

notebook_html_svg

July 2, 2023

1 Notebook, HTML, SVG, Javascript, JSON

Display SVG, JSON in a notebook.

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

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

1.1 HTML

```
[2]: %%html
      <i>italic</i>
```

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

```
[3]: from IPython.display import HTML
      HTML("<b>some bold text</b>")
```

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

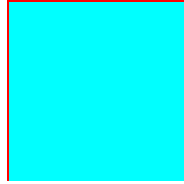
```
[4]: from IPython.display import HTML, display
      display(HTML("<b>some bold text</b>"))
```

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

1.2 SVG

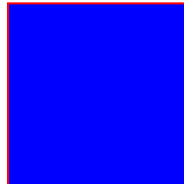
- [SVG language](#)

```
[5]: %%svg
      <svg>
        <rect x="10" y="10" height="100" width="100"
          style="stroke:#ff0000; fill: #00ffff"/>
      </svg>
```



```
[6]: from IPython.display import SVG
      SVG("""<svg>
          <rect x="10" y="10" height="100" width="100"
              style="stroke:#ff0000; fill: #0000ff"/>
        </svg>""")
```

[6]:



1.3 Javascript

1.3.1 with magic commands

Usually, the javascript needs a zone in the page where the output will be inserted. That's what the next cell does.

```
[7]: %%html
      <div id="myfirstid"></div>
```

<IPython.core.display.HTML object>

This cell will populate the zone just above. The rendering is not great when the notebook is converted into RST. Look into the HTML conversion.

```
[8]: %%javascript
      var zone = document.getElementById("myfirstid");
      var student = new Array();
      student[1]=23;
      student[2]=34;
      student[3]=67;
      student[4]=76;
      student[5]=51;
      student[6]=72;
      zone.innerHTML = "<b>Student results</b>";
      zone.innerHTML += "<ul>";
      var numStudents=0;
```

```

var sum=0;
var ans=0;
for(var i in student){
    zone.innerHTML += "<li>Student "+i+": "+student[i]+"%";
    numStudents++;
    sum=sum+student[i];
}
zone.innerHTML += "</ul>";
ans=Math.round(sum/numStudents);
zone.innerHTML += "student average is "+ans+"%";

```

<IPython.core.display.Javascript object>

1.3.2 with customized objects

```

[9]: from IPython.display import display_html, display_javascript
import uuid

class MyCustomObject(object):
    def __init__(self, width="100%", height="100%", divid=None):
        self.uuid = divid if divid else str(uuid.uuid4())
        self.width = width
        self.height = height

    def _ipython_display_(self):
        display_html('<div id="{}" style="height: {}; width: {};"></div>'.format(self.
        uuid, self.width, self.height),
                    raw=True)

    js = """
        var zone = document.getElementById("myfirstid");
        var student = new Array();
        student[1]=23;
        student[2]=34;
        student[3]=67;
        student[4]=76;
        student[5]=51;
        student[6]=72;
        zone.innerHTML = "<b>Student results</b>";
        zone.innerHTML += "<ul>";
        var numStudents=0;
        var sum=0;
        var ans=0;
        for(var i in student){
            zone.innerHTML += "<li>Student "+i+": "+student[i]+"%";
            numStudents++;
            sum=sum+student[i];
        }
        zone.innerHTML += "</ul>";
        ans=Math.round(sum/numStudents);
        zone.innerHTML += "student average is "+ans+"%";
        """.replace("myfirstid", str(self.uuid))
display_javascript(js, raw=True)

```

```
[10]: MyCustomObject()
```

1.4 JSON

Inspired from [Pretty JSON Formatting in IPython Notebook](#). We use the trick described just above. The only addition is the use of an external library `renderjson`. The code is here : [json_helper.py](#).

```
[11]: from jyquickhelper import JSONJS
      JSONJS(dict(name="xavier", city="Paris", objs=[dict(number=10, street="River"))])
```

```
[11]: <jyquickhelper.jspy.render_nb_json.RenderJSON at 0x1db60d1e198>
```

```
[12]:
```