Hieroglyph Documentation

Release 0.6.dev

Nathan R. Yergler

August 17, 2013

CONTENTS

Hieroglyph is an extension for Sphinx which builds HTML slides from ReStructured Text documents.

Whether you're already writing documentation with Sphinx, or just want to create presentations from easy to manage plain text source files, Hieroglyph can help. Check out *Getting Started with Hieroglyph* for a walk through using Hieroglyph.

CHAPTER

ONE

GETTING STARTED WITH HIEROGLYPH

Hieroglyph is an extension for Sphinx which builds HTML slides from ReStructured Text documents. Hieroglyph lets you leverage Sphinx and its large collection of extensions to create rich documents that are accessible to anyone with a web browser. It also includes tools that help you, as the presenter, to share your presentation.

This document walks through creating a presentation with Hieroglyph and Sphinx. After reading this, you will be able to use Hieroglyph to create slides, and be ready to explore additional features and extensions available through Sphinx.

1.1 Install Hieroglyph and Dependencies

To get started, you need to install Hieroglyph and its dependencies. Hieroglyph is written in Python, so if you don't have that installed, you'll need to install it first.

Once Python is installed, you can install Hieroglyph (along with an dependencies it needs with easy_install or pip.

```
$ easy_install hieroglyph
```

Installing Hieroglyph will also install its dependencies, including Sphinx and docutils, if needed.

1.2 Create a Project

After you've installed Hieroglyph and Sphinx, you can create a new project. A Sphinx project defines where to look for the source files and what extensions to enable. You can start your project using the **hieroglyph-quickstart** program included with Hieroglyph.

```
$ hieroglyph-quickstart
```

hieroglyph-quickstart will ask you questions about your presentation project. Not all of these make sense if you're just creating a presentation (as opposed to a presentation and other documentation simultaneously), so you can usually just accept the defaults.

1.3 Adding Hieroglyph to an Exiting Project

If you have an existing Sphinx project, or you used **sphinx-quickstart** instead of **hieroglyph-quickstart**, you'll need to enable Hieroglyph in the `conf.py` configuration file. Open `conf.py` and find the `extensions` definition:

```
extensions = []
```

Your definition may have items in the list if you answered "yes" to any of the Sphinx Quickstart questions. We need to add `hieroglyph` to this list:

```
extensions = ['hieroglyph']
```

That enables Hieroglyph for the project.

1.4 Authoring Slides

Once you've enabled Hieroglyph for your Sphinx project, you can begin authoring your slides. Hieroglyph uses ReStructured Text for slides, and by default sections in the document map to slides.

You can open up `index.rst` (assuming you chose the default name when you ran quickstart) and add some content.

```
====================
 Presentation Title
====================

First Slide
===========

Some content on the first slide.

Second Slide
============

* A
* Bulleted
* List
```

Here we've made three slides: a title slide (with "Presentation Title" on it), a first slide with a sentence on it, and a second slide with a bulleted list.

1.4.1 Generating Your Slides

Now that we've written some simple slides in ReStructured Text, we can generate the HTML slides from that. To do that we use of the included *Hieroglyph Builders*.

```
$ sphinx-build -b slides . ./_build/slides
```

sphinx-build will read the `conf.py` file, load the `index.rst` we've been editing, and generate the slides in the `./_build/slides` directory. After running **sphinx-build**, that directory will contain an `index.html` file, along with all of the CSS and Javascript needed to render the slides.

1.4.2 Incremental slides

It's common to have a slide with a list of items that are shown one at a time. Hieroglpyh supports this through the use of the `build` class. Let's add a third slide to `index.rst` that incrementally displays a bulleted list.

```
Show Bullets Incrementally
==========================

.. rst-class:: build
```

```
- Adding the ``build`` class to a container
- To incrementally show its contents
- Remember that *Sphinx* maps the basic ``class`` directive to
  ``rst-class``
```

Here the `rst-class` directive causes the next element to be built incrementally.

1.4.3 Displaying Images

You can include any image in a slide using the `image` directive. Just drop them in the `_static` directory in your project.

Hieroglyph also includes some support for showing an image as the full slide using the `figure` directive. For example, the Hieroglyph introductory slide deck uses the following markup:

```
.. figure:: /_static/hieroglyphs.jpg
   :class: fill

   CC BY-SA http://www.flickr.com/photos/tamburix/2900909093/
```

The caption (license information above) is styled as an overlay on the image.

1.4.4 The `slide` directive

In addition to mapping ReStructured Text sections to slides, you can create a slide at any point in your document using the `slide` directive. The `slide` directive allows you insert a slide at some place other than a heading. This can be useful when you're writing a single document that you'll present as slides as well as text. For example, if you're writing a narrative tutorial and want to include the slides in the same document, the `slide` directive makes this straight-forward.

Let's consider how the example of an incremental slide would look using the `slide` directive:

```
.. slide:: Show Bullets Incrementally
   :level: 2

   .. rst-class:: build

   - Adding the ``build`` class to a container
   - To incrementally show its contents
   - Remember that *Sphinx* maps the basic ``class`` directive to
     ``rst-class``
```

Note that here we need to specify the `level` option to let Sphinx know which level this slide corresponds to. In Sphinx and Hieroglyph, the document title is level 1, the next heading level is level 2, etc.

Unlike slides generated automatically from headings and content, slides defined using the `slide` directive will only show up when generating slides. If you generate normal HTML output or a PDF of your Sphinx project, the contents of the directive will be removed.

This example shows how to add slides with the `slide` directive, but sometimes you *only* want to use `slide` directives. In that case you can disable `autoslides`.

1.4.5 Slide-only and non-slide content

Another useful tool for mixing narrative documentation with slides is the ability to exclude content from slides or vice versa. Hieroglyph provides two directives for just this purpose. The `ifslides` directive only includes its contents

when building slides. The counterpart, `ifnotslides`, only includes its content when building other targets. The latter, in particular, may be used to include notes that you'd like to print with HTML or PDF output, but not include in the slides.

1.4.6 Presenter Notes

Use the `note` directive to insert "presenter notes" that are only visible on the presenter console. Full reStructuredText formatting is supported within the notes.

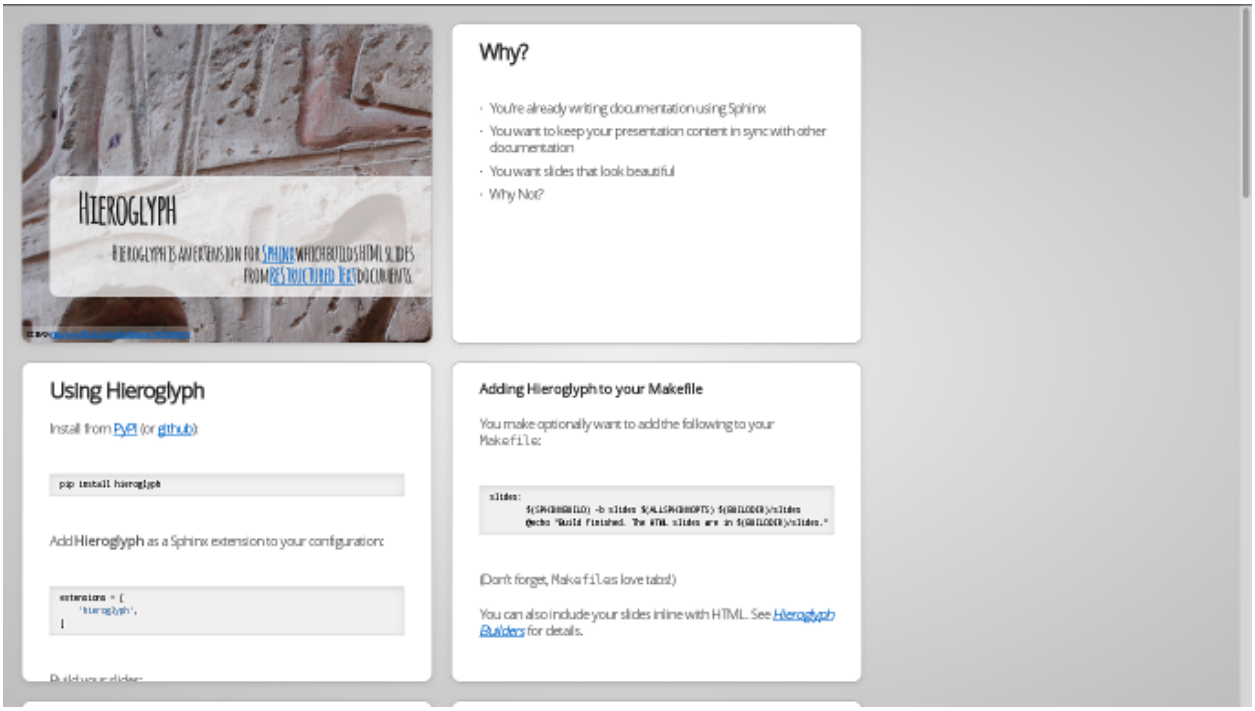
```
.. note::

   * Make sure to mention the important background story for
     this slide.
```

1.5 Viewing Your Slides

When you open the slide HTML in your browser, it looks something like this:

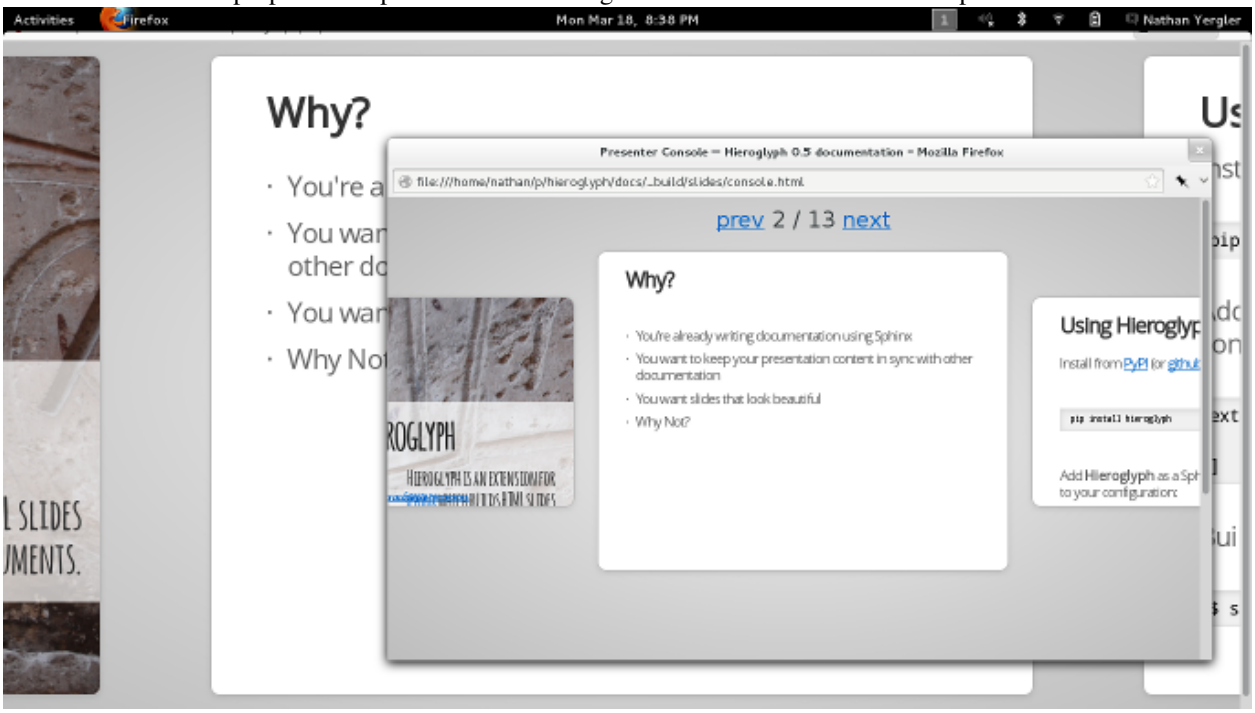
You can use the space bar to advance to the next slide, or the left and right arrows to move back and forward, respectively. Sometimes you want to skim through your slides quickly to find something, or jump ahead or back. You can use the *Slide Table* view for this. Just press `t` in the browser and the slides will shrink down.



You can click on a slide to jump there, or press `t` again to exit the slide table.

1.5.1 Presenter Console

If you're presenting your slides, it's often helpful to be able to see what's coming next. Hieroglyph includes a *Presenter's Console* for this purpose. Just press `c` when viewing the slides and the console will open in a new window.



Moving the slides backward or forward in either window will keep the other in sync.

1.6 Styling Your Slides

The simplest way to style your presentation is to add a custom CSS file. There are two steps to adding custom CSS: first, create the CSS file, and second, tell Hieroglyph to include it in the output.

Hieroglyph generates `article` tags for slides, and adds classes based on their level. That's enough to start some basic styling. Create a new file, `custom.css`, in the `_static` directory in your project. For this example, we'll change the background color of all slides to light blue, and make the title slide's text (`<h1>`) red.

```
article {
    background-color: light-blue;
}

article h1 {
    color: red;
}
```

The `_static` directory contains static assets that can be included in your output.

After you've created your CSS file, tell Sphinx about it by setting `slide_theme_options` in `conf.py`:

```
slide_theme_options = {'custom_css': 'custom.css'}
```

After you re-build your slides, you'll see the new CSS take effect.

1.7 Additional Options

Hieroglyph has several configuration options which allow you to control how it generates slides and how those slides are connected to HTML output. See *Configuration Options* for a full list.

1.8 Sphinx Extensions

Hieroglyph is built on Sphinx, which has a wide variety of extensions available. These extensions can help you create diagrams, include code snippets, render mathematical formulas, or embed maps. All of these extensions are available to Hieroglpyh, which makes it a flexible and extensible program for creating presentations.

CHAPTER

TWO

STYLING SLIDES

2.1 Styling

- Slides are contained in `<article>` elements
- Each slide has an HTML `id` that corresponds to the permalink ID generated by Sphinx (for example, you're currentling reading `styling`).
- The heading level is added as a class; ie, `level-2`
- Slides may be styled using a theme, or custom CSS.

2.2 Included Themes

Hieroglyph includes two *themes*.

`slides`

Two slides levels: the first level of headers become "section" headers, and the second become the real content.

`single-level`

Only one style of slide, every slide has a title at the top.

2.3 Setting the Theme

You can set your theme using the `slide_theme` configuration setting.

```
slide_theme = 'single-level'
```

If you're using a custom theme, you can also set the directory to look in for themes:

```
slide_theme_path = '...'
```

2.4 Incremental slides

It's common to have a slide with a list of items that are shown one at a time. Hieroglpyh supports this through the use of the `build` class. Let's add a third slide to `index.rst` that incrementally displays a bulleted list.

```
Show Bullets Incrementally
==========================

.. rst-class:: build

- Adding the ``build`` class to a container
- To incrementally show its contents
- Remember that *Sphinx* maps the basic ``class`` directive to
  ``rst-class``
```

Here the `rst-class` directive causes the next element to be built incrementally.

2.5 Displaying Images

You can include any image in a slide using the `image` directive. Just drop them in the `_static` directory in your project.

Hieroglyph also includes some support for showing an image as the full slide using the `figure` directive. For example, the Hieroglyph introductory slide deck uses the following markup:

```
.. figure:: /_static/hieroglyphs.jpg
   :class: fill

   CC BY-SA http://www.flickr.com/photos/tamburix/2900909093/
```

The caption (license information above) is styled as an overlay on the image.

2.6 Setting a Class on Slides

You can set the CSS class on a slide using the normal `rst-class` directive. (Sphinx remaps `class` to `rst-class` to avoid conflicts.) For example:

```
.. rst-class:: myclass

Slide Heading
-------------
```

The `rst-class` directive applies to the *next* following element (the heading `Slide Heading` in this example).

You can also set a default class on slides using the `slide_classes` option of the `slideconf` directive. Note that specifying an explicit class will override the `slide_classes`.

2.7 Included Styles

Hieroglyph includes some classes that for styling slides:

- `appear`

  Case the slide to just appear, replacing the previous slide, instead of sliding from the right to left.

- `fade-in`

  Causes the slide to quickly fade in and out, instead of sliding from the right to left.

2.8 Custom CSS

The standard Hieroglyph themes support adding a custom stylesheet with the `slide_theme_options` dict in `conf.py`:

```
slide_theme_options = {'custom_css': 'custom.css'}
```

The custom CSS file should be located in the `html_static_path` (`_static` by default).

2.9 Adding Javascript

In addition to a custom CSS file, it is sometimes useful to include some custom Javascript for your slides. You can put this in your static directory (`_static` by default), and then reference it in the `slide_theme_options` dict in `conf.py`:

```
slide_theme_options = {'custom_js': 'myslides.js'}
```

2.10 Creating Themes

Hieroglyph themes are based on Sphinx's HTML themes. Themes are either a directory or zipfile, which contains a `theme.conf` file, templates you wish to override, and a `static/` directory which contains images, CSS, etc.

When defining a slide theme, inherit from the `slides` theme for basic support. For example, the `single-level` them has the following `theme.conf`:

```
[theme]
inherit = slides
stylesheet = single.css

[options]
custom_css =
```

In order to include the base slide styling, your theme's stylesheet should begin with:

```
@import url(slides.css);
```

`slides.css` will be supplied by the base theme (`slides`).

See the Sphinx documentation for themes for more information.

CHAPTER

THREE

ADVANCED USAGE

3.1 The `slide` directive

Instead of (or in addition to) section headings, Hieroglyph also includes a directive that may be used to indicate a Slide should be created. The directive may have a title specified, as well as a level parameter.

For example:

```
.. slide:: The Slide Title
   :level: 2

   This Slide would appear as a level two slide.
```

3.2 Interlinking HTML Output

Hieroglyph supports linking between slides and HTML output, such as from the Sphinx HTML builders. In order to do this successfully, the slide and HTML builders used must correspond to one another. That is, the `SlideBuilder` must be used with the `StandaloneHTMLBuilder`, and the `DirectorySlideBuilder` must be used with the `DirectoryHTMLBuilder`.

For example, runnning:

```
$ make html slides
```

Will generate HTML and slides if interlinking is enabled. See *Interlinking HTML Output* for information on enabling interlinking in the configuration.

3.3 Per-File Configuration

When working with multi-file projects, there may be cases when it is desirable to override the theme or set configuration value for specific files. This can be accomplished using the `slideconf` directive:

```
.. slideconf::
   :theme: single-level
```

Values specified in a `slideconf` directive override defaults specified in `conf.py`. If more than one `slideconf` appears in a document, only the last one is used.

CHAPTER

FOUR

CONFIGURATION OPTIONS

Hieroglyph supports several configuration settings, which can be set in the project's Sphinx configuration file. If you used `sphinx-quickstart` to begin your project, this will be `conf.py` in the project directory.

4.1 Basic Configuration

autoslides
Default: `True`

When `autoslides` is True, Hieroglyph will generate slides from the document sections. If autoslides is set to False, only generate slides from the `slide` directive.

This can be overridden on a per-document basis using the `slideconf` directive.

slide_theme
Default: `slides`

The theme to use when generating slides. Hieroglyph includes two themes, `slides` and `single-level`.

This can be overridden on a per-document basis using the `slideconf` directive.

See *Styling Slides* for more information.

slide_levels
Default: `3`

Number of Sphinx section levels to convert to slides; note that the document title is level 1. Heading levels greater than slide levels will simply be treated as slide content.

4.2 Slide Numbers

slide_numbers
Default: `False`

If set to `True`, slide numbers will be added to the HTML output.

4.3 Themes

slide_theme_options
Default: `{}`

Theme specific options as a `dict`.

See *Custom CSS* for more information.

slide_theme_path
Default: `[]`.

A list of paths to look for themes in.

For more information on styling and themes, see *Styling Slides*.

4.4 Interlinking HTML Output

Interlinking HTML Output can be enabled for slides, HTML, or both.

slide_link_to_html
Default: `False`

Link from slides to HTML.

slide_link_html_to_slides
Default: `False`

Link from HTML to slides.

slide_link_html_sections_to_slides
Default: `False`

Link individual HTML sections to specific slides.

Note that `slide_link_html_to_slides` must be enabled for this to have any effect.

4.4.1 Relative Paths

The slide/HTML interlinking needs to know how to find the slide and HTML output from the other side. There are two configuration parameters for this. They're configured to work with Sphinx and Hieroglyph's standard configuration (output in sub-directories of a common build directory) by default .

slide_relative_path
Relative path from HTML to slides; default: `../slides/`

slide_html_relative_path
Relative path from slides to HTML; default: `../html/`

4.4.2 Additional Parameters

slide_html_slide_link_symbol
Default: §

Text used to link between HTML sections and slides.

This text is appended to the headings, similar to the section links in HTML output.

CHAPTER

FIVE

DIRECTIVES

.. ifslides::
Include the directive contents in the output only when building slides. That is, when one of the *Hieroglyph Builders* is used.

.. ifnotslides::
Exclude the contents of the directive from output when building slides. That is, when one of the *Hieroglyph Builders* is used.

Note: `ifslides` and `ifnotslides` were originally named `slides` and `notslides`, respectively. They were renamed prior to the addition of the `slide` directive, in order to be more explicit.

The old names work, but will show a warning during the build process. Expect the old names to be removed in some future version.

.. slideconf::
Configure slide-related options for the current document.

Some of the *Configuration Options* options can be overridden on a per document basis.

The `theme` option, if present, will set the theme for document. See the *theme documentation* for more information on themes.

The `autoslides` option, if present, must be `True` or `False`. If set to `True`, slides will be generated from the document headings and contents. If `autoslides` is `False`, slides will only be created with Sphinx encounters the *The slide directive*.

The `slide_classes` option allows you to specify classes that will be added to slides by default. This allows you, for example, to add a class that applies some styling to the slides. Note that if a slide has an explicit class set (ie, with the `rst-class` directive), the classes specified here *will not* be applied.

See *Per-File Configuration* for more information and examples.

.. slide:: `title`
Create a slide in the document. The directive takes the slide title as its argument, and some optional settings for the slide. For example:

```
.. slide:: Example Slide
   :level: 2

   This is an example slide.

   * Bullet 1
   * Bullet 2
```

The `level` option, if present, will set the level of the slide, which is used for *styling slides*.

By default, content contained in a `slide` directive will be excluded when building non-slide output. You can change this behavior by setting the `inline-contents` option to `True`. When `inline-contents` is set to `True`, the contents of the `slide` directive will be included in all output.

The `class` option, if present, will add the given class to the slide output.

The following example will set the class `red-slide` on the slide output, and include the slide content (the sentence and the bulleted listed, but not the title) in HTML output.

```
.. slide:: Warning!
   :level: 2
   :class: red-slide
   :inline-contents: True

   This error can occur when:

   * Microwaving metal
   * Leaving the gas on
   * Using a frayed electrical cord
```

CHAPTER

SIX

HIEROGLYPH BUILDERS

In Sphinx parlance, a "builder" is an output target. Sphinx includes several of its own, including ones for HTML pages, ePub documents, and PDF.

Hieroglyph adds additional builders for generating slides. The builder's "name" must be given to the **-b** command-line option of **sphinx-build** to select a builder.

You may want to add one (or more) of the Hieroglyph builders to your `Makefile` to make it easier to run the Sphinx builder.

For example, to add the `slides` builder to your Makefile, add the following target:

```
slides:
        $(SPHINXBUILD) -b slides $(ALLSPHINXOPTS) $(BUILDDIR)/slides
        @echo "Build finished. The HTML slides are in $(BUILDDIR)/slides."
```

(Remember, makefiles are indented using tabs, not spaces.) Available slide building classes.

class `hieroglyph.builder.`**`SlideBuilder`**(*app*)
This is the standard Slide HTML builder.

Its output is a directory with HTML, along with the needed style sheets, slide table, and presenter's console JavaScript.

Its name is `slides`.

class `hieroglyph.builder.`**`DirectorySlideBuilder`**(*app*)
This is the standard Directory Slide 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/`.

The output directry will include any needed style sheets, slide table, and presenter's console JavaScript.

Its name is `dirslides`.

class `hieroglyph.builder.`**`InlineSlideBuilder`**(**args*, ***kwargs*)
This is the Inline Slide HTML builder.

The inline slide builder add support for the `slide` directive to Sphinx's `StandaloneHTMLBuilder`, and adds an additional stylesheet to the output for basic inline display.

When using an inline builder `autoslides` is disabled.

Its name is `inlineslides`. New in version 0.5.

class hieroglyph.builder.**DirectoryInlineSlideBuilder**(**args*, ***kwargs*)
This is the Inline Slide Directory HTML builder.

The inline slide builder add support for the `slide` directive to Sphinx's `DirectoryHTMLBuilder`, and adds an additional stylesheet to the output for basic inline display.

When using an inline builder `autoslides` is disabled.

Its name is `dirinlineslides`. New in version 0.5.

6.1 Abstract Builders

Hieroglyph also defines two abstract builders. These classes are not capable of building slides on their own, but encapsulate most of the slide-specific functionality.

class hieroglyph.builder.**AbstractSlideBuilder**

apply_theme(*themename*, *themeoptions*)
Apply a new theme to the document.

This will store the existing theme configuration and apply a new one.

get_theme_config()
Return the configured theme name and options.

pop_theme()
Disable the most recent theme, and restore its predecessor.

post_process_images(*doctree*)
Pick the best candidate for all image URIs.

class hieroglyph.builder.**AbstractInlineSlideBuilder**(**args*, ***kwargs*)

CHAPTER

SEVEN

REFERENCES & INDICES

- *genindex*
- *modindex*
- *search*

CHAPTER

EIGHT

LICENSE

Hieroglyph is made available under a BSD license; see LICENSE for details.

Included slide CSS and JavaScript originally based on HTML 5 Slides licensed under the Apache Public License.

CHAPTER

NINE

RELATED PROJECTS

- Sphinx
- Docutils
- rst2s5
- HTML 5 Slides

PYTHON MODULE INDEX

h