
RISE Documentation

Release 0.0.0

Damian Avila

Sep 05, 2018

Contents

| | | |
|----------|--|-----------|
| 1 | What is RISE? | 3 |
| 2 | Detailed Contents | 5 |
| 2.1 | Installation | 5 |
| 2.1.1 | Disable and Removal | 6 |
| 2.2 | Usage | 6 |
| 2.2.1 | Creating a slideshow | 6 |
| 2.2.2 | Running a slideshow | 7 |
| 2.3 | Resources about RISE | 8 |
| 2.3.1 | github, readthedocs, gitter, youtube | 8 |
| 2.3.2 | Talks | 8 |
| 2.4 | Customizing RISE | 9 |
| 2.4.1 | What to configure | 9 |
| 2.4.2 | How to customize | 13 |
| 2.4.3 | Keyboard shortcuts and Jupyter actions | 18 |
| 2.5 | PDF Export | 19 |
| 2.5.1 | Using nbconvert | 19 |
| 2.5.2 | Using decktape | 20 |
| 2.6 | Changelog | 20 |
| 2.6.1 | Versions | 20 |
| 2.6.2 | Changes | 21 |
| 2.7 | Developer Documentation | 22 |
| 2.7.1 | Development | 22 |
| 2.7.2 | Releases | 23 |
| 3 | Feedback | 27 |

Reveal.js - Jupyter/IPython Slideshow Extension

CHAPTER 1

What is RISE?

As you know... we love **Jupyter** and we like **reveal.js** too.

With RISE, a Jupyter notebook extension, you can instantly turn your jupyter notebook into a live *reveal.js*-based presentation.

2.1 Installation

Note: To install RISE in development mode, see the [Developer Documentation](#).

From the most simple to the most complex one, you have 3 options:

1 - Using conda (recommended):

```
conda install -c damianavila82 rise
```

2 - Using pip (less recommended):

```
pip install RISE
```

and then one more step to install the JS and CSS in the proper places:

```
jupyter-nbextension install rise --py --sys-prefix
```

3 - The old way (are sure sure you want to go this path?):

To install this nbextension, simply run `python setup.py install` from the *RISE* repository (please use the latest tag available or try master if you want).

And then the same two step described in the pip-based installation:

```
jupyter-nbextension install rise --py --sys-prefix
```

and:

```
jupyter-nbextension enable rise --py --sys-prefix
```

Conclusion: If you use conda, life will be easy ;-)

Note: In all the options available the `--sys-prefix` option will install and enable the extension in the current environment, if you want a `--user` based or a `--system` based installation just use those option instead in the above commands.

2.1.1 Disable and Removal

You can disable RISE with:

```
jupyter-nbextension disable rise --py --sys-prefix
```

If you want to remove it from your environment:

```
jupyter-nbextension uninstall rise --py --sys-prefix
```

Alternative, you can also remove it with conda (if you already installed it using conda):

```
conda remove rise
```

2.2 Usage

You can see [in this youtube video](#) a very short session on how to use RISE to create and run a slideshow.

Let us emphasize the key points here.

2.2.1 Creating a slideshow

In the notebook menu, the “View” option contains a “Cell Toolbar” sub-menu that gives you access to the metadata for each cell. If you select the “Slideshow” preset, you will see in the right corner of each cell a little box where you can select the cell type.

You can choose between the following types:

- **slide:** this cell is the beginning of a new slide
- **subslide:** this cell is the beginning of a new subslide; that is to say, a new slide, but that `reveal.js` will display *below* the previous one instead of on the right;
- **fragment:** this is to split the contents of one slide into pieces; a cell marked as a fragment will create a break inside the slide; it will not show up right away, you will need to press Space one more time to see it.
- **skip:** this cell are ignored altogether in *reveal* mode, it will not appear either in the main view, nor in the speaker view.
- **notes:** similarly, this cell is marked to be discarded from the main view, but is meant to appear in the speaker view.

Note that as of RISE version 5.3, the support for speaker view is not working, so the *notes* cells will not show up at all either when in *reveal* mode.

Keyboard shortcuts

Starting with version 5.1.0 you can customize some keyboard shortcuts using the keyboard shortcut editor from the notebook UI.

We have defined 4 main shortcuts by default that you can change according to your needs:

- `Alt-r`, “Enter/Exit Live Reveal Slideshow”
- `Shift-i`, Toggle slide
- `Shift-u`, Toggle subslide
- `Shift-f`, Toggle fragment

2.2.2 Running a slideshow

Once enabled, the RISE Jupyter extension displays a new button (“Enter/Exit Live Reveal Slideshow”) in the toolbar, (also activable with `Alt-r` by default).

This starts the slideshow; you can return to normal notebook edition by either pressing `Alt-r` again, or by clicking on the cross-shaped icon on the upper right corner of each slide.

Navigation

It is *highly recommended* to use mainly **SpaceBar** to go forward, and **Shift-SpaceBar** to go backward (or the visual controller in the slideshow right bottom corner). This will follow the course of the presentation no matter what the detailed structure is (slides, subslides, fragments...).

In contrast, `right` and `left` arrows can have a confusing behaviours with respect to these 3 structural entities. Besides, `up` and `down` arrows are reserved to interact with notebook cells and cannot be used to navigate the slides, instead you can use `pgup` and `pgdown`.

Selection and evaluation

Essentially, when a code cell appears in the presentation, you simply need to press **Shift-Enter** to run it. This will move to the next cell if it is already displayed.

The default behaviour for RISE is to select the first code cell when a new slide or fragment shows up. This way, if your presentation has only markdown cells, you will not be bothered with cells being selected; on the other hand when you do have code cells, you can run the entire slideshow by just using **Space** and **Shift-Enter** as appropriate.

Other notes

- In presentation mode, you can know more about the reveal-specific shortcuts just pressing the help button at the slideshow left bottom corner.
- Darkish themes have css conflict with the notebook css, so it need customization to make them work (not provided by default).
- Markdown Images get left aligned by default. Enclose the image like `<center>![noimg](path/to/image.png)</center>` to center it.

Shift-Enter behaviour (historical note)

Starting version 5.1.0: We have developed a `smart_exec` functionality which essentially it is bound to the Shift-Enter keyboard shortcut and allows you to execute cells and then proceed to the next cell **WHEN** the context permits. It is pretty similar to the native behaviour in the notebook view but it takes into consideration the slideshow view limitations and particularities. You can find a demo notebook at `RISE/examples/showflow.ipynb`, but pretty sure you will find the behavior familiar enough to play with it immediately.

Prior to version 5.1.0: In contrast to the traditional Jupyter notebook, the `Shift-Enter` shortcut does not select the next cell, but will only run the cell (same as `Ctrl-Enter`). This is intentional to not switch slides by running a cell and because some problem arises when you inject new cells on the fly. When you exit the presentation mode, the behavior comes back to normal.

JupyterLab


Please be aware that as of 5.3 RISE is unfortunately not yet compatible with JupyterLab and must be used with the classic notebook.

See <https://github.com/damianavila/RISE/issues/270> for the github issue on this topic.

2.3 Resources about RISE

2.3.1 github, readthedocs, gitter, youtube

The following places are assets in the RISE development landscape:

- Demo notebook (no installation required)
 -
- Source code is on github <https://github.com/damianavila/RISE>
 -
- Documentation is hosted on readthedocs
 -
 - *Historical note:* please note that older location for the RISE documentation here: <https://damianavila.github.io/RISE/> has been phased out and is no longer updated.
- Chat room on gitter
 - 
- Videos on youtube
 - basic usage (4'30'')
https://youtu.be/sXyFa_r1nxA
 - see also below the talk from *SciPy 2014* (21')
<https://www.youtu.be/sZBKruEh0jI>

2.3.2 Talks

My old talk about **RISE** at *SciPy 2014* (click on the image to see it):



2.4 Customizing RISE

2.4.1 What to configure

Here's a list of things that can be customized. *See below for more details* on how to implement those settings.

- *presentation theme*
- *transiton between slides*
- *auto-launch presentation mode*
- *where to start the presentation*
- *automatic selection of cells*
- *slide sizes*
- *decoration (header/footer/background)*
- *vertical scrollbar*
- *using a leap motion controller*
- *native `reveal.js` settings*
- *custom CSS*

- *keyboard shortcuts*

Choosing a theme

You can configure the theme of your presentation (which controls the general look and feel of the presentation) with:

```
{
  ...
  "rise": {"theme": "sky"}
}
```

Choosing a transition

The transition configuration defines what happens in between slides:

```
{
  ...
  "rise": {"transition": "zoom"}
}
```

Automatically launch RISE

You can setup your notebook to start immediately with the slideshow view using the `autolaunch` config option. This typically is very helpful if you plan on publishing slideshows through something like `mybinder.org`:

```
{
  ...
  "rise": {"autolaunch": true}
}
```

Choosing where the slideshow begins

The following configuration changes where the slides begin. By default, RISE will start at the selected slide. To have it start at the first slide instead, use this configuration:

```
{
  ...
  "rise": {"start_slideshow_at": "beginning"}
}
```

Select cells based on the current slide

As you progress into your slideshow, you either move to a new (sub)slide, or show (or hide) a new fragment; whenever any of these events occur, you may wish to have the jupyter selection keep in sync or not; this is the purpose of the auto-select feature.

There are currently two settings that let you change the way auto-select behaves, here are their default values:

```
{
  ...
  "rise": {"auto_select": "code",
           "auto_select_fragment": true}
}
```

`auto_select` can be any of:

- `code` (the first code cell is auto-selected)
- `none` (no auto-selection)
- `first` (the first cell is auto-selected)

`auto_select_fragment` is a boolean that states whether auto-selection should select cells based on the current slide as a whole (when set to `false`) or restrict to the current fragment (when set to `true`, the default).

These settings are experimental and may change in the future; hopefully the current default behaviour is just fine. We might remove `auto_select_fragment` as a setting altogether; we might also turn `auto_select` into a mere boolean, since the current setting `auto_select = "first"` has not proved of any practical value. Regardless, it seems like the most meaningful combinations as of now are either `auto_select = "none"` - in which case the other setting is ignored, or `auto_select = "code"` and `auto_select_fragment = true`, which now is the default.

Change the width and height of slides

To control the width and height of your slides, use the following configuration:

```
{
  ...
  "rise": {"width": "90%",
           "height": "90%"}
}
```

Important notes

- remember that you can always use your browser's shortcuts to zoom in/out (Cmd/Ctrl + and Cmd/Ctrl -), and this way adjust the slide content to your screen/projector size.
- this method is *often preferable* than setting sizes. In particular it is dangerous to set sizes in pixels, as most often you cannot rehearse with the actual projector. We recommend setting relative sizes (in percents) rather than absolute ones (in px or cm).
- in any case you may want to increase the slide height to ensure that cell outputs fit within a single slide; keep in mind that cell contents tend to take more space as you run your code.

Decorating all slides

RISE offers two levels for inserting a static background. You can either

- define `overlay`, in which case you take full control,
- **or** you can define `header`, `footer` and `backimage`.

So if you define `overlay`, the 3 latter options will be ignored.

overlay

It is possible to add the config option `overlay` to build a constant background. It is wrapped in a `<div>`, so it can be text or html. In this case, the user is entirely responsible for styling. For example:

```
{
  ...
  "rise": {
    "overlay": "<div class='myheader'><h2>my company</h2></div><div class='myfooter'>
    ↪<h2>the date</h2></div>"
  }
}
```

header, footer and backimage

As a more limited, but often more convenient alternative, you can define any of the following 3 settings.

In this case, minimal styling is applied (floor and ceiling), but user is still responsible for cosmetic styling:

```
{
  ...
  "rise": {
    "backimage": "mybackimage.png",
    "header": "<h1>Hello</h1>",
    "footer": "<h3>World!</h3>"
  }
}
```

You can see some examples using these options at `RISE/examples/overlay.ipynb` and `RISE/examples/header-footer.ipynb`, or in binder respectively

Enable a right scroll bar

To enable a right scroll bar when your content exceeds the slide vertical height, use the following configuration:

```
{
  ...
  "rise": {"scroll": true}
}
```

Usage with Leap Motion

Reveal.js supports the [Leap Motion](#) controller. To control RISE slides with the Leap, put the [reveal leap plugin options](#) in your config with the following parameters:

```
{
  ...
  "rise": {
    "leap_motion": {
      "naturalSwipe" : true,      # Invert swipe gestures
      "pointerOpacity": 0.5,      # Set pointer opacity to 0.5
      "pointerColor"  : "#d80000" # Red pointer"nat.png"
    }
  }
}
```

(continues on next page)

(continued from previous page)

```
}
}
```

To disable it:

```
{
  ...
  "rise": {
    "leap_motion": "none"
  }
}
```

reveal.js configuration options

reveal.js offers a few configuration of its own, as described in [reveal.js's documentation](#). Out of this category, RISE will pass through the following settings:

- `controls` to enable or disable the lower right navigation arrows
- `progress` to enable or disable the thin progress bar at the bottom of the slideshow
- `slideNumber` that allows you to turn off, or customize, slide numbers. Set to boolean `false` to turn off, see [reveal.js's doc](#) for more details
- as well as `history`.

Adding custom CSS

RISE looks for two css files to apply CSS changes on top of the slideshow view:

- First, it attempts to load `rise.css`, and hence this will be applied to all notebooks in the current directory;
- Second, it attempts to load `the_notebook_name.css` and so this will hence be only applied to `the_notebook_name.ipynb`.

Both files need to be placed alongside with the notebook of interest, i.e. in the same directory. You can see some examples using this customization with `RISE/examples/showflow.ipynb`.

NOTE. The implementation of this feature is rather rough, both css files are blindly included without checking for their existence, which may result in error messages in your browser console, complaining about `No such file or directory`. These messages can be safely ignored. See also <https://github.com/damianavila/RISE/issues/353> about this.

2.4.2 How to customize

RISE can be customized in a lot of ways. As of RISE version 5.3, you can:

1. use `nnextensions_configurator`; this tool offers an interactive way to enable, disable and tweak all notebook extensions - see screenshot below;
2. define settings in JSON files, typically by using python scripts;
3. you can also embed settings in a specific notebook's metadata;
4. and you can also provide your own css file(s), that can supersede styling of the various DOM pieces.

The configurator

You may need to install a separate module:

```
pip3 install jupyter-nbextensions-configurator
```

or

```
conda install -c conda-forge jupyter_nbextensions_configurator
```

You should then see a fourth tab in jupyter's directory views, as depicted below. Settings are stored in JSON format, typically in

```
~/jupyter/nbconfig/notebook.json
```

localhost:9999/tree/git/RISE/examples#nbextensions_configurator

Logout

Files Running Clusters **Nbextensions**

Configurable nbextensions

☐ disable configuration for nbextensions without explicit compatibility (they may break your notebook environment, but can be useful to show for nbextension development)

filter: by description, section, or tags

| | | |
|--|---|---|
| <input type="checkbox"/> (some) LaTeX environments for Jupyter | <input type="checkbox"/> 2to3 Converter | <input type="checkbox"/> AddBefore |
| <input checked="" type="checkbox"/> Autopep8 | <input type="checkbox"/> AutoSaveTime | <input type="checkbox"/> Autoscroll |
| <input type="checkbox"/> Code Font Size | <input type="checkbox"/> Code prettify | <input type="checkbox"/> Codefolding |
| <input type="checkbox"/> Codefolding in Editor | <input type="checkbox"/> CodeMirror mode extensions | <input type="checkbox"/> Collapsible Headings |
| <input type="checkbox"/> Comment/Uncomment Hotkey | <input type="checkbox"/> contrib_nbextensions_help_item | <input type="checkbox"/> datestamper |
| <input type="checkbox"/> dragdrop/main | <input type="checkbox"/> Equation Auto Numbering | <input type="checkbox"/> ExecuteTime |
| <input type="checkbox"/> Exercise | <input type="checkbox"/> Exercise2 | <input type="checkbox"/> Export Embedded HTML |
| <input type="checkbox"/> Freeze | <input type="checkbox"/> Gist-it | <input type="checkbox"/> Help panel |
| <input type="checkbox"/> Hide Header | <input type="checkbox"/> Hide input | <input type="checkbox"/> Hide input all |
| <input type="checkbox"/> Highlight selected word | <input type="checkbox"/> highlighter | <input type="checkbox"/> Hinterland |
| <input type="checkbox"/> Initialization cells | <input checked="" type="checkbox"/> jupyter-js-widgets/extension | <input type="checkbox"/> Keyboard shortcut editor |
| <input type="checkbox"/> Launch QTConsole | <input type="checkbox"/> Limit Output | <input type="checkbox"/> Move selected cells |
| <input type="checkbox"/> Navigation-Hotkeys | <input checked="" type="checkbox"/> Nbextensions dashboard tab | <input checked="" type="checkbox"/> Nbextensions edit menu item |
| <input type="checkbox"/> nbTranslate | <input type="checkbox"/> Notify | <input type="checkbox"/> Printview |
| <input type="checkbox"/> Python Markdown | <input checked="" type="checkbox"/> RISE - Slideshow with reveal.js | <input type="checkbox"/> Rubberband |
| <input type="checkbox"/> Ruler | <input type="checkbox"/> Runtools | <input type="checkbox"/> Scratchpad |
| <input type="checkbox"/> ScrollDown | <input type="checkbox"/> search-replace/main | <input type="checkbox"/> Select CodeMirror Keymap |
| <input type="checkbox"/> SKILL Syntax | <input type="checkbox"/> Skip-Traceback | <input type="checkbox"/> Snippets |
| <input type="checkbox"/> Snippets Menu | <input type="checkbox"/> spellchecker | <input checked="" type="checkbox"/> Split Cells Notebook |
| <input type="checkbox"/> Table of Contents (2) | <input checked="" type="checkbox"/> table_beautifier | <input type="checkbox"/> Toggle all line numbers |
| <input type="checkbox"/> Tree Filter | <input type="checkbox"/> Variable Inspector | <input type="checkbox"/> zenmode |

RISE - Slideshow with reveal.js

Run notebook as a slideshow thanks to reveal.js

section: notebook
require path: `rise/main`
compatibility: `jupyter-4.3`, `jupyter-notebook-5.2`

Reveal.js Jupyter/IPython Slideshow Extension

Parameters reset

☐ autolaunch : autorun slideshow upon opening if set; active only on notebooks that have a 'livereveal' or 'rise' section in their metadata

start_slideshow_at : a string that describes where to start the slideshow this can be either 'beginning' (start on first slide) or 'selected' (start on the slide that contains the currently selected cell)

selected

auto_select : a string that specifies how to select cells when new contents is displayed during the slideshow (typically, new slide or new fragment). It can be either 'none' (no automatic selection), or 'code' (select first code cell in new content).

Using python

As an alternative way, you can tweak your local user's settings with a script rather than from the configurator. For example you can use python like shown in this example below, that leverages the JSON config manager from `traitlets`:

```
#!/usr/bin/env python3
from traitlets.config.manager import BaseJSONConfigManager
from pathlib import Path
path = Path.home() / ".jupyter" / "nbconfig"
cm = BaseJSONConfigManager(config_dir=str(path))
cm.update(
    "rise",
    {
        "theme": "sky",
        "transition": "zoom",
        "start_slideshow_at": "selected",
    }
)
```

Notes:

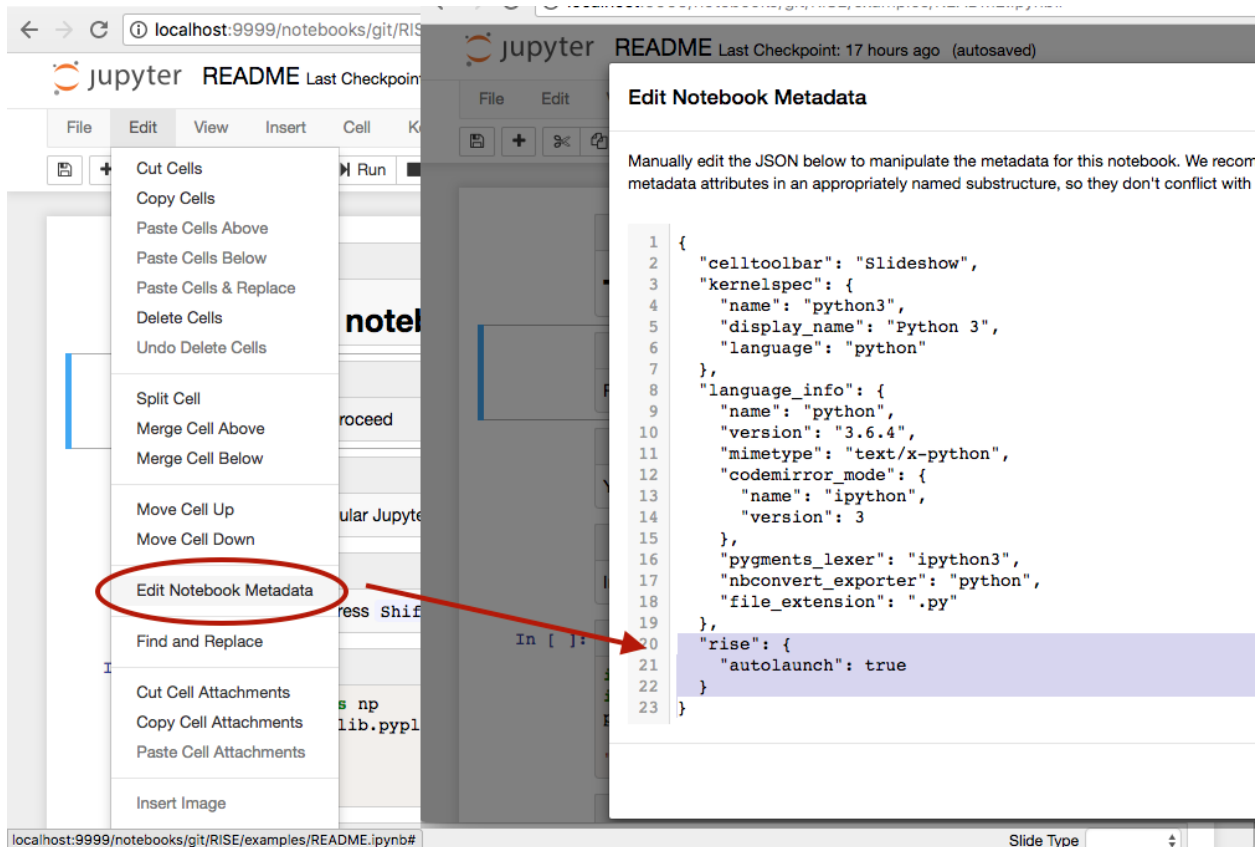
- the `config_dir` parameter should point at where the `nbconfig` is located. This will vary depending on your setup, and specifically on where you “installed” and “enabled” the `nbextension`.
- running the example above would result in the creation (or modification) of a file named `~/.jupyter/rise.json`, which is generally the right place to store user preferences,
- to adjust this path to your own setup, you can use `jupyter --paths`, and specifically the `config` section, to see the path locations that are applicable.
- for more information, see these docs:
 - <http://jupyter.readthedocs.io/en/latest/projects/jupyter-directories.html>
 - http://jupyter-notebook.readthedocs.io/en/latest/frontend_config.html.

Notebook metadata

These settings can also be stored in your notebook metadata, which holds a JSON object, that can be edited through Jupyter’s standard menu (Edit → Edit Notebook Metadata); typically it would look like this:

```
{
  ...
  "rise": {
    "theme": "serif",
    "transition": "zoom",
    ...
  },
  ...
}
```

You can edit notebook metadata as follows



Note on legacy naming

For the remaining of this section, let us forget about custom CSS for a while, and concentrate on the first 3 configuration methods : configurator, JSON files, and notebook metadata.

In all this document we refer to settings stored in a JSON key or filename `rise`. You may also see some notebooks using the `livereveal` key instead, which is an older name for the same project. For backward compatibility, both names are actually taken into account, however you should know that `rise` will take precedence on `livereveal` if the same setting is defined under both names.

You are encouraged to always use the `rise` naming as much as possible.

Order of precedence

The order of precedence between these 3 sources of configuration is as follows:

- a setting defined in the notebook's metadata is always valid; among these, as described above, settings in the `rise` category will override those defined in `livereveal` if both entries apply;
- if still undefined, a setting defined in the configurator will be valid; Finally, the following priorities apply:
- if still undefined, a setting defined in any of the JSON files considered by your jupyter server will be taken into account. Here again, `rise.json` supersedes `livereveal.json` in case of an overlap.

Apart from that, the scope of what is configurable through these various channels (configurator, JSON and metadata) is identical, so it is possible to use the configurator as some sort of an online reference manual, as it describes each and every setting.

Local setting vs hosted infrastructure

At this point you need to be aware that:

- settings changed through the configurator or JSON files - are stored on your own file system, typically in your home directory, and so are only applicable to people using this notebook server; generally it is used for user preferences or such.
 - *a contrario* settings embedded in a specific notebook's metadata will be applicable to all users that get their hands on that notebook, even if they end up in a mybinder instance via github.
-

2.4.3 Keyboard shortcuts and Jupyter actions

Here are the Jupyter actions registered by RISE:

| action name | key | behaviour |
|--------------------------|---------|---|
| ----- | | |
| RISE:slideshow | alt-r | enter/exit RISE Slideshow |
| RISE:smart-exec | | execute cell, move to the next if on same slide |
| RISE:toggle-slide | shift-i | (un)set current cell as a Slide cell |
| RISE:toggle-subslide | shift-u | (un)set current cell as a Sub-slide cell |
| RISE:toggle-fragment | shift-f | (un)set current cell as a Fragment cell |
| RISE:toggle-notes | | (un)set current cell as a Note cell |
| RISE:toggle-skip | | (un)set current cell as a Skip cell |
| RISE:render-all-cells | | render all cells (all cells go to command mode) |
| RISE:edit-all-cells | | edit all cells (all cells go to edit mode) |
| RISE:rise-nbconfigurator | shift-c | open the nbconfigurator pane in another tab |

Some, but not all, come bound to default keyboard shortcuts. There are 2 ways you can change the bindings

Through JSON

Like the other settings described in this section, you can define shortcuts in JSON with e.g.

```
{
  ...
  "rise": {
    "shortcuts": {
      "slideshow": "alt-a",
      "edit-all-cells": "ctrl-e"
    }
  }
}
```

With the above settings, RISE would **not** bind the default Alt-R key to RISE:slideshow, but it would bind Alt-A instead. It would also bind RISE:edit-all-cells to Ctrl-e.

Through custom.js

You can also use these actions in some regular javascript code, typically your ~/.jupyter/custom/custom.js. Here is an example that will attach one of these actions to a custom keyboard shortcut:

```
define(
    ['base/js/namespace'],
    function(Jupyter) {

        let command_shortcuts = Jupyter.keyboard_manager.command_shortcuts;

        // set / unset the 'Slide' tag in slideshow metadata
        command_shortcuts.set_shortcut(
            'alt-a', 'RISE:slideshow');
    })
```

Note that with this approach, you will end up with the `RISE:slideshow` action bound to **both** `Alt-R` and `Alt-A`.

Keyboard shortcut editors

The actions exposed to Jupyter are also present in Jupyter's mainstream keyboard shortcuts editor, that you can use to (un)define your custom shortcuts.

2.5 PDF Export

You can export your RISE presentation to PDF using the following procedures:

2.5.1 Using nbconvert

0 - This step will not be necessary when nbconvert makes a new release (<https://github.com/jupyter/nbconvert/pull/748>), but for now, if you want syntax highlighting in your printed slideshow, you need to follow these (or similar) instructions: <https://github.com/jupyter/notebook/issues/840#issuecomment-365176083>

1 - Generate the slides and serve them using nbconvert:

```
jupyter nbconvert --to slides your_talk.ipynb --post serve
```

It opens up a webpage in the browser at [http://127.0.0.1:8000/your_talk.slides.html/](http://127.0.0.1:8000/your_talk.slides.html#/)

2 - Add `?print-pdf` to the query string as http://127.0.0.1:8000/your_talk.slides.html?print-pdf

Note that you need to remove the `#` at the end. The page will render the slides vertically.

3 - Save to PDF in Chrome using the print option

- Open the in-browser print dialog (Cmd/Ctrl + P).
- Change the Destination setting to Save as PDF.
- Change the Layout to Landscape.
- Change the Margins to None.
- Enable the Background graphics option.
- Click Save.

Note that if you are using JavaScript-based packages like `bokeh` in your slides, you will need to ensure that any cells that define JS code used by other cells are *not* skipped by RISE. For instance, Bokeh plots will only be visible in the PDF output if you include the cell containing `output_notebook()` (or `hv.extension()` if using Bokeh via `HoloViews`), even if the live RISE presentation works fine when skipping those cells. You can use the *Notes* slide type for that cell if you want it to be omitted from the RISE slideshow but included in HTML or PDF output.

2.5.2 Using decktape

1 - Install decktape with:

```
npm install decktape
```

2 - Start the jupyter-notebook server (you don't have to start the RISE presentation, you even don't have to open any notebook at all):

```
jupyter notebook
```

NOTE: Make sure *autoLaunch* option is disabled, otherwise the decktape plugin will exit from the slideshow view before printing the slides. Discussion about this behavior lives at <https://github.com/astefanutti/decktape/issues/110>.

3 - Run decktape with:

```
`npm bin`/decktape rise <Jupyter-Notebook-URL> <Output-File>
```

More concretely, it looks something like the following:

```
`npm bin`/decktape rise http://localhost:8888/notebooks/your/notebook.ipynb?  
→token=YourIndividualJupyterNotebookSessionToken /path/to/outputfile.pdf
```

Note that the jupyter-notebook session token is needed. The token is shown to you when you start the jupyter-notebook server from commandline.

You can run into some problems using this approach:

1 - If you run decktape.js with wrong token first, or some other things first, it could fail. Restarting the jupyter-notebook server helped.

2 - If you have changed the default presentation size/width/height using the notebook metadata, you might have to adapt the call to include the `-s <width>x<height>` parameter:

```
`npm bin`/decktape rise -s 1500x900 https://localhost:8888/...
```

3 - If you experience issues when rendering svg files, please post your fix at [astefanutti/decktape#90](#)

4 - Math rendering problems: just try to rerender (issue posted at [astefanutti/decktape#91](#))

5 - Fragments don't show up at all. The current decktape rise plugin puts `fragments: false`, see <https://github.com/astefanutti/decktape/blob/master/plugins/rise.js#L40> which should render everything together but it is not working. When changing the above line to `fragments: true`, every fragment is rendered as a single slide which is a very efficient work around for the moment as you can simply delete the unwanted slides afterwards.

2.6 Changelog

2.6.1 Versions

1. **RISE** master branch will be following the **Jupyter** codebase.
2. There is also “released” tagged [branches](<https://github.com/damianavila/RISE/releases>) compatible with previous IPython versions:
 - 1.x tag compatible with **IPython** 1.x series
 - 2.x tag compatible with **IPython** 2.x series

- 3.x tag compatible with **IPython** 3.x series
- 3.x.1 tag also compatible with *notebook* 4.x series, but using old installation mechanism
- 4.0.0b1 tag compatible with the *notebook* 4.2 and above, beta release, please test and report any issues
- 5.0.0 tag compatible with *notebook* $\geq 5.0.0$
- 5.1.0 tag compatible with *notebook* $\geq 5.0.0$
- 5.2.0 tag compatible with *notebook* $\geq 5.0.0$
- 5.3.0 tag compatible with *notebook* $\geq 5.5.0$
- 5.4.1 tag compatible with *notebook* $\geq 5.5.0$

3. With **Jupyter** landing we will provide a conda and pip-installable packages too

NOTE: We will only maintain the latest released version.

2.6.2 Changes

- 5.4.1 * Support chalkboard functionality (<https://github.com/damianavila/RISE/pull/355>) * Support speaker notes (<https://github.com/damianavila/RISE/issues/174>) * Use a version number that npm can understand (<https://github.com/damianavila/RISE/pull/410>) * Enhancement in setup.py and reduction of hard-written versions (<https://github.com/damianavila/RISE/pull/399>) * Include LICENSE.md file in wheels (<https://github.com/damianavila/RISE/pull/394>) * Fix python_requires (<https://github.com/damianavila/RISE/pull/390>) * Remove conda recipe from the repo (<https://github.com/damianavila/RISE/issues/405>) * Make the configurator compatible with notebook 5.x versions (<https://github.com/damianavila/RISE/pull/414>) * Docs fixes in exportation section (<https://github.com/damianavila/RISE/pull/415>) * Make RISE compatible with python 3.7 (<https://github.com/damianavila/RISE/issues/406>) * Update changelog (<https://github.com/damianavila/RISE/pull/417>) * Bump 5.4.0 version (<https://github.com/damianavila/RISE/pull/418>)
- 5.4.0 packages were removed from PyPI because they were broken.
- 5.3.0
 - Auto enable nbextension when installing with pip (<https://github.com/damianavila/RISE/pull/342>)
 - Making rise compliant with nbextensions_configurator (<https://github.com/damianavila/RISE/pull/344>)
 - Documentation general review, fixes and improvements (<https://github.com/damianavila/RISE/pull/347>)
 - Mixup between *note* and *notes* (<https://github.com/damianavila/RISE/pull/372>)
 - Keep ? from popping up keyboard shortcuts (<https://github.com/damianavila/RISE/pull/373>)
 - Create shortcut to go to the configurator (<https://github.com/damianavila/RISE/pull/376>)
 - General review of *setup.py* (<https://github.com/damianavila/RISE/pull/387>)
- 5.2.0
 - Source code cleanup and normalization (<https://github.com/damianavila/RISE/pull/311>)
 - Add some docs updates (<https://github.com/damianavila/RISE/pull/312>)
 - Add sidebar for all doc pages (<https://github.com/damianavila/RISE/pull/314>)
 - Improve customization reference docs (<https://github.com/damianavila/RISE/pull/318>)
 - Set new defaults for `auto_select` and `start_slideshow_at` options (<https://github.com/damianavila/RISE/pull/323>)
 - Refactor actions and fix wide toolbar button (<https://github.com/damianavila/RISE/pull/324>)

- Update docs deployment instructions (<https://github.com/damianavila/RISE/pull/325>) and (<https://github.com/damianavila/RISE/pull/326>)
- Make the output observer aware of the scrolling needs (<https://github.com/damianavila/RISE/pull/327>)
- Add basic usage gif into the docs (<https://github.com/damianavila/RISE/pull/328>)
- Fix list not correctly displayed in docs (<https://github.com/damianavila/RISE/pull/338>)
- Add disable and removal section, add note about browser zoom in/out, add PDF export section and add a real changelog for 5.1.1 (<https://github.com/damianavila/RISE/pull/339>)

Previous lazy changelogs:

- 5.1.0: <https://github.com/damianavila/RISE/milestone/5?closed=1>
- 5.0.0: <https://github.com/damianavila/RISE/milestone/4?closed=1>
- 4.x series: <https://github.com/damianavila/RISE/milestone/1?closed=1>

2.7 Developer Documentation

Documentation to develop with RISE. See the sections below for more information.

2.7.1 Development

You can install RISE in development mode in this way:

Requirements

Use your usual package manager to install the required build tools. Essentially you will need:

- git,
- npm and nodejs,
- and of course jupyter;
- sphinx comes in handy to produce the documentation.

Clone the git repo

```
git clone https://github.com/damianavila/RISE.git
cd RISE
```

Prepare a development tree

Step 1. Install the JS dependencies:

```
npm install
```

Step 2. Copy reveal into the static folder and reset reveal.js styling:

```
npm run build
```

To remove `reveal.js` from the static folder you can use `npm run clean-reveal`.

Step 3. Build the CSS assets:

```
npm run build-css
```

Install RISE in developer mode

Second, let's install RISE in a editable form:

```
pip install -e .
jupyter-nbextension install rise --py --sys-prefix --symlink
jupyter-nbextension enable rise --py --sys-prefix
```

Notes:

- the `--symlink` argument is meant to allow you to modify the JavaScript code in-place.
- This feature however is probably not available in Win.
- If you cannot use this *symlink* trick, you will need to “re-install” the nbextension to actually see any changes you made.
- Also please make sure to properly and thoroughly reload your page in the browser; using *Shift* when reloading is generally a good idea.
- Finally, note that as of version 5.3.0, explicitly enabling the extension should no longer be required.

Convenience

If you change the `less` source often, it can be convenient to enable per-save automatic building of CSS, and for that you can use:

```
npm run watch-less
```

which will update the `css` code from `less` each time a change happens on the disk. Kill with Control-C when you are done.

2.7.2 Releases

Instructions and notes for preparing and publishing a release.

Pre-Release check

Step 1. Clean your local repo copy:

```
git clean -fdx
```

Step 2. Build the JS and CSS:

```
npm install
npm run build
```

Step 3. Check for updated version numbers in

- `package.json`

- `conda.recipe/meta.yaml`

Release

Step 4. Tag the repo with:

```
git tag -a release_tag -m "Release msg"
git push origin release_tag
```

Step 5. Build sdist and wheels packages:

```
python setup.py sdist
python setup.py bdist_wheel
```

Step 6. Build the conda packages

For linux and OSX packages:

```
RISE_RELEASE=1 conda build conda.recipe --python=3.6
RISE_RELEASE=1 conda build conda.recipe --python=3.5
RISE_RELEASE=1 conda build conda.recipe --python=2.7
```

and:

```
conda convert /path/to/conda-bld/linux-64/rise-<version_number>-py36_0.tar.bz2 -p_
↪ linux-32 -p linux-64 -p osx-64 -o conda_dist
conda convert /path/to/conda-bld/linux-64/rise-<version_number>-py35_0.tar.bz2 -p_
↪ linux-32 -p linux-64 -p osx-64 -o conda_dist
conda convert /path/to/conda-bld/linux-64/rise-<version_number>-py27_0.tar.bz2 -p_
↪ linux-32 -p linux-64 -p osx-64 -o conda_dist
```

For Windows packages, you need to build in a Win VM (shared folders will make you things easier):

```
set RISE_RELEASE=1
conda build conda.recipe --python=3.6
conda build conda.recipe --python=3.5
conda build conda.recipe --python=2.7
```

If the build hangs, there is probably a permission error, try to run again with `--croot %TEMP%`

then, convert them in the same Win VM:

```
conda convert C:\path\to\conda-bld\win-64\rise-<version_number>-py36_0.tar.bz2 -p win-
↪ 64 -p win-32 -o conda_dist
conda convert C:\path\to\conda-bld\win-64\rise-<version_number>-py35_0.tar.bz2 -p win-
↪ 64 -p win-32 -o conda_dist
conda convert C:\path\to\conda-bld\win-64\rise-<version_number>-py27_0.tar.bz2 -p win-
↪ 64 -p win-32 -o conda_dist
```

Note:

- You can increment the build number with the `RISE_BUILD_NUMBER` environment variable.

Step 7. Upload *sdist* and *wheels* to PyPI:

```
twine upload dist/*
```

Step 8. Upload conda packages to anaconda.org/damianavila82 (5 platforms x 3 pythons):

```
anaconda upload -u damianavila82 conda_dist/linux-32/*
anaconda upload -u damianavila82 conda_dist/linux-64/*
anaconda upload -u damianavila82 conda_dist/osx-64/*
anaconda upload -u damianavila82 conda_dist/win-32/*
anaconda upload -u damianavila82 conda_dist/win-64/*
```


CHAPTER 3

Feedback

If you have any feedback, or find any bugs, please [open an issue](#).