plotly

Star 23,446

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

> Plotly Studio: Transform any dataset into an interactive data application in minutes with AI. **Sign up for early access now.**

Dash Core Components

Dash ships with supercharged components for interactive user interfaces.

The Dash Core Components module (`dash.dcc`) gives you access to many interactive components, including dropdowns, checklists, and sliders.

Import `dash.dcc` with:

```
from dash import dcc
```

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

> **Tip**: In production Dash apps, we recommend using Dash Enterprise **Design Kit** to manage the styling and layout of Dash Core Components.

Dropdown

```
from dash import Dash, html, dcc

app = Dash()

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

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

Montréal

```
from dash import Dash, dcc, html

app = Dash()

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

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

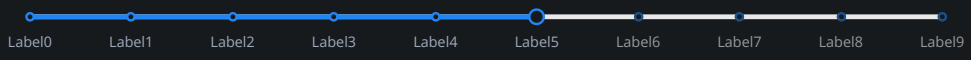
× Montréal

More Dropdown Examples and Reference

Slider

```
from dash import Dash, dcc, html

app = Dash()

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

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

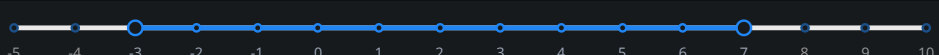

```
from dash import Dash, dcc, html

app = Dash()

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(debug=True)
```



More Slider Examples and Reference

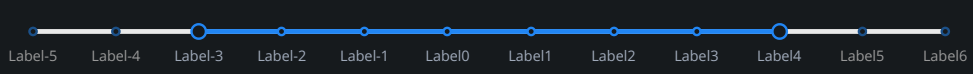
RangeSlider

```
from dash import Dash, dcc, html

app = Dash()

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

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




```
from dash import Dash, html, dcc

app = Dash()

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(debug=True)
```



More RangeSlider Examples and Reference

Input

```
from dash import Dash, dcc, html

app = Dash()

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

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

Enter a value...

More Input Examples and Reference

Textarea

```
from dash import Dash, dcc, html

app = Dash()

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

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

This is a TextArea component

Textarea Reference

Checkboxes

```
from dash import Dash, dcc, html

app = Dash()
```

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

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

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

```
from dash import Dash, dcc, html

app = Dash()

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

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

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

Checklist Properties

Radio Items

```
from dash import Dash, dcc, html

app = Dash()

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

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

- ( ) New York City
- (•) Montréal
- ( ) San Francisco

```
from dash import Dash, dcc, html

app = Dash()

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

```
if __name__ == '__main__':
    app.run(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:

```
from dash import Dash, html, dcc, callback, Input, Output, State, callback

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

app = 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')
])


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


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

SUBMIT

The input value was "None" and the button has been clicked None times

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

Sign up for Dash Club → Two free cheat sheets plus updates from Chris Parmer and Adam Schroeder delivered to your inbox every two months. Includes tips and tricks, community apps, and deep dives into the Dash architecture. **Join now**.

DatePickerSingle

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

app = Dash()

app.layout = html.Div([
    dcc.DatePickerSingle(
        id='date-picker-single',
```

```
        date=date(1997, 5, 10)
    )
])

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

More DatePickerSingle Examples and Reference

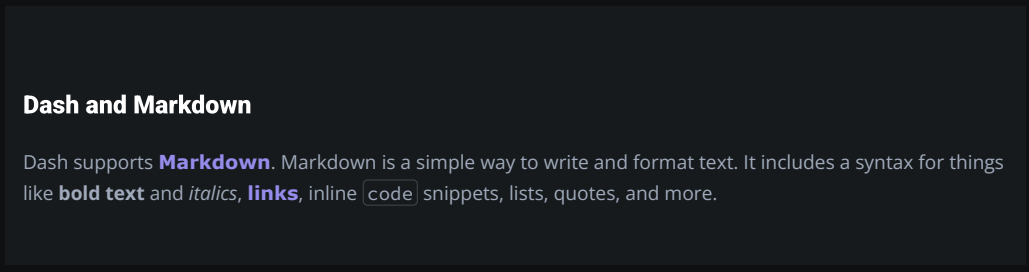
DatePickerRange

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

app = Dash()

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(debug=True)
```

More DatePickerRange Examples and Reference

Markdown

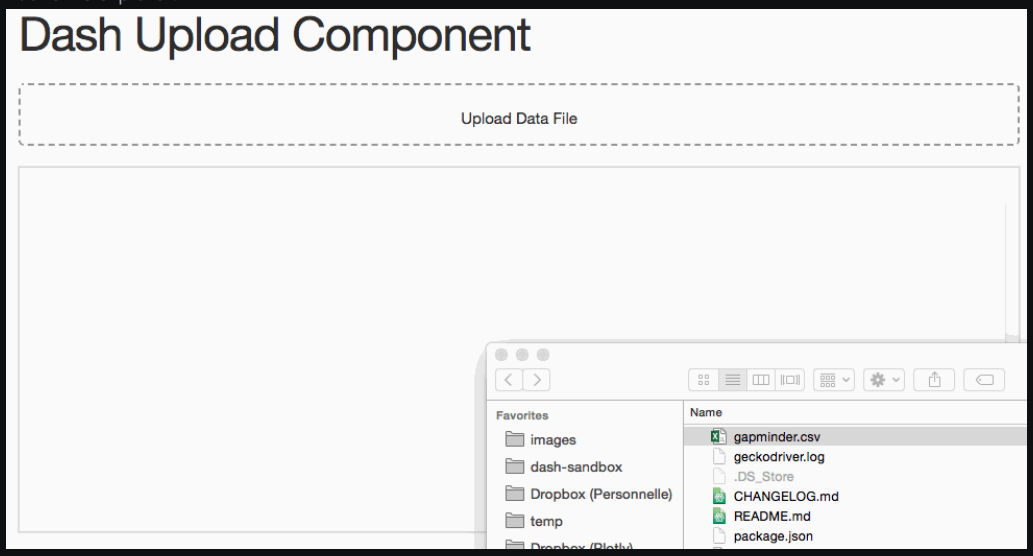
```
from dash import Dash, dcc, html

app = Dash()

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(debug=True)
```



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.

```
from dash import Dash, dcc, html, Input, Output, callback
from dash.exceptions import PreventUpdate

app = Dash(prevent_initial_callbacks=True)

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


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


if __name__ == "__main__":
    app.run(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, Input, Output, callback

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')
])

@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(debug=True)
```

Tab one

Tab two

Tab content 1

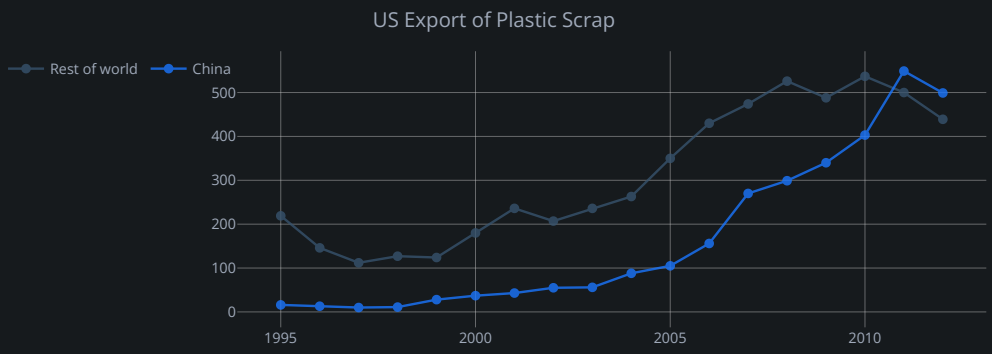
More Tabs Examples and Reference

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()

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(debug=True)
```



More Graphs Examples and Reference

View the **plotly.py docs**.

ConfirmDialogProvider

The `dcc.ConfirmDialogProvider` component sends a `dcc.ConfirmDialog` when a user clicks the `children` of the component. In the following example, an `html.Button` is provided as `children` to `dcc.ConfirmDialogProvider` and when the button is clicked, the dialog is displayed.

```
from dash import Dash, dcc, html

app = Dash()

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(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 as 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.*

In this example, one callback saves the value of the selected radio button to a memory store. When the value in the store changes, the second callback outputs the new value and the modified timestamp to a `html.Div` component.

```
from dash import Dash, dcc, html, Input, Output, State, callback

app = Dash()

app.layout = html.Div([
    dcc.Store(id='my-store'),
    dcc.RadioItems(['NYC', 'MTL', 'SF'], 'NYC', id='my-store-input'),
    html.Div(id='current-store')
])

@callback(
    Output('my-store', 'data'),
    Input('my-store-input', 'value')
)
def update_store(value):
    return value

@callback(
    Output('current-store', 'children'),
    Input('my-store', 'modified_timestamp'),
    State('my-store', 'data')
)
def display_store_info(timestamp, data):
    return f"The store currently contains {data} and the modified timestamp is {timestamp}"

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

(●) NYC
( ) MTL
( ) SF
The store currently contains NYC and the modified timestamp is 1752654167875

The store must be used with callbacks

More Store Examples and Reference

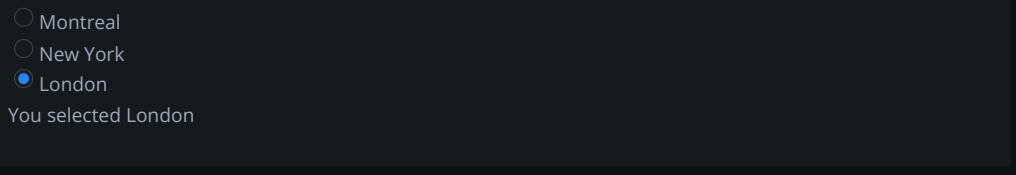
Loading Component

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

This example shows a spinner each time a user selects a radio button as the callback takes 2 seconds to update the `html.Div` component (`id=loading-demo`) inside the `dcc.Loading` component.

```
from dash import Dash, dcc, html, Input, Output, callback
import time

app = Dash()

app.layout = html.Div(
    [
        dcc.RadioItems(
            ["Montreal", "New York", "London"], "London", id="loading-demo-dropdown"
        ),
```

```
            dcc.Loading([html.Div(id="loading-demo")]),
        ]
    )

    @callback(Output("loading-demo", "children"), Input("loading-demo-dropdown", "value"))
    def update_loading_div(value):
        time.sleep(2)
        return f"You selected {value}"

    if __name__ == "__main__":
        app.run(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***