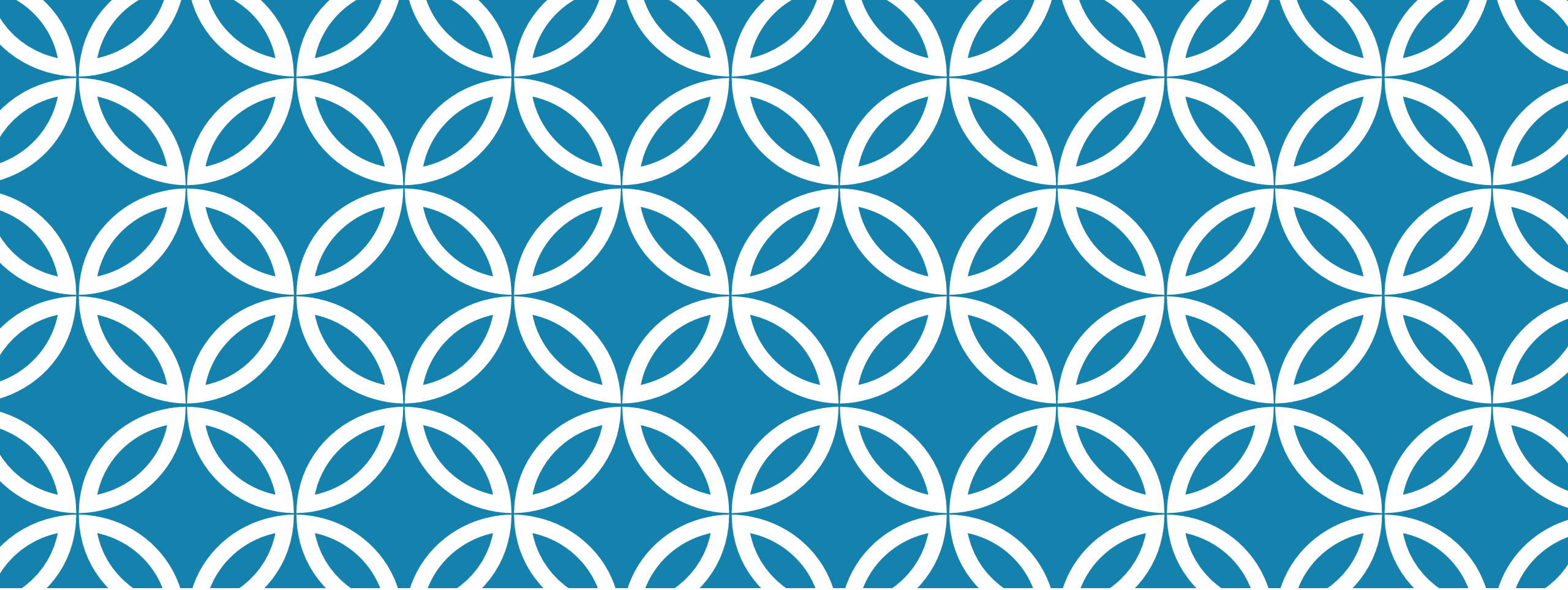


HCDE 530 Vis Lecture 2: Building Interactive Web Visualization in Python via Plotly/Dash

Nan-Chen Chen

Feb 15, 2018

In last week's lecture, we have learned...

Basic Setup and Usage of Dash/Plotly

Basic Visualization Concepts

Building a variety of charts using Dash/Plotly

- Barchart
- Linechart
- Small multiples
- Heatmap
- Scatter Plot

This week's lecture

More hands-on exercises!

Outline of today's lecture

Interactivity in Dash (10 mins)

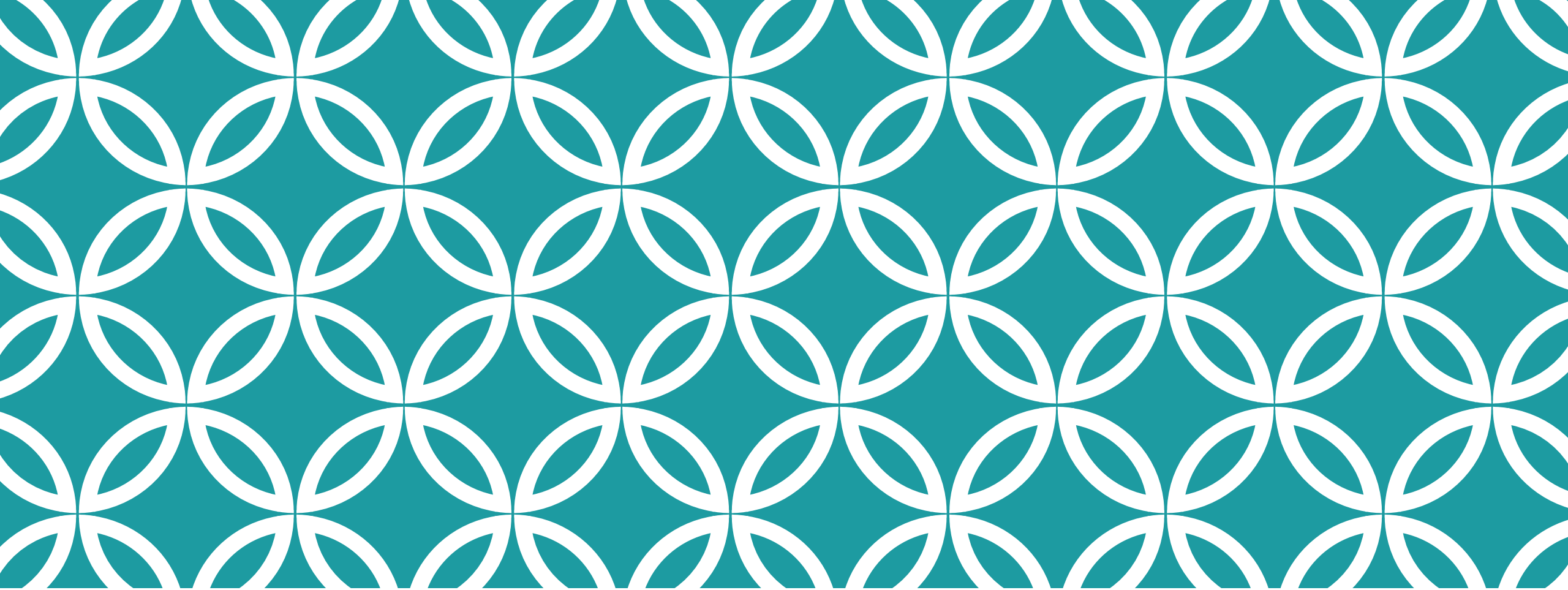
Building a barchart with data from a real time query (40 mins)

Triggering changes of one plot from another plot (5 mins)

Break (5 mins)

Building your own visualization using your mock data! (40 mins)

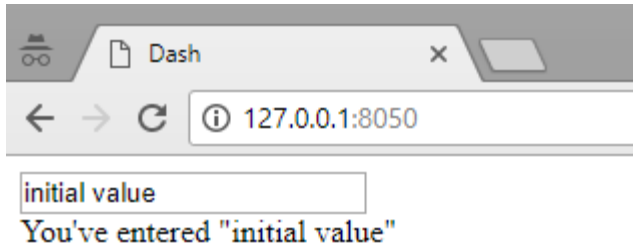
Wrap Up

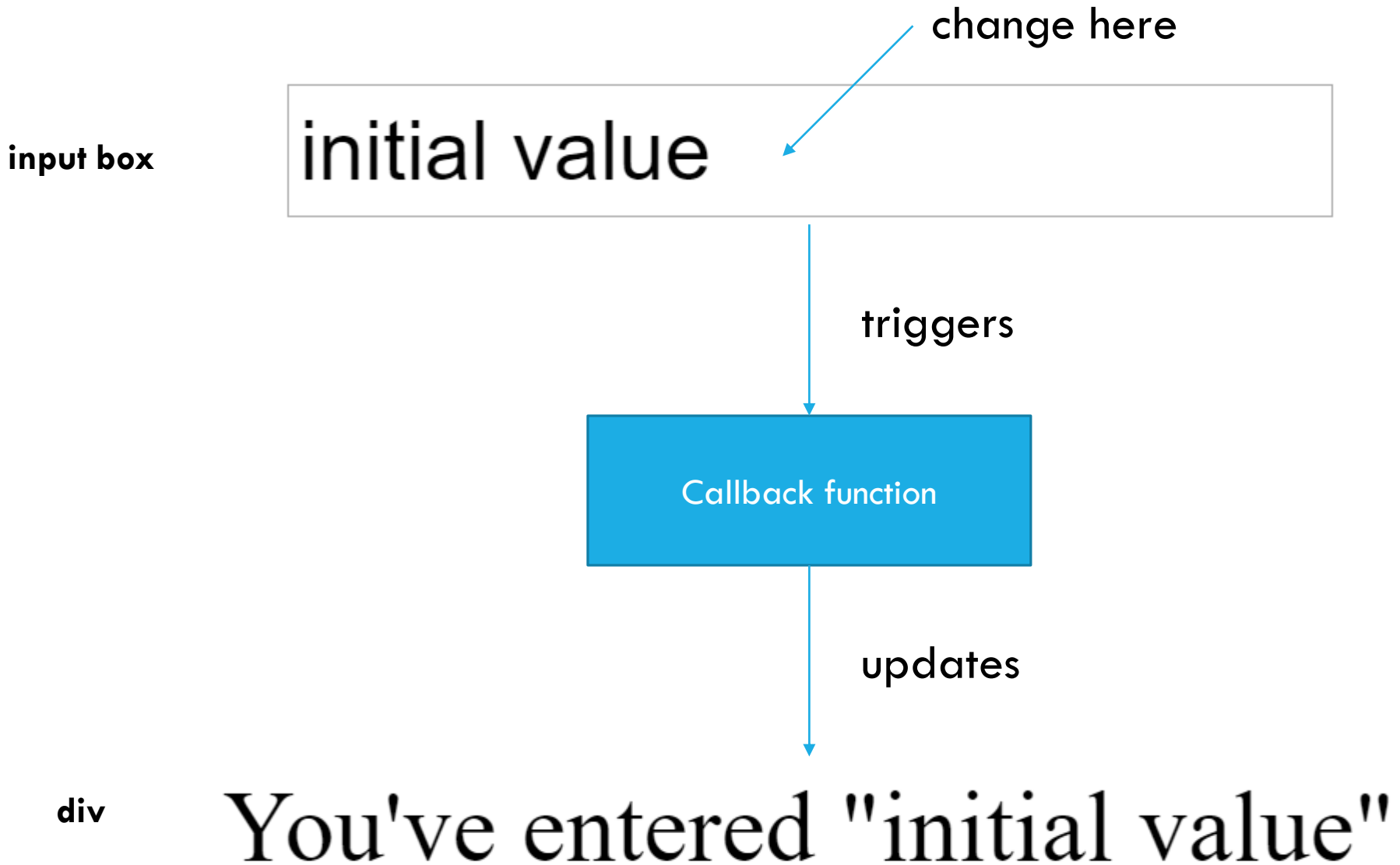


Interactivity in Dash

A super simple example of interactivity in Dash

01_dash-input-demo.py

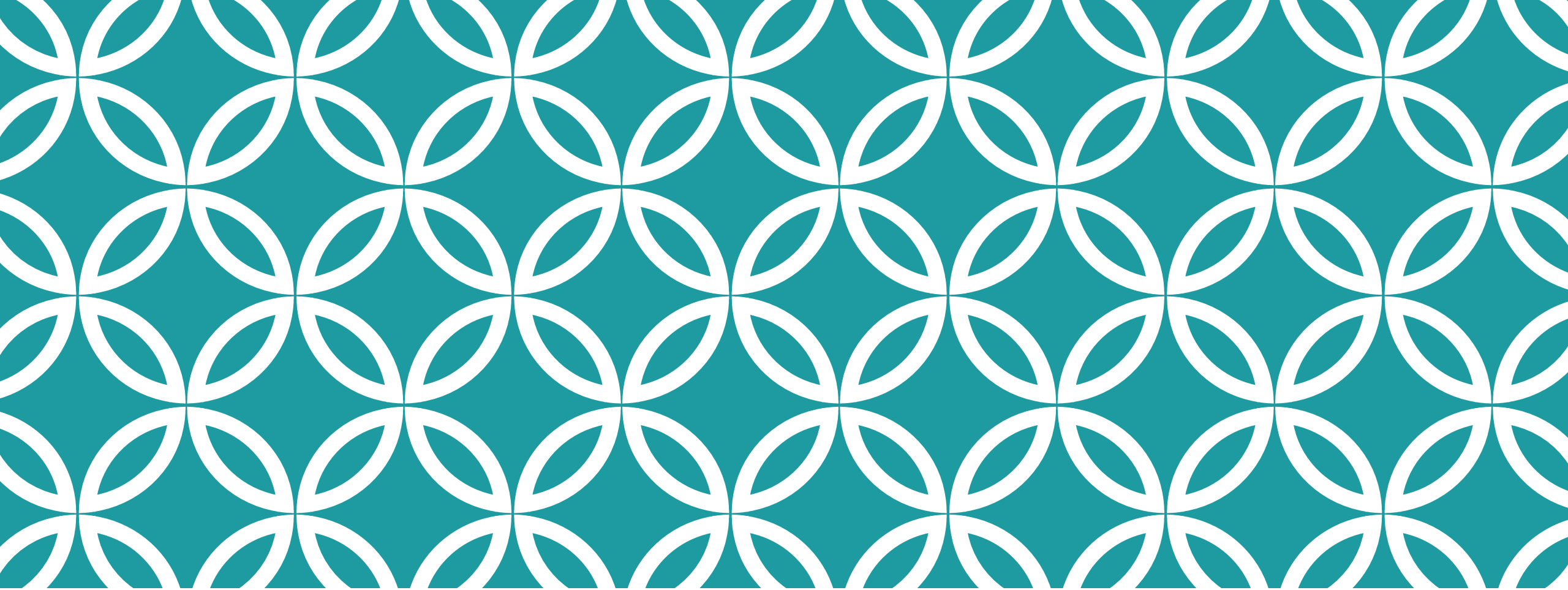



change here
input box
initial value
triggers
Callback function
updates
div
You've entered "initial value"

```python
import dash
from dash.dependencies import Input, Output
import dash_core_components as dcc
import dash_html_components as html

# initialize Dash app
app = dash.Dash()

# set the layout to have an input box and a div
app.layout = html.Div([
    dcc.Input(id='my-id', value='initial value', type='text'),
    html.Div(id='my-div')
])

# define callback to connect the input value with the content of the div
@app.callback(
    Output(component_id='my-div', component_property='children'),
    [Input(component_id='my-id', component_property='value')]
)
def update_output_div(input_value):
    return 'You\'ve entered "{}"'.format(input_value)

# start the app
if __name__ == '__main__':
    app.run_server()
```

input value

The updated content for the div

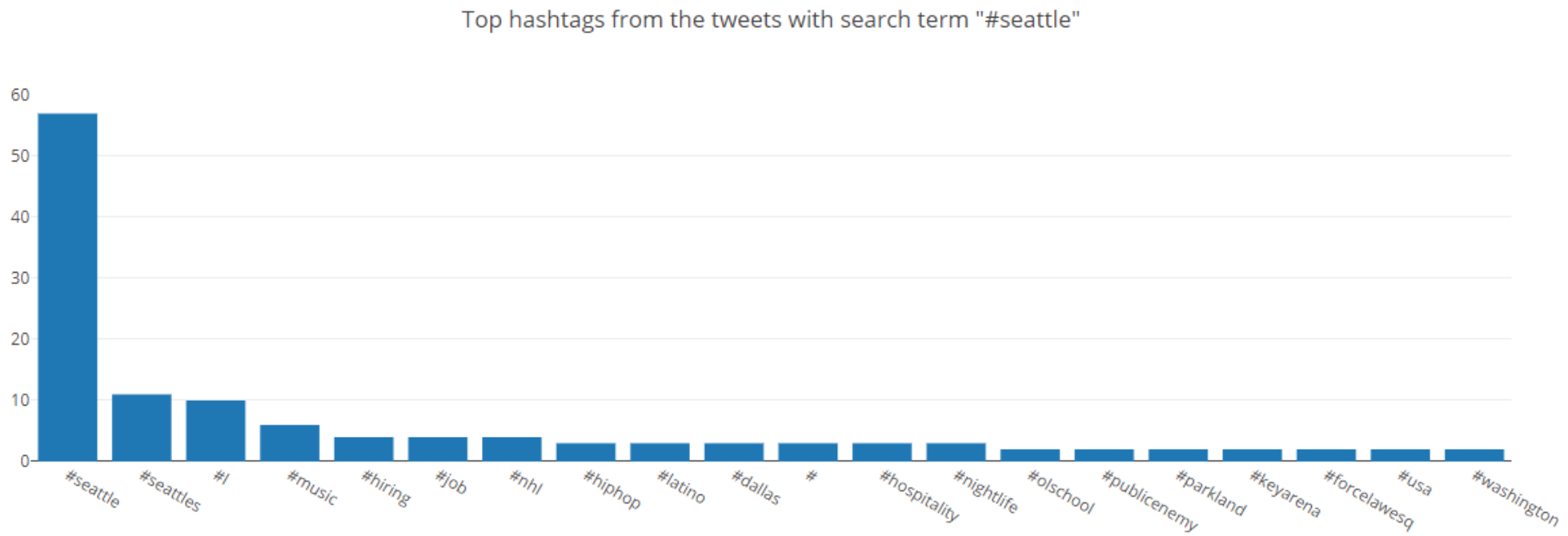


Building a barchart with data from a real time query

Demo!

Input box + Barchart

Enter a search term: #seattle



02_dash-search-barchart.py

A skeleton code for us to fill together!

Part 1: Based on Quiz 5

```
15  def simple_parse_hashtags(tweet=""):
16      # TODO: paste your hash tag parse function here
17
18  def get_hashtags(search_terms):
19      # TODO: fill your code based on the Quiz 5
20      #       but unlike the quiz 5, we want the return value to be a dictionary containing hashtags & counts
21      #       like:
22      #       {
23      #           "#Oscar2016": 1697,
24      #           "#Oscars": 19825,
25      #           "#ICD": 1393,
26      #           "#KCA": 1768,
27      #           "#oscars": 1688,
28      #           "#TBT": 1876,
29      #           "#Spotlight": 1148,
30      #           "#Oscars2016": 6312,
31      #           "#Oscar": 4629,
32      #           "#SpotlightMovie": 7592,
33      #           "#OscarsSoWhite": 1464
34      #       }
```

Part 2: Define the callback function

```
56  @app.callback(
57      Output(component_id='<TODO>', component_property='figure'),
58      [Input(component_id='<TODO>', component_property='value')]
59  )
60  def get_data(search_terms):
61      # TODO - Step 1: call get_hashtags() with the search_terms to get data
62      data = get_hashtags(search_terms)
63
64      # TODO - Step 2: convert the dictionary to two lists
65      #          (see the 02_dash-barchart-exercise.py example we had last week)
66
67      # TODO - Step 2.1: get x labels in order
68
69      # TODO - Step 2.2: get corresponding counts in order
70
71      # TODO - Step 3: return what we had in figure (a dictionary) in the 02_dash-barchart-exercise.py
72      #                but change the title to be
73      #                'Top hashtags from the tweets with search term "' + search_terms + '"'
74      }
```

Part 2 – step 0: Define input & output components

```
41  app.layout = html.Div(children=[
42      # H1 title on the page
43      html.H1(children='Input box + Barchart'),
44
45      # a div to put a short description
46      html.Label(children='Enter a search term:'),
47
48      dcc.Input(id='search-term', value='#seattle', type='text'),
49
50      # append the visualization to the page
51      dcc.Graph(
52          id='barchart'
53      )
54  ])
55
56  @app.callback(
57      Output(component_id='<TODO>', component_property='figure'),
58      [Input(component_id='<TODO>', component_property='value')]
59  )
```

?

?

Part 2 – step 1: call get_hashtags() with the search_terms to get data

```
60  def get_data(search_terms):
61      # TODO - Step 1: call get_hashtags() with the search_terms to get data
62
```

Part 2 – step 2:
convert the dictionary to two lists

```
# TODO - Step 2: convert the dictionary to two lists
#         (see the 02_dash-barchart-exercise.py example we had last week)

# TODO - Step 2.1: get x labels in order

# TODO - Step 2.2: get corresponding counts in order
```

Part 2 – step 3: return a figure config dictionary

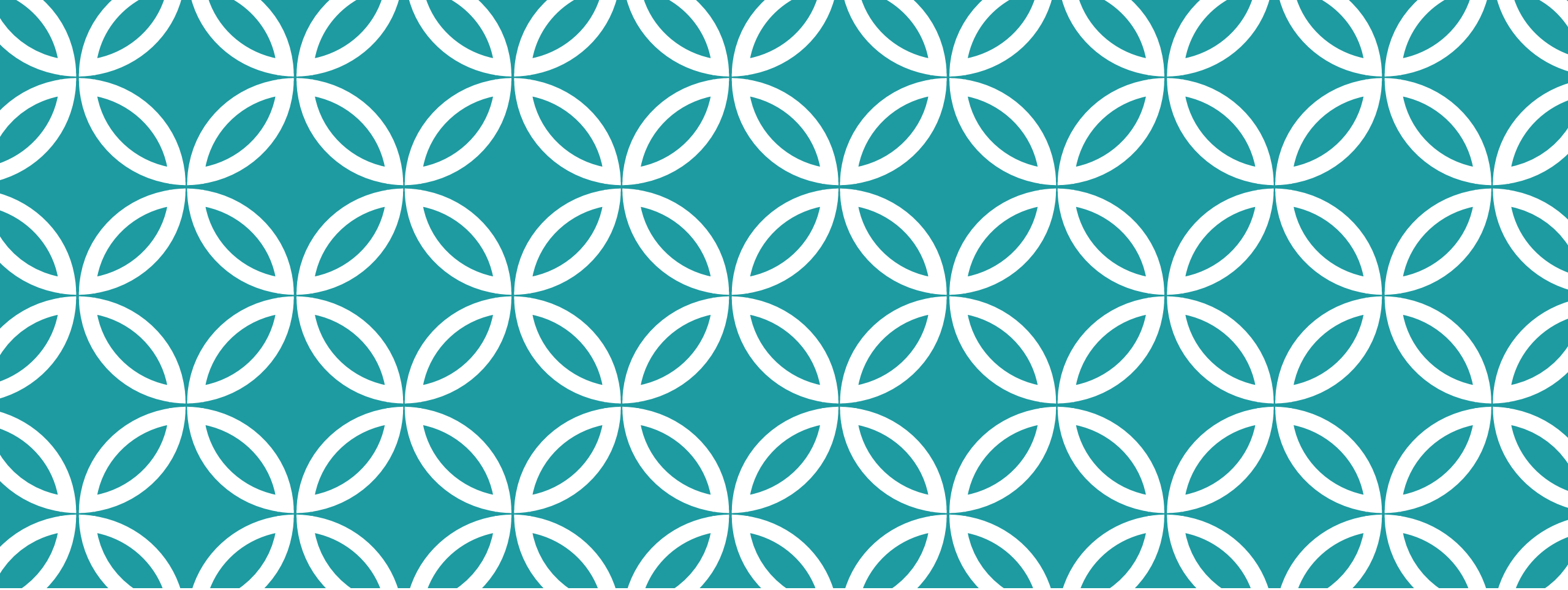
```
70      # TODO - Step 3: return what we had in figure (a dictionary) in the 02_dash-barchart-exercise.py
71      #                but change the title to be
72      #                'Top hashtags from the tweets with search term "' + search_terms + '"'
```

Additional step: remove punctuation & lowercase

```
14  def remove_puncutuation(token):
15      return re.sub(r'[^\w\s^#]','', token)
16
17  def simple_parse_hashtags(tweet=""):
18      hash_tags = []
19      if tweet:
20          token_list = tweet.split()
21          for token in token_list:
22              if token.startswith('#'):
23                  token = token.lower()
24                  token = remove_puncutuation(token)
25                  if len(token) > 0:
26                      hash_tags.append(token)
27      return hash_tags
```

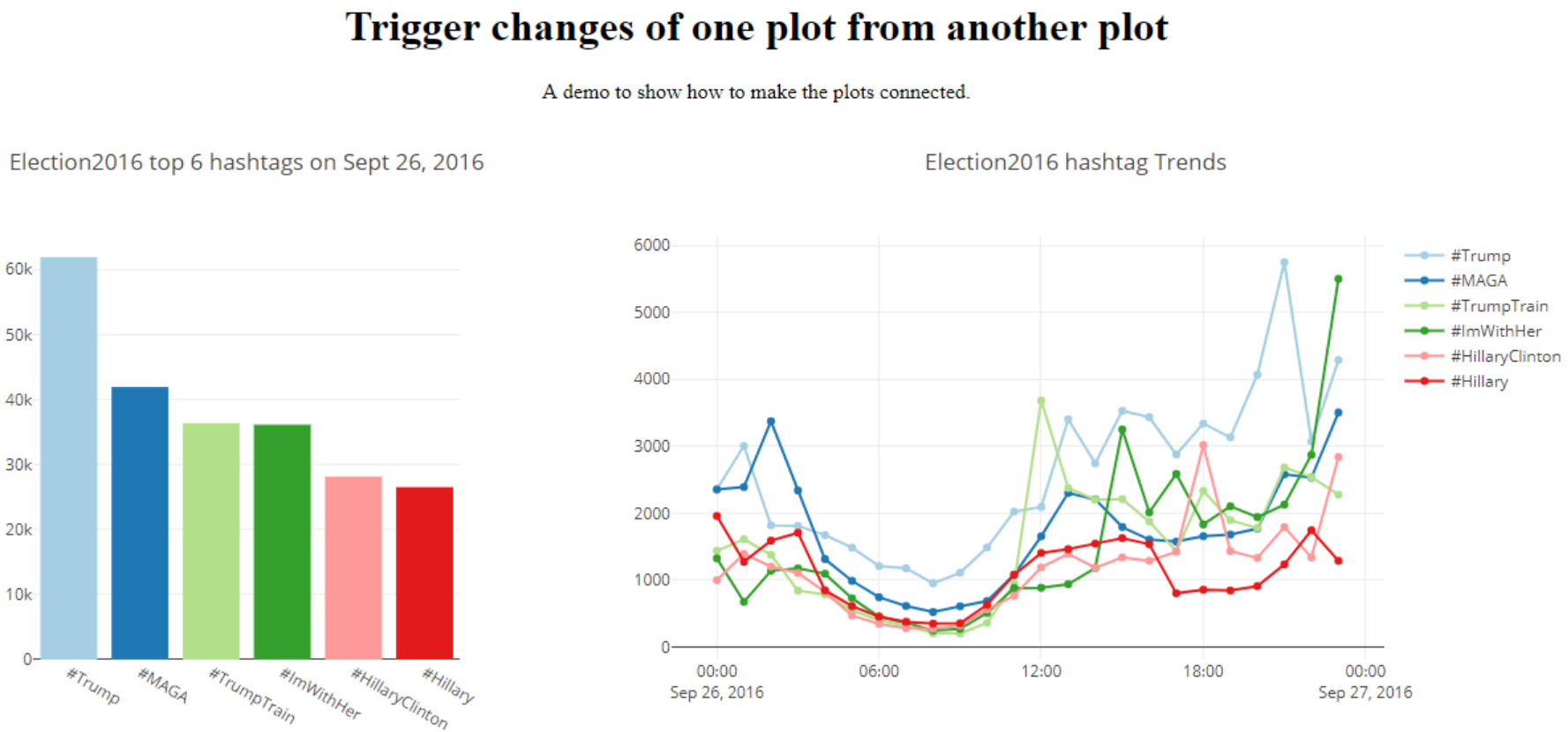
Additional step: only show top 20 hashtags

```
105     hashtags_in_order = hashtags_in_order[:20]
```



Triggering changes of one plot from another plot

03_dash-election2016-interaction.py



Defining two plots

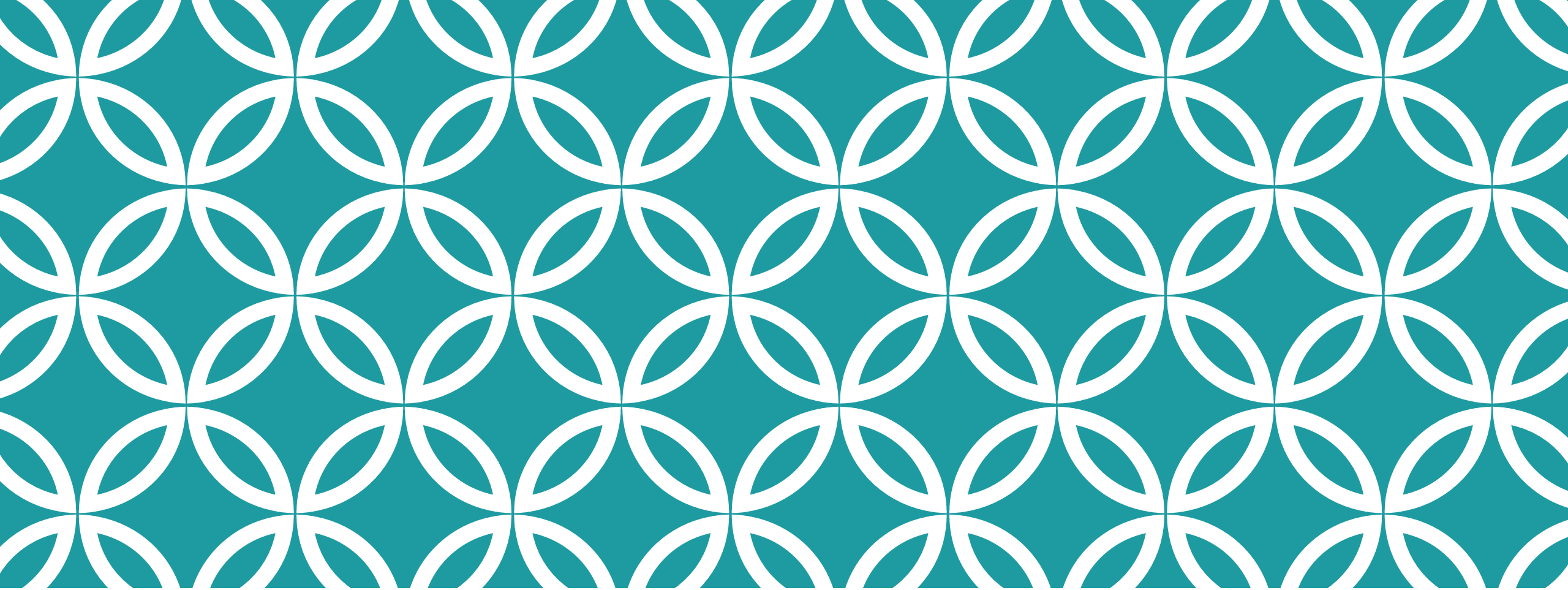
```
46          # the first graph is a barchart
47          dcc.Graph(
48              id='top-hashtag-barchart',
49              figure={
50                  # configure the data
51                  'data': [{
52                          # set x to be hashtags, and y to be the counts. We use bars to represent our data
53                          'x': top_hashtags,
54                          'y': [sum(data["trends"][hashtag]) for hashtag in top_hashtags],
55                          'type': 'bar',
56                          'marker': {
57                              'color': colors
58                          }
59                      },
60                  ],
61                  # configure the layout of the visualization -- set the title to be Election2016 hashtag Trends
62                  'layout': {
63                      'title': 'Election2016 top %s hashtags on Sept 26, 2016' % top
64                  },
65              },
66              clickData={"points": []},
67              config={'displayModeBar': False},
68              style={
69                 'width': 500
70              }
71          ),
72
73
74          # the second graph is a time series
75          dcc.Graph(
76              id='trend-series',
77              config={'displayModeBar': False},
78              clickData={"points": []}, style={
79                 'width': 800
80              }
81          ),
```

Update the figure of trend-series whe n top-hashtag-barchart is clicked

```
91  # define interaction: when click on the bar chart, the highlighting of the time series will be changed
92  @app.callback(
93      dash.dependencies.Output('trend-series', 'figure'),
94      [dash.dependencies.Input('top-hashtag-barchart', 'clickData')])
```

Update the line chart's data

```
95  def update_graph(click_data):
96      print click_data
97      series = []
98
99      for point in click_data["points"]:
100         if point["x"] not in highlighted_hashtags:
101             highlighted_hashtags.add(point["x"])
102         elif point["x"] in highlighted_hashtags:
103             highlighted_hashtags.remove(point["x"])
104
105     for idx, hashtag in enumerate(top_hashtags):
106         if len(highlighted_hashtags) == 0 or hashtag in highlighted_hashtags:
107             series.append(
108                 go.Scatter(
109                     x=[datetime(2016, 9, 26, h) for h in range(24)],
110                     y=data["trends"][hashtag],
111                     mode='lines+markers',
112                     name=hashtag,
113                     marker= {
114                         'color': hashtag_colors[hashtag]
115                     }
116                 )
117             )
118
119     return {
120             'data': series,
121             'layout': {
122                 'title': 'Election2016 hashtag Trends'
123             }
124         }
```



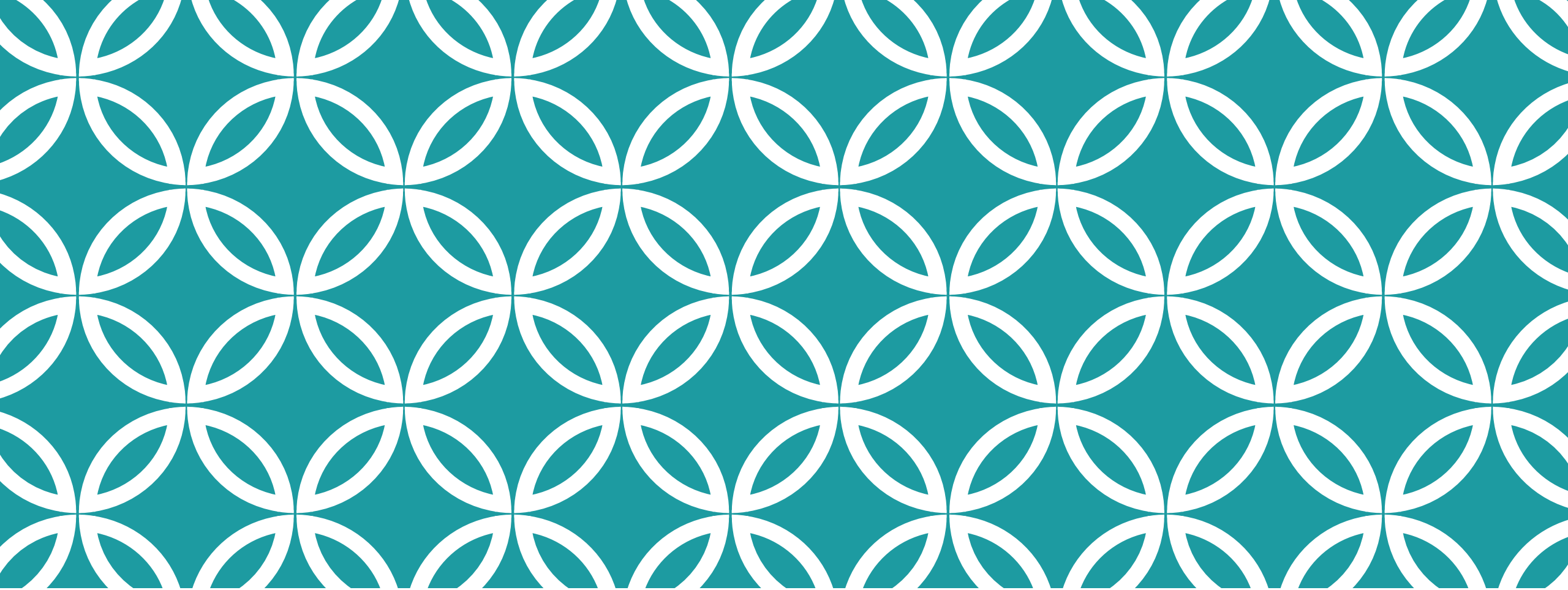
Building your own visualization using your mock data

If you don't have your own data...

1. Try to modify the codes from the last week but use one of the following two json files

- election2016_20160926_top30_hashtags_hourly_trends.json
- election2016_20161008_top30_hashtags_hourly_trends.json

2. Try to make the two plots in one page



Wrap Up...

In today's lecture, we have learned...

Basic interactivity in Dash

How to build a barchart with data from real time queries

A more complex example for interactivity

How to build visualization with your own data or to play with the examples we had

How to go further

https://community.plot.ly/

https://community.plot.ly/c/dash

Ask me questions!

Take HCDE 511 ☺ (no programming classes there though)

This is only a way to build interactive web visualization – it may not be useful to you at this point, but maybe one day you will need it!

Feedback Survey!

https://goo.gl/forms/gJLNhtmUKNHKgzuj2