NODE.JS AND R IN PYTHON

ANAND S, PYCONF HYDERABAD, 8 DEC 2019

@sanand0
OUR ANALYTICS TEAM LIKES USING R

BUT PRODUCTIONISING R IS HARD
SSO, MICRO-SERVICES, DISTRIBUTED COMPUTING
Our UI team likes using HTML/CSS/JS

Customize and download

Customize Bootstrap's components, Less variables, and jQuery plugins to get your very own version. Requires IE9+ or latest Safari, Chrome, or Firefox.

Less variables

Customize Less variables to define colors, sizes and more inside your custom CSS stylesheets.

Colors

Gray and brand colors for use across Bootstrap.

- @gray-base
  - #000

- @gray
  - lighten(@gray-base, 33.5%)

- @gray-lighter
  - lighten(@gray-base, 93.5%)

- @gray-dark
  - lighten(@gray-base, 20%)

- @gray-light
  - lighten(@gray-base, 46.7%)

- @gray-darker
  - lighten(@gray-base, 13.5%)

BUT PYTHON CAN’T HANDLE HTML/CSS/JS AS WELL AS NODE.JS
SHOULD people SWITCH LANGUAGES?

I don’t think so
WE CAN RUN R IN PYTHON

To run R in Gramex, install rpy2 first:

```
conda install rpy2  # Do not use pip. Gramex assumes you use conda
```

This installs a new R to the Anaconda PATH, ignoring the system R.

Caution: You'll have 2 R's in your system -- the Anaconda R and the system R. Running R from the command line will run whichever is first in your PATH. Installing a package in one does not install a package in the other.

Here is an example of a prime number calculation in R:

- **prime**: calculate prime numbers up to 100
- **prime?n=1000**: calculate prime numbers up to 1,000

See the Python source and the R script.

**R commands**

Call `gramex.ml.r('R expression')` to run an R command and return its result.
```
129   def r(code=None, path=None, rel=True, conda=True, convert=True,
130       repo='https://cran.microsoft.com/', **kwargs):
131       ...
132
133       Runs the R script and returns the result.
134
135       :arg str code: R code to execute.
136       :arg str path: R script path. Cannot be used if code is specified
137       :arg bool rel: True treats path as relative to the caller function's file
138       :arg bool conda: True overrides R_HOME to use the Conda R
139       :arg bool convert: True converts R objects to Pandas and vice versa
140       :arg str repo: CRAN repo URL
141
142       All other keyword arguments as passed as parameters
143       ...
144
145       # Use Conda R if possible
146       if conda:
147           r_home = _conda_r_home()
148           if r_home:
149               os.environ['R_HOME'] = r_home
```
This has the benefits of both R & Python

R for Statistical Analysis
R for exploratory analysis
R for reproducing scientific papers

Python for Deep Learning
Python for deploying analytics
Python for solving problems roughly

But the biggest benefit is...

If someone has created an R script, integrate it.

Don’t argue with them about changing languages.

Other libraries
rpy2, PypeR, pyRserve
WE RUN JavaScript (Node.js) IN Python

Run JavaScript

Gramex can run JavaScript code via node.js using gramepx.pnode.node.

- Run JavaScript
- JavaScript conversion

Run JavaScript

Gramex FunctionHandlers can run JavaScript code in node.js. Here is a simple example:

```python
from gramepx.pnode import node
from tornado.gen import coroutine, Return

coroutine
def total(handler):
    result = yield node.js('return Math.abs(x + y)', x=5, y=-10)  # Run code in JS
    raise Return(result)
```

This returns the result:

```json
{"error":null,"result":5}
```

Source

JavaScript conversion

Here is how node.js() works:

- Run any JS code. You can pass any JavaScript code to node.js(). Whatever the code returns is the result. For
def js(self, code=None, path=None, **kwargs):
    if self.conn is None:
        try:
            self.conn = yield websocket_connect(self.url, connect_timeout=self.timeout)
        except OSError as exc:
            import errno
            if exc.errno != errno.ECONNREFUSED:
                raise
        # TODO: node_path
        self.proc = yield daemon(
            [which('node'), self._path, '--port=%s' % self.port],
            first_line=re.compile(r'pynode: 1\d+.\d+ port: %s' % self.port),
            cwd=self.cwd,
        )
        self.conn = yield websocket_connect(self.url, connect_timeout=self.timeout)

        # code= takes preference over path=
        if code is not None:
This makes UI dynamic, and leverage existing JS code

Custom Bootstrap

`ui/bootsraptheme.css` is a customized version of Bootstrap with additional features.

You can customize Bootstrap by modifying `variable` from the URL. For example: `ui/bootstraptheme.css?primary=maroon` creates

See the list of Bootstrap variables you can change. Here are some examples:

- `?body-bg=yellow`
- `?link-color=black`
- `?paragraph-margin-bottom=5rem`
- `Reset theme`

In addition, any variable beginning with `color` is added as a theme color. For example, initially these buttons have no color:

```
.btn-color1 .btn-color-abc
```

But click on this link: `?color1-purple&color-abc=teal` to add a new `color1` and `color-abc` to the theme. All color methods like

You can change some common variables from the Change Theme button on at the top of this page to colors, fonts and style. They

- `?enable-rounded=false` (default: true)
- `?enable-shadows=true` (default: false)
- `?enable-gradients=true` (default: false)
- `?enable-transitions=false` (default: true)
- `?enable-responsive-font-sizes=true` (default: false)
- `?enable-print-styles=false` (default: true)

There are also additional components and utilities in the library. See the sidebar for a full list.
CaptureHandler takes screenshots

CaptureHandler takes screenshots of pages using either Chrome or PhantomJS.

- Chrome
- PhantomJS
- Screenshot service
- Screenshot library

Chrome

Chrome is the recommended engine from v1.23. To set it up:

- Install Node 8.x – earlier versions won’t work. Ensure that node is in your PATH.
- Uninstall and re-install Gramex (or run gramex install capture instead.)

Add this to gramex.yaml:

```
url:
  capture:
    pattern: /$YAMLURL/capture
    handler: CaptureHandler
    kwargs:
      engine: chrome
```

When Gramex runs, it starts node chromecapture.js --port 9900 running a node.js based web application (chromecapture.js) at port 9900.
This has the benefits of both JS & Python

JS for UI
JS for front-end interactivity

Python for data
Python for back-end processing

Another strong benefit is...

If someone has built it in JS, you can integrate it.

You don’t need to worry about re-creating it.

Other libraries
V8Py, Naked, PyExecJS
I believe we should...

Use available skills & libraries: Integrate, don’t re-write

Explore at Github.com / Gramener/GrameX

Ideeate as a group. Tweet: #RPY / #JSPY

Talk to anyone from Gramener

S.Anand @ Gramener.com