EECS E6893 Big Data Analytics
HW3: Data visualization

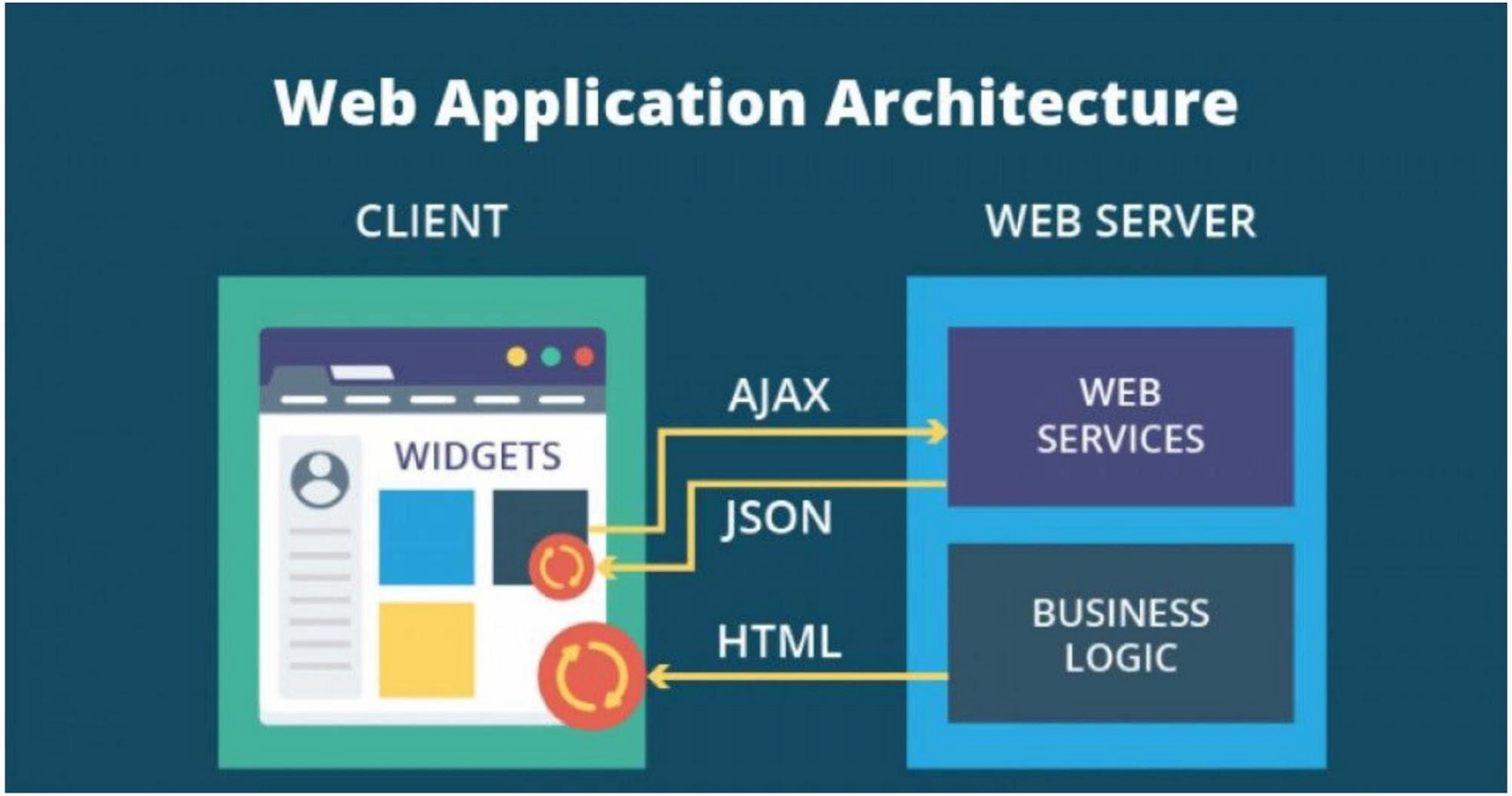
Tutorial I

Apurva Patel
amp2365@columbia.edu

Agenda

- Introduction of Web Application
  - HTML, CSS and JavaScript
  - 2 important things to know: SVG and DOM
- Using D3.js to do data visualization

Web Application



Reference: https://hackr.io/blog/web-application-architecture-definition-models-types-and-more

Client-side of Web

- HTML, CSS and JS are the parts of all websites that users directly interact with.
- HTML provides the *basic structure* of sites, which is enhanced and modified by other technologies like CSS and JavaScript.
- CSS is used to control *presentation, formatting, and layout*.
- JavaScript is used to control the *behavior* of different elements.

Reference: https://tutorial.techaltum.com/webdesigning.html

WEB DESIGNING
TECHALTUM
HTML (Structure)
CSS (Presentation)
Javascript (functionality)
tutorial.techaltum.com

Reference: https://tutorial.techaltum.com/webdesigning.html

What and Why?

- JavaScript is a programming language
- used by Web browsers to create a dynamic and interactive experience for the user.
- Most of the functions and applications that make the Internet indispensable to modern life are coded in some form of JavaScript.
- Some of the dynamic website enhancements performed by JavaScript are: Loading new content or data onto the page without reloading the page, Rollover effects and dropdown menus etc.
- Some of its most powerful features involve asynchronous interaction with a remote server.

Common Uses of JavaScript

- Form validation
- Page embellishments and special effects
- Navigation systems
- Basic math calculations
- Dynamic content manipulation
- Sample applications
  - Dashboard widgets in Mac OS X, Google Maps, Philips universal remotes, Writely word processor, hundreds of others…

JavaScript in Web Pages

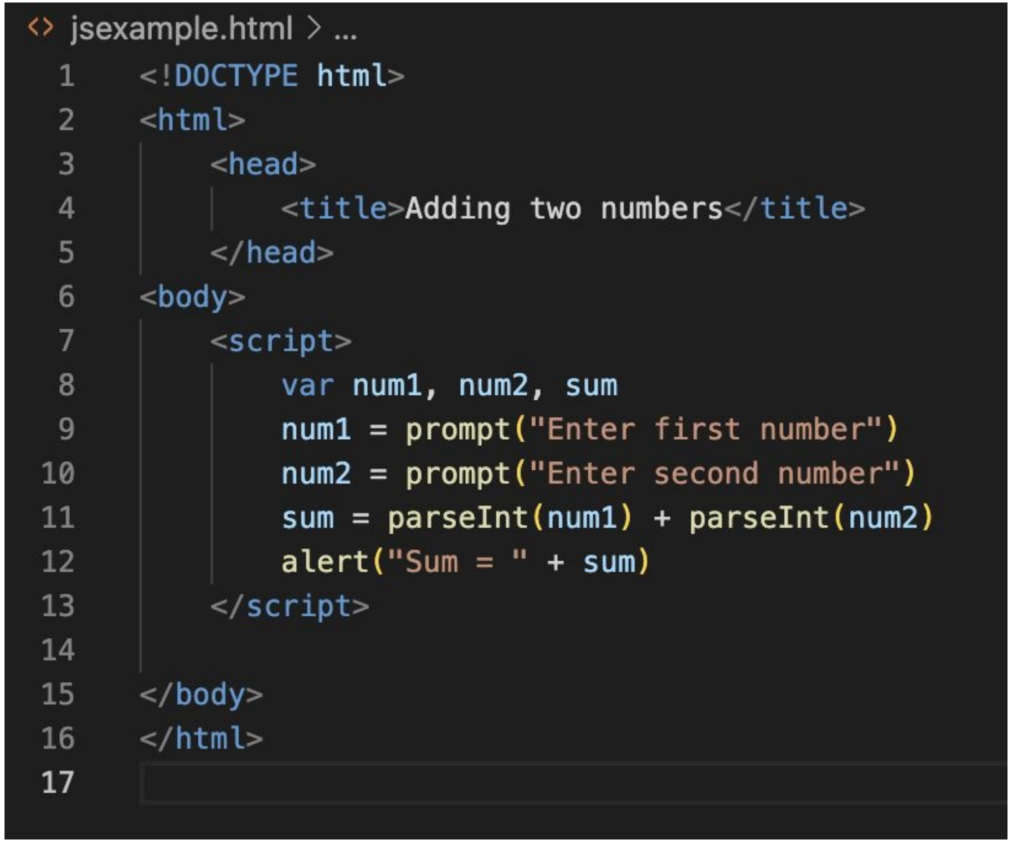
- Embedded in HTML page as <script> element
  - JavaScript written directly inside <script> element
    - <script> alert("Hello World!") </script>
  - Linked file as src attribute of the <script> element

    <script type="text/JavaScript" src=“functions.js"></script>
- Event handler attribute

  <a href="http://www.yahoo.com " onmouseover="alert('hi');">
- Pseudo-URL referenced by a link

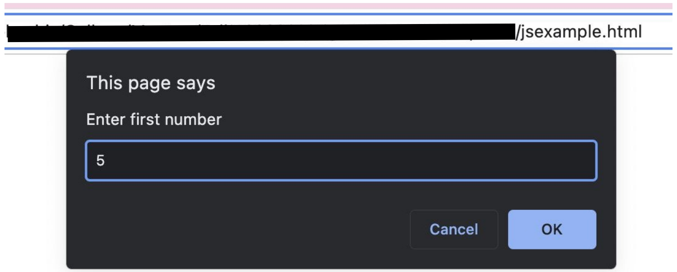
  <a href=“JavaScript: alert(‘You clicked’);”>Click me</a>

Example 1: Add Two Numbers



jsexample.html > ...

```
<!DOCTYPE html>
<html>
    <head>
        <title>Adding two numbers</title>
    </head>
<body>
    <script>
        var num1, num2, sum
        num1 = prompt("Enter first number")
        num2 = prompt("Enter second number")
        sum = parseInt(num1) + parseInt(num2)
        alert("Sum = " + sum)
    </script>

</body>
</html>

```

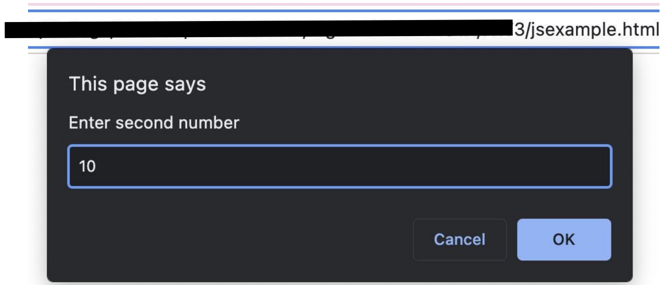


[redacted]/jsexample.html

This page says

Enter first number

5

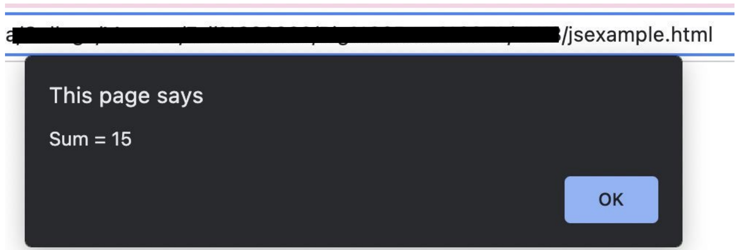


[redacted]3/jsexample.html

This page says

Enter second number

10

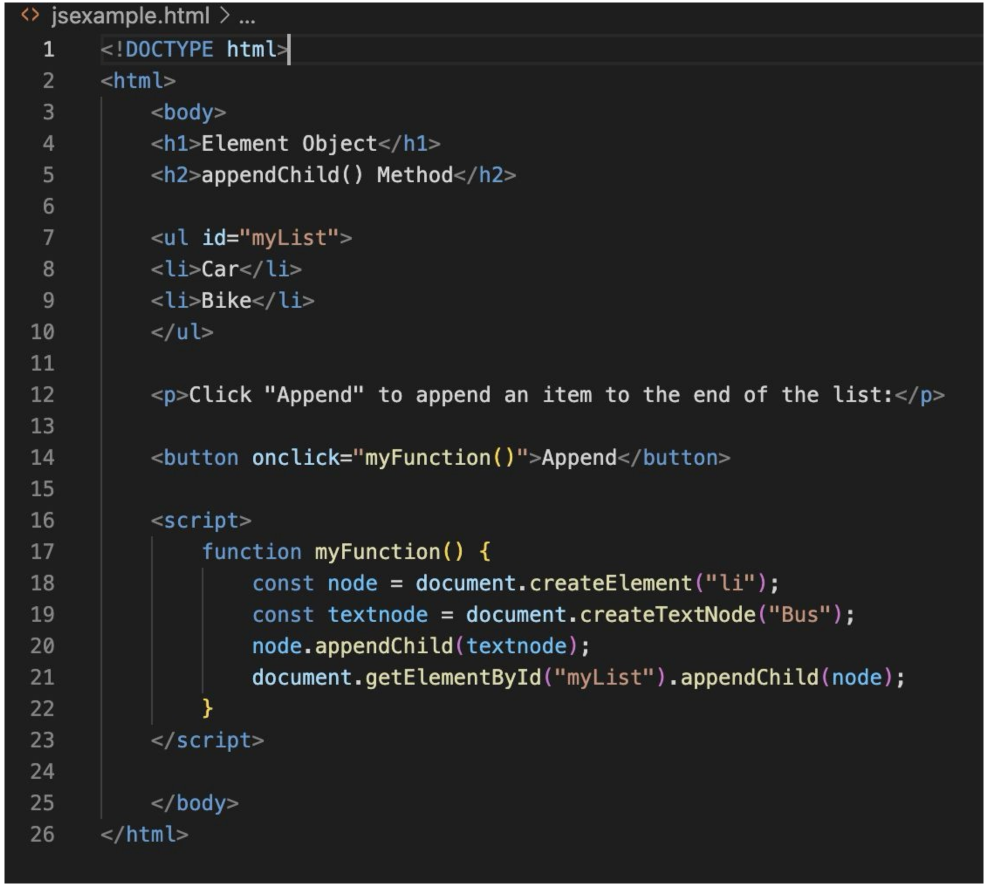


[redacted]/jsexample.html

This page says

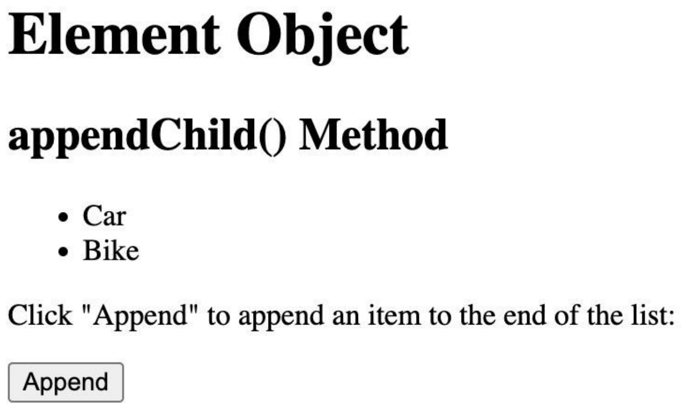
Sum = 15

Example 2: Page Manipulation



jsexample.html > ...

```html
<!DOCTYPE html>
<html>
    <body>
    <h1>Element Object</h1>
    <h2>appendChild() Method</h2>

    <ul id="myList">
    <li>Car</li>
    <li>Bike</li>
    </ul>

    <p>Click "Append" to append an item to the end of the list:</p>

    <button onclick="myFunction()">Append</button>

    <script>
        function myFunction() {
            const node = document.createElement("li");
            const textnode = document.createTextNode("Bus");
            node.appendChild(textnode);
            document.getElementById("myList").appendChild(node);
        }
    </script>

    </body>
</html>
```

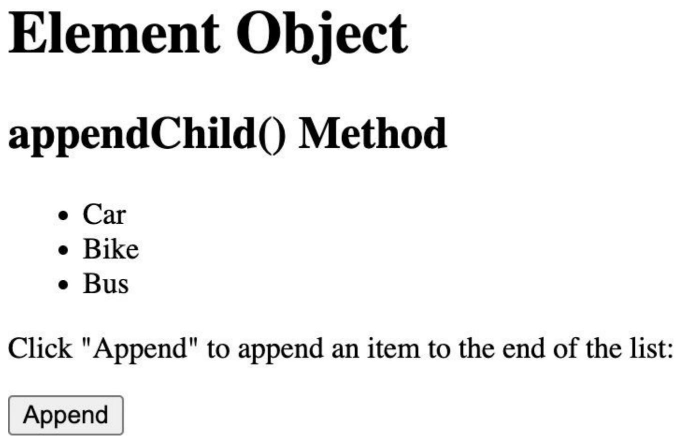


Element Object

appendChild() Method

- Car
- Bike

Click "Append" to append an item to the end of the list:



Element Object

appendChild() Method

- Car
- Bike
- Bus

Click "Append" to append an item to the end of the list:

Language Basics

- JavaScript is case sensitive whereas, onClick, ONCLICK, … are HTML, thus not case-sensitive
- Statements terminated by returns or semi-colons
  - x = x+1;   same as   x = x+1
- “Blocks” of statements enclosed in { …}
- Variables
  - Define using the var statement
  - Define implicitly by its first use, which must be an assignment
    - Implicit definition has global scope, even if occurs in nested scope!

JavaScript Primitive Data types

- Boolean: true and false
- Number: 64-bit floating point
  - Similar to Java double and Double
  - No integer type
  - Special values NaN (not a number) and Infinity
- String: sequence of zero or more Unicode chars
  - No separate character type (just strings of length 1)
  - Literal strings using ' or " characters (must match)
- Special objects: null and undefined

Objects

- An object is a collection of named properties
- Think of it as an associative array or hash table
  - Set of name:value pairs
    - objBob = {name: “Bob", grade: 'A', level: 3};
  - Play a role similar to lists in Lisp / Scheme
- New members can be added at any time
    - objBob.fullname = 'Robert';
- Can have methods

Functions

- Functions are objects with method called “( )”
  - A property of an object may be a function (=method)
    - function max(x,y) { if (x>y) return x; else return y;};
    - max.description = “return the maximum of two arguments”;
  - Local declarations may appear in function body
- Call can supply any number of arguments
  - functionname.length : # of arguments in definition
  - functionname.arguments.length : # arguments in call
  - Basic types are passed by value, objects by reference
- “Anonymous” functions
  - (function (x,y) {return x+y}) (2,3);

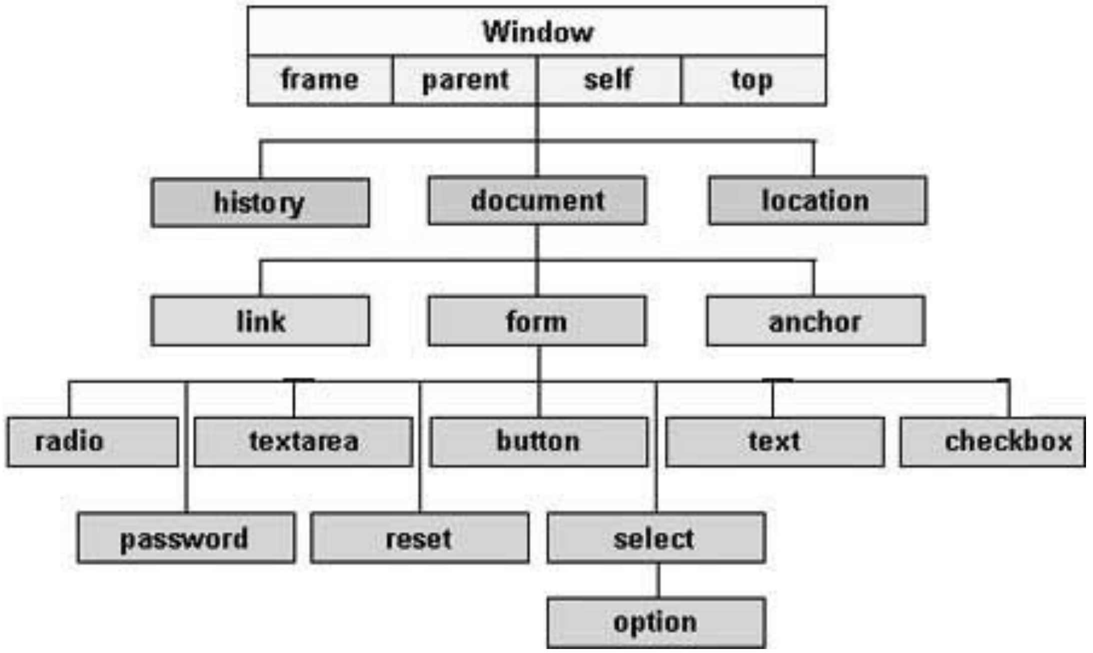
Document Object Model (DOM)

- HTML page is structured data
- DOM provides representation of this hierarchy
- Examples
  - Properties: document.alinkColor, document.URL, document.forms[ ], document.links[ ], document.anchors[ ], …
  - Methods: document.write(document.referrer)
    - These change the content of the page!
- Also Browser Object Model (BOM)
  - Window, Document, Frames[], History, Location, Navigator (type and version of browser)

Document Object Model (DOM)

The way a document content is accessed and modified is called the Document Object Model, or DOM. The Objects are organized in a hierarchy. This hierarchical structure applies to the organization of objects in a Web document.

- Window object − Top of the hierarchy. It is the outmost element of the object hierarchy.
- Document object − Each HTML document that gets loaded into a window becomes a document object. The document contains the contents of the page.
- Form object − Everything enclosed in the <form>...</form> tags sets the form object.
- Form control elements − The form object contains all the elements defined for that object such as text fields, buttons, radio buttons, and checkboxes.



JavaScript Document Object Model (DOM) hierarchy
Ref: https://www.tutorialspoint.com/javascript/javascript_html_dom.htm

Introduction to HTML

- HTML is a language for describing web pages.
- HTML stands for **H**yper **T**ext **M**arkup **L**anguage
- HTML is not a programming language, it is a **markup language**
- A markup language is a set of **markup tags**
- HTML uses **markup tags** to describe web pages

Objectives of HTML

- create, save and view a HTML document
- format a web page using section heading tags
- describe Ordered and Unordered lists
- explain graphics in HTML document
- describe hypertext links and making text/image link

HTML Tools

a) HTML Editor: it is the program that one uses to create and save HTML documents. They fall into two categories:

- Text based or code based which allows one to see the HTML code as one is creating a document.e.g. Notepad.
- Netscape composer

b) Web Browser: program to view and test the HTML documents. They translate Html encoded files into text, image, sounds and other features user see. Microsoft Internet Explorer, Netscape, Chrome are examples of browsers that enables user to view text and images and many more other World Wide Web features.

HTML Terminology

- Tag: Tags are always written within angles brackets. it is a piece of text is used to identify an element so that the browser realizes how to display its contents.e.g.<HTML> tag indicates the start of an HTML document .
  HTML tag can be two types. They are:-
  - Paired Tags :A tag is said to be a paired tag if text is placed between a tag and its companions tag.In paired tag ,the first tag is referred to as opening tag and the second tag is referred to as closing tag.
  - Unpaired Tags: An unpaired tag does not have a companion tag .unpaired tag also known as singular or Stand-Alone tags.e.g:<br>,<hr> etc.

HTML Terminology

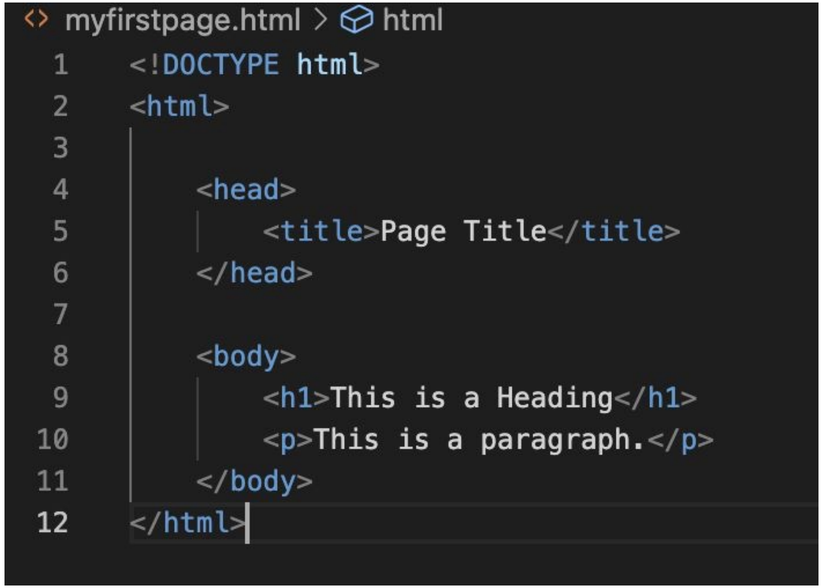
- Attribute: Attribute is the property of an tag that specified in the opening angle brackets. It supplies additional information like color,size,home font-style etc to the browser about a tag. E.g. most of the common attributes are height, color,width,src,border,align etc.

- DTD: Document Type Definition is a collection of rules written in standard Generalized Markup Language(SGML).HTML is define in terms of its DTDS. All the details of HTML tags, entities and related document structure are defined in the DTDS.

- ELEMENT: Element is the component of a document’s structure such as a title, a paragraph or a list. It can include an opening and a closing tag and the contents within it.

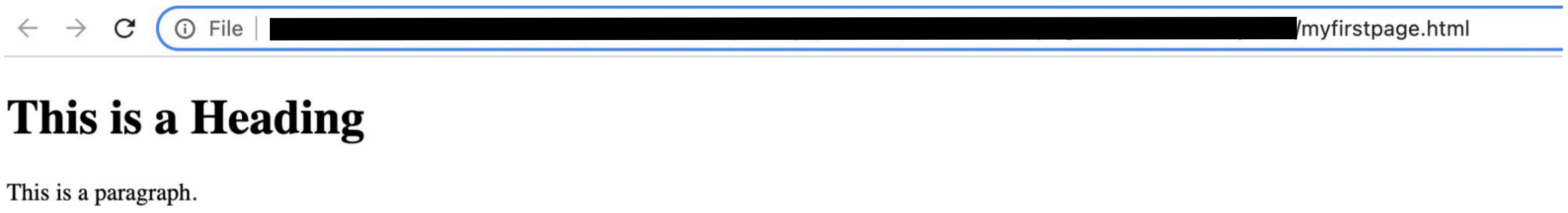
Steps to create a HTML file and view in browser

- Step-1: Open a text editor or notepad on your machine.
- Step-2: Enter the following lines of code:



```html
<!DOCTYPE html>
<html>

    <head>
        <title>Page Title</title>
    </head>

    <body>
        <h1>This is a Heading</h1>
        <p>This is a paragraph.</p>
    </body>
</html>
```

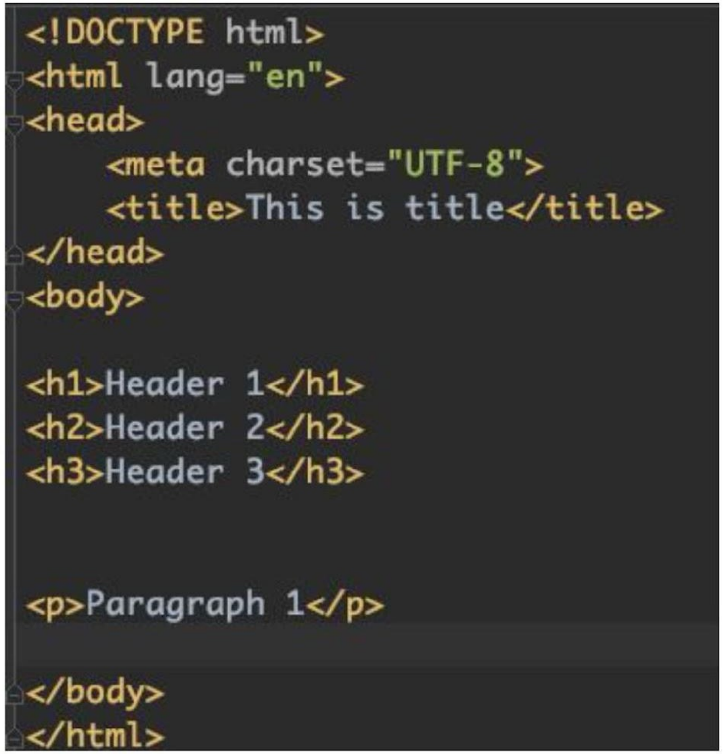
- Step-3: Save the file as myfirstpage.html (go to File-Save As give File name: myfirstpage.html-choose save as type: All Files-click save)
- Step-4: Viewing document in web browser (go to folder where file is saved and open it in any browser of your choice)



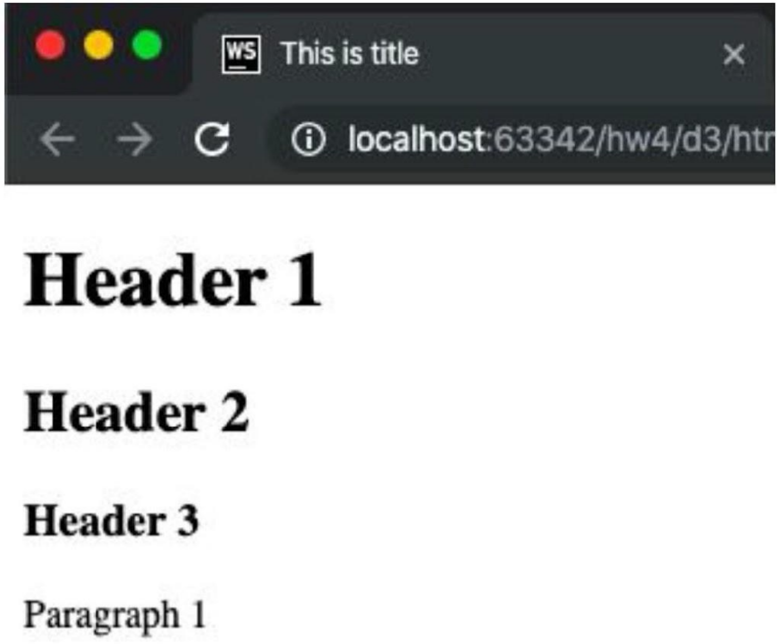
This is a Heading

This is a paragraph.

HTML



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>This is title</title>
</head>
<body>

<h1>Header 1</h1>
<h2>Header 2</h2>
<h3>Header 3</h3>


<p>Paragraph 1</p>

</body>
</html>
```



Header 1

Header 2

Header 3

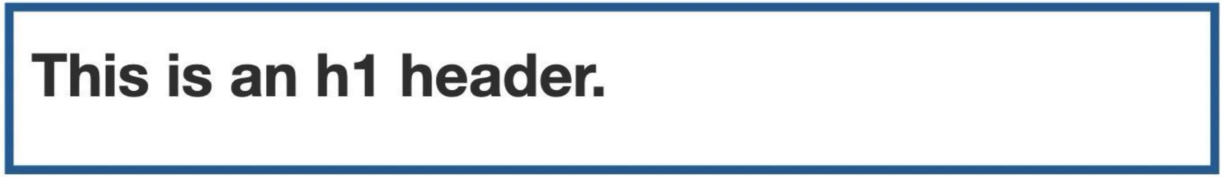
Paragraph 1

Styles

Rough timeline of HTML / CSS history:

Early 90s:

```
<h1>This is an h1 header.</h1>
```



This is an h1 header.

http://www.pmichaud.com/toast/

Styles

Mid 1990s

```
<p>This method of <font color="green" face="Times New Roman">
  styling</font> was deprecated in 1998--but it still works :-) .</p>
```

This method of styling was deprecated in 1998–but it still works :-) .

HTML tag history

http://www.martinrinehart.com/frontend-engineering/engineers/html/html-tag-history.html

Styles: External style sheet (preferred method)

Late 1990s - present: efforts to separate *style* from *content*

```
<head>
    <link rel="stylesheet" href="style.css">
</head>
```

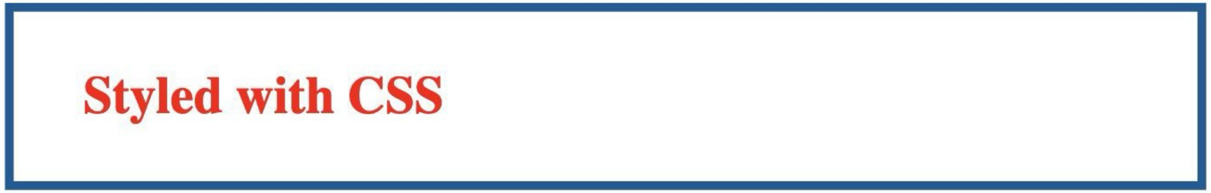
`style.css`:

```
.formal {color: red;
    font-size: 30px;
    font-family: Lucida Calligraphy;
    }
```

Styles: Internal style sheet

<style> tag in <head> section:

```
<head>
  <style type="text/css">
    .formal {color: red;
        font-size: 30px;
        font-family: Lucida Calligraphy;
        }
  </style>
</head>

<body>
  <h2 class="formal">Styled with CSS</h2>
</body>
```

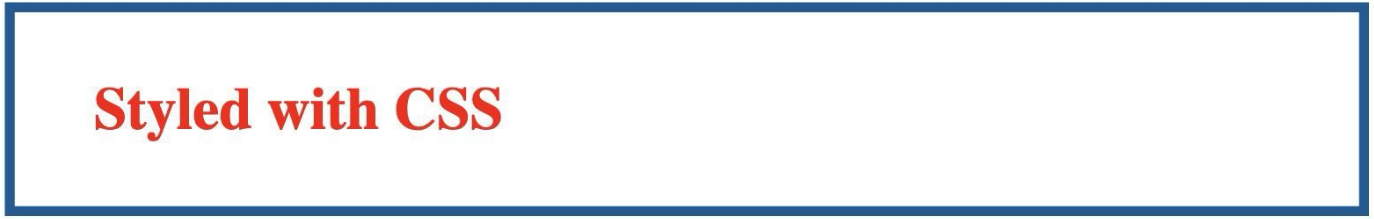

Styled with CSS

Styles: External style sheet

Preferred method of adding styles

Body of html file:

```
<body>
  <h2 class="formal">Styled with CSS</h2>
</body>
```

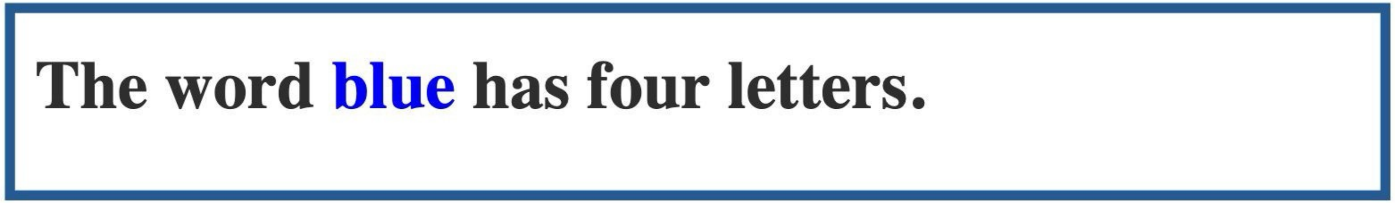

Styled with CSS

http://www.csszengarden.com/ (started 2003)

Styles: Inline style attributes

- Not recommended if you are adding styling manually
- However, JavaScript/D3 add styling *inline*

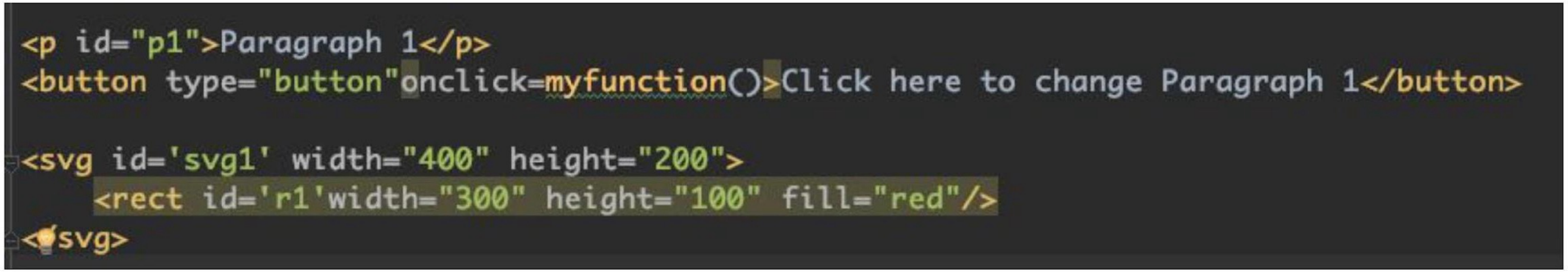
```
<h1 style="font-family: Bookman;">The word
<span style="color: blue;">blue</span>
has four letters.></h1>
```


The word blue has four letters.

view-source:http://www.dolekemp96.org/agenda/issues/education.htm

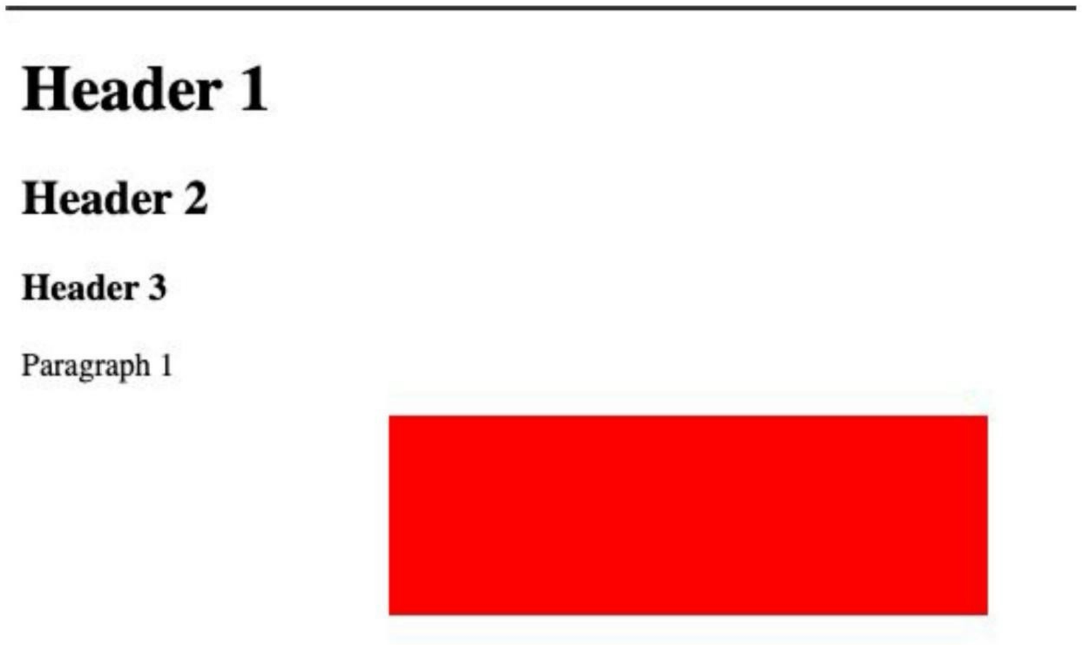
SVG in HTML

- SVG stands for Scalable Vector Graphics. It is used to define vector-based graphics for the Web
- **Every element and every attribute in SVG files can be animated**
- SVG integrates with other W3C standards such as the **DOM** and XSL



```
<p id="p1">Paragraph 1</p>
<button type="button"onclick=myfunction()>Click here to change Paragraph 1</button>

<svg id='svg1' width="400" height="200">
    <rect id='r1'width="300" height="100" fill="red"/>
<svg>
```

SVG in HTML



Header 1

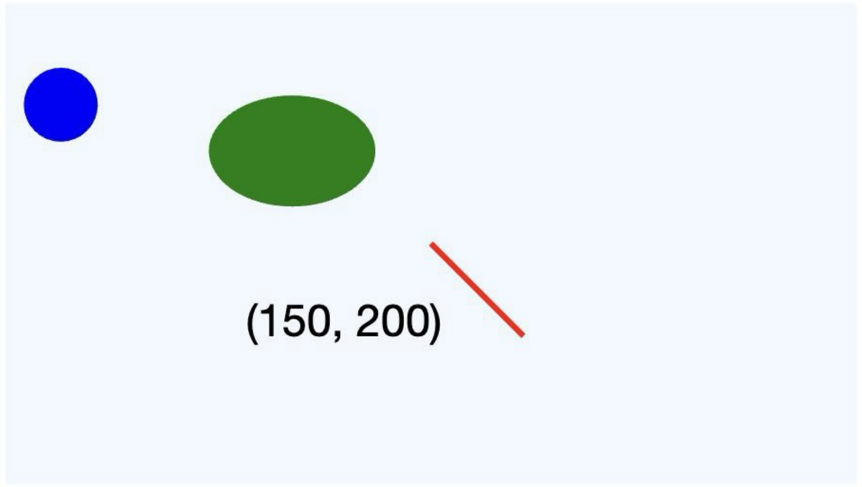
Header 2

Header 3

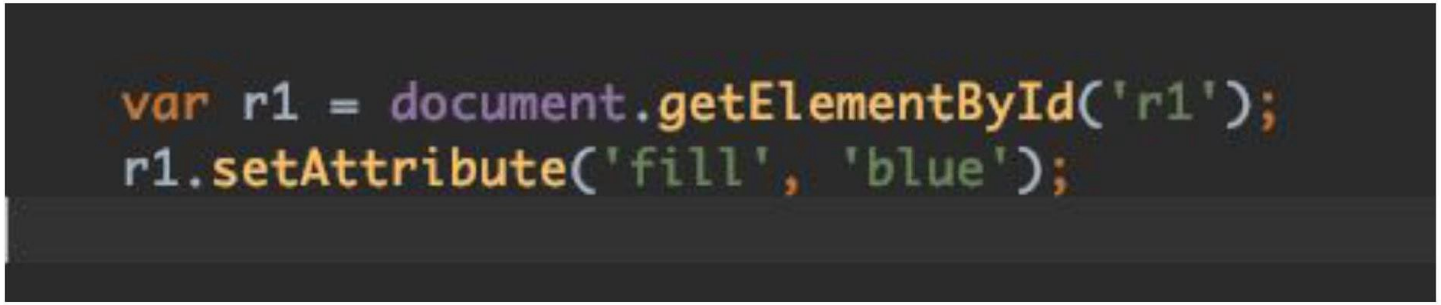
Paragraph 1

SVG

```
<svg width="500" height="300">  <!-- some SVG -->
    <rect x="20" y="20" width="460" height="260" fill="aliceblue"></rect>
    <circle cx="50" cy="75" r="20" fill="blue"></circle>
    <ellipse cx="175" cy="100" rx="45" ry="30" fill="green"></ellipse>
    <text x="150" y="200">(150, 200)</text>
    <line x1="250" y1="150" x2="300" y2="200" stroke="red" stroke-width="3"></line>
</svg>
```



What if we create a SVG elements and use DOM in Javascript to access its attributes?

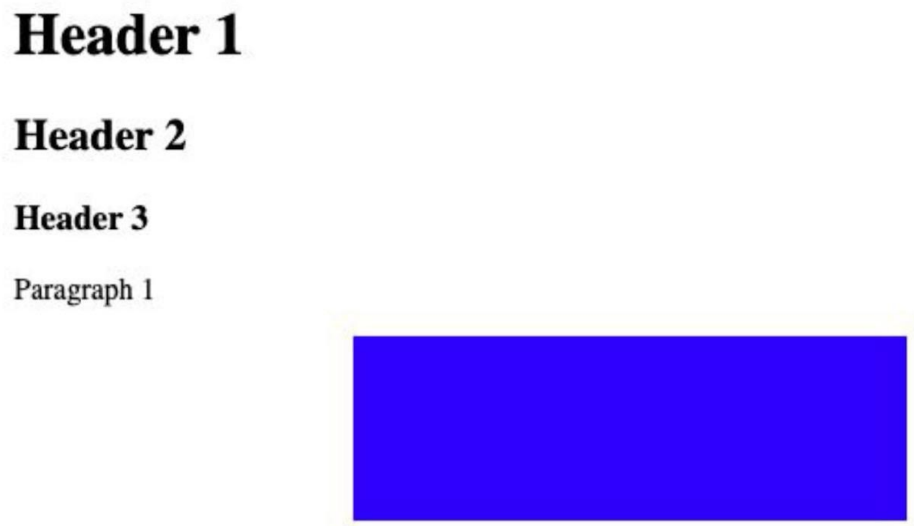


```
var r1 = document.getElementById('r1');
r1.setAttribute('fill', 'blue');
```

If we draw a series of SVG and texts based on data, and use DOM to control their attributes, then we get a simple charts!

- D3.js, a library to do this in a simple way

What if we create a SVG elements and use DOM in Javascript to access its attributes?


Header 1

Header 2

Header 3

Paragraph 1

If we draw a series of SVG and texts based on data, and use DOM to control their attributes, then we get a simple charts!

- D3.js, a library to do this in a simple way

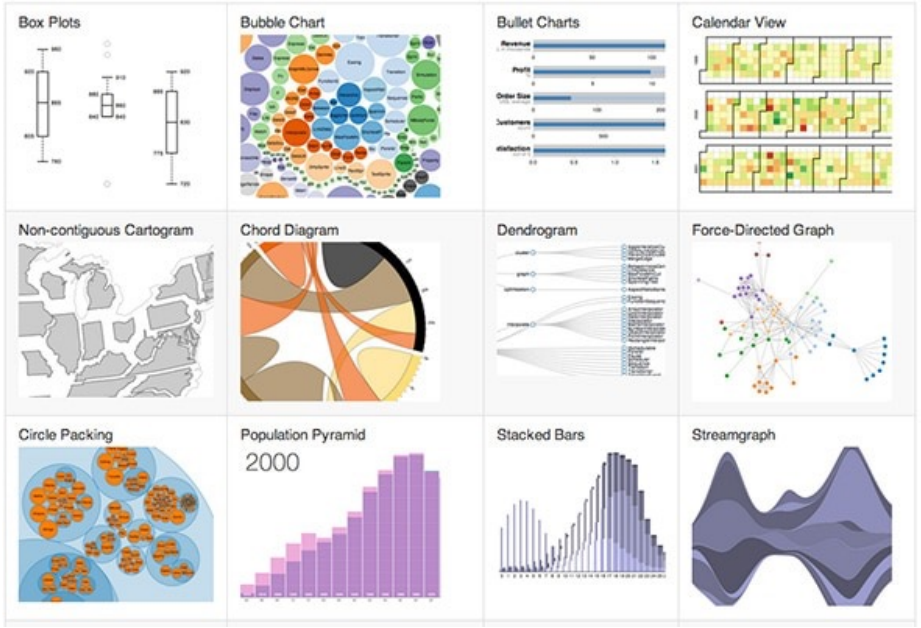
Introduction to D3.js

- What is D3.js?
- Why use D3.js?

Data-Driven Documents

What is D3.js

- D3 stands for “Data-Driven Documents”
- D3.js is a free, open-source JavaScript library for producing dynamic, interactive data visualizations in web browsers
- Its low-level approach built on web standards offers unparalleled flexibility in authoring dynamic, data-driven graphics
- Visit https://d3js.org/ for more tutorials!

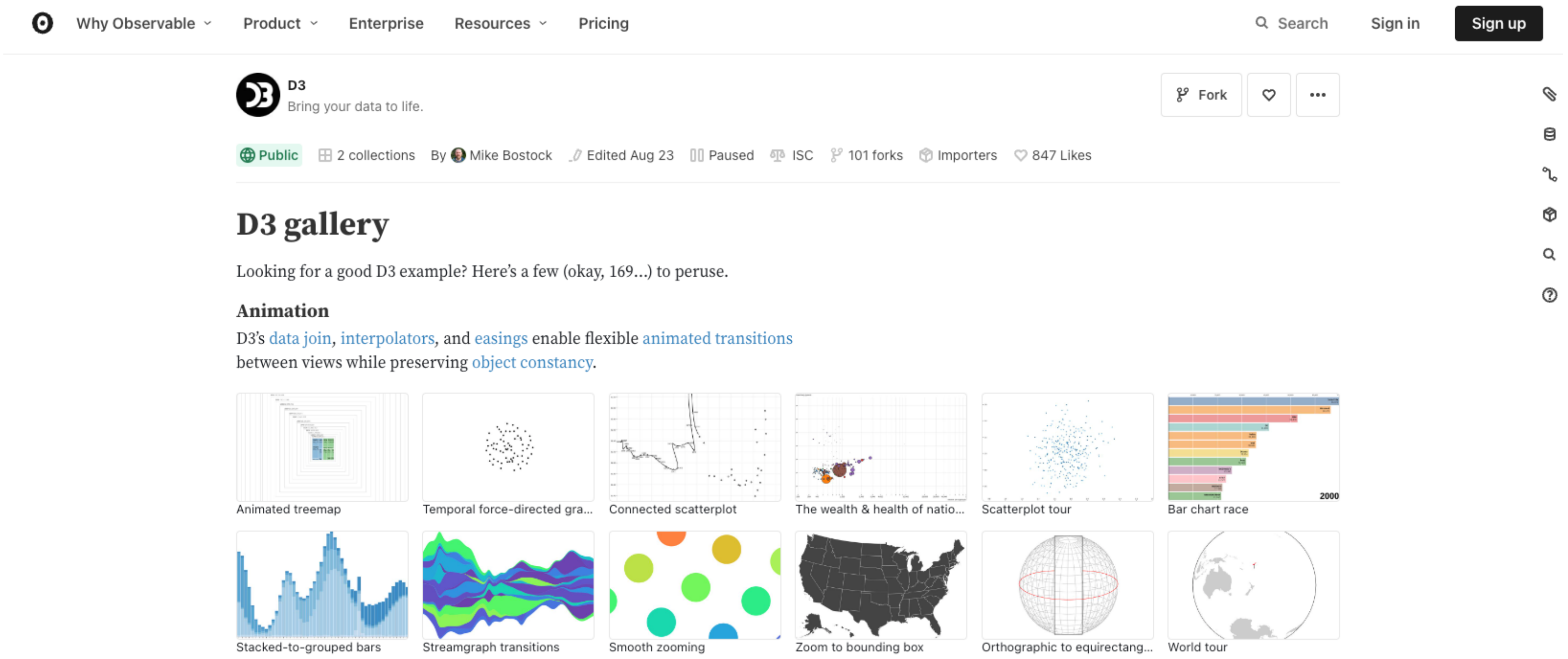


Why use D3.js

- Dynamic and Interactive
- Directly binds data to DOM (Document Object Model)
- Extensive flexibility and control over visualization
- Large community and extensive documentation
- D3 Links:
  - Homepage: https://d3js.org/
  - Github: https://github.com/d3/d3

D3 Live Examples

- D3 Examples page: https://observablehq.com/@d3/gallery



D3 gallery

Looking for a good D3 example? Here's a few (okay, 169...) to peruse.

Animation

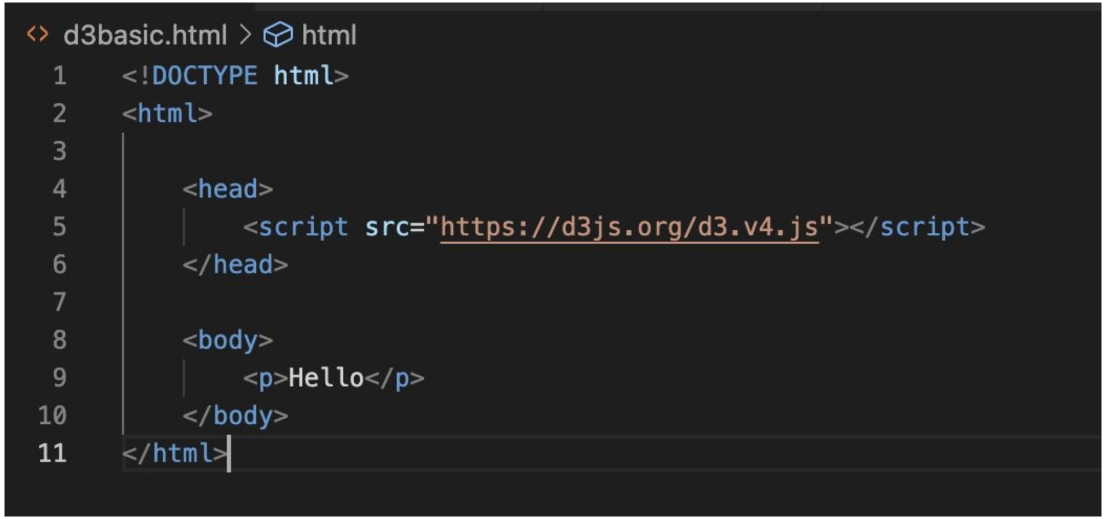
D3's data join, interpolators, and easings enable flexible animated transitions between views while preserving object constancy.

Running D3

- What does D3 require to run?
  - A web browser
  - D3 source code
  - Valid HTML document
  - Server
- Most people probably have a web browser
- D3 source code can be added to any HTML file by including:
  - <script src="https://d3js.org/d3.v6.min.js"></script>
- Server:
  - Can use a remote setup
  - Host a local server

Starting D3

- “src” in script tag
- Sample code to get started. Save file as d3basic.html

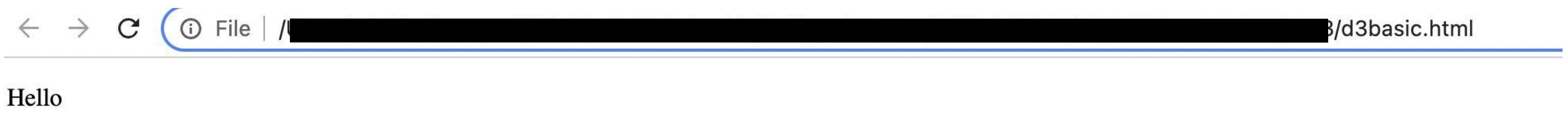


d3basic.html > html

```
<!DOCTYPE html>
<html>

    <head>
        <script src="https://d3js.org/d3.v4.js"></script>
    </head>

    <body>
        <p>Hello</p>
    </body>
</html>
```

d3basic.html

- If we want our server to display that page in our browser we can go to: localhost:8000/d3basic.html
- We could also change d3basic.html to be called index.html, and our localhost:8000 will default to that page
- Key components of this file:
  - Valid HTML
  - Included script tag for D3 source



File | /[redacted]/d3basic.html

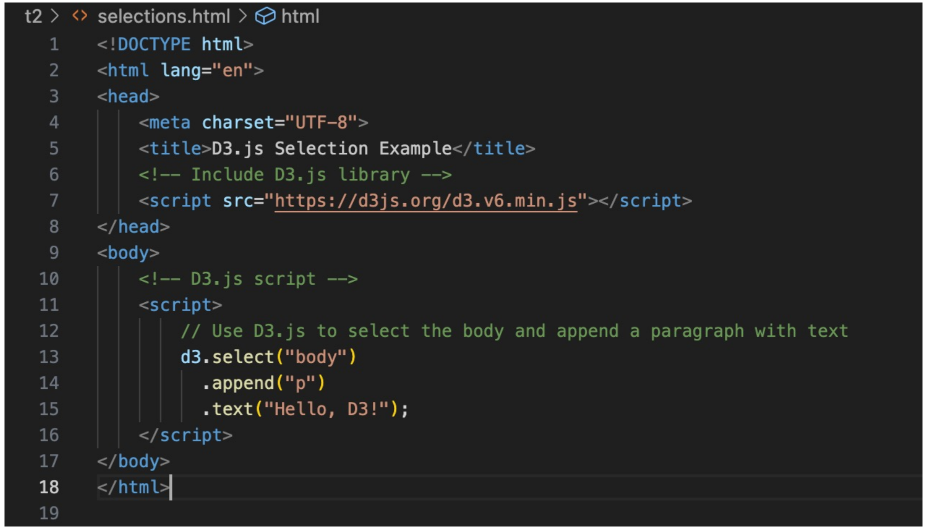
Hello

Core Concepts

- Selections
- Data Bindings
- Dynamic Properties
- Transitions

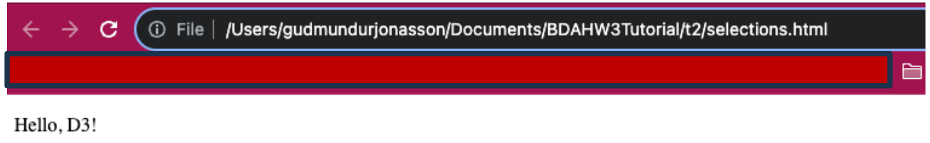
Selections

Selecting elements from the DOM



t2 > selections.html > html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>D3.js Selection Example</title>
    <!-- Include D3.js library -->
    <script src="https://d3js.org/d3.v6.min.js"></script>
</head>
<body>
    <!-- D3.js script -->
    <script>
        // Use D3.js to select the body and append a paragraph with text
        d3.select("body")
          .append("p")
          .text("Hello, D3!");
    </script>
</body>
</html>
```



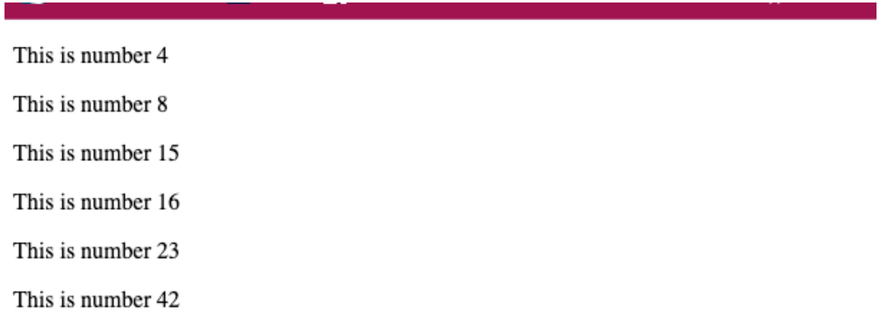
File | /Users/gudmundurjonasson/Documents/BDAHW3Tutorial/t2/selections.html

[redacted]

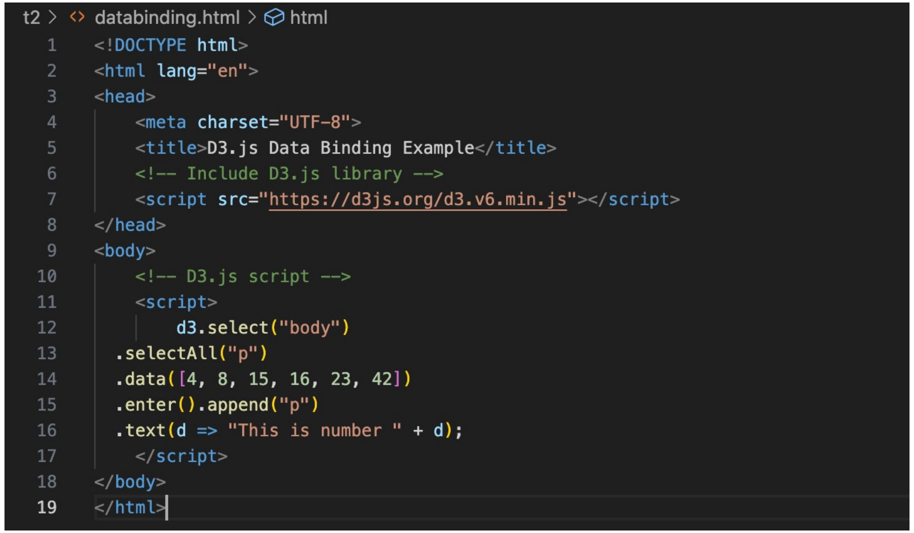
```
Hello, D3!
```

Data Binding

D3 allows binding of data to DOM elements



```
This is number 4

This is number 8

This is number 15

This is number 16

This is number 23

This is number 42
```

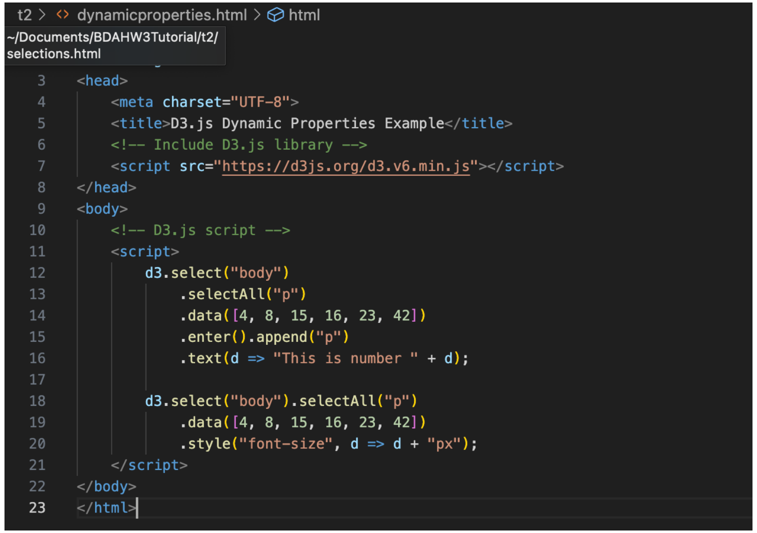


t2 > databinding.html > html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>D3.js Data Binding Example</title>
    <!-- Include D3.js library -->
    <script src="https://d3js.org/d3.v6.min.js"></script>
</head>
<body>
    <!-- D3.js script -->
    <script>
        d3.select("body")
  .selectAll("p")
  .data([4, 8, 15, 16, 23, 42])
  .enter().append("p")
  .text(d => "This is number " + d);
    </script>
</body>
</html>
```

Dynamic Properties

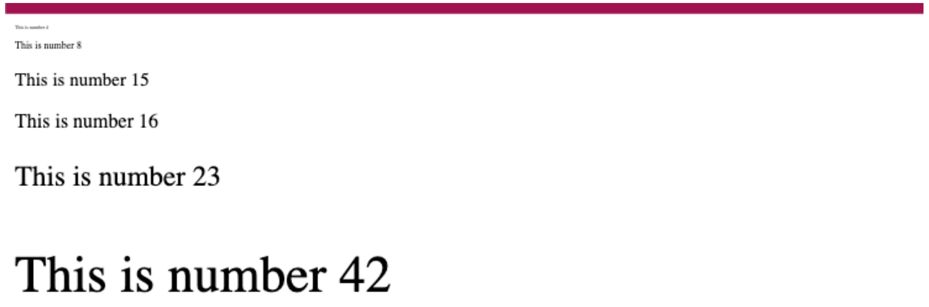
Set properties of DOM elements based on data



t2 > dynamicproperties.html > html

~/Documents/BDAHW3Tutorial/t2/selections.html

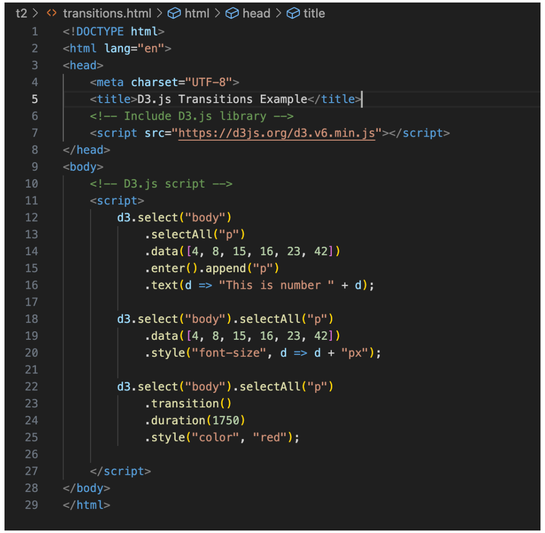
```
<head>
    <meta charset="UTF-8">
    <title>D3.js Dynamic Properties Example</title>
    <!-- Include D3.js library -->
    <script src="https://d3js.org/d3.v6.min.js"></script>
</head>
<body>
    <!-- D3.js script -->
    <script>
        d3.select("body")
            .selectAll("p")
            .data([4, 8, 15, 16, 23, 42])
            .enter().append("p")
            .text(d => "This is number " + d);

        d3.select("body").selectAll("p")
            .data([4, 8, 15, 16, 23, 42])
            .style("font-size", d => d + "px");
    </script>
</body>
</html>
```



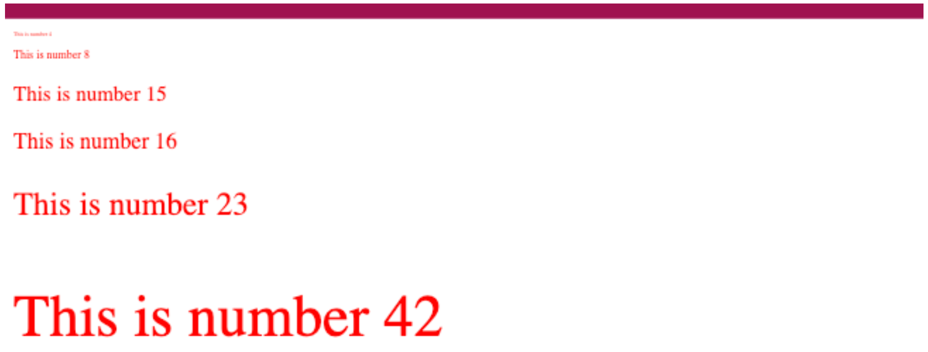
```
This is number 4
This is number 8
This is number 15
This is number 16
This is number 23
This is number 42
```

Transitions

Add animations to visualizations

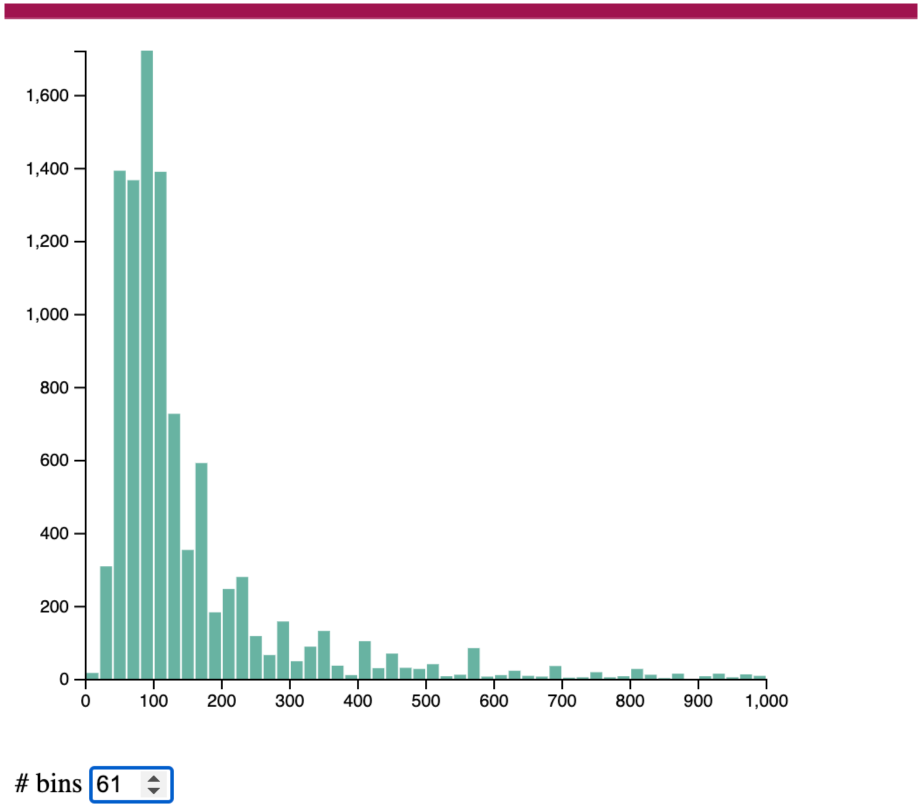


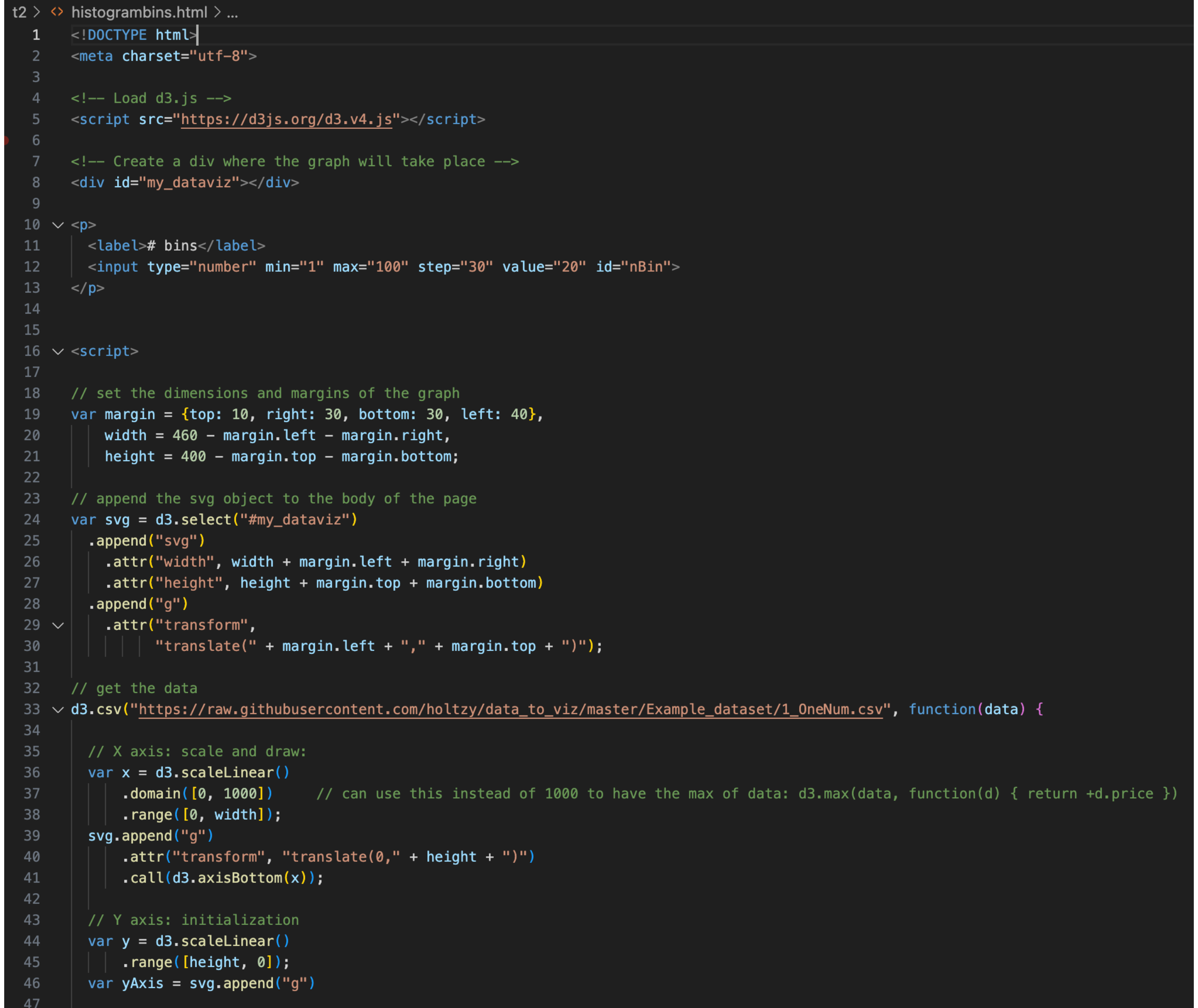
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>D3.js Transitions Example</title>
    <!-- Include D3.js library -->
    <script src="https://d3js.org/d3.v6.min.js"></script>
</head>
<body>
    <!-- D3.js script -->
    <script>
        d3.select("body")
            .selectAll("p")
            .data([4, 8, 15, 16, 23, 42])
            .enter().append("p")
            .text(d => "This is number " + d);

        d3.select("body").selectAll("p")
            .data([4, 8, 15, 16, 23, 42])
            .style("font-size", d => d + "px");

        d3.select("body").selectAll("p")
            .transition()
            .duration(1750)
            .style("color", "red");

    </script>
</body>
</html>
```



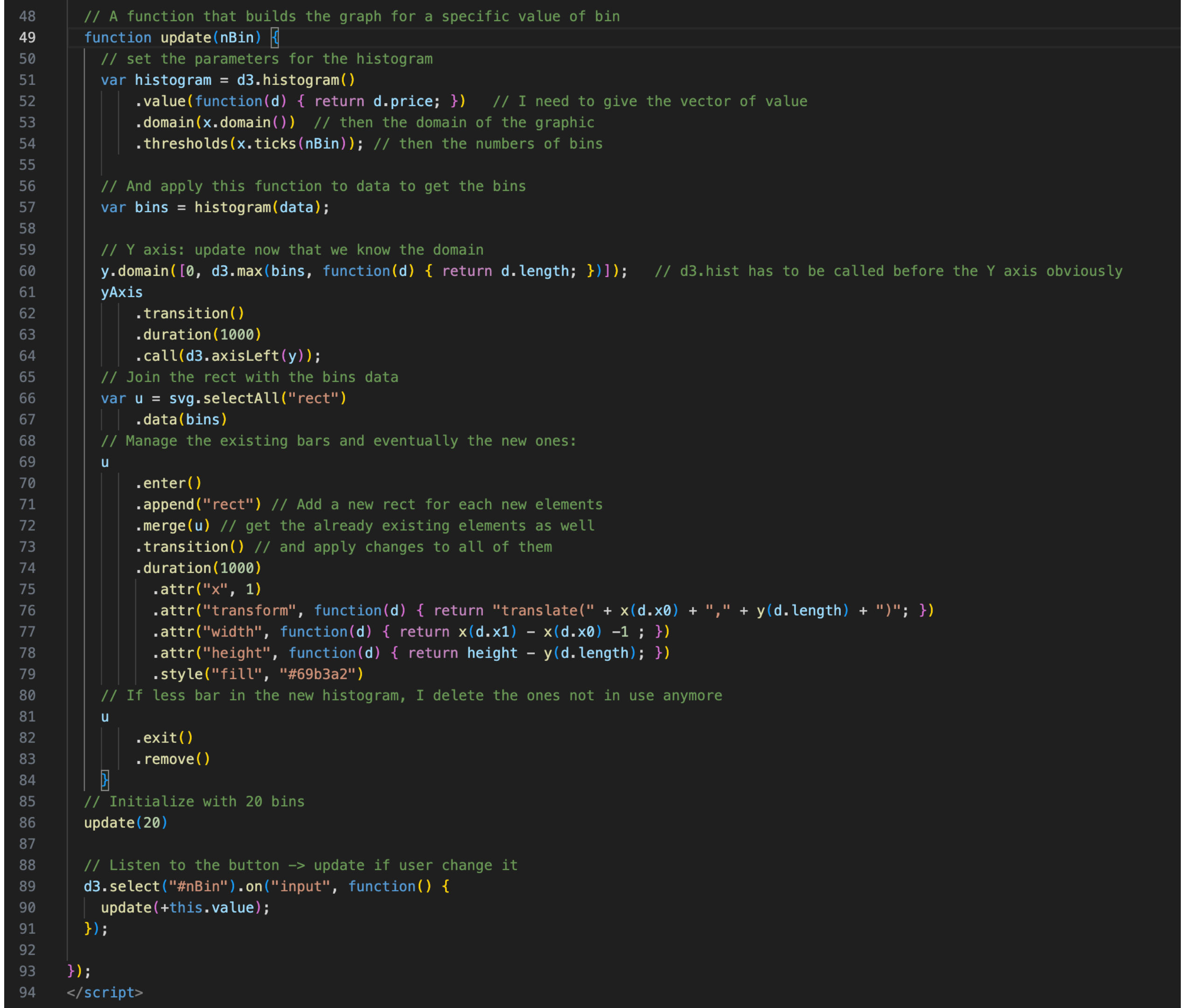
```
This is number 4
This is number 8
This is number 15
This is number 16
This is number 23
This is number 42
```

Histogram with interactive component example



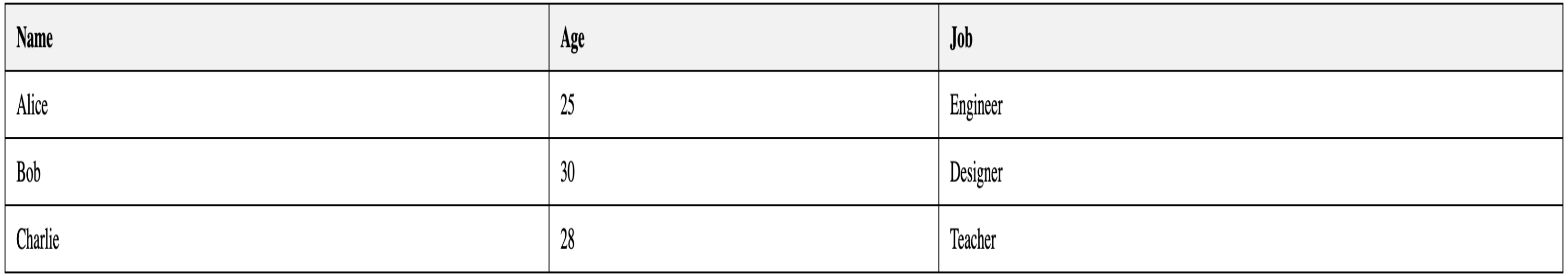


```
<!DOCTYPE html>
<meta charset="utf-8">

<!-- Load d3.js -->
<script src="https://d3js.org/d3.v4.js"></script>

<!-- Create a div where the graph will take place -->
<div id="my_dataviz"></div>

<p>
  <label># bins</label>
  <input type="number" min="1" max="100" step="30" value="20" id="nBin">
</p>


<script>

// set the dimensions and margins of the graph
var margin = {top: 10, right: 30, bottom: 30, left: 40},
    width = 460 - margin.left - margin.right,
    height = 400 - margin.top - margin.bottom;

// append the svg object to the body of the page
var svg = d3.select("#my_dataviz")
  .append("svg")
    .attr("width", width + margin.left + margin.right)
    .attr("height", height + margin.top + margin.bottom)
  .append("g")
    .attr("transform",
          "translate(" + margin.left + "," + margin.top + ")");

// get the data
d3.csv("https://raw.githubusercontent.com/holtzy/data_to_viz/master/Example_dataset/1_OneNum.csv", function(data) {

  // X axis: scale and draw:
  var x = d3.scaleLinear()
      .domain([0, 1000])     // can use this instead of 1000 to have the max of data: d3.max(data, function(d) { return +d.price })
      .range([0, width]);
  svg.append("g")
      .attr("transform", "translate(0," + height + ")")
      .call(d3.axisBottom(x));

  // Y axis: initialization
  var y = d3.scaleLinear()
      .range([height, 0]);
  var yAxis = svg.append("g")
```

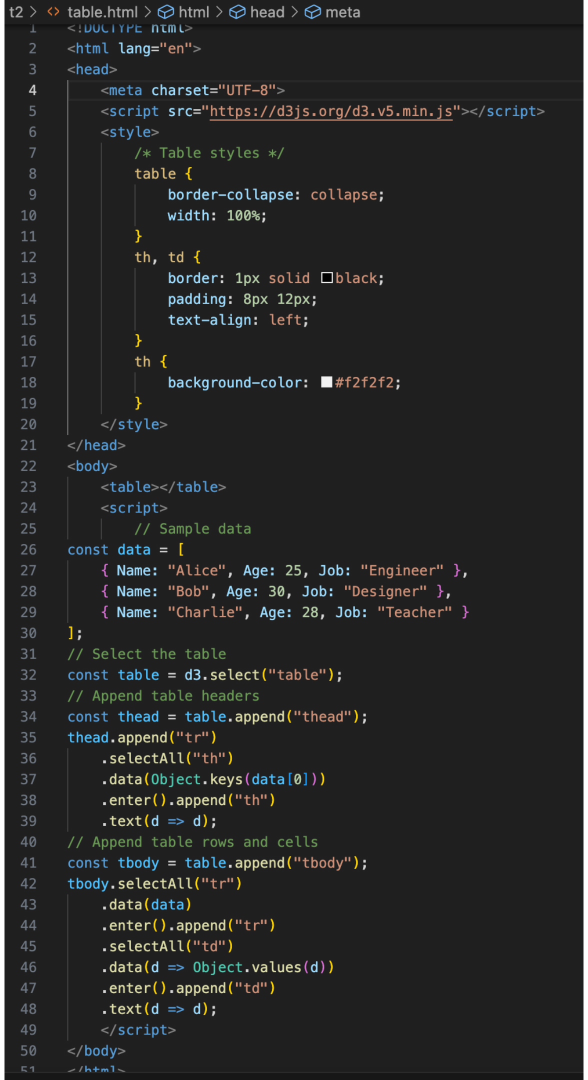


```javascript
  // A function that builds the graph for a specific value of bin
  function update(nBin) {
    // set the parameters for the histogram
    var histogram = d3.histogram()
        .value(function(d) { return d.price; })   // I need to give the vector of value
        .domain(x.domain())  // then the domain of the graphic
        .thresholds(x.ticks(nBin)); // then the numbers of bins

    // And apply this function to data to get the bins
    var bins = histogram(data);

    // Y axis: update now that we know the domain
    y.domain([0, d3.max(bins, function(d) { return d.length; })]);   // d3.hist has to be called before the Y axis obviously
    yAxis
        .transition()
        .duration(1000)
        .call(d3.axisLeft(y));
    // Join the rect with the bins data
    var u = svg.selectAll("rect")
        .data(bins)
    // Manage the existing bars and eventually the new ones:
    u
        .enter()
        .append("rect") // Add a new rect for each new elements
        .merge(u) // get the already existing elements as well
        .transition() // and apply changes to all of them
        .duration(1000)
          .attr("x", 1)
          .attr("transform", function(d) { return "translate(" + x(d.x0) + "," + y(d.length) + ")"; })
          .attr("width", function(d) { return x(d.x1) - x(d.x0) -1 ; })
          .attr("height", function(d) { return height - y(d.length); })
          .style("fill", "#69b3a2")
    // If less bar in the new histogram, I delete the ones not in use anymore
    u
        .exit()
        .remove()
    }
  // Initialize with 20 bins
  update(20)

  // Listen to the button -> update if user change it
  d3.select("#nBin").on("input", function() {
    update(+this.value);
  });

});
</script>
```

Table example with CSS style

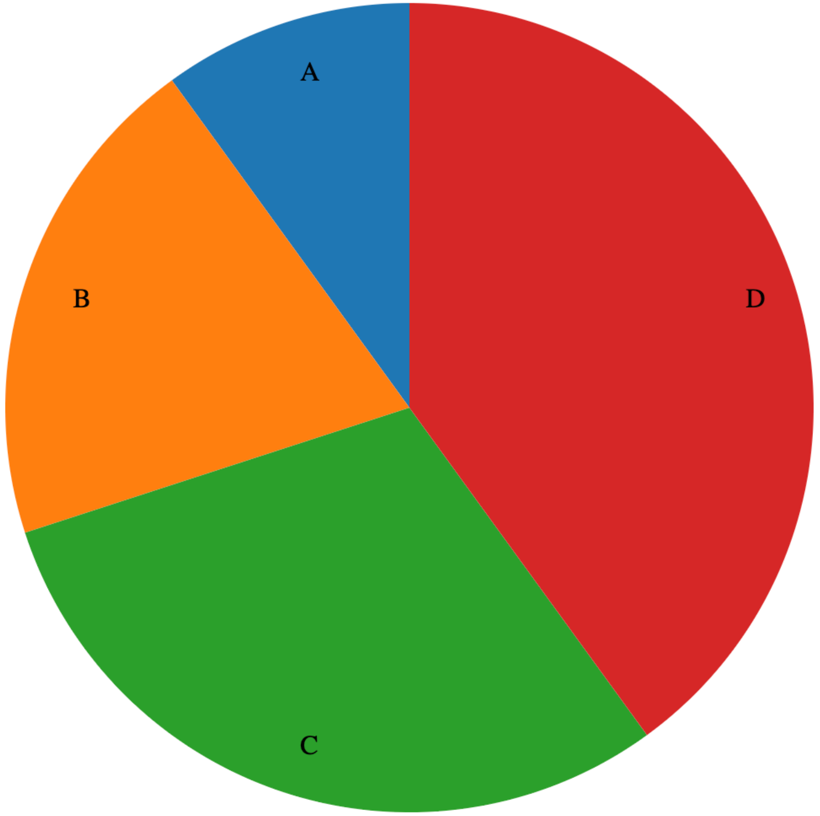


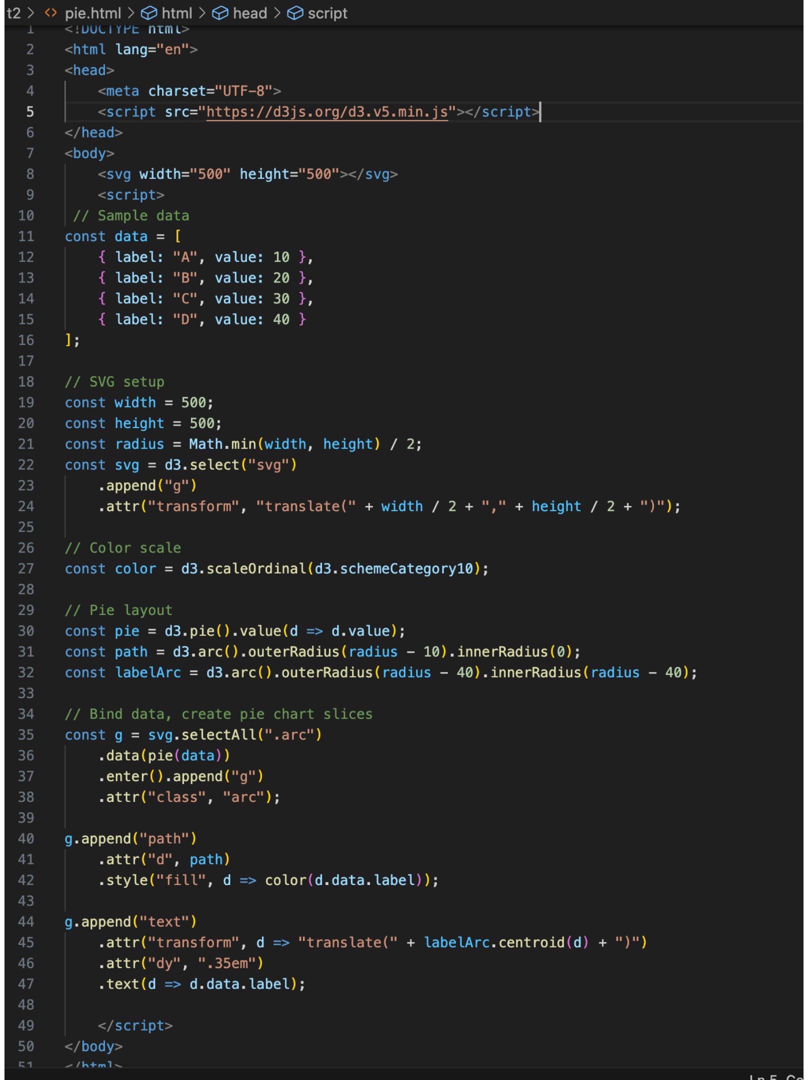
Name	Age	Job
Alice	25	Engineer
Bob	30	Designer
Charlie	28	Teacher



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script src="https://d3js.org/d3.v5.min.js"></script>
    <style>
        /* Table styles */
        table {
            border-collapse: collapse;
            width: 100%;
        }
        th, td {
            border: 1px solid black;
            padding: 8px 12px;
            text-align: left;
        }
        th {
            background-color: #f2f2f2;
        }
    </style>
</head>
<body>
    <table></table>
    <script>
        // Sample data
const data = [
    { Name: "Alice", Age: 25, Job: "Engineer" },
    { Name: "Bob", Age: 30, Job: "Designer" },
    { Name: "Charlie", Age: 28, Job: "Teacher" }
];
// Select the table
const table = d3.select("table");
// Append table headers
const thead = table.append("thead");
thead.append("tr")
    .selectAll("th")
    .data(Object.keys(data[0]))
    .enter().append("th")
    .text(d => d);
// Append table rows and cells
const tbody = table.append("tbody");
tbody.selectAll("tr")
    .data(data)
    .enter().append("tr")
    .selectAll("td")
    .data(d => Object.values(d))
    .enter().append("td")
    .text(d => d);
    </script>
</body>
</html>
```

Pie Chart example





```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <script src="https://d3js.org/d3.v5.min.js"></script>
</head>
<body>
    <svg width="500" height="500"></svg>
    <script>
 // Sample data
const data = [
    { label: "A", value: 10 },
    { label: "B", value: 20 },
    { label: "C", value: 30 },
    { label: "D", value: 40 }
];

// SVG setup
const width = 500;
const height = 500;
const radius = Math.min(width, height) / 2;
const svg = d3.select("svg")
    .append("g")
    .attr("transform", "translate(" + width / 2 + "," + height / 2 + ")");

// Color scale
const color = d3.scaleOrdinal(d3.schemeCategory10);

// Pie layout
const pie = d3.pie().value(d => d.value);
const path = d3.arc().outerRadius(radius - 10).innerRadius(0);
const labelArc = d3.arc().outerRadius(radius - 40).innerRadius(radius - 40);

// Bind data, create pie chart slices
const g = svg.selectAll(".arc")
    .data(pie(data))
    .enter().append("g")
    .attr("class", "arc");

g.append("path")
    .attr("d", path)
    .style("fill", d => color(d.data.label));

g.append("text")
    .attr("transform", d => "translate(" + labelArc.centroid(d) + ")")
    .attr("dy", ".35em")
    .text(d => d.data.label);

    </script>
</body>
</html>
```

References

- https://www.w3schools.com/js/js_htmldom.asp
- Vitaly Shmatikov CS 345 Introduction to JavaScript
- Sarbjit Kaur Introduction to HTML
- D3.js
- https://d3js.org/
- https://developer.mozilla.org/en-US/docs/web/SVG
- https://livebook.manning.com/book/d3js-in-action-second-edition/chapter-7/70
- https://d3-graph-gallery.com/index.html

Next tutorial :

- Graph and Network Analysis using D3.js
- Hosting webpages on Apache Webserver

THANK YOU !!