rst2html5slides Documentation

Release 1.0

André Felipe Dias

Sep 27, 2017

Contents

Presentations based on web frameworks such as impress.js, jmpress.js and deck.js are great alternatives to traditional software packages:

1. HTML, CSS and javascript files are plain text files and thus they are friendly to version control, making easy to follow and visualize their changes between revisions;
2. There is a separation between content and presentation design
3. They have new and great visual effects such as slide transitions and positional effects
4. A web presentation is portable. It runs on any modern browser

On the other hand, creating presentations directly in HTML is painful. A better approach is to use another markup language that is easier to read and write, and then translate the text to HTML.

I could have listed or cooked up a few reasons to justify reStructuredText as the best choice. The truth is that the best markup language is the one you already know. So, if you also use reStructuredText in other projects or contexts, then rst2html5slides is perfect for you to create your presentations.

CHAPTER 1

Usage

rst2html5slides extends rst2html5 to generate a deck of slides from a reStructuredText file that can be used with any web presentation framework such as impress.js, jmpress.js or deck.js.

rst2html5slides source is located at http://bitbucket.org/andre_felipe_dias/rst2html5slides but you can install it with pip:

```
$ pip install rst2html5slides
```

Later, you will also need to install or download the chosen web presentation framework. For now, let's concentrate on making presentations.

Important: Before proceeding, it is important that you already know reStructuredText basic syntax. I'd like to suggest two links:

- The official reStructuredText Primer
- The Sphinx reStructuredText Primer

Making Presentations

By default, every section of a reStructuredText (rst) document becomes a slide. A typical presentation in reStructured-Text (rst) has the following pattern:

```
Topic 1
=======

* Item A
* Item B


Topic 2
```

```
=======

* Item C
* Item D
```

When converted by rst2html5slides, the result is:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
</head>
<body>
<deck>
    <slide id="topic-1">
        <header>
            <h1>Topic 1</h1>
        </header>
        <section>
            <ul>
                <li>Item A</li>
                <li>Item B</li>
            </ul>
        </section>
    </slide>
    <slide id="topic-2">
        <header>
            <h1>Topic 2</h1>
        </header>
        <section>
            <ul>
                <li>Item C</li>
                <li>Item D</li>
            </ul>
        </section>
    </slide>
</deck>
</body>
</html>
```

Note: Note that both `<deck>` and `<slide>` are custom HTML tags. The former wraps all slides while the latter wraps each slide. You can style and select them in CSS and Javascript but you also can change them if prefer or need to *adapt your presentation to a framework or stylesheet*.

A rst section is initiated by a title of any level, but in rst2html5slides, only sections from the highest level become slides:

```
Slide 1
=======

Slide contents

Slide 2
=======

Subtitle
```

```
--------

* item A
* item B
```

will be translated to:

```
...
<body>
<deck>
    <slide id="slide-1">
        <header>
            <h1>Slide 1</h1>
        </header>
        <section>Slide contents</section>
    </slide>
    <slide id="slide-2">
        <header>
            <h1>Slide 2</h1>
            <h2>Subtitle</h2>
        </header>
        <section>
            <ul>
                <li>item A</li>
                <li>item B</li>
            </ul>
        </section>
    </slide>
</deck>
</body>
</html>
```

You are not limited to section titles, paragraphs and lists. All other reStructuredText constructs are available to your presentation such as images, tables, code-blocks etc.

Untitled Slides

Sometimes, you might want an untitled slide. This can be accomplished using a transition marker:

```
Untitled slide 1

----

Untitled slide 2

----

Untitled slide 3
```

which becomes:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
</head>
```

```
<body>
<deck>
    <slide>
        <section>Untitled slide 1</section>
    </slide>
    <slide>
        <section>Untitled slide 2</section>
    </slide>
    <slide>
        <section>Untitled slide 3</section>
    </slide>
</deck>
</body>
</html>
```

Some Useful reStructuredText Directives to Presentations

reStructuredText has two useful directives to register some document metadata directly into the HTML head section: title and meta:

```
.. title:: My Presentation Title
.. meta::
    :author: André Felipe Dias
    :keywords: mercurial, web presentation, rst2html5slides
```

which translates to:

```
<!DOCTYPE html>
<html>
<head>
    <title>My Presentation Title</title>
    <meta charset="utf-8" />
    <meta content="André Felipe Dias" name="author" />
    <meta content="mercurial, web presentation, rst2html5slides" name="keywords" />
</head>
<body></body>
</html>
```

Another important directive is class, which sets the “class” attribute value on its content or on the first immediately following non-comment element. When making presentations, you use `class` directive to set the class of a slide:

```
.. class:: logo-background

Topic X
=======

paragraph


.. class:: section-title

----

New Chapter

```

```
Topic Y
=======

.. class:: hint

* This also works
* Substructure here
```

which translates to:

```
...
<deck>
    <slide class="logo-background" id="topic-x">
        <header>
            <h1>Topic X</h1>
        </header>
        <section>paragraph</section>
    </slide>
    <slide class="section-title">
        <section>New Chapter</section>
    </slide>
    <slide id="topic-y">
        <header>
            <h1>Topic Y</h1>
        </header>
        <section>
            <ul class="hint">
                <li>This also works</li>
                <li>Substructure here</li>
            </ul>
        </section>
    </slide>
</deck>
...
```

Slide Attributes

Slide attributes are set via rst fields declared just **above the desired slide**. There is no restriction on field name or value. They are directly included as slide tag attributes. Example:

```
:id: Opening
:data-x: 1000
:data-y: 500
:data-scale: 3

Presentation Opening
```

resulting in:

```
...
<deck>
    <slide data-y="500" data-x="1000" id="Opening" data-scale="3">
        <section>Presentation Opening</section>
    </slide>
```

```
</deck>
...
```

Manual Positioning of Slides

You can position slides manually through `data-*` attributes which are used by impress.js and jmpress.js to set the position/zoom/rotation of a slide. The most common fields are:

- `data-x`: The horizontal position of a slide in pixels. Can be negative.
- `data-y`: The vertical position of a slide in pixels. Can be negative.
- `data-z`: This controls the position of the slide on the z-axis. Setting this value to -3000 means it's positioned -3000 pixels away. This is only useful when you use data-rotate-x or data-rotate-y, otherwise it will only give the impression that the slide is made smaller, which isn't really useful.
- `data-scale`: Sets the scale of a slide, which is what creates the zoom. Defaults to 1. A value of 4 means the slide is four times larger. In short: Lower means zooming in, higher means zooming out.
- `data-rotate-x`: The rotation of a slide in the x-axis, in degrees. This means you are moving the slide in a third dimension compared with other slides. This is generally cooll effect, if used right.
- `data-rotate-y`: The rotation of a slide in the x-axis, in degrees.
- `data-rotate-z`: The rotation of a slide in the x-axis, in degrees. This will cause the slide to be rotated clockwise or counter-clockwise.
- `data-rotate`: The same as data-rotate-z.

Unless there is an automatic distribution function defined, the same set of `data-*` attributes are applied to the following slides until you overwrite some of their values. For example:

```
:data-x: 1000
:data-y: 1000

slide 1

:data-x: 2000

----

slide 2

:data-y: -1000

----

slide 3
```

results in:

```
...
<deck>
    <slide data-x="1000" data-y="1000">
        <section>slide 1</section>
    </slide>
    <slide data-x="2000" data-y="1000">
        <section>slide 2</section>
    </slide>
    <slide data-x="2000" data-y="-1000">
        <section>slide 3</section>
    </slide>
</deck>
...
```

Note that `slide 2` kept the same `data-y` value from `slide 1` and `slide 3` has the same `data-x` value from `slide 2`.

Attention: Positional attributes must be declared above the desired slide. So, `:data-x: 2000` refers to `slide 2` while `:data-y: -1000` changes `slide 3`.

Automatic Positioning of Slides

Manual positioning is annoying if all you need is a simple presentation. In such cases, you should use one of the automatic distribution functions provided by `rst2html5slides`. For the time being, there are three functions available:

1. `linear`: horizontal distribution by regular increments of `data-x`.
2. `grid`: similar to linear, but a new line is created at every 4 slides. This number can be changed passing an additional parameter.
3. `grid_rotate`: similar to `grid`, but the slide is rotated in 180 degrees when line changes.

To define an automatic positioning function, use a presentation directive at the beginning of the presentation:

```
.. presentation
    :distribution: grid 2

slide 1

----

slide 2

----

slide 3
```

which translates to:

```
...
<deck>
    <slide data-x="0">
        <section>slide 1</section>
    </slide>
    <slide data-x="1600">
        <section>slide 2</section>
    </slide>
    <slide data-y="1600" data-x="0">
        <section>slide 3</section>
    </slide>
```

```
</deck>
...
```

Tip: `grid 1` makes a column.

Increment Values

The default value for increment `data-x` and `data-y` is 1600. To change this, specify different values with `increment` attribute declaring one or two values (see the next section for an example). A single value will be applied to both `data-x` and `data-y`.

Interfering on Automatic Positioning

Even using automatic positioning, you can change the position of one slide setting some `data-*` attributes. The automatic distribution will restart from this slide.

Example:

```
.. presentation::
    :distribution: grid_rotate 2
    :increment: 1000 800

slide 1

:data-x: -500
:data-y: -1000
:data-scale: 3
:data-rotate-z: 90

----

The automatic positioning restarts at this slide
due to its custom positioning

----

Same line yet

----

New line, with rotation
```

resulting in:

```
...
<deck>
    <slide data-x="0" data-rotate-z="0">
        <section>slide 1</section>
    </slide>
    <slide data-y="-1000" data-x="-500" data-rotate-z="90" data-scale="3">
        <section>The automatic positioning restarts at this slide due to its custom␣
↪positioning</section>
```

```
    </slide>
    <slide data-y="-1000" data-x="500" data-rotate-z="90" data-scale="3">
        <section>Same line yet</section>
    </slide>
    <slide data-y="-200" data-x="500" data-rotate-z="269.9" data-scale="3">
        <section>New line, with rotation</section>
    </slide>
</deck>
...
```

Adapting to a Presentation Framework

Each presentation framework has its particularities or choices regarding style and structure of slides, but it is not always necessary to change the structure of a raw rst2html5slides translation. For example, jmpress.js is rather flexible. You could configuring it to accept a `deck` tag as the root element and `slide` tags as steps:

```
$(function() {
    $('deck').jmpress({
        stepSelector: 'slide'
    });
});
```

rst2html5slides translations can also be used unchanged in deck.js:

```
$(function() {
    $.deck("slide");
});
```

Adjusting Presentation Structure

You might want to change the rst2html5slides default structure to adapt its result to a framework structure or to an existent style. For example, *impress.js* expects a presentation strictly follow the structure:

```
<body>
    <div id="impress">
        <div class="step">
            ...
        </div>
        <div class="step">
            ...
        </div>
        ...
    </div>
</body>
```

rst2html5slides result structure can be configured by `--deck-selector` and `--slide-selector` parameters or by a `presentation` directive with `deck_selector` and `slide_selector` options. To configure rst2html5slides to conform to impress.js you can use:

```
$ rst2html5slides --deck-selector 'div#impress' --slide-selector 'div.step' ...
```

or

```
.. presentation::
    :deck_selector: div#impress
    :slide_selector: div.step
```

Tip: Follow the CSS selector format to specify tag, class and id.

Templates

Just changing HTML structure isn't enough. You also need to include all framework files in presentations. One way is passing parameters to rst2html5slides:

- `--script`
- `--script-defer`
- `--stylesheet`

Example:

```
$ rst2html5slides.py \
    --stylesheet jmpress.css \
    --script http://code.jquery.com/jquery-latest.min.js \
    --script jmpress.js \
    --script-defer jmpress_init.js \
    --deck-selector '#jmpress' \
    example.rst example.html
```

However, the simplest way is through templates where all necessary links, scripts and even additional constructs are already declared and the job of rst2html5slides is just filling in the contents of the presentation. Example:

```
<!DOCTYPE html>
<html {html_attr}>
<head>
    <!-- styles and scripts for a jmpress.js presentation -->
    <meta content="width=device-width, maximum-scale=1.0, initial-scale=1.0, user-
↪scalable=yes" name="viewport" />
    <link href="css/jmpress.css" rel="stylesheet" />
    <link href="css/default.css" rel="stylesheet" />
    <link href="css/pygments.css" rel="stylesheet" />
    <script src="http://code.jquery.com/jquery-latest.min.js"></script>
    <script src="js/jmpress.js"></script>
    <script src="js/jmpress_init.js" defer="defer"></script>
    {head}
<body>{body}
<div class="hint">
    <p>Use a spacebar or arrow keys to navigate</p>
</div>
</body>
</html>
```

```
$ rst2html5slides --template jmpress_template.html example.rst example.html
```

CHAPTER 2

Examples

Important: All plataform files (such as deck.js or jmpress.js) are supposed to be already downloaded at `css` and `javascript` directories.

1. simple.rst

A simple presentation with 4 slides. One of them has a subtitle and another one doesn't have a title.

```
Slide 1
=======

Slide contents

Slide 2
=======

Subtitle
--------

* item A
* item B

----

No title here

Final Slide
===========

This is the last one
```

To convert it to a deck presentation, there is no additional adjustments needed since deck.js follows a linear flow. The command to build it is shown below:

```
rst2html5slides \
    --template templates/deck-template.html \
    simple.rst simple_deck.html
```

You can see the result here.

To build a jmpress/impress presentation, it is necessary to define some sort of positional distribution. Fortunately, `rst2html5slides` provides a few default functions. The command to build a jmpress presentation from the same previous simple presentation is:

```
rst2html5slides \
    --template templates/jmpress_template.html \
    --stylesheet 'css/simple.css' \
    --deck-selector 'div#jmpress' \
    --slide-selector 'div.step' \
    --distribution linear \
    simple.rst simple_jmpress.html
```

See the result: simple_jmpress.html.

2. jmpress.rst

Impress.js has a nice demo presentation that shows how it works. Jmpress.js also has the same presentation. Now, rst2html5slides shows how to produce that presentation from a reStructuredText file:

```
.. title:: impress example | jmpress.js | rst2html5slides
.. meta::
    :viewport: width=device-width, maximum-scale=1.0, initial-scale=1.0, user-
↪scalable=yes

.. presentation::
    :deck-selector: div#jmpress
    :slide-selector: div.step


:data-x: -1000
:data-y: -1500
:id: bored
:class: slide

Aren't you just **bored** with all those slides-based presentations?


:data-x: 0
:class: slide

Don't you think that presentations given in **modern browsers** shouldn't
**copy the limits** of 'classic' slide decks?


:data-x: 1000
:class: slide
```

```
Would you like to **impress your audience** with
**stunning visualization** of your talk?


:data-x: 0
:data-y: 0
:data-scale: 4
:id: title

.. role:: try
.. role:: footnote
.. role:: title

:try:`then you should try`

:title:`impress.js` :sup:`*`

:footnote:`* no rhyme intended`


:data-x: 850
:data-y: 3000
:data-rotate: 90
:data-scale: 5
:id: its

It's a **presentation tool**
inspired by the idea behind `prezi.com <http://prezi.com>`_
and based on the **power of CSS3 transforms and transitions** in modern browsers.


:data-x: 3500
:data-y: 2100
:data-rotate: 180
:data-scale: 6
:id: big

.. role:: thoughts

visualize your **big** :thoughts:`thoughts`


:id: tiny
:data-x: 2825
:data-y: 2325
:data-z: -3000
:data-rotate: 300
:data-scale: 1

and **tiny** ideas


:id: ing
:data-x: 3500
:data-y: -850
:data-z: 0
:data-rotate: 270
:data-scale: 6
```

```
.. role:: positioning
.. role:: rotating
.. role:: scaling

by :positioning:`positioning`, :rotating:`rotating` and :scaling:`scaling`
them on an infinite canvas


:id: imagination
:data-x: 6700
:data-y: -300
:data-scale: 6
:data-rotate: 0
:data-z: 0

.. role:: imagination

the only **limit** is your :imagination:`imagination`


:id: source
:data-x: 6300
:data-y: 2000
:data-rotate: 20
:data-scale: 4

want to know more?
`use the source <http://github.com/bartaz/impress.js>`_, Luke!


:id: one-more-thing
:data-x: 6000
:data-y: 4000
:data-rotate: 0
:data-scale: 2

one more thing...


:id: its-in-3d
:data-x: 6200
:data-y: 4300
:data-z: -100
:data-rotate-x: -40
:data-rotate-y: 10
:data-scale: 2

.. role:: have
.. role:: you
.. role:: noticed
.. role:: its
.. role:: in
.. role:: footnote

:have:`have` :you:`you` :noticed:`noticed` :its:`it's` in **3D**:sup:`*`?
:footnote:`* beat that, prezi :)`
```

```

:id: overview
:data-x: 3000
:data-y: 1500
:data-scale: 10
:data-rotate-x: 0
:data-rotate-y: 0
:data-rotate-z: 0

.. empty slide
```

You can build it using a template:

```
rst2html5slides \
    --template templates/jmpress_template.html \
    jmpress.rst jmpress_via_template.html
```

or via parameters:

```
rst2html5slides \
    --stylesheet "http://fonts.googleapis.com/css?family=Open+Sans:regular,semibold,
↪italic,italicsemibold|PT+Sans:400,700,400italic,700italic|PT+Serif:400,700,
↪400italic,700italic" \
    --stylesheet css/impress.css \
    --script http://code.jquery.com/jquery-latest.min.js \
    --script js/jmpress/jmpress.js \
    --script-defer js/jmpress/jmpress_init.js \
    jmpress.rst jmpress_via_parameters.html
```

The first and the second generated files are equivalent and pretty close to the originals, aren't they?