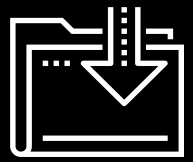
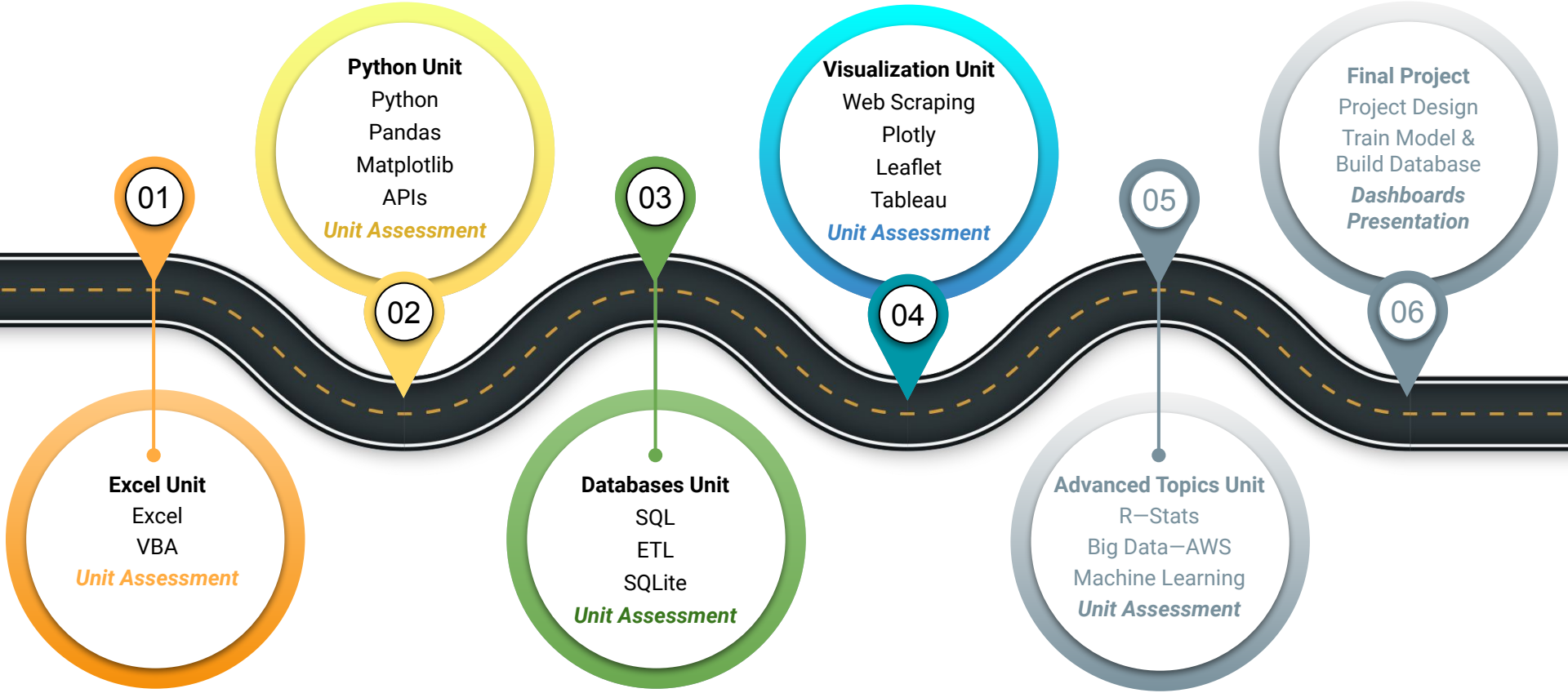
Plotly.js

Data Boot Camp

Lesson 12.2



The Big Picture

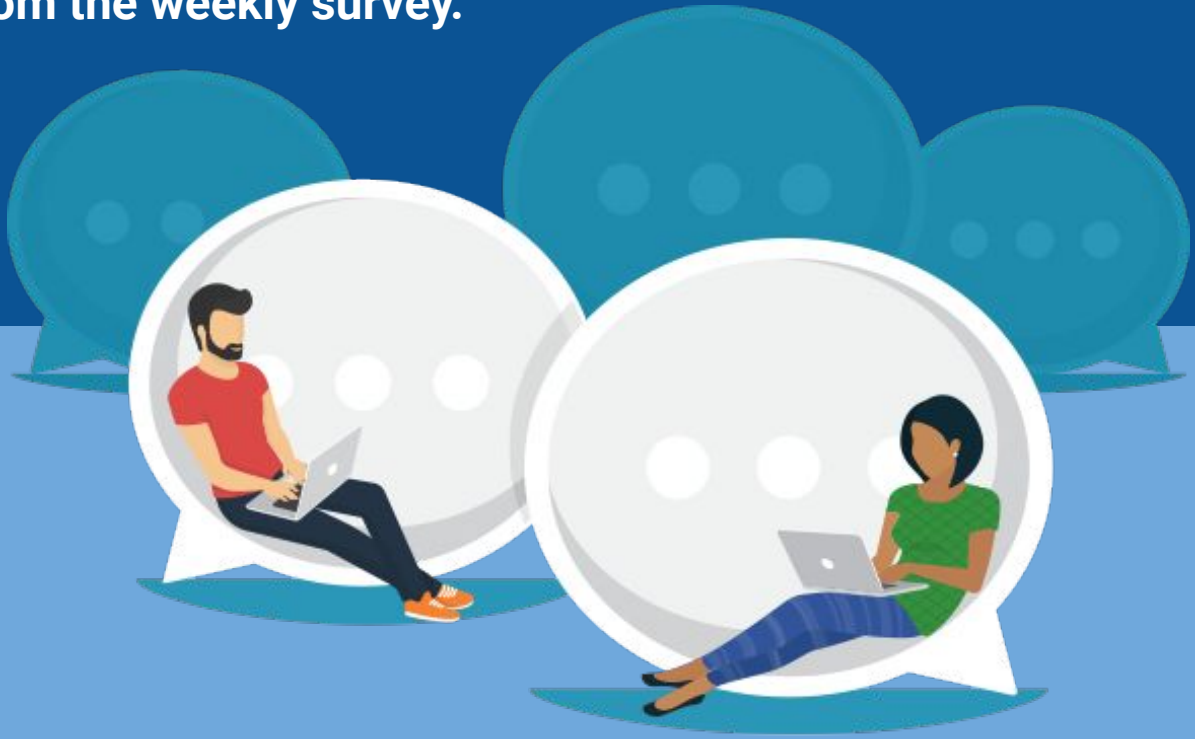


Congratulations on reaching the midpoint of the program!

You'll be asked to take a mid-course survey soon. It'll be a little different from the weekly survey.

Make sure to take your time with the questions.

We want to hear what you have to say!



This Week: Plotly.js

By the end of this week, you'll know how to:

- Create basic plots with Plotly, including bar charts, line charts, and pie charts
- Use `D3.json()` to fetch external data, such as CSV files and web APIs
- Parse data in JSON format
- Use functional programming in JavaScript to manipulate data
- Use JavaScript's Math library to manipulate numbers
- Use event handlers in JavaScript to add interactivity to a data visualization
- Deploy an interactive chart to GitHub Pages

Today's Agenda

By completing today's activities, you'll learn the following skills:

01 Multiple traces in plots

02 Dynamic events on webpages

03 Use D3 to load in data from an outside source

Make sure you've downloaded any relevant class files!

This Week's Challenge

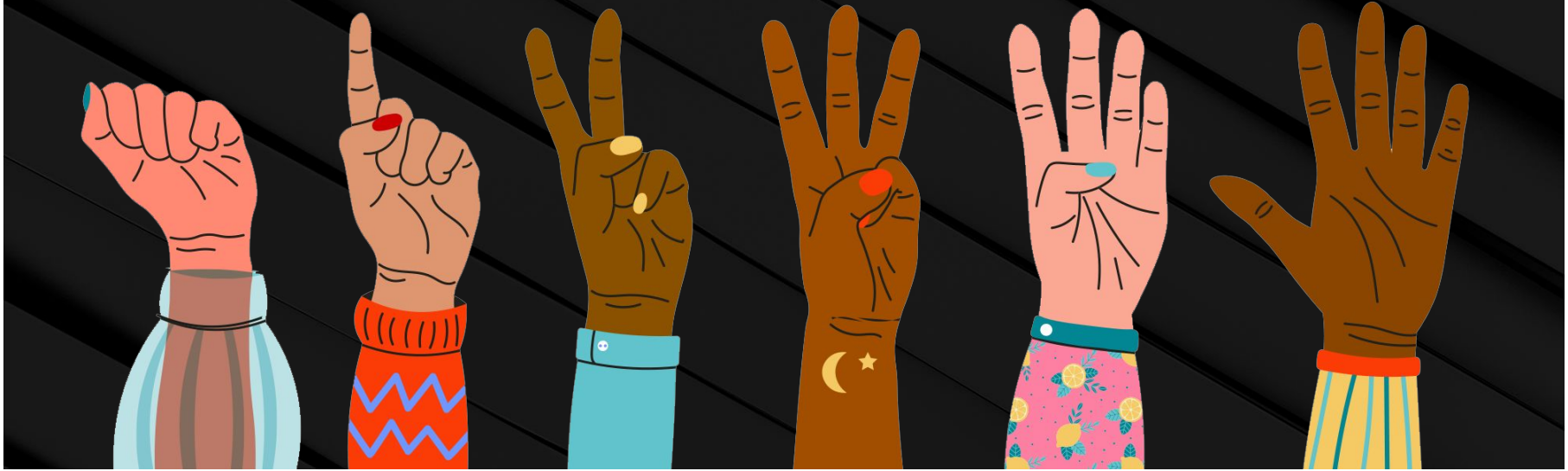
Using the skills learned throughout the week, create a customized dashboard on a webpage using a horizontal bar chart, a bubble chart, and a gauge chart.

Module 12

Today’s Agenda

FIST TO FIVE:

How comfortable do you feel with this topic?



Multiple Traces

Time to Code

Plotting Multiple Traces

Suggested Time:

5 minutes

Questions?

Activity: Multiple Traces

In this activity, you will use functional programming techniques to create a Plotly chart with multiple traces.

Suggested Time:

15 minutes

Time’s Up! Let’s Review.

This was a fairly challenging activity, as it requires using functional programming techniques.

Activity Review: Multiple Traces

For the first trace, which deals with Greek gods, defining the x-axis points can be done by using `map()`to return the `pair` value from the dataset.

```
var trace1 = {
x: data.map(row => row.pair),
y: data.map(row => row.greekSearchResults),
text: data.map(row => row.greekName),
name: "Greek",
type: "bar"
};
```

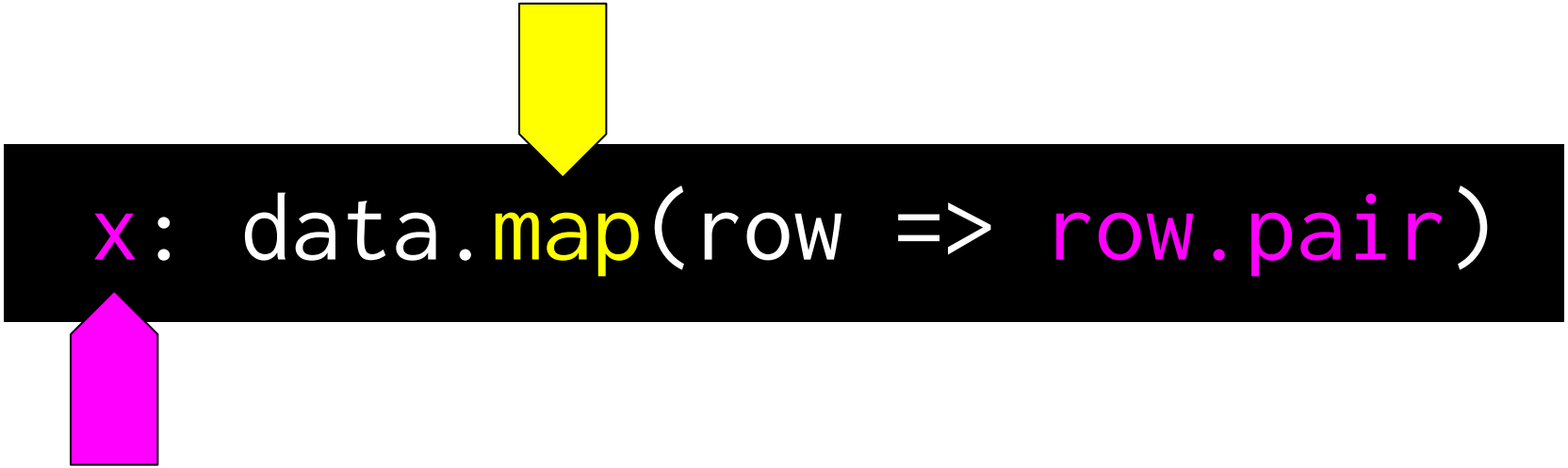
Activity Review: Multiple Traces

Here, `row => row.pair` is essentially a shortcut for writing `function (row) {return row.pair;}`.

```
var trace1 = {
x: data.map(row => row.pair),
y: data.map(row => row.greekSearchResults),
text: data.map(row => row.greekName),
name: "Greek",
type: "bar"
};
```

Activity Review: Multiple Traces

`map()` is used to transform each row in the dataset to its `pair` attribute.



`x` becomes an array of `row.pair` values.

Activity Review: Multiple Traces

The second trace deals with Roman gods.

Everything here is analogous to trace 1:

```
// Trace 2 for the Roman Data
var trace2 = {
  x: data.map(row => row.pair),
  y: data.map(row => row.romanSearchResults),
  text: data.map(row => row.romanName),
  name: "Roman",
  type: "bar"
};
```

Activity Review: Multiple Traces

The rest of the plot is created by storing the traces in an array, creating a layout, and plotting.

```
// Combining both traces
var traceData = [trace1, trace2];

// Apply the group barmode to the layout
var layout = {
  title: "Greek vs Roman gods search results",
  barmode: "group"
};

// Render the plot to the div tag with id "plot"
Plotly.newPlot("plot", traceData, layout)
```

Questions?

Click & Dropdown Events

Instructor Demonstration

An Eventful Click

D3 is used to select the document `body` and create an event handler that calls `updatePage()` when a change takes place.

An Eventful Click

D3 is used to select the document `body` and create an event handler that calls `updatePage()` when a change takes place.

```
d3.selectAll("body").on("change", updatePage);

function updatePage() {
    var dropdownMenu = d3.selectAll("#selectOption").node();
    var dropdownMenuID = dropdownMenu.id;
    var selectedOption = dropdownMenu.value;
}
```

An Eventful Click

The dropdown menu's `id` and `value` attributes are assigned to variables and then logged to the console.

```
d3.selectAll("body").on("change", updatePage);

function updatePage() {
    var dropdownMenu = d3.selectAll("#selectOption").node();
    var dropdownMenuID = dropdownMenu.id;
    var selectedOption = dropdownMenu.value;
}
```

An Eventful Click

Two key takeaways:

01

A dropdown menu is created in the HTML document.

02

A D3 event handler calls a custom function to print the dropdown menu's attributes to the console.

Questions?

Instructor Demonstration

Dropdown Events and Plotly

Dropdown Events and Plotly

- A default plot is rendered on the page.
- A change takes place in the DOM when a dropdown menu item is selected.
- A function is triggered with the DOM element's value as its argument.
- The function uses Plotly's `restyle()` method to modify an existing plot.

Questions?

Activity: A Musical Pie

In this activity, you will enhance your event-handling chops by creating a dynamic pie chart using Plotly. When a country is selected from the dropdown menu, its dataset will be displayed in the browser.

Suggested Time:

20 minutes

Time’s Up! Let’s Review.

Instructor Demonstration

D3.json

Time to Code

Dynamically Selected City Forecasts

Suggested Time:

20 minutes

Questions?

GitHub Pages

You will need to deploy this week's Challenge to GitHub Pages.

This Week's Challenge

01

The data for the homework will be in a JSON file.

02

You will use the `d3.json()` method to fetch data from the JSON file and visualize it.

03

You will need to upload the JSON file to GitHub, along with the HTML and JavaScript script files.

Plotly Visualization with a Data File

The benefits of deploying a Plotly visualization with a data file:

01

It makes a publicly available data visualization that is much more visually appealing than a published Jupyter Notebook.

02

The ability to read in data from local files means that data sources aren't limited to placing API calls.

Time to Code

Deployment to GitHub Pages

Suggested Time:

20 minutes

Deploy an Existing Project to GitHub Pages

01 Go to the GitHub website and create a new repository by clicking **New**

02 The repository must be made public in order to be deployed to GitHub Pages.

03 Clone the repository by copying the URL and entering `git clone <url>` in the CLI.

04 Next, `push` the project to GitHub.

05 Navigate to the directory of the repository.

06 Copy and paste the HTML, JavaScript, and JSON files from the Solved folder into the repository.

Deploy an Existing Project to GitHub Pages

07 `git add`

08 `git commit -m "<your message here>`

09 `git push origin main`

10 Next, `push` the project to GitHub.

11 Next, go to the project page on GitHub and click on **Settings** to configure for deployment.

12 Under Settings, go to GitHub Pages and select **main branch**. Click **Save**.

Questions?