slidy2 backend plugin for Asciidoc

Jean-Michel Inglebert <inglebert@iut-blagnac.fr>

Revision History

Revision 1.0.3	August, 2011	JMI

Table of Contents

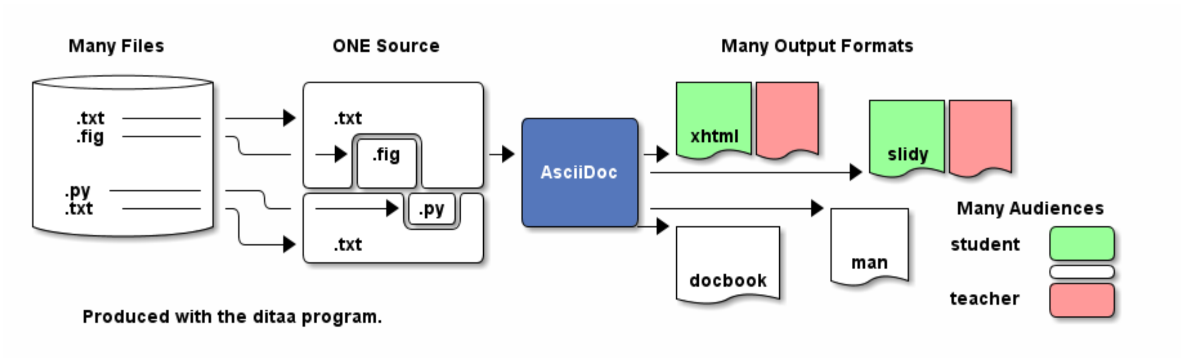
AsciiDoc from *Stuart Rackham* is a Text based document generation tool.

W3C HTML Slidy © from *Dave Raggett* is an HTML slideshow tool.

This document describes the *slidy2* backend plugin which extends the distributed AsciiDoc *slidy* backend.

This backend plugin requires AsciiDoc 8.6.6 or newer to run.

LaTeX, DocBook and AsciiDoc follow this reusability template : *One Source* splitted in *Many Files* to produce *Many Output Formats* for *Many Audiences*.



1. Goal

How to produce Slidy output format from any AsciiDoc source file without defining a new document structure to produce slides.

LaTeX beamer and prosper or DocBook slides tools require to define a new document structure to produce slides.

2. Added features

1. Produces a slide for each AsciiDoc section level
2. Easily merges too short sections contents in one slide
3. Easily splits long contents to few slides
4. Extends incremental display scope
5. Adds an incremental images block
6. Adds slide *footnotes*
7. Adds SVG *callouts*
8. Adds a *backgound* block and a `slidebackground` attribute
9. Adds a `slidefontsizeadjust` attribute

2.1. A slide for each *AsciiDoc* section level

With this contrib, a new slide is produced *on each* AsciiDoc *section level* (1..4).

When numbering is not set, the `:slidetitleindentcar:` attribute value is inserted for each subsection title.

The `slidetitleindentcar` character defaults to » and can be redefined at any point with the following statement :

```
:slidetitleindentcar: *
```

2.2. Merging sections contents

If you want to merge some sections contents, simply insert the following new *nopagebreak* block macro (within a conditional block to not interfere with your non-slide outputs) :

```
ifdef::backend-slidy2[>>>]
```

the *nopagebreak* instruction takes effect only for the next section. You should repeat it to merge more than one section.

any section level (1..4) can be merged in this way.

For exemple, the next three sections will appear on the same slide in this *slidy2* output.

level 3 section

some content

level 3 section merged

some content

level 4 section merged

some content

2.3. Splitting too long content

If you want to split some long content, simply insert the usual AsciiDoc *pagebreak* (within a conditional block to avoid pagebreak in your non-slide outputs) :

```
ifdef::backend-slidy2[<<<]
```

the `pagebreak` template will recall the last *section title* on each generated subslide and give it a subslide number.

To preserve AsciiDoc document structure

1. do not insert conditionnal pagebreak inside AsciiDoc *blocks*
2. do not insert conditionnal pagebreak inside AsciiDoc *tables*

3. do not insert conditionnal pagebreak inside AsciiDoc *numbered list* (items numbering will restart)

See *AsciiDoc* userguide *Slidy* output for some examples.

2.4. Incremental display extended to

- paragraph
- listingblock
- literalblock
- quoteblock
- verseblock
- qanda list
- callout list
- table

You have to switch to the slidy output format to see how incremental display works on theses AsciiDoc constructs.

2.5. Incremental images block

You can define incremental images blocks as follows :

```
images::last[first,second,...]
```

You can use `data-uri` attribute to embed incremental images.

It's a *slidy2* **only** macro, so you have to use it in a conditionnal block. For exemple :

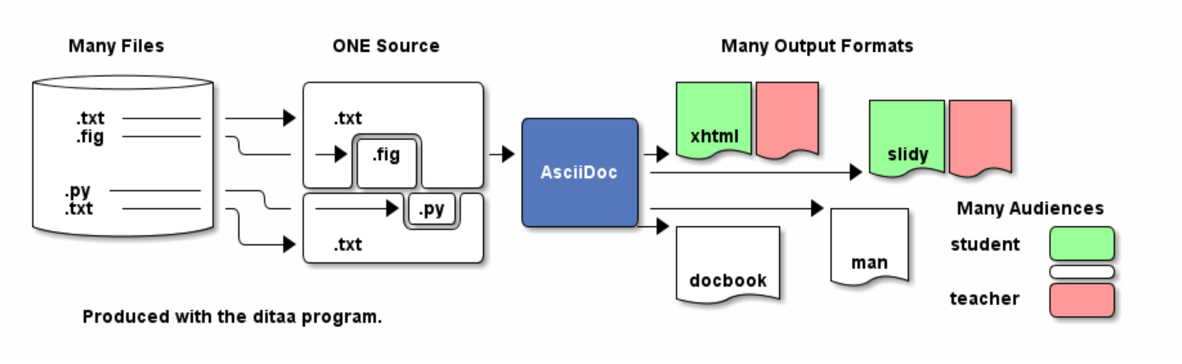
```
ifdef::backend-slidy2[]
 images::last[first,second,...]
endif::backend-slidy2[]

ifndef::backend-slidy2[]
 image::last[]
endif::backend-slidy2[]
```

The following statement :

```
images::./images/osmfmoma4.png[./images/osmfmoma1.png,./images/osmfmoma2.png,./images/os
```

renders incremently in *Slidy* output :



2.6. Footnotes

In *Slidy* output, *footnotes* are placed at the bottom of the slide where they are defined.

A footnote [1], a second footnote [2], a footnote with a reference ID [3] and a reference to another footnote [6] (defined in the next slide in slidy output).

In slidy output, this slide illustrates `footnote[]`, `footnoteref[]` and `pagebreak` combination.

Two more footnotes [4] [5], a footnote with a reference ID [6] and a reference to another footnote [3] (defined in the previous slide in slidy output).

Caveats

1. There is no generated link for this *Slidy footnotes*.
2. Do not insert *newline* character inside your AsciiDoc *footnote* text.

2.7. SVG callouts

With SVG, callout icons are bigger and will gracefully scale on window resizing.

`callout-inlinemacro` and `listtags-callout` produce an `object` tag which uses SVG or PNG images depending on the browser SVG compatibility.

Callout produced markup.

```
<object data="{icon={iconsdir}/callouts/{index}.svg}"          ❶❷
        type="image/svg+xml" title="{index}"                   ❸
        width="{slidecalloutwidth=4%}">                        ❹
 <img src="{iconsdir}/callouts/{index}.png" alt="{index}" />   ❺
</object>                                                      ❻
```

❶❻ HTML `object` tag

[1]A first example footnote.
[2]A second example footnote.
[3]myId referenced footnote.
[4]A third example footnote.
[5]A fourth example footnote.
[6]anotherId referenced footnote.

❷❸ SVG icon
❹ default SVG icon width
❺ PNG icon used if SVG is not supported

`slidecalloutwidth` attribute

At any point you can change *callout width* with the following statement :

```
ifdef::backend-slidy2[:slidecalloutwidth: 7%]   ❶
```

❶ use a relative unit (%) if you want that callout icons scale with window size.

and you can return to the default with :

```
ifdef::backend-slidy2[:slidecalloutwidth!:]
```

How SVG callouts were generated

The following *python program* was used in the `backends/slidy2/images/icons/callouts/` directory to produce SVG callouts.

svg_callouts.py.

```
#!/usr/bin/env python
# creates SVG callout icons
# to run under the backends/slidy2/images/icons/callouts/ directory

numbers = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
for nb in numbers:
 file = "%s.svg" % nb
 try:
  f = open(file, "wt")
  f.write('<?xml version="1.0" standalone="no"?>\n')
  f.write('<?xml-stylesheet href="callout.css" type="text/css"?>\n')
  f.write('<svg viewBox="0 0 100 100" xmlns="http://www.w3.org/2000/svg" version="1.1">\
  f.write('<g>\n')
  f.write('  <circle cx="50" cy="50" r="50" class="callout" />\n')
  f.write('  <text x="50" y="80" text-anchor="middle" font-family="Trebuchet MS" font-we
  f.write('</g>\n')
  f.write('</svg>\n')
  f.close()
  print "%s created" % file
 except IOError, (errno, strerror):
  print "*** ERROR %s : File %s *** %s" % (errno, file, strerror)
```

SVG callouts style

Each generated callout svg file references the `callout.css` stylesheet.

You can customize *color* and *font* by putting the following definitions in the `images/icons/callouts/callout.css` stylesheet.

callout.css.

```
circle.callout { fill: red; }                  ❶❷
text.callout { fill: blue; font-family: Times; } ❸❹
```

❶❸ SVG css style for SVG tag
❷❹ you cannot put this style in your usual css stylesheet

2.8. Slidy background block

You can define *Slidy* backgrounds at any point in your document source, as follows :

```
[optional backgroundName,optional background block CSS style]
&&&&
 AsciiDoc markup
&&&&
```

You can use `data-uri` attribute to embed background images.

Do not give a name to define the default background

For example, the following definition creates the first background used in the *Slidy* output of this document :

```
[asciidocslidy,opacity: 0.4; position: absolute; left: 60%; top: 40%;]
&&&&
[width="100%",cols="7,^3",frame="none",grid="none"]
|===========================================
| | *Powered by*
| | image:./images/asciidoc.png["AsciiDoc"]
| | *and*
| | image:./images/w3c_home.png["W3C"] *Slidy*
|===========================================
&&&&
```

2.9. `slidebackground` attribute

At any point you can switch to some background with the following statement :

```
 ifdef::backend-slidy2[:slidebackground: asciidocslidy]
```

and you can disable any background with the following one :

```
 ifdef::backend-slidy2[:slidebackground!:]
```

2.10. `slidefontsizeadjust` attribute

If you set the `slidefontsizeadjust` attribute, a `meta` markup will be inserted in the *Slidy* output.

As usual, you can set `slidefontsizeadjust` attribute on the *command line* :

```
asciidoc ... -a slidefontsizeadjust=-2 ...
```

or in your *source header* :

```
:slidefontsizeadjust: -2
```

3. *slidy2* backend quickref

Table 1. Attributes

Attribute	Header Only	Notes
`:copyright:`	Yes	Footer content
`:duration:`	Yes	Estimated number of minutes
`:incremental:`		When set, produces *Slidy* incremental output
`:slidetitleindentcar:`		Inserted before subsection levels , if numbering is off. (default=»)
`:slidecalloutwidth:`		SVG callouts width (default=4%)
`:slidebackground:`		Switch to the given background
`:slidefontsizeadjust:`	Yes	Globaly adjust fontsize (+-N)

Table 2. Macros

Macro	Notes
`images::last[first,second,…]`	Produces incremental images markup
>>>	Merges the next section with the current one
<<<	Splits current content

Table 3. Blocks

Block	Notes
&&&&	Defines a Slidy background

4. *AsciiDoc* userguide *Slidy* output

For a more complete example consider the *AsciiDoc* userguide *Slidy* output produced with this contrib.

The *AsciiDoc 8.6.5* userguide *slidy* output produced with the AsciiDoc **default** *slidy* backend contains 47 slides with many too long contents.

The following `asciidoc.txt` **customization** have introduced :

1. about 100 *slidy2* conditionnal `pagebreak` (<<<) to split long contents
2. about 20 *slidy2* conditionnal `nopagebreak` (>>>) to merge shorts sections
3. about 10 *slidy2* conditionnal `pagebreak` to split long lists or tables

to produce this userguide865_slidy2.txt customized source file and this userguide865_slidy2.slidy.html *Slidy* output (with 311 slides).

5. Extra features added to W3C Slidy © slideshows

W3C HTML Slidy © from *Dave Raggett* is an HTML slideshow tool.

5.1. Added features

The *slidy2* backend plugin distribution contains a modified version of the W3C Slidy © `slidy.js` file that enable you to :

1. toggle incremental display during a slideshow
2. toggle css color-sets during a slideshow
3. display the keys mapping during a slideshow

5.2. Usage

1. Press **I** to toggle incremental display ON/OFF at any point during a slideshow

1. Press **L** to circularly toggle css color-set at any point during a slideshow

   1. create your css color set and give it a name containing *color_set*
   2. link or include your color sets as usual :

      a. `<link rel="stylesheet" href="/path/to/your_color_set.css" type="text/css" />`

      b. `<style type="text/css" title="your_color_set"> … </style>`

   With Firefox, you can switch through the *View* → *Page Style* menu.

2. Press **M** to show the key mapping at any point during a slideshow

5.3. Implementation notes

Incremental display toggle

1. `w3c_slidy.incremental_display` added
2. in each incremental display related method, sets `incremental` to `false` if `w3c_slidy.incremental_display` is `false`
3. keyboard **I** key added to toggle `w3c_slidy.incremental_display` value

css color-sets toggle

1. `w3c_slidy.color_sets` array added
2. `w3c_slidy.init_color_sets()` added
3. `w3c_slidy.toggle_color_set()` added
4. keyboard **L** key added to circularly toggle `w3c_slidy.color_set` value

Some predefined color sets :

- ../stylesheets/slidy2_color_set_black.css
- ../stylesheets/slidy2_color_set_blue.css
- ../stylesheets/slidy2_color_set_green.css
- ../stylesheets/slidy2_color_set_none.css
- ../stylesheets/slidy2_color_set_yellow.css

key mapping display

1. `w3c_slidy.keymap` added
2. keyboard **M** key added to display the key mapping

Source diff file

- slidy_to_slidy2.diff

6. How to use *slidy2* backend plugin

This backend plugin requires AsciiDoc 8.6.6 or newer to run.

6.1. Install the *slidy2* backend plugin

This AsciiDoc backend plugin is hosted at http://code.google.com/p/asciidoc-slidy2-backend-plugin/.

1. Download the latest release (1.0.3) zip file from http://code.google.com/p/asciidoc-slidy2-backend-plugin/downloads/list
2. Install the release with one of the following AsciiDoc `backend` commands :

```
> asciidoc --backend install slidy2-v1.0.3.zip ❶
> asciidoc --backend install slidy2-v1.0.3.zip /etc/asciidoc/backends ❷
```

❶ will install *slidy2* backend in the user homedir
❷ will install *slidy2* backend in the given place

6.2. Use the *slidy2* backend plugin

You can produce your *Slidy* outputs with the following command :

```
> asciidoc --backend slidy2 -o doc.slidy.html doc.txt
```

note Add the `data-uri` attribute to produce a self contained presentation (with css, javascript and images included) :

```
> asciidoc -a data-uri --backend slidy2 -o doc.slidy.html doc.txt
```

7. How to generate *slidy2* documentation

```
> cd .../backends/slidy2/doc
> edit make.txt paths accordingly to your platform
> asciidoc -a nt    --unsafe make.txt        ❶
> asciidoc -a posix --unsafe make.txt        ❷
```

❶ on windows © platforms
❷ on Unix like platforms

note theses commands will produce a `make.html` trace file.

8. How to run *slidy2* backend plugin tests

You can run test suite on any platform from the `tests` directory :

```
> cd .../backends/slidy2/tests
> python slidy2_UnitTest.py
```