HACKING asciidoctor-reveal.js

Table of Contents

Short instructions that aim to help potential contributors.

Getting Started

- Setup the Asciidoctor-revealjs plugin in development mode
- Modify the slim templates in `templates/`
- Templates need to be compiled before being used, do so with:

```
bundle exec rake build
```

- Then using the following command will render slides with your template changes baked in:

```
bundle exec asciidoctor-revealjs <source.adoc>
```

The next section will provide further help on how to use `print` statements or a debugger to assist development.

Inspect the template system

To understand what you have access to in templates you can inject some ruby. With the slim templating system, this is done by prepending the lines with a dash (`-`) and inserting a ruby statement. Two complementary approaches can be used to explore the context offered by asciidoctor through the template system:

- logging on the command line via print-like statements
- jump into the context through an interactive debugger

NOTE Debugging is only supported via the Ruby ecosystem. You need to recompile the templates when you make changes to them. This can be done by running:

```
bundle exec rake build
```

Print debugging information

For example to see which attributes are available, you can print them by adding these lines in the `.slim` file of interest:

```
- puts @document.attributes.inspect
- puts @attributes.inspect
- puts @document.methods
```

Other generally useful ruby specific introspection:

```
- puts instance_variables
- puts local_variables
```

One might find `pp` to produce better output (and in some cases not):

```
- require 'pp'
- pp @document.attributes
```

Interactively debug a template

Pry is a powerful debugger for ruby that features tab-completion. It is very useful to discover a complex object hierarchy like what asciidoctor offers.

Initial Setup

```
bundle --path=.bundle/gems --binstubs=.bundle/.bin
```

Usage

In order to be dropped into the debugger at a specific point in a template simply add the following two lines in the relevant `.slim` template file:

```
- require 'pry'
- binding.pry
```

Recompile the templates with:

```
bundle exec rake build
```

Then run `asciidoctor-revealjs` from the command-line to generate your document and you'll be dropped in the debugger:

```
$ bundle exec asciidoctor-revealjs examples/video.adoc
asciidoctor: WARNING: level-sections.adoc: line 29: section title out of sequence:
expected level 2, got level 3

From: /home/olivier/src/asciidoc/asciidoctor-
reveal.js/templates/slim/section.html.slim @ line 3 :

    1: - hide_title = (title = self.title) == '!'
    2: - require 'pry'
 => 3: - binding.pry
    4: / parent section of vertical slides set
    5: - if @level == 1 && !(subsections = sections).empty?
    6:   section
    7:     section id=(hide_title ? nil : @id) data-transition=(attr 'data-
transition') data-transition-speed=(attr 'data-transition-speed') data-
background=(attr 'data-background') data-background-size=(attr 'data-background-size')
data-background-repeat=(attr 'data-background-repeat') data-background-
transition=(attr 'data-background-transition')
    8:       - unless hide_title

[1] pry(#<Asciidoctor::Section>)>
```

Then using commands like the following allows you to explore interactively asciidoctor's API and object model with syntax highlighting:

```
[1] pry(#<Asciidoctor::Section>)> @document
```

You can also query asciidoctor's documentation:

```
[4] pry(#<Asciidoctor::Section>)> ? find_by
```

If you install the `pry-byebug` gem you get additional debugging capabilities. See the gem's documentation for details.

Since 1.1.0, templates are compiled. It is easier to inject the debug triggering statements and use the templates directly instead of debugging compiled templates. You can call the slim templates directly with:

```
bundle exec asciidoctor-revealjs --trace -T templates/ examples/customcss.adoc
```

References

- https://github.com/asciidoctor/asciidoctor.org/issues/80#issuecomment-145698579
- http://pryrepl.org/

- http://discuss.asciidoctor.org/Interactively-debugging-a-template-with-a-REPL-td4498.html

Manual Tests

In order to help troubleshoot issues and test syntax improvements, some minimalist asciidoc test files are provided. You can render the tests files and then load them in a browser and check if `asciidoctor-revealjs` behaves as expected.

Initial Setup

Make sure to have a working version of `asciidoctor-reveals` this is usually done with `bundler`:

```
bundle config --local github.https true
bundle --path=.bundle/gems --binstubs=.bundle/.bin
bundle exec rake build
```

Go to `test/doctest` folder and install `reveal.js`:

```
cd test/doctest/
git clone https://github.com/hakimel/reveal.js.git
```

Render tests into .html

From the project’s root directory:

```
bundle exec rake doctest::generate FORCE=yes
```

Open rendered files

NOTE Right now, doctest issue #12 means that the generated examples will not be pretty.

You can open the generated `.html` in `test/doctest/` in a Web browser.

Asciidoctor API’s gotchas

Attribute inheritence

The attr and attr? methods inherit by default. That means if they don’t find the attribute defined on the node, they look on the document.

You only want to enable inheritance if you intend to allow an attribute of the same name to be controlled globally. That might be good for configuring transitions. For instance:

```
= My Slides
:transition-speed: fast

== First Slide
```

However, there may be attributes that you don’t want to inherit. If that’s the case, you generally use the form:

```
attr('name', nil, false)
```

The second parameter value is the default attribute value, which is nil by default.

Relevant documentation: http://www.rubydoc.info/github/asciidoctor/asciidoctor/Asciidoctor%2FAbstractNode%3Aattr

Merge / Review policy

Any non-trivial change should be integrated in master via a pull-request. This gives the community a chance to participate and helps write better code because it encourages people to review their own patches.

Pull requests should come from personal forks in order not the clutter the upstream repository.

Wait time

Once a pull request is submitted, let it sit for 24-48 hours for small changes. If you get positive feedback you can merge before the sitting time frame. If you don’t get feedback, just merge after the sitting time frame.

Larger changes should sit longer at around a week. Positive feedback or no feedback should be handled like for small changes.

Breaking changes should sit until a prominent contributor comments on the changes. Ping `@mojavelinux` and `@obilodeau` if necessary.

Remember that this is a slower moving project since people are not designing slides everyday. Well, for most people.

Work-in-progress pull-requests

Letting know to the maintainers that you are working on a feature or a fix is useful. Early communication often times save time consuming mistakes or avoids duplicated effort. We encourage contributors to communicate with us early.

Branches on forks of this project are not very visible to maintainers as much as pull requests (PR). For this reason we used to recommend sending a PR even if it’s not ready and prepend "WIP" in front of its name to let everyone see that you are working on a specific topic. Now, instead of prepending "WIP", we recommend using GitHub "draft pull request" feature instead.

'needs review' label

You can apply that label to a pull request that is complete and ready for review.

Makes triaging easier.

Node package

Test a local asciidoctor-reveal.js version

In order to test the Node package, you first need to build the converter into Javascript and create a tarball of the project.

```
$ bundle exec rake build:js
$ npm pack
```

That last command will produce a file named `asciidoctor-reveal.js-<version>.tgz` in the working directory.

Then, create a test project adjacent to the clone of the *asciidoctor-reveal.js* repository:

```
$ mkdir test-project
$ cd test-project
```

Now, install the dependencies from the tarball:

```
$ npm i --save ../asciidoctor-reveal.js/asciidoctor-reveal.js-<version>.tgz
```

NOTE The relative portion of the last command is where you are installing the local `asciidoctor-reveal.js` version from.

Then proceed as documented in the `README.adoc`.

Binary package compatibility with Asciidoctor.js

Asciidoctor.js is source-to-source compiled into JavaScript from Ruby using Opal. The JavaScript generated requires a specific version of the Opal-runtime for it to work with Node.js. This project is source-to-source compiled into JavaScript from Ruby using Opal too. In order for Asciidoctor.js to be able to call code from this converter, the versions of Opal (both runtime and compiler) must be compatible. Right now we track the exact git revision of Opal used by Asciidoctor.js and make sure that we match. Here is how:

Versions known to work together can be found by looking at the Asciidoctor.js release notes, just replace &lt;tag&gt; with the <code>asciidoctor.js</code> release you are interested in: <a href="https://github.com/asciidoctor/asciidoctor.js/releases/tag/&lt;tag&gt" class="bare">https://github.com/asciidoctor/asciidoctor.js/releases/tag/&lt;tag&gt</a>;. Then that Opal version and git revision (if required) must be specified in <code>asciidoctor-revealjs.gemspec</code>.

Starting with 3.0.0 we aim to retain binary compatibility between Asciidoctor.js and Asciidoctor-reveal.js. This should allow other Asciidoctor extensions to be called along with this converter. Asciidoctor.js is no longer a direct dependency but should be seen as a tool that powers this converter. We need to allow users to have flexibility in the version they choose to run. Asciidoctor.js maintainer told us that he is going to consider binary package incompatibility a major break and so we adjusted our README to tell users to install with a specific version range.

We will track and maintain the README on the major version supported and recommended:

- In the version range to install by default for a given release (and on master)
- In the compatibility matrix

See this issue for background details on that topic.

Asciidoctor.js versioning policy is available here.

Debugging

To debug the JavaScript application, just add `--node-arg=--inspect-brk` to the npx command to run the application. For example:

```
npx --node-arg=--inspect-brk asciidoctor-revealjs -v presentation.adoc
```

Then open the Chrome Dev Tools and click on the Node logo in the top left corner.

RubyGem package

Test a local asciidoctor-revealjs version

Compile the converter:

```
$ bundle exec rake build
```

In a clean directory besides the `asciidoctor-reveal.js` repository, create the following `Gemspec` file:

```
source 'https://rubygems.org'
gem 'asciidoctor-revealjs', :path => '../asciidoctor-reveal.js'
```

Then run:

```
$ bundle --path=.bundle/gems --binstubs=.bundle/.bin
```

Release process

1. Make sure that the highlight plugin code embed in *lib/asciidoctor-revealjs/highlightjs.rb* is up-to-date with the version of reveal.js
2. Do we need to do anything regarding our Opal dependency and Asciidoctor.js? See our section on the topic.
3. Update dependencies and test the package in both languages

   ```
   bundle update
   bundle exec rake build
   bundle exec rake test
   bundle exec rake examples:convert
   npm install
   npm update --no-save
   bundle exec rake build:js
   npm test
   npm run examples
   ```

4. Commit the updated dependencies
5. Update the version in `lib/asciidoctor-revealjs/version.rb` and `package.json`
6. Update the changelog
   - Generate author list with:

     ```
     git log <prev-version-tag>.. --format="%aN" --reverse | perl -e 'my %dedupe;
     while (<STDIN>) { print unless $dedupe{$_}++}' | sort
     ```

7. Prepare release commit
   - Add the "Slim compiled to Ruby" converter to the git tree (otherwise ignored to avoid noise to the repo)

     ```
     bundle exec rake build
     git add -f lib/asciidoctor-revealjs/converter.rb
     ```

   - commit msg: Prepare %version% release
   - release commit (--allow-empty) msg: Release %version%
8. Tag the release commit
   - Annotated Tag msg: Version %version%

9. Push your changes (including the tag)
10. Make a release on github (from changelog and copy from previous releases)
    - Useful vim regex for AsciiDoc to Markdown:

      ```
      :%s/{uri-issue}\(\d\+\)\[#\d\+]/#\1/gc
      :%s/{project-name}/asciidoctor-reveal.js/gc
      :%s/\(.*\)::/### \1/gc
      :%s/{uri-repo}/https:\/\/github.com\/asciidoctor\/asciidoctor-reveal.js/gc
      ```

    - Save as draft
11. Pushing the gem on rubygems.org:

    ```
    $ bundle exec rake build
    $ gem build asciidoctor-revealjs.gemspec
    $ gem push asciidoctor-revealjs-X.Y.Z.gem
    ```

12. Check that the new version is available on rubygems.org
13. Generate the javascript version of the Ruby converter

    ```
    $ bundle exec rake build:js
    ```

14. Publish the node package on npm:

    ```
    $ npm login # only required if not already authenticated
    $ npm publish
    ```

15. Check that the new version is available on npmjs.com
16. Make binaries release
    - Run `npm run package`. Binaries built will be in `dist/`. Upload them to the GitHub release page.
17. Publish previously saved GitHub release draft
18. Update version in `lib/asciidoctor-revealjs/version.rb` and `package.json` (+1 bugfix and append '-dev')
    - Remove the "Slim compiled to Ruby" converter to the git tree (to avoid noise to the repo and `git status` noise)

      ```
      git rm --cached lib/asciidoctor-revealjs/converter.rb
      ```

    - commit msg: Begin development on next release
19. Submit a PR upstream to sync the documentation on asciidoctor.org
    - Modify this page: https://github.com/asciidoctor/asciidoctor.org/edit/master/docs/asciidoctor-

revealjs.adoc

20. Submit a PR downstream to update Asciidoctor reveal.js version inside docker-asciidoctor
    - Modify the `Dockerfile`, `Makefile` and `README.adoc` of: https://github.com/asciidoctor/docker-asciidoctor
21. Submit a PR downstream to update AsciidoctorJ reveal.js version
    - Modify `gradle.properties`, `asciidoctorj-revealjs/gradle.properties` and `asciidoctorj-revealjs/build.gradle` in: https://github.com/asciidoctor/asciidoctorj-reveal.js

Ruby and asciidoctor-doctest tests

Running tests

We recommend tests to be run with a fresh install of all dependencies in a local folder that won't affect your ruby install (a `.bundle/` in this directory):

```
bundle --path=.bundle/gems --binstubs=.bundle/.bin
```

Then you can execute the tests with:

```
bundle exec rake doctest
```

However, if you have all dependencies properly installed this command should run the tests successfully:

```
rake doctest
```

Generating HTML test target

Tests were bootstrapped by generating them from asciidoctor-doctest's test corpus and current asciidoctor-revealjs' slim template engine. This is done using the following command:

```
bundle exec rake doctest:generate FORCE=y
```

Custom tests

Files in the `examples/` directory are used as tests. Resulting slides are kept in `test/doctest/`.