plotly

Star 23,446

Dash Python > ***Part 1. Layout***

Dash Layout

This is the 1st chapter of the **Dash Fundamentals**. The **next chapter** covers Dash callbacks.

This tutorial will walk you through a fundamental aspect of Dash apps, the app `layout`, through six self-contained apps.

For production Dash apps, we recommend styling the app `layout` with Dash Enterprise **Design Kit**.

Dash apps are composed of two parts. The first part is the "`layout`", which describes what the app looks like. The second part describes the interactivity of the app and will be covered in the **next chapter**.

Note: Throughout this documentation, each Python code example can be run either by saving it to an `app.py` file and using `python app.py` or by running it in a Jupyter notebook cell.

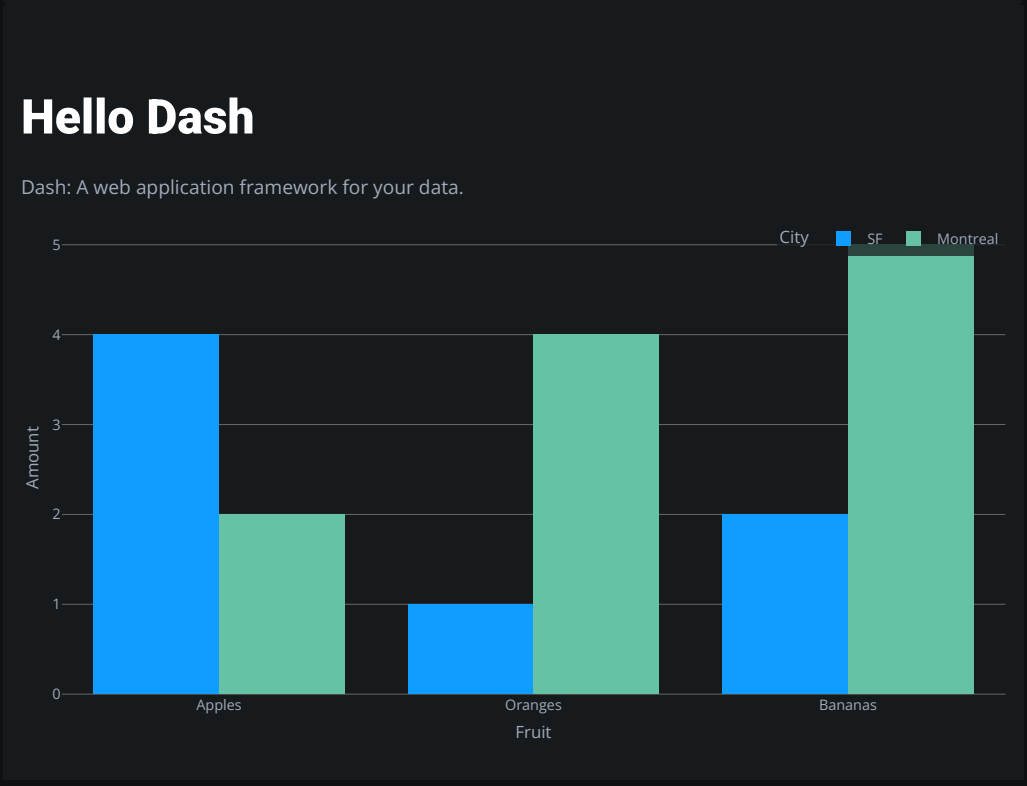
If you're using Dash Enterprise's **Data Science Workspaces**, copy & paste the below code into your Workspace (**see video**).

Find out if your company is using Dash Enterprise

To get started, create a file named `app.py`, copy the code below into it, and then run it with `python app.py`.

```
# Run this app with `python app.py` and
# visit http://127.0.0.1:8050/ in your web browser.


from dash import Dash, html, dcc
import plotly.express as px
import pandas as pd

app = Dash()

# assume you have a "long-form" data frame
# see https://plotly.com/python/px-arguments/ for more options
df = pd.DataFrame({
    "Fruit": ["Apples", "Oranges", "Bananas", "Apples", "Oranges", "Bananas"],
    "Amount": [4, 1, 2, 2, 4, 5],
    "City": ["SF", "SF", "SF", "Montreal", "Montreal", "Montreal"]
})

fig = px.bar(df, x="Fruit", y="Amount", color="City", barmode="group")

app.layout = html.Div(children=[
    html.H1(children='Hello Dash'),

    html.Div(children='''
        Dash: A web application framework for your data.
    '''),

    dcc.Graph(
        id='example-graph',
        figure=fig
    )
])
```

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




```
$ python app.py
...Running on http://127.0.0.1:8050/ (Press CTRL+C to quit)
```

Visit **http://127.0.0.1:8050/** in your web browser. You should see an app that looks like the one above.

Note:

1. The `layout` is composed of a tree of "components" such as `html.Div` and `dcc.Graph`.
2. The Dash HTML Components module (`dash.html`) has a component for every HTML tag. The `html.H1(children='Hello Dash')` component generates a `<h1>Hello Dash</h1>` HTML element in your app.
3. Not all components are pure HTML. The Dash Core Components module (`dash.dcc`) contains higher-level components that are interactive and are generated with JavaScript, HTML, and CSS through the React.js library.
4. Each component is described entirely through keyword attributes. Dash is *declarative*: you will primarily describe your app through these attributes.
5. The `children` property is special. By convention, it's always the first attribute which means that you can omit it: `html.H1(children='Hello Dash')` is the same as `html.H1('Hello Dash')`. It can contain a string, a number, a single component, or a list of components.
6. The fonts in your app will look a little bit different than what is displayed here. This app is using a custom CSS stylesheet and Dash Enterprise **Design Kit** to modify the default styles of the elements. You can learn more about custom CSS in the **CSS tutorial**.

Making Your First Change

Dash includes "hot-reloading". This feature is activated by default when you run your app with `app.run(debug=True)`. This means that Dash will automatically refresh your browser when you make a change in your code.

Give it a try: change the title "Hello Dash" in your app or change the `x` or the `y` data. Your app should auto-refresh with your change.

> Don't like hot-reloading? You can turn this off with python `app.run(dev_tools_hot_reload=False)`. Learn more in

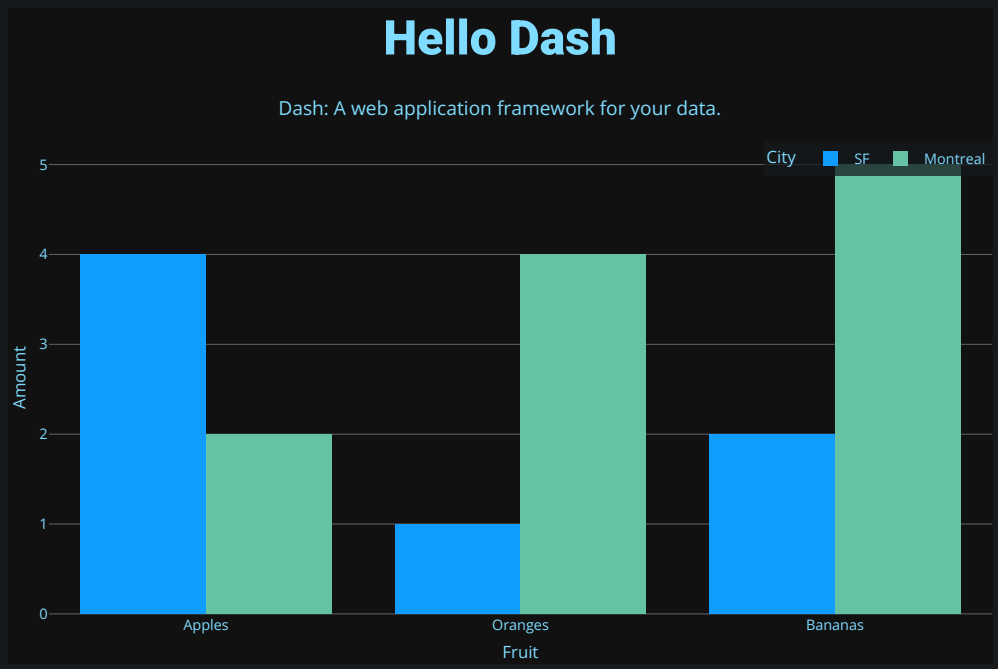
Dash Dev Tools documentation Questions? See the **community forum hot reloading discussion**.

More about HTML Components

Dash HTML Components (`dash.html`) contains a component class for every HTML tag as well as keyword arguments for all of the HTML arguments.

Let's customize the text in our app by modifying the inline styles of the components. Create a file named `app.py` with the following code:

```
# Run this app with `python app.py` and
# visit http://127.0.0.1:8050/ in your web browser.


from dash import Dash, dcc, html
import plotly.express as px
import pandas as pd

app = Dash()

colors = {
    'background': '#111111',
    'text': '#7FDBFF'
}

# assume you have a "long-form" data frame
# see https://plotly.com/python/px-arguments/ for more options
df = pd.DataFrame({
    "Fruit": ["Apples", "Oranges", "Bananas", "Apples", "Oranges", "Bananas"],
    "Amount": [4, 1, 2, 2, 4, 5],
    "City": ["SF", "SF", "SF", "Montreal", "Montreal", "Montreal"]
})

fig = px.bar(df, x="Fruit", y="Amount", color="City", barmode="group")

fig.update_layout(
    plot_bgcolor=colors['background'],
    paper_bgcolor=colors['background'],
    font_color=colors['text']
)

app.layout = html.Div(style={'backgroundColor': colors['background']}, children=[
    html.H1(
        children='Hello Dash',
        style={
            'textAlign': 'center',
            'color': colors['text']
        }
    ),

    html.Div(children='Dash: A web application framework for your data.', style={
        'textAlign': 'center',
        'color': colors['text']
    }),

    dcc.Graph(
        id='example-graph-2',
        figure=fig
    )
])

if __name__ == '__main__':
```



In this example, we modified the inline styles of the `html.Div` and `html.H1` components with the `style` property.

```
html.H1('Hello Dash', style={'textAlign': 'center', 'color': '#7FDBFF'})
```

The above code is rendered in the Dash app as `<h1 style="text-align: center; color: #7FDBFF">Hello Dash</h1>`.

There are a few important differences between the `dash.html` and the HTML attributes:

1. The `style` property in HTML is a semicolon-separated string. In Dash, you can just supply a dictionary.
2. The keys in the `style` dictionary are **camelCased**. So, instead of `text-align`, it's `textAlign`.
3. The HTML `class` attribute is `className` in Dash.
4. The children of the HTML tag is specified through the `children` keyword argument. By convention, this is always the *first* argument and so it is often omitted.

Besides that, all of the available HTML attributes and tags are available to you within your Python context.

Reusable Components

By writing our markup in Python, we can create complex reusable components like tables without switching contexts or languages.

Here's a quick example that generates a `Table` from a Pandas dataframe. Create a file named `app.py` with the following code:

```
# Run this app with `python app.py` and
# visit http://127.0.0.1:8050/ in your web browser.


from dash import Dash, html
import pandas as pd

df = pd.read_csv('https://gist.githubusercontent.com/chriddyp/c78bf172206ce24f77d6363a2d754b59/

def generate_table(dataframe, max_rows=10):
    return html.Table([
        html.Thead(
```

```
            html.Tr([html.Th(col) for col in dataframe.columns])
        ),
        html.Tbody([
            html.Tr([
                html.Td(dataframe.iloc[i][col]) for col in dataframe.columns
            ]) for i in range(min(len(dataframe), max_rows))
        ])
    ])


app = Dash()

app.layout = html.Div([
    html.H4(children='US Agriculture Exports (2011)'),
    generate_table(df)
])

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

US Agriculture Exports (2011)

Unnamed: 0	state	total exports	beef	pork	poultry	dairy	fruits fresh	fruits proc	total fruits
0	Alabama	1390.63	34.4	10.6	481	4.06	8	17.1	25.11
1	Alaska	13.31	0.2	0.1	0	0.19	0	0	0
2	Arizona	1463.17	71.3	17.9	0	105.48	19.3	41	60.27
3	Arkansas	3586.02	53.2	29.4	562.9	3.53	2.2	4.7	6.88
4	California	16472.88	228.7	11.1	225.4	929.95	2791.8	5944.6	8736.
5	Colorado	1851.33	261.4	66	14	71.94	5.7	12.2	17.99
6	Connecticut	259.62	1.1	0.1	6.9	9.49	4.2	8.9	13.1
7	Delaware	282.19	0.4	0.6	114.7	2.3	0.5	1	1.53
8	Florida	3764.09	42.6	0.9	56.9	66.31	438.2	933.1	1371.
9	Georgia	2860.84	31	18.9	630.4	38.38	74.6	158.9	233.5

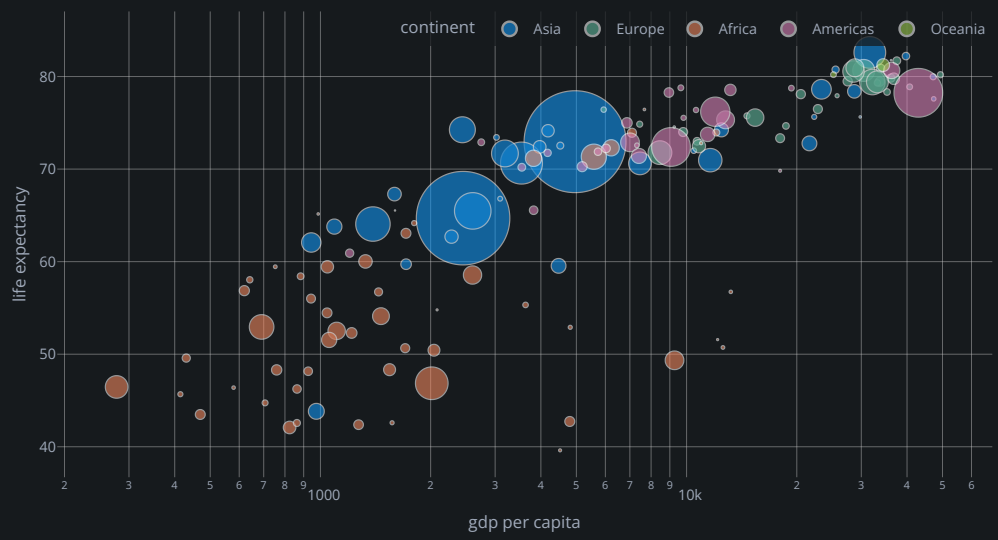
More about Visualization

The Dash Core Components module (`dash.dcc`) includes a component called `Graph` that renders interactive data visualizations using the open source **plotly.js** JavaScript graphing library. Plotly.js supports over 35 chart types and renders charts in both vector-quality SVG and high-performance WebGL. Check out the **plotly.py documentation and gallery** to learn more.

Here's an example that creates a scatter plot from a Pandas dataframe. Create a file named `app.py` with the following code:

```
# Run this app with `python app.py` and
# visit http://127.0.0.1:8050/ in your web browser.


from dash import Dash, dcc, html
import plotly.express as px
import pandas as pd
```

```
app = Dash()

df = pd.read_csv('https://gist.githubusercontent.com/chriddyp/5d1ea79569ed194d432e56108a04d188/

fig = px.scatter(df, x="gdp per capita", y="life expectancy",
                 size="population", color="continent", hover_name="country",
                 log_x=True, size_max=60)

app.layout = html.Div([
    dcc.Graph(
        id='life-exp-vs-gdp',
        figure=fig
    )
])

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



These graphs are interactive and responsive. ***Hover*** *over points to see their values,* ***click*** *on legend items to toggle traces,* ***click and drag*** *to zoom,* ***hold down shift, and click and drag*** *to pan.*

Markdown

While Dash exposes HTML through Dash HTML Components (`dash.html`), it can be tedious to write your copy in HTML. For writing blocks of text, you can use the `Markdown` component in Dash Core Components (`dash.dcc`). Create a file named `app.py` with the following code:

```
# Run this app with `python app.py` and
# visit http://127.0.0.1:8050/ in your web browser.


from dash import Dash, html, dcc

app = Dash()

markdown_text = '''
### Dash and Markdown

Dash apps can be written in Markdown.
Dash uses the [CommonMark](http://commonmark.org/)
specification of Markdown.
Check out their [60 Second Markdown Tutorial](http://commonmark.org/help/)
if this is your first introduction to Markdown!
'''

app.layout = html.Div([
    dcc.Markdown(children=markdown_text)
])
```

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

Dash and Markdown

Dash apps can be written in Markdown. Dash uses the **CommonMark** specification of Markdown. Check out their **60 Second Markdown Tutorial** if this is your first introduction to Markdown!

Core Components

Dash Core Components (`dash.dcc`) includes a set of higher-level components like dropdowns, graphs, markdown blocks, and more.

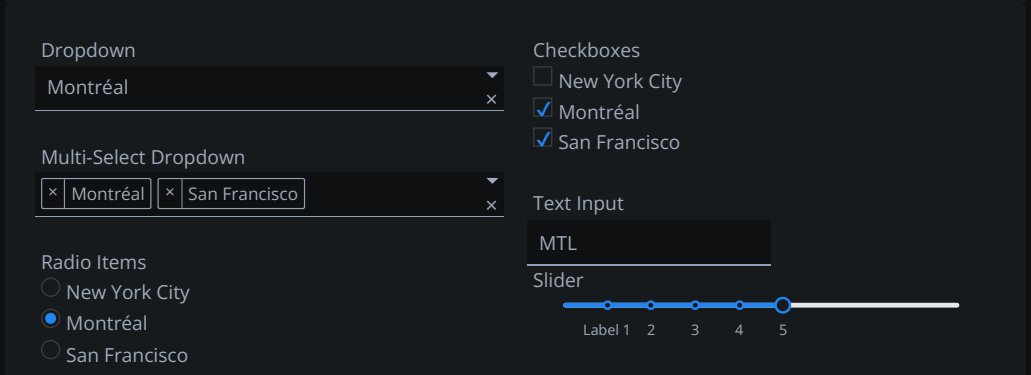
Like all Dash components, they are described entirely declaratively. Every option that is configurable is available as a keyword argument of the component.

We'll see many of these components throughout the tutorial. You can view all of the available components in the **Dash Core Components overview**.

Here are a few of the available components. Create a file named `app.py` with the following code:

```
# Run this app with `python app.py` and
# visit http://127.0.0.1:8050/ in your web browser.


from dash import Dash, html, dcc

app = Dash()

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

        html.Br(),
        html.Label('Multi-Select Dropdown'),
        dcc.Dropdown(['New York City', 'Montréal', 'San Francisco'],
                     ['Montréal', 'San Francisco'],
                     multi=True),

        html.Br(),
        html.Label('Radio Items'),
        dcc.RadioItems(['New York City', 'Montréal', 'San Francisco'], 'Montréal'),
    ], style={'padding': 10, 'flex': 1}),

    html.Div(children=[
        html.Label('Checkboxes'),
        dcc.Checklist(['New York City', 'Montréal', 'San Francisco'],
                      ['Montréal', 'San Francisco']
        ),

        html.Br(),
        html.Label('Text Input'),
        dcc.Input(value='MTL', type='text'),

        html.Br(),
        html.Label('Slider'),
        dcc.Slider(
            min=0,
            max=9,
            marks={i: f'Label {i}' if i == 1 else str(i) for i in range(1, 6)},
            value=5,
        ),
    ], style={'padding': 10, 'flex': 1})
], style={'display': 'flex', 'flexDirection': 'row'})
```

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



Help

Dash components are declarative: every configurable aspect of these components is set during instantiation as a keyword argument.

Call `help` in your Python console on any of the components to learn more about a component and its available arguments.

```
>>> help(dcc.Dropdown)
class Dropdown(dash.development.base_component.Component)
|  A Dropdown component.
|  Dropdown is an interactive dropdown element for selecting one or more
|  items.
|  The values and labels of the dropdown items are specified in the `options`
|  property and the selected item(s) are specified with the `value` property.
|
|  Use a dropdown when you have many options (more than 5) or when you are
|  constrained for space. Otherwise, you can use RadioItems or a Checklist,
|  which have the benefit of showing the users all of the items at once.
|
|  Keyword arguments:
|  - id (string; optional)
|  - className (string; optional)
|  - disabled (boolean; optional): If true, the option is disabled
|  - multi (boolean; optional): If true, the user can select multiple values
|  - options (list; optional)
|  - placeholder (string; optional): The grey, default text shown when no option is selected
|  - value (string | list; optional): The value of the input. If `multi` is false (the default)
|  then value is just a string that corresponds to the values
|  provided in the `options` property. If `multi` is true, then
|  multiple values can be selected at once, and `value` is an
|  array of items with values corresponding to those in the
|  `options` prop.
```

Summary

The `layout` of a Dash app describes what the app looks like. The `layout` is a hierarchical tree of components, or a list of components (in Dash 2.17 and later).

Dash HTML Components (`dash.html`) provides classes for all of the HTML tags and the keyword arguments describe the HTML attributes like `style`, `class`, and `id`. Dash Core Components (`dash.dcc`) generates higher-level components like controls and graphs.

For reference, see:

- **Dash Core Components overview**
- **Dash HTML Components overview**

The next part of the Dash Fundamentals covers how to make these apps interactive. **Dash Fundamentals Part 2: Basic Callbacks**