ipywidgets Documentation

Release 5.0.0.dev

https://jupyter.org

March 07, 2016

CHAPTER 1

Contents

1.1 Contributing

We appreciate contributions from the community.

We follow the IPython Contributing Guide and Jupyter document on Contributing to the Code

1.2 Developer Install

To install ipywidgets from git, you will need npm and the latest development copy of the Jupyter notebook because everything in the ipywidgets repository is developed using Jupyter notebook master. If you want to have a copy of ipywidgets that works against a stable version of the notebook, checkout the appropriate tag (see the Compatibility table above).

If you install using sudo, you need to make sure that npm is also available in the PATH used with sudo.

1. clone the repo:

```
git clone https://github.com/ipython/ipywidgets
cd ipywidgets
```

2. Dev-install of the package (run from repo directory):

```
pip install -v -e .
```

3. Build the Jupyter Widgets package

```
cd jupyter-js-widgets
npm install
cd ..
```

4. Install the Jupyter Widgets nbextension

```
cd widgetsnbextension
npm install
npm run update:widgets
pip install -v -e .
cd ..
```

After you've made changes to `jupyter-js-widgets` and you'd like to test those changes, run the following, empty your browsers cache, and refresh the page.

```
    cd widgetsnbextension
    npm run update:widgets
    cd ..
```

TIPS: If you have any problems with the above install procedure, make sure that permissions on npm and pip related install directories are correct. Sometimes it helps to clear cached files too by running `git clean -dfx`. Also, when you make changes to the Javascript, if you're not seeing the changes it could be your browser caching aggressively. Try using "incognito" or "private" browsing tabs to avoid caching.

Index - Back - Next

1.3 Widget List

1.3.1 Complete list

For a complete list of the GUI widgets available to you, you can list the registered widget types. `Widget` and `DOMWidget`, not listed below, are base classes.

```
In [1]: import ipywidgets as widgets
        widgets.Widget.widget_types.values()
```

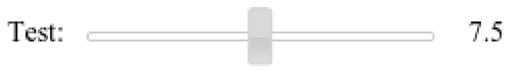
```
Out[1]: dict_values([<class 'ipywidgets.widgets.widget_selection.Dropdown'>, <class 'ipywic
```

1.3.2 Numeric widgets

There are 8 widgets distributed with IPython that are designed to display numeric values. Widgets exist for displaying integers and floats, both bounded and unbounded. The integer widgets share a similar naming scheme to their floating point counterparts. By replacing `Float` with `Int` in the widget name, you can find the Integer equivalent.

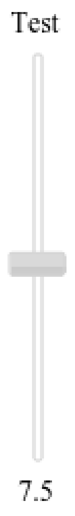
FloatSlider

```
In [2]: widgets.FloatSlider(
            value=7.5,
            min=5.0,
            max=10.0,
            step=0.1,
            description='Test:',
        )
```



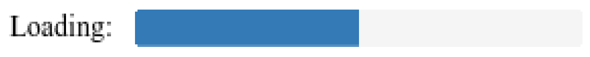
Sliders can also be **displayed vertically**.

```
In [3]: widgets.FloatSlider(
            value=7.5,
            min=5.0,
            max=10.0,
            step=0.1,
            description='Test',
            orientation='vertical',
        )
```



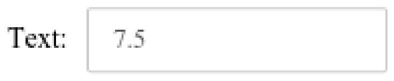
FloatProgress

```
In [4]: widgets.FloatProgress(
            value=7.5,
            min=5.0,
            max=10.0,
            step=0.1,
            description='Loading:',
        )
```



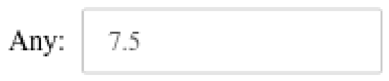
BoundedFloatText

```
In [5]: widgets.BoundedFloatText(
            value=7.5,
            min=5.0,
            max=10.0,
            description='Text:',
        )
```



FloatText

```
In [6]: widgets.FloatText(
            value=7.5,
            description='Any:',
        )
```

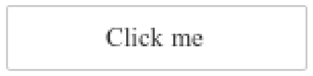


1.3.3 Boolean widgets

There are three widgets that are designed to display a boolean value.

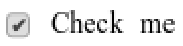
ToggleButton

```
In [7]: widgets.ToggleButton(
            description='Click me',
            value=False,
        )
```



Checkbox

```
In [8]: widgets.Checkbox(
            description='Check me',
            value=True,
        )
```



Valid

The valid widget provides a read-only indicator.

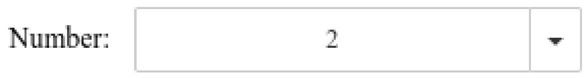
```
In [9]: widgets.Valid(
            value=True,
        )
```

1.3.4 Selection widgets

There are four widgets that can be used to display single selection lists, and one that can be used to display multiple selection lists. All inherit from the same base class. You can specify the **enumeration of selectable options by passing a list**. You can **also specify the enumeration as a dictionary**, in which case the **keys will be used as the item displayed** in the list and the corresponding **value will be returned** when an item is selected.

Dropdown

```
In [10]: from IPython.display import display
         w = widgets.Dropdown(
             options=['1', '2', '3'],
             value='2',
             description='Number:',
         )
         display(w)
```

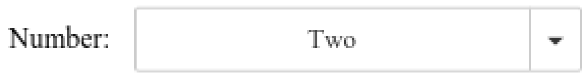


```
In [11]: w.value
```

```
Out[11]: '2'
```

The following is also valid:

```
In [12]: w = widgets.Dropdown(
             options={'One': 1, 'Two': 2, 'Three': 3},
             value=2,
             description='Number:',
         )
         display(w)
```



```
In [13]: w.value
```

```
Out[13]: 2
```

Furthermore, if a dropdown contains too man items, a scrollbar is automatically added.

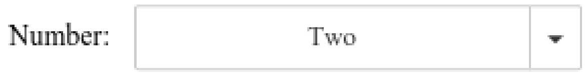
```
In [14]: from IPython.display import display
         w = widgets.Dropdown(
             options=['1', '2', '3', '4', '5', '6', '7', '8', '9'],
             value='2',
             description='Number:',
         )
         display(w)
```

```
In [15]: w.value
```

```
Out[15]: '2'
```

The following is also valid:

```
In [16]: w = widgets.Dropdown(
             options={'One': 1, 'Two': 2, 'Three': 3,
                      'Four': 4, 'Five': '5', 'Six': 6,
                      'Seven': 7, 'Eight': 8, 'Nine': 9},
             value=2,
             description='Number:',
         )
         display(w)
```

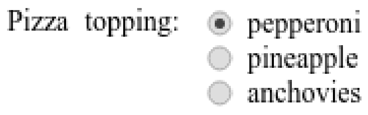


```
In [17]: w.value
```

```
Out[17]: 2
```

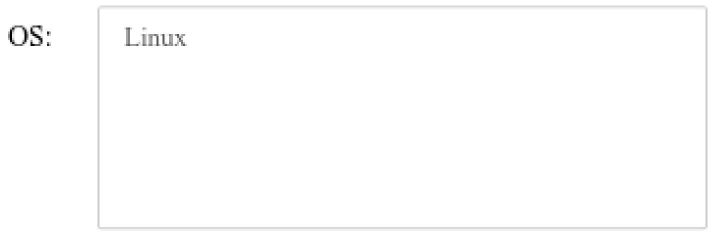
RadioButtons

```
In [18]: widgets.RadioButtons(
             description='Pizza topping:',
             options=['pepperoni', 'pineapple', 'anchovies'],
         )
```



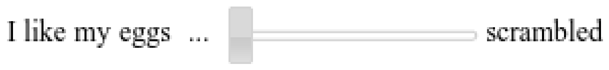
Select

```
In [19]: widgets.Select(
             description='OS:',
             options=['Linux', 'Windows', 'OSX'],
         )
```



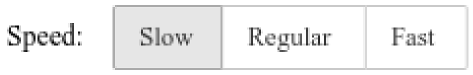
SelectionSlider

```
In [20]: widgets.SelectionSlider(
             description='I like my eggs ...',
             options=['scrambled', 'sunny side up', 'poached', 'over easy'],
         )
```



ToggleButtons

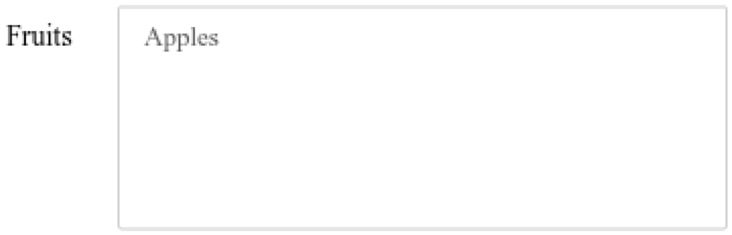
```
In [21]: widgets.ToggleButtons(
             description='Speed:',
             options=['Slow', 'Regular', 'Fast'],
         )
```



SelectMultiple

Multiple values can be selected with shift and/or ctrl (or command) pressed and mouse clicks or arrow keys.

```
In [22]: w = widgets.SelectMultiple(
             description="Fruits",
             options=['Apples', 'Oranges', 'Pears']
         )
         display(w)
```



```
In [23]: w.value
```

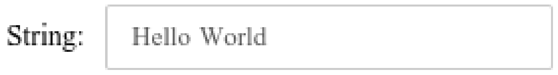
```
Out[23]: ('Apples',)
```

1.3.5 String widgets

There are 4 widgets that can be used to display a string value. Of those, the `Text` and `Textarea` widgets accept input. The `Latex` and `HTML` widgets display the string as either Latex or HTML respectively, but do not accept input.

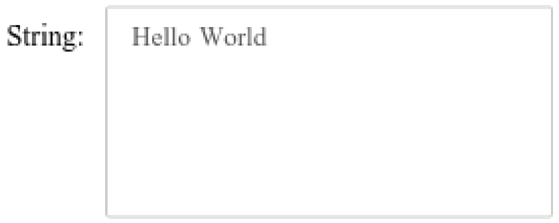
Text

```
In [24]: widgets.Text(
             description='String:',
             value='Hello World',
         )
```



Textarea

```
In [25]: widgets.Textarea(
             description='String:',
             value='Hello World',
         )
```



Latex

```
In [26]: widgets.Latex(
              value="$$\\frac{n!}{k!(n-k)!} = \\binom{n}{k}$$",
         )
```

$$\frac{n!}{k!(n-k)!} = \binom{n}{k}$$

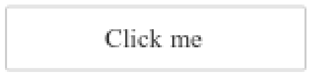
HTML

```
In [27]: widgets.HTML(
              value="Hello <b>World</b>"
         )
```

Hello **World**

1.3.6 Button

```
In [28]: widgets.Button(description='Click me')
```



Index - Back - Next