



EECS E6893 Big Data Analytics

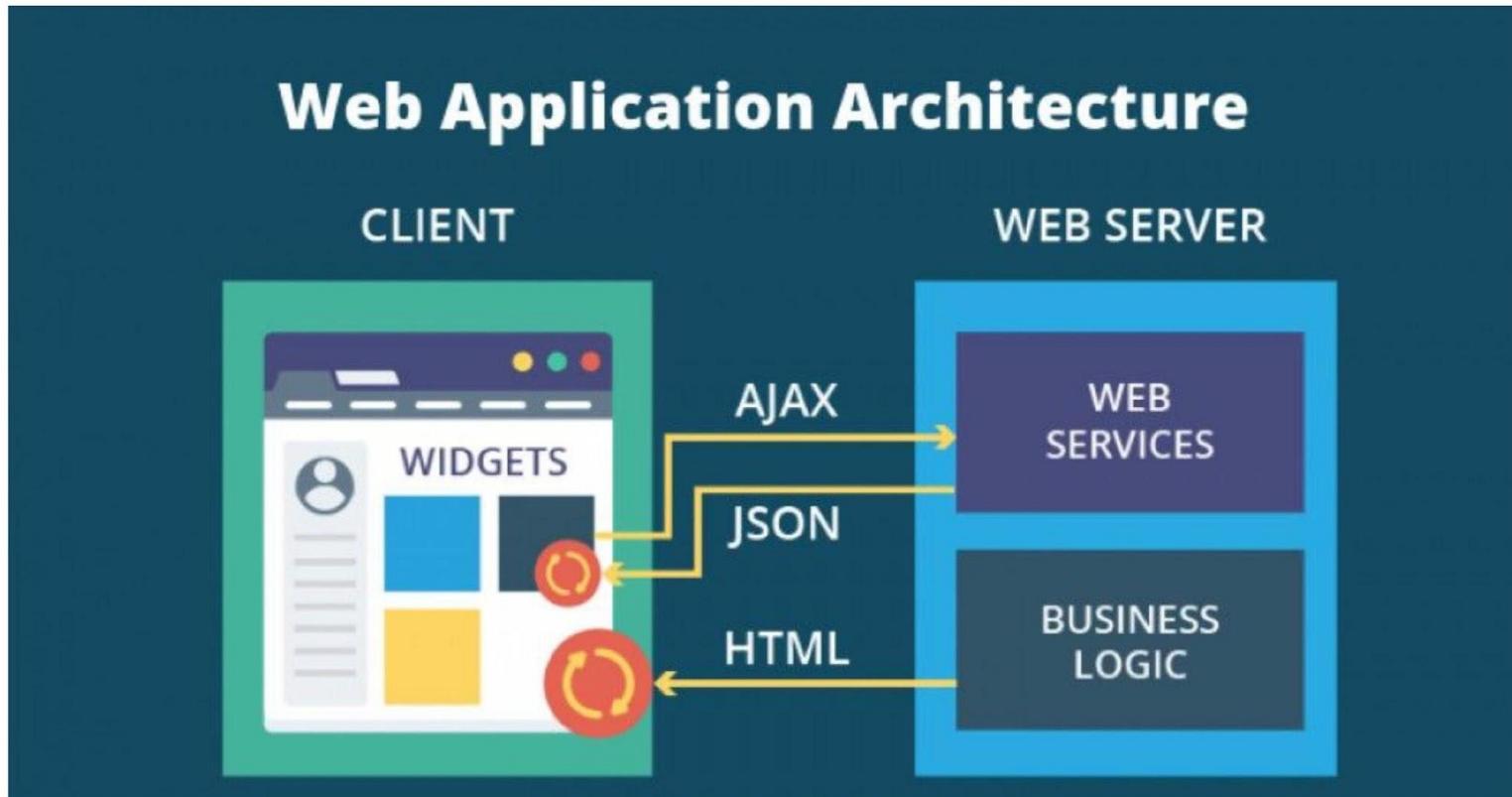
HW3: Data visualization

Srividya Inampudi, si2396@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
- Introduction of Apache HTTP Server

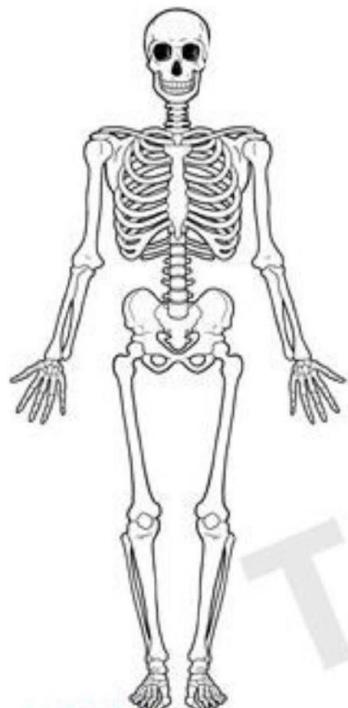
Web Application



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.

WEB DESIGNING



HTML (Structure)



CSS (Presentation)



Javascript (functionality)
tutorial.techaltum.com

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

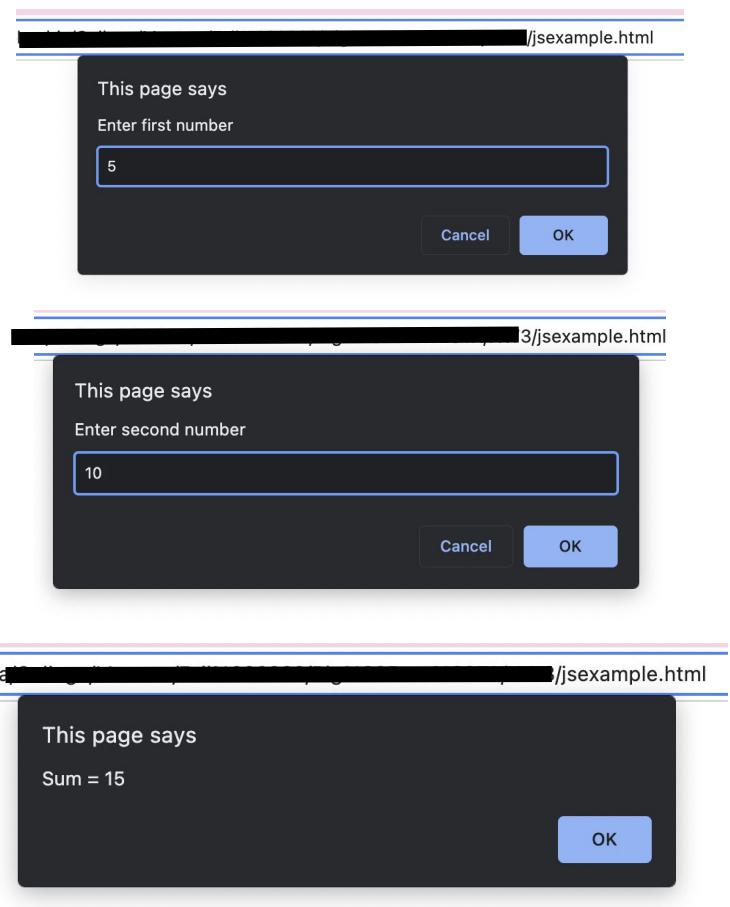
- Form validation
- Page embellishments and special effects
- Navigation systems
- Basic math calculations
- Dynamic content manipulation
- Sample applications
 - Dashboard widgets in Mac OS X, Google Maps, Philips universal remotes, Writely word processor, hundreds of others...

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
 - ``
- Pseudo-URL referenced by a link
 - `Click me`

Example 1: Add Two Numbers

```
< jsexample.html > ...
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Adding two numbers</title>
5      </head>
6  <body>
7      <script>
8          var num1, num2, sum
9          num1 = prompt("Enter first number")
10         num2 = prompt("Enter second number")
11         sum = parseInt(num1) + parseInt(num2)
12         alert("Sum = " + sum)
13     </script>
14
15 </body>
16 </html>
17
```



Example 2: Page Manipulation

```
< jsexample.html > ...
1  <!DOCTYPE html>
2  <html>
3      <body>
4          <h1>Element Object</h1>
5          <h2>appendChild() Method</h2>
6
7          <ul id="myList">
8              <li>Car</li>
9              <li>Bike</li>
10             </ul>
11
12             <p>Click "Append" to append an item to the end of the list:</p>
13
14             <button onclick="myFunction()">Append</button>
15
16             <script>
17                 function myFunction() {
18                     const node = document.createElement("li");
19                     const textnode = document.createTextNode("Bus");
20                     node.appendChild(textnode);
21                     document.getElementById("myList").appendChild(node);
22                 }
23             </script>
24
25             </body>
26         </html>
```

Element Object

appendChild() Method

- Car
- Bike

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

Append

Element Object

appendChild() Method

- Car
- Bike
- Bus

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

Append

Language Basics

- JavaScript is case sensitive
 - 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

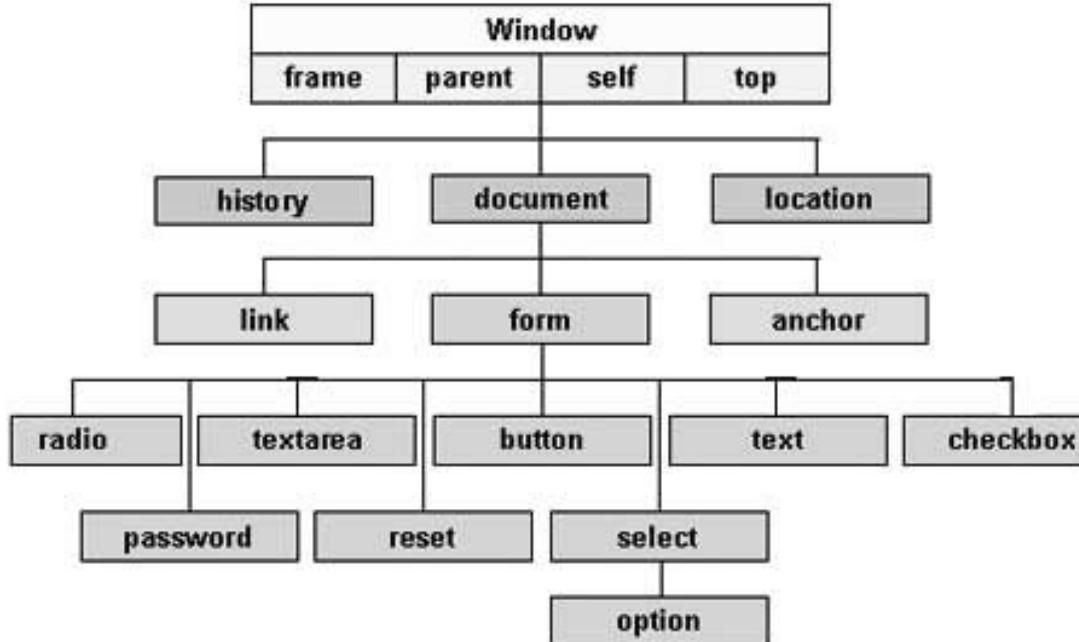
- An object is a collection of named properties
- Think of it as an associative array or hash table
 - Set of name:value pairs
 - `objBob = {name: "Bob", grade: 'A', level: 3};`
 - Play a role similar to lists in Lisp / Scheme
- New members can be added at any time
 - `objBob.fullname = 'Robert';`
-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



JavaScript Document Object Model (DOM) hierarchy

Ref: https://www.tutorialspoint.com/javascript/javascript_html_dom.htm

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.

Introduction to HTML

- HTML is a language for describing web pages.
- HTML stands for **Hyper Text Markup Language**
- 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

World Wide Web

- The **World Wide Web** (abbreviated as **WWW** or **W3** and commonly known as **the Web**) is a system of interlinked hypertext documents accessed via the Internet. With a web browser, one can view web pages that may contain text, images, videos, and other multimedia and navigate between them via hyperlinks.

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 angle brackets. It is a piece of text 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 companion 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:
,<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:

```
<> myfirstpage.html > ⚒ html
1   <!DOCTYPE html>
2   <html>
3
4       <head>
5           <title>Page Title</title>
6       </head>
7
8       <body>
9           <h1>This is a Heading</h1>
10          <p>This is a paragraph.</p>
11      </body>
12  </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.

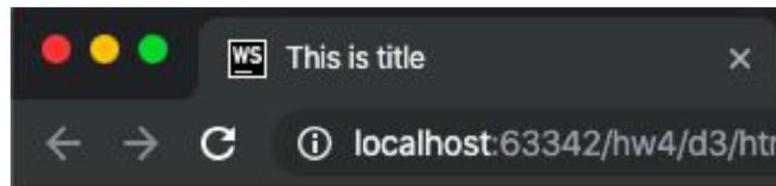
HTML

```
<!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.martinrinhart.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](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

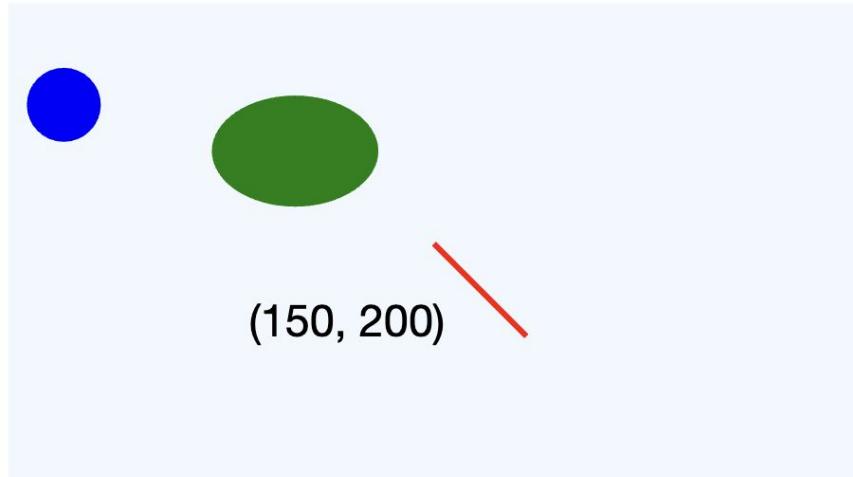
Header 3

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

D3.js



D3.js is a JavaScript library for manipulating documents based on data. It helps you bring data to life using HTML, SVG, and CSS. It provides a data-driven approach to DOM manipulation.

Visit <https://d3js.org> for more tutorials!

What is D3?

- **D3.js** is a JavaScript library for manipulating documents based on data. (<https://d3js.org/>)
- At its core, D3 is a graphics library for the web
- D3 is a targeted library – data visualization
- D3 visualizations can be embedded into any web page
- D3 Links:
 - Homepage: <https://d3js.org/>
 - Github: <https://github.com/d3/d3>
 - Latest Online Release: <https://d3js.org/d3.v5.min.js>

D3 Live Examples

Gallery

Mike Bostock edited this page on Oct 27, 2018 · 1287 revisions

Wiki ▸ Gallery

Welcome to the D3 gallery! More examples are available for forking on [Observable](#); see [my profile](#) and the [visualization collection](#). Feel free to publish and share your own!

Visual Index

Box Plots
Bubble Chart
Bullet Charts
Calendar View
Non-contiguous Cartogram
Chord Diagram
Dendrogram
Force-Directed Graph
Circle Packing
Population Pyramid 2000
Stacked Bars
Streamgraph

Pages 62

Data-Driven Documents

- Home
- Gallery
- Examples
- Tutorials
- Plugins

Help

- Stack Overflow
- Slack
- Google Group
- Gitter

- D3 Examples page:
<https://github.com/d3/d3/wiki/Gallery>
- Can there be more content on web page than just a D3 visualization?
- Does D3 support interaction?
- Can there be multiple D3 visualizations on one web page?

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.v5.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
1   <!DOCTYPE html>
2   <html>
3
4   |   <head>
5   |       <script src="https://d3js.org/d3.v4.js"></script>
6   |   </head>
7
8   |   <body>
9   |       <p>Hello</p>
10  |   </body>
11 </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



Hello

Example: Simple Bar Chart

```
<svg id="svg2"></svg>>
<script src="https://d3js.org/d3.v4.min.js"></script>
<script>
  var data = [10, 20, 30, 40, 50];
  var svgWidth = 640, svgHeight = 320;

  var svg = d3.select('svg')
    .attr("width", svgWidth)
    .attr("height", svgHeight);

  var barChart = svg.selectAll("rect")
    .data(data)
    .enter()
    .append("rect")
    .attr("class", "bar")
    .attr('x', 20)
    .attr('y', function(d,i){return i*30+100})
    .attr('height', 25)
    .attr('width', function(d){return d});

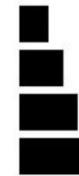
</script>>
```

Header 1

Header 2

Header 3

Paragraph 1



```
var data = [10,20,30,40,50];
var svgWidth = 640, svgHeight = 320;
```

Declaration of data and variables

```
var svg = d3.select('svg')
    .attr("width", svgWidth)
    .attr("height", svgHeight);
```

D3 provides operating on arbitrary sets of nodes called *selections*. You can manipulate individual nodes and set the attributes

```
var barChart = svg.selectAll("rect")
    .data(data)
    .enter()
    .append("rect")
    .attr("class", "bar")
    .attr('x', 20)
    .attr('y', function(d,i){return i*30+100})
    .attr('height', 25)
    .attr('width', function(d){return d});
print>>
```

Tricky part of D3: Once you bound data with selection, each element in the data array is paired with the corresponding node in the selection. If there are fewer nodes than data, you can use enter() to appending nodes.

Again, please visit <https://d3js.org> for more tutorials!

Hosting webpage on Apache web server using virtual host (MAC users)

Step 1: Install xcode

```
(base) srividayainampudi@Srividyas-Air ~ % xcode-select --install
xcode-select: error: command line tools are already installed, use "Software Update" to install updates
(base) srividayainampudi@Srividyas-Air ~ % █
```

Step 2: Install Homebrew

Go to brew.sh in your browser and copy the command there to your terminal-

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

```
(base) srividayainampudi@Srividyas-Air ~ % /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
⇒ Checking for `sudo` access (which may request your password)...
⇒ This script will install:
/opt/homebrew/bin/brew
/opt/homebrew/share/doc/homebrew
/opt/homebrew/share/man/man1/brew.1
/opt/homebrew/share/zsh/site-functions/_brew
/opt/homebrew/etc/bash_completion.d/brew
/opt/homebrew
```

⇒ Installation successful!

⇒ Homebrew has enabled anonymous aggregate formulae and cask analytics.

Read the analytics documentation (and how to opt-out) here:

<https://docs.brew.sh/Analytics>

No analytics data has been sent yet (nor will any be during this `install` run).

⇒ Homebrew is run entirely by unpaid volunteers. Please consider donating:

<https://github.com/Homebrew/brew#donations>

⇒ Next steps:

- Run these three commands in your terminal to add Homebrew to your PATH:

```
echo '# Set PATH, MANPATH, etc., for Homebrew.' >> /Users/srividyaampudi/.zprofile
echo 'eval "$( $(/opt/homebrew/bin/brew shellenv)"' >> /Users/srividyaampudi/.zprofile
eval "$( $(/opt/homebrew/bin/brew shellenv)"'
```

- Run `brew help` to get started

- Further documentation:

<https://docs.brew.sh>

Step 3: Add Homebrew to your PATH

Follow instructions after installation

```
(base) srividyaampudi@Srividya-Air ~ % echo '# Set PATH, MANPATH, etc., for Homebrew.' >> /Users/srividyaampudi/.zprofile
(base) srividyaampudi@Srividya-Air ~ % echo 'eval "$( $(/opt/homebrew/bin/brew shellenv)"' >> /Users/srividyaampudi/.zprofile
(base) srividyaampudi@Srividya-Air ~ % eval "$( $(/opt/homebrew/bin/brew shellenv)"'
(base) srividyaampudi@Srividya-Air ~ % █
```

Step 4: Install apache2

Command - brew install apache2

```
(base) srividya.inampudi@Srividya-Air ~ % brew install apache2
==> Downloading https://ghcr.io/v2/homebrew/core/apr/manifests/1.7.0_3
#####
100.0%
==> Downloading https://ghcr.io/v2/homebrew/core/apr/blobs/sha256:02e6b44b3284fa471cce15592a8666
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sha256:02e6b44b328
#####
100.0%
```

Step 5: Start apache server

```
(base) srividya.inampudi@Srividya-Air ~ % sudo apachectl start
(base) srividya.inampudi@Srividya-Air ~ % █
```

This will start Apache HTTP server which can be tested by visiting localhost on the browser. The localhost 8080 gives the response as shown below:

← → C ⓘ localhost:8080

It works!

Step 6: Open httpd config file

- If you're using Intel-based Mac: vim /usr/local/etc/httpd/httpd.conf
- If you're using Mac with Apple Silicon: vim /opt/homebrew/etc/httpd/httpd.conf

Step 7: Update these lines

Listen 8080 to Listen 80

DocumentRoot "/usr/local/var/www" to DocumentRoot "/Users/your_account/Sites"

<Directory "/usr/local/var/www"> to <Directory "/Users/your_account/Sites">

AllowOverride None to AllowOverride All

```
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses.
#
#Listen 12.34.56.78:80
Listen 80
```

```
#  
DocumentRoot "/Users/srividya.inampudi/Sites"  
<Directory "/Users/srividya.inampudi/Sites">  
#  
# Possible values for the Options directive are "None", "All",  
# or any combination of:  
#   Indexes Includes FollowSymLinks SymLinksIfOwnerMatch ExecCGI MultiViews  
#  
# Note that "MultiViews" must be named *explicitly* --- "Options All"  
# doesn't give it to you.  
#  
# The Options directive is both complicated and important. Please see  
# http://httpd.apache.org/docs/2.4/mod/core.html#options  
# for more information.  
#  
Options Indexes FollowSymLinks  
  
#  
# AllowOverride controls what directives may be placed in .htaccess files.  
# It can be "All", "None", or any combination of the keywords:  
#   AllowOverride FileInfo AuthConfig Limit  
#  
AllowOverride All  
  
#  
# Controls who can get stuff from this server.  
#
```

Step 8: Uncomment this line

LoadModule rewrite_module lib/httpd/modules/mod_rewrite.so

Update these lines

User _www to User your_account

Group _www to Group staff

```
LoadModule alias_module lib/httpd/modules/mod_alias.so
LoadModule rewrite_module lib/httpd/modules/mod_rewrite.so

<IfModule unixd_module>
#
# If you wish httpd to run as a different user or group, you must run
# httpd as root initially and it will switch.
#
# User/Group: The name (or #number) of the user/group to run httpd as.
# It is usually good practice to create a dedicated user and group for
# running httpd, as with most system services.
#
User srividyainpudi
Group staff
```

Step 9: Update this line

ServerName www.example.com:8080 to ServerName localhost

```
# If your host doesn't have a registered DNS name, enter its IP address here.
#
ServerName localhost

#
# Deny access to the entirety of your server's filesystem. You must
# explicitly permit access to web content directories in other
# <Directory> blocks below.
#
```

Step 10: Create Sites folder and add your html file to the folder

```
(base) srividya@Srividya-Air ~ % mkdir Sites  
(base) srividya@Srividya-Air ~ %  
(base) srividya@Srividya-Air ~ %  
(base) srividya@Srividya-Air ~ % cd Sites  
(base) srividya@Srividya-Air Sites % ls  
jsexample.html
```

Step 11. Restart apache

```
(base) srividya@Srividya-Air Sites % sudo apachectl stop  
(base) srividya@Srividya-Air Sites %  
(base) srividya@Srividya-Air Sites %  
(base) srividya@Srividya-Air Sites %  
(base) srividya@Srividya-Air Sites % sudo apachectl start
```

Step 12: Test localhost to see your hosted website

<http://localhost>



Sample HTML to host on apache webserver

- Car
 - Ford

Step 13: Stop apache

```
(base) srividyainampudi@Srividyas-Air Sites % sudo apachectl stop  
Password:  
(base) srividyainampudi@Srividyas-Air Sites %
```

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>

Acknowledgement

Part of the slides is credited to Juncai Liu and Yiwen Fang

THANK YOU