Hieroglyph Documentation

Release 0.7.1.dev0

Nathan R. Yergler

Jul 31, 2019

Contents

1 Getting Started with Hieroglyph **3**
- 1.1 Install Hieroglyph and Dependencies . . . 3
- 1.2 Create a Project . . . 3
- 1.3 Adding Hieroglyph to an Existing Project . . . 4
- 1.4 Authoring Slides . . . 4
- 1.5 Viewing Your Slides . . . 7
- 1.6 Styling Your Slides . . . 9
- 1.7 Additional Options . . . 9
- 1.8 Sphinx Extensions . . . 9

2 Styling Slides **11**
- 2.1 Styling . . . 11
- 2.2 Included Themes . . . 11
- 2.3 Setting the Theme . . . 11
- 2.4 Incremental slides (builds) . . . 12
- 2.5 Setting a Class on Slides . . . 12
- 2.6 Slide Classes . . . 12
- 2.7 Custom CSS . . . 13
- 2.8 Slide Transitions . . . 13
- 2.9 Adding Javascript . . . 13

3 Theme Features **15**
- 3.1 Slides & Single Level . . . 15
- 3.2 Slides2 . . . 15

4 Advanced Usage **19**
- 4.1 The `slide` directive . . . 19
- 4.2 Splitting Sections with the `nextslide` directive . . . 19
- 4.3 Interlinking HTML Output . . . 20
- 4.4 Per-File Configuration . . . 20

5 Configuration Options **21**
- 5.1 Basic Configuration . . . 21
- 5.2 Slide Numbers . . . 22
- 5.3 Slide Footer . . . 22
- 5.4 Themes . . . 22
- 5.5 Interlinking HTML Output . . . 22

6 Creating Themes **25**
6.1 theme.conf . . . 25
6.2 layout.html . . . 26
6.3 slide.html . . . 26
6.4 Additional Pages . . . 26

7 Restructured Text Directives **29**

8 Hieroglyph Builders **31**
8.1 Abstract Builders . . . 32

9 Developing Hieroglyph **33**
9.1 Running Tests . . . 33

10 References & Indices **35**

11 Release History **37**
11.1 News . . . 37

12 License **41**

13 Related Projects **43**
13.1 Hieroglyph Smoke Test . . . 43

Python Module Index **47**

Index **49**

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.

You can connect with other Hieroglyph users and the developers via the hieroglyph-users email list. A Gmane archive is also available.

CHAPTER 1

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.

Note: Attention Mac users

Mac users may run into an error where a dependency isn't found, such as this error below.

```
$ hieroglyph-quickstart
Traceback (most recent call last):
  File "/usr/local/bin/hieroglyph-quickstart", line 5, in <module>
    from pkg_resources import load_entry_point
        :
pkg_resources.DistributionNotFound: six
```

This is a result of having your installed version of Python conflict with the one that Apple provides as part of Mac OS X. This may be rectified simply by editing the first line of the newly-installed `/usr/local/bin/hieroglyph-quickstart`. Change it from `#!/usr/bin/python` to `#!/usr/bin/env python`.

Another issue you may run into is that the Sphinx wrapper may require a specific version, i.e., anything that looks like "==1.1.2" in `/usr/local/bin/sphinx-build`. If you've got another version of Sphinx already installed, then it's likely newer and will be able to handle it. IOW, just remove references to "==1.1.2" in that file, and it should work.

1.3 Adding Hieroglyph to an Existing 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
============
```

(continues on next page)

(continued from previous page)

```
* 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 one of the included *Hieroglyph Builders*.

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

As an alternative, if you have `make` on your system, the quickstart installs a `slides` directive in the `Makefile` which executes `sphinx-build`, so all you'd need to do is the following:

```
$ make 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 Quotes

A standard reStructuredText quote will be interpreted as a quote slide, multiple quotes or additional content (on the same slide) are not supported.

The attribution is standard reStructuredText, and optional.

Note that most themes include a `quote` class, which you can apply to the `slide` directive (or the section) for better formatting.

1.4.5 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 straightforward.

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*.

1.4.6 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.

1.4.7 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.

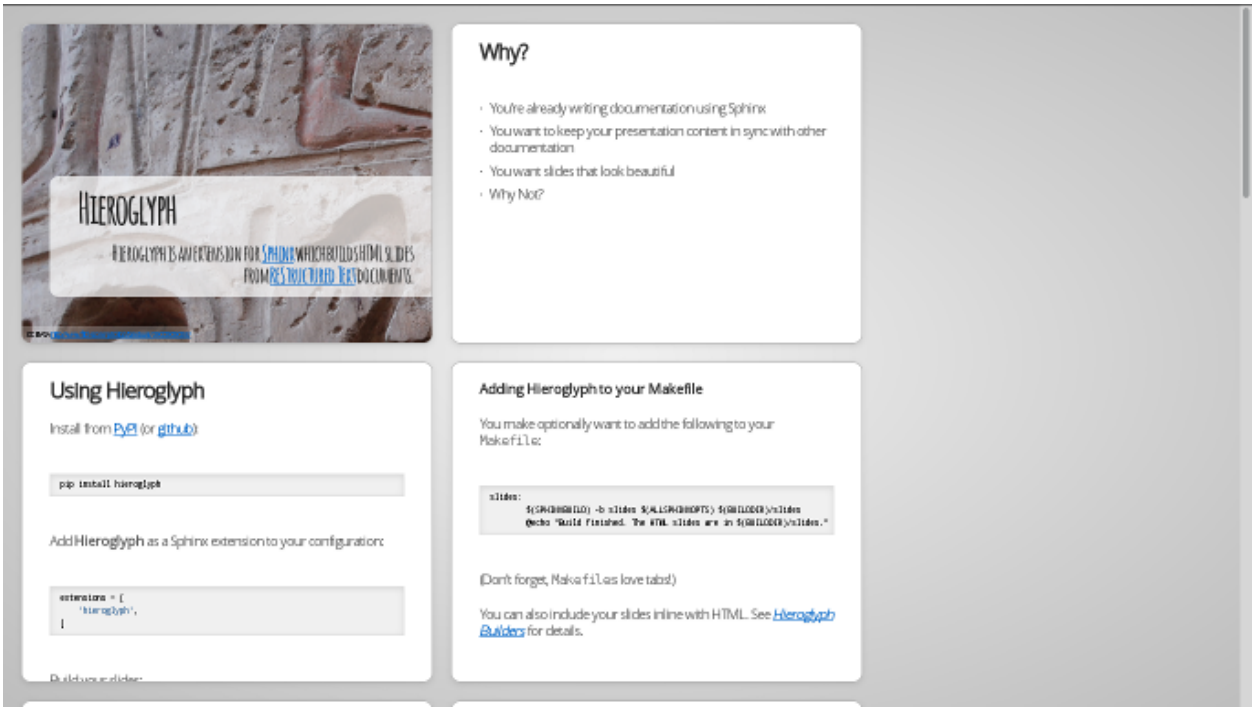
```
.. note::

   * 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:

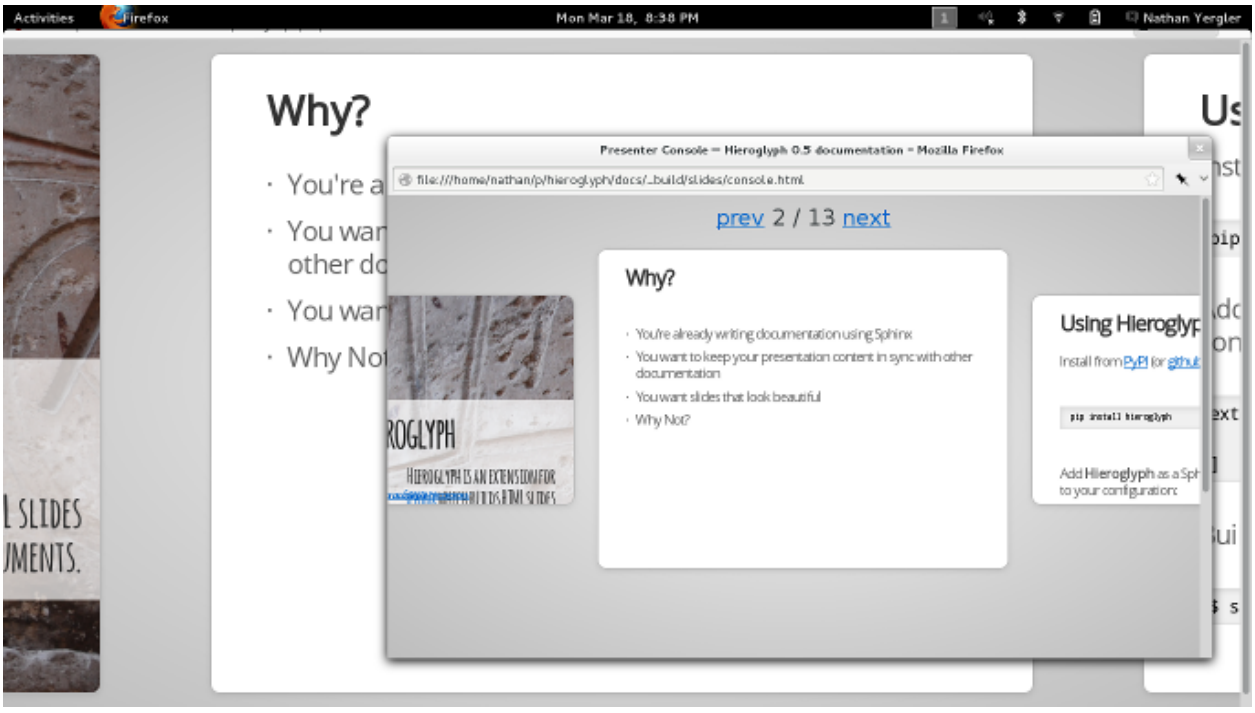
You can use the space bar to advance to the next slide, or the left and right arrows to move back and forward, respectively. Sometimes you want to skim through your slides quickly to find something, or jump ahead or back. You can use the *Slide Table* view for this. Just press `t` in the browser and 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 2

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.
- You can enable *`slide numbers`* or a *`slide footer`* with configuration settings.

2.2 Included Themes

Hieroglyph includes three *themes*.

`slides`

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

`single-level`

Based on the default `slides` theme. Only one style of slide, every slide has a title at the top.

`slides2`

Based on the updated (2012+) Google I/O HTML slides template. This theme is new in 0.7.

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 (builds)

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 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`.

2.6 Slide Classes

Hieroglyph includes some pre-defined style classes.

`title-image`

> Designed to be used as a title slide with a full screen image. Use the *figure* directive to specifiy the image and caption.

2.7 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).

2.8 Slide Transitions

Most themes use a default transition between slides. For the *slides* and *slides2* theme, the next slide moves in from the right, sliding on top of the previous one. This isn't always desirable: sometimes you want to skip the transition so that you can do more interesting "builds" using multiple slides.

The slide transitions are implemented using CSS transitions. The slides2 CSS includes the following declaration:

```
slides > slide {
  -webkit-transition: all 0.6s ease-in-out;
  -moz-transition: all 0.6s ease-in-out;
  -o-transition: all 0.6s ease-in-out;
  transition: all 0.6s ease-in-out;
}
```

This tells the browser to take 0.6s to transition between slides.

If you want to omit the transition altogether, you can add *Custom CSS* to override this.

```
slides > slide {
  -webkit-transition: none;
  -moz-transition: none;
  -o-transition: none;
  transition: none;
}
```

If you only want to selectively change the transition timing, you can define a class and *set a class on the slide*.

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'}
```

CHAPTER 3

Theme Features

3.1 Slides & Single Level

3.1.1 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.

3.1.2 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.

3.2 Slides2

The `slides2` theme was added in Hieroglyph 0.7, and as based on the Google I/O 2012+ HTML slide templates.

3.2.1 Theme Options

The `slides2` theme requires presentation metadata in the `conf.py` file. You can specify one or more presenters; presenter information will be included on the title and end slides automatically.

```
slide_theme_options = {
    'presenters': [
        {
            'name': 'The Author',
            'twitter': '@author',
            'www': 'http://example.com/author',
            'github': 'http://github.com/author/example'
        },
    ],
}
```

In addition to the presenter metadata, the following options may be specified in `slide_theme_options`:

`subtitle` Default: `""`

The presentation title will be taken from `conf.py`; if you would like to display a sub-title on the title slide, specify it here.

`use_builds` Default: `true`

`use_prettify` Default: `true`

`enable_slide_areas` Default: `true`

`enable_touch` Default: `true`

`favicon` Default: `""`

3.2.2 Title & End Slides

The title and end slides contain presentation metadata and links. Unlike the other slides, they are generated directly from template fragments. You can override these by providing a `title_slide.html` or `end_slide.html` template in the `_templates` directory of your project.

For example, `title_slide.html` with a full-bleed background image might look like this:

```
<slide class="title-slide segue nobackground fill"
       style="background-image: url(_static/insect_trap.jpg)">
  <hgroup class="auto-fadein">
    <h1 class="white" data-config-title><!-- populated from slide_config.json --></h1>
    <h2 data-config-subtitle><!-- populated from slide_config.json --></h2>
    <h2 data-config-presenter><!-- populated from slide_config.json --></h2>
  </hgroup>
  <footer class="source white">
    CC BY-NC-SA // www.flickr.com/photos/boobook48/5041751802/
  </footer>

</slide>
```

An `end_slide.html` template might look like this:

```
<slide class="thank-you-slide segue nobackground">
  <article class="flexbox vleft auto-fadein">
    <h2>Thank You!</h2>
```

(continues on next page)

(continued from previous page)

```
  </article>
  <p class="auto-fadein" data-config-contact>
    <!-- populated from slide_config.json -->
  </p>
</slide>
```

3.2.3 Displaying Images

3.2.4 Included Styles

3.2.5 Incremental Slides (Builds)

In addition to the *common incremental slide support*, the `slides2` theme supports more granular builds. Items with the class `build-item-x` (where `x` is a number) will be incrementally display, in numerical order.

For example, you can show items from bottom to top on a slide:

```
.. rst-class:: build-item-3

This will be shown third

.. rst-class:: build-item-2

This will be shown second

.. rst-class:: build-item-1

This will be shown first
```

If multiple items have the same number, they will both be displayed at the same time.

Warning: `build-item-*-only` and `build-item-*class-*` are experimental and their behavior may change considerably as we learn more.

Items may also be displayed *only* at a specific index. That is, displayed, then hidden again. Appending the suffix `-only` to the `build-item-` class activates this behavior.

CHAPTER 4

Advanced Usage

4.1 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.
```

4.2 Splitting Sections with the `nextslide` directive

In addition to section headings and the *slide* directive, text sections may be split into multiple slides using the *nextslide* directive.

When building slides, *nextslide* will split the content at the point of the directive and copy the section title.

Consider the following example:

```
Section Title
=============

some content

.. nextslide::

additional content
```

When building slides, this will generate two slides with the name **Section Title**.

A different title may be specified as an argument to *nextslide*.

The `increment` argument will add an index to the subsequent slide titles. For example:

```
Section Title
=============

some content

.. nextslide::
   :increment:

additional content

.. nextslide::
   :increment:

conclusion
```

Will generate three slides, with the titles **Section Title**, **Section Title (2)**, and **Section Title (3)**, respectively.

4.3 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.

4.4 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 5

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.

5.1 Basic Configuration

`slide_title`
Default: inherit from `html_title`

Sets the title of slide project generated. This title will be used in the HTML title of the output.

`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.

5.2 Slide Numbers

slide_numbers
Default: `False`

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

5.3 Slide Footer

slide_footer
Default: `None`

Text that will be added to the bottom of every slide.

5.4 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*.

5.5 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.

5.5.1 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/`

5.5.2 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 6

Creating Themes

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

To create a new theme, follow these steps:

1. Create a directory for your theme.

   This directory will need to be somewhere Sphinx and Hieroglyph can find it. You can set the html_theme_path in your project to the path, if needed. If you plan to distribute your theme, simply add the sphinx_themes entry point.

2. Create a `theme.conf`

   `theme.conf` defines your theme's options and what theme it's based on. Good choices are `basic` (bare-bones Sphinx theme), `slides`, or `slides2` (Hieroglyph themes).

   The Sphinx documentation describes what goes into theme.conf.

3. Create `layout.html`, if needed.

   If your theme needs any custom HTML around the slides content, override `layout.html`. At the minimum, this template *must* define a `body` block.

4. Create a `slide.html` file.

   The `slide.html` template is rendered for each individual slide.

5. Specify additional pages to generate in the `theme.conf`

6.1 theme.conf

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`).

6.2 layout.html

The `layout.html` template defines the container for your presentation. This includes links to CSS and Javascript files, as well as any markup needed for the presentation.

Sphinx (and therefore Hieroglyph) uses Jinja for templating. The Sphinx templating documentation has general information and a list of available helper functions.

When creating links in `layout.html` (and other templates), it's important to use the `pathto` function. The pathto function will ensure the link is generated with the correct path when the presentation is built.

6.3 slide.html

The `slide.html` template is rendered for each individual slide. The following variables are available for your templates:

`id` The HTML ID for this slide. This is generated by Sphinx from the title content.

`title` The slide title. This may contain nested HTML if the slide title includes inline markup.

`level` The slide heading level.

`content` HTML content for the slide.

`content_classes` A list of all classes that had the `content-` prefix. The prefix is stripped in the list.

> This may be useful if your markup requires classes for elements that don't directly map to RST constructs. For example, the `slides2` theme uses this for setting classes on the inner `article` HTML element.

`slide_classes` A list of remaining classes; ie, those without the `content-` prefix.

`classes` A list of all classes assigned to the slide, regardless of prefix. No prefixes are stripped in `classes`.

`slide_number` The slide number for the current slide.

`config` A reference to the Sphinx Configuration.

6.4 Additional Pages

Hieroglyph also allows specification of extra pages to build in the theme configuration. Any key in `options` that begins with `extra_pages_` specifies an additional page to be built. The base `slides` theme specifies the console in this manner:

```
[options]
custom_css =
custom_js =
extra_pages_console = console.html
```

The value of the key (`console.html` in this case) specifies the template to use to render the page.

CHAPTER 7

Restructured Text 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
```

.. nextslide:: `title`

Splits the content at the directive when building slides. An option title may be specified as an argument. If not specified, the title of the current section will be copied.

Consider the following example:

```
Section Title
=============

some content

.. nextslide::

additional content
```

When building slides, this will generate two slides with the name **Section Title**.

The `increment` argument, if present, will append an index to the slide title.

The `classes` arguments, if present, contains a list of classes that will be applied to the newly created section.

CHAPTER 8

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.

8.1 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.

get_theme_options()
Return a dict of theme options, combining defaults and overrides.

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 9

Developing Hieroglyph

Hieroglyph uses Buildout to manage dependencies and development.

1. Check out the repository:

```
$ git clone git@github.com:nyergler/hieroglyph.git
```

2. Bootstrap and run buildout:

```
$ python bootstrap.py
$ ./bin/buildout
```

After running Buildout, you can run `./bin/python` to execute an interpreter with Hieroglyph and its dependencies installed.

9.1 Running Tests

The unit tests can be run via `setup.py`:

```
$ ./bin/python setup.py test
```

Tox can be used to run the tests with both Python 2 and 3. The Tox configuration will run the tests with Sphinx 1.1.x, Sphinx 1.2.x, and the development branch. Note that Hieroglyph requires Tox 1.8.

```
$ tox
```

9.1.1 Jasmine Tests for Javascript

There are some Jasmine tests in `src/jstests` that test theme Javascript functionality. You can open `src/jstests/SpecRunner.html` in your browser to run those. Alternately, you can install the `jasmine` gem to do so.

If you have Bundler installed, get started by installing the necessary gems:

```
$ bundle install
```

Then run the tests using `rake`:

```
$ rake jasmine:ci JASMINE_CONFIG_PATH=./src/jstests/jasmine.yml
```

CHAPTER 10

References & Indices

- genindex
- modindex
- search

CHAPTER 11

Release History

11.1 News

11.1.1 0.7.1

Release date: 28 March 2015

- Add *title-iamge* slide class
- Bug fix: *slides2* theme includes extra slides (#92)
- Bug fix: entry point invocation relies on new setuptools (#94)
- Bug fix: RST quotes raise exceptions (#93)

11.1.2 0.7

Release date: 16 March 2015

- Themes now use a template fragment for rendering individual slides. (Issue #49)
- Bug fix: files in `html_static_dir` will now override theme files. (Issue #54)
- Added `slide_title` configuration setting. (Issue #56)
- Added `slides2` theme, based on updated Google HTML 5 slide templates. (Issue #48)
- Added more flexible incremental slides using build-item-* classes.
- Theme creation documentation (#53)
- Section slide CSS updated for printing. (Issue #37)
- Handle titles with embedded markup w/ nextslide directive. (#85)
- Improved quickstart script

11.1.3 0.6.5

Release date: 1 March 2014

- Revert "Javascript loading is now deferred until the end of the document." This caused issues with content in ifnotslides blocks. (Issue #33)
- Fixed an issue with path mangling for generated images (ie, from blockdiag).
- Added support for `slide_footer` (Issue #44)
- Converted slide condition directive processing to use Docutils transforms. This allows section headings to appear in tables of contents correctly. (Issue #25)
- Added `nextslide` directive for splitting sections. (Issue #46)
- Section classes are now added to the generated slides (Issue #50).

11.1.4 0.6

Release date: 8 August 2013

- The `note` directive in a slide now shows up as notes in the presenter console. Thanks, Doug Hellmann!
- Javascript loading is now deferred until the end of the document.
- Allow projects to specify custom Javascript to be included with slides.
- slides.js now fires an event when the slides are resized.
- Support setting default classes on slides in a document.
- Added `appear` and `fade-in` slide classes for alternate transitions.
- Added `hieroglyph-quickstart` script.
- Added testing framework, initial tests for directives
- Fixed bug where content was not removed with autoslides was disabled
- Slides created with the slide directive may omit have only a title, or only content (Issue #30)
- Slide numbering was often incorrect when dealing with multiple slide levels; this has been correct (Issue #26)
- Better page break handling when printing slides (PR #31). Thanks, tjadevries!

11.1.5 0.5.5

Release date: 19 March 2013

- Rewrote, updated, and expanded documentation, including the addition of the Getting Started guide.
- Added `inline-contents` option to the `slide` directive.
- Fixed bug with image path calculation for documents in nested trees. This primarily impacted images generated by other extensions, such as blockdiag.
- Added support for marking a section as a slide when autoslides are disabled.
- All slide-related nodes are now left intact when pruning the tree.
- Fixed bug related to changing themes between documents that resulted in Sphinx reporting Template Not Found.
- Fixed level calculation for slides created with the `slide` directive.

- Simplified processing of `slideconf` nodes: previously an attempt was made to remove them when not building slides. This was fragile, and led to breakage in the latex and texinfo builders. They're now skipped properly for all builtin Sphinx builders.
- Updated Javascript for incremental slides to work with recent builds of Chrome

11.1.6 0.5

Release date: 24 December 2012

- Added support for `slide` directive
- Added `autoslides` config parameter to allow disabling automatic generation of slides from document text.
- Added inline slide builder.
- Renamed `slides` and `notslides` directives to `ifslides` and `ifnotslides`, respectively. The old names will continue to work for a while, the rename just makes them more expressive.
- Changed key for toggling slide table view to `t` (was `ESC`).
- Fixed problems with styling nested lists
- Fixed incompatibility with latex-pdf builder

11.1.7 0.4

Release date: 27 September 2012

- Print-specific styling for printing slides
- Template and javascript clean-up/reorganization
- More accurate display of scaled slides on Slide Table
- Initial implementation of Presenter Console
- Themes and docs include font files locally
- Changed interlink configuration keys to be more consistent.
- Support for file-specific theme configuration
- Support for slide numbering

11.1.8 0.3.2

Release date: 5 June 2012

- Correctly generate relative links between HTML & Slides

11.1.9 0.3.1

Release date: 5 June 2012

- Added content, code missing from the 0.3 release.
- Updated README to reflect changes in 0.3.
- Changed docs configuration to build HTML + Slides.

11.1.10 0.3

Release date: 4 June 2012

- Provide directory and standalone based builders.
- Added `slides` and `notslides` directives.
- Fix up absolute image paths from things like blockdiag
- Preliminary support for linking between HTML to Slides
- Preliminary slide table support

Backward Incompatible Changes:

- Builders have been renamed to `slides` and `dirslides`. If your `Makefile` refers to `html5slides` or `dirhtml5slides`, you will need to update it.

11.1.11 0.2

Release date: 10 March 2012

- Initial implementation of Sphinx builder.
- Two themes: slides and single-level
- Basic documentation

CHAPTER 12

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 13

Related Projects

- Sphinx
- Docutils
- rst2s5
- HTML 5 Slides

13.1 Hieroglyph Smoke Test

This is the Hieroglyph Smoke Test. It contains visual tests for verifying Hieroglyph functionality.

This section should not be included in slides.

13.1.1 Bulleted Lists

- First Point
- Second Point
- Third Point

Another notation:

- First Point
- Second Point
- Third Point

13.1.2 Enumerated Lists

This list will be numbered:

1. First Point
2. Second Point
3. Third Point

This list will be lettered:

A. First Point
B. Second Point
C. Third Point

Nested Lists

- First
- Second
  - Sub Item 1
  - Sub Item 2

13.1.3 Code Highlighting

This block will be highlighted as Python:

```
def func(a):
    print 'The value of a is %(a)s' % locals()
```

This block will be highlighted as Javascript:

```
function func(a) {
    console.log('The value of a is ', a);
}
```

This block will not be highlighted:

```
def func(a):
    """I am not highlighted."""
```

13.1.4 Admonitions

The `note` admonition is used to create notes in the presenter console.

Note: This is a *note* admonition. It will not appear in the slides.

Warning: Warnings, however, stay where they belong.

Note: Notes can appear anywhere in the slide content.

13.1.5 Hieroglyph Features

The following slides test Hieroglyph features.

Incremental Slides

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

Splitting Sections

The `nextslide` directive will split a single section into multiple slides.

The `increment` option tells Hieroglyph to add `(2)` (and subsequent indices) to the title.

Python Module Index

h

hieroglyph.builder, 31

Index

A

AbstractInlineSlideBuilder (*class in hieroglyph.builder*), 32
AbstractSlideBuilder (*class in hieroglyph.builder*), 32
apply_theme() (*hieroglyph.builder.AbstractSlideBuilder method*), 32
autoslides
    configuration value, 21

C

configuration value
    autoslides, 21
    slide_footer, 22
    slide_html_relative_path, 23
    slide_html_slide_link_symbol, 23
    slide_levels, 21
    slide_link_html_sections_to_slides, 22
    slide_link_html_to_slides, 22
    slide_link_to_html, 22
    slide_numbers, 22
    slide_relative_path, 23
    slide_theme, 21
    slide_theme_options, 22
    slide_theme_path, 22
    slide_title, 21

D

DirectoryInlineSlideBuilder (*class in hieroglyph.builder*), 32
DirectorySlideBuilder (*class in hieroglyph.builder*), 31

G

get_theme_config() (*hieroglyph.builder.AbstractSlideBuilder method*), 32
get_theme_options() (*hieroglyph.builder.AbstractSlideBuilder method*), 32

H

hieroglyph.builder (*module*), 31

I

ifnotslides (*directive*), 29
ifslides (*directive*), 29
InlineSlideBuilder (*class in hieroglyph.builder*), 31

N

nextslide (*directive*), 30

P

pop_theme() (*hieroglyph.builder.AbstractSlideBuilder method*), 32
post_process_images() (*hieroglyph.builder.AbstractSlideBuilder method*), 32

S

slide (*directive*), 29
slide_footer
    configuration value, 22
slide_html_relative_path
    configuration value, 23
slide_html_slide_link_symbol
    configuration value, 23
slide_levels
    configuration value, 21
slide_link_html_sections_to_slides
    configuration value, 22
slide_link_html_to_slides
    configuration value, 22
slide_link_to_html

configuration value, 22
slide_numbers
configuration value, 22
slide_relative_path
configuration value, 23
slide_theme
configuration value, 21
slide_theme_options
configuration value, 22
slide_theme_path
configuration value, 22
slide_title
configuration value, 21
SlideBuilder (*class in hieroglyph.builder*), 31
slideconf (*directive*), 29