iJavascript as a teaching and sharing tool

iJavascript is a Node/Javascipt kernel for the Jupyter Project, itself an extension and evolution of iPython to tackle languages other than Python for interactive computing.

The formal iJavascript definition:

> IJavascript is a Javascript kernel for the Jupyter notebook. The Jupyter notebook combines the creation of rich-text documents (including equations, graphs and videos) with the execution of code in a number of programming languages. The execution of code is carried out by means of a kernel that implements the Jupyter messaging protocol.
>
> The IJavascript kernel executes Javascript code inside a Node.js session. And thus, it behaves as the Node.js REPL does, providing access to the Node.js library and to any installed npm modules.

configuring and installing iJavascript and prerequisites

To install IJavascript in Ubuntu 16.04 LTS, run:

```
sudo apt-get install nodejs-legacy npm ipython ipython-notebook
sudo npm install -g ijavascript
```

In Windows, Anaconda offers a convenient distribution to install Python and many other packages, such as Jupyter and IJavascript.

In macOS, Homebrew and pip can be used to install IJavascript and its prerequisites. This is a three step process if you are starting from scratch:

First we instal Homebrew and use it to install `pkg-config`, node and `zeromq`.

The node version installed by Homebrew is not the same as the one you get when you manually download and install from the Node website. I've chose then Homebrew way because it makes installation the same regardless of OS version.

If you've already installed Node, either manually or through NVM then you can skip installing Node but must install the other packages.

```
ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/r
brew install pkg-config node zeromq
```

The next step uses Python and Pip to install the Python bindings for ZeroMQ and the Jupyter core engine.

```
easy_install pip
pip install --upgrade pyzmq jupyter
```

The final step is to install iJavascript globally using NPM.

```
npm install -g ijavascript
```

IJavascript provides 5 executables: ijsinstall, ijsnotebook, ijsconsole, ijskernel and ijs. Their purpose and basic use is described below.

ijsinstall (IJavascript kernel spec installer): ijsinstall registers the IJavascript kernel with Jupyter, so that other tools (e.g. the Jupyter notebook) can invoke it.

ijsnotebook: (IJavascript notebook): After running ijsinstall, Jupyter notebook users can invoke the Jupyter notebook as usual. ijsnotebook is provided for convenience to users of the IPython notebook prior to version 3. ijsnotebook is a wrapper around ipython notebook.

ijsconsole (IJavascript console): ijsconsole is provided for convenience to users as a wrapper around ipython console. The following command flags are recognised:

ijskernel (IJavascript kernel): ijskernel is the executable invoked by Jupyter tools (e.g. the notebook) and that appears in the kernel spec that ijsintall creates for IJavascript. You won't need this command, unless you want to create a custom

kernel spec.

ijs (Deprecated CLI) ijs is provided for backwards-compatibility. It will be removed in the next major-version update. Please, use ijsinstall or ijsnotebook instead.

What can you do with iJavascript

Figure 1: Example iJavascript notebook

iJavascript provides interactive notebooks written in Node-based Javascript. Because you're using Node you also get access to the Node package ecosystem. Because you're using Jupyter as the base for the notebooks these can be shared and published like your python content.

Some additional things you can do

Authoring Async Code: Many node.js APIs are async, and you can write async code in notebook cells too! We need to be careful when running async code because we are not guaranteed an execution order and the results may be unexpected

Figure 2: Example notebook async JS code

Working with JSON: JSON is everywhere, and you can use a %%json cell to easily declare JSON data. The notebook provides auto-complete functionality which extends to this JSON data.

Figure 3: Using JSON In iJavascript notebook

HTTP requests: You can use the notebook interface to experiment with HTTP APIs

using the HTTP client provided by request node module.

Figure 4: Example notebook creating HTTP request

Client-script: You can easily add HTML markup to your notebook using an `%%html` cell and client-script using a `%%script` block to use a variety of javascript libraries such as d3.js.

Figure 5: iJavascript notebook using HTML magic token and svg

Example Code Snipets