plotly

Star 23,447

Dash Python > ***Live Updates***

Live Updating Components

The `dcc.Interval` Component

Components in Dash usually update through user interaction like selecting a dropdown, dragging a slider, or hovering over points.

If you're building an application for monitoring, you may want to update components in your application every few seconds or minutes.

The `dcc.Interval` element allows you to update components on a predefined interval. The `n_intervals` property is an integer that is automatically incremented every time `interval` milliseconds pass. You can listen to this variable inside your app's `callback` to fire the callback on a predefined interval.

This example pulls data from live satellite feeds and updates the graph and the text every second.

```
import datetime

import dash
from dash import Dash, dcc, html, Input, Output, callback
import plotly

# pip install pyorbital
from pyorbital.orbital import Orbital
satellite = Orbital('TERRA')

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

app = Dash(__name__, external_stylesheets=external_stylesheets)
app.layout = html.Div(
    html.Div([
        html.H4('TERRA Satellite Live Feed'),
        html.Div(id='live-update-text'),
        dcc.Graph(id='live-update-graph'),
        dcc.Interval(
            id='interval-component',
            interval=1*1000, # in milliseconds
            n_intervals=0
        )
    ])
)


@callback(Output('live-update-text', 'children'),
              Input('interval-component', 'n_intervals'))
def update_metrics(n):
    lon, lat, alt = satellite.get_lonlatalt(datetime.datetime.now())
    style = {'padding': '5px', 'fontSize': '16px'}
    return [
        html.Span('Longitude: {0:.2f}'.format(lon), style=style),
        html.Span('Latitude: {0:.2f}'.format(lat), style=style),
        html.Span('Altitude: {0:0.2f}'.format(alt), style=style)
    ]


# Multiple components can update everytime interval gets fired.
@callback(Output('live-update-graph', 'figure'),
```

```
              Input('interval-component', 'n_intervals'))
def update_graph_live(n):
    satellite = Orbital('TERRA')
    data = {
        'time': [],
        'Latitude': [],
        'Longitude': [],
        'Altitude': []
    }

    # Collect some data
```


Updates on Page Load

By default, Dash apps store the `app.layout` in memory. This ensures that the `layout` is only computed once, when the app starts.

If you set `app.layout` to a function, then you can serve a dynamic layout on every page load.

For example, if your `app.layout` looked like this:

```
import datetime

import dash
from dash import html

app.layout = html.H1('The time is: ' + str(datetime.datetime.now()))

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

then your app would display the time when the app was started.

If you change this to a function, then a new `datetime` will get computed everytime you refresh the page. Give it a try:

```
import datetime

import dash
from dash import html

def serve_layout():
    return html.H1('The time is: ' + str(datetime.datetime.now()))

app.layout = serve_layout

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

> **Heads up! You need to write `app.layout = serve_layout` *not* `app.layout = serve_layout()`. That is, define `app.layout` to the actual function instance.**

You can combine this with **time-expiring caching** and serve a unique `layout` every hour or every day and serve the computed `layout` from memory in between.

Dash Python ***> Live Updates***