Python in the Browser

Intro to Brython

PyCon 2014

Susan Tan
Software Engineer
Email: susan.tan.fleckerl@gmail.com
Twitter: @ArcTanSusan

About Me

- Susan Tan, a recent college graduate
- Also, an open source contributor to Python projects: Open Hatch, Django, IPython notebook
- Previously, web application engineer at Flixster with Rotten Tomatoes
- I'm a Python dev for hire! I'm looking for Python jobs in San Francisco.

This is a talk about a Javascript library.

Quick Live Demo of Brython

Table of Contents

- Why Python in the browser?
- Existing Features in Brython
- Comparison of to-do list apps
- Simple speed tests
- Limitations
- Future of Brython and Javascript

Summary: An evaluation of Brython from a user perspective.

What this talk is not about

- an ad pitch
- different architectural approaches in creating python implementations in the browser

Why Python in the browser?

- For who?
  - Front-end engineers
  - Scientists
- Python for numerical computing in front-end
  - alternative or complement to D3?
- Goals:
  - Instantly interactive computation-heavy UI widgets
  - Browser-only web applications powered by Python.

What are some existing Python-to-Javascript compilers?

Stand-alone compilers
PythonJS
Pyjaco
Pyjamas
Py2JS

In-browser implementations
Skulpt
Brython

Why Python in the Browser?

- Python is more fun to write and is more readable than Javascript
- Javascript starts to get messy and unreadable very quickly

Features of Brython

- Support for core python modules, libraries, packages
- DOM API
- Avoid problems in Javascript

How to get started with Brython

1. Get the Brython.js file.

2. Load the Javascript library brython.js:

```
<script src="/path/to/brython.js">
```

3. Embed Python code inside here:

```
<script type="text/python">
```

4. Include this body tag:

```
<body onload="brython()">
```

Minimal Setup

```
<html>
<head>
<script src="path/to/brython.js"></script>
</head>
<body onload="brython()">
<script type="text/python">

"""
Your Python Code Here.
"""


</script>
</body>
</html>
```

How to access DOM elements

Some Text Here

in jquery

```
$("#UniqueIDHere")
```

in brython

```
from browser import doc
doc["UniqueIDHere"]
```

or

```
doc.get(selector="#UniqueIDHere")
```

How to add and remove css classes

in jquery

```
$("div").addClass("fooClass")
$("div").removeClass("fooClass")
```

in brython

```
div.classList.remove("fooClass")
div.classList.add("anotherclass")
```

How to create DOM elements

```
<div id="IdHere">Some Text Here</div>
```

in jquery

```
$("#IdHere").append(NewDivElement)
```

in brython

To create a new div element and store it in a

```
from browser import html
NewDivElement = html.DIV(Class="view", Id="ur
```

Then append the new div to a newly created list element:

```
html.LI() <= NewDivElement
```

How to bind event handlers

```
// Bind onclick handler on button?
def sayHi():
    print("hello")
```

in jquery

```
$("#mybutton").bind("click", sayHi)
```

in brython

```
doc["mybutton"].bind('click', sayHi)
```

How to access local storage

in javascript

```
localStorage.setItem("todo_item", "Make a cup of tea")
localStorage.getItem("todo_item")
localStorage.removeItem('favoriteflavor')
```

in brython

```
from browser.local_storage import storage
storage['foo']='bar'
del storage['foo']
```

Comparison of to-do list apps

Live Demo Time Again

Source Code on GitHub:
http://bit.ly/1nODxED

Design

Source: http://todomvc.com/

Simple Timing Tests

```
def my_func():
    a = 0
    N = 100,000
    for x in range(N):
        a += 1
```

```
# IPython notebook magic syntax
%timeit -n 1000 my_func()
```

```
# Brython code
import time
t0 = time.time()
(My Brython code here)
print(time.time()-t0)
```

Resulting Averages

Python 27: 8 ms

Javascript: 5ms, Brython: 190 ms

Another Timing Test

```
def my_func2():
    a = 0
    i = 0
    N = 100,000
    while i < N:
        a += 1
        i += 1
```

Resulting Averages

Python 27: 8.57 ms

Javascript: 5 ms

Brython: 1,000 ms

Why?

```
var $next12=getattr(iter(getattr(range,"__call__")(Number(100000))),"__r
var $no_break12=true;while(true){
    try{
        var x=$globals["x"]=$next12();None;
    }
    catch($err){
    if(__BRYTHON__.is_exc($err,[__builtins__.StopIteration]))
    }
     else{throw($err)}}
    __BRYTHON__.line_info=[3,"__main__"];None;
    var $temp=Number(1);None;
    if($temp.$fast_augm && a.$fast_augm){a+=$temp;$globals["a"]=a}
    else if(!hasattr(a,"__iadd__")){
      var a=$globals["a"]=getattr(a,"__add__")($temp);None;
    }
    else {
        a=$globals["a"]=getattr(a,"__iadd__")($temp)
    }
}
```

If it's slow, why use it?

- You don't have to
- True full-stack Python for web development
- Front-end data-intensive visualizations
  - Core Python math, datetime, json modules are supported

Limitations

- Very slow
- No support for scientific python libraries
  - scikit-learn, nltk, scipy are not supported.
- No debugger tool for pausing code during execution

Why is it hard to get a new tool or library adopted as mainstream?

- What problem does it solve?
- Is it easy to use and learn?
- Who else is using it? Why?
- Is it in active development?

The Future of Javascript?

- Javascript 6 is coming to Firefox
- A very large growing ecosystem
  - asm, dart, typescript
  - MVC frameworks: Angular, Backbone, Ember, Knockout
  - A crowded mess
- Still be mainstream for another 5 years.

The Future of Client-side Python?

Exciting times are ahead.

- mpld3?
- PyPy.js?
- Vispy?
- IPython notebook?

Thanks for listening!
Questions?

Susan Tan
@ArcTanSusan

Extra Reference Slides

How Brython manipulates the JS namespace via global window

```
from browser import window
window.echo = echo
```

Brython 2.0

- Names of functions and classes defined in Brython are no longer available in Javascript global namespace.
- The only names included in the global Javascript namespace are __BRYTHON__ (used internally) and brython(), the function called on page load.

New List Comprehrension Syntax in Ecmascript

```
// Before (by hand)
var foo = (function(){
            var result = [];
            for (var x of y)
              result.push(x*x);
            return result;
          })();

// Before (assuming y has a map() method)
var foo = y.map(function(x) { return x*x });



// After
var foo = [for (x of y) x*x];
```

New Generator Comprehension in Ecmascript

```
// Before
var bar = (function*(){ for (var x of y) yield y })();

// After
var bar = (for (x of y) y);
```