Code exploration tool

A lot of the code I write for blog posts gets tested in one or more browsers through the console. It's tedious and errors are hard to debug (not that it's the console's job but...) so I'd like to use something other than DevTools to do exploratory coding in Javascript, similar but not quite the same as, doing Literate Programming

Jupyter, JupyterLab and JupyterHub

I first heard of iPython, now Jupyter, when I was looking at doing work in Data visualization. Over time there have been additional language kernels including Javascript and Go.

Working with Javascript in such an environment is intriguing as it allows you to work in a Node environment with immediate feedback, as if you were working in a web-based REPL with the possibility of adding things like JSDOM or an alternative to simulate a browser environment.

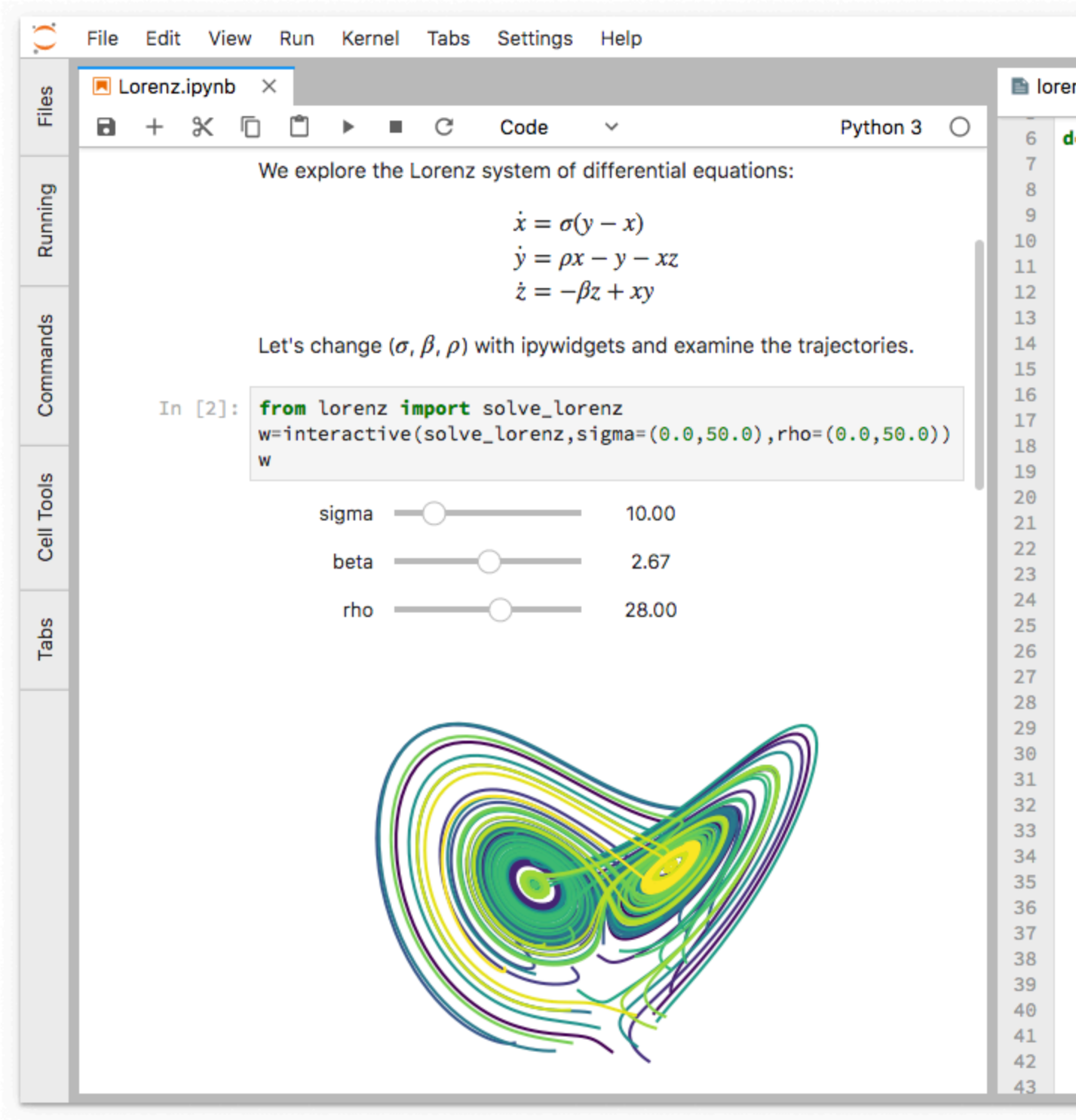

Figure 1: PythonLab, the next generrationn Jupyter notebook

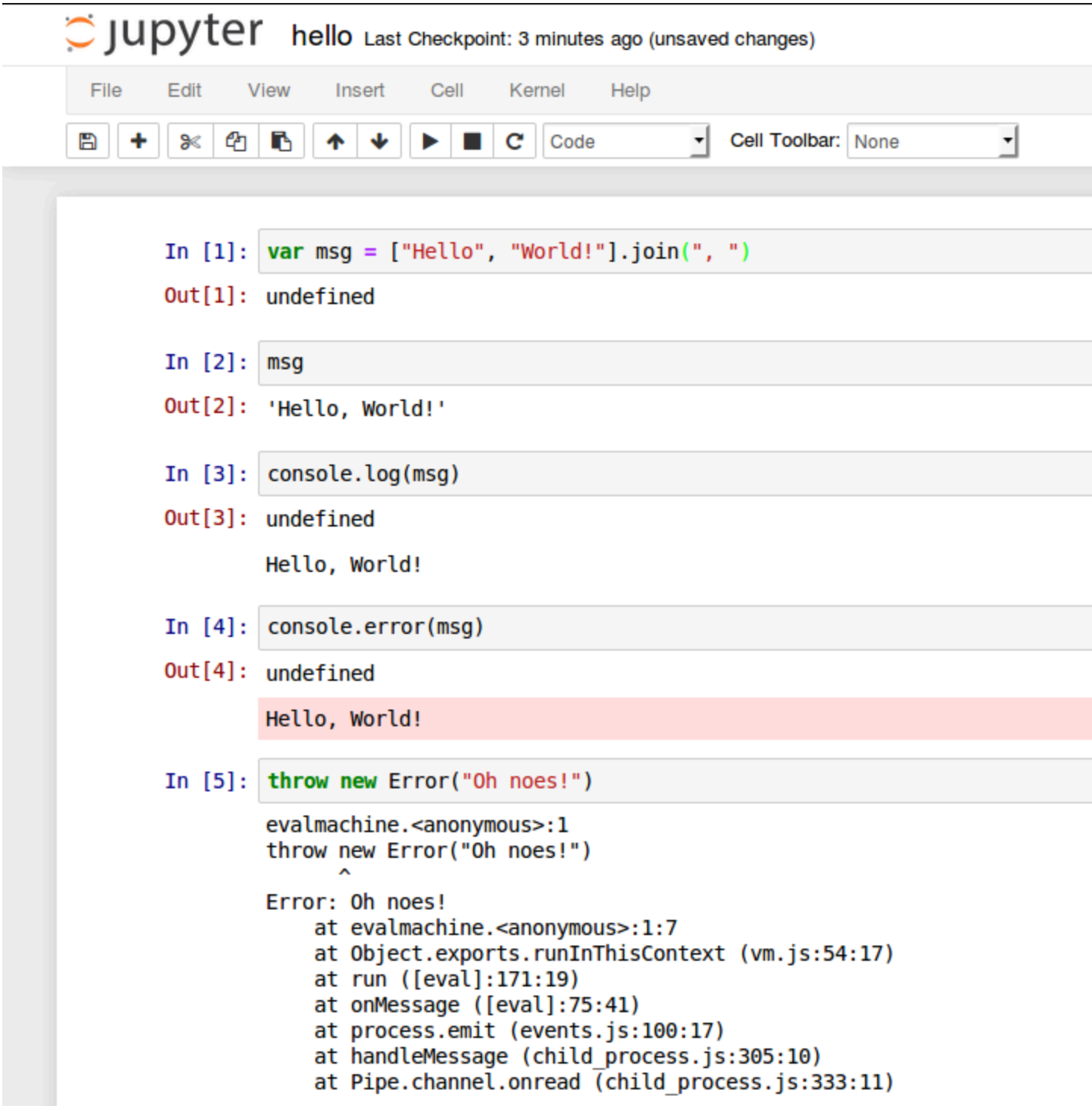

Figure 2: Jupyter Notebook working in Javascript

So what can we use Jupyter and its ecosystem for?

- To work on code problems (our own or someone elses') in a fully interactive environment
- To share code results either by sharing notebooks or publishing to a collaborative environment
- Integrate code with other elements such as external images, visualizations generated from code, PDF, Markdown and other formats

- Allow for multiple simultaneous edits to a notebook, similar to what G Suite files (docs, sheets, slides) do

There are more areas where tools like Jupyter may be useful but these are the oness that come to mind first.

Getting Started: Jupyter and Notebook

These instructions are the basis for what comes next and assume a level of familiarity with command line tools and python. It also asumes that Python 3 has been installed on your system as `python3` with Pip installed by default as `pip3`.

First, install pip3; if it's already installed it'll check if it's the latest version and upgrade it if it's not.

```
pip3 install -U pip
```

Then install the Jupyter Notebook using the same technique as we did for Pip. Install it if it's not available and upgrade it if installed and not the latest version.

```
pip3 install -U jupyter
```

To test that it works just run the notebook using the following command:

```
jupyter notebook
```

Getting Started: Javascript

Now that we have the basic Jupyter installed we can install Javascript support. Run the following command

```
npm install -g ijavascript
```

This will make Javascript available as a language for new notebooks.

If you use `nvm` make sure that you're using the version of Node where you

installed the `ijavascript` package, otherwise you will get errors when starting a new Javascript notebook.

Testing the installation: Jupyter Notebook

IF it works it should open your default web browser and provide a list of files in your current directory.

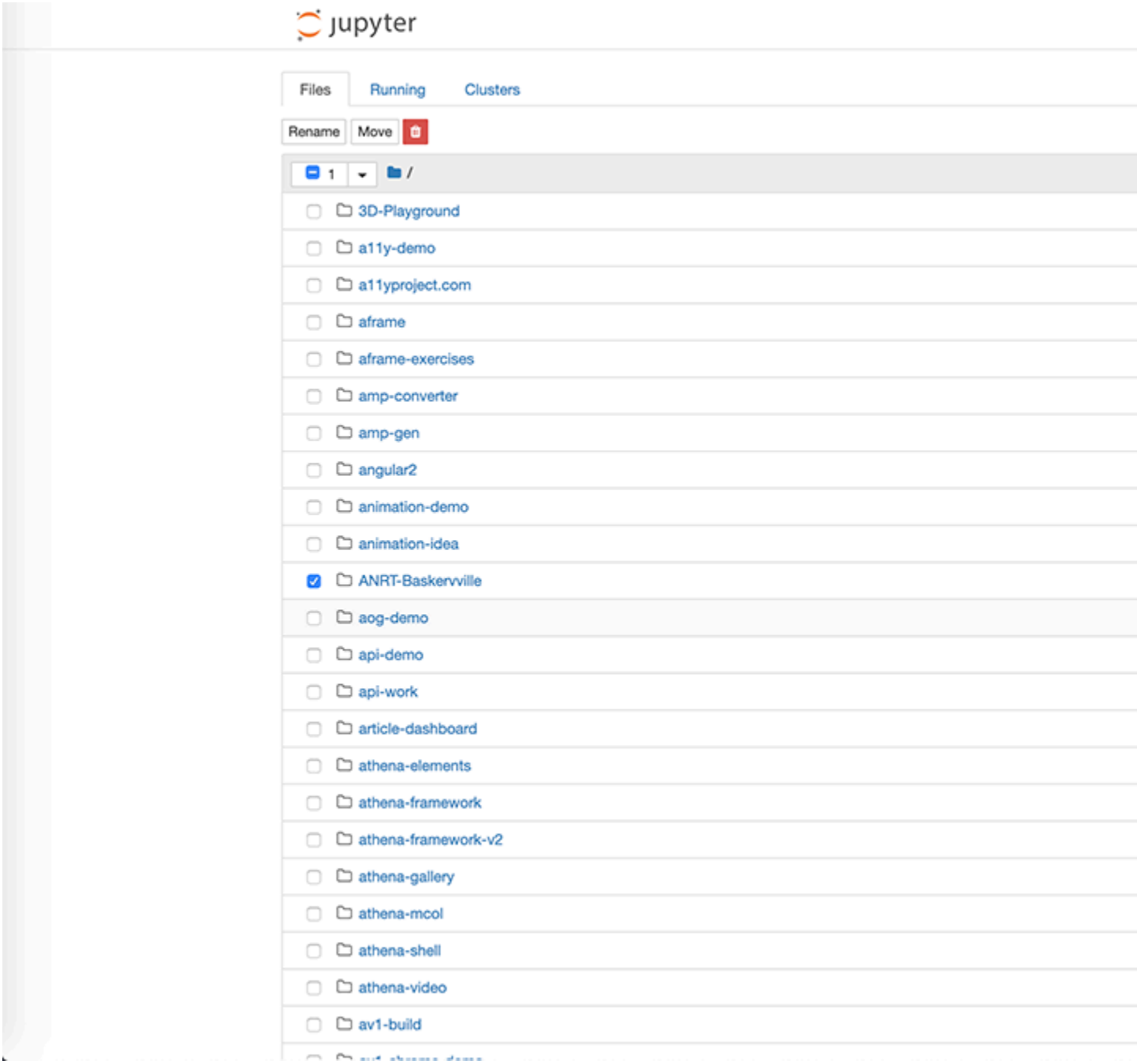

Figure 3: Jupyter Notebook Starting Screen

You can click the new button (on the right side of the screen) and it'll give you a list of available languages; this particular configuration Python2, Python3, Go and

Javascript/Node available.

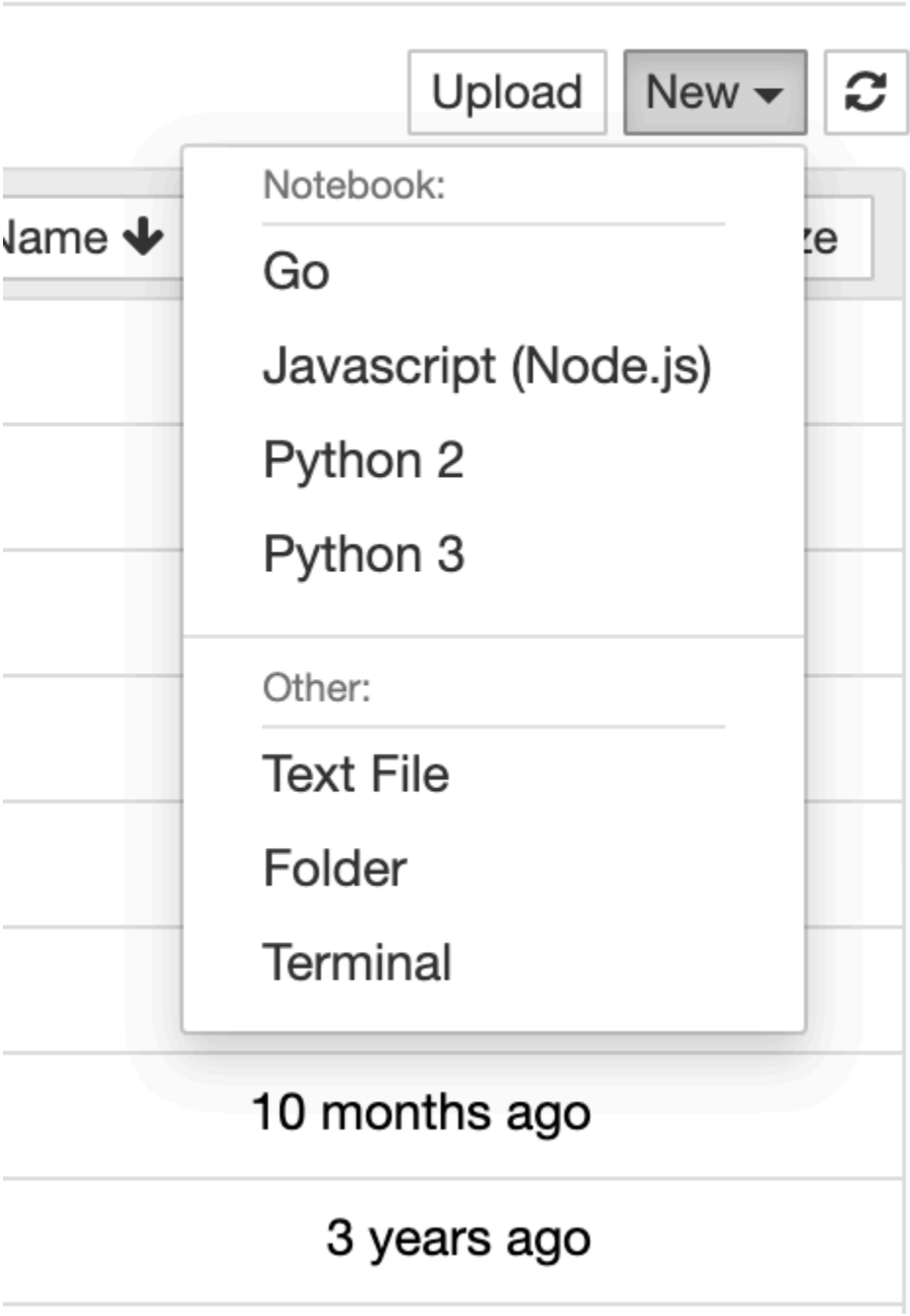

Figure 4: List of available languages available when creating a new Jupyter notebook

Getting Started: JupyterLab

Jupyter notebooks do a lot of awesome things but the UI is not the best. There is another way to view Jupyter notebooks: `JupyterLab`.

As shown in the image below, the interface for JupyterLab is cleaner and easier

to use than the notebook interface discussed earlier.

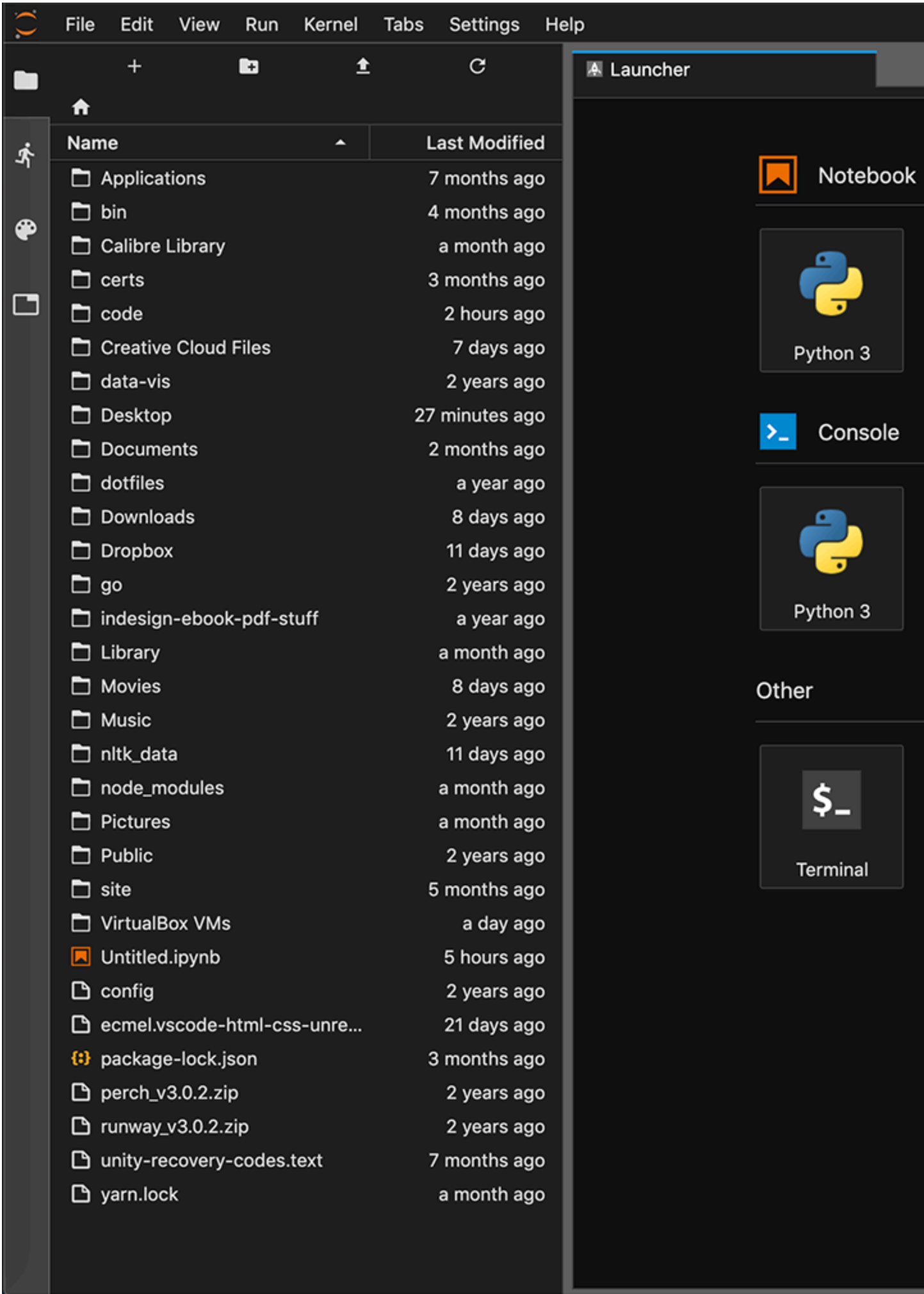
File Edit View Run Kernel Tabs Settings Help
Launcher
Name
Last Modified
Applications 7 months ago
bin 4 months ago
Calibre Library a month ago
certs 3 months ago
code 2 hours ago
Creative Cloud Files 7 days ago
data-vis 2 years ago
Desktop 27 minutes ago
Documents 2 months ago
dotfiles a year ago
Downloads 8 days ago
Dropbox 11 days ago
go 2 years ago
indesign-ebook-pdf-stuff a year ago
Library a month ago
Movies 8 days ago
Music 2 years ago
nltk_data 11 days ago
node_modules a month ago
Pictures a month ago
Public 2 years ago
site 5 months ago
VirtualBox VMs a day ago
Untitled.ipynb 5 hours ago
config 2 years ago
ecmel.vscode-html-css-unre... 21 days ago
package-lock.json 3 months ago
perch_v3.0.2.zip 2 years ago
runway_v3.0.2.zip 2 years ago
unity-recovery-codes.text 7 months ago
yarn.lock a month ago
Notebook
Python 3
Console
Python 3
Other
Terminal

Figure 5: JupyterLab launcher in dark mode showing available languages

To install JupyterLab run the following command

```
pip3 install -U jupyterlab
```

and then test it by running

```
jupyter lab
```

If it works, it will automatically open a browser and show the luncher discussed earlier.

Now what?

We've installed all the components for JupyterLab and are ready to go, what do we do with it?

I normally test code for my blog posts in Chrome's console. This gives me a browser environment without having to install anything.

But it's tedious and debugging is harder than working with code embedded on the hosting page. If we're working in Node we should be able to use a Jupyter Javascript notebook as is but, if we're working on browser code then we might have a problem as Node doesn't have the same globals as browsers do and may not support the same exact set of features.

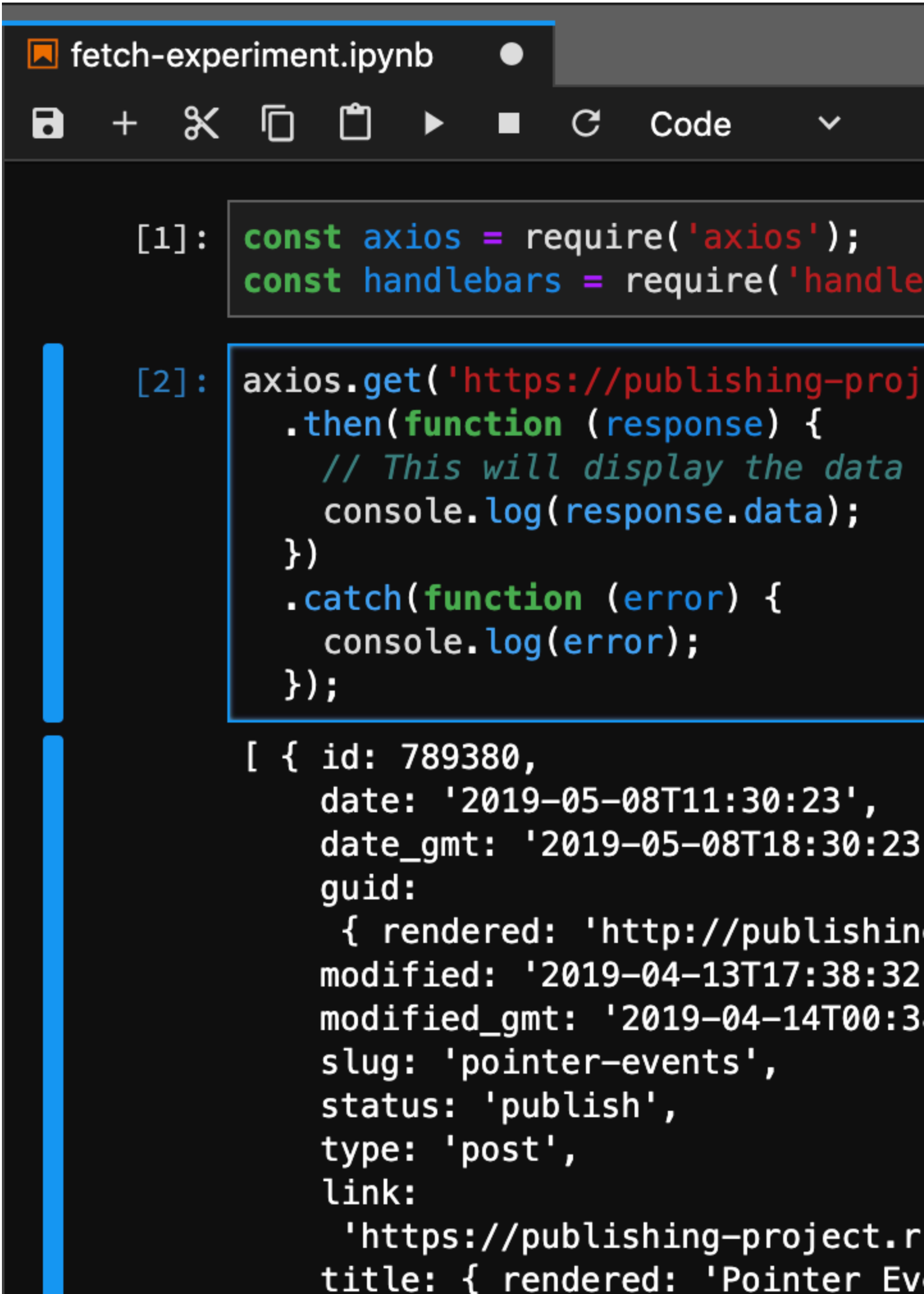
fetch-experiment.ipynb
Code
[1]:
const axios = require('axios');
const handlebars = require('handle
[2]:
axios.get('https://publishing-proj
.then(function (response) {
// This will display the data
console.log(response.data);
})
.catch(function (error) {
console.log(error);
});
[{ id: 789380,
date: '2019-05-08T11:30:23',
date_gmt: '2019-05-08T18:30:23
guid:
{ rendered: 'http://publishin
modified: '2019-04-13T17:38:32
modified_gmt: '2019-04-14T00:3
slug: 'pointer-events',
status: 'publish',
type: 'post',
link:
'https://publishing-project.r
title: { rendered: 'Pointer Ev

Figure 6: Javascript code running inside Jupyter Lab

We can import code from Node packages.

```
npm init --yes
```

```
npm i axios handlebars
```

In this case we import `axios` to handle fetching the data and `handlebars` to display the resulting data.

```
const axios = require('axios');
const handlebars = require('handlebars');
```

In our first iteration we use Axios to `get` JSON daata from our API and, for now, loog the data to the console.

```
axios.get('https://publishing-project.rivendellweb.net/wp-json/wp/v2/posts
  .then(function (response) {
    // This will display the data
    console.log(response.data);
  })
  .catch(function (error) {
    console.log(error);
  });
```

The first problem I hit with this demo is that I need the following HTML to work so the example works properly and I can't get it to work so, until I do that means that we're limited to working with Node APIs amd modules.

```
$$.html(`<div id="myContent">hello world</div>
```

```
<template id="post-list-template">

  {{#each posts}}
  <div class="post">
    <h1>{{title.rendered}}</h1>
    <div>
      {{{content.rendered}}}
    </div>
  </div>
  {{/each}}

</template>`)

console.log($$.html)
```

```
<div id="myContent"></div>

<template id="post-list-template">

  {{#each posts}}
  <div class="post">
    <h1>{{title.rendered}}</h1>
    <div>
      {{{content.rendered}}}
    </div>
  </div>
  {{/each}}

</template>
```

The alternative is the jsdom package to

Observable (FKA: d3.express)

Links

- Jupyter
    - Jupyter
    - JupyterLab is Ready for Users
    - Setting Up Jupyter Notebook with JavaScript, Python2 and Python3 Support
    - The Littlest JupyterHub
        - Installing on Google Cloud
        - Installing on Digital Ocean
        - Installing on Amazon Web Services
    - JupyterLab
        - JupyterLab extensions on NPM
        - JupyterLab extensions in Github
- Emulating a browser
    - JSDOM
- Observable
    - A better way to code
    - Observable Site
    - Downloading and embedding notebooks