plotly | Dash

Star 16,107

Python

Dash Python ***> Dash Core Components***

Dash Core Components

Dash ships with supercharged components for interactive user interfaces.

The Dash Core Components module (`dash.dcc`) can be imported and used with `from dash import dcc` and gives you access to many interactive components, including, dropdowns, checklists, and sliders.

The `dcc` module is part of Dash and you'll find the source for it in the **Dash GitHub repo**.

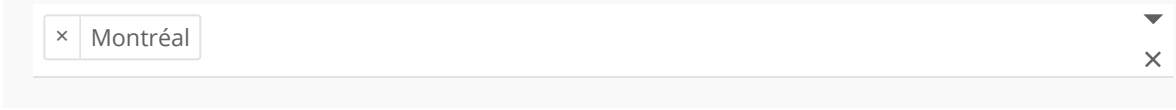
> For production Dash apps, the Dash Core Components styling & layout should be managed with Dash Enterprise **Design Kit**.

Dropdown

```
from dash import Dash, html, dcc

app = Dash(__name__)

app.layout = html.Div([
    dcc.Dropdown(['New York City', 'Montréal', 'San Francisco'], 'Montréal')
])

if __name__ == '__main__':
    app.run_server(debug=True)
```

Montréal

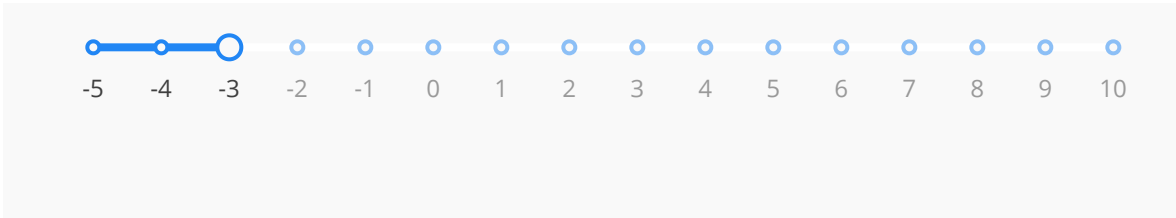
```
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
    dcc.Dropdown(['New York City', 'Montréal', 'San Francisco'], 'Montréal', multi=T
])

if __name__ == '__main__':
    app.run_server(debug=True)
```



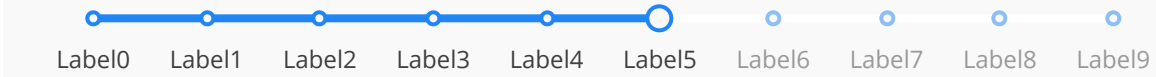
More Dropdown Examples and Reference

Slider

```
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
    dcc.Slider(-5, 10, 1, value=-3)
])

if __name__ == '__main__':
    app.run_server(debug=True)
```




```
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
    dcc.Slider(0, 9, marks={i: f'Label{i}' for i in range(10)}, value=5)
])

if __name__ == '__main__':
    app.run_server(debug=True)
```

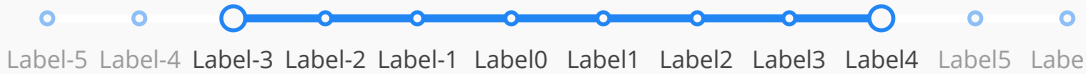


More Slider Examples and Reference

RangeSlider

```
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
    dcc.RangeSlider(-5, 10, 1, count=1, value=[-3, 7])
])

if __name__ == '__main__':
    app.run_server(debug=True)
```

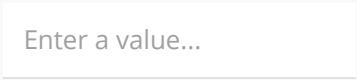

```
from dash import Dash, html, dcc

app = Dash(__name__)

app.layout = html.Div([
    dcc.RangeSlider(-5, 6,
        marks={i: f'Label{i}' for i in range(-5, 7)},
        value=[-3, 4]
    )
])

if __name__ == '__main__':
    app.run_server(debug=True)
```



More RangeSlider Examples and Reference

Input

```
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
    dcc.Input(
        placeholder='Enter a value...',
        type='text',
        value=''
    )
])

if __name__ == '__main__':
    app.run_server(debug=True)
```



More Input Examples and Reference

Textarea

```
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
    dcc.Textarea(
        placeholder='Enter a value...',
        value='This is a TextArea component',
        style={'width': '100%'}
    )
])

if __name__ == '__main__':
    app.run_server(debug=True)
```

This is a TextArea component

Textarea Reference

Checkboxes

```
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
    dcc.Checklist(['New York City', 'Montréal', 'San Francisco'],
                  ['Montréal', 'San Francisco'])
])

if __name__ == '__main__':
    app.run_server(debug=True)
```

- [ ] New York City
- [x] Montréal
- [x] San Francisco

```
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
    dcc.Checklist(
        ['New York City', 'Montréal', 'San Francisco'],
        ['Montréal', 'San Francisco'],
        inline=True
    )
])

if __name__ == '__main__':
    app.run_server(debug=True)
```

☐ New York City ☑ Montréal ☑ San Francisco

Checklist Properties

Radio Items

```
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
    dcc.RadioItems(['New York City', 'Montréal', 'San Francisco'], 'Montréal')
])

if __name__ == '__main__':
    app.run_server(debug=True)
```

- ○ New York City
- ◉ Montréal
- ○ San Francisco

```
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
    dcc.RadioItems(
        ['New York City', 'Montréal', 'San Francisco'],
        'Montréal',
        inline=True
    )
])

if __name__ == '__main__':
    app.run_server(debug=True)
```

○ New York City ◉ Montréal ○ San Francisco

Radio Items Reference

Button

There actually is no `Button` component in `dash_core_components`. The regular `dash_html_components.Button` component does the job quite well, but we've included it here

because this is the one plain `html` component that's commonly used as a callback input:

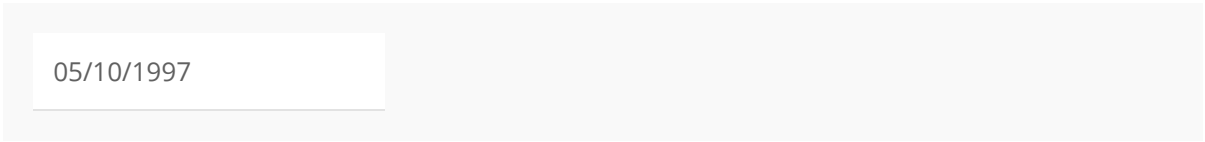
```
import dash
from dash import html, dcc

external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']

app = dash.Dash(__name__, external_stylesheets=external_stylesheets)
app.layout = html.Div([
    html.Div(dcc.Input(id='input-box', type='text')),
    html.Button('Submit', id='button-example-1'),
    html.Div(id='output-container-button',
             children='Enter a value and press submit')
])


@app.callback(
    dash.dependencies.Output('output-container-button', 'children'),
    [dash.dependencies.Input('button-example-1', 'n_clicks')],
    [dash.dependencies.State('input-box', 'value')])
def update_output(n_clicks, value):
    return 'The input value was "{}" and the button has been clicked {} times'.forma
        value,
        n_clicks
    )


if __name__ == '__main__':
    app.run_server(debug=True)
```

html.Button Reference For more on `dash.dependencies.State`, see the tutorial on **basic callbacks**.

DatePickerSingle

```
from dash import Dash, dcc, html
from datetime import date

app = Dash(__name__)
```

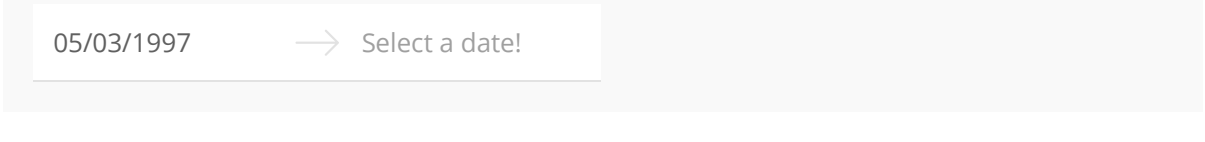
```
app.layout = html.Div([
    dcc.DatePickerSingle(
        id='date-picker-single',
        date=date(1997, 5, 10)
    )
])

if __name__ == '__main__':
    app.run_server(debug=True)
```



More DatePickerSingle Examples and Reference

DatePickerRange

```
from dash import Dash, dcc, html
from datetime import date

app = Dash(__name__)

app.layout = html.Div([
    dcc.DatePickerRange(
        id='date-picker-range',
        start_date=date(1997, 5, 3),
        end_date_placeholder_text='Select a date!'
    )
])

if __name__ == '__main__':
    app.run_server(debug=True)
```



More DatePickerRange Examples and Reference

Markdown

```
from dash import Dash, dcc, html

app = Dash(__name__)
```

```
app.layout = html.Div([
    dcc.Markdown('''
        #### Dash and Markdown
        Dash supports [Markdown](http://commonmark.org/help).
        Markdown is a simple way to write and format text.
        It includes a syntax for things like **bold text** and *italics*,
        [links](http://commonmark.org/help), inline `code` snippets, lists,
        quotes, and more.
    ''')
])

if __name__ == '__main__':
    app.run_server(debug=True)
```

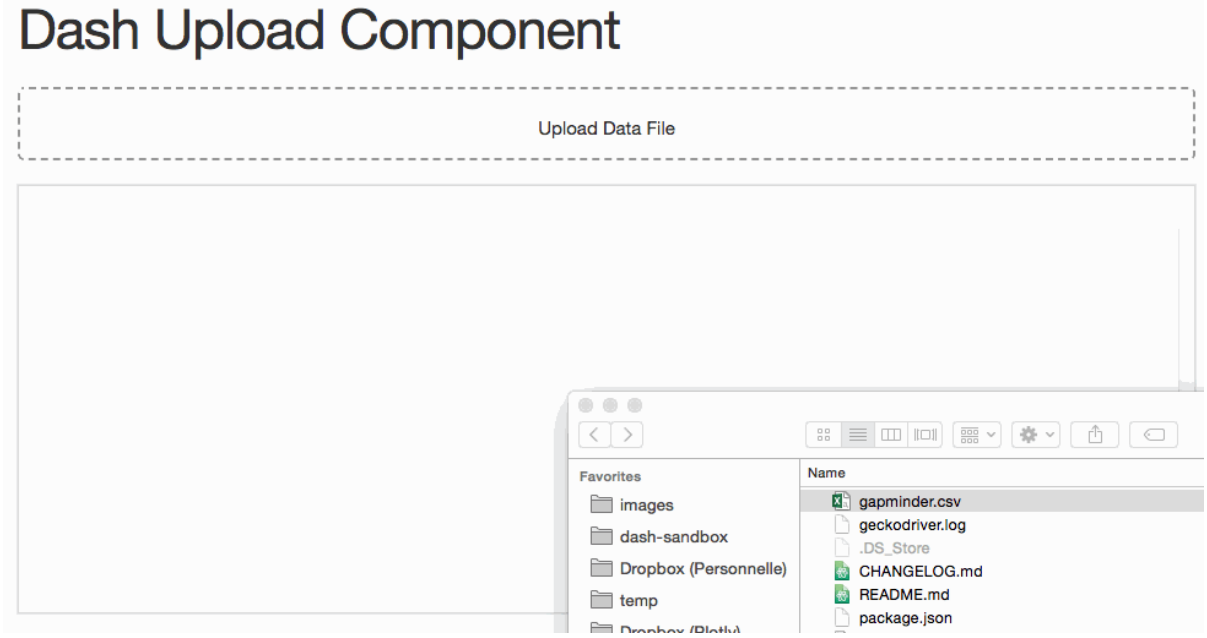
Dash and Markdown

Dash supports **Markdown**. Markdown is a simple way to write and format text. It includes a syntax for things like **bold text** and *italics*, **links**, inline `code` snippets, lists, quotes, and more.

More Markdown Examples and Reference

Upload Component

The `dcc.Upload` component allows users to upload files into your app through drag-and-drop or the system's native file explorer.



More Upload Examples and Reference

Download Component

The `dcc.Download` component allows users to download files from your app through their browser.

```
import dash
from dash.dependencies import Output, Input
from dash import dcc, html

app = dash.Dash(prevent_initial_callbacks=True)

app.layout = html.Div(
    [html.Button("Download Text", id="btn_txt"), dcc.Download(id="download-text-inde
)


@app.callback(Output("download-text-index", "data"), Input("btn_txt", "n_clicks"))
def func(n_clicks):
    if n_clicks is None:
        raise dash.exceptions.PreventUpdate
    else:
        return dict(content="Hello world!", filename="hello.txt")


if __name__ == "__main__":
    app.run_server(debug=True)
```

DOWNLOAD TEXT

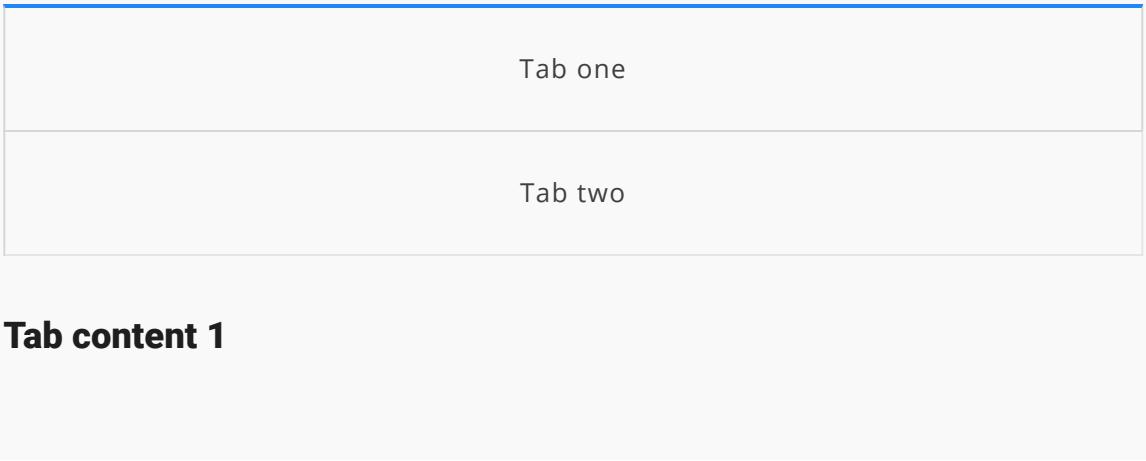
More Download Examples and Reference

Tabs

The Tabs and Tab components can be used to create tabbed sections in your app.

```
from dash import Dash, dcc, html

from dash.dependencies import Input, Output

external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']

app = Dash(__name__, external_stylesheets=external_stylesheets)

app.layout = html.Div([
    dcc.Tabs(id="tabs", value='tab-1', children=[
```

```
        dcc.Tab(label='Tab one', value='tab-1'),
        dcc.Tab(label='Tab two', value='tab-2'),
    ]),
    html.Div(id='tabs-content')
])

@app.callback(Output('tabs-content', 'children'),
              Input('tabs', 'value'))
def render_content(tab):
    if tab == 'tab-1':
        return html.Div([
            html.H3('Tab content 1')
        ])
    elif tab == 'tab-2':
        return html.Div([
            html.H3('Tab content 2')
        ])


if __name__ == '__main__':
    app.run_server(debug=True)
```



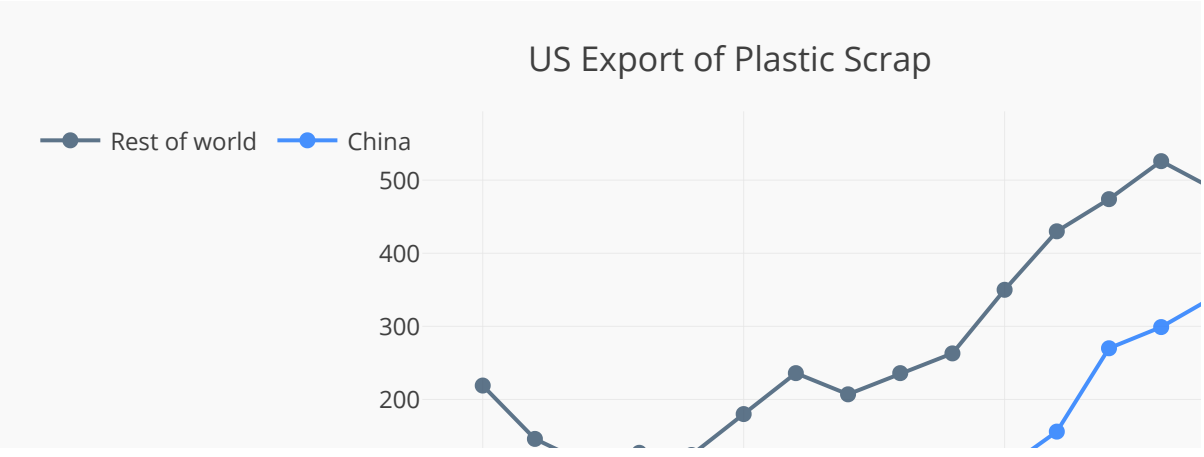
More Tabs Examples and Reference

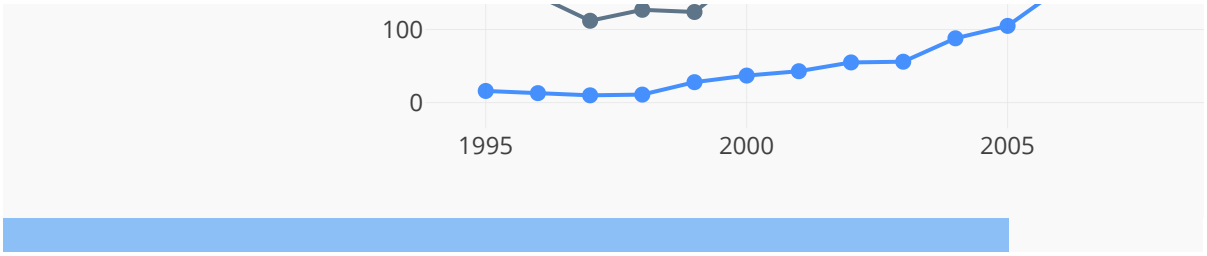
Graphs

The `Graph` component shares the same syntax as the open-source `plotly.py` library. View the **plotly.py docs** to learn more.

```
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
    dcc.Graph(
        figure=dict(
```

```
            data=[
                dict(
                    x=[1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003,
                    2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012],
                    y=[219, 146, 112, 127, 124, 180, 236, 207, 236, 263,
                    350, 430, 474, 526, 488, 537, 500, 439],
                    name='Rest of world',
                    marker=dict(
                        color='rgb(55, 83, 109)'
                    )
                ),
                dict(
                    x=[1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003,
                    2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012],
                    y=[16, 13, 10, 11, 28, 37, 43, 55, 56, 88, 105, 156, 270,
                    299, 340, 403, 549, 499],
                    name='China',
                    marker=dict(
                        color='rgb(26, 118, 255)'
                    )
                )
            ],
            layout=dict(
                title='US Export of Plastic Scrap',
                showlegend=True,
                legend=dict(
                    x=0,
                    y=1.0
                ),
                margin=dict(l=40, r=0, t=40, b=30)
            )
        ),
        style={'height': 300},
        id='my-graph-example'
    )
])

if __name__ == '__main__':
    app.run_server(debug=True)
```





More Graphs Examples and Reference

View the **plotly.py docs**.

ConfirmDialog

The `dcc.ConfirmDialog` component send a dialog to the browser asking the user to confirm or cancel with a custom message.

```
from operator import imod
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
    dcc.ConfirmDialog(
        id='confirm',
        message='Danger danger! Are you sure you want to continue?'
    )
])

if __name__ == '__main__':
    app.run_server(debug=True)
```

More ConfirmDialog Examples and Reference

There is also a dcc.ConfirmDialogProvider, it will automatically wrap a child component to send a `dcc.ConfirmDialog` when clicked.

```
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
    dcc.ConfirmDialogProvider(
        children=html.Button(
            'Click Me',
        ),
        id='danger-danger',
```

```
        message='Danger danger! Are you sure you want to continue?'
    )
])

if __name__ == '__main__':
    app.run_server(debug=True)
```

CLICK ME

More ConfirmDialogProvider Examples and Reference

Store

The store component can be used to keep data in the visitor's browser. The data is scoped to the user accessing the page. **Three types of storage (storage_type prop):**

- `memory`: default, keep the data as long the page is not refreshed.
- `local`: keep the data until it is manually cleared.
- `session`: keep the data until the browser/tab closes.

For local/session, the data is serialized as json when stored.

```
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
    dcc.Store(id='my-store', data={'my-data': 'data'})
])

if __name__ == '__main__':
    app.run_server(debug=True)
```

The store must be used with callbacks

More Store Examples and Reference

Logout Button

The logout button can be used to perform logout mechanism.

It's a simple form with a submit button, when the button is clicked, it will submit the form to the `logout_url` prop.

Please note that no authentication is performed in Dash by default and you have to implement the authentication yourself.

More Logout Button Examples and Reference

Loading component

The Loading component can be used to wrap components that you want to display a spinner for, if they take too long to load.

It does this by checking if any of the Loading components' children have a `loading_state` prop set where `is_loading` is true.

If true, it will display one of the built-in CSS spinners.

```
import dash_design_kit as ddk
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
    dcc.Loading([
        # ...
    ])
])

if __name__ == '__main__':
    app.run_server(debug=True)
```

More Loading Component Examples and Reference

Location

The location component represents the location bar in your web browser. Through its `href`, `pathname`, `search` and `hash` properties you can access different portions of your app's url. For example, given the url `http://127.0.0.1:8050/page-2?a=test#quiz`:

- `href` = `"http://127.0.0.1:8050/page-2?a=test#quiz"`
- `pathname` = `"/page-2"`
- `search` = `"?a=test"`

- `hash` = `"#quiz"`

```
dcc.Location(id="url", refresh=false)
```

More Location Examples and Reference

Dash Python ***> Dash Core Components***