CSE 503

Introduction to Computer Science for Non-Majors

Dr. Eric Mikida
epmikida@buffalo.edu
208 Capen Hall

Day 21

JavaScript Libraries and Web Servers

Announcements

- Lab #3 AutoLab should be open by tonight

Recap

- **H**yper **T**ext **M**arkup **L**anguage **(HTML)** allows us to describe the layout and content of a webpage
  - Elements are defined with open and close tags, ie ** ** and ** **
  - The content goes between the tags
  - Tags define how the content is formatted/displayed
- JavaScript files can be loaded into your webpage as well to give it more power
  - JavaScript can access elements by their ID and update their content

HTML

```
<html>
<head></head>
<body>
  <h1>First Web Page</h1>
  <p>My Content</p>
  <div id="myDiv"></div>
  <script src="myCode.js"></script>
</body>
</html>
```

HTML

```
<html>
<head></head>
<body>
  <h1>First Web Page</h1>
  <p>My Content</p>
  <div id="myDiv"></div>
  <script src="myCode.js"></script>
</body>
</html>
```

Elements are defined by open and close tags

HTML

```
<html>
<head></head>
<body>
  <h1>First Web Page</h1>
  <p>My Content</p>
  <div id="myDiv"></div>
  <script src="myCode.js"></script>
</body>
</html>
```

Content goes between the tags

The tags define how the content is shown on the webpage

HTML

```
<html>
<head></head>
<body>
  <h1>First Web Page</h1>
  <p>My Content</p>
  <div id="myDiv"></div>
  <script src="myCode.js"></script>
</body>
</html>
```

Tags can also have additional configuration defined via key-value pairs in the open tag

HTML

```
<html>
<head></head>
<body>
  <h1>First Web Page</h1>
  <p>My Content</p>
  <div id="myDiv"></div>
  <script src="myCode.js"></script>
</body>
</html>
```

By giving an element an id, we are able to access/modify that element in our JavaScript code

HTML

```
<html>
<head></head>
<body>
  <h1>First Web Page</h1>
  <p>My Content</p>
  <div id="myDiv"></div>
  <script src="myCode.js"></script>
</body>
</html>
```

Script elements can load and execute our JavaScript code

Front End JavaScript

Our JavaScript can access elements by id:

```
let myDiv = document.getElementById("myDiv");
```

Front End JavaScript

Our JavaScript can access elements by id:

```
let myDiv = document.getElementById("myDiv");
```

An element is just an object in JavaScript. We can modify the key-value pairs just like any other object. The **`innerHTML`** key can be used to set the content of the element:

```
myDiv.innerHTML = "Content added from JavaScript";
```

Front End JavaScript

Our JavaScript can access elements by id:

```
let myDiv = document.getElementById("myDiv");
```

An element is just an object in JavaScript. We can modify the key-value pairs just like any other object. The **`innerHTML`** key can be used to set the content of the element:

```
myDiv.innerHTML = "Content added from JavaScript";
```

More Info: https://www.w3schools.com/js/js_htmldom.asp

Front End JavaScript

Now when the body loads and our script runs, our page looks like this:

JavaScript Libraries

```
<html>
<head>
  <script src="https://momentjs.com/downloads/moment.js"></script>
</head>
<body>
  <h1>First Web Page</h1>
  <p>My Content</p>
  <p id="datetime"></p>
  <div id="myDiv"></div>
  <script src="myCode.js"></script>
</body>
</html>
```

JavaScript Libraries

```
<html>
<head>
  <script src="https://momentjs.com/downloads/moment.js"></script>
</head>
<body>
  <h1>First Web Page</h1>
  <p>My Content</p>
  <p id="datetime"></p>
  <div id="myDiv"></div>
  <script src="myCode.js"></script>
</body>
</html>
```

We can also load external libraries.

By adding our script element to the head, the library will be loaded without displaying any content.

JavaScript Libraries

```
<html>
<head>
  <script src="https://momentjs.com/downloads/moment.js"></script>
</head>
<body>
  <h1>First Web Page</h1>
  <p>My Content</p>
  <p id="datetime"></p>
  <div id="myDiv"></div>
  <script src="myCode.js"></script>
</body>
</html>
```

We also added a new <p> element so that our JavaScript has a place to display the date and time.

JavaScript Libraries

When the page is loaded, there will be a request to automatically download the library.

The library is simply a JavaScript file that someone else has written and shared with us for free. This is called ***open-source*** since everyone has access to the source code.

Source: https://github.com/moment/moment/

Using the Moment Library

As before, we must first get the element to display our information:

```
let elem = document.getElementById("datetime");
```

Using the Moment Library

As before, we must first get the element to display our information:

```
let elem = document.getElementById("datetime");
```

The library defines a function named **`moment()`** which returns the current time as an *object*. This object contains other properties and functions.

Using the Moment Library

As before, we must first get the element to display our information:

```
let elem = document.getElementById("datetime");
```

The library defines a function named **`moment()`** which returns the current time as an *object*. This object contains other properties and functions.

Here we get the current time and call the **`format()`** function to choose how it's displayed using date/time placeholders (ex. MM for a 2 digit month).

```
elem.innerHTML = moment().format("YYYY-MM-DD<br/><b>h:mm:ss a</b>");
```

Extending our Example

Now our web page also displays the current time…or does it? What is the web page actually displaying?

Extending our Example

Now our web page also displays the current time…or does it? What is the web page actually displaying?

*Can we make it display the **current time**, not the time the page was loaded?*

Extending our Example

Now our web page also displays the current time…or does it? What is the web page actually displaying?

*Can we make it display the **current time**, not the time the page was loaded?*

YES!

Automatically Updating Your Page

First wrap our previous code in a function:

```
function displayTime(){
    let elem = document.getElementById("datetime");
    elem.innerHTML = moment().format("YYYY-MM-DD<br/><b>h:mm:ss a</b>");
}
```

Automatically Updating Your Page

First wrap our previous code in a function:

```
function displayTime(){
    let elem = document.getElementById("datetime");
    elem.innerHTML = moment().format("YYYY-MM-DD<br/><b>h:mm:ss a</b>");
}
```

The built-in function **`setInterval`** allows us to automatically call a function repeatedly. The first argument is the function to call. The second argument is a Number representing a time in milliseconds.

```
setInterval(displayTime, 1000);
```

Automatically Updating Your Page

First wrap our previous code in a function:

```
function displayTime(){
    let elem = document.getElementById("datetime");
    elem.innerHTML = moment().format("YYYY-MM-DD<br/><b>h:mm:ss a</b>");
}
```

The built-in function **`setInterval`** allows us to automatically call a function repeatedly. The first argument is the function to call. The second argument is a Number representing a time in milliseconds.

```
setInterval(displayTime, 1000);
```

This makes the **`displayTime`** function get called every second (1000 milliseconds)

Plot.ly

Plot.ly is a JavaScript library that helps us generate plots and graphs

https://plot.ly/javascript/

**There is also a Python version of this library*

HTML File

```
<html>
<head>
  <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
</head>
<body>
  <div id="myPlot"></div>
  <script src="plots.js"></script>
</body>
</html>
```

HTML File

```
<html>
<head>
  <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
</head>
<body>
  <div id="myPlot"></div>
  <script src="plots.js"></script>
</body>
</html>
```

Load the plot.ly library in **<head>**

HTML File

```
<html>
<head>
  <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
</head>
<body>
  <div id="myPlot"></div>
  <script src="plots.js"></script>
</body>
</html>
```

Create a ** ** where we can display our plot

HTML File

```
<html>
<head>
  <script src="https://cdn.plot.ly/plotly-latest.min.js"></script>
</head>
<body>
  <div id="myPlot"></div>
  <script src="plots.js"></script>
</body>
</html>
```

Execute a JavaScript file we have written that creates and displays the plot

JavaScript File (plot.js in this case)

The library has a function named **`newPlot`** that takes the id of an HTML element (the element should be a div), and an array as parameters.

To make a line graph the array will contain object(s) with keys "x" and "y":

```
let data = [
  { x: [1930, 1940, 1950, 1960],
    y: [573076, 575901, 580132, 532759] }
];

Plotly.newPlot('myPlot', data);
```

JavaScript File (plot.js in this case)

The library has a function named **`newPlot`** that takes the id of an HTML element (the element should be a div), and an array as parameters.

To make a line graph the array will contain object(s) with keys "x" and "y":

```
let data = [
  { x: [1930, 1940, 1950, 1960],
    y: [573076, 575901, 580132, 532759] }
];

Plotly.newPlot('myPlot', data);
```

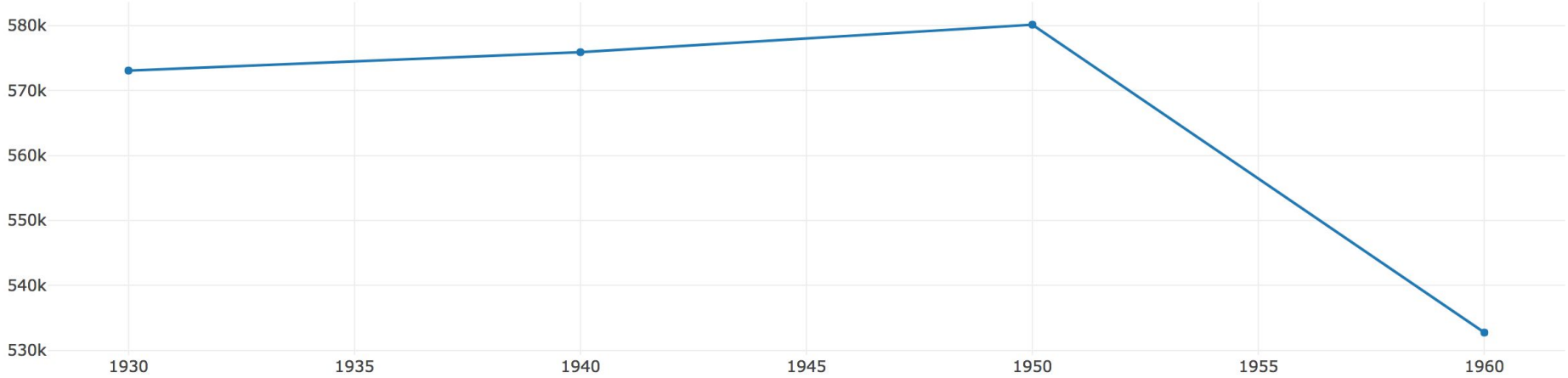
JavaScript File (plot.js in this case)

The library has a function named **`newPlot`** that takes the id of an HTML element (the element should be a div), and an array as parameters.

To make a line graph the array will contain object(s) with keys "x" and "y":

```
let data = [
  { x: [1930, 1940, 1950, 1960],
    y: [573076, 575901, 580132, 532759] }
];

Plotly.newPlot('myPlot', data);
```

Our Beautiful Plot



Let's Make it Nicer

```
let data = [
  { x: [1930, 1940, 1950, 1960],
    y: [573076, 575901, 580132, 532759],
    name: "Buffalo Population" }
];

let layout = {
    title: "Buffalo Population",
    xaxis: { title: "year" },
    yaxis: { title: "population" }
}

Plotly.newPlot('myPlot', data, layout);
```

Let's Make it Nicer

```
let data = [
  { x: [1930, 1940, 1950, 1960],
    y: [573076, 575901, 580132, 532759],
    name: "Buffalo Population" }
];

let layout = {
    title: "Buffalo Population",
    xaxis: { title: "year" },
    yaxis: { title: "population" }
}

Plotly.newPlot('myPlot', data, layout);
```

Add a name to our line (only displayed if we have multiple lines or add a legend)

Let's Make it Nicer

```
let data = [
  { x: [1930, 1940, 1950, 1960],
    y: [573076, 575901, 580132, 532759],
    name: "Buffalo Population" }
];

let layout = {
    title: "Buffalo Population",
    xaxis: { title: "year" },
    yaxis: { title: "population" }
}

Plotly.newPlot('myPlot', data, layout);
```

Add a name to our line (only displayed if we have multiple lines or add a legend)

Create a layout object with key-value pairs for title, xaxis, and yaxis.

Let's Make it Nicer

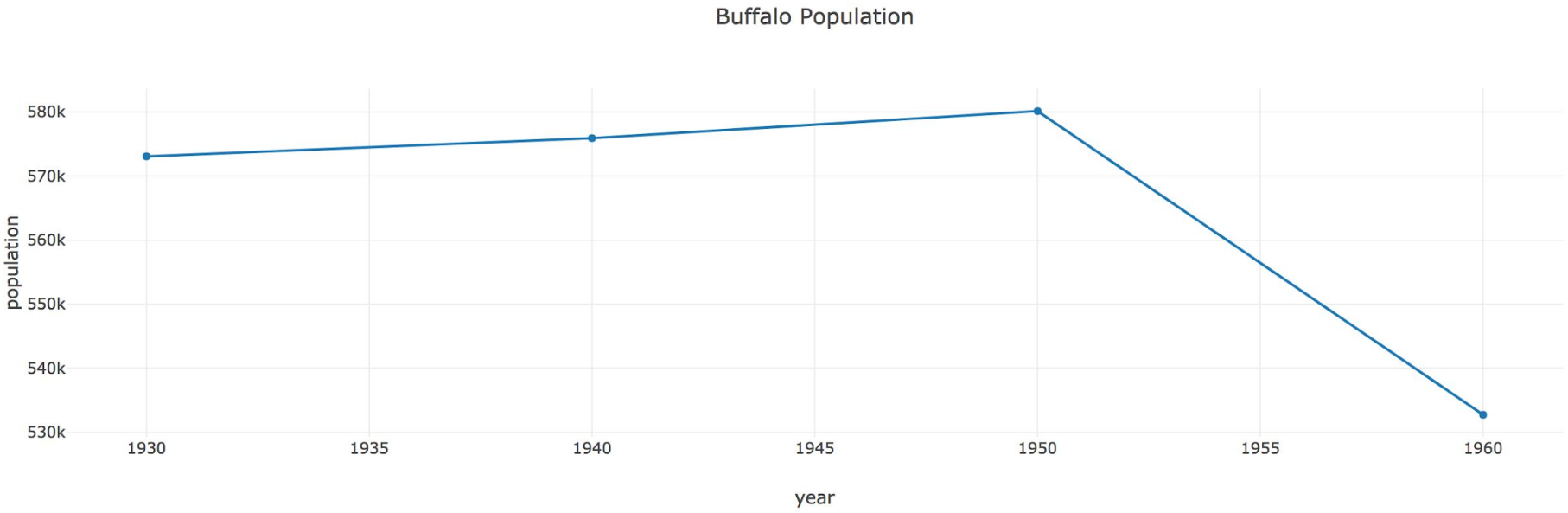
```
let data = [
  { x: [1930, 1940, 1950, 1960],
    y: [573076, 575901, 580132, 532759],
    name: "Buffalo Population" }
];

let layout = {
    title: "Buffalo Population",
    xaxis: { title: "year" },
    yaxis: { title: "population" }
}

Plotly.newPlot('myPlot', data, layout);
```

Add a name to our line (only displayed if we have multiple lines or add a legend)

Create a layout object with key-value pairs for title, xaxis, and yaxis.

Pass layout to **`newPlot`**

Much Nicer!


Buffalo Population
580k
570k
560k
550k
540k
530k
population
1930
1935
1940
1945
1950
1955
1960
year

Adding More Lines

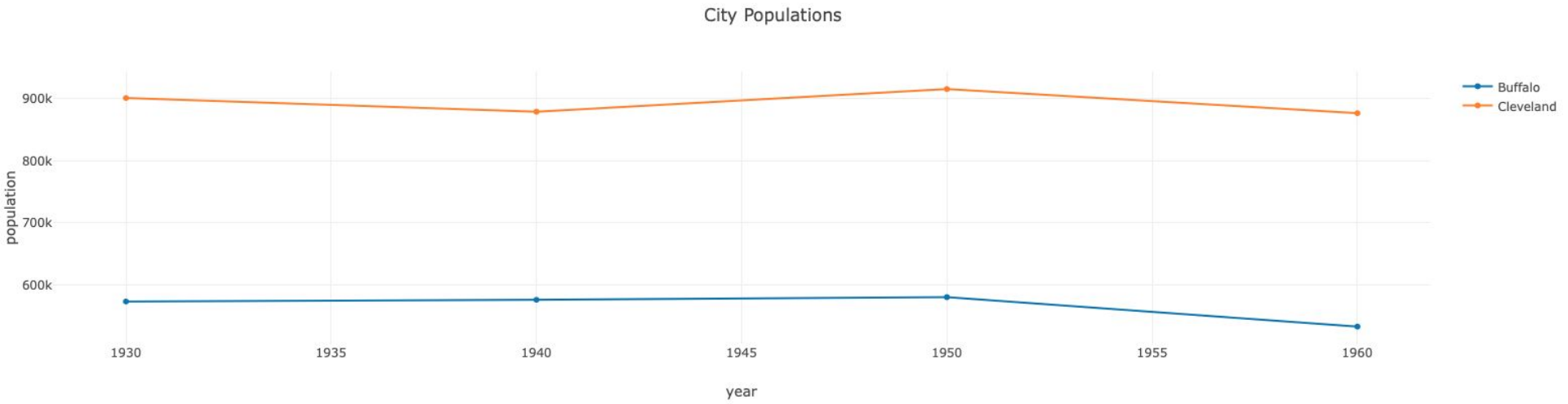
```
let data = [
    { x: [1930,1940,1950,1960],  y: [573076,575901,580132,532759], name: "Buffalo"},
    { x: [1930,1940,1950,1960],  y: [900429,878336,914808,876050], name: "Cleveland"}
];

let layout = {
    "title": "City Populations",
    xaxis: { "title": "year" },
    yaxis:{ "title": "population" }
}

Plotly.newPlot('myPlot', data, layout);
```

Adding More Lines

```
let data = [
    { x: [1930,1940,1950,1960],  y: [573076,575901,580132,532759], name: "Buffalo"},
    { x: [1930,1940,1950,1960],  y: [900429,878336,914808,876050], name: "Cleveland"}
];

let layout = {
    "title": "City Populations",
    xaxis: { "title": "year" },
    yaxis:{ "title": "population" }
}

Plotly.newPlot('myPlot', data, layout);
```

Data is an array, so we can have multiple objects that we want to plot!

Our Final Plot



A Note on Libraries

Reminder: You are learning concepts, not memorizing specific examples

We're showing you how to make line graphs in lecture…but there are many more graphs we can make with plot.ly, and many more ways for us to manipulate how our graphs are displayed.

You have the basic conceptual knowledge, and now you can read the Plot.ly documentation to learn more!