pyjs_brython

April 26, 2022

1 brython, brythonmagic

brython is an implementation of Python in javascript, byrthonmagic makes it available from a notebook.
documentation source installation tutorial

[1]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

[1]:
```
<IPython.core.display.HTML object>
```

1.1 example

[2]:
```
%load_ext brythonmagic
```

[3]:
```
%%html
<!--<script type="text/javascript" src="http://www.brython.info/src/brython_dist.js"></
  ↪script>-->
<script type="text/javascript" src="http://brython.info/src/brython.js">
</script>
```

```
<IPython.core.display.HTML object>
```

[4]:
```
html = """
<div id="paragraph">Hi</div>
"""
```

[5]:
```
%%brython -c html_ex -h html
from browser import doc

doc["paragraph"].style = {"color": "yellow",
                          "fontSize": "100px",
                          "lineHeight": "150px",
                          "textAlign": "center",
                          "backgroundColor": "black"}
```

```
<IPython.core.display.HTML object>
```

[6]:
```
%%brython -s my_dummy_function
def dummy_function(some_text):
```

```
    print(some_text)
```

```
<IPython.core.display.HTML object>
```

[7]:
```
%%brython -S my_dummy_function
dummy_function('Hi')
```

```
<IPython.core.display.HTML object>
```

[8]:
```
%%brython -c simple_example
from browser import doc, html

for i in range(10):
    doc["simple_example"] <= html.P(i)
```

```
<IPython.core.display.HTML object>
```

[9]:
```
%%brython -c canvas_example
from browser.timer import request_animation_frame as raf
from browser.timer import cancel_animation_frame as caf
from browser import doc, html
from time import time
import math

# First we create a table to insert the elements
table = html.TABLE(cellpadding = 10)
btn_anim = html.BUTTON('Animate', Id="btn-anim", type="button")
btn_stop = html.BUTTON('Stop', Id="btn-stop", type="button")
cnvs = html.CANVAS(Id="raf-canvas", width=256, height=256)

table <= html.TR(html.TD(btn_anim + btn_stop) +
                 html.TD(cnvs))

doc['canvas_example'] <= table
# Now we access the canvas context
ctx = doc['raf-canvas'].getContext( '2d' )

# And we create several functions in charge to animate and stop the draw animation
toggle = True

def draw():
    t = time() * 3
    x = math.sin(t) * 96 + 128
    y = math.cos(t * 0.9) * 96 + 128
    global toggle
    if toggle:
        toggle = False
    else:
        toggle = True
    ctx.fillStyle = 'rgb(200,200,20)' if toggle else 'rgb(20,20,200)'
    ctx.beginPath()
```

```
    ctx.arc( x, y, 6, 0, math.pi * 2, True)
    ctx.closePath()
    ctx.fill()

def animate(i):
    global id
    id = raf(animate)
    draw()

def stop(i):
    global id
    print(id)
    caf(id)

doc["btn-anim"].bind("click", animate)
doc["btn-stop"].bind("click", stop)
```

```
<IPython.core.display.HTML object>
```

[10]:
```
from brythonmagic import load_js_lib
load_js_lib("http://d3js.org/d3.v3.js")
```

```
<IPython.core.display.Javascript object>
```

[11]:
```
%%brython -c simple_d3
from browser import window, document, html
from javascript import JSObject

d3 = window.d3

container = JSObject(d3.select("#simple_d3"))
svg = container.append("svg").attr("width", 100).attr("height", 100)
circle1 = svg.append("circle").style("stroke", "gray").style("fill", "gray").attr("r",␣
 ↪40)
circle1.attr("cx", 50).attr("cy", 50).attr("id", "mycircle")

circle2 = svg.append("circle").style("stroke", "gray").style("fill", "white").
 ↪attr("r", 20)
circle2.attr("cx", 50).attr("cy", 50)

def over(ev):
    document["mycircle"].style.fill = "blue"

def out(ev):
    document["mycircle"].style.fill = "gray"

document["mycircle"].bind("mouseover", over)
document["mycircle"].bind("mouseout", out)
```

```
<IPython.core.display.HTML object>
```

[12]:
```
%%brython -c manipulating
from browser import document, html

def hide(ev):
    divs = document.get(selector = 'div.input')
    for div in divs:
        div.style.display = "none"

def show(ev):
    divs = document.get(selector = 'div.input')
    for div in divs:
        div.style.display = "inherit"

document["manipulating"] <= html.BUTTON('Hide code cells', Id="btn-hide")
document["btn-hide"].bind("click", hide)

document["manipulating"] <= html.BUTTON('Show code cells', Id="btn-show")
document["btn-show"].bind("click", show)
```

```
<IPython.core.display.HTML object>
```

[13]:
```
load_js_lib("https://cdnjs.cloudflare.com/ajax/libs/openlayers/4.6.5/OpenLayers.js")
```

```
<IPython.core.display.Javascript object>
```

[14]:
```
%%brython -c ol_map
from browser import document, window
from javascript import JSConstructor, JSObject

## Div layout
document['ol_map'].style.width = "800px"
document['ol_map'].style.height = "400px"
document['ol_map'].style.border = "1px solid black"

OpenLayers = window.OpenLayers

## Map
_map = JSConstructor(OpenLayers.Map)('ol_map')

## Addition of a OpenStreetMap layer
_layer = JSConstructor(OpenLayers.Layer.OSM)('Simple OSM map')
_map.addLayer(_layer)

## Map centered on Lon, Lat = (-3.671416, 40.435897) and a zoom = 14
## with a projection = "EPSG:4326" (Lat-Lon WGS84)
_proj = JSConstructor(OpenLayers.Projection)("EPSG:4326")
_center = JSConstructor(OpenLayers.LonLat)(-3.671416, 40.435897)
_center.transform(_proj, _map.getProjectionObject())
_map.setCenter(_center, 10)

## Addition of some points around the defined location
lons = [-3.670, -3.671, -3.672, -3.672, -3.672,
```

```
        -3.671, -3.670, -3.670]
lats = [40.435, 40.435, 40.435, 40.436, 40.437,
        40.437, 40.437, 40.436]

site_points = []
site_style = {}

points_layer = JSConstructor(OpenLayers.Layer.Vector)("Point Layer")
_map.addLayer(points_layer)

for lon, lat in zip(lons, lats):
    point = JSConstructor(OpenLayers.Geometry.Point)(lon, lat)
    point.transform(_proj, _map.getProjectionObject())
    _feat = JSConstructor(OpenLayers.Feature.Vector)(point)
    points_layer.addFeatures(_feat)
```

```
<IPython.core.display.HTML object>
```

[15]:
```
load_js_lib("http://cdnjs.cloudflare.com/ajax/libs/raphael/2.1.2/raphael-min.js")
```

```
<IPython.core.display.Javascript object>
```

[16]:
```
%%brython -c raphael_ex
from browser import window
from javascript import JSObject

Raphael = window.Raphael

paper = JSObject(Raphael("raphael_ex", 400, 400))

#Draw rectagle
rect = paper.rect(1,1,398,398)
rect.attr("stroke", "black")

#Draw orbits
for rot in range(90,280,60):
    ellipse = paper.ellipse(200, 200, 180, 50)
    ellipse.attr("stroke", "gray")
    ellipse.rotate(rot)

#Draw nucleus
nucleus = paper.circle(200,200,40)
nucleus.attr("fill", "black")

# Draw electrons
electron = paper.circle(200, 20, 10)
electron.attr("fill", "red")
electron = paper.circle(44, 290, 10)
electron.attr("fill", "yellow")
electron = paper.circle(356, 290, 10)
electron.attr("fill", "blue")
```

```
<IPython.core.display.HTML object>
```

[17]:
```
%%brython
from browser import doc, html

def show_cell_number(on = True):
    cells = doc.get(selector = '.input_prompt')
    for i, cell in enumerate(cells):
        if on:
            if 'In' in cell.html and '<br>' not in cell.html:
                cell.html += "<br>cell #" + str(i)
        else:
            if 'In' in cell.text:
                cell.html = cell.html.split('<br>')[0]

show_cell_number(on = True)
```

```
<IPython.core.display.HTML object>
```

[18]:
```

```