iDash

Building interactive web apps in Dash

PyData Warsaw 2018

iDash

idash.pl

Who we are

Our training

- Machine learning,
- Artificial intelligence,
- Neural Networks,
- Bayesian inference,
- Statistical modelling,
- Efficient reporting,
- Web scraping,
- Data visualisation,
- Data related web app frameworks (Dash, Shiny),
- Relational and non-relational databases,
- Introductions to Data Science languages (Python, R, SQL).

and much more.

What's Dash?

What's Dash?

- Python framework for building **analytical** web apps.

What's Dash?

- Python framework for building **analytical** web apps.
- Allows to quickly share your results in an appealing way.

What's Dash?

- Python framework for building **analytical** web apps.
- Allows to quickly share your results in an appealing way.
- No other programming language required - just Python!

What's Dash?

- Python framework for building **analytical** web apps.
- Allows to quickly share your results in an appealing way.
- No other programming language required - just Python!
- Uses Plotly.js, React and Flask under the hood.

What's Dash?

- Python framework for building **analytical** web apps.
- Allows to quickly share your results in an appealing way.
- No other programming language required - just Python!
- Uses Plotly.js, React and Flask under the hood.
- It's **free**! (MIT License)

PLOTCON 2016: Chris Parmer, Dash: Shiny for Python

Dataset

Two CSV's:

- **flights.csv** - contains information about flights from three New York City airports in 2013. It stores data such as:
  - flight hour (**hour** column),
  - flight distance (**distance** column),
  - departure delay (**dep_delay** column)
  - etc.
- **airlines.csv** - contains airline short name (**carrier** column) and full name (**name** column).

App demo

Agenda

- Dash background

Agenda

- Dash background
- Static layout elements

Agenda

- Dash background
- Static layout elements
- Dynamic layout elements

Agenda

- Dash background
- Static layout elements
- Dynamic layout elements
- Making the output react to the input

Agenda

- Dash background
- Static layout elements
- Dynamic layout elements
- Making the output react to the input
- Basics of caching

Technical details

Python & packages

We recommend using Python 3. Packages required: `dash`, `dash_core_components`, `dash_html_components`, `dash_table`, `pandas`, `numpy`, `plotly`.

Materials

- Workshop slide deck **idash.pl/dash_intro**
- Project on **github repository**
- Final Dash app on **dash.idash.pl**

Let's get started!

Open the project!

App structure

app.py

App structure

```python
# Load packages
import dash
import dash_html_components as html

# Initialize the app
app = dash.Dash()

# Create a layout
app.layout = html.Div()

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

How to build the **layout**

Who of you know basics of HTML?

Who of you know basics of CSS?

Dash HTML Components

Dash HTML Components

The `dash_html_components` library contains functions for wrapping any content within given HTML tag.

Dash HTML Components

The `dash_html_components` library contains functions for wrapping any content within given HTML tag.

```python
import dash_html_components as html

html.H1(
  children = 'Hello!'
)
```

Hello!

Dash HTML Components

All HTML tags are available but you'll most likely end up using only few of them:

- **HTML:** `<div>Container</div>` **Dash:** `html.Div()`

Dash HTML Components

All HTML tags are available but you'll most likely end up using only few of them:

- **HTML:** `<div>Container</div>` **Dash:** `html.Div()`
- **HTML:** `<h1>First level heading</h1>` **Dash:** `html.H1()`

Dash HTML Components

All HTML tags are available but you'll most likely end up using only few of them:

- **HTML:** `<div>Container</div>` **Dash:** `html.Div()`
- **HTML:** `<h1>First level heading</h1>` **Dash:** `html.H1()`
- **HTML:** `<p>Paragraph</p>` **Dash:** `html.P()`

Dash HTML Components

All HTML tags are available but you'll most likely end up using only few of them:

- **HTML:** `<div>Container</div>` **Dash:** `html.Div()`
- **HTML:** `<h1>First level heading</h1>` **Dash:** `html.H1()`
- **HTML:** `<p>Paragraph</p>` **Dash:** `html.P()`
- **HTML:** `<br/>` - newline **Dash:** `html.Br()`

Dash HTML Components

All HTML tags are available but you'll most likely end up using only few of them:

- **HTML:** `<div>Container</div>` **Dash:** `html.Div()`
- **HTML:** `<h1>First level heading</h1>` **Dash:** `html.H1()`
- **HTML:** `<p>Paragraph</p>` **Dash:** `html.P()`
- **HTML:** `<br/>` - newline **Dash:** `html.Br()`

...at least on the beginning. Full list of HTML tags can be found here.

Dash HTML Components

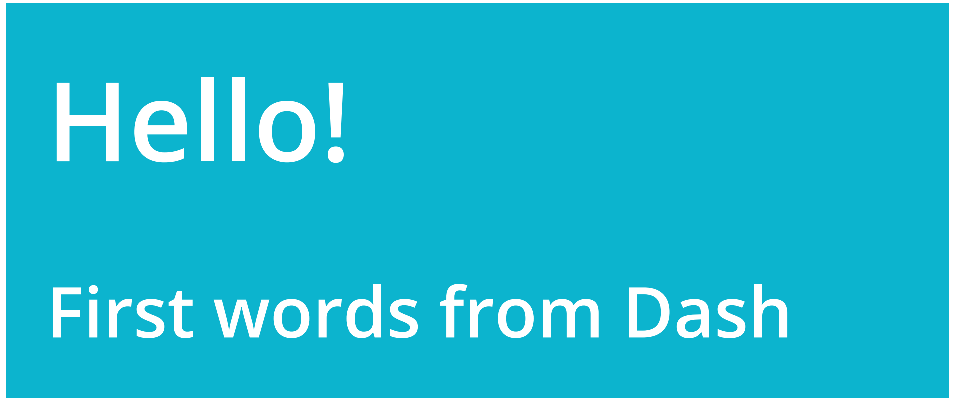
Tags can be **nested** and can contain **attributes** in a form of function arguments.

Dash HTML Components

Tags can be **nested** and can contain **attributes** in a form of function arguments.

```
import dash_html_components as html

html.Div(
  children = [
    html.H1('Hello!'),
    html.H3('First words from Dash')
  ],
  style = {
    'backgroundColor' : '#0cb4ce',
    'color' : '#fff'
  },
  className = 'myCssClass'
)
```



Hello!

First words from Dash

Dash HTML Components

Catches to remember:

- In Dash you need to **capitalize** the first letter of all HTML tag names.

Dash HTML Components

Catches to remember:

- In Dash you need to **capitalize** the first letter of all HTML tag names.
- The `class` attribute is called `className`.

Dash HTML Components

Catches to remember:

- In Dash you need to **capitalize** the first letter of all HTML tag names.
- The `class` attribute is called `className`.
- The `children` argument is always the first one, so it's name can be omitted.

Dash HTML Components

Catches to remember:

- In Dash you need to **capitalize** the first letter of all HTML tag names.
- The `class` attribute is called `className`.
- The `children` argument is always the first one, so it's name can be omitted.
- The `style` attribute takes the form of a Python dictionary where all keys are **camel cased**.

Dash Core Components

Dash Core Components

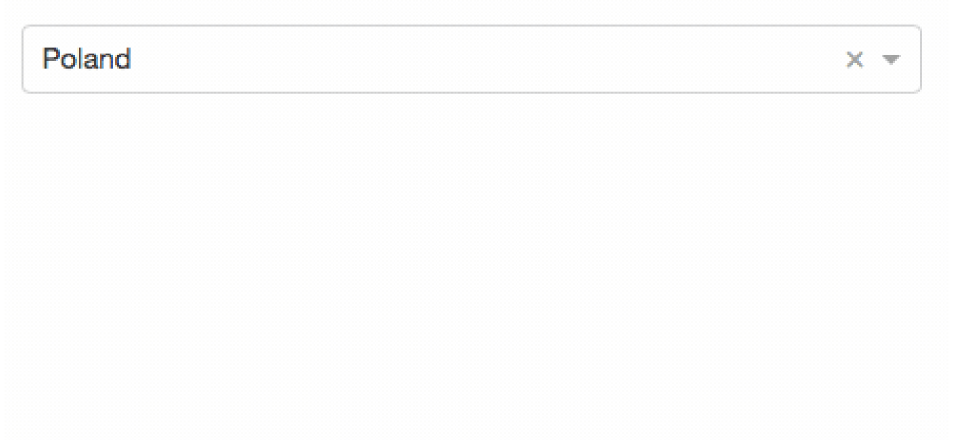
The `dash_core_components` library provides components allowing to build feature rich, interactive interfaces.

Dash Core Components

The `dash_core_components` library provides components allowing to build feature rich, interactive interfaces.

```
import dash_core_components as dcc

dcc.Dropdown(
  id="country",
  options=[
    {'label': 'Poland', 'value': 'PL'},
    {'label': 'Germany', 'value': 'DE'},
    {'label': 'United States', 'value': 'US'}
  ],
  value='PL'
)
```

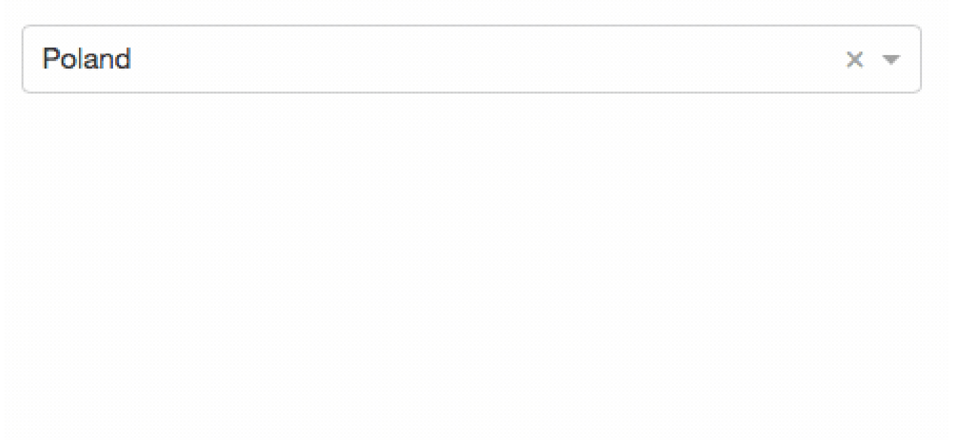


Dash Core Components

The `dash_core_components` library provides components allowing to build feature rich, interactive interfaces.

```
import dash_core_components as dcc

dcc.Dropdown(
  id="country",
  options=[
    {'label': 'Poland', 'value': 'PL'},
    {'label': 'Germany', 'value': 'DE'},
    {'label': 'United States', 'value': 'US'}
  ],
  value='PL'
)
```

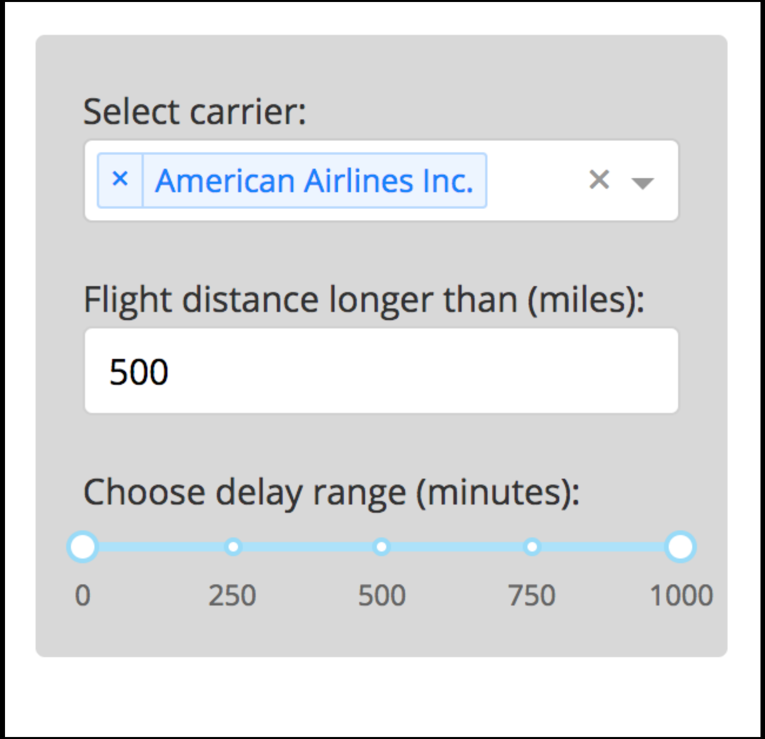


Full list of available components can be found here.

Task 1 (15 minutes)

Create new HTML layout element (`div`) containing the following components:

- A component which shows a list of airlines ("name" column in the "airlines" data frame). The list should enable choosing more than one airline at once. Use `ui-element` class.
- A component, which allows to determine a minimal flight distance. Use `ui-element` class.
- A component, in which user is able to choose a range from 0 to 1000. Try to add marks from 0 to 1000 with step 250.


Select carrier:

× American Airlines Inc.

Flight distance longer than (miles):

500

Choose delay range (minutes):

The div will get grey and positioned correctly if you add `sidebar`, `four` and `columns` CSS classes.

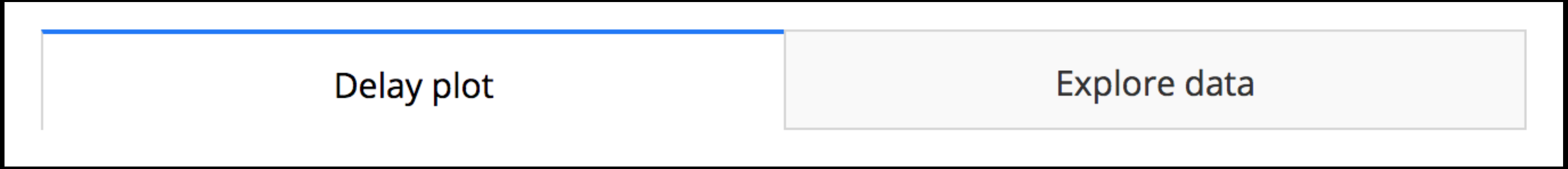
https://bit.ly/2DlpfYO

Task 2 (6 minutes)

One of the core components is a tabset. Try to add one on the right. To do this, create another HTML `div` element with `eight` and `columns` classes.

It should contain two tabs - `"Delay plot"` and `"Explore data"`:

- `Tabs` should have an id = `tabs`,
- Each `Tab` should have `custom-tab` class,
- Each `Tab` should have `custom-tab` class once is selected.



https://bit.ly/2TfTZAa

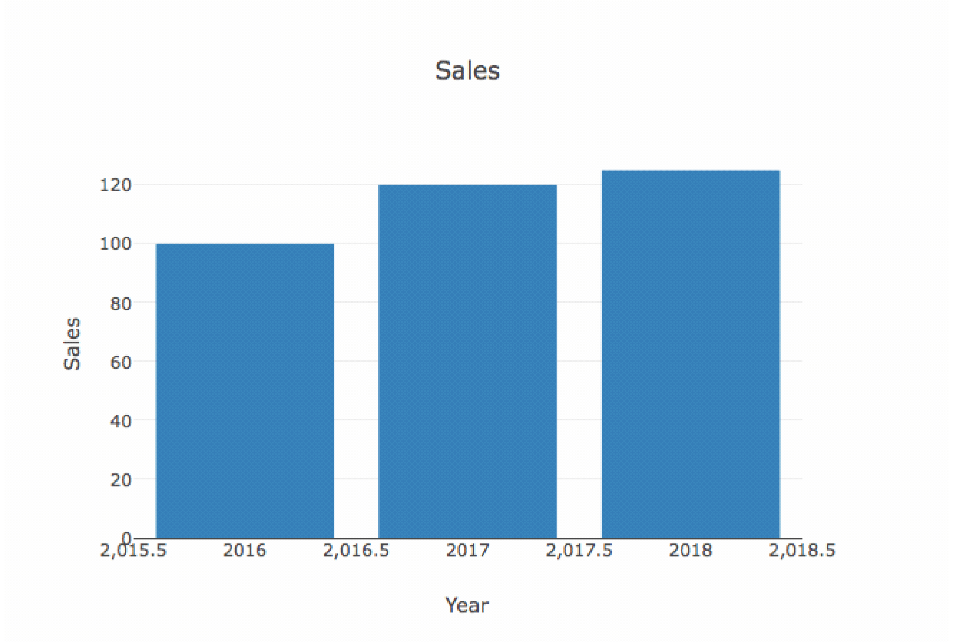
Plots & tables

Plots

The `dash-core-components` library contains a `Graph` component allowing to use plotly visualisations in the app.

```
import dash_core_components as dcc
import plotly.graph_objs as go

dcc.Graph(
  id="my-graph",
  figure=go.Figure(
    data=[
      go.Bar(
        x=[2016, 2017, 2018],
        y=[100, 120, 125],
      )
    ],
    layout=go.Layout(
      title='Sales',
      yaxis=dict(title='Sales'),
      xaxis=dict(title='Year')
    )
  )
)
```



Adding a static plot

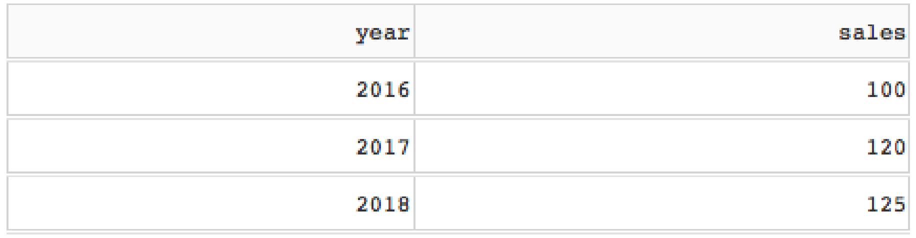
Tables

The `dash_table` library contains a component allowing to easily render tables.

```
import dash_table
import pandas as pd

data=pd.DataFrame({
  'year': [2016, 2017, 2018],
  'sales': [100, 120, 125]
})

dash_table.DataTable(
  id='table',
  columns=[
    {"name": i, "id": i} for i in data.columns
  ],
  data=data.to_dict("rows")
)
```

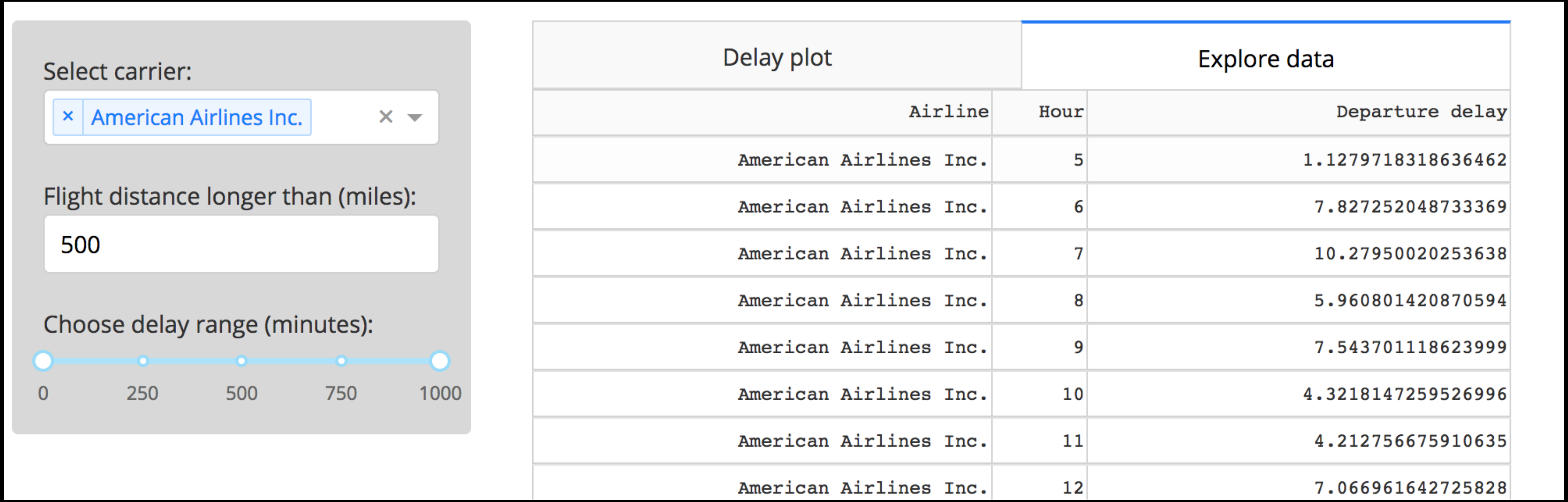


year	sales
2016	100
2017	120
2018	125

Task 3 (5 minutes)

Create a static table in the `"Explore data"` which presents the same aggregated data as shown in the plot. You can find a proper chunk of code in `static_elements.py` script.

Remember to include it in the `layout`.



Airline	Hour	Departure delay
American Airlines Inc.	5	1.1279718318636462
American Airlines Inc.	6	7.827252048733369
American Airlines Inc.	7	10.27950020253638
American Airlines Inc.	8	5.960801420870594
American Airlines Inc.	9	7.543701118623999
American Airlines Inc.	10	4.3218147259526996
American Airlines Inc.	11	4.212756675910635
American Airlines Inc.	12	7.066961642725828

https://bit.ly/2Bbx95P

Questions?

Making the app react to user input (callbacks)

What is reactivity?

Callbacks

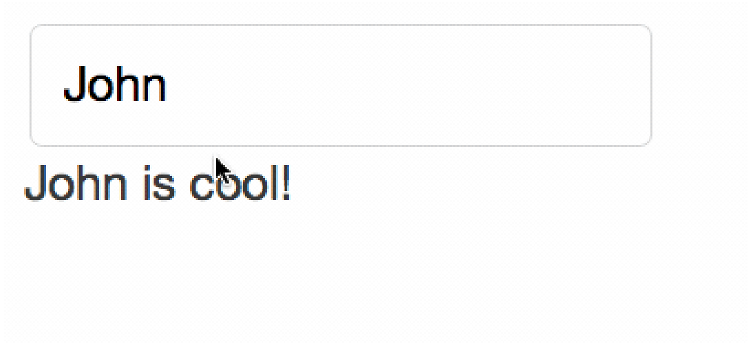
Callbacks **bind** components responsible for **inputing values** (like sliders, dropdowns etc.) with components responsible for presenting results (plots, tables, etc.).

Callbacks

Callbacks **bind** components responsible for **inputing values** (like sliders, dropdowns etc.) with components responsible for presenting results (plots, tables, etc.).

```
app.layout = html.Div([
  dcc.Input(id='name', value='John', type='text'),
  html.Div(id='sentence')
])

@app.callback(
  dash.dependencies.Output(
    component_id='sentence',
    component_property='children'
  ),
  [dash.dependencies.Input(
    component_id='name',
    component_property='value'
  )]
)
def update_output_div(input_value):
  return f'{input_value} is cool!'
```


John

John is cool!

Building text object with input/s in our app

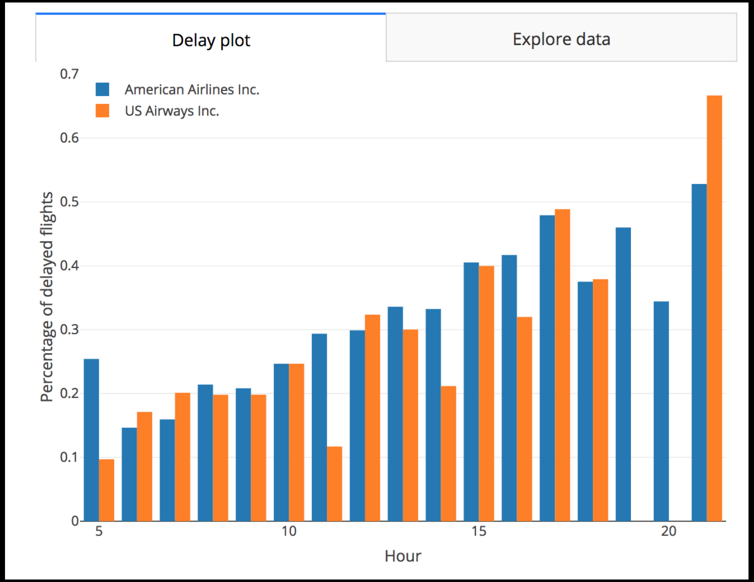
Task 4 (10 minutes)

Let's make a reactive plot! Use `callback`; it should response to any changes made in control panel. Use the code from `static_element.py`

1. Based on `static_sum_flights` function, create a `reactive_sum_flights` function including all dependencies (core components) to modify `agg_data` DataFrame.

2. The plot should show the percentage of delayed flights per hour in terms of airlines.

3. *Make the table react to user input as well.



Task 4 (10 minutes)

Hint How to calculate percentage of delayed flights https://bit.ly/2Ptw8Pc

What do you think can be improved in the app code?

Sharing data between callbacks (caching)

Global variables

How about making a callback so it modifies a global variable?

Global variables

How about making a callback so it modifies a global variable?

This will break your app!

Global variables

How about making a callback so it modifies a global variable?

This will break your app!

By default Dash serves the app to multiple users at once using a single Python process. If you modify a global variable **other user sessions might be influenced**.

Global variables

How about making a callback so it modifies a global variable?

This will break your app!

By default Dash serves the app to multiple users at once using a single Python process. If you modify a global variable **other user sessions might be influenced**.

Dash can also be run using multiple Python workers that executes callbacks on parallel. However, worker memory is not shared. If you change a global variable in one callback, workers handling other callbacks won't see that change!

Sharing data with a hidden div

- There are many ways caching can be accomplished in Dash.

Sharing data with a hidden div

- There are many ways caching can be accomplished in Dash.
- The basic one is just **binding the results of one callback to a hidden HTML element** and making other callbacks react on change of the value of this hidden element.

Sharing data with a hidden div

- There are many ways caching can be accomplished in Dash.
- The basic one is just **binding the results of one callback to a hidden HTML element** and making other callbacks react on change of the value of this hidden element.

Link to the example

Sharing data with a hidden div

```
### app.layout ###
html.Div(id='data', style={'display': 'none'})
```

Sharing data with a hidden div

```
### app.layout ###
html.Div(id='data', style={'display': 'none'})
```

```
### First callback ###
@app.callback(
    dash.dependencies.Output('data', 'children')
   [Input(...)]
)
def foo_fun():
   df = ...
    return df.to_json(date_format = 'iso', orient = 'split')
```

Sharing data with a hidden div

```
### app.layout ###
html.Div(id='data', style={'display': 'none'})
```

```
### First callback ###
@app.callback(
    dash.dependencies.Output('data', 'children')
   [Input(...)]
)
def foo_fun():
   df = ...
    return df.to_json(date_format = 'iso', orient = 'split')
```

Then, you can use `pd.read_json()` to retrieve the data in another callback.

Task 5* (7 minutes)

Modify `callbacks` in order to remove duplicated code.

https://bit.ly/2QLjJTm

We're looking for trainers!

If you need training, you know who to contact! ;)

iDash

mb@idash.pl mo@idash.pl