it can also be set or modified from a \sphinxsetup{...} LaTeX command inserted via the raw¹³³⁸ directive, or also from a LaTeX environment associated to a *container class* and using such \sphinxsetup{...}.

Default: {gray}{0.86}

There is also TableMergeColorHeader. If used, sets a specific color for merged single-row cells in the header.

Added in version 5.3.0.

TableRowColorOdd

Sets the background color for odd rows in tables (the row count starts at 1 at the first non-header row). Has an effect only if the <code>latex_table_style</code> contains 'colorrows' or for specific tables assigned the colorrows class.

Default: {gray}{0.92}

There is also TableMergeColorOdd.

Added in version 5.3.0.

TableRowColorEven

Sets the background color for even rows in tables.

Default {gray}{0.98}

There is also TableMergeColorEven.

Added in version 5.3.0.

verbatimsep

The separation between code lines and the frame.

See Additional CSS-like 'sphinxsetup' keys for its alias pre_padding and additional keys.

Default: \fboxsep

verbatimborder

The width of the frame around *code-blocks*. See also *Additional CSS-like 'sphinxsetup' keys* for pre_border-width.

Default: \fboxrule

Important

Since 8.1.0 it is possible to style separately the topic 1339 , contents 1340 , and sidebar 1341 directives, and their defaults differ. See *Additional CSS-like 'sphinxsetup' keys*. The next three keys are kept as legacy interface not distinguishing between the three directives.

¹³³⁸ https://docutils.sourceforge.io/docs/ref/rst/directives.html#raw

shadowsep

This legacy option sets the padding (same in all directions) simultaneously for the topic 1342, contents 1343, and sidebar 1344 directives.

shadowsize

This legacy option sets the shadow width simultaneously for the topic 1345, contents 1346, and sidebar 1347 directives.

shadowrule

This legacy option sets the border-width (same on all sides) simultaneously for the topic ¹³⁴⁸, contents ¹³⁴⁹, and sidebar ¹³⁵⁰ directives

Important

At 7.4.0 all admonitions (not only danger-type) use the possibilities which were added at 5.1.0 and 6.2.0. All defaults have changed.

iconpackage

The name of the LaTeX package used for icons in the admonition titles. It defaults to fontawesome5 or to fall-back fontawesome. In case neither one is available the option value will automatically default to none, which means that no attempt at loading a package is done. Independently of this setting, arbitrary LaTeX code can be associated to each admonition type via div.<type>_icon-title keys which are described in the Additional CSS-like 'sphinxsetup' keys section. If these keys are not used, Sphinx will either apply its default choices of icons (if fontawesome{5,} is available) or not draw the icon at all. Notice that if fall-back fontawesome is used the common icon for caution and danger will default to "bolt" not "radiation", which is only found in fontawesome5.

Added in version 7.4.0.

${\tt noteBorderColor}, {\tt hintBorderColor}, {\tt importantBorderColor}, {\tt tipBorderColor}$

The color for the admonition border.

Default: {RGB}{134,152,155}.

Changed in version 7.4.0.

noteBgColor, hintBgColor, importantBgColor, tipBgColor

The color for the admonition background.

Default: {RGB}{247,247,247}.

Added in version 6.2.0.

Changed in version 7.4.0.

noteTextColor, hintTextColor, importantTextColor, tipTextColor

The color for the admonition contents.

```
1339 https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic
```

¹³⁴⁰ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents

¹³⁴¹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar

¹³⁴² https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

¹³⁴³ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents

https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar

¹³⁴⁵ https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

¹³⁴⁶ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents

¹³⁴⁷ https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar

¹³⁴⁸ https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

¹³⁴⁹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents

https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar
 https://docutils.sourceforge.io/docs/ref/rst/directives.html#caution

¹³⁵² https://docutils.sourceforge.io/docs/ref/rst/directives.html#danger

Default: unset (contents text uses ambient text color, a priori black)

Added in version 6.2.0: To be considered experimental until 7.0.0. These options have aliases div. note_TeXcolor (etc) described in *Additional CSS-like 'sphinxsetup' keys*. Using the latter will let Sphinx switch to a more complex LaTeX code, which supports the customizability described in *Additional CSS-like 'sphinxsetup' keys*.

noteTeXextras, hintTeXextras, importantTeXextras, tipTeXextras

Some extra LaTeX code (such as **\bfseries** or **\footnotesize**) to be executed at start of the contents.

Default: empty

Added in version 6.2.0: To be considered experimental until 7.0.0. These options have aliases div. note_TeXextras (etc) described in *Additional CSS-like 'sphinxsetup' keys*.

noteborder, hintborder, importantborder, tipborder

The width of the border. See Additional CSS-like 'sphinxsetup' keys for keys allowing to configure separately each border width.

Default: 0.5pt

warningBorderColor, and (caution|attention|danger|error)BorderColor

The color for the admonition border.

Default: {RGB}{148,0,0} except for error which uses red.

Changed in version 7.4.0.

warningBgColor, cautionBgColor, attentionBgColor, dangerBgColor, errorBgColor

The background color for the admonition background.

Default: {RGB}{247,247,247}.

Changed in version 7.4.0.

warningborder, cautionborder, attentionborder, dangerborder, errorborder

The width of the admonition frame. See *Additional CSS-like 'sphinxsetup' keys* for keys allowing to configure separately each border width.

Default: 1pt except for error which uses 1.25pt.

Changed in version 7.4.0.

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

Added in version 1.5.

HeaderFamily

default \sffamily\bfseries. Sets the font used by headings.

7.3 Additional CSS-like 'sphinxsetup' keys

Added in version 5.1.0: For code-block, topic ¹³⁵³ and contents ¹³⁵⁴ directive, and strong-type admonitions (warning ¹³⁵⁵, error ¹³⁵⁶, ...).

Added in version 6.2.0: Also the note¹³⁵⁷, hint¹³⁵⁸, important¹³⁵⁹ and tip¹³⁶⁰ admonitions can be styled this way. Using for them *any* of the listed options will trigger usage of a more complex LaTeX code than the one used per default (sphinxheavybox vs sphinxlightbox). Setting the new noteBgColor (or hintBgColor, ...) also triggers usage of sphinxheavybox for note¹³⁶¹ (or hint¹³⁶², ...).

Added in version 7.4.0: For *all* admonition types, the default configuration does set a background color (hence the richer sphinxheavybox is always used).

Important

Further, all admonition titles are by default styled using a colored row and an icon, which are modeled on the current rendering of Sphinx own docs at https://www.sphinx-doc.org. CSS-named alike keys are added to set the foreground and background colors for the title as well as the LaTeX code for the icon.

Added in version 7.4.0: Customizability of the seealso and todo directives.

Added in version 8.1.0: Separate customizability and new defaults for the topic 1363, contents 1364, and sidebar 1365 directives.

Perhaps in future these 5.1.0 (and 6.2.0) novel settings will be optionally imported from some genuine CSS external file, but currently they have to be used via the 'sphinxsetup' interface (or the \sphinxsetup LaTeX command inserted via the raw¹³⁶⁶ directive) and the CSS syntax is only imitated.

Important

Low-level LaTeX errors causing a build failure can happen if the input syntax is not respected.

• In particular colors must be input as for the other color related options previously described, i.e. either in the \definecolor syntax or via the \colorlet syntax:

```
...<other options>
div.warning_border-TeXcolor={rgb}{1,0,0},% \definecolor syntax
div.error_background-TeXcolor=red!10,% \colorlet syntax
...<other options>
```

• A colon in place of the equal sign will break LaTeX.

 $^{^{1353}\} https://docutils.sourceforge.io/docs/ref/rst/directives.html\#topic$

¹³⁵⁴ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents

¹³⁵⁵ https://docutils.sourceforge.io/docs/ref/rst/directives.html#warning

¹³⁵⁶ https://docutils.sourceforge.io/docs/ref/rst/directives.html#error

¹³⁵⁷ https://docutils.sourceforge.io/docs/ref/rst/directives.html#note

¹³⁵⁸ https://docutils.sourceforge.io/docs/ref/rst/directives.html#hint

¹³⁵⁹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#important

https://docutils.sourceforge.io/docs/ref/rst/directives.html#tip

¹³⁶¹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#note

¹³⁶² https://docutils.sourceforge.io/docs/ref/rst/directives.html#hint

¹³⁶³ https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

¹³⁶⁴ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents

¹³⁶⁵ https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar

¹³⁶⁶ https://docutils.sourceforge.io/docs/ref/rst/directives.html#raw

- ...border-width or ...padding expect a *single* dimension: they can not be used so far with space separated dimensions.
- ...top-right-radius et al. values may be either a single or *two* space separated dimensions.
- Dimension specifications must use TeX units such as pt or cm or in. The px unit is recognized by pdflatex and lualatex but not by xelatex or platex.
- It is allowed for such specifications to be so-called "dimensional expressions", e.g. \fboxsep+2pt or 0. 5\baselineskip are valid inputs. The expressions will be evaluated only at the typesetting time. Be careful though if using as in these examples TeX control sequences to double the backslash or to employ a raw Python string for the value of the 'sphinxsetup' key.
- As a rule, avoid inserting unneeded spaces in the key values: especially for the radii an input such 2 pt 3pt will break LaTeX. Beware also that \fboxsep \fboxsep will not be seen as space separated in LaTeX. You must use something such as {\fboxsep} \fboxsep. Or use directly 3pt 3pt which is a priori equivalent and simpler.

The options are all named in a similar pattern which depends on a prefix, which is then followed by an underscore, then the property name.

Directive	Option prefix	LaTeX environment
code-block	pre	sphinxVerbatim
literalinclude	pre	sphinxVerbatim
topic ¹³⁶⁷	div.topic	sphinxtopic
contents ¹³⁶⁸	div.contents	sphinxcontents
sidebar ¹³⁶⁹	div.sidebar	sphinxsidebar
note ¹³⁷⁰	div.note	sphinxnote
warning ¹³⁷¹	div.warning	sphinxwarning
further admonition types <type></type>	<pre>div.<type></type></pre>	sphinx <type></type>
seealso	div.seealso	sphinxseealso
todo	div.todo	sphinxtodo

Here are now these options as well as their common defaults. Replace below cprefix by the actual prefix as explained above. Don't forget the underscore separating the prefix from the property names.

- <prefix>_border-top-width,
 <prefix>_border-right-width,
 - The second of th
 - <prefix>_border-bottom-width,
 - <prefix>_border-left-width,

<prefix>_border-width. The latter can (currently) be only a single dimension which then sets all four
others.

The default is that all those dimensions are equal. They are set to:

- 0.4pt for code-block.
- 0.5pt for topic 1372 and contents 1373 directive,

¹³⁶⁷ https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents

 ¹³⁶⁹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar
 1370 https://docutils.sourceforge.io/docs/ref/rst/directives.html#note

¹³⁷¹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#warning

¹³⁷² https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

¹³⁷³ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents

- 1pt for sidebar¹³⁷⁴ directive,
- 0.5pt for note¹³⁷⁵ and other "light" admonitions,
- 0.5pt for seealso and todo directives,
- 1pt for warning¹³⁷⁶ and other "strong" admonitions except error¹³⁷⁷ which uses 1.25pt.

Changed in version 7.4.0: Changed defaults for topic ¹³⁷⁸ and error ¹³⁷⁹.

Changed in version 8.1.0: Distinct from topic 1380 defaults for sidebar 1381.

- fix>_padding-top,
 <prefix>_padding-right,
 <prefix>_padding-bottom,
 <prefix>_padding-left,

<prefix>_padding. The latter can (currently) be only a single dimension which then sets all four others.

The defaults:

- all four 3pt for code-block,
- 6pt, 7pt, 6pt, 7pt for topic 1382,
- 10pt, 7pt, 12pt, 7pt for contents 1383,
- 6pt, 5.5pt, 6pt, 5.5pt for sidebar 1384,
- 6pt, 7pt, 6pt, 7pt for all "light" admonitions as well as the seealso and todo directives.
- 6pt, 6.5pt, 6pt, 6.5pt for the strong admonition types except error¹³⁸⁵ which uses horizontal padding of 6.25pt.

Changed in version 7.4.0: All defaults were changed, except for *code-block*. Admonitions are set-up so that left (or right) padding plus left (or right) border-width add up always to 7.5pt, so contents align well vertically across admonition types on same page in PDF. This is only a property of defaults, not a constraint on possible user choices.

Changed in version 8.1.0: Separate defaults for topic 1386, contents 1387, and sidebar 1388.

```
• <prefix>_border-top-left-radius,
  <prefix>_border-top-right-radius,
  <prefix>_border-bottom-right-radius,
  <prefix>_border-bottom-left-radius,
```

¹³⁷⁴ https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar

¹³⁷⁵ https://docutils.sourceforge.io/docs/ref/rst/directives.html#note

¹³⁷⁶ https://docutils.sourceforge.io/docs/ref/rst/directives.html#warning

¹³⁷⁷ https://docutils.sourceforge.io/docs/ref/rst/directives.html#error

¹³⁷⁸ https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

¹³⁷⁹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#error

¹³⁸⁰ https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar

¹³⁸² https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

 $^{^{1383}\} https://docutils.sourceforge.io/docs/ref/rst/directives.html\#table-of-contents$

¹³⁸⁴ https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar

¹³⁸⁵ https://docutils.sourceforge.io/docs/ref/rst/directives.html#error

¹³⁸⁶ https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

¹³⁸⁷ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents

¹³⁸⁸ https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar

<prefix>_border-radius. This last key sets the first four to its assigned value. Each key value can be either
a single, or two, dimensions which are then space separated.

The defaults:

- 3pt for code-block (since 6.0.0) and all corners,
- 8pt for all corners of topic 1389,
- 12pt for the bottom right corner of contents ¹³⁹⁰, others use **0pt**,
- 12pt for the top-left and bottom-right corners for sidebar¹³⁹¹, **Opt** for top-right and bottom-left.
- all radii set to 5pt for note¹³⁹², hint¹³⁹³ and tip¹³⁹⁴,
- Opt, i.e. straight corners for all other directives.

Changed in version 7.4.0: topic 1395 and note 1396-like admonitions acquire (at least one) rounded corners.

Changed in version 8.1.0: Separate defaults for topic 1397, contents 1398, and sidebar 1399.

See a remark above about traps with spaces in LaTeX.

- erix>_box-shadow is special in so far as it may be:
 - the none keyword,
 - or a single dimension (giving both x-offset and y-offset),
 - or two dimensions (separated by a space),
 - or two dimensions followed by the keyword inset.

The x-offset and y-offset may be negative. A negative x-offset means that the shadow is on the left. The shadow extends into the page margin, whether the offset is positive or negative.

The default is none *except* for the contents 1400 directive which uses 4pt 4pt.

Changed in version 8.1.0: No shadow per default for topic 1401 and sidebar 1402.

• Fix>_border-TeXcolor,

```
<prefix>_background-TeXcolor,
```

<prefix>_box-shadow-TeXcolor,

<prefix>_TeXcolor. These are colors.

Since 6.0.0 the border and background colors of code-block, default respectively to {RGB}{32,32,32} (i.e. {HTML}{202020}), and {RGB}{242,242,242} (i.e. {gray}{0.95} or {HTML}{F2F2F2}).

At 7.4.0 other directives acquire non-black/white default border and background colors. Here they are using xcolor hexadecimal notation (which requires always 6 hexadecimal digits):

- {HTML}{F7F7F7} serves as background color to all.

¹³⁸⁹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

¹³⁹⁰ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents

¹³⁹¹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar

¹³⁹² https://docutils.sourceforge.io/docs/ref/rst/directives.html#note

¹³⁹³ https://docutils.sourceforge.io/docs/ref/rst/directives.html#hint

¹³⁹⁴ https://docutils.sourceforge.io/docs/ref/rst/directives.html#tip

¹³⁹⁵ https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

¹³⁹⁶ https://docutils.sourceforge.io/docs/ref/rst/directives.html#note

¹³⁹⁷ https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

¹³⁹⁸ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents

¹³⁹⁹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar

¹⁴⁰⁰ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents

¹⁴⁰¹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

 $^{^{1402}\} https://docutils.sourceforge.io/docs/ref/rst/directives.html \# side bar$

- {HTML}{86989B} is border color of light admonitions (inclusive of seealso and todo) as well as of topic¹⁴⁰³, contents¹⁴⁰⁴ and sidebar¹⁴⁰⁵ directives.
- {HTML}{940000} is border color of warning 1406 -type admonitions, except error 1407 which uses {HTML}{B40000}.

The only directives displaying a shadow per default are contents¹⁴⁰⁸ and sidebar¹⁴⁰⁹. The shadow-color for the former is {HTML}{6C6C6C} and for the latter {HTML}{9EACAF}.

The refix>_TeXcolor stands for the CSS property "color", i.e. it influences the text color of the contents.
As for the three other options, the naming TeXcolor is to stress that the input syntax is the TeX one for colors
not an HTML/CSS one. If package xcolor is available in the LaTeX installation, one can use directly named
colors as key values. Consider passing options such as dvipsnames, svgnames or x11names to xcolor via
'passoptionstopackages' key of latex_elements.

If refix>_TeXcolor is set, a \color command is inserted at start of the directive contents; for admonitions, this happens after the heading which reproduces the admonition type.

The next keys, for admonitions, $topic^{1410}$, $contents^{1411}$, and $sidebar^{1412}$, were all three added at 7.4.0 (and 8.1.0 for the latter three).

• div.<type>_title-background-TeXcolor: the background color for the title.

Important

The colored title-row is produced as a result of the Sphinx default definitions for the various \sphinxstyle<type>title commands, which employ the \sphinxdotitlerow LaTeX command. See Macros.

- div.<type>_title-foreground-TeXcolor: the color to be used for the icon (it applies only to the icon, not to the title of the admonition).
- div.<type>_title-icon: the LaTeX code responsible for producing the icon. For example, the default for note¹⁴¹³ is div.note_title-icon=\faIcon{info-circle}. This uses a command from the LaTeX fontawesome5 package, which is loaded automatically if available.

If neither fontawesome 5 nor fall-back fontawesome (for which the associated command is \faicon, not \faIcon) are found, or if the iconpackage key of 'sphinxsetup' is set to load some other user-chosen package, or no package at all, all the title-icons default to empty LaTeX code. It is up to user to employ this interface to inject the icon (or anything else) into the PDF output.

1 Note

- ¹⁴⁰³ https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic
- 1404 https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents
- 1405 https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar
- 1406 https://docutils.sourceforge.io/docs/ref/rst/directives.html#warning
- 1407 https://docutils.sourceforge.io/docs/ref/rst/directives.html#error
- 1408 https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents
- 1409 https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar
- 1410 https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic
- ¹⁴¹¹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents
- 1412 https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar
- 1413 https://docutils.sourceforge.io/docs/ref/rst/directives.html#note

• All directives support box-decoration-break to be set to slice.

Changed in version 6.2.0: Formerly, only *code-block* did. The default remains clone for all other directives, but this will probably change at 7.0.0.

• The corners of rounded boxes may be elliptical.

Changed in version 6.2.0: Formerly, only circular rounded corners were supported and a rounded corner forced the whole frame to use the same constant width from prefix>_border-width.

• Inset shadows are incompatible with rounded corners. In case both are specified the inset shadow will simply be ignored.

Changed in version 6.2.0: Formerly it was to the contrary the rounded corners which were ignored in case an inset shadow was specified.

• refix>_TeXcolor and <prefix>_TeXextras are new with 6.2.0.

Usefulness is doubtful in the case of *code-block*:

- pre_TeXcolor will influence only the few non-Pygments highlighted tokens; it does color the line numbers, but if one wants to color *only* them one has to go through the fancyvrb interface.
- pre_TeXextras=\footnotesize (as an example) is equivalent to setting 'fvset' key value to
 r'\fvset{fontsize=\footnotesize}'.

Consider these options experimental and that some implementation details may change. For example if the pre_TeXextras LaTeX commands were put by Sphinx in another location it could override the 'fvset' effect, perhaps this is what will be done in a future release.

- Rounded boxes are done using the pict2e¹⁴¹⁴ interface to some basic PDF graphics operations. If this LaTeX package can not be found the build will proceed and render all boxes with straight corners.
- Elliptic corners use the ellipse¹⁴¹⁵ LaTeX package which extends pict2e¹⁴¹⁶. If this LaTeX package can not be found rounded corners will be circular arcs (or straight if pict2e¹⁴¹⁷ is not available).

The following legacy behavior applies:

- For *code-block* or *literalinclude*, padding and border-width and shadow (if any) will go into the margin; the code lines remain at the same place independently of the values of the padding and border-width, except for being shifted vertically of course to not overwrite other text due to the width of the border or external shadow.
- For the other directives, shadows extend horizontally into the page margins, but the border and the extra padding are kept within the text area.
- code-block and literalinclude use the same LaTeX environment and commands and are not separately
 customizable.

7.4 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.

¹⁴¹⁴ https://ctan.org/pkg/pict2e

¹⁴¹⁵ https://ctan.org/pkg/ellipse

¹⁴¹⁶ https://ctan.org/pkg/pict2e

¹⁴¹⁷ https://ctan.org/pkg/pict2e

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:

Name, maps argument #1 to:	
\sphinxstrong	\textbf{#1}
\sphinxcode	\texttt{#1}
\sphinxbfcode	<pre>\textbf{\sphinxcode{#1}}</pre>
\sphinxemail	\textsf{#1}
\sphinxtablecontinued	\textsf{#1}
\sphinxtitleref	\emph{#1}
\sphinxmenuselection	\emph{#1}
\sphinxguilabel	\emph{#1}
\sphinxkeyboard	\sphinxcode{#1}
\sphinxaccelerator	\underline{#1}
\sphinxcrossref	\emph{#1}
\sphinxtermref	\emph{#1}
\sphinxsamedocref	\emph{#1}
\sphinxparam	\emph{#1}
\sphinxtypeparam	1
\sphinxoptional	[#1] with larger brackets, see source

Added in version 1.4.5: Use of \sphinx prefixed macro names to limit possibilities of conflict with LaTeX packages.

Added in version 1.8: \sphinxguilabel

Added in version 3.0: \sphinxkeyboard

Added in version 6.2.0: \sphinxparam, \sphinxsamedocref

Added in version 7.1.0: \sphinxparamcomma which defaults to a comma followed by a space \sphinxparamcommaoneperline. It is sed for one-parameter-per-line signatures (see maximum_signature_line_length) and defaults to \texttt{,}.

Signatures of Python functions are rendered as name<space>(parameters) or name<space>[type

parameters]<space>(parameters) (see PEP 695¹⁴¹⁸) where the length of <space> is set to Opt by default. This can be changed via \setlength{\sphinxsignaturelistskip}{1ex} for instance.

• More text styling:

Name, maps argument #1 to:	
\sphinxstyleindexentry	\texttt{#1}
\sphinxstyleindexextra	(\emph{#1}) (with a space upfront)
\sphinxstyleindexpageref	, \pageref{#1}
\sphinxstyleindexpagemain	\textbf{#1}
\sphinxstyleindexlettergroup	{\Large\sffamily#1}\nopagebreak1mm
\sphinxstyleindexlettergroupDefault	check source, too long for here
\sphinxstyletopictitle	<pre>\textbf{#1}\par\medskip</pre>
\sphinxstylesidebartitle	<pre>\textbf{#1}\par\medskip</pre>
\sphinxstyleothertitle	\textbf{#1}
\sphinxstylesidebarsubtitle	~\\\textbf{#1} \smallskip
\sphinxstyletheadfamily	\sffamily (this one has no argument)
\sphinxstyleemphasis	\emph{#1}
\sphinxstyleliteralemphasis	\emph{\sphinxcode{#1}}
\sphinxstylestrong	\textbf{#1}
\sphinxstyleliteralstrong	\sphinxbfcode{#1}
\sphinxstyleabbreviation	\textsc{#1}
\sphinxstyleliteralintitle	\sphinxcode{#1}
\sphinxstylecodecontinued	{\footnotesize(#1)}}
\sphinxstylecodecontinues	{\footnotesize(#1)}}
\sphinxstylenotetitle	\sphinxdotitlerow{note}{#1}
\sphinxstylehinttitle	\sphinxdotitlerow{hint}{#1}
\sphinxstyleimportanttitle	\sphinxdotitlerow{important}{#1}
\sphinxstyletiptitle	\sphinxdotitlerow{tip}{#1}
\sphinxstylewarningtitle	\sphinxdotitlerow{warning}{#1}
\sphinxstylecautiontitle	\sphinxdotitlerow{caution}{#1}
\sphinxstyleattentiontitle	\sphinxdotitlerow{attention}{#1}
\sphinxstyledangertitle	\sphinxdotitlerow{danger}{#1}
\sphinxstyleerrortitle	\sphinxdotitlerow{error}{#1}
\sphinxstyleseealsotitle	\sphinxdotitlerow{seealso}{#1}
\sphinxstyletodotitle	\sphinxdotitlerow{todo}{#1}
\sphinxstyletopictitle	\sphinxdotitlerow{topic}{#1}
\sphinxstylecontentstitle	\sphinxdotitlerow{contents}{#1}
\sphinxstylesidebartitle	\sphinxdotitlerow{sidebar}{#1}

1 Note

To let this table fit on the page width in PDF output we have lied a bit. For instance, the actual definition of \sphinxstylenotetitle is:

\newcommand\sphinxstylenotetitle[1]%
{\sphinxdotitlerow{note}{\sphinxremovefinalcolon{#1}}}

The same remark applies to all other similar commands associated with admonitions. The topic 1419 , contents 1420 , and sidebar 1421 do not use \sphinxremovefinalcolon as they don't need it.

¹⁴¹⁸ https://peps.python.org/pep-0695/

Added in version 1.5: These macros were formerly hard-coded as non customizable \textt, \emph, etc...

Added in version 1.6: \sphinxstyletheadfamily which defaults to \sffamily and allows multiple paragraphs in header cells of tables.

Added in version 1.6.3: \sphinxstylecodecontinued and \sphinxstylecodecontinues.

Added in version 1.8: \sphinxstyleindexlettergroup, \sphinxstyleindexlettergroupDefault.

Added in version 6.2.0: \sphinxstylenotetitle et al. The #1 is the localized name of the directive, with a final colon. Wrap it as \sphinxremovefinalcolon{#1} if this final colon is to be removed.

Added in version 7.4.0: Added the \sphinxdotitlerowwithicon LaTeX command.

Changed in version 8.1.0: \sphinxdotitlerowwithicon now detects automatically if an icon is associated or not with the rendered element used as first argument.

Added in version 8.1.0: Make \sphinxdotitlerow an alias to \sphinxdotitlerowwithicon.

Added in version 8.1.0: Titles of $topic^{1422}$, contents¹⁴²³, and $sidebar^{1424}$ directives are also styled using **\sphinxdotitlerow** (they have no default icons associated with them).

• \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 \leftlestate \leftlestate \leftlestate \text{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.

Hint

If adding to preamble the loading of tocloft package, also add to preamble $\mbox{renewcommandsphinxtableofcontentshook}\{\}\ \mbox{else it will reset $$\logarter{lossection}$ and $$\logarter{lossection}$ cancelling tocloft customization.}$

• \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.

Added in version 1.8.3.

• \sphinxcite: A wrapper of standard \cite for citation references.

¹⁴¹⁹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

¹⁴²⁰ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents

¹⁴²¹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar

¹⁴²² https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

¹⁴²³ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents

¹⁴²⁴ https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar

The \sphinxbox command

Added in version 6.2.0.

The \sphinxbox[key=value,...]{inline text} command can be used to "box" inline text elements with all the customizability which has been described in *Additional CSS-like 'sphinxsetup' keys*. It is a LaTeX command with one optional argument, which is a comma-separated list of key=value pairs, as for *The sphinxsetup configuration setting*. Here is the complete list of keys. They don't use any prefix.

- · border-width,
- border-top-width, border-right-width, border-bottom-width, border-left-width,
- padding,
- padding-top, padding-right, padding-bottom, padding-left,
- border-radius.
- border-top-left-radius, border-top-right-radius, border-bottom-right-radius,
- box-shadow.
- border-TeXcolor, background-TeXcolor, box-shadow-TeXcolor, TeXcolor,
- TeXextras.
- and addstrut which is a boolean key, i.e. to be used as addstrut=true, or addstrut alone where =true is
 omitted, or addstrut=false.

This last key is specific to \sphinxbox and it means to add a \strut so that heights and depths are equalized across various instances on the same line with varying contents. The default is addstrut=false. The combination addstrut, padding-bottom=0pt, padding-top=1pt is often satisfactory.

Refer to *Additional CSS-like 'sphinxsetup' keys* for important syntax information regarding the other keys. The default configuration uses no shadow, a border-width of **fboxrule**, a padding of **fboxsep**, circular corners with radii **fboxsep** and background and border colors as for the default rendering of code-blocks.

When a \sphinxbox usage is nested within another one, it will ignore the options of the outer one: it first resets all options to their default state as they were prior to applying the outer box options, then it applies its own specific ones.

One can modify these defaults via the command \sphinxboxsetup{key=value,...}. The effect is cumulative, if one uses this command multiple times. Here the options are a mandatory argument so are within curly braces, not square brackets.

Here is some example of use:

A utility \newsphinxbox is provided to create a new boxing macro, say \foo which will act exactly like \sphinxbox but with a given extra configuration:

```
% the specific options to \foo are within brackets
\newsphinxbox[border-radius=0pt, box-shadow=2pt 2pt]{\foo}
% then use this \foo, possibly with some extra options still:
\protected\def\sphinxguilabel#1{\foo{#1}}
\protected\def\sphinxmenuselection#1{\foo[box-shadow-TeXcolor=gray]{#1}}
```

Boxes rendered with \foo obey as the ones using directly \sphinxbox the current configuration as set possibly midway in document via \sphinxboxsetup (from a raw¹⁴²⁵ LaTeX mark-up), the only difference is that they have an initial additional set of default extras.

In the above examples, you can probably use \renewcommand syntax if you prefer it to \protected\def (with [1] in place of #1 then).

Environments

• A figure 1426 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.

Added 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 1427 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 (colors, 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,...

• Environment for the *seealso* directive: sphinxseealso. It takes one argument which will be the localized string See also followed with a colon.

Added in version 6.1.0.

Changed in version 6.2.0: Colon made part of the mark-up rather than being inserted by the environment for coherence with how admonitions are handled generally.

¹⁴²⁵ https://docutils.sourceforge.io/docs/ref/rst/directives.html#raw

 $^{^{1426}\} https://docutils.sourceforge.io/docs/ref/rst/directives.html\#figure$

¹⁴²⁷ https://docutils.sourceforge.io/docs/ref/rst/directives.html#warning

• Environment for the *todo* directive: sphinxtodo. It takes one argument, namely the localization of Todo (with a colon at the end; the default rendering will remove that colon and put the localized string in its own colored title-row).

Added in version 7.4.0.

• The topic 1428, contents 1429 and sidebar 1430 directives are associated with respectively sphinxtopic, sphinxcontents, and sphinxsidebar environments.

Added in version 1.4.2: Former code refactored into an environment allowing page breaks.

Changed in version 1.5: Options shadowsep, shadowsize, shadowrule.

Added in version 8.1.0: Separate environments (all three wrappers around sphinxShadowBox) and separate customizability.

• The literal blocks (via :: or code-block), and literal includes (literalinclude) 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 page breaks. 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.

Added in version 1.5: Options verbatimwithframe, verbatimwrapslines, verbatimsep, verbatimborder.

Added in version 1.6.6: Support for :emphasize-lines: option

Added 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).

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.

Added 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.

• The role¹⁴³¹ directive allows to mark inline text with class arguments. This is handled in LaTeX output via the \DUrole dispatcher command as in Docutils¹⁴³². Object signatures also use \DUrole for some components, with one or two-letters class names as in HTML output.

 $^{^{1428}\} https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic$

https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents

¹⁴³⁰ https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar

¹⁴³¹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#role

 $^{^{1432}\} https://docutils.sourceforge.io/docs/user/latex.html\#custom-interpreted-text-roles$

Changed in version 8.1.0: When multiple classes are injected via a a custom role, the LaTeX output uses nested **\DUrole**'s as in the Docutils documentation 1433. Formerly it used a single **\DUrole** with comma separated classes, making the LaTeX customization more arduous.

• Docutils container 1434 directives are supported in LaTeX output: to let a container class with name foo influence the final PDF via LaTeX, it is only needed to define in the preamble an environment sphinxclassfoo. A simple example would be:

\newenvironment{sphinxclassred}{\color{red}}{}

Currently the class names must contain only ASCII characters and avoid characters special to LaTeX such as $\$. Added in version 4.1.0.

As an experimental feature, Sphinx can use user-defined template file for LaTeX source if you have a file named _templates/latex.tex.jinja in your project.

Additional files longtable.tex.jinja, tabulary.tex.jinja and tabular.tex.jinja can be added to _templates/ to configure some aspects of table rendering (such as the caption position).

Added in version 1.6: currently all template variables are unstable and undocumented.

Changed in version 7.4: Added support for the .jinja file extension, which is preferred. The old file names remain supported. (latex.tex_t, longtable.tex_t, tabulary.tex_t, and tabular.tex_t)

 $^{^{1433}\} https://docutils.sourceforge.io/docs/user/latex.html \# custom-interpreted-text-roles$

¹⁴³⁴ https://docutils.sourceforge.io/docs/ref/rst/directives.html#container

CHAPTER

EIGHT

GET SUPPORT

For questions or to report problems with Sphinx:

- Please verify that your question does not exist on StackOverflow¹⁴³⁵ (with the tag python-sphinx)
- open a topic at the Github discussions 1436 page,
- open an issue at the Github issues 1437 page,
- or join the sphinx-users 1438 mailing list on Google Groups,

Examples of other projects using Sphinx can be found in the examples page.

See also the guide by ReadTheDocs¹⁴³⁹ on how to get started with Sphinx.

There is a translation team in Transifex 1440 of this documentation, thanks to the Sphinx document translators.

¹⁴³⁵ https://stackoverflow.com/questions/tagged/python-sphinx

¹⁴³⁶ https://github.com/sphinx-doc/sphinx/discussions

¹⁴³⁷ https://github.com/sphinx-doc/sphinx/issues

¹⁴³⁸ https://groups.google.com/group/sphinx-users

¹⁴³⁹ https://docs.readthedocs.io/en/stable/intro/getting-started-with-sphinx.html

¹⁴⁴⁰ https://www.transifex.com/sphinx-doc/sphinx-doc/dashboard/

CONTRIBUTE TO SPHINX

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.

9.1 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.

Get help

The Sphinx community maintains a number of mailing lists and IRC channels.

Stack Overflow with tag python-sphinx¹⁴⁴¹

Questions and answers about use and development.

GitHub Discussions Q&A¹⁴⁴²

Question-and-answer style forum for discussions.

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.libera.chat

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 on GitHub.

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. Please also include the output of **sphinx-build --bug-report**.

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.

¹⁴⁴¹ https://stackoverflow.com/questions/tagged/python-sphinx

¹⁴⁴² https://github.com/orgs/sphinx-doc/discussions/categories/q-a

¹⁴⁴³ https://github.com/sphinx-doc/sphinx/issues

Contribute code

The Sphinx source code is managed using Git and is hosted on GitHub¹⁴⁴⁴. 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 opening 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 start a discussion 1445.

These are the basic steps needed to start developing on Sphinx.

- 1. Create an account on GitHub.
- 2. Fork¹⁴⁴⁶ the main Sphinx repository (sphinx-doc/sphinx¹⁴⁴⁷) using the GitHub interface.
- 3. Clone the forked repository to your machine.

```
git clone https://github.com/<USERNAME>/sphinx
cd sphinx
```

4. 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 .
```

5. Create a new working branch. Choose any name you like.

```
git switch -c feature-xyz
```

6. Hack, hack, hack.

Write your code along with tests that shows that the bug was fixed or that the feature works as expected.

7. Add a bullet point to CHANGES.rst 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.'
```

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.

8. Push changes in the branch to your forked repository on GitHub:

```
git push origin feature-xyz
```

- 9. Submit a pull request from your branch to the master branch.
- 10. Wait for a core developer or contributor to review your changes.

¹⁴⁴⁴ https://github.com/sphinx-doc/sphinx

¹⁴⁴⁵ https://github.com/orgs/sphinx-doc/discussions/

¹⁴⁴⁶ https://github.com/sphinx-doc/sphinx/fork

¹⁴⁴⁷ https://github.com/sphinx-doc/sphinx

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.rst 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 as follows:

```
ruff check .
mypy
```

Unit tests

Sphinx is tested using pytest¹⁴⁴⁸ for Python code and Jasmine¹⁴⁴⁹ 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 3.12:

```
tox -e py312
```

• Arguments to **pytest** can be passed via **tox**, e.g., in order to run a particular test:

```
tox -e py312 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
```

Ţip

jasmine requires a Firefox binary to use as a test browser.

On Unix systems, you can check the presence and location of the firefox binary at the command-line by running command -v firefox.

¹⁴⁴⁸ https://docs.pytest.org/en/latest/

¹⁴⁴⁹ https://jasmine.github.io/

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.
- Tests should be quick and only test the relevant components, as we aim that *the test suite should not take more than a minute to run*. In general, avoid using the app fixture and app.build() unless a full integration test is required.

Added in version 1.8: Sphinx also runs JavaScript tests.

Changed in version 1.5.2: Sphinx was switched from nose to pytest.

Contribute documentation

Contributing to documentation involves modifying the source files found in the doc/ folder. To get started, you should first follow *Getting started*, and then take the steps below to work with the documentation.

The following sections describe how to get started with contributing documentation, as well as key aspects of a few different tools that we use.

Build the documentation

To build the documentation, run the following command:

```
sphinx-build -M html ./doc ./build/sphinx --fail-on-warning
```

This will parse the Sphinx documentation's source files and generate HTML for you to preview in build/sphinx/html.

You can also build a **live version of the documentation** that you can preview in the browser. It will detect changes and reload the page any time you make edits. To do so, run the following command:

```
sphinx-autobuild ./doc ./build/sphinx/
```

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 Babel¹⁴⁵⁰ to extract messages and maintain the catalog files. The utils directory contains a helper script, babel_runner.py.

- Use python babel_runner.py extract to update the .pot template.
- Use python babel_runner.py update to update all existing language catalogs in sphinx/locale/*/ LC_MESSAGES with the current messages in the template file.
- Use python babel_runner.py compile to compile the .po files to binary .mo files and .js files.

When an updated .po file is submitted, run python babel_runner.py compile 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 Transifex¹⁴⁵¹. 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

¹⁴⁵⁰ https://babel.pocoo.org/en/latest/

¹⁴⁵¹ https://www.transifex.com/sphinx-doc/sphinx-1/

sphinx/locale and then run tx pull -f -l LANG where LANG is an existing language identifier. It is good practice to run python babel_runner.py update 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 --fresh-env* option.
- Use the *sphinx-build --pdb* 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 1452.

Updating generated files

- JavaScript stemming algorithms in sphinx/search/non-minified-js/*.js are generated using snow-ball by cloning the repository, executing make dist_libstemmer_js and then unpacking the tarball which is generated in dist directory.
 - Minified files in sphinx/search/minified-js/*.js are generated from non-minified ones using **uglifyjs** (installed via npm), with -m option to enable mangling.
- The searchindex.js files found in the tests/js/fixtures/* directories are generated by using the standard Sphinx HTML builder on the corresponding input projects found in tests/js/roots/*. The fixtures provide test data used by the Sphinx JavaScript unit tests, and can be regenerated by running the utils/generate_js_fixtures.py script.

9.2 Sphinx's release process

Versionina

Sphinx adheres to **PEP 440**¹⁴⁵⁴ versions, with a major.minor.micro scheme for the *release segment* (e.g. 1.2.3). The major, minor, and micro version parts should be altered as follows:

- The major version part should be incremented for incompatible behavior change and public API updates.
- The minor version part should be incremented for most releases of Sphinx, where backwards-compatibility of API and features are preserves.
- The micro version part should only be incremented for urgent bugfix-only releases.

When the major version part is incremented, the minor and micro version parts must be set to 0. When the minor version part is incremented, the micro version part must be set to 0.

New major versions should come with a beta-testing period before the final release.

¹⁴⁵² https://docutils.sourceforge.io/docs/user/config.html

¹⁴⁵³ https://github.com/snowballstem/snowball

¹⁴⁵⁴ https://peps.python.org/pep-0440/

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 PendingDeprecationWarning¹⁴⁵⁵, i.e. it will not get displayed by default.
- Sphinx 3.x will still contain the backwards-compatible replica, but RemovedInSphinx40Warning will be a subclass of DeprecationWarning¹⁴⁵⁶ 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 PYTHONWARNINGS¹⁴⁵⁷ 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 warnings¹⁴⁵⁸) for more details.

¹⁴⁵⁵ https://docs.python.org/3/library/exceptions.html#PendingDeprecationWarning

¹⁴⁵⁶ https://docs.python.org/3/library/exceptions.html#DeprecationWarning

¹⁴⁵⁷ https://docs.python.org/3/using/cmdline.html#envvar-PYTHONWARNINGS

¹⁴⁵⁸ https://docs.python.org/3/library/warnings.html#describing-warning-filters

Python version support policy

Sphinx supports at all minor versions of Python released in the past 3 years from the anticipated release date with a minimum of 3 minor versions of Python. This policy is derived from SPEC 0^{1459} , a scientific Python domain standard.

For example, a version of Sphinx released in May 2025 would support Python 3.11, 3.12, and 3.13.

This is a summary table with the current policy:

Date	Python
05 Oct 2023	3.10+
04 Oct 2024	3.11+
24 Oct 2025	3.12+
01 Oct 2026	3.13+
01 Oct 2027	3.14+

Release procedures

The release procedures are listed in utils/release-checklist.rst.

9.3 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.rst entry.

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.

¹⁴⁵⁹ https://scientific-python.org/specs/spec-0000/

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*.

9.4 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.

SPHINX FAQ

This is a list of Frequently Asked Questions about Sphinx. Feel free to suggest new entries!

10.1 How do I...

... create PDF files without LaTeX?

rinohtype¹⁴⁶⁰ 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 *Tutorials*.

... convert from my existing docs using MoinMoin markup?

The easiest way is to convert to xhtml, then convert xhtml to reStructuredText¹⁴⁶¹. 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-contrib¹⁴⁶² repository.

10.2 Using Sphinx with...

Read the Docs

Read the Docs¹⁴⁶³ 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 Started¹⁴⁶⁴ guide is a good place to start.

Epydoc

There's a third-party extension providing an api role¹⁴⁶⁵ which refers to Epydoc's API docs for a given identifier.

¹⁴⁶⁰ https://github.com/brechtm/rinohtype

¹⁴⁶¹ https://docutils.sourceforge.io/sandbox/xhtml2rest/xhtml2rest.py

¹⁴⁶² https://github.com/sphinx-contrib/

¹⁴⁶³ https://readthedocs.org

 $^{^{1464}\} https://docs.readthedocs.io/en/stable/intro/getting-started-with-sphinx.html$

¹⁴⁶⁵ https://git.savannah.gnu.org/cgit/kenozooid.git/tree/doc/extapi.py

Doxygen

Michael Jones has developed a reStructuredText/Sphinx bridge to doxygen called breathe 1466.

SCons

Glenn Hutchings has written a SCons build script to build Sphinx documentation; it is hosted here: https://bitbucket-archive.softwareheritage.org/projects/zo/zondo/sphinx-scons.html

PyPI

Jannis Leidel wrote a setuptools command¹⁴⁶⁷ 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 sphinx-wiki¹⁴⁶⁸, 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' : 'https://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:

 $^{^{1466}\} https://github.com/breathe-doc/breathe/tree/master$

¹⁴⁶⁷ https://pypi.org/project/Sphinx-PyPI-upload/

¹⁴⁶⁸ https://bitbucket-archive.softwareheritage.org/projects/ke/kevindunn/sphinx-wiki.html

3. Add searchbox.html to the html_sidebars configuration value.

10.3 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 ¹⁴⁶⁹ is a text processing system for converting plain text documentation into other, richer formats. As noted in the docutils documentation ¹⁴⁷⁰, 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 - reStructuredText¹⁴⁷¹ - 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 tools¹⁴⁷², 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.

10.4 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

¹⁴⁶⁹ https://docutils.sourceforge.io/

¹⁴⁷⁰ https://docutils.sourceforge.io/docs/dev/hacking.html

¹⁴⁷¹ https://docutils.sourceforge.io/rst.html

¹⁴⁷² https://docutils.sourceforge.io/docs/user/tools.html

characters on a line. You may even use $FontForge^{1473}$ 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 Epubcheck¹⁴⁷⁴, Calibre¹⁴⁷⁵, FBreader¹⁴⁷⁶ (although it does not render the CSS), and Bookworm¹⁴⁷⁷. For Bookworm, you can download the source from https://code.google.com/archive/p/threepress and run your own local server.
- 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 <code>epub_post_files</code> 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.
- kindlegen¹⁴⁷⁸ 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:

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.
```

¹⁴⁷³ https://fontforge.github.io/

¹⁴⁷⁴ https://github.com/IDPF/epubcheck

¹⁴⁷⁵ https://calibre-ebook.com/

¹⁴⁷⁶ https://fbreader.org/

¹⁴⁷⁷ https://www.oreilly.com/bookworm/index.html

¹⁴⁷⁸ https://www.amazon.com/gp/feature.html?docId=1000765211

10.5 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).

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
```

One can disable generation of the inline references in a document with texinfo_cross_references. That makes an info file more readable with stand-alone reader (info).

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:

10.5. Texinfo info 449

info:Texinfo#makeinfo_options

CHAPTER

ELEVEN

SPHINX AUTHORS

11.1 Maintainers

Listed alphabetically in forename, surname order

- Adam Turner <@AA-Turner>
- Bénédikt Tran <@picnixz>
- Chris Sewell <@chrisjsewell>
- François Freitag <@francoisfreitag>
- Jakob Lykke Andersen <@jakobandersen>
- Jean-François Burnol <@jfbu>
- Stephen Finucane <@stephenfin>
- Takayuki Shimizukawa <@shimizukawa>
- Takeshi Komiya <@tk0miya>

11.2 Contributors

Listed alphabetically in forename, surname order

- Adrián Chaves (Gallaecio) coverage builder improvements
- Alastair Houghton Apple Help builder
- Alexander Todorov inheritance_diagram tests and improvements
- Andi Albrecht agogo theme
- Antonio Valentino qthelp builder, docstring inheritance
- Antti Kaihola doctest extension (skipif option)
- Barry Warsaw setup command improvements
- Ben Egan Napoleon improvements
- Benjamin Peterson unittests
- Blaise Laflamme pyramid theme
- Bruce Mitchener Minor epub improvement
- Buck Evan dummy builder
- Charles Duffy original graphviz extension

- Chris Lamb reproducibility fixes
- Christopher Perkins autosummary integration
- Dan MacKinlay metadata fixes
- Daniel Bültmann todo extension
- Daniel Neuhäuser JavaScript domain, Python 3 support (GSOC)
- Daniel Pizetta inheritance diagram improvements
- Dave Kuhlman original LaTeX writer
- Doug Hellmann graphviz improvements
- Eric N. Vander Weele autodoc improvements
- Etienne Desautels apidoc module
- Ezio Melotti collapsible sidebar JavaScript
- Filip Vavera napoleon todo directive
- Glenn Matthews python domain signature improvements
- Gregory Szorc performance improvements
- Henrique Bastos SVG support for graphviz extension
- Hernan Grecco search improvements
- Hong Xu svg support in imgmath extension and various bug fixes
- Horst Gutmann internationalization support
- Hugo van Kemenade support FORCE_COLOR and NO_COLOR
- Ian Lee quickstart improvements
- Jacob Mason websupport library (GSOC project)
- Jeppe Pihl literalinclude improvements
- Joel Wurtz cellspanning support in LaTeX
- John Waltman Texinfo builder
- Josip Dzolonga coverage builder
- Julien Palard Colspan and rowspan in text builder
- Kevin Dunn MathJax extension
- KINEBUCHI Tomohiko typing Sphinx as well as docutils
- Kurt McKee documentation updates
- Lars Hupfeldt Nielsen OpenSSL FIPS mode md5 bug fix
- Łukasz Langa partial support for autodoc
- Marco Buttu doctest extension (pyversion option)
- Martin Hans autodoc improvements
- Martin Larralde additional napoleon admonitions
- Martin Mahner nature theme
- Matthew Fernandez todo extension fix

- Matthew Woodcraft text output improvements
- Michael Droettboom inheritance_diagram extension
- Michael Wilson Intersphinx HTTP basic auth support
- Nathan Damon bugfix in validation of static paths in html builders
- Pauli Virtanen autodoc improvements, autosummary extension
- A. Rafey Khan improved intersphinx typing
- Rob Ruana napoleon extension
- Robert Lehmann gettext builder (GSOC project)
- Roland Meister epub builder
- Sebastian Wiesner image handling, distutils support
- Stefan Seefeld toctree improvements
- Stefan van der Walt autosummary extension
- T. Powers HTML output improvements
- Taku Shimizu epub3 builder
- Thomas Lamb linkcheck builder
- Thomas Waldmann apidoc module fixes
- Tim Hoffmann theme improvements
- Vince Salvino JavaScript search improvements
- Will Maier directory HTML builder
- Zac Hatfield-Dodds doctest reporting improvements, intersphinx performance

11.3 Former maintainers

Listed alphabetically in forename, surname order

Former maintainers are those who haven't committed in the last two years. Those on the list below may become active maintainers again at any time.

- Armin Ronacher <@mitsuhiko>
- Daniel Neuhäuser <@DasIch>
- Georg Brandl <georg@python.org>
- Rob Ruana <@RobRuana>
- Robert Lehmann <@lehmannro>
- Timotheus Kampik <@TimKam>
- Yoshiki Shibukawa <@shibukawa>

Many thanks for all contributions!

COMMAND-LINE TOOLS

These are the applications provided as part of Sphinx.

12.1 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 directory under source directory.

--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).

-1 LANGUAGE, --language=LANGUAGE

Document language. (see language).

--suffix=SUFFIX

Source file suffix. (see *source_suffix*).

--master=MASTER

Master document name. (see root_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.

Changed in version 7.3: Support for disabling the make-mode will be removed in Sphinx 8.

Removed in version 8.0: The --no-use-make-mode option. The --use-make-mode now has no effect.

--makefile, --no-makefile

Create (or not create) makefile.

--batchfile, --no-batchfile

Create (or not create) batchfile

Project templating

Added 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 quick-start. Following Jinja2 template files are allowed:

- root_doc.rst.jinja
- conf.py.jinja
- Makefile.jinja
- Makefile.new.jinja
- make.bat.jinja
- make.bat.new.jinja

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. For a list of available builders, refer to *Builders*.

By default, everything that is outdated is built. Output only for selected files can be built by specifying individual filenames.

Options

-M buildername

Select a builder, using the *make-mode*. See *Builders* for a list of all of Sphinx's built-in builders. Extensions can add their own builders.

Important

Sphinx only recognizes the -M option if it is used first, along with the source and output directories, before any other options are passed. For example:

sphinx-build -M html ./source ./build --fail-on-warning

The *make-mode* provides the same build functionality as a default *Makefile or Make.bat*, and provides the following additional build pipelines:

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**.

1 Note

The default output directory locations when using *make-mode* differ from the defaults when using -b.

- doctrees are saved to <outputdir>/doctrees
- output files are saved to <outputdir>/<builder name>

Added in version 1.2.1.

-b buildername, --builder buildername

Selects a builder.

See Builders for a list of all of Sphinx's built-in builders. Extensions can add their own builders.

Changed in version 7.3: Add --builder long option.

-a, --write-all

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.)

1 Note

This option does not re-read source files. To read and re-process every file, use --fresh-env instead.

Changed in version 7.3: Add --write-all long option.

-E, --fresh-env

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.

Changed in version 7.3: Add -- fresh-env long option.

-t tag, --tag tag

Define the tag *tag*. This is relevant for *only* directives that include their content only if certain tags are set. See *including content based on tags* for further detail.

Added in version 0.6.

Changed in version 7.3: Add --tag long option.

-d path, --doctree-dir 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).

Changed in version 7.3: Add --doctree-dir long option.

-j N, --jobs N

Distribute the build over N processes in parallel, to make building on multiprocessor machines more effective. This feature only works on systems supporting "fork". Windows is not supported. 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. Defaults to 1.

Added in version 1.2: This option should be considered *experimental*.

Changed in version 1.7: Support auto argument.

Changed in version 6.2: Add -- jobs long option.

-c path, --config-dir 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.

Added in version 0.3.

Changed in version 7.3: Add --config-dir long option.

-C, --isolated

Don't look for a configuration file; only take options via the --define option.

Added in version 0.5.

Changed in version 7.3: Add --isolated long option.

-D setting=value, --define 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.

Changed in version 7.3: Add --define long option.

-A name=value, --html-define name=value

Make the *name* assigned to *value* in the HTML templates.

Added in version 0.5.

Changed in version 7.3: Add --html-define long option.

-n, --nitpicky

Run in nitpicky 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".

Changed in version 7.3: Add --nitpicky long option.

-N. --no-color

Do not emit colored output.

Changed in version 1.6: Add --no-color long option.

--color

Emit colored output. Auto-detected by default.

Added in version 1.6.

-v, --verbose

Increase verbosity (log-level). This option can be given up to three times to get more debug logging output. It implies -T.

Added in version 1.2.

Changed in version 7.3: Add --verbose long option.

-q, --quiet

Do not output anything on standard output, only write warnings and errors to standard error.

Changed in version 7.3: Add --quiet long option.

-Q, --silent

Do not output anything on standard output, also suppress warnings. Only errors are written to standard error.

Changed in version 7.3: Add --silent long option.

-w file, --warning-file file

Write warnings (and errors) to the given file, in addition to standard error.

Changed in version 7.3: ANSI control sequences are stripped when writing to file.

Changed in version 7.3: Add --warning-file long option.

-W, --fail-on-warning

Turn warnings into errors. This means that **sphinx-build** exits with exit status 1 if any warnings are generated during the build.

Changed in version 7.3: Add --fail-on-warning long option.

Changed in version 8.1: **sphinx-build** no longer exits on the first warning, but instead runs the entire build and exits with exit status 1 if any warnings were generated. This behaviour was previously enabled with *--keep-going*.

--keep-going

From Sphinx 8.1, --keep-going is always enabled. Previously, it was only applicable whilst using --fail-on-warning, which by default exited **sphinx-build** on the first warning. Using --keep-going runs **!sphinx-build** to completion and exits with exit status 1 if errors are encountered.

Added in version 1.8.

Changed in version 8.1: **sphinx-build** no longer exits on the first warning, meaning that in effect --fail-on-warning is always enabled. The option is retained for compatibility, but may be removed at some later date.

-T, --show-traceback

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.

Added in version 1.2.

Changed in version 7.3: Add --show-traceback long option.

-P, --pdb

 $(Useful\ for\ debugging\ only.)\ Run\ the\ Python\ debugger,\ pdb^{1479},\ if\ an\ unhandled\ exception\ occurs\ while\ building.$

Changed in version 7.3: Add --pdb long option.

--exception-on-warning

Raise an exception when a warning is emitted during the build. This can be useful in combination with --pdb to debug warnings.

Added in version 8.1.

-h, --help, --version

Display usage summary or Sphinx version.

Added 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.

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'.

¹⁴⁷⁹ https://docs.python.org/3/library/pdb.html#module-pdb



1 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').

NO_COLOR

When set (regardless of value), sphinx-build will not use color in terminal output. NO_COLOR takes precedence over FORCE_COLOR. See no-color.org 1480 for other libraries supporting this community standard.

Added in version 4.5.0.

FORCE_COLOR

When set (regardless of value), sphinx-build will use color in terminal output. NO_COLOR takes precedence over FORCE_COLOR.

Added in version 4.5.0.

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)

12.2 Additional applications

sphinx-apidoc

Synopsis

sphinx-apidoc [OPTIONS] -o <OUTPUT_PATH> <MODULE_PATH> [EXCLUDE_PATTERN ...]

¹⁴⁸⁰ https://no-color.org/

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_PATTERNs* given are fnmatch-style¹⁴⁸¹ 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.

-1, --follow-links

Follow symbolic links. Defaults to False.

-n, --dry-run

Do not create or remove any files.

-s <suffix>

Suffix for the source files generated. Defaults to rst.

-d <MAXDEPTH>

Maximum depth for the generated table of contents file. Defaults to 4.

--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.

--remove-old

Remove existing files in the output directory that are not created anymore. Not compatible with --full.

-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.

Added in version 1.2.

¹⁴⁸¹ https://docs.python.org/3/library/fnmatch.html

-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.

Added in version 1.2.

--implicit-namespaces

By default sphinx-apidoc processes sys.path searching for modules only. Python 3.3 introduced **PEP 420**¹⁴⁸² 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

Added 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.jinja
- package.rst.jinja
- toc.rst.jinja
- root_doc.rst.jinja
- conf.py.jinja
- Makefile.jinja
- Makefile.new.jinja
- make.bat.jinja
- make.bat.new.jinja

¹⁴⁸² https://peps.python.org/pep-0420/

In detail, please refer the system template files Sphinx provides. (sphinx/templates/apidoc and sphinx/templates/quickstart)

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 fnmatch¹⁴⁸³-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.

-a, --respect-module-all

Document exactly the members in a module's __all__ attribute.

--remove-old

Remove existing files in the output directory that are not generated anymore.

Example

Given the following directory structure:



(continues on next page)

¹⁴⁸³ https://docs.python.org/3/library/fnmatch.html#module-fnmatch

(continued from previous page)

and assuming docs/index.rst contained the following:

```
Modules
======

.. autosummary::
    :toctree: modules

    foobar.foo
    foobar.bar
    foobar.bar
```

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.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

THIRTEEN

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:

```
.. directive-name:: argument ...
:option: value

Content of the directive.
```

See *Directives* for more information.

document name

Since reStructuredText 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 *directives* and *roles*) to describe and link to *objects* 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

root document

The document that contains the root toctree directive.

object

The basic building block of Sphinx documentation. Every "object directive" (e.g. *py: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

FOURTEEN

CHANGELOG

14.1 Release 8.1.2 (in development)

Bugs fixed

14.2 Release 8.1.1 (released Oct 11, 2024)

Bugs fixed

- #13006: Use the preferred https://www.cve.org/ URL for the :cve: role. Patch by Hugo van Kemenade.
- #13007: LaTeX: Improve resiliency when the required fontawesome or fontawesome5 packages are not installed. Patch by Jean-François B.

14.3 Release 8.1.0 (released Oct 10, 2024)

Dependencies

- #12756: Add lower-bounds to the sphinxcontrib-* dependencies. Patch by Adam Turner.
- #12833: Update the LaTeX parskip package from 2001 to 2018. Patch by Jean-François B.

Incompatible changes

- #12763: Remove unused internal class sphinx.util.Tee. Patch by Adam Turner.
- #12822: LaTeX: for Unicode engines, the 'fvset' default is changed to '\\fvset\fontsize=auto\' from '\\ fvset\fontsize=\\small\'. Code-blocks are unchanged as FreeMono is now loaded with Scale=0.9. An adjustment to existing projects is needed only if they used a custom 'fontpkg' configuration and did not set 'fvset'. Patch by Jean-François B.
- #12875: Disable smartquotes for languages: zh_CN and zh_TW by default. Patch by A. Rafey Khan.

Deprecated

- #12762: Deprecate sphinx.util.import_object. Use importlib.import_module()¹⁴⁸⁴ instead. Patch by Adam Turner.
- #12766: Deprecate sphinx.util.FilenameUniqDict and sphinx.util.DownloadFiles. Patch by Adam Turner

 $^{^{1484}\} https://docs.python.org/3/library/importlib.html\#importlib.import_module$

Features added

- #11328: Mention evaluation of templated content during production of static output files. Patch by James Addison.
- #12704: LaTeX: make contents¹⁴⁸⁵, topic¹⁴⁸⁶, and sidebar¹⁴⁸⁷ directives separately customizable for PDF output. Patch by Jean-François B. and Bénédikt Tran.
- #12474: Support type-dependent search result highlighting via CSS. Patch by Tim Hoffmann.
- #12652: LaTeX: Add math_numsep support to latex builder. Patch by Thomas Fanning and Jean-François B.
- #12743: No longer exit on the first warning when --fail-on-warning is used. Instead, exit with a non-zero status if any warnings were generated during the build. Patch by Adam Turner.
- #12743: Add *sphinx-build --exception-on-warning*, to raise an exception when warnings are emitted during the build. Patch by Adam Turner and Jeremy Maitin-Shepard.
- #12907: Add html_last_updated_use_utc to allow using universal time (GMT/UTC) instead of local time for the date-time supplied to html_last_updated_fmt. Patch by Adam Turner.
- #12910: Copyright entries now support the '%Y' placeholder to substitute the current year. This is helpful for reducing the reliance on Python modules such as time¹⁴⁸⁸ or datetime¹⁴⁸⁹ in conf.py. See *the docs* for further detail. Patch by Adam Turner.
- #11781: Add roles for referencing CVEs (:cve:) and CWEs (:cwe:). Patch by Hugo van Kemenade.
- #11809: Improve the formatting for RFC section anchors. Patch by Jakub Stasiak and Adam Turner.
- #12852: Support a *Builder.supported_linkcode* attribute for builders to enable use of *sphinx.ext.linkcode*-generated references. Patch by James Knight.
- #12949: Print configuration options that differ from the pickled environment. This can be helpful in diagnosing the cause of a full rebuild. Patch by Adam Turner.

Bugs fixed

- #12514: intersphinx: fix the meaning of a negative value for *intersphinx_cache_limit*. Patch by Shengyu Zhang.
- #12722: LaTeX: avoid TeX reporting Overfull \hbox from too long strings in a codeline when the problem has actually been solved thanks to *verbatimforcewraps*. Patch by Jean-François B.
- #12730: The UnreferencedFootnotesDetector transform has been improved to more consistently detect unreferenced footnotes. Note, the priority of the transform has been changed from 200 to 622, so that it now runs after the docutils Footnotes resolution transform. Patch by Chris Sewell.
- #12778: LaTeX: let 'sphinxsetup' div.topic_box-shadow key if used with only one dimension set both x-offset and y-offset as per documentation. Patch by Jean-François B.
- #12587: Do not warn when potential ambiguity detected during Intersphinx resolution occurs due to duplicate targets that differ case-insensitively. Patch by James Addison.
- #12639: Fix singular and plural search results text. Patch by Hugo van Kemenade.
- #12645: Correctly support custom gettext output templates. Patch by Jeremy Bowman.
- #12717: LaTeX: let -q (quiet) option for **sphinx-build** -M latexpdf or make latexpdf (0=-q) get passed to latexmk. Let -Q (silent) apply as well to the PDF build phase. Patch by Jean-François B.

¹⁴⁸⁵ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents

¹⁴⁸⁶ https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

¹⁴⁸⁷ https://docutils.sourceforge.io/docs/ref/rst/directives.html#sidebar

¹⁴⁸⁸ https://docs.python.org/3/library/time.html#module-time

¹⁴⁸⁹ https://docs.python.org/3/library/datetime.html#module-datetime

- #12744: LaTeX: Classes injected by a custom interpreted text role now give rise to nested \DUrole's, rather than a single one with comma separated classes. Patch by Jean-François B.
- #12831: LaTeX: avoid large voids sometimes occurring at page bottoms. Patch by Jean-François B.
- #11970, #12551: singlehtml builder: make target URIs to be same-document references in the sense of **RFC 3986**, §4.4¹⁴⁹⁰, e.g., index.html#foo becomes #foo. (note: continuation of a partial fix added in Sphinx 7.3.0) Patch by James Addison (with reference to prior work by Eric Norige).
- #12735: Fix PEP 695¹⁴⁹¹ generic classes LaTeX output formatting. Patch by Jean-François B. and Bénédikt Tran.
- #12782: intersphinx: fix double forward slashes when generating the inventory file URL (user-defined base URL
 of an intersphinx project are left untouched even if they end with double forward slashes). Patch by Bénédikt
 Tran.
- #12796: Enable parallel reading if requested, even if there are fewer than 6 documents. Patch by Matthias Geier.
- #12844: Restore support for :noindex: for the <code>js:module</code> and <code>py:module</code> directives. Patch by Stephen Finucane.
- #12916: Restore support for custom templates named with the legacy _t suffix during apidoc RST rendering (regression in 7.4.0). Patch by James Addison.
- #12451: Only substitute copyright notice years with values from SOURCE_DATE_EPOCH for entries that match the current system clock year, and disallow substitution of future years. Patch by James Addison and Adam Turner.
- #12905: intersphinx: fix flipped use of *intersphinx_cache_limit*, which always kept the cache for positive values, and always refreshed it for negative ones. Patch by Nico Madysa.
- #12888: Add a warning when document is included in multiple toctrees and ensure deterministic resolution of global toctree in parallel builds by choosing the lexicographically greatest parent document. Patch by A. Rafey Khan
- #12995: Significantly improve performance when building the search index for Chinese languages. Patch by Adam Turner.
- #12767: Builder.write() is typed as final, meaning that the write-started event may be relied upon by extensions. A new Builder.write_documents() method has been added to control how documents are written. This is intended for builders that do not output a file for each document. Patch by Adam Turner.

Testing

• #12141: Migrate from the deprecated karma JavaScript test framework to the actively-maintained jasmine framework. Test coverage is unaffected. Patch by James Addison.

14.4 Prior releases

Sphinx 8.0

Release 8.0.2 (released Jul 30, 2024)

Bugs fixed

• Fix the pygments.Formatter.__class_getitem__ patch. Patch by Adam Turner.

¹⁴⁹⁰ https://datatracker.ietf.org/doc/html/rfc3986.html#section-4.4

¹⁴⁹¹ https://peps.python.org/pep-0695/

Release 8.0.1 (released Jul 30, 2024)

Bugs fixed

• Patch pygments.Formatter.__class_getitem__ in Pygments 2.17. Patch by Adam Turner.

Release 8.0.0 (released Jul 29, 2024)

Dependencies

• #12633: Drop Python 3.9 support.

Incompatible changes

- Remove deprecated functions from sphinx.util:
 - Removed sphinx.util.path_stabilize (use sphinx.util.osutil.path_stabilize).
 - Removed sphinx.util.display_chunk (use sphinx.util.display.display_chunk).
 - Removed sphinx.util.status_iterator (use sphinx.util.display.status_iterator).
 - Removed sphinx.util.SkipProgressMessage (use sphinx.util.display. SkipProgressMessage).
 - Removed sphinx.util.progress_message (use sphinx.util.display.progress_message).
 - Removed sphinx.util.epoch_to_rfc1123 (use sphinx.http_date.epoch_to_rfc1123).
 - Removed sphinx.util.rfc1123_to_epoch (use sphinx.http_date.rfc1123_to_epoch).
 - Removed sphinx.util.save_traceback (use sphinx.exceptions.save_traceback).
 - Removed sphinx.util.format_exception_cut_frames (use sphinx.exceptions. format_exception_cut_frames).
 - Removed sphinx.util.xmlname_checker (use sphinx.builders.epub3._XML_NAME_PATTERN).

Patch by Adam Turner.

- Removed sphinx.util.osutil.cd() (use contextlib.chdir() ¹⁴⁹²). Patch by Adam Turner.
- Removed sphinx.util.typing.stringify() (use sphinx.util.typing.stringify_annotation()). Patch by Adam Turner.
- #12593: Raise an error for invalid html_sidebars values. Patch by Adam Turner.
- #12593: Raise an error in Theme.get_config() for invalid sections. Patch by Adam Turner.
- #11693: Remove support for old-style Makefile and make.bat output in sphinx-quickstart.
- #11693: Remove the --no-use-make-mode, -M, --use-make-mode, and -m options from sphinx-quickstart. Patch by Adam Turner.
- Removed the tuple interface to sphinx.ext.autodoc.ObjectMember. Patch by Adam Turner.
- #12630: Sphinx 8 makes two changes to the linkcheck configuration defaults:
 - linkcheck_allow_unauthorized is now False by default.
 - linkcheck_report_timeouts_as_broken is now False by default.

Patch by James Addison.

• #12597: Change the default of show_warning_types from False to True. Patch by Chris Sewell.

¹⁴⁹² https://docs.python.org/3/library/contextlib.html#contextlib.chdir

- #12083: Remove support for the old (2008–2010) Sphinx 0.5 and Sphinx 0.6 intersphinx_mapping format. Patch by Bénédikt Tran and Adam Turner.
- #12096: Do not overwrite user-supplied files when copying assets unless forced with force=True. Patch by Adam Turner.
- #12646: Remove sphinx.util.inspect.isNewType(). Use isinstance(obj, typing.NewType) instead on Python 3.10 and newer. Patch by Adam Turner.
- Remove the long-deprecated (since Sphinx 2) alias to VersionChange in sphinx.directives.other (Deprecated since Sphinx 2). Use sphinx.domains.changeset.VersionChange directly. Patch by Adam Turner.

Deprecated

- #12643: Renamed sphinx.ext.intersphinx.normalize_intersphinx_mapping to sphinx.ext. intersphinx.validate_intersphinx_mapping. The old name will be removed in Sphinx 10. Patch by Adam Turner.
- #12650, #12686, #12690: Extend the deprecation for string methods on Path¹⁴⁹³ objects to Sphinx 9. Use os. fspath()¹⁴⁹⁴ to convert Path¹⁴⁹⁵ objects to strings, or Path¹⁴⁹⁶'s methods to work with path objects. Patch by Adam Turner.

Sphinx 7.4

Release 7.4.7 (released Jul 20, 2024)

Bugs fixed

- #12096: Warn when files are overwritten in the build directory. Patch by Adam Turner and Bénédikt Tran.
- #12620: Ensure that old-style object description options are respected. Patch by Adam Turner.
- #12601, #12625: Support callable objects in Annotated type metadata in the Python domain. Patch by Adam Turner.
- #12601, #12622: Resolve Annotated warnings with sphinx.ext.autodoc, especially when using dataclasses 1497 as type metadata. Patch by Adam Turner.
- #12589, #12626: autosummary: Fix warnings with autolink. Patch by Adam Turner.

Release 7.4.6 (released Jul 18, 2024)

Bugs fixed

- #12589, #9743, #12609: autosummary: Do not add the package prefix when generating autosummary directives for modules within a package. Patch by Adam Turner.
- #12613: Reduce log severity for ambiguity detection during inventory loading. Patch by James Addison.

Release 7.4.5 (released Jul 16, 2024)

Bugs fixed

• #12593, #12600: Revert coercing the type of selected httml_sidebars values to a list. Log an error message when string values are detected. Patch by Adam Turner.

 $^{^{1493}\} https://docs.python.org/3/library/pathlib.html\#pathlib.Path$

https://docs.python.org/3/library/os.html#os.fspath

¹⁴⁹⁵ https://docs.python.org/3/library/pathlib.html#pathlib.Path

¹⁴⁹⁶ https://docs.python.org/3/library/pathlib.html#pathlib.Path

¹⁴⁹⁷ https://docs.python.org/3/library/dataclasses.html#module-dataclasses

• #12594: LaTeX: since 7.4.0, *seealso* and other "light" admonitions now break PDF builds if they contain a figure ¹⁴⁹⁸ directive; and also if they are contained in a table cell (rendered by tabulary). Patch by Jean-François B.

Release 7.4.4 (released Jul 15, 2024)

Bugs fixed

• #12585, #12586: Do not warn when an intersphinx inventory contains case-insensitively ambiguous duplicate items. Patch by James Addison.

Release 7.4.3 (released Jul 15, 2024)

Bugs fixed

• #12582: Restore support for list-styled *source_suffix* values with extensions that register parsers. Patch by Adam Turner.

Release 7.4.2 (released Jul 15, 2024)

Bugs fixed

• #12580, #12583: Resolve failures with the C domain on incremental builds with Sphinx 7.3.7 and earlier. Patch by Adam Turner.

Release 7.4.1 (released Jul 15, 2024)

Bugs fixed

- Fix invalid HTML when a rubric node with invalid heading-level is used. Patch by Adam Turner.
- #12579, #12581: Restore support for typing. ParamSpec in autodoc. Patch by Adam Turner.

Release 7.4.0 (released Jul 15, 2024)

Dependencies

- #12555: Drop Docutils 0.18.1 and Docutils 0.19 support. Patch by Adam Turner.
- LaTeX: the xcolor package is now required (but is for example part of Ubuntu texlive-latex-recommended which has always been required).
- LaTeX: the fontawesome5 LaTeX package is needed for the default choices of icons now used in admonition titles in PDF output; but if unavailable the PDF build will simply silently omit rendering such icons. Check the documentation of the iconpackage key of 'sphinxsetup' for more.

Deprecated

• LaTeX: the sphinxlightbox environment is not used anymore, all types of admonitions use (by default) only sphinxheavybox.

Features added

• #11165: Support the officially recommended 1499 . jinja suffix for template files. Patch by James Addison and Adam Turner

¹⁴⁹⁸ https://docutils.sourceforge.io/docs/ref/rst/directives.html#figure

¹⁴⁹⁹ https://jinja.palletsprojects.com/en/latest/templates/#template-file-extension

- #12325: Flatten Union[Literal[T], Literal[U], ...] to Literal[T, U, ...] when turning annotations into strings. Patch by Adam Turner.
- #12319: sphinx.ext.extlinks: Add extlink-{name} CSS class to links. Patch by Hugo van Kemenade.
- #12387: Improve CLI progress message, when copying assets. Patch by INADA Nakoi and Bénédikt Tran.
- #12361: Add BuildEnvironment.parser. Patch by Chris Sewell.
- #12358: Add Sphinx. fresh_env_used. Patch by Chris Sewell.
- #12329: Add detection of ambiguous std:label and std:term references during loading and resolution of Intersphinx targets. Patch by James Addison.
- #12422: Do not duplicate "navigation" in aria-label of built-in themes. Patch by Thomas Weißschuh
- #12421: Include project name in logo_alt of built-in themes. Patch by Thomas Weißschuh
- #12448: Add *sphinx-apidoc --remove-old* option. Patch by Chris Sewell.
- #12456: Add *sphinx-autogen --remove-old* option. Patch by Chris Sewell.
- #12479: Add warning subtype toc.no_title. Patch by Ondřej Navrátil.
- #12492: Add helper methods for parsing reStructuredText content into nodes from within a directive.
 - parse_content_to_nodes() parses the directive's content and returns a list of Docutils nodes.
 - parse_text_to_nodes() parses the provided text and returns a list of Docutils nodes.
 - parse_inline() parses the provided text into inline elements and text nodes.

Patch by Adam Turner.

- #12258: Support typing_extensions.Unpack Patch by Bénédikt Tran and Adam Turner.
- #12524: Add a class option to the *toctree* directive. Patch by Tim Hoffmann.
- #12536: Add the *confval* directive. Patch by Adam Turner.
- #12537: c_id_attributes, c_paren_attributes, cpp_id_attributes, and cpp_paren_attributes can now be a tuple of strings. c_extra_keywords, gettext_additional_targets, html_domain_indices, latex_domain_indices, and texinfo_domain_indices, can now be a set of strings. Patch by Adam Turner.
- #12523: Added configuration option, *math_numsep*, to define the separator for math numbering. Patch by Thomas Fanning
- #11592: Add *coverage_modules* to the coverage builder to allow explicitly specifying which modules should be documented. Patch by Stephen Finucane.
- #7896, #11989: Add a *py:type* directive for documenting type aliases, and a *py:type* role for linking to them. Patch by Ashley Whetter.
- #12549: Add optional description argument to Sphinx.add_config_value(). Patch by Chris Sewell.
- #6792: Prohibit module import cycles in sphinx.ext.autosummary. Patch by Trevor Bekolay.
- #12508: LaTeX: Revamped styling of all admonitions, with addition of a title row with icon. Patch by Jean-François B.
- #11773: Display Annotated annotations with their metadata in the Python domain. Patch by Adam Turner and David Stansby.
- #12506: Add heading-level option to *rubric* directive. Patch by Chris Sewell.
- #12567: Add the write-started event. Patch by Chris Sewell.

Bugs fixed

- #12314: Properly format collections.abc.Callable in annotations. Patch by Adam Turner.
- #12162: Fix a performance regression in the C domain that has been present since version 3.0.0. Patch by Donald Hunter.
- #12320: Fix removal of anchors from search summaries (regression in 7.3.0). Patch by Will Lachance.
- #12251: Fix merge_domaindata() in sphinx.ext.duration. Patch by Matthias Geier.
- #12224: Properly detect WebP files. Patch by Benjamin Cabé.
- #12380: LaTeX: Avoid footnote markers Page N when N is already the current page number. Patch by Jean-François B.
- #12410: LaTeX: for French and 'lualatex' as *latex_engine* use babel as with 'xelatex' (and not polyglossia). Patch by Jean-François B.
- #8807, #12520: LaTeX: let todolist produce correct hyperlinks in PDF. Patch by Jean-François B.
- #12416: Ensure that configuration setting aliases are always synchronised when one value or the other is modified. Patch by Bénédikt Tran.
- #12220: Fix loading custom template translations for en locale. Patch by Nicolas Peugnet.
- #12459: Add valid-type arguments to the linkcheck_rate_limit_timeout configuration setting. Patch by James Addison.
- #12331: Resolve data-URI-image-extraction regression from v7.3.0 affecting builders without native support for data-URIs in their output format. Patch by James Addison.
- #12494: Fix invalid genindex.html file produced with translated docs (regression in 7.1.0). Patch by Nicolas Peugnet.
- #11961: Omit anchor references from document title entries in the search index, removing duplication of search results. Patch by James Addison.
- #12425: Use Docutils' SVG processing in the HTML builder and remove Sphinx's custom logic. Patch by Tunç Basar Köse.
- #12391: Adjust scoring of matches during HTML search so that document main titles tend to rank higher than subsection titles. In addition, boost matches on the name of programming domain objects relative to title/subtitle matches. Patch by James Addison and Will Lachance.
- #9634: Do not add a fallback language by stripping the country code. Patch by Alvin Wong.
- #12352: Add domain objects to the table of contents in the same order as defined in the document. Previously, each domain used language-specific nesting rules, which removed control from document authors. Patch by Jakob Lykke Andersen and Adam Turner.
- #11041: linkcheck: Ignore URLs that respond with non-Unicode content. Patch by James Addison.
- #12543: Fix **PEP 695**¹⁵⁰⁰ formatting for LaTeX output. Patch by Bénédikt Tran.

Testing

• karma: refactor HTML search tests to use fixtures generated by Sphinx. Patch by James Addison.

¹⁵⁰⁰ https://peps.python.org/pep-0695/

Sphinx 7.3

Release 7.3.7 (released Apr 19, 2024)

Bugs fixed

- #12299: Defer loading themes defined via entry points until their explicit use by the user or a child theme. Patch by Adam Turner.
- #12305: Return the default value for theme.get_config() with an unsupported theme configuration section. Patch by Adam Turner.

Release 7.3.6 (released Apr 17, 2024)

Bugs fixed

- #12295: Re-export all AST types in the C and C++ domains. Patch by Adam Turner.
- #12295: Re-export various objects from sphinx.domains.python._annotations in sphinx.domains.python. Patch by Jacob Chesslo and Adam Turner.

Release 7.3.5 (released Apr 17, 2024)

Bugs fixed

• #12295: Re-export various objects from sphinx.domains.python._object in sphinx.domains.python. Patch by Jacob Chesslo and Adam Turner.

Release 7.3.4 (released Apr 17, 2024)

Bugs fixed

• Handle cases when Any is not an instance of type. Patch by Adam Turner.

Release 7.3.3 (released Apr 17, 2024)

Bugs fixed

• #12290: Fix a false-positive warning when setting a configuration value with Any as the valid type to a type other than the value's default. Patch by Adam Turner.

Release 7.3.2 (released Apr 17, 2024)

Bugs fixed

- Preload all themes defined via entry points. Patch by Adam Turner.
- Fix a bad interaction between the 'Furo' theme and the new-style for configuration values. Patch by Adam Turner.

Release 7.3.1 (released Apr 17, 2024)

Dependencies

• Require tomli on Python 3.10 and earlier. Patch by Adam Turner.

Release 7.3.0 (released Apr 16, 2024)

Dependencies

- #11858: Increase the minimum supported version of Alabaster to 0.7.14. Patch by Adam Turner.
- #11411: Support Docutils 0.21¹⁵⁰¹. Patch by Adam Turner.
- #12012: Use types-docutils instead of docutils-stubs.

Deprecated

- #11693: Support for old-style Makefile and make.bat output in **sphinx-quickstart**, and the associated options -M, -m, --no-use-make-mode, and --use-make-mode.
- #11285: Direct access to sphinx.testing.util.SphinxTestApp._status or sphinx.testing.util.SphinxTestApp._warning is deprecated. Use the public properties sphinx.testing.util.SphinxTestApp.status and sphinx.testing.util.SphinxTestApp.warninginstead.Patch by Bénédikt Tran.
- tests: sphinx.testing.util.strip_escseq() is deprecated in favour of sphinx.util.console. strip_colors(). Patch by Bénédikt Tran.

Features added

- #12265: Support theme configuration via theme.toml.
- #11701: HTML Search: Adopt the new < search> 1502 element. Patch by Bénédikt Tran.
- #11776: Add long option names to sphinx-build. Patch by Hugo van Kemenade, Adam Turner, Bénédikt Tran, and Ezio Melotti.
- Organise the sphinx-build options into groups. Patch by Adam Turner.
- #11855: Defer computation of configuration values. Patch by Adam Turner.
- Add :no-search: as an alias of the :nosearch: metadata field. Patch by Adam Turner.
- #11803: autodoc: Use an overridden __repr__() function in an enum, if defined. Patch by Shengyu Zhang.
- #11825: Allow custom targets in the manpage role. Patch by Nicolas Peugnet.
- #11892: Improved performance when resolving cross references in the C++ domain. Patch by Rouslan Korneychuk.
- #11905: Add a versionremoved directive. Patch by Hugo van Kemenade, Adam Turner, and C.A.M. Gerlach.
- #11981: Improve rendering of signatures using slice syntax, e.g., def foo(arg: np.float64[:,:]) -> None:
- The manpage builder now adds OSC 8¹⁵⁰³ anchors to hyperlinks, using the groff¹⁵⁰⁴ device control command.
- #11015: Change the text of the *versionadded* directive from New in [...] to Added in [...]. Patch by Bénédikt Tran.
- #12131: Added *show_warning_types* configuration option. Patch by Chris Sewell.
- #12193: Improve external warnings for unknown roles. In particular, suggest related role names if an object type is mistakenly used. Patch by Chris Sewell.

 $^{^{1501}\} https://docutils.sourceforge.io/RELEASE-NOTES.html\#release-0-21-2024-04-09$

¹⁵⁰² https://developer.mozilla.org/en-US/docs/Web/HTML/Element/search

¹⁵⁰³ https://gist.github.com/egmontkob/eb114294efbcd5adb1944c9f3cb5feda

 $^{^{1504}\} https://lists.gnu.org/archive/html/groff/2021-10/msg00000.html$

• Add public type alias *sphinx.util.typing.ExtensionMetadata*. This can be used by extension developers to annotate the return type of their setup function. Patch by Chris Sewell.

Bugs fixed

- #11668: Raise a useful error when theme.conf is missing. Patch by Vinay Sajip.
- #11622: Ensure that the order of keys in searchindex.js is deterministic. Patch by Pietro Albini.
- #11617: ANSI control sequences are stripped from the output when writing to a warnings file with -w. Patch by Bénédikt Tran.
- #11666: Skip all hidden directories in CatalogRepository.pofiles. Patch by Aryaz Eghbali.
- #9686: html builder: Fix MathJax lazy loading when equations appear in titles. Patch by Bénédikt Tran.
- #11483: singlehtml builder: Fix MathJax lazy loading when the index does not contain any math equations. Patch by Bénédikt Tran.
- #11697: HTML Search: add 'noindex' meta robots tag. Patch by James Addison.
- #11678: Fix a possible ZeroDivisionError in sphinx.ext.coverage. Patch by Stephen Finucane.
- #11756: LaTeX: build error with recent TeXLive due to missing substitutefont package (triggered if using fontenc with T2A option and document language is not a Cyrillic one). Patch by Jean-François B.
- #11675: Fix rendering of progression bars in environments that do not support ANSI control sequences. Patch by Bénédikt Tran.
- #11861: Whitelist more types with an incorrect __module__ attribute. Patch by Adam Turner.
- #11715: Apply tls_verify and tls_cacerts config to ImageDownloader. Patch by Nick Touran.
- Allow hyphens in group names for productionlist cross-references. Patch by Adam Turner.
- #11433: Added the <code>linkcheck_allow_unauthorized</code> configuration option. Set this option to False to report HTTP 401 (unauthorized) server responses as broken. Patch by James Addison.
- #11868: linkcheck: added a distinct timeout reporting status code. This can be enabled by setting linkcheck_report_timeouts_as_broken to False. Patch by James Addison.
- #11869: Refresh the documentation for the linkcheck_timeout setting. Patch by James Addison.
- #11874: Configure a default 30-second value for linkcheck_timeout. Patch by James Addison.
- #11886: Print the Jinja2 template path chain in TemplateNotFound exceptions. Patch by Colin Marquardt.
- #11598: Do not use query components in URLs for assets in EPUB rendering. Patch by David Runge.
- #11904: Support unary subtraction when parsing annotations. Patch by James Addison.
- #11925: Blacklist the sphinxprettysearchresults extension; the functionality it provides was merged into Sphinx v2.0.0. Patch by James Addison.
- #11917: Fix rendering of annotated inherited members for Python 3.9. Patch by Janet Carson.
- #11935: C Domain: Fix namespace-pop context. Patch by Frank Dana.
- #11923: Avoid zombie processes when parallel builds fail. Patch by Felix von Drigalski.
- #11353: Support enumeration classes inheriting from mixin or data types. Patch by Bénédikt Tran.
- #11962: Fix target resolution when using :paramtype: fields. Patch by Bénédikt Tran.
- #11944: Use anchor in search preview. Patch by Will Lachance.
- #12008: Fix case-sensitive lookup of std:label names in intersphinx inventory. Patch by Michael Goerz.

- #11958: HTML Search: Fix partial matches overwriting full matches. Patch by William Lachance.
- #11959: Fix multiple term matching when word appears in both title and document. Patch by Will Lachance.
- #11474: Fix doctrees caching causing files not be rebuilt in some cases, e.g., when *numfig* is True. Patch by Bénédikt Tran.
- #11278: autodoc: Fix rendering of functools.singledispatchmethod¹⁵⁰⁵ combined with @classmethod¹⁵⁰⁶. Patch by Bénédikt Tran.
- #11894: Do not add checksums to css files if building using the htmlhelp builder. Patch by reduerK akiM.
- #12052: Remove <script> and <style> tags from the content of search result summary snippets. Patch by James Addison.
- #11578: HTML Search: Order non-main index entries after other results. Patch by Brad King.
- #12147: autosummary: Fix a bug whereby the wrong file extension may be used, when multiple suffixes are specified in *source_suffix*. Patch by Sutou Kouhei.
- #10786: improve the error message when a file to be copied (e.g., an asset) is removed during Sphinx execution. Patch by Bénédikt Tran.
- #12040: HTML Search: Ensure that document titles that are partially-matched by the user search query are included in search results. Patch by James Addison.
- #11970: singlehtml builder: make target URIs to be same-document references in the sense of RFC 3986, §4.4¹⁵⁰⁷, e.g., index.html#foo becomes #foo. Patch by Eric Norige.
- #12271: Partially revert Docutils' r9562¹⁵⁰⁸ to fix EPUB files. Patch by Adam Turner.
- #12253: Escape reserved path characters in the remote images post-transform download cache. Patch by James Addison and Adam Turner.

Testing

- Reorganise tests into directories. Patch by Adam Turner.
- Clean up global state in SphinxTestApp. Patch by Adam Turner.
- #11285: pytest.mark.sphinx() and sphinx.testing.util.SphinxTestApp accept warningiserror, keep_going and verbosity as keyword arguments. Patch by Bénédikt Tran.
- #11285: sphinx.testing.util.SphinxTestApp *status* and *warning* arguments are checked to be io. StringIO¹⁵⁰⁹ objects (the public API incorrectly assumed this without checking it). Patch by Bénédikt Tran.
- Report the result of test_run_epubcheck as skipped instead of success when either Java or epubcheck are not available.
- Use dynamic allocation of unused port numbers for the test HTTP(S) servers. As a side-effect, this removes the need for test server lockfiles, meaning that any remaining tests/test-server.lock files can safely be deleted.

 $^{^{1505}\} https://docs.python.org/3/library/functools.html\#functools.single dispatchmethod$

¹⁵⁰⁶ https://docs.python.org/3/library/functions.html#classmethod

¹⁵⁰⁷ https://datatracker.ietf.org/doc/html/rfc3986.html#section-4.4

¹⁵⁰⁸ https://sourceforge.net/p/docutils/code/9562/

¹⁵⁰⁹ https://docs.python.org/3/library/io.html#io.StringIO

Sphinx 7.2

Release 7.2.6 (released Sep 13, 2023)

Bugs fixed

- #11679: Add the SPHINX_AUTODOC_RELOAD_MODULES environment variable, which if set reloads modules when using autodoc with TYPE_CHECKING = True. Patch by Matt Wozniski and Adam Turner.
- #11679: Use importlib.reload()¹⁵¹⁰ to reload modules in autodoc. Patch by Matt Wozniski and Adam Turner.

Release 7.2.5 (released Aug 30, 2023)

Bugs fixed

- #11645: Fix a regression preventing autodoc from importing modules within packages that make use of if typing.TYPE_CHECKING: to guard circular imports needed by type checkers. Patch by Matt Wozniski.
- #11634: Fixed inheritance diagram relative link resolution for sibling files in a subdirectory. Patch by Albert Shih.
- #11659: Allow ?config=... in mathjax_path.
- #11654: autodoc: Fail with a more descriptive error message when an object claims to be an instance of type, but is not a class. Patch by James Braza.
- 11620: Cease emitting *source-read* events for files read via the include¹⁵¹¹ directive.
- 11620: Add a new *include-read* for observing and transforming the content of included files via the include directive.
- #11627: Restore support for copyright lines of the form YYYY when SOURCE_DATE_EPOCH is set.

Release 7.2.4 (released Aug 28, 2023)

Bugs fixed

- #11618: Fix a regression in the MoveModuleTargets transform, introduced in #10478 (#9662).
- #11649: linkcheck: Resolve hanging tests for timezones west of London and incorrect conversion from UTC to offsets from the UNIX epoch. Patch by Dmitry Shachnev and Adam Turner.

Release 7.2.3 (released Aug 23, 2023)

Dependencies

• #11576: Require sphinxcontrib-serializinghtml 1.1.9.

Bugs fixed

- Fix regression in autodoc.Documenter.parse_name().
- Fix regression in JSON serialisation.
- #11543: autodoc: Support positional-only parameters in classmethod methods when autodoc_preserve_defaults is True.

¹⁵¹⁰ https://docs.python.org/3/library/importlib.html#importlib.reload

¹⁵¹¹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#include

¹⁵¹² https://docutils.sourceforge.io/docs/ref/rst/directives.html#include

• Restore support string methods on path objects. This is deprecated and will be removed in Sphinx 8. Use os. fspath()¹⁵¹³ to convert Path¹⁵¹⁴ objects to strings, or Path¹⁵¹⁵'s methods to work with path objects.

Release 7.2.2 (released Aug 17, 2023)

Bugs fixed

- Fix the signature of the StateMachine.insert_input() patch, for when calling with keyword arguments.
- Fixed membership testing (in) for the str¹⁵¹⁶ interface of the asset classes (_CascadingStyleSheet and _JavaScript), which several extensions relied upon.
- Fixed a type error in SingleFileHTMLBuilder._get_local_toctree, includehidden may be passed as a string or a boolean.
- Fix :noindex: for PyModule and JSModule.

Release 7.2.1 (released Aug 17, 2023)

Bugs fixed

- Restored the the str¹⁵¹⁷ interface of the asset classes (_CascadingStyleSheet and _JavaScript), which several extensions relied upon. This will be removed in Sphinx 9.
- Restored calls to Builder.add_{css,js}_file(), which several extensions relied upon.
- Restored the private API TocTree.get_toctree_ancestors(), which several extensions relied upon.

Release 7.2.0 (released Aug 17, 2023)

Dependencies

- #11511: Drop Python 3.8 support.
- #11576: Require Pygments 2.14 or later.

Deprecated

- #11512: Deprecate sphinx.util.md5 and sphinx.util.sha1. Use hashlib instead.
- #11526: Deprecate sphinx.testing.path. Use os.path or pathlib instead.
- #11528: Deprecate sphinx.util.split_index_msg and sphinx.util.split_into. Use sphinx.util.index_entries.split_index_msg instead.
- Deprecate sphinx.builders.html.Stylesheet and sphinx.builders.html.Javascript. Use sphinx.application.Sphinx.add_css_file() and sphinx.application.Sphinx.add_js_file() instead.
- #11582: Deprecate sphinx.builders.html.StandaloneHTMLBuilder.css_files and sphinx. builders.html.StandaloneHTMLBuilder.script_files. Use sphinx.application.Sphinx. add_css_file() and sphinx.application.Sphinx.add_js_file() instead.
- #11459: Deprecate sphinx.ext.autodoc.preserve_defaults.get_function_def(). Patch by Bénédikt Tran.

¹⁵¹³ https://docs.python.org/3/library/os.html#os.fspath

¹⁵¹⁴ https://docs.python.org/3/library/pathlib.html#pathlib.Path

¹⁵¹⁵ https://docs.python.org/3/library/pathlib.html#pathlib.Path

¹⁵¹⁶ https://docs.python.org/3/library/stdtypes.html#str

¹⁵¹⁷ https://docs.python.org/3/library/stdtypes.html#str

Features added

- #11526: Support os.PathLike types and pathlib.Path objects in many more places.
- #5474: coverage: Print summary statistics tables. Patch by Jorge Leitao.
- #6319: viewcode: Add *viewcode_line_numbers* to control whether line numbers are added to rendered source code. Patch by Ben Krikler.
- #9662: Add the :no-typesetting: option to suppress textual output and only create a linkable anchor. Patch by Latosha Maltba.
- #11221: C++: Support domain objects in the table of contents. Patch by Rouslan Korneychuk.
- #10938: doctest: Add doctest_show_successes option. Patch by Trey Hunner.
- #11533: Add :no-index:, :no-index-entry:, and :no-contents-entry:.
- #11572: Improve debug logging of reasons why files are detected as out of date. Patch by Eric Larson.
- #10678: Emit source-read events for files read via the include 1518 directive. Patch by Halldor Fannar.
- #11570: Use short names when using PEP 585¹⁵¹⁹ built-in generics. Patch by Riccardo Mori.
- #11300: Improve SigElementFallbackTransform fallback logic and signature text elements nodes. See *the documentation* for more details. Patch by Bénédikt Tran.
- Allow running Sphinx with python -m sphinx build \ldots

Bugs fixed

- #11077: graphviz: Fix relative links from within the graph. Patch by Ralf Grubenmann.
- #11529: Line Block in LaTeX builder outputs spurious empty token. Patch by Adrian Vollmer.
- #11196: autosummary: Summary line extraction failed with "e.g."
- #10614: Fixed a number of bugs in inheritance diagrams that resulted in missing or broken links. Patch by Albert Shih.
- #9428: Exclude substitution definitions when running the gettext builder. Patch by Alvin Wong.
- #10795: Raise a descriptive error if graphviz_dot is falsy.
- #11546: Translated nodes identical to their original text are now marked with the translated=True attribute.
- #10049: html: Change "Permalink" to "Link" for title text in link anchors.
- #4225: Relax Pygments parsing on lexing failures.
- #11246: Allow inline links in the first line of a docstring and one-line type comments #: :meta ...: when using *sphinx.ext.napoleon*. Patch by Bénédikt Tran.
- #10930: Highlight all search terms on the search results page. Patch by Dmitry Shachnev.
- #11473: Type annotations containing Literal 1520 enumeration values now render correctly. Patch by Bénédikt Tran.
- #11591: Fix support for C coverage in sphinx.ext.coverage extension. Patch by Stephen Finucane.
- #11594: HTML Theme: Enhancements to horizontal scrolling on smaller devices in the agogo theme. Patch by Lukas Engelter.

¹⁵¹⁸ https://docutils.sourceforge.io/docs/ref/rst/directives.html#include

¹⁵¹⁹ https://peps.python.org/pep-0585/

¹⁵²⁰ https://docs.python.org/3/library/typing.html#typing.Literal

• #11459: Fix support for async and lambda functions in sphinx.ext.autodoc.preserve_defaults. Patch by Bénédikt Tran.

Testing

- #11577: pytest: Fail tests on "XPASS".
- #11577: pytest: Use "importlib" import mode.
- #11577: pytest: Set PYTHONWARNINGS=error.
- #11577: pytest: Set strict config and strict markers.

Sphinx 7.1

Release 7.1.2 (released Aug 02, 2023)

Bugs fixed

• #11542: linkcheck: Properly respect *linkcheck_anchors* and do not spuriously report failures to validate anchors. Patch by James Addison.

Release 7.1.1 (released Jul 27, 2023)

Bugs fixed

• #11514: Fix SOURCE_DATE_EPOCH in multi-line copyright footer. Patch by Bénédikt Tran.

Release 7.1.0 (released Jul 24, 2023)

Incompatible changes

• Releases are no longer signed, given the change in PyPI policy¹⁵²¹.

Deprecated

• #11412: Emit warnings on using a deprecated Python-specific index entry type (namely, module, keyword, operator, object, exception, statement, and builtin) in the *index* directive, and set the removal version to Sphinx 9. Patch by Adam Turner.

Features added

- #11415: Add a checksum to JavaScript and CSS asset URIs included within generated HTML, using the CRC32 algorithm.
- require_sphinx() now allows the version requirement to be specified as (major, minor).
- #11011: Allow configuring a line-length limit for object signatures, via <code>maximum_signature_line_length</code> and the domain-specific variants. If the length of the signature (in characters) is greater than the configured limit, each parameter in the signature will be split to its own logical line. This behaviour may also be controlled by options on object description directives, for example <code>py:function:single-line-parameter-list</code>. Patch by Thomas Louf, Adam Turner, and Jean-François B.
- #10983: Support for multiline copyright statements in the footer block. Patch by Stefanie Molin
- sphinx.util.display.status_iterator now clears the current line with ANSI control codes, rather than overprinting with space characters.

¹⁵²¹ https://blog.pypi.org/posts/2023-05-23-removing-pgp/

- #11431: linkcheck: Treat SSL failures as broken links. Patch by James Addison.
- #11157: Keep the translated attribute on translated nodes.
- #11451: Improve the traceback displayed when using sphinx-build -T in parallel builds. Patch by Bénédikt Tran
- #11324: linkcheck: Use session-basd HTTP requests.
- #11438: Add support for the *py:class* and *py:function* directives for PEP 695 (generic classes and functions declarations) and PEP 696 (default type parameters). Multi-line support (#11011) is enabled for type parameters list and can be locally controlled on object description directives, e.g., *py:function:single-line-type-parameter-list*. Patch by Bénédikt Tran.
- #11484: linkcheck: Allow HTML anchors to be ignored on a per-URL basis via linkcheck_anchors_ignore_for_url while still checking the validity of the page itself. Patch by Bénédikt Tran
- #1246: Add translation progress statistics and inspection support, via a new substitution (|translation progress|) and a new configuration variable (translation_progress_classes). These enable determining the percentage of translated elements within a document, and the remaining translated and untranslated elements.

Bugs fixed

- Restored the footnote-reference class that has been removed in the latest (unreleased) version of Docutils.
- #11486: Use RFC 8081¹⁵²² font file MIME types in the EPUB builder. Using the correct MIME type will prevent warnings from epubcheck and will generate a valid EPUB.
- #11435: Use microsecond-resolution timestamps for outdated file detection in BuildEnvironment. get_outdated_files.
- #11437: Top-level headings starting with a reStructuredText role now render properly when *rst_prolog* is set. Previously, a file starting with the below would have improperly rendered due to where the prologue text was inserted into the document.

Patch by Bénédikt Tran.

• #11337: Fix a MemoryError in sphinx.ext.intersphinx when using None or typing.* as inline type references. Patch by Bénédikt Tran (picnixz)

Testing

• #11345: Always delete docutils.conf in test directories when running SphinxTestApp.cleanup().

Sphinx 7.0

Release 7.0.1 (released May 12, 2023)

Dependencies

• #11411: Support Docutils 0.20¹⁵²³. Patch by Adam Turner.

¹⁵²² https://datatracker.ietf.org/doc/html/rfc8081.html

¹⁵²³ https://docutils.sourceforge.io/RELEASE-NOTES.html#release-0-20-2023-05-04

Bugs fixed

• #11418: Clean up remaining references to sphinx.setup_command following the removal of support for setuptools. Patch by Willem Mulder.

Release 7.0.0 (released Apr 29, 2023)

Incompatible changes

- #11359: Remove long-deprecated aliases for MecabSplitter and DefaultSplitter in sphinx.search.ja.
- #11360: Remove deprecated make_old_id functions in domain object description classes.
- #11363: Remove the Setuptools integration (build_sphinx hook in setup.py).
- #11364: Remove deprecated sphinx.ext.napoleon.iterators module.
- #11365: Remove support for the jsdump format in sphinx.search.
- #11366: Make locale a required argument to sphinx.util.i18n.format_date().
- #11370: Remove deprecated sphinx.util.stemmer module.
- #11371: Remove deprecated sphinx.pycode.ast.parse() function.
- #11372: Remove deprecated sphinx.io.read_doc() function.
- #11373: Removed deprecated sphinx.util.get_matching_files() function.
- #11378: Remove deprecated sphinx.util.docutils.is_html5_writer_available() function.
- #11379: Make the env argument to Builder subclasses required.
- #11380: autosummary: Always emit grouped import exceptions.
- #11381: Remove deprecated style key for HTML templates.
- #11382: Remove deprecated sphinx.writers.latex.LaTeXTranslator.docclasses attribute.
- #11383: Remove deprecated sphinx.builders.html.html5_ready and sphinx.builders.html.
 HTMLTranslator attributes.
- #11385: Remove support for HTML 4 output.

Sphinx 6.2

Release 6.2.1 (released Apr 25, 2023)

Bugs fixed

• #11355: Revert the default type of nitpick_ignore and nitpick_ignore_regex to list.

Release 6.2.0 (released Apr 23, 2023)

Dependencies

• Require Docutils 0.18.1 or greater.

Incompatible changes

• LaTeX: removal of some internal TeX \dimen registers (not previously publicly documented) as per 5.1.0 code comments in sphinx.sty: \sphinxverbatimsep, \sphinxverbatimborder, \sphinxshadowsep, \sphinxshadowsize, and \sphinxshadowrule. (refs: #11105)

- Remove .egg support from pycode ModuleAnalyser; Python eggs are a now-obsolete binary distribution format
- #11089: Remove deprecated code in sphinx.builders.linkcheck. Patch by Daniel Eades
- Remove internal-only sphinx.locale.setlocale

Deprecated

- #11247: Deprecate the legacy intersphinx_mapping format
- sphinx.util.osutil.cd is deprecated in favour of contextlib.chdir.

Features added

- #11277: *autoproperty* allows the return type to be specified as a type comment (e.g., # type: () -> int). Patch by Bénédikt Tran
- #10811: Autosummary: extend __all__ to imported members for template rendering when option autosummary_ignore_module_all is set to False. Patch by Clement Pinard
- #11147: Add a content_offset parameter to nested_parse_with_titles(), allowing for correct line numbers during nested parsing. Patch by Jeremy Maitin-Shepard
- Update to Unicode CLDR 42
- Add a -- jobs synonym for -j. Patch by Hugo van Kemenade
- LaTeX: a command \sphinxbox for styling text elements with a (possibly rounded) box, optional background color and shadow, has been added. See *The* \sphinxbox command. (refs: #11224)
- LaTeX: add \sphinxstylenotetitle, ..., \sphinxstylewarningtitle, ..., for an extra layer of mark-up freeing up \sphinxstrong for other uses. See *Macros*. (refs: #11267)
- LaTeX: note¹⁵²⁴, hint¹⁵²⁵, important¹⁵²⁶ and tip¹⁵²⁷ can now each be styled as the other admonitions, i.e. possibly with a background color, individual border widths and paddings, possibly rounded corners, and optional shadow. See *Additional CSS-like 'sphinxsetup' keys*. (refs: #11234)
- LaTeX: admonitions and topic 1528 (and contents 1529) directives, and not only *code-block*, support box-decoration-break=slice.
- LaTeX: let rounded boxes support up to 4 distinct border-widths (refs: #11243)
- LaTeX: new options noteTextColor, noteTeXextras et al. See Additional CSS-like 'sphinxsetup' keys.
- LaTeX: support elliptical corners in rounded boxes. (refs: #11254)
- #11150: Include source location in highlighting warnings, when lexing fails. Patch by Jeremy Maitin-Shepard
- #11281: Support for imgmath_latex = 'tectonic' or = 'xelatex'. Patch by Dimitar Dimitrov
- #11109, #9643: Add python_display_short_literal_types option for condensed rendering of Literal types.

¹⁵²⁴ https://docutils.sourceforge.io/docs/ref/rst/directives.html#note

¹⁵²⁵ https://docutils.sourceforge.io/docs/ref/rst/directives.html#hint

 $^{^{1526}\} https://docutils.sourceforge.io/docs/ref/rst/directives.html\#important$

¹⁵²⁷ https://docutils.sourceforge.io/docs/ref/rst/directives.html#tip

¹⁵²⁸ https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

¹⁵²⁹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents

Bugs fixed

- #11079: LaTeX: figures with align attribute may disappear and strangely impact following lists
- #11093: LaTeX: fix "multiply-defined references" PDF build warnings when one or more reST labels directly precede an *py:module* or *automodule* directive. Patch by Bénédikt Tran (picnixz)
- #11110: LaTeX: Figures go missing from latex pdf if their files have the same base name and they use a post transform. Patch by aaron-cooper
- LaTeX: fix potential color leak from shadow to border of rounded boxes, if shadow color is set but border color is not
- LaTeX: fix unintended 1pt upwards vertical shift of code blocks frames respective to contents (when using rounded corners)
- #11235: LaTeX: added \color in topic (or admonition) contents may cause color leak to the shadow and border at a page break
- #11264: LaTeX: missing space before colon after "Voir aussi" for seealso directive in French
- #11268: LaTeX: longtable with left alignment breaks out of current list indentation context in PDF. Thanks to picnixz.
- #11274: LaTeX: external links are not properly escaped for \sphinxupquote compatibility
- #11147: Fix source file/line number info in object description content and in other uses of nested_parse_with_titles. Patch by Jeremy Maitin-Shepard.
- #11192: Restore correct parallel search index building. Patch by Jeremy Maitin-Shepard
- Use the new Transifex tx client

Testing

- Fail testing when any Python warnings are emitted
- Migrate remaining unittest. TestCase style test functions to pytest style
- · Remove tests that rely on setuptools

Sphinx 6.1

Release 6.1.3 (released Jan 10, 2023)

Bugs fixed

- #11116: Reverted to previous Sphinx 5 node copying method
- #11117: Reverted changes to parallel image processing from Sphinx 6.1.0
- #11119: Suppress ValueError in the linkcheck builder

Release 6.1.2 (released Jan 07, 2023)

Bugs fixed

- #11101: LaTeX: div.topic_padding key of sphinxsetup documented at 5.1.0 was implemented with name topic_padding
- #11099: LaTeX: shadowrule key of sphinxsetup causes PDF build to crash since Sphinx 5.1.0
- #11096: LaTeX: shadowsize key of sphinxsetup causes PDF build to crash since Sphinx 5.1.0

- #11095: LaTeX: shadow of topic 1530 and contents 1531 boxes not in page margin since Sphinx 5.1.0
- #11100: Fix copying images when running under parallel mode.

Release 6.1.1 (released Jan 05, 2023)

Bugs fixed

• #11091: Fix util.nodes.apply_source_workaround for literal_block nodes with no source information in the node or the node's parents.

Release 6.1.0 (released Jan 05, 2023)

Dependencies

• Adopted the Ruff¹⁵³² code linter.

Incompatible changes

• #10979: gettext: Removed support for pluralisation in get_translation. This was unused and complicated other changes to sphinx.locale.

Deprecated

- sphinx.util functions:
 - Renamed sphinx.util.typing.stringify() to sphinx.util.typing. stringify_annotation()
 - Moved sphinx.util.xmlname_checker() to sphinx.builders.epub3. _XML_NAME_PATTERN

Moved to sphinx.util.display:

- sphinx.util.status_iterator
- sphinx.util.display_chunk
- sphinx.util.SkipProgressMessage
- sphinx.util.progress_message

Moved to sphinx.util.http_date:

- sphinx.util.epoch_to_rfc1123
- sphinx.util.rfc1123_to_epoch

Moved to sphinx.util.exceptions:

- sphinx.util.save_traceback
- sphinx.util.format_exception_cut_frames

¹⁵³⁰ https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

¹⁵³¹ https://docutils.sourceforge.io/docs/ref/rst/directives.html#table-of-contents

¹⁵³² https://github.com/charliermarsh/ruff

Features added

- Cache doctrees in the build environment during the writing phase.
- Make all writing phase tasks support parallel execution.
- #11072: Use PEP 604 (X | Y) display conventions for typing.Optional and typing.Optional types within the Python domain and autodoc.
- #10700: autodoc: Document typing.NewType() types as classes rather than 'data'.
- Cache doctrees between the reading and writing phases.

Bugs fixed

- #10962: HTML: Fix the multi-word key name lookup table.
- Fixed support for Python 3.12 alpha 3 (changes in the enum module).
- #11069: HTML Theme: Removed outdated "shortcut" link relation keyword.
- #10952: Properly terminate parallel processes on programme interruption.
- #10988: Speed up TocTree.resolve() through more efficient copying.
- #6744: LaTeX: support for seealso directive should be via an environment to allow styling.
- #11074: LaTeX: Can't change sphinxnote to use sphinxheavybox starting with 5.1.0

Sphinx 6.0

Release 6.0.1 (released Jan 05, 2023)

Dependencies

• Require Pygments 2.13 or later.

Bugs fixed

• #10944: imgmath: Fix resolving image paths for files in nested folders.

Release 6.0.0 (released Dec 29, 2022)

Dependencies

- #10468: Drop Python 3.6 support
- #10470: Drop Python 3.7, Docutils 0.14, Docutils 0.15, Docutils 0.16, and Docutils 0.17 support. Patch by Adam Turner

Incompatible changes

• #7405: Removed the jQuery and underscore.js JavaScript frameworks.

These frameworks are no longer be automatically injected into themes from Sphinx 6.0. If you develop a theme or extension that uses the jQuery, \$, or \$u global objects, you need to update your JavaScript to modern standards, or use the mitigation below.

The first option is to use the sphinxcontrib.jquery¹⁵³³ extension, which has been developed by the Sphinx team and contributors. To use this, add sphinxcontrib.jquery to the extensions list in conf.py, or call app. setup_extension("sphinxcontrib.jquery") if you develop a Sphinx theme or extension.

¹⁵³³ https://github.com/sphinx-contrib/jquery/

The second option is to manually ensure that the frameworks are present. To re-add jQuery and underscore.js, you will need to copy jquery.js and underscore.js from the Sphinx repository 1534 to your static directory, and add the following to your layout.html:

Patch by Adam Turner.

- #10471, #10565: Removed deprecated APIs scheduled for removal in Sphinx 6.0. See *Deprecated APIs* for details. Patch by Adam Turner.
- #10901: C Domain: Remove support for parsing pre-v3 style type directives and roles. Also remove associated configuration variables c_allow_pre_v3 and c_warn_on_allowed_pre_v3. Patch by Adam Turner.

Features added

• #10924: LaTeX: adopt better looking defaults for tables and code-blocks. See <code>latex_table_style</code> and the <code>pre_border-radius</code> and <code>pre_background-TeXcolor</code> Additional CSS-like 'sphinxsetup' keys for the former defaults and how to re-enact them if desired.

Bugs fixed

• #10984: LaTeX: Document latex_additional_files behavior for files with .tex extension.

Sphinx 5.3

Release 5.3.0 (released Oct 16, 2022)

- #10759: LaTeX: add latex_table_style and support the 'booktabs', 'borderless', and 'colorrows' styles. (thanks to Stefan Wiehler for initial pull requests #6666, #6671)
- #10840: One can cross-reference including an option value like :option:`--module=foobar`, :option:`--module[=foobar]`, or :option:`--module foobar`. Patch by Martin Liska.
- #10881: autosectionlabel: Record the generated section label to the debug log.
- #10268: Correctly URI-escape image filenames.
- #10887: domains: Allow sections in all the content of all object description directives (e.g. *py:function*). Patch by Adam Turner

Sphinx 5.2

Release 5.2.3 (released Sep 30, 2022)

- #10878: Fix base64 image embedding in sphinx.ext.imgmath
- #10886: Add :nocontentsentry: flag and global domain table of contents entry control option. Patch by Adam Turner

 $^{^{1534}\} https://github.com/sphinx-doc/sphinx/tree/v5.3.0/sphinx/themes/basic/static$

Release 5.2.2 (released Sep 27, 2022)

• #10872: Restore link targets for autodoc modules to the top of content. Patch by Dominic Davis-Foster.

Release 5.2.1 (released Sep 25, 2022)

Bugs fixed

- #10861: Always normalise the pycon3 lexer to pycon.
- Fix using sphinx.ext.autosummary with modules containing titles in the module-level docstring.

Release 5.2.0.post0 (released Sep 24, 2022)

· Recreated source tarballs for Debian maintainers.

Release 5.2.0 (released Sep 24, 2022)

Dependencies

• #10356: Sphinx now uses declarative metadata with pyproject.toml to create packages, using PyPA's flit project as a build backend. Patch by Adam Turner.

Deprecated

• #10843: Support for HTML 4 output. Patch by Adam Turner.

Features added

- #10738: napoleon: Add support for docstring types using 'of', like type of type. Example: tuple of int.
- #10286: C++, support requires clauses not just between the template parameter lists and the declaration.
- #10755: linkcheck: Check the source URL of raw directives that use the url option.
- #10781: Allow ref role to be used with definitions and fields.
- #10717: HTML Search: Increase priority for full title and subtitle matches in search results
- #10718: HTML Search: Save search result score to the HTML element for debugging
- #10673: Make toctree accept 'genindex', 'modindex' and 'search' docnames
- #6316, #10804: Add domain objects to the table of contents. Patch by Adam Turner
- #6692: HTML Search: Include explicit *index* directive index entries in the search index and search results. Patch by Adam Turner
- #10816: imgmath: Allow embedding images in HTML as base64
- #10854: HTML Search: Use browser localstorage for highlight control, stop storing highlight parameters in URL query strings. Patch by Adam Turner.

Bugs fixed

- #10723: LaTeX: 5.1.0 has made the 'sphinxsetup' verbatimwithframe=false become without effect.
- #10257: C++, ensure consistent non-specialization template argument representation.
- #10729: C++, fix parsing of certain non-type template parameter packs.
- #10715: Revert #10520: "Fix" use of sidebar classes in agogo.css_t

Sphinx 5.1

Release 5.1.1 (released Jul 26, 2022)

Bugs fixed

- #10701: Fix ValueError in the new deque based sphinx.ext.napoleon iterator implementation.
- #10702: Restore compatibility with third-party builders.

Release 5.1.0 (released Jul 24, 2022)

Dependencies

• #10656: Support Docutils 0.19¹⁵³⁵. Patch by Adam Turner.

Deprecated

- #10467: Deprecated sphinx.util.stemmer in favour of snowballstemmer. Patch by Adam Turner.
- #9856: Deprecated sphinx.ext.napoleon.iterators.

Features added

- #10444: html theme: Allow specifying multiple CSS files through the stylesheet setting in theme.conf or by setting html_style to an iterable of strings.
- #10366: std domain: Add support for emphasising placeholders in *option* directives through a new *option_emphasise_placeholders* configuration option.
- #10439: std domain: Use the repr of some variables when displaying warnings, making whitespace issues easier to identify.
- #10571: quickstart: Reduce content in the generated conf.py file. Patch by Pradyun Gedam.
- #10648: LaTeX: CSS-named-alike additional 'sphinxsetup' keys allow to configure four separate border-widths, four paddings, four corner radii, a shadow (possibly inset), colours for border, background, shadow for each of the code-block, topic, attention, caution, danger, error and warning directives.
- #10655: LaTeX: Explain non-standard encoding in LatinRules.xdy
- #10599: HTML Theme: Wrap consecutive footnotes in an <aside> element when using Docutils 0.18 or later, to allow for easier styling. This matches the behaviour introduced in Docutils 0.19. Patch by Adam Turner.
- #10518: config: Add include_patterns as the opposite of exclude_patterns. Patch by Adam Turner.

Bugs fixed

- #10594: HTML Theme: field term colons are doubled if using Docutils 0.18+
- #10596: Build failure if Docutils version is 0.18 (not 0.18.1) due to missing Node.findall()
- #10506: LaTeX: build error if highlighting inline code role in figure caption (refs: #10251)
- #10634: Make -P (pdb) option work better with exceptions triggered from events
- #10550: py domain: Fix spurious whitespace in unparsing various operators (+, -, ~, and **). Patch by Adam Turner (refs: #10551).
- #10460: logging: Always show node source locations as absolute paths.

¹⁵³⁵ https://docutils.sourceforge.io/RELEASE-NOTES.html#release-0-19-2022-07-05

- HTML Search: HTML tags are displayed as a part of object name
- HTML Search: search snippets should not be folded
- HTML Search: Minor errors are emitted on fetching search snippets
- HTML Search: The markers for header links are shown in the search result
- #10520: HTML Theme: Fix use of sidebar classes in agogo.css_t.
- #6679: HTML Theme: Fix inclusion of hidden toctrees in the agogo theme.
- #10566: HTML Theme: Fix enable_search_shortcuts does not work
- #8686: LaTeX: Text can fall out of code-block at end of page and leave artifact on next page
- #10633: LaTeX: user injected \color commands in topic or admonition boxes may cause color leaks in PDF due to upstream framed.sty¹⁵³⁶ bug
- #10638: LaTeX: framed coloured boxes in highlighted code (e.g. highlighted diffs using Pygments style 'manni') inherit thickness of code-block frame
- #10647: LaTeX: Only one \label is generated for desc_signature node even if it has multiple node IDs
- #10579: i18n: UnboundLocalError is raised on translating raw directive
- #9577, #10088: py domain: Fix warning for duplicate Python references when using :any: and autodoc.
- #10548: HTML Search: fix minor summary issues.

Sphinx 5.0

Release 5.0.2 (released Jun 17, 2022)

Features added

• #10523: HTML Theme: Expose the Docutils's version info tuple as a template variable, docutils_version_info. Patch by Adam Turner.

Bugs fixed

- #10538: autodoc: Inherited class attribute having docstring is documented even if autodoc_inherit_docstring is disabled
- #10509: autosummary: autosummary fails with a shared library
- #10497: py domain: Failed to resolve strings in Literal. Patch by Adam Turner.
- #10523: HTML Theme: Fix double brackets on citation references in Docutils 0.18+. Patch by Adam Turner.
- #10534: Missing CSS for nav.contents in Docutils 0.18+. Patch by Adam Turner.

Release 5.0.1 (released Jun 03, 2022)

- #10498: gettext: TypeError is raised when sorting warning messages if a node has no line number. Patch by Adam Turner.
- #10493: HTML Theme: topic 1537 directive is rendered incorrectly with Docutils 0.18. Patch by Adam Turner.
- #10495: IndexError is raised for a *kbd* role having a separator. Patch by Adam Turner.

¹⁵³⁶ https://ctan.org/pkg/framed

¹⁵³⁷ https://docutils.sourceforge.io/docs/ref/rst/directives.html#topic

Release 5.0.0 (released May 30, 2022)

Dependencies

5.0.0 b1

• #10164: Support Docutils 0.18¹⁵³⁸. Patch by Adam Turner.

Incompatible changes

5.0.0 b1

- #10031: autosummary: sphinx.ext.autosummary.import_by_name() now raises ImportExceptionGroup instead of ImportError when it failed to import target object. Please handle the exception if your extension uses the function to import Python object. As a workaround, you can disable the behavior via grouped_exception=False keyword argument until v7.0.
- #9962: texinfo: Customizing styles of emphasized text via @definfoenclose command was not supported because the command was deprecated since texinfo 6.8
- #2068: intersphinx_disabled_reftypes has changed default value from an empty list to ['std:doc'] as avoid too surprising silent intersphinx resolutions. To migrate: either add an explicit inventory name to the references intersphinx should resolve, or explicitly set the value of this configuration variable to an empty list.
- #10197: html theme: Reduce body_min_width setting in basic theme to 360px
- #9999: LaTeX: separate terms from their definitions by a CR (refs: #9985)
- #10062: Change the default language to 'en' if any language is not set in conf.py

5.0.0 final

• #10474: *language* does not accept None as it value. The default value of language becomes to 'en' now. Patch by Adam Turner and Takeshi KOMIYA.

Deprecated

5.0.0 b1

• #10028: jQuery and underscore.js will no longer be automatically injected into themes from Sphinx 6.0. If you develop a theme or extension that uses the jQuery, \$, or \$u global objects, you need to update your JavaScript or use the mitigation below.

To re-add jQuery and underscore.js, you will need to copy jquery.js and underscore.js from the Sphinx repository 1539 to your static directory, and add the following to your layout.html:

Patch by Adam Turner.

- setuptools integration. The build_sphinx sub-command for setup.py is marked as deprecated to follow the policy of setuptools team.
- The locale argument of sphinx.util.i18n:babel_format_date() becomes required

 $^{^{1538}\} https://docutils.sourceforge.io/RELEASE-NOTES.html\#release-0-18-2021-10-26$

¹⁵³⁹ https://github.com/sphinx-doc/sphinx/tree/v5.3.0/sphinx/themes/basic/static

- The language argument of sphinx.util.i18n:format_date() becomes required
- sphinx.builders.html.html5_ready
- sphinx.io.read_doc()
- sphinx.util.docutils.__version_info__
- sphinx.util.docutils.is_html5_writer_available()
- sphinx.writers.latex.LaTeXWriter.docclasses

Features added

5.0.0 b1

- #9075: autodoc: The default value of autodoc_typehints_format is changed to 'smart'. It will suppress the leading module names of typehints (ex. io.StringIO -> StringIO).
- #8417: autodoc: :inherited-members: option now takes multiple classes. It allows to suppress inherited members of several classes on the module at once by specifying the option to automodule directive
- #9792: autodoc: Add new option for autodoc_typehints_description_target to include undocumented return values but not undocumented parameters.
- #10285: autodoc: singledispatch functions having typehints are not documented
- autodoc: autodoc_typehints_format now also applies to attributes, data, properties, and type variable bounds
- #10258: autosummary: Recognize a documented attribute of a module as non-imported
- #10028: Removed internal usages of JavaScript frameworks (jQuery and underscore.js) and modernised doctools.js and searchtools.js to EMCAScript 2018. Patch by Adam Turner.
- #10302: C++, add support for conditional expressions (?:).
- #5157, #10251: Inline code is able to be highlighted via role 1540 directive
- #10337: Make sphinx-build faster by caching Publisher object during build. Patch by Adam Turner.

Bugs fixed

5.0.0 b1

- #10200: apidoc: Duplicated submodules are shown for modules having both .pyx and .so files. Patch by Adam Turner and Takeshi KOMIYA.
- #10279: autodoc: Default values for keyword only arguments in overloaded functions are rendered as a string literal
- #10280: autodoc: autodoc_docstring_signature unexpectedly generates return value typehint for constructors if docstring has multiple signatures
- #10266: autodoc: autodoc_preserve_defaults does not work for mixture of keyword only arguments with/without defaults
- #10310: autodoc: class methods are not documented when decorated with mocked function
- #10305: autodoc: Failed to extract optional forward-ref'ed typehints correctly via autodoc_type_aliases
- #10421: autodoc: autodoc_preserve_defaults doesn't work on class methods
- #10214: html: invalid language tag was generated if language contains a country code (ex. zh_CN)

¹⁵⁴⁰ https://docutils.sourceforge.io/docs/ref/rst/directives.html#role

- #9974: html: Updated jQuery version from 3.5.1 to 3.6.0
- #10236: html search: objects are duplicated in search result
- #9962: texinfo: Deprecation message for @definfoenclose command on building texinfo document
- #10000: LaTeX: glossary terms with common definition are rendered with too much vertical whitespace
- #10188: LaTeX: alternating multiply referred footnotes produce a ? in pdf output
- #10363: LaTeX: make 'howto' title page rule use \linewidth for compatibility with usage of a twocolumn class option
- #10318: :prepend: option of literalinclude directive does not work with :dedent: option

5.0.0 final

- #9575: autodoc: The annotation of return value should not be shown when autodoc_typehints="description"
- #9648: autodoc: *args and **kwargs entries are duplicated when autodoc_typehints="description"
- #8180: autodoc: Docstring metadata ignored for attributes
- #10443: epub: EPUB builder can't detect the mimetype of .webp file
- #10104: gettext: Duplicated locations are shown if 3rd party extension does not provide correct information
- #10456: py domain: :meta: fields are displayed if docstring contains two or more meta-field
- #9096: sphinx-build: the value of progress bar for parallel build is wrong
- #10110: sphinx-build: exit code is not changed when error is raised on builder-finished event

Sphinx 4.5

Release 4.5.0 (released Mar 28, 2022)

Incompatible changes

- #10112: extlinks: Disable hardcoded links detector by default
- #9993, #10177: std domain: Disallow to refer an inline target via ref role

Deprecated

sphinx.ext.napoleon.docstring.GoogleDocstring._qualify_name()

Features added

- #10260: Enable FORCE_COLOR and NO_COLOR for terminal colouring
- #10234: autosummary: Add "autosummary" CSS class to summary tables
- #10125: extlinks: Improve suggestion message for a reference having title
- #10112: extlinks: Add extlinks_detect_hardcoded_links to enable hardcoded links detector feature
- #9494, #9456: html search: Add a config variable html_show_search_summary to enable/disable the search summaries
- #9337: HTML theme, add option enable_search_shortcuts that enables / as a Quick search shortcut and Esc shortcut that removes search highlighting.
- #10107: i18n: Allow to suppress translation warnings by adding #noqa comment to the tail of each translation message

- #10252: C++, support attributes on classes, unions, and enums.
- #10253: pep role now generates URLs based on peps.python.org 1541

- #9876: autodoc: Failed to document an imported class that is built from native binary module
- #10133: autodoc: Crashed when mocked module is used for type annotation
- #10146: autodoc: autodoc_default_options does not support no-value option
- #9971: autodoc: TypeError is raised when the target object is annotated by unhashable object
- #10205: extlinks: Failed to compile regexp on checking hardcoded links
- #10277: html search: Could not search short words (ex. "use")
- #9529: LaTeX: named auto numbered footnote (ex. [#named]) that is referred multiple times was rendered to a question mark
- #9924: LaTeX: multi-line cpp:function directive has big vertical spacing in Latexpdf
- #10158: LaTeX: excessive whitespace since v4.4.0 for undocumented variables/structure members
- #10175: LaTeX: named footnote reference is linked to an incorrect footnote if the name is also used in the different document
- #10269: manpage: Failed to resolve the title of *ref* cross references
- #10179: i18n: suppress "rST localization" warning
- #10118: imgconverter: Unnecessary availability check is called for remote URIs
- #10181: napoleon: attributes are displayed like class attributes for google style docstrings when napoleon_use_ivar is enabled
- #10122: sphinx-build: make.bat does not check the installation of sphinx-build command before showing help

Sphinx 4.4

Release 4.4.0 (released Jan 17, 2022)

Dependencies

- #10007: Use importlib_metadata for python-3.9 or older
- #10007: Drop setuptools

Features added

- #9075: autodoc: Add a config variable *autodoc_typehints_format* to suppress the leading module names of typehints of function signatures (ex. io.StringIO -> StringIO)
- #9831: Autosummary now documents only the members specified in a module's __all__ attribute if autosummary_ignore_module_all is set to False. The default behaviour is unchanged. Autogen also now supports this behavior with the --respect-module-all switch.
- #9555: autosummary: Improve error messages on failure to load target object
- #9800: extlinks: Emit warning if a hardcoded link is replaceable by an extlink, suggesting a replacement.
- #9961: html: Support nested <kbd> HTML elements in other HTML builders

¹⁵⁴¹ https://peps.python.org

- #10013: html: Allow to change the loading method of JS via loading_method parameter for Sphinx. add_js_file()
- #9551: html search: "Hide Search Matches" link removes "highlight" parameter from URL
- #9815: html theme: Wrap sidebar components in div to allow customizing their layout via CSS
- #9827: i18n: Sort items in glossary by translated terms
- #9899: py domain: Allows to specify cross-reference specifier (. and ~) as :type: option
- #9894: linkcheck: add option linkcheck_exclude_documents to disable link checking in matched documents.
- #9793: sphinx-build: Allow to use the parallel build feature in macOS on macOS and Python3.8+
- #10055: sphinx-build: Create directories when -w option given
- #9993: std domain: Allow to refer an inline target (ex. _`target name`) via ref role
- #9981: std domain: Strip value part of the option directive from general index
- #9391: texinfo: improve variable in samp role
- #9578: texinfo: Add texinfo_cross_references to disable cross references for readability with standalone readers
- #9822, #9062: add new Intersphinx role *external* for explicit lookup in the external projects, without resolving to the local project.

- #9866: autodoc: doccomment for the imported class was ignored
- #9883: autodoc: doccomment for the alias to mocked object was ignored
- #9908: autodoc: debug message is shown on building document using NewTypes with Python 3.10
- #9968: autodoc: instance variables are not shown if __init__ method has position-only-arguments
- #9194: autodoc: types under the "typing" module are not hyperlinked
- #10009: autodoc: Crashes if target object raises an error on getting docstring
- #10058: autosummary: Imported members are not shown when autodoc_class_signature = 'separated'
- #9947: i18n: topic directive having a bullet list can't be translatable
- #9878: mathjax: MathJax configuration is placed after loading MathJax itself
- #9932: napoleon: empty "returns" section is generated even if no description
- #9857: Generated RFC links use outdated base url
- #9909: HTML, prevent line-wrapping in literal text.
- #10061: html theme: Configuration values added by themes are not be able to override from conf.py
- #10073: imgconverter: Unnecessary availability check is called for "data" URIs
- #9925: LaTeX: prohibit also with 'xelatex' line splitting at dashes of inline and parsed literals
- #9944: LaTeX: extra vertical whitespace for some nested declarations
- #9940: LaTeX: Multi-function declaration in Python domain has cramped vertical spacing in latexpdf output
- #10015: py domain: types under the "typing" module are not hyperlinked defined at info-field-list

- #9390: texinfo: Do not emit labels inside footnotes
- #9413: xml: Invalid XML was generated when cross referencing python objects
- #9979: Error level messages were displayed as warning messages
- #10057: Failed to scan documents if the project is placed onto the root directory
- #9636: code-block: :dedent: without argument did strip newlines

Sphinx 4.3

Release 4.3.2 (released Dec 19, 2021)

Bugs fixed

• #9917: C and C++, parse fundamental types no matter the order of simple type specifiers.

Release 4.3.1 (released Nov 28, 2021)

Features added

• #9864: mathjax: Support changing the loading method of MathJax to "defer" via mathjax_options

Bugs fixed

- #9838: autodoc: AttributeError is raised on building document for functions decorated by functools.lru_cache
- #9879: autodoc: AttributeError is raised on building document for an object having invalid __doc__ attribute
- #9844: autodoc: Failed to process a function wrapped with functools.partial if autodoc_preserve_defaults enabled
- #9872: html: Class namespace collision between autodoc signatures and Docutils 0.17
- #9868: imgmath: Crashed if the dvisvgm command failed to convert equation
- #9864: mathjax: Failed to render equations via MathJax v2. The loading method of MathJax is back to "async" method again

Release 4.3.0 (released Nov 11, 2021)

Dependencies

• Support Python 3.10

Incompatible changes

- #9649: searchindex.js: the embedded data has changed format to allow objects with the same name in different domains.
- #9672: The rendering of Python domain declarations is implemented with more Docutils nodes to allow better CSS styling. It may break existing styling.
- #9672: the signature of domains.python.PyObject.get_signature_prefix has changed to return a list of nodes instead of a plain string.
- #9695: domains.js.JSObject.display_prefix has been changed into a method get_display_prefix which now returns a list of nodes instead of a plain string.
- #9695: The rendering of Javascript domain declarations is implemented with more Docutils nodes to allow better CSS styling. It may break existing styling.

• #9450: mathjax: Load MathJax via "defer" strategy

Deprecated

- sphinx.ext.autodoc.AttributeDocumenter._datadescriptor
- sphinx.writers.html.HTMLTranslator._fieldlist_row_index
- sphinx.writers.html.HTMLTranslator._table_row_index
- sphinx.writers.html5.HTML5Translator._fieldlist_row_index
- sphinx.writers.html5.HTML5Translator._table_row_index

Features added

- #9639: autodoc: Support asynchronous generator functions
- #9664: autodoc: autodoc-process-bases supports to inject reST snippet as a base class
- #9691: C, added new info-field retval for c: function and c:macro.
- C++, added new info-field retval for cpp: function.
- #9618: i18n: Add gettext_allow_fuzzy_translations to allow "fuzzy" messages for translation
- #9672: More CSS classes on Python domain descriptions
- #9695: More CSS classes on Javascript domain descriptions
- #9683: Revert the removal of add_stylesheet() API. It will be kept until the Sphinx 6.0 release
- #2068, add intersphinx_disabled_reftypes for disabling interphinx resolution of cross-references that do not have an explicit inventory specification. Specific types of cross-references can be disabled, e.g., std:doc or all cross-references in a specific domain, e.g., std:*.
- #9623: Allow to suppress "toctree contains reference to excluded document" warnings using suppress_warnings

Bugs fixed

- #9630: autodoc: Failed to build cross references if primary_domain is not 'py'
- #9644: autodoc: Crashed on getting source info from problematic object
- #9655: autodoc: mocked object having doc comment is warned unexpectedly
- #9651: autodoc: return type field is not generated even if autodoc_typehints_description_target is set to "documented" when its info-field-list contains :returns: field
- #9657: autodoc: The base class for a subclass of mocked object is incorrect
- #9607: autodoc: Incorrect base class detection for the subclasses of the generic class
- #9755: autodoc: memory addresses are shown for aliases
- #9752: autodoc: Failed to detect type annotation for slots attribute
- #9756: autodoc: Crashed if classmethod does not have __func__ attribute
- #9757: autodoc: autodoc_inherit_docstrings does not effect to overridden classmethods
- #9781: autodoc: autodoc_preserve_defaults does not support hexadecimal numeric
- #9630: autosummary: Failed to build summary table if primary_domain is not 'py'

- #9670: html: Fix download file with special characters
- #9710: html: Wrong styles for even/odd rows in nested tables
- #9763: html: parameter name and its type annotation are not separated in HTML
- #9649: HTML search: when objects have the same name but in different domains, return all of them as result instead of just one.
- #7634: intersphinx: references on the file in sub directory are broken
- #9737: LaTeX: hlist is rendered as a list containing "aggedright" text
- #9678: linkcheck: file extension was shown twice in warnings
- #9697: py domain: An index entry with parens was registered for py:method directive with:property: option
- #9775: py domain: Literal typehint was converted to a cross reference when autodoc_typehints='description'
- #9708: needs_extension failed to check double-digit version correctly
- #9688: Fix Sphinx patched code 1542 does not recognize :class: option
- #9733: Fix for logging handler flushing warnings in the middle of the docs build
- #9656: Fix warnings without subtype being incorrectly suppressed
- Intersphinx, for unresolved references with an explicit inventory, e.g., proj:myFunc, leave the inventory prefix in the unresolved text.

Sphinx 4.2

Release 4.2.0 (released Sep 12, 2021)

Features added

- #9445: autodoc: Support class properties
- #9479: autodoc: Emit a warning if target is a mocked object
- #9560: autodoc: Allow to refer NewType instances with module name in Python 3.10 or above
- #9447: html theme: Expose the version of Sphinx in the form of tuple as a template variable sphinx_version_tuple
- #9594: manpage: Suppress the title of man page if description is empty
- #9445: py domain: py:property directive supports :classmethod: option to describe the class property
- #9524: test: SphinxTestApp can take builddir as an argument
- #9535: C and C++, support more fundamental types, including GNU extensions.

- #9608: apidoc: apidoc does not generate a module definition for implicit namespace package
- #9504: autodoc: generate incorrect reference to the parent class if the target class inherites the class having _name attribute
- #9537, #9589: autodoc: Some objects under typing module are not displayed well with the HEAD of 3.10
- #9487: autodoc: typehint for cached_property is not shown

¹⁵⁴² https://docutils.sourceforge.io/docs/ref/rst/directives.html#code

- #9509: autodoc: AttributeError is raised on failed resolving typehints
- #9518: autodoc: autodoc_docstring_signature does not effect to __init__() and __new__()
- #9522: autodoc: PEP 585 style typehints having arguments (ex. list[int]) are not displayed well
- #9481: autosummary: some warnings contain non-existing filenames
- #9568: autosummary: summarise overlined sectioned headings correctly
- #9600: autosummary: Type annotations which contain commas in autosummary table are not removed completely
- #9481: c domain: some warnings contain non-existing filenames
- #9481: cpp domain: some warnings contain non-existing filenames
- #9456: html search: abbreation marks are inserted to the search result if failed to fetch the content of the page
- #9617: html search: The JS requirement warning is shown if browser is slow
- #9267: html theme: CSS and JS files added by theme were loaded twice
- #9585: py domain: :type: option for py:property directive does not create a hyperlink
- #9576: py domain: Literal typehint was converted to a cross reference
- #9535 comment: C++, fix parsing of defaulted function parameters that are function pointers.
- #9564: smartquotes: don't adjust typography for text with language-highlighted :code: role.
- #9512: sphinx-build: crashed with the HEAD of Python 3.10

Sphinx 4.1

Release 4.1.2 (released Jul 27, 2021)

Incompatible changes

• #9435: linkcheck: Disable checking automatically generated anchors on github.com (ex. anchors in reST/Markdown documents)

Bugs fixed

- #9489: autodoc: Custom types using typing. NewType are not displayed well with the HEAD of 3.10
- #9490: autodoc: Some objects under typing module are not displayed well with the HEAD of 3.10
- #9436, #9471: autodoc: crashed if autodoc_class_signature = "separated"
- #9456: html search: html_copy_source can't control the search summaries
- #9500: LaTeX: Failed to build Japanese document on Windows
- #9435: linkcheck: Failed to check anchors in github.com

Release 4.1.1 (released Jul 15, 2021)

Dependencies

- #9434: sphinxcontrib-htmlhelp-2.0.0 or above
- #9434: sphinxcontrib-serializinghtml-1.1.5 or above

• #9438: html: HTML logo or Favicon specified as file not being found on output

Release 4.1.0 (released Jul 12, 2021)

Dependencies

• Support jinja2-3.0

Deprecated

- The app argument of sphinx.environment.BuildEnvironment becomes required
- sphinx.application.Sphinx.html_theme
- sphinx.ext.autosummary._app
- sphinx.util.docstrings.extract_metadata()

Features added

- #8107: autodoc: Add class-doc-from option to autoclass directive to control the content of the specific class like autoclass_content
- #8588: autodoc: autodoc_type_aliases now supports dotted name. It allows you to define an alias for a class with module name like foo.bar.BazClass
- #9175: autodoc: Special member is not documented in the module
- #9195: autodoc: The arguments of typing.Literal are wrongly rendered
- #9185: autodoc: autodoc_typehints allows 'both' setting to allow typehints to be included both in the signature and description
- #4257: autodoc: Add autodoc_class_signature to separate the class entry and the definition of __init__()
 method
- #8061, #9218: autodoc: Support variable comment for alias classes
- #3014: autodoc: Add autodoc-process-bases to modify the base classes of the class definitions
- #9272: autodoc: Render enum values for the default argument value better
- #9384: autodoc: autodoc_typehints='none' now erases typehints for variables, attributes and properties
- #3257: autosummary: Support instance attributes for classes
- #9358: html: Add "heading" role to the toctree items
- #9225: html: Add span tag to the return typehint of method/function
- #9129: html search: Show search summaries when html_copy_source = False
- #9307: html search: Prevent corrections and completions in search field
- #9120: html theme: Eliminate prompt characters of code-block from copyable text
- #9176: i18n: Emit a debug message if message catalog file not found under locale_dirs
- #9414: LaTeX: Add xeCJKVerbAddon to default fvset config for Chinese documents
- #9016: linkcheck: Support checking anchors on github.com
- #9016: linkcheck: Add a new event linkcheck-process-uri to modify URIs before checking hyperlinks

- #6525: linkcheck: Add linkcheck_allowed_redirects to mark hyperlinks that are redirected to expected URLs as "working"
- #1874: py domain: Support union types using | in info-field-list
- #9268: py domain: python_use_unqualified_type_names supports type field in info-field-list
- #9097: Optimize the parallel build
- #9131: Add nitpick_ignore_regex to ignore nitpicky warnings using regular expressions
- #9174: Add Sphinx.set_html_assets_policy to tell extensions to include HTML assets in all the pages. Extensions can check this via Sphinx.registry.html_assets_policy
- C++, add support for
 - inline variables,
 - consteval functions,
 - constinit variables,
 - char8_t,
 - explicit(<constant expression>) specifier,
 - digit separators in literals, and
 - constraints in placeholder type specifiers, aka. adjective syntax (e.g., Sortable auto &v).
- C, add support for digit separators in literals.
- #9166: LaTeX: support containers in LaTeX output

- #8872: autodoc: stacked singledispatches are wrongly rendered
- #8597: autodoc: a docsting having metadata only should be treated as undocumented
- #9185: autodoc: typehints for overloaded functions and methods are inaccurate
- #9250: autodoc: The inherited method not having docstring is wrongly parsed
- #9283: autodoc: autoattribute directive failed to generate document for an attribute not having any comment
- #9364: autodoc: single element tuple on the default argument value is wrongly rendered
- #9362: autodoc: AttributeError is raised on processing a subclass of Tuple[()]
- #9404: autodoc: TypeError is raised on processing dict-like object (not a class) via autoclass directive
- #9317: html: Pushing left key causes visiting the next page at the first page
- #9381: html: URL for html_favicon and html_log does not work
- #9270: html theme : pyramid theme generates incorrect logo links
- #9217: manpage: The name of manpage directory that is generated by man_make_section_directory is not correct
- #9350: manpage: Fix font isn't reset after keyword at the top of samp role
- #9306: Linkcheck reports broken link when remote server closes the connection on HEAD request
- #9280: py domain: "exceptions" module is not displayed
- #9418: py domain: a Callable annotation with no parameters (e.g. Callable[[], None]) will be rendered with a bracket missing (Callable[], None])

- #9319: quickstart: Make sphinx-quickstart exit when conf.py already exists
- #9387: xml: XML Builder ignores custom visitors
- #9224: :param: and :type: fields does not support a type containing whitespace (ex. Dict[str, str])
- #8945: when transforming typed fields, call the specified role instead of making an single xref. For C and C++, use the expr role for typed fields.

Sphinx 4.0

Release 4.0.3 (released Jul 05, 2021)

Features added

- C, add C23 keywords _Decimal32, _Decimal64, and _Decimal128.
- #9354: C, add c_extra_keywords to allow user-defined keywords during parsing.
- Revert the removal of sphinx.util:force_decode() to become some 3rd party extensions available again during 5.0

Bugs fixed

- #9330: changeset domain: versionchanged with contents being a list will cause error during pdf build
- #9313: LaTeX: complex table with merged cells broken since 4.0
- #9305: LaTeX: backslash may cause Improper discretionary list pdf build error with Japanese engines
- #9354: C, remove special macro names from the keyword list. See also c_extra_keywords.
- #9322: KeyError is raised on PropagateDescDomain transform

Release 4.0.2 (released May 20, 2021)

Dependencies

• #9216: Support jinja2-3.0

Incompatible changes

- #9222: Update Underscore.js to 1.13.1
- #9217: manpage: Stop creating a section directory on build manpage by default (see man_make_section_directory)

Bugs fixed

- #9210: viewcode: crashed if non importable modules found on parallel build
- #9240: Unknown node error for pending_xref_condition is raised if an extension that does not support the node installs a missing-reference handler

Release 4.0.1 (released May 11, 2021)

- #9189: autodoc: crashed when ValueError is raised on generating signature from a property of the class
- #9188: autosummary: warning is emitted if list value is set to autosummary_generate

- #8380: html search: tags for search result are broken
- #9198: i18n: Babel emits errors when running compile_catalog
- #9205: py domain: The :canonical: option causes "more than one target for cross-reference" warning
- #9201: websupport: UndefinedError is raised: 'css_tag' is undefined

Release 4.0.0 (released May 09, 2021)

Dependencies

4.0.0b1

- Drop python 3.5 support
- Drop Docutils 0.12 and 0.13 support
- LaTeX: add tex-gyre font dependency

4.0.0b2

• Support Docutils 0.17. Please notice it changes the output of HTML builder. Some themes do not support it, and you need to update your custom CSS to upgrade it.

Incompatible changes

4.0.0b1

- #8539: autodoc: info-field-list is generated into the class description when autodoc_typehints='description' and autoclass_content='class' set
- #8898: extlinks: "%s" becomes required keyword in the link caption string
- domain: The Index class becomes subclasses of abc. ABC to indicate methods that must be overridden 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 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
- #8915: html theme: Emit a warning for sphinx_rtd_theme 0.2.4 or older
- #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

- #8937: Use explicit title instead of <no title>
- #8487: The :file: option for csv-table directive now recognizes an absolute path as a relative path from source directory

4.0.0b2

• #9023: Change the CSS classes on cpp:expr and cpp:texpr.

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.ext.autodoc.directive.DocumenterBridge.filename_set
- sphinx.ext.autodoc.directive.DocumenterBridge.warn()
- sphinx.registry.SphinxComponentRegistry.get_source_input()
- sphinx.registry.SphinxComponentRegistry.source_inputs
- sphinx.transforms.FigureAligner
- sphinx.util.pycompat.convert_with_2to3()
- sphinx.util.pycompat.execfile_()
- sphinx.util.smartypants
- sphinx.util.typing.DirectiveOption

Features added

4.0.0b1

- #8924: autodoc: Support bound argument for TypeVar
- #7383: autodoc: Support typehints for properties
- #5603: autodoc: Allow to refer to a python class using its canonical name when the class has two different names; a canonical name and an alias name
- #8539: autodoc: Add autodoc_typehints_description_target to control the behavior of autodoc_typehints=description
- #8841: autodoc: autodoc_docstring_signature will continue to look for multiple signature lines without backslash character
- #7549: autosummary: Enable autosummary_generate by default
- #8898: extlinks: Allow %s in link caption string
- #4826: py domain: Add:canonical: option to python directives to describe the location where the object is defined
- #7199: py domain: Add *python_use_unqualified_type_names* to suppress the module name of the python reference if it can be resolved (experimental)
- #7068: py domain: Add py:property directive to describe a property

- #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
- #9036: html theme: Allow to inherite the search page
- #8938: imgconverter: Show the error of the command availability check
- #7830: Add debug logs for change detection of sources and templates
- #8201: Emit a warning if toctree contains duplicated entries
- #8326: master_doc is now renamed to root_doc
- #8942: C++, add support for the C++20 spaceship operator, <=>.
- #7199: A new node, sphinx.addnodes.pending_xref_condition has been added. It can be used to choose appropriate content of the reference by conditions.

4.0.0b2

- #8818: autodoc: Super class having Any arguments causes nitpicky warning
- #9095: autodoc: TypeError is raised on processing broken metaclass
- #9110: autodoc: metadata of Generic Alias is not rendered as a reference in py37+
- #9098: html: copy-range protection for doctests doesn't work in Safari
- #9103: LaTeX: imgconverter: conversion runs even if not needed
- #8127: py domain: Ellipsis in info-field-list causes nitpicky warning
- #9121: py domain: duplicated warning is emitted when both canonical and its alias objects are defined on the document
- #9023: More CSS classes on domain descriptions, see *Doctree node classes added by Sphinx* for details.
- #8195: mathjax: Rename mathjax_config to mathjax2_config and add mathjax3_config

Bugs fixed

4.0.0b1

- #8917: autodoc: Raises a warning if function has wrong __globals__ value
- #8415: autodoc: a TypeVar imported from other module is not resolved (in Python 3.7 or above)
- #8992: autodoc: Failed to resolve types. Traceback Type type annotation
- #8905: html: html_add_permalinks=None and html_add_permalinks="" are ignored
- #8380: html search: Paragraphs in search results are not identified as
- #8915: html theme: The translation of sphinx_rtd_theme does not work
- #8342: Emit a warning if a unknown domain is given for directive or role (ex.:unknown:doc:)
- #7241: LaTeX: No wrapping for cpp:enumerator
- #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)
- #8881: LaTeX: The depth of bookmarks panel in PDF is not enough for navigation

- #8874: LaTeX: the fix to two minor Pygments LaTeXFormatter output issues ignore Pygments style
- #8925: LaTeX: 3.5.0 verbatimmaxunderfull setting does not work as expected
- #8980: LaTeX: missing line break in \pysigline
- #8995: LaTeX: legacy \pysiglinewithargsret does not compute correctly available horizontal space and should use a ragged right style
- #9009: LaTeX: "release" value with underscore leads to invalid LaTeX
- #8911: C++: remove the longest matching prefix in cpp_index_common_prefix instead of the first that matches
- C, properly reject function declarations when a keyword is used as parameter name.
- #8933: viewcode: Failed to create back-links on parallel build
- #8960: C and C++, fix rendering of (member) function pointer types in function parameter lists.
- C++, fix linking of names in array declarators, pointer to member (function) declarators, and in the argument to sizeof....
- C, fix linking of names in array declarators.

4.0.0b2

• C, C++, fix KeyError when an alias directive is the first C/C++ directive in a file with another C/C++ directive later.

4.0.0b3

• #9167: html: Failed to add CSS files to the specific page

Sphinx 3.5

Release 3.5.4 (released Apr 11, 2021)

Dependencies

• #9071: Restrict Docutils to 0.16

Bugs fixed

- #9078: autodoc: Async staticmethods and classmethods are considered as non async coroutine-functions with Python3.10
- #8870, #9001, #9051: html theme: The style are not applied with Docutils 0.17
 - toctree captions
 - The content of sidebar directive
 - figures

Release 3.5.3 (released Mar 20, 2021)

Features added

• #8959: using UNIX path separator in image directive confuses Sphinx on Windows

Release 3.5.2 (released Mar 06, 2021)

Bugs fixed

- #8943: i18n: Crashed by broken translation messages in ES, EL and HR
- #8936: LaTeX: A custom LaTeX builder fails with unknown node error
- #8952: Exceptions raised in a Directive cause parallel builds to hang

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

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
- #759: autodoc: Add a new configuration autodoc_preserve_defaults as an experimental feature. It preserves the default argument values of functions in source code and keep them not evaluated for readability.
- #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_sections
- #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.

- #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: questionnaire 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.

Sphinx 3.4

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()

Release 3.4.2 (released Jan 04, 2021)

- #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

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 comparison via __eq_ method
- #8565: linkcheck: Fix PriorityQueue crash when link tuples are not comparable

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_rate_limit_timeout for details.

- #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 uninitialized 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.

Sphinx 3.3

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

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

Sphinx 3.2

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 parameters, broken by the fix of #7944

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_param 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

- #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 parameters declaration in numpydoc was wrongly recognized when napoleon_use_param=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.

Sphinx 3.1

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"

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

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.

- #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-process-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.

Sphinx 3.0

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

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

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

Release 3.0.1 (released Apr 11, 2020)

Incompatible changes

• #7418: std domain: term role becomes case sensitive

- #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

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 re-enable 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 ancestor 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 signature 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 entity, 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

Sphinx 2.4

Release 2.4.5 (released Nov 18, 2021)

Dependencies

• #9807: Restrict Docutils to 0.17.x or older

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

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

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

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

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_ast¹⁵⁴³ is required)
- #7051: autodoc: Support instance variables without defaults (PEP-526)

¹⁵⁴³ https://github.com/python/typed_ast

- #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

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 encoded 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

Sphinx 2.3

Release 2.3.1 (released Dec 22, 2019)

Bugs fixed

• #6936: sphinx-autogen: raises AttributeError

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 gettext_language_team to customize headers of POT file

Bugs fixed

- #6668: LaTeX: Longtable before header has incorrect distance (refs: latex3/latex2e#173¹⁵⁴⁴)
- #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)

1544 https://github.com/latex3/latex2e/issues/173

- #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
- #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

Sphinx 2.2

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

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

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 accessibility 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 rebuilt

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

Sphinx 2.1

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

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

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 %0% 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 rst:directive:option directive to describe the option for directive
- #6306: html: Add a label to search form for accessibility 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)

Sphinx 2.0

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

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 <code>latex_engine</code> set to <code>'xelatex'</code> or to <code>'lualatex'</code> requires (by default) the <code>FreeFont</code> fonts, which in Ubuntu xenial are provided by package <code>fonts-freefont-otf</code>, and e.g. in Fedora 29 via package <code>texlive-gnu-freefont</code>.
- 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 <code>latex_elements</code> 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.build_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.
- $\bullet \ \ \text{The importer argument of sphinx.ext.autodoc.importer._MockModule}$

- 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 results¹⁵⁴⁵ is no longer necessary. Note that changes to the search function of your custom or 3rd-party HTML template might overwrite this improvement.

 $^{^{1545}\} https://github.com/sphinx-contrib/sphinx-pretty-searchresults$

- #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

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()

Sphinx 1.8

Release 1.8.6 (released Nov 18, 2021)

Dependencies

• #9807: Restrict Docutils to 0.17.x or older

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

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

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

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

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

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
- $\bullet \ \ sphinx.writers.latex.LaTeXTranslator.restrict_footnote() \ is \ deprecated$
- $\bullet \ \, \text{sphinx.writers.latex.LaTeXTranslator.unrestrict_footnote()} \ is \ deprecated$
- 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
- $\bullet \ \ \text{sphinx.writers.html} \\ \text{15.HTMLTranslator.highlightlangopts} \ \text{is deprecated} \\$

- 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.egref 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.

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
- 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.

Sphinx 1.7

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

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

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 <code>latex_engine</code> 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

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

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

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

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

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

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.

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

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.

Sphinx 1.6

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

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 smartguotes_action and smartguotes_excludes (refs: #4142, #4357)

- #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:

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)

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

Release 1.6.3 (released Jul 02, 2017)

Features added

• latex: hint that code-block continues on next page (refs: #3764, #3792)

- #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

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: RTD¹⁵⁴⁶ PDF builds of Sphinx own docs are missing an index entry in the bookmarks and table of contents. This is rtfd/readthedocs.org#2857¹⁵⁴⁷ 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.

¹⁵⁴⁶ https://readthedocs.org/

¹⁵⁴⁷ https://github.com/rtfd/readthedocs.org/issues/2857

- #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

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)

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)

Release 1.6 (unreleased)

• not released (because of package script error)

Sphinx 1.5

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

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

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)

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

- #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: 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

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 ¹⁵⁴⁸ 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)

 $^{^{1548}\} https://docutils.sourceforge.io/docs/ref/rst/directives.html \# parsed-literal$

- 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
- LaTeX, pass dvipdfm option to geometry package for Japanese documents (ref #3363)
- Fix parselinenos() could not parse left half open range (cf. "-4")

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.

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 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.
- $\bullet \ \ Sphinx\ does\ not\ ship\ anymore\ a\ custom\ modified\ LaTeX\ style\ file\ {\tt tabulary}.\ The\ non-modified\ package\ is\ used.$
- #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 1inkcheck_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.

Sphinx 1.4

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

Release 1.4.8 (released Oct 1, 2016)

Bugs fixed

• #2996: The wheel package of Sphinx got crash with ImportError

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

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

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.

- 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.

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

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

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).

- #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

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.

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 Language¹⁵⁴⁹ 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
- #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

¹⁵⁴⁹ https://unicode.org/reports/tr35/tr35-dates.html#Date_Format_Patterns

- #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.

- #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.

Sphinx 1.3

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

Release 1.3.5 (released Jan 24, 2016)

- 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)

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 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
- #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

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.

Release 1.3.2 (released Nov 29, 2015)

Features added

• #1935: Make "numfig_format" overridable in latex_elements.

- #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 Type-Error
- #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.

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.

Release 1.3 (released Mar 10, 2015)

Incompatible changes

- Roles ref, term and menusel now don't generate emphasis 1550 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.

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.

- 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.

¹⁵⁵⁰ https://docutils.sourceforge.io/docs/ref/rst/roles.html#emphasis

- #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.

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

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.

- 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.

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 1551 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.

1551 https://docutils.sourceforge.io/docs/ref/rst/directives.html#default-role

- 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 is and underscore is 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.
- 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 literalinelude 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)

Sphinx 1.2

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.

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.

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 1552 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.

¹⁵⁵² https://docutils.sourceforge.io/docs/ref/rst/directives.html#figure

- #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 tags in the HTML writer, which is not valid HTML.
- #1147: Don't emit a sidebar search box in the "singlehtml" builder.

Documentation

• #1325: Added a "Intersphinx" tutorial section. (doc/tutorial.rst)

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.

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.

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.

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 literals. 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 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.

Sphinx 1.1

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.

Release 1.1.2 (Nov 1, 2011) – 1.1.1 is a silly version number anyway!

• #809: Include custom fixers in the source distribution.

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.

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.

Sphinx 1.0

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.

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.

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.

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.

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.

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.

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.

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.

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 include¹⁵⁵³ 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.
 - 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.

¹⁵⁵³ https://docutils.sourceforge.io/docs/ref/rst/directives.html#include

- 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.

Sphinx 0.6

Release 0.6.7 (Jun 05, 2010)

- #440: Remove usage of a Python >= 2.5 API in the literalinclude directive.
- Fix a bug that prevented some references being generated in the LaTeX builder.
- #428: Add some missing CSS styles for standard Docutils classes.
- #432: Fix UnicodeErrors while building LaTeX in translated locale.

Release 0.6.6 (May 25, 2010)

- Handle raw nodes in the text writer.
- Fix a problem the Qt help project generated by the qthelp builder that would lead to no content being displayed in the Qt Assistant.
- #393: Fix the usage of Unicode characters in mathematic formulas when using the pngmath extension.

- #404: Make \and work properly in the author field of the latex_documents setting.
- #409: Make the highlight_language config value work properly in the LaTeX builder.
- #418: Allow relocation of the translation JavaScript files to the system directory on Unix systems.
- #414: Fix handling of Windows newlines in files included with the literalinclude directive.
- #377: Fix crash in linkcheck builder.
- #387: Fix the display of search results in dirhtml output.
- #376: In autodoc, fix display of parameter defaults containing backslashes.
- #370: Fix handling of complex list item labels in LaTeX output.
- #374: Make the doctest_path config value of the doctest extension actually work.
- Fix the handling of multiple toctrees when creating the global TOC for the toctree() template function.
- Fix the handling of hidden toctrees when creating the global TOC for the toctree() template function.
- Fix the handling of nested lists in the text writer.
- #362: In autodoc, check for the existence of __self__ on function objects before accessing it.
- #353: Strip leading and trailing whitespace when extracting search words in the search function.

Release 0.6.5 (Mar 01, 2010)

- In autodoc, fix the omission of some module members explicitly documented using documentation comments.
- #345: Fix cropping of sidebar scroll bar with stickysidebar option of the default theme.
- #341: Always generate UNIX newlines in the quickstart Makefile.
- #338: Fix running with -C under Windows.
- In autodoc, allow customizing the signature of an object where the built-in mechanism fails.
- #331: Fix output for enumerated lists with start values in LaTeX.
- Make the start-after and end-before options to the literalinclude directive work correctly if not used together.
- #321: Fix link generation in the LaTeX builder.

Release 0.6.4 (Jan 12, 2010)

- Improve the handling of non-Unicode strings in the configuration.
- #316: Catch OSErrors occurring when calling graphviz with arguments it doesn't understand.
- Restore compatibility with Pygments >= 1.2.
- #295: Fix escaping of hyperref targets in LaTeX output.
- #302: Fix links generated by the :doc: role for LaTeX output.
- #286: collect todo nodes after the whole document has been read; this allows placing substitution references in todo items.
- #294: do not ignore an explicit today config value in a LaTeX build.
- The alt text of inheritance diagrams is now much cleaner.
- Ignore images in section titles when generating link captions.
- #310: support exception messages in the testoutput blocks of the doctest extension.

- #293: line blocks are styled properly in HTML output.
- #285: make the locale_dirs config value work again.
- #303: html_context values given on the command line via -A should not override other values given in conf.py.
- Fix a bug preventing incremental rebuilds for the dirhtml builder.
- #299: Fix the mangling of quotes in some literal blocks.
- #292: Fix path to the search index for the dirhtml builder.
- Fix a Jython compatibility issue: make the dependence on the parser module optional.
- #238: In autodoc, catch all errors that occur on module import, not just ImportError.
- Fix the handling of non-data, but non-method descriptors in autodoc.
- When copying file times, ignore OSErrors raised by os.utime().

Release 0.6.3 (Sep 03, 2009)

- Properly add C module filenames as dependencies in autodoc.
- #253: Ignore graphviz directives without content instead of raising an unhandled exception.
- #241: Fix a crash building LaTeX output for documents that contain a todolist directive.
- #252: Make it easier to change the build dir in the Makefiles generated by quickstart.
- #220: Fix CSS so that displaymath really is centered.
- #222: Allow the "Footnotes" header to be translated.
- #225: Don't add whitespace in generated HTML after inline tags.
- #227: Make literalinclude work when the document's path name contains non-ASCII characters.
- #229: Fix autodoc failures with members that raise errors on getattr().
- #205: When copying files, don't copy full stat info, only modification times.
- #232: Support non-ASCII metadata in Qt help builder.
- Properly format bullet lists nested in definition lists for LaTeX.
- Section titles are now allowed inside only directives.
- #201: Make centered directive work in LaTeX output.
- #206: Refuse to overwrite an existing master document in sphinx-quickstart.
- #208: Use MS-sanctioned locale settings, determined by the language config option, in the HTML help builder.
- #210: Fix nesting of HTML tags for displayed math from pngmath extension.
- #213: Fix centering of images in LaTeX output.
- #211: Fix compatibility with Docutils 0.5.

Release 0.6.2 (Jun 16, 2009)

- #130: Fix obscure IndexError in doctest extension.
- #167: Make glossary sorting case-independent.
- #196: Add a warning if an extension module doesn't have a setup() function.
- #158: Allow '..' in template names, and absolute template paths; Jinja 2 by default disables both.

- When highlighting Python code, ignore extra indentation before trying to parse it as Python.
- #191: Don't escape the tilde in URIs in LaTeX.
- Don't consider contents of source comments for the search index.
- Set the default encoding to utf-8-sig to handle files with a UTF-8 BOM correctly.
- #178: apply add_function_parentheses config value to C functions as promised.
- #173: Respect the Docutils title directive.
- #172: The obj role now links to modules as promised.
- #19: Tables now can have a "longtable" class, in order to get correctly broken into pages in LaTeX output.
- Look for Sphinx message catalogs in the system default path before trying sphinx/locale.
- Fix the search for methods via "classname.methodname".
- #155: Fix Python 2.4 compatibility: exceptions are old-style classes there.
- #150: Fix display of the "sphinxdoc" theme on Internet Explorer versions 6 and 7.
- #146: Don't fail to generate LaTeX when the user has an active .docutils configuration.
- #29: Don't generate visible "-{-}" in option lists in LaTeX.
- Fix cross-reference roles when put into substitutions.
- Don't put image "alt" text into table-of-contents entries.
- In the LaTeX writer, do not raise an exception on too many section levels, just use the "subparagraph" level for all of them.
- #145: Fix autodoc problem with automatic members that refuse to be getattr()'d from their parent.
- If specific filenames to build are given on the command line, check that they are within the source directory.
- Fix autodoc crash for objects without a __name__.
- Fix intersphinx for installations without urllib2.HTTPSHandler.
- #134: Fix pending_xref leftover nodes when using the todolist directive from the todo extension.

Release 0.6.1 (Mar 26, 2009)

- #135: Fix problems with LaTeX output and the graphviz extension.
- #132: Include the autosummary "module" template in the distribution.

Release 0.6 (Mar 24, 2009)

New features added

- Incompatible changes:
 - Templating now requires the Jinja2 library, which is an enhanced version of the old Jinja1 engine. Since the syntax and semantic is largely the same, very few fixes should be necessary in custom templates.
 - The "document" div tag has been moved out of the layout.html template's "document" block, because
 the closing tag was already outside. If you overwrite this block, you need to remove your "document" div
 tag as well.
 - The autodoc_skip_member event now also gets to decide whether to skip members whose name starts with underscores. Previously, these members were always automatically skipped. Therefore, if you handle this event, add something like this to your event handler to restore the old behavior:

```
if name.startswith('_'):
    return True
```

- Theming support, see the new section in the documentation.
- Markup:
 - Due to popular demand, added a :doc: role which directly links to another document without the need of creating a label to which a :ref: could link to.
 - #4: Added a :download: role that marks a non-document file for inclusion into the HTML output and links to it.
 - Added an only directive that can selectively include text based on enabled "tags". Tags can be given on the command line. Also, the current builder output format (e.g. "html" or "latex") is always a defined tag.
 - #10: Added HTML section numbers, enabled by giving a :numbered: flag to the toctree directive.
 - #114: Added an abbr role to markup abbreviations and acronyms.
 - The literalinclude directive now supports several more options, to include only parts of a file.
 - The toctree directive now supports a :hidden: flag, which will prevent links from being generated in place of the directive this allows you to define your document structure, but place the links yourself.
 - #123: The glossary directive now supports a :sorted: flag that sorts glossary entries alphabetically.
 - Paths to images, literal include files and download files can now be absolute (like /images/foo.png).
 They are treated as relative to the top source directory.
 - + #52: There is now a hlist directive, creating a compact list by placing distributing items into multiple columns.
 - #77: If a description environment with info field list only contains one :param: entry, no bullet list is generated.
 - #6: Don't generate redundant
 for top-level TOC tree items, which leads to a visual separation of TOC entries.
 - #23: Added a classmethod directive along with method and staticmethod.
 - Scaled images now get a link to the unscaled version.
 - SVG images are now supported in HTML (via <object> and <embed> tags).
 - Added a toctree callable to the templates, and the ability to include external links in toctrees. The 'collapse' keyword argument indicates whether or not to only display subitems of the current page. (Defaults to True.)

· Configuration:

- The new config value rst_epilog can contain reST that is appended to each source file that is read. This
 is the right place for global substitutions.
- The new html_add_permalinks config value can be used to switch off the generated "paragraph sign" permalinks for each heading and definition environment.
- The new html_show_sourcelink config value can be used to switch off the links to the reST sources in the sidebar.
- The default value for htmlhelp_basename is now the project title, cleaned up as a filename.
- The new modindex_common_prefix config value can be used to ignore certain package names for module index sorting.

- The new trim_footnote_reference_space config value mirrors the Docutils config value of the same name and removes the space before a footnote reference that is necessary for reST to recognize the reference.
- The new latex_additional_files config value can be used to copy files (that Sphinx doesn't copy automatically, e.g. if they are referenced in custom LaTeX added in latex_elements) to the build directory.

• Builders:

- The HTML builder now stores a small file named .buildinfo in its output directory. It stores a hash of config values that can be used to determine if a full rebuild needs to be done (e.g. after changing html_theme).
- New builder for Qt help collections, by Antonio Valentino.
- The new DirectoryHTMLBuilder (short name dirhtml) creates a separate directory for every page, and places the page there in a file called index.html. Therefore, page URLs and links don't need to contain .html.
- The new html_link_suffix config value can be used to select the suffix of generated links between HTML files.
- #96: The LaTeX builder now supports figures wrapped by text, when using the figwidth option and right/left alignment.

• New translations:

- Italian by Sandro Dentella.
- Ukrainian by Petro Sasnyk.
- Finnish by Jukka Inkeri.
- Russian by Alexander Smishlajev.

• Extensions and API:

- New graphviz extension to embed graphviz graphs.
- New inheritance_diagram extension to embed... inheritance diagrams!
- New autosummary extension that generates summaries of modules and automatic documentation of modules.
- Autodoc now has a reusable Python API, which can be used to create custom types of objects to autodocument (e.g. Zope interfaces). See also Sphinx.add_autodocumenter().
- Autodoc now handles documented attributes.
- Autodoc now handles inner classes and their methods.
- Autodoc can document classes as functions now if explicitly marked with autofunction.
- Autodoc can now exclude single members from documentation via the exclude-members option.
- Autodoc can now order members either alphabetically (like previously) or by member type; configurable either with the config value autodoc_member_order or a member-order option per directive.
- The function Sphinx.add_directive() now also supports Docutils 0.5-style directive classes. If they inherit from sphinx.util.compat.Directive, they also work with Docutils 0.4.
- There is now a Sphinx.add_lexer() method to be able to use custom Pygments lexers easily.
- There is now Sphinx.add_generic_role() to mirror the Docutils' own function.

· Other changes:

- Config overrides for single dict keys can now be given on the command line.

- There is now a doctest_global_setup config value that can be used to give setup code for all doctests in the documentation.
- Source links in HTML are now generated with rel="nofollow".
- Quickstart can now generate a Windows make.bat file.
- #62: There is now a -w option for sphinx-build that writes warnings to a file, in addition to stderr.
- There is now a -W option for sphinx-build that turns warnings into errors.

Sphinx 0.5

Release 0.5.2 (Mar 24, 2009)

- Properly escape | in LaTeX output.
- #71: If a decoding error occurs in source files, print a warning and replace the characters by "?".
- Fix a problem in the HTML search if the index takes too long to load.
- Don't output system messages while resolving, because they would stay in the doctrees even if keep_warnings is
 false.
- #82: Determine the correct path for dependencies noted by docutils. This fixes behavior where a source with dependent files was always reported as changed.
- Recognize toctree directives that are not on section toplevel, but within block items, such as tables.
- Use a new RFC base URL, since rfc.org seems down.
- Fix a crash in the todolist directive when no todo items are defined.
- Don't call LaTeX or dvipng over and over again if it was not found once, and use text-only latex as a substitute in that case.
- Fix problems with footnotes in the LaTeX output.
- Prevent double hyphens becoming en-dashes in literal code in the LaTeX output.
- Open literalinclude files in universal newline mode to allow arbitrary newline conventions.
- Actually make the -Q option work.
- #86: Fix explicit document titles in toctrees.
- #81: Write environment and search index in a manner that is safe from exceptions that occur during dumping.
- #80: Fix UnicodeErrors when a locale is set with setlocale().

Release 0.5.1 (Dec 15, 2008)

- #67: Output warnings about failed doctests in the doctest extension even when running in quiet mode.
- #72: In pngmath, make it possible to give a full path to LaTeX and dvipng on Windows. For that to work, the pngmath_latex and pngmath_dvipng options are no longer split into command and additional arguments; use pngmath_latex_args and pngmath_dvipng_args to give additional arguments.
- Don't crash on failing doctests with non-ASCII characters.
- Don't crash on writing status messages and warnings containing unencodable characters.
- Warn if a doctest extension block doesn't contain any code.
- Fix the handling of :param: and :type: doc fields when they contain markup (especially cross-referencing roles).

- #65: Fix storage of depth information for PNGs generated by the pngmath extension.
- Fix autodoc crash when automethod is used outside a class context.
- #68: Fix LaTeX writer output for images with specified height.
- #60: Fix wrong generated image path when including images in sources in subdirectories.
- Fix the JavaScript search when html copy source is off.
- Fix an indentation problem in autodoc when documenting classes with the option autoclass_content =
 "both" set.
- Don't crash on empty index entries, only emit a warning.
- Fix a typo in the search JavaScript code, leading to unusable search function in some setups.

Release 0.5 (Nov 23, 2008) - Birthday release!

New features added

- Markup features:
 - Citations are now global: all citation defined in any file can be referenced from any file. Citations are collected in a bibliography for LaTeX output.
 - Footnotes are now properly handled in the LaTeX builder: they appear at the location of the footnote reference in text, not at the end of a section. Thanks to Andrew McNamara for the initial patch.
 - "System Message" warnings are now automatically removed from the built documentation, and only written to stderr. If you want the old behavior, set the new config value keep_warnings to True.
 - Glossary entries are now automatically added to the index.
 - Figures with captions can now be referred to like section titles, using the :ref: role without an explicit link text.
 - Added cmember role for consistency.
 - Lists enumerated by letters or roman numerals are now handled like in standard reST.
 - The seealso directive can now also be given arguments, as a short form.
 - You can now document several programs and their options with the new program directive.
- HTML output and templates:
 - Incompatible change: The "root" relation link (top left in the relbar) now points to the master_doc by default, no longer to a document called "index". The old behavior, while useful in some situations, was somewhat unexpected. Override the "rootrellink" block in the template to customize where it refers to.
 - The JavaScript search now searches for objects before searching in the full text.
 - TOC tree entries now have CSS classes that make it possible to style them depending on their depth.
 - Highlighted code blocks now have CSS classes that make it possible to style them depending on their language.
 - HTML <meta> tags via the Docutils meta¹⁵⁵⁴ directive are now supported.
 - SerializingHTMLBuilder was added as new abstract builder that can be subclassed to serialize build
 HTML in a specific format. The PickleHTMLBuilder is a concrete subclass of it that uses pickle as
 serialization implementation.

¹⁵⁵⁴ https://docutils.sourceforge.io/docs/ref/rst/directives.html#meta

- JSONHTMLBuilder was added as another SerializingHTMLBuilder subclass that dumps the generated HTML into JSON files for further processing.
- The rellinks block in the layout template is now called linktags to avoid confusion with the relbar links.
- The HTML builders have two additional attributes now that can be used to disable the anchor-link creation
 after headlines and definition links.
- Only generate a module index if there are some modules in the documentation.
- New and changed config values:
 - Added support for internationalization in generated text with the language and locale_dirs config values. Many thanks to language contributors:
 - * Horst Gutmann German
 - * Pavel Kosina Czech
 - * David Larlet French
 - * Michał Kandulski Polish
 - Yasushi Masuda Japanese
 - * Guillem Borrell Spanish
 - * Luc Saffre and Peter Bertels Dutch
 - * Fred Lin Traditional Chinese
 - * Roger Demetrescu Brazilian Portuguese
 - * Rok Garbas Slovenian
 - The new config value highlight_language set a global default for highlighting. When 'python3' is selected, console output blocks are recognized like for 'python'.
 - Exposed Pygments' lexer guessing as a highlight "language" guess.
 - The new config value latex_elements allows to override all LaTeX snippets that Sphinx puts into the generated .tex file by default.
 - Added exclude_dirnames config value that can be used to exclude e.g. CVS directories from source file search.
 - Added source_encoding config value to select input encoding.

• Extensions:

- The new extensions sphinx.ext.jsmath and sphinx.ext.pngmath provide math support for both HTML and LaTeX builders.
- The new extension sphinx.ext.intersphinx half-automatically creates links to Sphinx documentation of Python objects in other projects.
- The new extension sphinx.ext.todo allows the insertion of "To do" directives whose visibility in the output can be toggled. It also adds a directive to compile a list of all todo items.
- sphinx.ext.autodoc has a new event autodoc-process-signature that allows tuning function signature introspection.
- sphinx.ext.autodoc has a new event autodoc-skip-member that allows tuning which members are included in the generated content.
- Respect __all__ when autodocumenting module members.

- The automodule directive now supports the synopsis, deprecated and platform options.
- Extension API:
 - Sphinx.add_node() now takes optional visitor methods for the HTML, LaTeX and text translators; this prevents having to manually patch the classes.
 - Added Sphinx.add_javascript() that adds scripts to load in the default HTML template.
 - Added new events: source-read, env-updated, env-purge-doc, missing-reference, build-finished.
- Other changes:
 - Added a command-line switch -Q: it will suppress warnings.
 - Added a command-line switch -A: it can be used to supply additional values into the HTML templates.
 - Added a command-line switch -C: if it is given, no configuration file conf.py is required.
 - Added a distutils command build_sphinx: When Sphinx is installed, you can call python setup.py build_sphinx for projects that have Sphinx documentation, which will build the docs and place them in the standard distutils build directory.
 - In quickstart, if the selected root path already contains a Sphinx project, complain and abort.

Bugs fixed

- #51: Escape configuration values placed in HTML templates.
- #44: Fix small problems in HTML help index generation.
- Fix LaTeX output for line blocks in tables.
- #38: Fix "illegal unit" error when using pixel image widths/heights.
- Support table captions in LaTeX output.
- #39: Work around a bug in Jinja that caused "<generator ...>" to be emitted in HTML output.
- Fix a problem with module links not being generated in LaTeX output.
- Fix the handling of images in different directories.
- #29: Support option lists in the text writer. Make sure that dashes introducing long option names are not contracted to en-dashes.
- Support the "scale" option for images in HTML output.
- #25: Properly escape quotes in HTML help attribute values.
- Fix LaTeX build for some description environments with :noindex:.
- #24: Don't crash on uncommon casing of role names (like :Class:).
- Only output ANSI colors on color terminals.
- Update to newest fncychap.sty, to fix problems with non-ASCII characters at the start of chapter titles.
- Fix a problem with index generation in LaTeX output, caused by hyperref not being included last.
- Don't disregard return annotations for functions without any parameters.
- Don't throw away labels for code blocks.

Sphinx 0.4

Release 0.4.3 (Oct 8, 2008)

- Fix a bug in autodoc with directly given autodoc members.
- Fix a bug in autodoc that would import a module twice, once as "module", once as "module.".
- Fix a bug in the HTML writer that created duplicate id attributes for section titles with Docutils 0.5.
- Properly call super() in overridden blocks in templates.
- Add a fix when using XeTeX.
- Unify handling of LaTeX escaping.
- Rebuild everything when the extensions config value changes.
- Don't try to remove a nonexisting static directory.
- Fix an indentation problem in production lists.
- Fix encoding handling for literal include files: literalinclude now has an encoding option that defaults to UTF-8.
- Fix the handling of non-ASCII characters entered in quickstart.
- Fix a crash with nonexisting image URIs.

Release 0.4.2 (Jul 29, 2008)

- Fix rendering of the samp role in HTML.
- Fix a bug with LaTeX links to headings leading to a wrong page.
- Reread documents with globbed toctrees when source files are added or removed.
- Add a missing parameter to PickleHTMLBuilder.handle_page().
- Put inheritance info always on its own line.
- Don't automatically enclose code with whitespace in it in quotes; only do this for the samp role.
- · autodoc now emits a more precise error message when a module can't be imported or an attribute can't be found.
- The JavaScript search now uses the correct file name suffix when referring to found items.
- The automodule directive now accepts the inherited-members and show-inheritance options again.
- You can now rebuild the docs normally after relocating the source and/or doctree directory.

Release 0.4.1 (Jul 5, 2008)

- Added sub-/superscript node handling to TextBuilder.
- Label names in references are now case-insensitive, since reST label names are always lowercased.
- Fix linkcheck builder crash for malformed URLs.
- Add compatibility for admonitions and Docutils 0.5.
- Remove the silly restriction on "rubric" in the LaTeX writer: you can now write arbitrary "rubric" directives, and only those with a title of "Footnotes" will be ignored.
- Copy the HTML logo to the output _static directory.
- Fix LaTeX code for modules with underscores in names and platforms.
- Fix a crash with nonlocal image URIs.

- Allow the usage of :noindex: in automodule directives, as documented.
- Fix the delete() docstring processor function in autodoc.
- Fix warning message for nonexisting images.
- Fix JavaScript search in Internet Explorer.

Release 0.4 (Jun 23, 2008)

New features added

- tocdepth can be given as a file-wide metadata entry, and specifies the maximum depth of a TOC of this file.
- The new config value default_role can be used to select the default role for all documents.
- Sphinx now interprets field lists with fields like :param foo: in description units.
- The new staticmethod directive can be used to mark methods as static methods.
- HTML output:
 - The "previous" and "next" links have a more logical structure, so that by following "next" links you can traverse the entire TOC tree.
 - The new event html-page-context can be used to include custom values into the context used when rendering an HTML template.
 - Document metadata is now in the default template context, under the name metadata.
 - The new config value html_favicon can be used to set a favicon for the HTML output. Thanks to Sebastian Wiesner.
 - The new config value html_use_index can be used to switch index generation in HTML documents off.
 - The new config value html_split_index can be used to create separate index pages for each letter, to be used when the complete index is too large for one page.
 - The new config value html_short_title can be used to set a shorter title for the documentation which
 is then used in the navigation bar.
 - The new config value html_show_sphinx can be used to control whether a link to Sphinx is added to the HTML footer.
 - The new config value html_file_suffix can be used to set the HTML file suffix to e.g. .xhtml.
 - The directories in the html_static_path can now contain subdirectories.
 - The module index now isn't collapsed if the number of submodules is larger than the number of toplevel
 modules.
- The image directive now supports specifying the extension as .*, which makes the builder select the one that matches best. Thanks to Sebastian Wiesner.
- The new config value exclude_trees can be used to exclude whole subtrees from the search for source files.
- Defaults for configuration values can now be callables, which allows dynamic defaults.
- The new TextBuilder creates plain-text output.
- Python 3-style signatures, giving a return annotation via ->, are now supported.
- Extensions:
 - The autodoc extension now offers a much more flexible way to manipulate docstrings before including them into the output, via the new autodoc-process-docstring event.

- The autodoc extension accepts signatures for functions, methods and classes now that override the signature got via introspection from Python code.
- The autodoc extension now offers a show-inheritance option for autoclass that inserts a list of bases after the signature.
- The autodoc directives now support the noindex flag option.

Bugs fixed

- Correctly report the source location for docstrings included with autodoc.
- Fix the LaTeX output of description units with multiple signatures.
- Handle the figure directive in LaTeX output.
- Handle raw admonitions in LaTeX output.
- Fix determination of the title in HTML help output.
- Handle project names containing spaces.
- Don't write SSI-like comments in HTML output.
- Rename the "sidebar" class to "sphinxsidebar" in order to stay different from reST sidebars.
- Use a binary TOC in HTML help generation to fix issues links without explicit anchors.
- Fix behavior of references to functions/methods with an explicit title.
- Support citation, subscript and superscript nodes in LaTeX writer.
- Provide the standard "class" directive as "cssclass"; else it is shadowed by the Sphinx-defined directive.
- Fix the handling of explicit module names given to autoclass directives. They now show up with the correct module name in the generated docs.
- Enable autodoc to process Unicode docstrings.
- The LaTeX writer now translates line blocks with \raggedright, which plays nicer with tables.
- Fix bug with directories in the HTML builder static path.

Sphinx 0.3

Release 0.3 (May 6, 2008)

New features added

- The toctree directive now supports a glob option that allows glob-style entries in the content.
- If the pygments_style config value contains a dot it's treated as the import path of a custom Pygments style
 class.
- A new config value, exclude_dirs, can be used to exclude whole directories from the search for source files.
- The configuration directory (containing conf.py) can now be set independently from the source directory. For that, a new command-line option -c has been added.
- A new directive tabularcolumns can be used to give a tabular column specification for LaTeX output. Tables now use the tabulary package. Literal blocks can now be placed in tables, with several caveats.
- A new config value, latex_use_parts, can be used to enable parts in LaTeX documents.
- Autodoc now skips inherited members for classes, unless you give the new inherited-members option.

- A new config value, autoclass_content, selects if the docstring of the class' __init__ method is added to the directive's body.
- Support for C++ class names (in the style Class::Function) in C function descriptions.
- Support for a toctree_only item in items for the latex_documents config value. This only includes the documents referenced by TOC trees in the output, not the rest of the file containing the directive.

Bugs fixed

- sphinx.htmlwriter: Correctly write the TOC file for any structure of the master document. Also encode non-ASCII characters as entities in TOC and index file. Remove two remaining instances of hard-coded "documentation".
- sphinx.ext.autodoc: descriptors are detected properly now.
- sphinx.latexwriter: implement all reST admonitions, not just note and warning.
- Lots of little fixes to the LaTeX output and style.
- Fix OpenSearch template and make template URL absolute. The html_use_opensearch config value now
 must give the base URL.
- Some unused files are now stripped from the HTML help file build.

Sphinx 0.2

Release 0.2 (Apr 27, 2008)

Incompatible changes

- Jinja, the template engine used for the default HTML templates, is now no longer shipped with Sphinx. If it is not installed automatically for you (it is now listed as a dependency in setup.py), install it manually from PyPI. This will also be needed if you're using Sphinx from a SVN checkout; in that case please also remove the sphinx/jinja directory that may be left over from old revisions.
- The clumsy handling of the index.html template was removed. The config value html_index is gone, and html_additional_pages should be used instead. If you need it, the old index.html template is still there, called defindex.html, and you can port your html_index template, using Jinja inheritance, by changing your template:

```
{% extends "defindex.html" %}
{% block tables %}
... old html_index template content ...
{% endblock %}
```

and putting 'index': name of your template in html_additional_pages.

• In the layout template, redundant blocks were removed; you should use Jinja's standard {{ super() }} mechanism instead, as explained in the (newly written) templating docs.

New features added

- Extension API (Application object):
 - Support a new method, add_crossref_type. It works like add_description_unit but the directive
 will only create a target and no output.
 - Support a new method, add_transform. It takes a standard Docutils Transform subclass which is then
 applied by Sphinx's reader on parsing reST document trees.

- Add support for other template engines than Jinja, by adding an abstraction called a "template bridge".
 This class handles rendering of templates and can be changed using the new configuration value "template bridge".
- The config file itself can be an extension (if it provides a setup() function).

• Markup:

- New directive, currentmodule. It can be used to indicate the module name of the following documented things without creating index entries.
- Allow giving a different title to documents in the toctree.
- Allow giving multiple options in a cmdoption directive.
- Fix display of class members without explicit class name given.
- Templates (HTML output):
 - index.html renamed to defindex.html, see above.
 - There's a new config value, html_title, that controls the overall "title" of the set of Sphinx docs. It is used instead everywhere instead of "Projectname vX.Y documentation" now.
 - All references to "documentation" in the templates have been removed, so that it is now easier to use Sphinx for non-documentation documents with the default templates.
 - Templates now have an XHTML doctype, to be consistent with Docutils' HTML output.
 - You can now create an OpenSearch description file with the html_use_opensearch config value.
 - You can now quickly include a logo in the sidebar, using the html_logo config value.
 - There are new blocks in the sidebar, so that you can easily insert content into the sidebar.

• LaTeX output:

- The sphinx.sty package was cleaned of unused stuff.
- You can include a logo in the title page with the latex_logo config value.
- You can define the link colors and a border and background color for verbatim environments.

Thanks to Jacob Kaplan-Moss, Talin, Jeroen Ruigrok van der Werven and Sebastian Wiesner for suggestions.

Bugs fixed

- sphinx.ext.autodoc: Don't check __module__ for explicitly given members. Remove "self" in class constructor argument list.
- sphinx.htmlwriter: Don't use os.path for joining image HREFs.
- sphinx.htmlwriter: Don't use SmartyPants for HTML attribute values.
- sphinx.latexwriter: Implement option lists. Also, some other changes were made to sphinx.sty in order to enhance compatibility and remove old unused stuff. Thanks to Gael Varoquaux for that!
- sphinx.roles: Fix referencing glossary terms with explicit targets.
- sphinx.environment: Don't swallow TOC entries when resolving subtrees.
- sphinx.quickstart: Create a sensible default latex_documents setting.
- sphinx.builder, sphinx.environment: Gracefully handle some user error cases.
- sphinx.util: Follow symbolic links when searching for documents.

Sphinx 0.1

Release 0.1.61950 (Mar 26, 2008)

• sphinx.quickstart: Fix format string for Makefile.

Release 0.1.61945 (Mar 26, 2008)

- sphinx.htmlwriter, sphinx.latexwriter: Support the .. image:: directive by copying image files to the output directory.
- sphinx.builder: Consistently name "special" HTML output directories with a leading underscore; this means _sources and _static.
- sphinx.environment: Take dependent files into account when collecting the set of outdated sources.
- sphinx.directives: Record files included with .. literalinclude:: as dependencies.
- sphinx.ext.autodoc: Record files from which docstrings are included as dependencies.
- sphinx.builder: Rebuild all HTML files in case of a template change.
- sphinx.builder: Handle unavailability of TOC relations (previous/ next chapter) more gracefully in the HTML builder.
- sphinx.latexwriter: Include fncychap.sty which doesn't seem to be very common in TeX distributions. Add a clean target in the latex Makefile. Really pass the correct paper and size options to the LaTeX document class.
- setup: On Python 2.4, don't egg-depend on Docutils if a Docutils is already installed else it will be overwritten.

Release 0.1.61843 (Mar 24, 2008)

- sphinx.quickstart: Really don't create a makefile if the user doesn't want one.
- setup: Don't install scripts twice, via setuptools entry points and distutils scripts. Only install via entry points.
- sphinx.builder: Don't recognize the HTML builder's copied source files (under _sources) as input files if the source suffix is .txt.
- sphinx.highlighting: Generate correct markup for LaTeX Verbatim environment escapes even if Pygments is not installed
- sphinx.builder: The WebHTMLBuilder is now called PickleHTMLBuilder.
- sphinx.htmlwriter: Make parsed-literal blocks work as expected, not highlighting them via Pygments.
- sphinx.environment: Don't error out on reading an empty source file.

Release 0.1.61798 (Mar 23, 2008)

- sphinx: Work with Docutils SVN snapshots as well as 0.4.
- sphinx.ext.doctest: Make the group in which doctest blocks are placed selectable, and default to 'default'.
- sphinx.ext.doctest: Replace <BLANKLINE> in doctest blocks by real blank lines for presentation output, and remove doctest options given inline.
- sphinx.environment: Move doctest_blocks out of block_quotes to support indented doctest blocks.
- sphinx.ext.autodoc: Render .. automodule:: docstrings in a section node, so that module docstrings can contain proper sectioning.
- sphinx.ext.autodoc: Use the module's encoding for decoding docstrings, rather than requiring ASCII.

14.4. Prior releases 637

Release 0.1.61611 (Mar 21, 2008)

• First public release.

PROJECTS USING SPHINX

This is an incomplete list of projects that use Sphinx for their documentation. If you would like to add a project, please create an issue or pull request on GitHub¹⁵⁵⁵.

I've grouped the list into sections to make it easier to find interesting examples.

15.1 Documentation using the alabaster theme

- Alabaster¹⁵⁵⁶
- Blinker¹⁵⁵⁷
- Calibre¹⁵⁵⁸
- CherryPy¹⁵⁵⁹
- Click¹⁵⁶⁰ (customized)
- coala¹⁵⁶¹ (customized)
- CodePy¹⁵⁶²
- Django Q¹⁵⁶³
- Eve¹⁵⁶⁴ (Python REST API framework)
- Fabric¹⁵⁶⁵
- Fityk¹⁵⁶⁶
- Flask¹⁵⁶⁷
- Flask-OpenID¹⁵⁶⁸
- Invoke¹⁵⁶⁹

¹⁵⁵⁵ https://github.com/sphinx-doc/sphinx/

¹⁵⁵⁶ https://alabaster.readthedocs.io/

¹⁵⁵⁷ https://blinker.readthedocs.io/

¹⁵⁵⁸ https://manual.calibre-ebook.com/

¹⁵⁵⁹ https://cherrypy.readthedocs.io/

¹⁵⁶⁰ https://click.palletsprojects.com/

¹⁵⁶¹ https://docs.coala.io/

¹⁵⁶² https://documen.tician.de/codepy/

¹⁵⁶³ https://django-q.readthedocs.io/

¹⁵⁶⁴ https://docs.python-eve.org/

¹⁵⁶⁵ https://docs.fabfile.org/

¹⁵⁶⁶ https://fityk.nieto.pl/

¹⁵⁶⁷ https://flask.palletsprojects.com/

¹⁵⁶⁸ https://pythonhosted.org/Flask-OpenID/

¹⁵⁶⁹ https://docs.pyinvoke.org/

- Jinja¹⁵⁷⁰
- Lino¹⁵⁷¹ (customized)
- Linux kernel¹⁵⁷² (customized)
- marbl¹⁵⁷³
- MeshPv¹⁵⁷⁴
- Molecule¹⁵⁷⁵
- Momotor LTI¹⁵⁷⁶
- Podman¹⁵⁷⁷
- PvCUDA¹⁵⁷⁸
- PyOpenCL¹⁵⁷⁹
- PyLangAcq¹⁵⁸⁰
- python-apt¹⁵⁸¹
- PyVisfile¹⁵⁸²
- Requests¹⁵⁸³
- searx 1584
- Spyder¹⁵⁸⁵ (customized)
- Tablib¹⁵⁸⁶
- urllib3¹⁵⁸⁷ (customized)
- Werkzeug¹⁵⁸⁸
- Write the Docs¹⁵⁸⁹

15.2 Documentation using the classic theme

- Advanced Generic Widgets 1590 (customized)
- Apache CouchDB¹⁵⁹¹ (customized)

```
1570 https://jinja.palletsprojects.com/
```

¹⁵⁷¹ https://www.lino-framework.org/

¹⁵⁷² https://www.kernel.org/doc/html/latest/index.html

¹⁵⁷³ https://getmarbl.readthedocs.io/

¹⁵⁷⁴ https://documen.tician.de/meshpy/

¹⁵⁷⁵ https://molecule.readthedocs.io/

¹⁵⁷⁶ https://momotor.org/doc/lti/canvas/

¹⁵⁷⁷ https://docs.podman.io/

¹⁵⁷⁸ https://documen.tician.de/pycuda/

¹⁵⁷⁹ https://documen.tician.de/pyopencl/

¹⁵⁸⁰ https://pylangacq.org/

¹⁵⁸¹ https://apt-team.pages.debian.net/python-apt/

¹⁵⁸² https://documen.tician.de/pyvisfile/

¹⁵⁸³ https://requests.readthedocs.io/

¹⁵⁸⁴ https://asciimoo.github.io/searx/

¹⁵⁸⁵ https://docs.spyder-ide.org/

¹⁵⁸⁶ http://docs.python-tablib.org/

¹⁵⁸⁷ https://urllib3.readthedocs.io/

¹⁵⁸⁸ https://werkzeug.palletsprojects.com/

¹⁵⁸⁹ https://www.writethedocs.org/

¹⁵⁹⁰ https://xoomer.virgilio.it/infinity77/AGW_Docs/

¹⁵⁹¹ https://docs.couchdb.org/

- APSW¹⁵⁹²
- Arb¹⁵⁹³
- Beautiful Soup¹⁵⁹⁴
- Blender API¹⁵⁹⁵
- Bugzilla¹⁵⁹⁶
- Buildbot¹⁵⁹⁷
- CMake¹⁵⁹⁸ (customized)
- Chaco¹⁵⁹⁹ (customized)
- DEAP¹⁶⁰⁰ (customized)
- Director¹⁶⁰¹
- EZ-Draw¹⁶⁰² (customized)
- Generic Mapping Tools (GMT)¹⁶⁰³ (customized)
- Genomedata¹⁶⁰⁴
- GetFEM¹⁶⁰⁵ (customized)
- Glasgow Haskell Compiler¹⁶⁰⁶ (customized)
- Grok¹⁶⁰⁷ (customized)
- GROMACS¹⁶⁰⁸
- GSL Shell¹⁶⁰⁹
- Hands-on Python Tutorial 1610
- Kaa¹⁶¹¹ (customized)
- Leo¹⁶¹² (customized)
- Mayavi¹⁶¹³ (customized)
- MediaGoblin¹⁶¹⁴ (customized)

```
1592 https://rogerbinns.github.io/apsw/
```

¹⁵⁹³ https://arblib.org/

¹⁵⁹⁴ https://www.crummy.com/software/BeautifulSoup/bs4/doc/

¹⁵⁹⁵ https://docs.blender.org/api/current/

¹⁵⁹⁶ https://bugzilla.readthedocs.io/

¹⁵⁹⁷ https://docs.buildbot.net/latest/

¹⁵⁹⁸ https://cmake.org/documentation/

¹⁵⁹⁹ https://docs.enthought.com/chaco/

¹⁶⁰⁰ https://deap.readthedocs.io/

¹⁶⁰¹ https://pythonhosted.org/director/

¹⁶⁰² https://pageperso.lis-lab.fr/~edouard.thiel/ez-draw/doc/en/html/ez-manual.html

¹⁶⁰³ https://docs.generic-mapping-tools.org/latest/

¹⁶⁰⁴ https://noble.gs.washington.edu/proj/genomedata/doc/1.3.3/

¹⁶⁰⁵ https://getfem.org/

¹⁶⁰⁶ https://downloads.haskell.org/~ghc/latest/docs/html/users_guide/

¹⁶⁰⁷ https://web.archive.org/web/20230708190705/http://grok.zope.org/doc/current/

¹⁶⁰⁸ https://manual.gromacs.org/documentation/

¹⁶⁰⁹ https://www.nongnu.org/gsl-shell/

¹⁶¹⁰ http://anh.cs.luc.edu:80/python/hands-on/3.1/handsonHtml/

https://freevo.github.io/kaa-base/

¹⁶¹² https://leo-editor.github.io/leo-editor/

¹⁶¹³ https://docs.enthought.com/mayavi/mayavi/

¹⁶¹⁴ https://mediagoblin.readthedocs.io/

- mpmath¹⁶¹⁵
- OpenCV¹⁶¹⁶ (customized)
- OpenEXR¹⁶¹⁷
- OpenGDA¹⁶¹⁸
- phpDocumentor¹⁶¹⁹ (customized)
- Plone¹⁶²⁰ (customized)
- PvEMD¹⁶²¹
- Pyevolve¹⁶²²
- Pygame¹⁶²³ (customized)
- PvMOI¹⁶²⁴
- PyQt4¹⁶²⁵ (customized)
- PyQt5¹⁶²⁶ (customized)
- Python 2¹⁶²⁷
- Python 3¹⁶²⁸ (customized)
- Python Packaging Authority¹⁶²⁹ (customized)
- Ring programming language 1630 (customized)
- SageMath¹⁶³¹ (customized)
- Segway¹⁶³²
- simuPOP¹⁶³³ (customized)
- SymPy¹⁶³⁴
- TurboGears¹⁶³⁵ (customized)
- tvtk¹⁶³⁶
- Varnish¹⁶³⁷ (customized, alabaster for index)

```
1615 https://mpmath.org/doc/current/
```

¹⁶¹⁶ https://docs.opencv.org/

¹⁶¹⁷ https://excamera.com/articles/26/doc/index.html

¹⁶¹⁸ https://alfred.diamond.ac.uk/documentation/

¹⁶¹⁹ https://docs.phpdoc.org/

¹⁶²⁰ https://docs.plone.org/

¹⁶²¹ https://pyemd.readthedocs.io/

¹⁶²² https://pyevolve.sourceforge.net/

¹⁶²³ https://www.pygame.org/docs/

¹⁶²⁴ https://dsuch.github.io/pymqi/

¹⁶²⁵ https://pyqt.sourceforge.net/Docs/PyQt4/

¹⁶²⁶ https://pyqt.sourceforge.net/Docs/PyQt5/

¹⁶²⁷ https://docs.python.org/2/

¹⁶²⁸ https://docs.python.org/3/

¹⁶²⁹ https://www.pypa.io/

¹⁶³⁰ https://ring-lang.github.io/doc1.20/

¹⁶³¹ https://doc.sagemath.org/

¹⁶³² https://noble.gs.washington.edu/proj/segway/doc/1.1.0/segway.html

¹⁶³³ https://bopeng.github.io/simuPOP/

¹⁶³⁴ https://docs.sympy.org/

¹⁶³⁵ https://turbogears.readthedocs.io/

¹⁶³⁶ https://docs.enthought.com/mayavi/tvtk/

¹⁶³⁷ https://www.varnish-cache.org/docs/

- Waf¹⁶³⁸
- wxPython Phoenix¹⁶³⁹ (customized)
- z3c¹⁶⁴⁰
- zc.async¹⁶⁴¹ (customized)
- Zope¹⁶⁴² (customized)

15.3 Documentation using the sphinxdoc theme

- ABRT¹⁶⁴³
- cartopy¹⁶⁴⁴
- Jython¹⁶⁴⁵
- LLVM¹⁶⁴⁶
- PyCantonese¹⁶⁴⁷
- Pyre¹⁶⁴⁸
- pySPACE¹⁶⁴⁹
- Pysparse¹⁶⁵⁰
- PyTango¹⁶⁵¹
- Python Wild Magic 1652 (customized)
- RDKit1653
- Reteisi¹⁶⁵⁴ (customized)
- Turbulenz¹⁶⁵⁵

15.4 Documentation using the nature theme

- Alembic¹⁶⁵⁶
- Cython¹⁶⁵⁷

¹⁶³⁸ https://waf.io/apidocs/

¹⁶³⁹ https://wxpython.org/Phoenix/docs/html/main.html

¹⁶⁴⁰ https://www.ibiblio.org/paulcarduner/z3ctutorial/

¹⁶⁴¹ https://pythonhosted.org/zc.async/

¹⁶⁴² https://www.zope.dev/

¹⁶⁴³ https://abrt.readthedocs.io/

¹⁶⁴⁴ https://scitools.org.uk/cartopy/docs/latest/

¹⁶⁴⁵ https://jython.readthedocs.io/

¹⁶⁴⁶ https://llvm.org/docs/

¹⁶⁴⁷ https://pycantonese.org/

¹⁶⁴⁸ https://pyre.readthedocs.io/

¹⁶⁴⁹ https://pyspace.github.io/pyspace/

¹⁶⁵⁰ https://pysparse.sourceforge.net/

¹⁶⁵¹ https://pytango.readthedocs.io

¹⁶⁵² https://vmlaker.github.io/pythonwildmagic/

¹⁶⁵³ https://www.rdkit.org/docs/

¹⁶⁵⁴ https://www.reteisi.org/contents.html

¹⁶⁵⁵ http://docs.turbulenz.com/

¹⁶⁵⁶ https://alembic.sqlalchemy.org/

¹⁶⁵⁷ https://docs.cython.org/

- easybuild1658
- libLAS¹⁶⁵⁹ (customized)
- Lmod¹⁶⁶⁰
- MapServer¹⁶⁶¹ (customized)
- PyWavelets¹⁶⁶²
- Setuptools 1663
- Spring Python¹⁶⁶⁴
- StatsModels¹⁶⁶⁵ (customized)
- Sylli¹⁶⁶⁶

15.5 Documentation using another builtin theme

- Breathe¹⁶⁶⁷ (haiku)
- Breezy (fork of Bazaar)¹⁶⁶⁸ (agogo)
- MPipe¹⁶⁶⁹ (sphinx13)
- NLTK¹⁶⁷⁰ (agogo)
- PyPubSub¹⁶⁷¹ (bizstyle)
- Pylons¹⁶⁷² (pyramid)
- Pyramid web framework¹⁶⁷³ (pyramid)
- RxDock¹⁶⁷⁴ (bizstyle)
- Sphinx¹⁶⁷⁵ (sphinx13) :-)
- Valence¹⁶⁷⁶ (haiku, customized)

15.6 Documentation using Read the Docs Sphinx Theme

Read the Docs Sphinx Theme documentation¹⁶⁷⁷

```
1658 https://easybuild.readthedocs.io/
1659 https://liblas.org/
1660 https://lmod.readthedocs.io/
1661 https://mapserver.org/
1662 https://pywavelets.readthedocs.io/
1663 https://setuptools.readthedocs.io/
1664 https://docs.spring.io/spring-python/1.2.x/sphinx/html/
1665 https://www.statsmodels.org/
1666 https://sylli.sourceforge.net/
1667 https://breathe.readthedocs.io/
1668 https://www.breezy-vcs.org/doc/en/
1669 https://vmlaker.github.io/mpipe/
1670 https://www.nltk.org/
1671 https://pypubsub.readthedocs.io/
1672 https://docs.pylonsproject.org/projects/pylons-webframework/
1673 https://docs.pylonsproject.org/projects/pyramid/
1674 https://rxdock.gitlab.io/documentation/devel/html/
1675 https://www.sphinx-doc.org/
1676 https://docs.valence.desire2learn.com/
```

1677 https://sphinx-rtd-theme.readthedocs.io/

- Aesara (fork of Theano)¹⁶⁷⁸
- Annotator¹⁶⁷⁹
- Ansible 1680 (customized)
- Arcade¹⁶⁸¹
- $aria2^{1682}$
- ASE¹⁶⁸³
- asvin¹⁶⁸⁴
- Autofac¹⁶⁸⁵
- BigchainDB¹⁶⁸⁶
- Blender Reference Manual 1687
- Blocks¹⁶⁸⁸
- bootstrap-datepicker¹⁶⁸⁹
- Certbot¹⁶⁹⁰
- CKAN¹⁶⁹¹
- Copr Buildsystem¹⁶⁹² (customized)
- Coreboot¹⁶⁹³
- Chainer¹⁶⁹⁴ (customized)
- citeproc-js¹⁶⁹⁵
- cloud-init¹⁶⁹⁶
- CodeIgniter¹⁶⁹⁷
- Conda¹⁶⁹⁸
- Corda¹⁶⁹⁹
- Dask¹⁷⁰⁰

```
1678 https://aesara.readthedocs.io/
```

¹⁶⁷⁹ https://docs.annotatorjs.org/

¹⁶⁸⁰ https://docs.ansible.com/

¹⁶⁸¹ https://arcade.academy/

¹⁶⁸² https://aria2.github.io/manual/en/html/

¹⁶⁸³ https://wiki.fysik.dtu.dk/ase/

¹⁶⁸⁴ https://asvin.readthedocs.io/

¹⁶⁸⁵ https://docs.autofac.org/

¹⁶⁸⁶ https://docs.bigchaindb.com/

¹⁶⁸⁷ https://docs.blender.org/manual/

¹⁶⁸⁸ https://blocks.readthedocs.io/

¹⁶⁸⁹ https://bootstrap-datepicker.readthedocs.io/

¹⁶⁹⁰ https://certbot.eff.org/docs/

¹⁶⁹¹ https://docs.ckan.org/

¹⁶⁹² https://docs.pagure.org/copr.copr/

¹⁶⁹³ https://doc.coreboot.org/

¹⁶⁹⁴ https://docs.chainer.org/

¹⁶⁹⁵ https://citeproc-js.readthedocs.io/

¹⁶⁹⁶ https://cloudinit.readthedocs.io/

¹⁶⁹⁷ https://www.codeigniter.com/user_guide/

¹⁶⁹⁸ https://conda.io/docs/

¹⁶⁹⁹ https://docs.corda.net/

¹⁷⁰⁰ https://dask.pydata.org/

- Databricks¹⁷⁰¹ (customized)
- Dataiku DSS¹⁷⁰²
- DNF¹⁷⁰³
- Distro Tracker¹⁷⁰⁴
- Django-cas-ng¹⁷⁰⁵
- dj-stripe¹⁷⁰⁶
- edX¹⁷⁰⁷
- Electrum¹⁷⁰⁸
- ESWP31709
- Ethereum Homestead¹⁷¹⁰
- Exhale¹⁷¹¹
- Faker¹⁷¹²
- Fidimag¹⁷¹³
- Flake8¹⁷¹⁴
- Flatpak¹⁷¹⁵
- FluidDyn¹⁷¹⁶
- Fluidsim¹⁷¹⁷
- Gallium¹⁷¹⁸
- GeoNode¹⁷¹⁹
- Glances¹⁷²⁰
- Godot¹⁷²¹
- Graylog¹⁷²²
- GPAW¹⁷²³ (customized)

```
1701 https://docs.databricks.com/
```

¹⁷⁰² https://doc.dataiku.com/

¹⁷⁰³ https://dnf.readthedocs.io/

¹⁷⁰⁴ https://qa.pages.debian.net/distro-tracker/

¹⁷⁰⁵ https://djangocas.dev/docs/

¹⁷⁰⁶ https://dj-stripe.github.io/dj-stripe/

¹⁷⁰⁷ https://docs.edx.org/

https://docs.electrum.org/

¹⁷⁰⁹ https://eswp3.readthedocs.io/

¹⁷¹⁰ https://www.ethdocs.org/

¹⁷¹¹ https://exhale.readthedocs.io/

¹⁷¹² https://faker.readthedocs.io/

¹⁷¹³ https://fidimag.readthedocs.io/

¹⁷¹⁴ https://flake8.pycqa.org/

¹⁷¹⁵ https://docs.flatpak.org/

¹⁷¹⁶ https://fluiddyn.readthedocs.io/

¹⁷¹⁷ https://fluidsim.readthedocs.io/

¹⁷¹⁸ https://gallium.readthedocs.io/

¹⁷¹⁹ https://docs.geonode.org/

¹⁷²⁰ https://glances.readthedocs.io/

¹⁷²¹ https://godot.readthedocs.io/

¹⁷²² https://docs.graylog.org/

¹⁷²³ https://wiki.fysik.dtu.dk/gpaw/

- HDF5 for Python (h5py)¹⁷²⁴
- HyperKitty¹⁷²⁵
- Hyperledger Fabric ¹⁷²⁶
- IdentityServer¹⁷²⁷
- Idris¹⁷²⁸
- Inkscape¹⁷²⁹ (customized)
- javasphinx¹⁷³⁰
- Jupyter Notebook¹⁷³¹
- Kanboard¹⁷³²
- Lasagne¹⁷³³
- latexindent.pl¹⁷³⁴
- Learning Apache Spark with Python 1735
- LibCEED¹⁷³⁶
- Linguistica¹⁷³⁷
- Mailman¹⁷³⁸
- MathJax¹⁷³⁹
- MDTraj¹⁷⁴⁰ (customized)
- Mesa 3D¹⁷⁴¹
- micca MICrobial Community Analysis 1742
- MicroPython¹⁷⁴³
- Mink¹⁷⁴⁴
- Mockery¹⁷⁴⁵
- mod_wsgi¹⁷⁴⁶

```
1724 https://docs.h5py.org/
```

https://hyperkitty.readthedocs.io/

¹⁷²⁶ https://hyperledger-fabric.readthedocs.io/

¹⁷²⁷ https://docs.identityserver.io/

¹⁷²⁸ https://docs.idris-lang.org/

¹⁷²⁹ https://inkscape-manuals.readthedocs.io/

¹⁷³⁰ https://bronto-javasphinx.readthedocs.io/

https://jupyter-notebook.readthedocs.io/

¹⁷³² https://docs.kanboard.org/

¹⁷³³ https://lasagne.readthedocs.io/

¹⁷³⁴ https://latexindentpl.readthedocs.io/

¹⁷³⁵ https://runawayhorse001.github.io/LearningApacheSpark

¹⁷³⁶ https://libceed.readthedocs.io/

¹⁷³⁷ https://linguistica-uchicago.github.io/lxa5/

¹⁷³⁸ https://docs.list.org/

¹⁷³⁹ https://docs.mathjax.org/

¹⁷⁴⁰ https://mdtraj.org/

¹⁷⁴¹ https://docs.mesa3d.org/

¹⁷⁴² https://micca.readthedocs.io/

¹⁷⁴³ https://docs.micropython.org/

https://mink.behat.org/

¹⁷⁴⁵ https://docs.mockery.io/

¹⁷⁴⁶ https://modwsgi.readthedocs.io/

- MoinMoin¹⁷⁴⁷
- Mopidy¹⁷⁴⁸
- mpi4py¹⁷⁴⁹
- MyHDL¹⁷⁵⁰
- Mypy¹⁷⁵¹
- Netgate Docs¹⁷⁵²
- Nextcloud Server¹⁷⁵³
- Nextflow¹⁷⁵⁴
- nghttp2¹⁷⁵⁵
- NICOS¹⁷⁵⁶ (customized)
- OpenFAST¹⁷⁵⁷
- Panda3D¹⁷⁵⁸ (customized)
- Pelican¹⁷⁵⁹
- picamera¹⁷⁶⁰
- Pillow¹⁷⁶¹
- pip¹⁷⁶²
- Paver¹⁷⁶³
- peewee¹⁷⁶⁴
- Phinx¹⁷⁶⁵
- phpMyAdmin¹⁷⁶⁶
- PHPUnit¹⁷⁶⁷
- PHPWord¹⁷⁶⁸
- PROS¹⁷⁶⁹ (customized)

```
1747 https://moin-20.readthedocs.io/
```

¹⁷⁴⁸ https://docs.mopidy.com/

¹⁷⁴⁹ https://mpi4py.readthedocs.io/

¹⁷⁵⁰ https://docs.myhdl.org/

¹⁷⁵¹ https://mypy.readthedocs.io/

¹⁷⁵² https://docs.netgate.com/

¹⁷⁵³ https://docs.nextcloud.com/#server

¹⁷⁵⁴ https://www.nextflow.io/docs/latest/index.html

¹⁷⁵⁵ https://nghttp2.org/documentation/

¹⁷⁵⁶ https://forge.frm2.tum.de/nicos/doc/nicos-master/

¹⁷⁵⁷ https://openfast.readthedocs.io/

¹⁷⁵⁸ https://docs.panda3d.org/

¹⁷⁵⁹ https://docs.getpelican.com/

¹⁷⁶⁰ https://picamera.readthedocs.io/

¹⁷⁶¹ https://pillow.readthedocs.io/

¹⁷⁶² https://pip.pypa.io/

¹⁷⁶³ https://paver.readthedocs.io/

¹⁷⁶⁴ https://docs.peewee-orm.com/

¹⁷⁶⁵ https://docs.phinx.org/

¹⁷⁶⁶ https://docs.phpmyadmin.net/

¹⁷⁶⁷ https://phpunit.readthedocs.io/

¹⁷⁶⁸ https://phpword.readthedocs.io/

¹⁷⁶⁹ https://pros.cs.purdue.edu/v5/

- Pweave¹⁷⁷⁰
- pyca/cryptograhpy¹⁷⁷¹
- pyglet¹⁷⁷²
- PyNaCl¹⁷⁷³
- pyOpenSSL¹⁷⁷⁴
- PyPy¹⁷⁷⁵
- python-sqlparse¹⁷⁷⁶
- PyVISA¹⁷⁷⁷
- Read The Docs¹⁷⁷⁸
- RenderDoc¹⁷⁷⁹
- ROCm Platform¹⁷⁸⁰
- Free your information from their silos (French)¹⁷⁸¹ (customized)
- Releases Sphinx extension 1782
- Otile¹⁷⁸³
- Quex¹⁷⁸⁴
- QuTiP¹⁷⁸⁵
- Sawtooth¹⁷⁸⁶
- Scapy¹⁷⁸⁷
- SimGrid¹⁷⁸⁸
- SimPy¹⁷⁸⁹
- six¹⁷⁹⁰
- Solidity¹⁷⁹¹
- Sonos Controller (SoCo)¹⁷⁹²

```
1770 https://mpastell.com/pweave/
```

¹⁷⁷¹ https://cryptography.io/

¹⁷⁷² https://pyglet.readthedocs.io/

¹⁷⁷³ https://pynacl.readthedocs.io/

¹⁷⁷⁴ https://www.pyopenssl.org/

¹⁷⁷⁵ https://doc.pypy.org/

¹⁷⁷⁶ https://sqlparse.readthedocs.io/

¹⁷⁷⁷ https://pyvisa.readthedocs.io/

¹⁷⁷⁸ https://docs.readthedocs.io/

¹⁷⁷⁹ https://renderdoc.org/docs/

¹⁷⁸⁰ https://rocmdocs.amd.com/

¹⁷⁸¹ https://redaction-technique.org/

¹⁷⁸² https://releases.readthedocs.io/

¹⁷⁸³ https://docs.qtile.org/

¹⁷⁸⁴ https://quex.sourceforge.net/doc/html/main.html

¹⁷⁸⁵ https://qutip.readthedocs.io/en/latest/

¹⁷⁸⁶ https://sawtooth.splinter.dev/docs

¹⁷⁸⁷ https://scapy.readthedocs.io/

¹⁷⁸⁸ https://simgrid.org/doc/latest/

¹⁷⁸⁹ https://simpy.readthedocs.io/

¹⁷⁹⁰ https://six.readthedocs.io/

¹⁷⁹¹ https://solidity.readthedocs.io/

¹⁷⁹² https://docs.python-soco.com/

- Sphinx AutoAPI¹⁷⁹³
- sphinx-argparse¹⁷⁹⁴
- sphinx-tabs¹⁷⁹⁵
- Sphinx-Gallery¹⁷⁹⁶ (customized)
- Sphinx with Github Webpages 1797
- SpotBugs¹⁷⁹⁸
- StarUML¹⁷⁹⁹
- Sublime Text Unofficial Documentation 1800
- SunPy¹⁸⁰¹
- Sylius¹⁸⁰²
- Syncthing¹⁸⁰³
- Tango Controls¹⁸⁰⁴ (customized)
- ThreatConnect¹⁸⁰⁵
- TrueNAS¹⁸⁰⁶ (customized)
- Tuleap¹⁸⁰⁷
- TYPO3¹⁸⁰⁸ (customized)
- Veyon¹⁸⁰⁹
- Ubiquity¹⁸¹⁰
- uWSGI¹⁸¹¹
- virtualenv¹⁸¹²
- Wagtail¹⁸¹³
- Web Application Attack and Audit Framework (w3af)¹⁸¹⁴
- Weblate¹⁸¹⁵

```
1793 https://sphinx-autoapi.readthedocs.io/
1794 https://sphinx-argparse.readthedocs.io/
```

¹⁷⁹⁵ https://sphinx-tabs.readthedocs.io/

¹⁷⁹⁶ https://sphinx-gallery.readthedocs.io/

¹⁷⁹⁷ https://runawayhorse001.github.io/SphinxGithub

¹⁷⁹⁸ https://spotbugs.readthedocs.io/

¹⁷⁹⁹ https://docs.staruml.io/

¹⁸⁰⁰ https://docs.sublimetext.info/

¹⁸⁰¹ https://docs.sunpy.org/

¹⁸⁰² https://docs.sylius.com/

¹⁸⁰³ https://docs.syncthing.net/

¹⁸⁰⁴ https://tango-controls.readthedocs.io/

¹⁸⁰⁵ https://docs.threatconnect.com/

¹⁸⁰⁶ https://www.ixsystems.com/documentation/truenas/

¹⁸⁰⁷ https://tuleap.net/doc/en/

¹⁸⁰⁸ https://docs.typo3.org/

¹⁸⁰⁹ https://docs.veyon.io/

¹⁸¹⁰ https://micro-framework.readthedocs.io/

¹⁸¹¹ https://uwsgi-docs.readthedocs.io/

¹⁸¹² https://virtualenv.readthedocs.io/

¹⁸¹³ https://docs.wagtail.io/

¹⁸¹⁴ https://docs.w3af.org/

¹⁸¹⁵ https://docs.weblate.org/

- x265¹⁸¹⁶
- Zeek¹⁸¹⁷
- Zulip¹⁸¹⁸

15.7 Documentation using Sphinx Bootstrap Theme

Sphinx Bootstrap Theme documentation 1819

- Bootstrap Theme¹⁸²⁰
- C/C++ Software Development with Eclipse¹⁸²¹
- Dataverse¹⁸²²
- e-cidadania¹⁸²³
- Hangfire¹⁸²⁴
- Hedge 1825
- ObsPy¹⁸²⁶
- OPNFV¹⁸²⁷
- Pootle¹⁸²⁸
- PyUblas¹⁸²⁹
- seaborn¹⁸³⁰

15.8 Documentation using PyData Sphinx Theme

Pydata Sphinx Theme documentation 1831

- Arviz¹⁸³²
- Astropy¹⁸³³
- Binder¹⁸³⁴
- Bokeh¹⁸³⁵

```
1816 https://x265.readthedocs.io/
```

¹⁸¹⁷ https://docs.zeek.org/

¹⁸¹⁸ https://zulip.readthedocs.io/

¹⁸¹⁹ https://ryan-roemer.github.io/sphinx-bootstrap-theme/

¹⁸²⁰ https://ryan-roemer.github.io/sphinx-bootstrap-theme/

¹⁸²¹ https://eclipsebook.in/

¹⁸²² https://guides.dataverse.org/

¹⁸²³ https://e-cidadania.readthedocs.io/

¹⁸²⁴ https://docs.hangfire.io/

¹⁸²⁵ https://documen.tician.de/hedge/

¹⁸²⁶ https://docs.obspy.org/

¹⁸²⁷ https://docs.opnfv.org/

¹⁸²⁸ https://docs.translatehouse.org/projects/pootle/

¹⁸²⁹ https://documen.tician.de/pyublas/

¹⁸³⁰ https://seaborn.pydata.org/

¹⁸³¹ https://pydata-sphinx-theme.readthedocs.io/

¹⁸³² https://python.arviz.org/en/stable/

¹⁸³³ https://docs.astropy.org/

¹⁸³⁴ https://mybinder.readthedocs.io/en/latest/

¹⁸³⁵ https://docs.bokeh.org/en/latest/

- CuPy¹⁸³⁶
- EnOSlib¹⁸³⁷
- Fairlearn¹⁸³⁸
- Feature-engine¹⁸³⁹
- Jupyter¹⁸⁴⁰
- Jupyter Book¹⁸⁴¹
- Matplotlib¹⁸⁴²
- MegEngine¹⁸⁴³
- MNE-Python¹⁸⁴⁴
- NetworkX¹⁸⁴⁵
- Numpy¹⁸⁴⁶
- Pandas¹⁸⁴⁷
- PhaseFieldX¹⁸⁴⁸
- Pystra (continuation of PyRe)¹⁸⁴⁹
- PyVista¹⁸⁵⁰
- SciPy¹⁸⁵¹
- Seaborn¹⁸⁵²
- SEPAL¹⁸⁵³

15.9 Documentation using Furo theme

Furo theme documentation ¹⁸⁵⁴

- Furo theme¹⁸⁵⁵
- pytest¹⁸⁵⁶
- Python Packaging User Guide¹⁸⁵⁷

¹⁸³⁶ https://docs.cupy.dev/en/stable/

¹⁸³⁷ https://discovery.gitlabpages.inria.fr/enoslib/

¹⁸³⁸ https://fairlearn.org/main/

¹⁸³⁹ https://feature-engine.readthedocs.io/en/latest/

¹⁸⁴⁰ https://docs.jupyter.org/en/latest/

¹⁸⁴¹ https://jupyterbook.org/en/stable/intro.html

¹⁸⁴² https://matplotlib.org/stable/index.html

¹⁸⁴³ https://megengine.org.cn/doc/stable/en/

¹⁸⁴⁴ https://mne.tools/stable/

¹⁸⁴⁵ https://networkx.org/documentation/stable/

¹⁸⁴⁶ https://numpy.org/doc/stable/

¹⁸⁴⁷ https://pandas.pydata.org/docs/

¹⁸⁴⁸ https://phasefieldx.readthedocs.io/

¹⁸⁴⁹ https://pystra.github.io/pystra/

¹⁸⁵⁰ https://docs.pyvista.org/

¹⁸⁵¹ https://docs.scipy.org/doc/scipy/

¹⁸⁵² https://seaborn.pydata.org/

¹⁸⁵³ https://docs.sepal.io/en/latest/index.html

¹⁸⁵⁴ https://pradyunsg.me/furo/

¹⁸⁵⁵ https://pradyunsg.me/furo/

¹⁸⁵⁶ https://docs.pytest.org/

¹⁸⁵⁷ https://packaging.python.org/

• Python Developer's Guide¹⁸⁵⁸

15.10 Documentation using a custom theme or integrated in a website

- AIOHTTP¹⁸⁵⁹
- Apache Cassandra¹⁸⁶⁰
- Boto 3¹⁸⁶¹
- CakePHP¹⁸⁶²
- Ceph¹⁸⁶³
- Chef¹⁸⁶⁴
- CKAN¹⁸⁶⁵
- Confluent Platform¹⁸⁶⁶
- Django¹⁸⁶⁷
- django CMS¹⁸⁶⁸
- Doctrine¹⁸⁶⁹
- Enterprise Toolkit for Acrobat products 1870
- FreeFEM¹⁸⁷¹
- fmt¹⁸⁷²
- Gameduino¹⁸⁷³
- gensim¹⁸⁷⁴
- GeoServer¹⁸⁷⁵
- gevent¹⁸⁷⁶
- GHC Glasgow Haskell Compiler 1877
- Guzzle¹⁸⁷⁸
- H2O.ai¹⁸⁷⁹

```
1858 https://devguide.python.org/
```

¹⁸⁵⁹ https://docs.aiohttp.org/

¹⁸⁶⁰ https://cassandra.apache.org/doc/

https://cassandra.apache.org/doc

¹⁸⁶² https://book.cakephp.org/

¹⁸⁶³ https://docs.ceph.com/docs/master/

¹⁸⁶⁴ https://docs.chef.io/

¹⁸⁶⁵ https://docs.ckan.org/

¹⁸⁶⁶ https://docs.confluent.io/

¹⁸⁶⁷ https://docs.djangoproject.com/

¹⁸⁶⁸ https://docs.django-cms.org/

¹⁸⁶⁹ https://www.doctrine-project.org/

¹⁸⁷⁰ https://www.adobe.com/devnet-docs/acrobatetk/

¹⁸⁷¹ https://doc.freefem.org/introduction/

¹⁸⁷² https://fmt.dev/

¹⁸⁷³ https://excamera.com/sphinx/gameduino/

¹⁸⁷⁴ https://radimrehurek.com/gensim/

¹⁸⁷⁵ https://docs.geoserver.org/

¹⁸⁷⁶ https://www.gevent.org/

¹⁸⁷⁷ https://downloads.haskell.org/~ghc/master/users-guide/

¹⁸⁷⁸ https://docs.guzzlephp.org/

¹⁸⁷⁹ https://docs.h2o.ai/

- Heka¹⁸⁸⁰
- Istihza (Turkish Python documentation project)¹⁸⁸¹
- JupyterHub¹⁸⁸²
- Kombu¹⁸⁸³
- Lasso¹⁸⁸⁴
- Mako¹⁸⁸⁵
- MirrorBrain¹⁸⁸⁶
- Mitiq¹⁸⁸⁷
- MongoDB¹⁸⁸⁸
- Music21¹⁸⁸⁹
- MyHDL¹⁸⁹⁰
- ndnSIM¹⁸⁹¹
- nose¹⁸⁹²
- ns-3¹⁸⁹³
- ObjectListView¹⁸⁹⁴
- OpenERP¹⁸⁹⁵
- OpenCV¹⁸⁹⁶
- Open Dylan¹⁸⁹⁷
- OpenTURNS¹⁸⁹⁸
- Open vSwitch¹⁸⁹⁹
- PlatformIO¹⁹⁰⁰
- Psycopg¹⁹⁰¹
- PyEphem¹⁹⁰²

```
1880 https://hekad.readthedocs.io/
```

¹⁸⁸¹ https://python-istihza.yazbel.com/

¹⁸⁸² https://jupyterhub.readthedocs.io/

¹⁸⁸³ https://kombu.readthedocs.io/

¹⁸⁸⁴ http://www.lassoguide.com/

¹⁸⁸⁵ https://docs.makotemplates.org/

¹⁸⁸⁶ https://mirrorbrain-docs.readthedocs.io/

¹⁸⁸⁷ https://mitiq.readthedocs.io/

¹⁸⁸⁸ https://docs.mongodb.com/

¹⁸⁸⁹ https://web.mit.edu/music21/doc/

¹⁸⁹⁰ https://docs.myhdl.org/

¹⁸⁹¹ https://ndnsim.net/current/

¹⁸⁹² https://nose.readthedocs.io/

¹⁸⁹³ https://www.nsnam.org/documentation/

¹⁸⁹⁴ https://objectlistview.sourceforge.net/python/

¹⁸⁹⁵ https://doc.odoo.com/

¹⁸⁹⁶ https://docs.opencv.org/

¹⁸⁹⁷ https://opendylan.org/

¹⁸⁹⁸ https://openturns.github.io/openturns/latest/

¹⁸⁹⁹ https://docs.openvswitch.org/

¹⁹⁰⁰ https://docs.platformio.org/

¹⁹⁰¹ https://www.psycopg.org/docs/

¹⁹⁰² https://rhodesmill.org/pyephem/

- Pygments¹⁹⁰³
- Plone User Manual (German)¹⁹⁰⁴
- PSI4¹⁹⁰⁵
- PyMOTW¹⁹⁰⁶
- python-aspectlib¹⁹⁰⁷ (sphinx_py3doc_enhanced_theme¹⁹⁰⁸)
- QGIS¹⁹⁰⁹
- Roundup¹⁹¹⁰
- SaltStack¹⁹¹¹
- scikit-learn¹⁹¹²
- Scrapy¹⁹¹³
- Selenium¹⁹¹⁴
- Self¹⁹¹⁵
- Substance D¹⁹¹⁶
- Sulu¹⁹¹⁷
- SQLAlchemy¹⁹¹⁸
- tinyTiM¹⁹¹⁹
- Twisted¹⁹²⁰
- Ubuntu Packaging Guide¹⁹²¹
- WTForms¹⁹²²

15.11 Homepages and other non-documentation sites

- Arizona State University PHY494/PHY598/CHM598 Simulation approaches to Bio-and Nanophysics¹⁹²⁴ (classic)

```
1903 https://pygments.org/docs/
```

¹⁹⁰⁴ https://www.hasecke.com/plone-benutzerhandbuch/4.0/

¹⁹⁰⁵ https://www.psicode.org/psi4manual/master/index.html

¹⁹⁰⁶ https://pymotw.com/2/

¹⁹⁰⁷ https://python-aspectlib.readthedocs.io/

¹⁹⁰⁸ https://pypi.org/project/sphinx_py3doc_enhanced_theme/

¹⁹⁰⁹ https://qgis.org/en/docs/index.html

¹⁹¹⁰ https://www.roundup-tracker.org/

¹⁹¹¹ https://docs.saltstack.com/

¹⁹¹² https://scikit-learn.org/stable/

¹⁹¹³ https://doc.scrapy.org/

¹⁹¹⁴ https://docs.seleniumhq.org/docs/

¹⁹¹⁵ https://www.selflanguage.org/

¹⁹¹⁶ https://docs.pylonsproject.org/projects/substanced/

¹⁹¹⁷ https://docs.sulu.io/

¹⁹¹⁸ https://docs.sqlalchemy.org/

¹⁹¹⁹ https://tinytim.sourceforge.net/docs/2.0/

¹⁹²⁰ https://twistedmatrix.com/documents/current/

¹⁹²¹ https://packaging.ubuntu.com/html/

¹⁹²² https://wtforms.readthedocs.io/

¹⁹²³ https://www.columbia.edu/~alan/django-jsonapi-training/

¹⁹²⁴ https://becksteinlab.physics.asu.edu/pages/courses/2013/SimBioNano/

- Benoit Boissinot 1925 (classic, customized)
- EBI Cloud Consultancy Team¹⁹²⁶ (sphinx_rtd_theme)
- Eric Holscher¹⁹²⁷ (alabaster)
- Florian Diesch¹⁹²⁸
- Institute for the Design of Advanced Energy Systems (IDAES)¹⁹²⁹ (sphinx_rtd_theme)
- IDAES Examples 1930 (sphinx rtd theme)
- Lei Ma's Statistical Mechanics lecture notes 1931 (sphinx_bootstrap_theme)
- PyXLL¹⁹³² (sphinx_bootstrap_theme, customized)
- SciPy Cookbook 1933 (sphinx_rtd_theme)
- Tech writer at work blog ¹⁹³⁴ (custom theme)
- UC Berkeley ME233 Advanced Control Systems II course 1935 (sphinxdoc)
- Željko Svedružić's Biomolecular Structure and Function Laboratory (BioSFLab)¹⁹³⁶ (sphinx_bootstrap_theme)

15.12 Books produced using Sphinx

- "The Art of Community" (Japanese translation) 1937
- "Die Wahrheit des Sehens. Der DEKALOG von Krzysztof Kieślowski" 1938
- "Expert Python Programming" 1939
- "Expert Python Programming" (Japanese translation) 1940
- "Expert Python Programming 2nd Edition" (Japanese translation) 1941
- "The Hitchhiker's Guide to Python" 1942
- "LassoGuide" 1943
- "Learning Sphinx" (in Japanese) 1944
- "Learning System Programming with Go (Japanese)" 1945
- "Mercurial: the definitive guide (Second edition)" 1946

```
1925 https://bboissin.appspot.com/
1926 https://tsi-ccdoc.readthedocs.io/
1927 https://ericholscher.com/
1928 https://www.florian-diesch.de/
1929 https://idaes-pse.readthedocs.io/
1930 https://idaes.github.io/examples-pse/
1931 https://statisticalphysics.leima.is/
1932 https://www.pyxll.com/
1933 https://scipy-cookbook.readthedocs.io/
1934 https://documatt.com/blog/
1935 https://berkeley-me233.github.io/
1936 https://svedruziclab.github.io/
<sup>1937</sup> https://www.oreilly.co.jp/books/9784873114958/
1938 https://literatur.hasecke.com/post/die-wahrheit-des-sehens-dekalog-kieslowski/
1939 https://www.packtpub.com/application-development/expert-python-programming
<sup>1940</sup> https://www.amazon.co.jp/dp/4048686291/
1941 https://www.amazon.co.jp/dp/4048930613/
1942 https://docs.python-guide.org/
1943 http://www.lassoguide.com/
1944 https://www.oreilly.co.jp/books/9784873116488/
1945 https://www.lambdanote.com/products/go
```

1946 https://book.mercurial-scm.org/

- "Mithril The fastest clientside MVC (Japanese)" 1947
- · "Pioneers and Prominent Men of Utah"
- "Pomodoro Technique Illustrated" (Japanese translation) 1948
- "Professional Software Development" 1949
- "Python Professional Programming" (in Japanese) 1950
- "Python Professional Programming 2nd Edition" (in Japanese) 1951
- "Python Professional Programming 3rd Edition" (in Japanese) 1952
- Python Course by Yuri Petrov (Russian)¹⁹⁵³
- "Real World HTTP Learning The Internet and Web Technology via its history and code (Japanese)" 1954
- "Redmine Primer 5th Edition (in Japanese)" 1955
- "The repoze.bfg Web Application Framework" 1956
- "The Self-Taught Programmer" (Japanese translation) 1957
- "Simple and Steady Way of Learning for Software Engineering" (in Japanese) 1958
- "Software-Dokumentation mit Sphinx" 1959
- "Theoretical Physics Reference" ¹⁹⁶⁰
- "The Varnish Book" 1961

15.13 Theses produced using Sphinx

- "Content Conditioning and Distribution for Dynamic Virtual Worlds" 1962
- "The Sphinx Thesis Resource" 1963

15.14 Projects integrating Sphinx functionality

- Read the Docs¹⁹⁶⁴, a software-as-a-service documentation hosting platform, uses Sphinx to automatically build documentation updates that are pushed to GitHub.
- Spyder¹⁹⁶⁵, the Scientific Python Development Environment, uses Sphinx in its help pane to render rich documentation for functions, classes and methods automatically or on-demand.

```
<sup>1947</sup> https://www.oreilly.co.jp/books/9784873117447/
1948 https://www.amazon.co.jp/dp/4048689525/
1949 https://mixmastamyk.bitbucket.io/pro_soft_dev/
1950 https://www.amazon.co.jp/dp/4798032948/
1951 https://www.amazon.co.jp/dp/479804315X/
<sup>1952</sup> https://www.amazon.co.jp/dp/4798053821/
1953 https://www.yuripetrov.ru/edu/python
1954 https://www.oreilly.co.jp/books/9784873118048/
1955 https://www.shuwasystem.co.jp/products/7980html/4825.html
1956 https://www.amazon.com/repoze-bfg-Web-Application-Framework-Version/dp/0615345379
1957 https://www.amazon.co.jp/dp/4822292274/
<sup>1958</sup> https://www.amazon.co.jp/dp/477414259X/
<sup>1959</sup> https://www.amazon.de/dp/1497448689/
1960 https://www.theoretical-physics.net/
1961 https://info.varnish-software.com/the-varnish-book
1962 https://www.cs.princeton.edu/research/techreps/TR-941-12
1963 https://jterrace.github.io/sphinxtr/
1964 https://readthedocs.org/
1965 https://docs.spyder-ide.org/current/panes/help.html
```

PYTHON MODULE INDEX

a sphinx.addnodes, 371 sphinx.application, 324 b sphinx.builders, 147	sphinx.ext.ifconfig, 230 sphinx.ext.imgconverter, 231 sphinx.ext.imgmath, 238 sphinx.ext.inheritance_diagram, 231 sphinx.ext.intersphinx, 234 sphinx.ext.jsmath, 241 sphinx.ext.linkcode, 237
sphinx.builders.changes, 156 sphinx.builders.dirhtml, 148 sphinx.builders.dummy, 156 sphinx.builders.epub3, 151 sphinx.builders.gettext, 155 sphinx.builders.html, 148 sphinx.builders.latex, 151 sphinx.builders.linkcheck, 156	sphinx.ext.mathbase, 238 sphinx.ext.mathjax, 240 sphinx.ext.napoleon, 242 sphinx.ext.todo, 252 sphinx.ext.viewcode, 252
sphinx.builders.manpage, 153 sphinx.builders.singlehtml, 149 sphinx.builders.texinfo, 153 sphinx.builders.text, 152 sphinx.builders.xml, 157 C conf, 77	<pre>p sphinx.parsers, 370 S sphinxcontrib.applehelp, 150 sphinxcontrib.devhelp, 150</pre>
d docutils.parsers.rst, 359 sphinx.directives, 367 sphinx.domains, 362 sphinx.domains.python, 369	sphinxcontrib.htmlhelp, 149 sphinxcontrib.qthelp, 149 t sphinx.testing, 389
sphinx.environment, 352 sphinx.environment.collectors, 357 sphinx.errors, 343 sphinx.ext.autodoc, 193 sphinx.ext.autosectionlabel, 213 sphinx.ext.autosummary, 214 sphinx.ext.coverage, 218 sphinx.ext.doctest, 220 sphinx.ext.duration, 226 sphinx.ext.extlinks, 226 sphinx.ext.githubpages, 227 sphinx.ext.graphviz, 227	

660 Python Module Index

INDEX