Intro to HTML & Javascript
(with a little CSS, +libraries/frameworks)

CS171 – Lab 1

1/31/14

Introduction for today

- HTML & CSS
  - Build and style a basic webpage

- Javascript
  - Introduce & review Javascript syntax and style.
  - Link to libraries & start writing Javascript.
  - Learn about the console and how to use Chrome's inspector tool.
  - Learn basic techniques to debugging Javascript

- Homework 1
  - Get started with running a basic webserver & understanding the code

Before we begin

- Useful development tools.
- Target platform for all problem sets.
- Your work (assignments, etc) **must be functional** in Chrome.

What is HTML?

What is HTML?

- Hypertext Markup Language (HTML)
- *Markup* language, not a programming language.
  - Provides instruction on how to render a page.

- HTML is consists of *elements* built with *start tags and end tags.*
  - *Start tag example:* <head>
  - *End tag example:* </head>
  - Element example: <head> …. (stuff goes here) …. </head>

In HTML, all *elements* are formed via *tags.*

In HTML, all *elements* are formed via *tags.*

- HTML includes elements which include images, objects, videos, lists, links, quotes, and other items….. there are many types of tags!
  - (use google!)

- Tags also include *attributes*.
  - Example: *anchor tag.*
  - *<a>*
  - Adding the href attribute allows for the creation of a link
  - CS50.

HTML tags you might need to know

<title>My webpage</title>
Puts the title of the web page in the title bar of your browser.

My link
Creates links! Use the href attribute to represent the url.

<img> or <img />
Indicates that an image will be shown on the page. Use the src attribute to link to the image, use the alt attribute is used to provide a short description to the image.

** This is a paragraph! **
Creates paragraph text in the document.

<h1> to <h6>
<h1>Some header</h1>
Provides header structure to your text. <h1> is the most important heading, <h2> is less important, etc

**
...
**
Creates an ordered list (like a numbered list)

**
...
**
Creates an unordered list (like bullet points)

**
List item **
Creates an item that belongs to a list

<html>...</html>
Creates the entire HTML container.

<head>...</head>
Creates the header (generally where the title and links to style sheets/scripts are found)

<body>...</body>
Creates the section of html that contains content

<table>...</table>
Creates a table in the document.

<tr>...</tr>
Specifies a table row in a table.

<td>..</tr>
Specifices a table cell in a table row

<link rel="stylesheet" type="text/css" href="theme.css">
Points to an external file is linked to the current html document (often used for CSS external stylesheets)

** ... **
A division of a page. Used as an additional means to provide structure to HTML.

<script>...</script>
Includes text inside this tag as script, most commonly text/javascript.

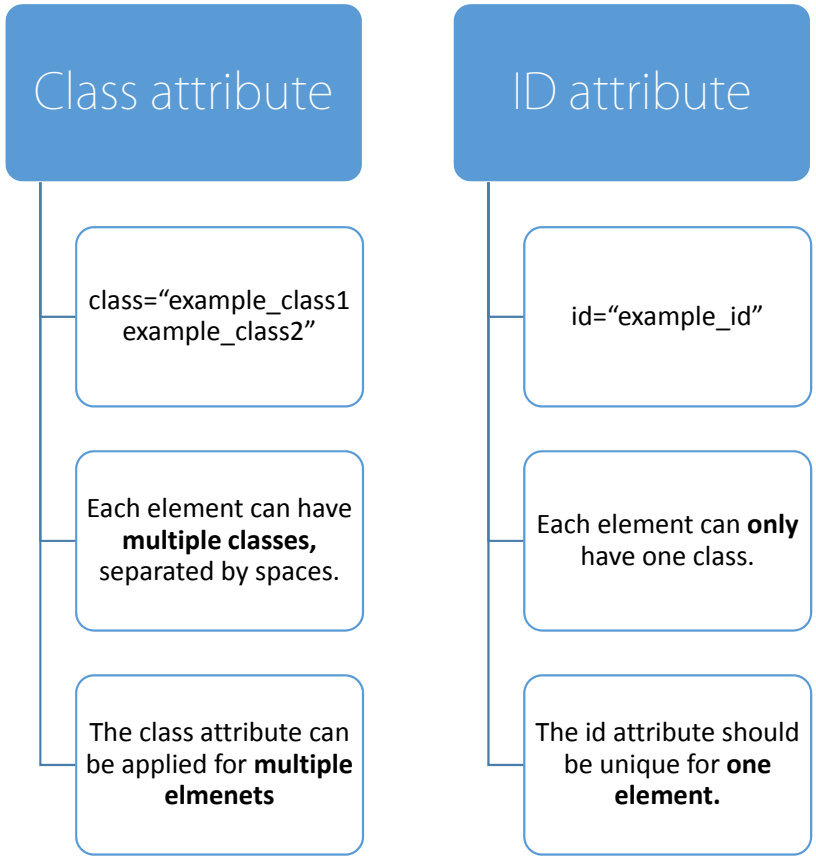
<form>..</form>
Area enclosed by this tag is an HTML form that can accept user input.

<input></input> or <input />
Used inside an HTML form and is used to accept user input or submit the input, has attributes type and label which specify type of form input element.

<!-- ... –>
HTML comment tag. Used to add text to your document that will not be displayed in the browser and is useful to document the page.

In HTML, *all* elements can have attributes

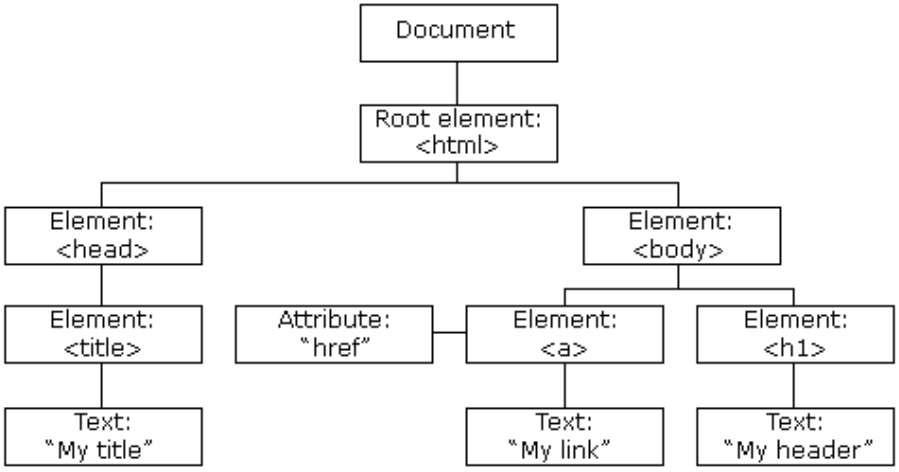
- HTML elements commonly have **class** and **id** attributes.
  - These attributes are included inside the opening tag.
  - `<h1 id="main_header">Visualization of Taxes</h1>`
  - `<a class="navigation_link header_link">Menu</a>`
  - `<a href="https://www.cs50.net/" id="cs50_link" class="menu_link">CS50</a>`
- These attributes are used to 'select' HTML elements later through CSS / Javascript



HTML follows a tree structure

- Your web browser renders HTML with a Document Object Model (DOM):
  - Note: objects in the DOM tree can be added / removed / manipulated by Javascript
- We'll check this out later in Chrome's element inspector

```
<html>
    <head>
        <title>My title</title>
    </head>
    <body>
        <a href="">My link</a>
       <h1>My header</h1>
    </body>
</html>
```



Getting Started with HTML

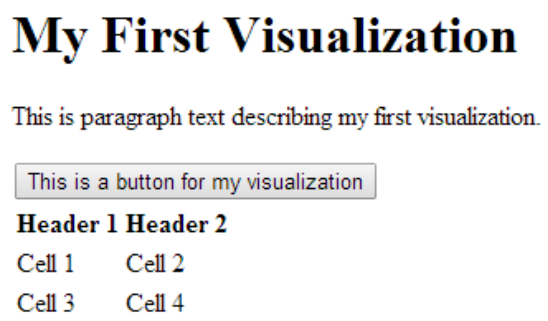
- You don’t need to memorize all your tags!
  - But you should know how to use class and id attributes for HTML elements.

  These attributes will help ‘label’ HTML elements so you can do things with them.

Open up your favorite text editor and create a new file called ‘index.html.’

This doesn't look that good.

```
<html>
    <head>
    <title>My First Visualization!</title>

    <!-- My first visualization, created by __; 1/28/14 -->

    </head>
    <body>
        <h1>My First Visualization</h1>
        <p>This is paragraph text describing my first visualization.</p>
              <button>This is a button for my visualization</button>
        <table>
            <thead>
                <tr>
                    <th>Header 1</th>
                    <th>Header 2</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>Cell 1</td>
                    <td>Cell 2</td>
                </tr>
                <tr>
                    <td>Cell 3</td>
                    <td>Cell 4</td>
                 </tr>
             </tbody>
         </table>

    </body>
</html>
```



What is CSS?

What is CSS?

- Cascading Style Sheets
  - Stylesheet language for the web, makes webpages look good.
  - Includes the fonts, colors, and many other properties for web pages.
- CSS is applied to HTML elements, with CSS properties

- Examples of CSS properties:
  - `color:red`
  - `background-color:#fff;`
  - `padding:20px;`
  - `font-size:18px;`
  - `display:none;`

How do you start using CSS?

Inline (inside the HTML):

```
<table style="margin-top:20px;">…</table>
```

In the document:

```
<style type="text/css">
     #datatable {
     margin-top:20px;
     }
</style>
```

In a separate file:

```
<link href="path/to/file.css" rel="stylesheet">
```

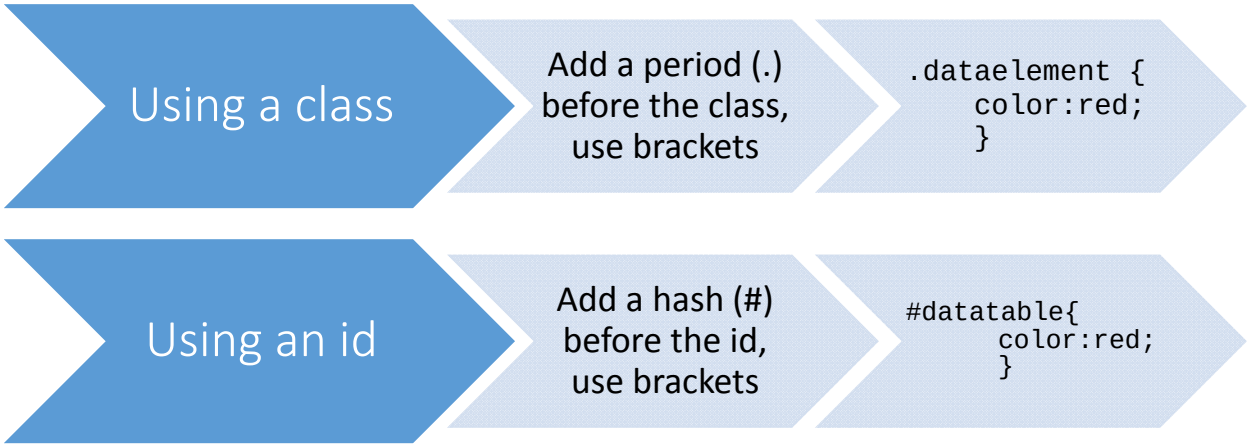
How does external CSS work?

In a Cascading Style Sheet file you 'select' elements based on class and id attributes in the HTML, then apply properties.

1. Add 'class' and 'id' attributes to HTML elements:

   Ex: `<table>` to `<table id="datatable1" class="dataelement">`

2. Add CSS *selectors* to these classes/ids in the CSS document:



Learning CSS

There are a lot of different types of selectors and CSS properties, we won't have time to cover it all. (google is your friend)

CSS Demo

1. Create a new file called 'styles.css' in the same directory as the previous HTML file you worked on
2. Add this to your header:

```
<link href="styles.css" rel="stylesheet">
```

Chrome Developer Tools

Google Chrome has a set of tools that makes styling and visualizing the HTML document easier:

Menu > Tools > ‘Developer Tools’

or

Ctrl + Shift + I

or

Right click on webpage > Inspect element

What is Javascript?

What is Javascript?

- A programming language for web browsers generally executed client-side (*in* your web browser)
- Javascript code is written in ‘scripts’ which are run when the page renders.
- There are two ways to include Javascript
  - In the document:

```
<script type="text/javascript">
        alert(“This is your first line of Javascript!”!);
</script>
```

  - **In a separate file:**

```
<script src=“viz.js"></script>
```

(this simply loads the content of the file as if it were Javascript)

A Javascript file is simply a text file with a .js extension which contains Javascript.

Why Javascript?

- Javascript can do a lot!

With the object model, JavaScript gets all the power it needs to create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

W3schools.com

- Example: responding to clicks, adding svg (graphics) elements, creating elements from data, etc.

Javascript 101

- Check out the browser methods at: http://overapi.com/javascript/
- If you are completely new, also refer to the tutorial at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide
- For those experienced with programming, the syntax is similar to C and PHP.

Programming (with Javascript) can be frustrating!

Javascript 101 – First Steps

This is actually just HTML telling the browser that Javascript is coming.

```
<script>
      alert(“hello world!”);

      //This is your first comment!
</script>
```

This is a line of Javascript. Here we call the alert() function with the string “hello world!” The alert() function displays a message box in your browser.

You can create comments in Javscript by adding ‘//’ to the beginning of the line. The browser will ignore these lines.

You can include script tags anywhere in your document. The page will execute the Javascript in the order that is included in the page.

Javascript 101 – Functions

- Functions are sets of code that are executed. To execute a piece of code, you *call a function*. You can also *call a function* with *arguments.*
- Some functions are already defined by the browser (this is what gives us interactivity with the web browser)!
- In the earlier example, we called the function *alert* with the argument *hello world*.

```
<script>
        alert(“hello world!”);
</script>
//(the function alert is already defined in the browser)
```

Javascript 101 – Functions (cont)

Two ways to define functions in Javascript:

```
function myfirstfunction(){
//some code to be executed
}
```

```
var myfirstfunction = function(){
//some code to be executed
}
```

To run the code in the function, simply call

```
<script>
	myfirstfunction();
</script>
```

Javascript 101 – Functions (cont., 2)

You don’t always need functions!

Any code in the script tag is automatically run as the page ‘reaches’ the script tag.

```
<script>
        alert(“Hello world”);
        //This code is run as the page redners
</script>
```

Why use functions?

- To pass in arguments and prevent code redundancy/repetition.
- To use functions that have already been written for you.
- To make your code better! (and other reasons)

Javascript 101 – Variables

Javascript allows you to store variables (values) to be used and manipulated later. Creating a variable involves *declaring* a variable and *naming* the variable. Javascript is dynamically typed (so you don't need to declare what *type* of value it is):

```
<script>
        var num1 = 1;
        var num2 = 2;
        var string = "two";
        var bool = false;
 </script>
```

Do you need to use var when declaring the variable for the first time?

Generally, yes. While your code may work without the 'var' declaration, the scope of your variable is global. Unless you know what you're doing, always use var.

Javascript 101 – Variable Scope

Javascript variables have the scope in which they are declared:

```
<script>
    var g = "global";
    function go() {
        var l = "local";
    }

    go();

    alert(g); // alerts with ‘global’
    alert(l); // throws a reference error
</script>
```

Javascript 101 – Basic Operations in JS

```
<script>
	var num1 = 1;
	var num2 = 2;
	var str2 = “two”;
	var str3 = “three”;
	var bool = false;

	var num3 = num1 * num2;	// 2
	var foo = str2 + str3;	// twothree
	var bar = !bool;		// true
	var baz = num1 + str3;	// 1three

</script>
```

Javascript 101 – Arrays

Javascript allows you to create **arrays** which can store related values. Creating an array is similar to creating a variable:

```
<script>
	var days = [‘Mon’, ‘Tue’, ‘Wed’]
	alert(days[0]); //alerts user with ‘Mon’
	var foo = days[1]+days[2]; // TueWed
	alert(days.length); //alerts user with 3
</script>
```

Arrays are zero-indexed.

Which means that the *first* item in the array has an index value of zero.

Arrays have a length property.

To access the length of a list, use (arrayname).length

Javascript 101 – Arrays (2)

JavaScript includes a number of methods for array manipulation:

- Array.shift()
  - Removes the first item from an array, returns the removed element
- Array.pop()
  - Removes and returns the last item
- Array.push()
  - Adds one (or more) items to the end of an array

- …see documentation for more.

Javascript 101 – Logic and Control (1)

If statements

How it works

```
if (condition)
{
        // do this
}
else if (condition)
{
        //do something else
}
else
{
        //do something different
}
```

Example

```
<script>
        var days = [‘Mon’, ‘Tue’, ‘Wed’];
        if (days[0]==‘Mon’)
        {
                alert(“Monday”);
        }
        else if (days[0]==‘Tue’)
        {
                alert(“Tuesday”)
        }
        else
        {
                alert(“Wednesday”);
        }
</script>
```

Javascript 101 – Logic and Control (3)

Different types of loops

How it works

```
//For loops
for(initializations; condition; updates)
{
        //do something repeatedly
}



//While loops
while   (condition)
{
        //do something repeatedly
}
```

Example

```
<script>
        var days = [‘Mon’, ‘Tue’, ‘Wed’];
        //For loop 1
        for(var i in array){alert(days[i]);}

        //For loop 2
        for(var i=0;i<days.length;i++){
                alert(days[i]);
        }

        //While loop     var i=0;
        while(i<days.length){
                alert(days[i]);
                i++; //Prevent infinite loops
        }
</script>
```

Javascript 101 – The Console

Google Chrome has a Javascript console that makes debugging (find errors in your code) and interacting with your Javascript easier:

Menu > Tools > ‘JavaScript Console’

or

Ctrl + Shift + J

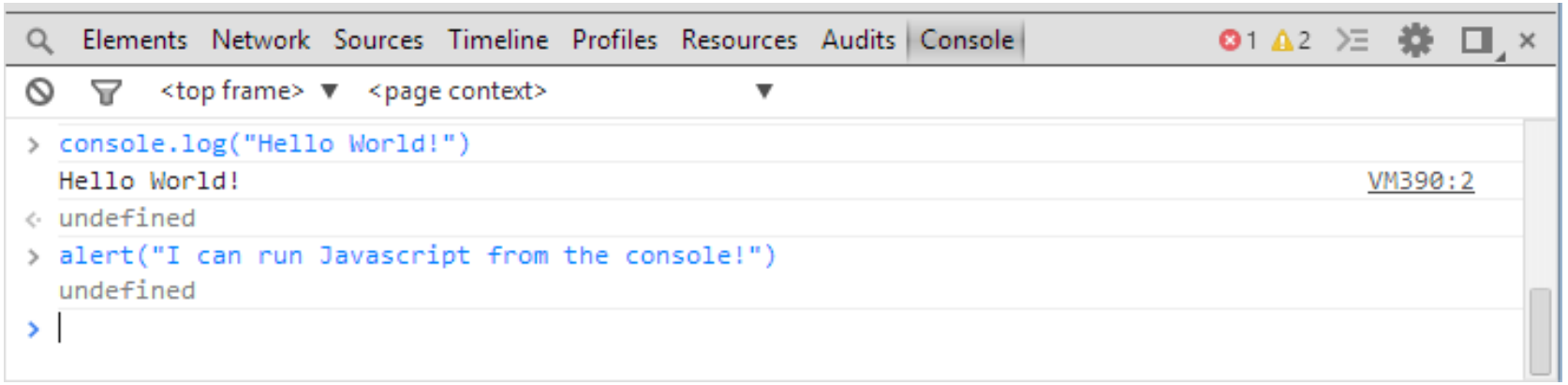
or

Right click on webpage > Inspect element > Console

Javascript 101-Using the Console

In the console:

- You will see errors that occurred while running your Javascript
- Messages that you sent yourself via the console.log() function
- The output from Javascript that you run in the console.



Javascript 101-Using the Console

The console.log function is incredibly useful.

Try it! (in the console, you don't need to write script tags, just type each statement in)

```
<script>
        var days = [‘Mon’, ‘Tue’, ‘Wed’];

        for(var i in days){
                console.log(days[i]);
        }
</script>
```

The console is a great place to examine the structure and value of your variables. To examine data, use console.log(variable).

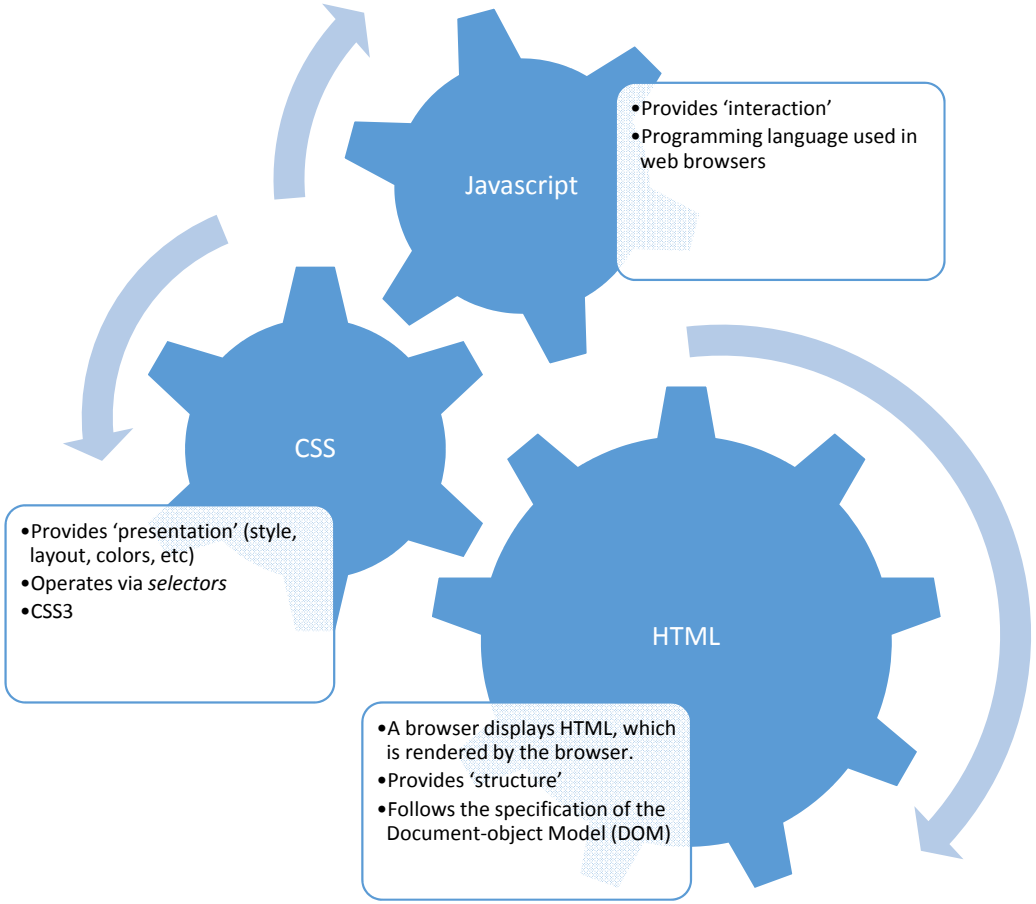
Learning basic Javascript

Open the previous HTML file

What we're doing:

- Create a <script> </script> tag in the header.
  - Create a function called **`clickFunction`**
  - **`clickFunction`** will use:
    - Alert
    - Console
- Add **`onclick="clickFunction()"`** to the button element.
- If you finish early: explore Javascript functions in the console

Recap


Javascript
•Provides 'interaction'
•Programming language used in web browsers
CSS
•Provides 'presentation' (style, layout, colors, etc)
•Operates via *selectors*
•CSS3
HTML
•A browser displays HTML, which is rendered by the browser.
•Provides 'structure'
•Follows the specification of the Document-object Model (DOM)

What is SVG?

- What is SVG?
  - Stands for Scalable Vector Graphics, a standard for graphics in web pages.
  - SVG allows us to create graphic objects in markup, and also manipulate them with Javascript!

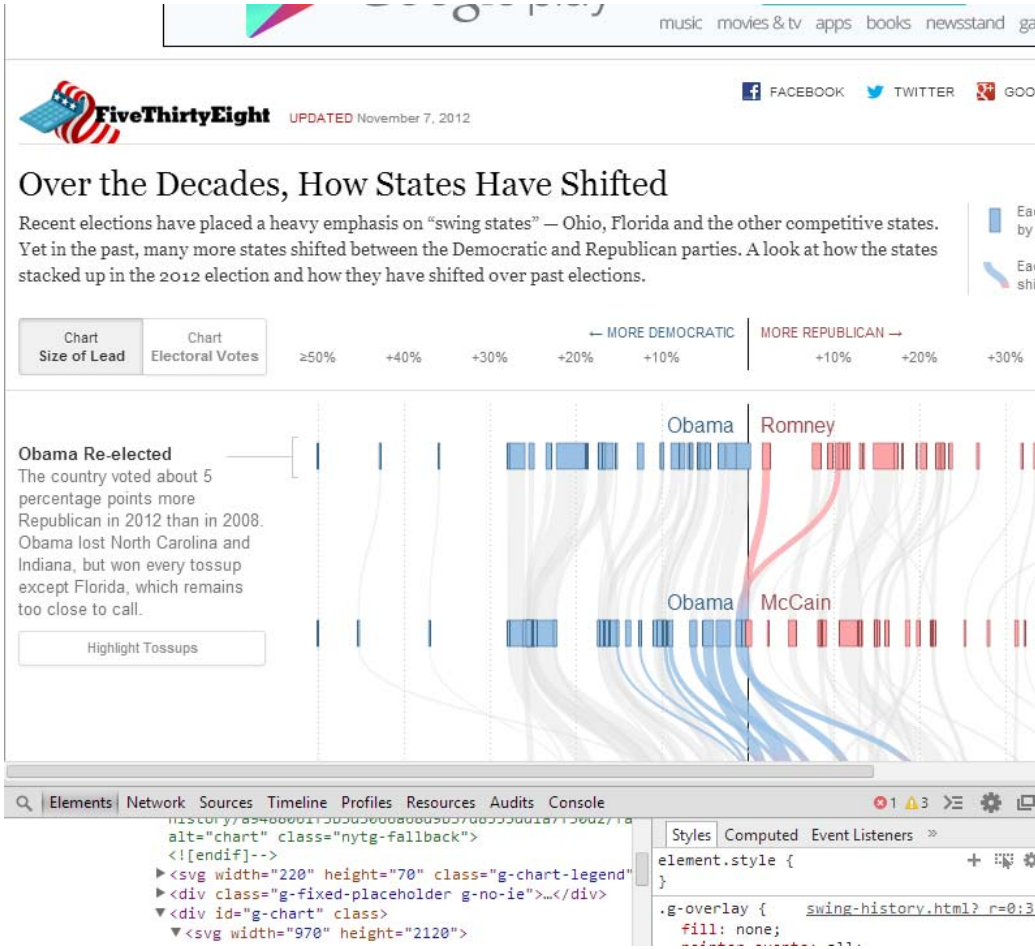
- Example:

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
<rect width="300" height="100" style="fill:rgb(0,0,255);stroke-width:1;stroke:rgb(0,0,0)" />
</svg>
```

- SVG Elements can also be styled with CSS

Why SVG?

- SVG allows us to build visualizations!



Getting started with SVG

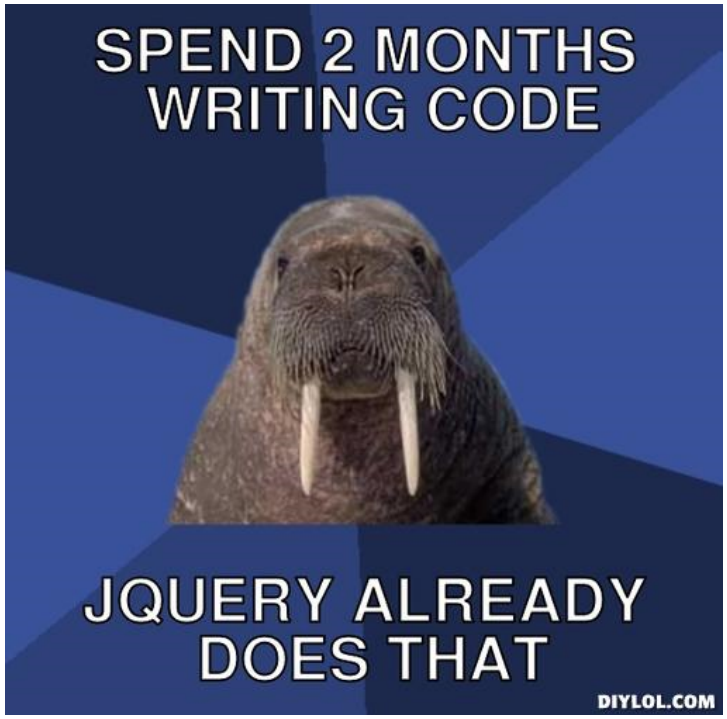
- Browse the specification at: http://www.w3.org/Graphics/SVG/IG/resources/svgprimer.html

- Tags you might want to try:
  - <rect></rect>
  - <circle></circle>

- To get started, add this to your index.html file:

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
     <rect width="300" height="100" style="fill:rgb(0,0,255);stroke-width:1;stroke:rgb(0,0,0)" />
     <circle cx="80" cy="170" r="40“ fill="yellow" stroke="blue" />
</svg>
```

- Again, SVG Elements can also be styled with CSS



Libraries and Frameworks

Don’t waste time reinventing the wheel!

Libraries and Frameworks

- What if someone already made libraries to make programming in Javascript easier and faster?
- What if someone already created CSS stylesheets that include standards in current UI design?

jQuery	D3.js	Bootstrap
• Makes DOM manipulation, animation, event-handling, and AJAX easier and simple. • Most widely used Javascript library, many plugins are built with jQuery • Extremely powerful.	• “data-driven documents” • Some overlap with jQuery in terms of event handling, selection, etc. • Allows for ‘binding’ of data to elements!	• Front-end framework dependent on jQuery • Focused on presentation, makes elements across the web consistent • Extremely popular. You’ll probably recognize some (modified) bootstrap when you see it.

Libraries and Frameworks

How do you include jQuery, d3.js, and Boostrap?

- http://d3js.org/
- http://getbootstrap.com/
- http://jquery.com/

Include these elements in your HTML header:

```
<script src="http://d3js.org/d3.v3.min.js" charset="utf-8"></script>
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>

<link rel="stylesheet" href="http://netdna.bootstrapcdn.com/bootstrap/3.0.3/css/bootstrap.min.css">
<script src="http://netdna.bootstrapcdn.com/bootstrap/3.0.3/js/bootstrap.min.js"></script>
```

These tags allow you to include the JavaScript and CSS files written by other people into your HTML document. (you can also download them locally and include them that way if you don't always have internet connection).

jQuery vs d3.js + learning more

If you have already used jQuery:

- There is significant overlap:

```
$('#foo')
.css('background', '#000')
.click(function() {})
.append($('<div></div>'));
```

```
d3.select('#foo')
 .style('background', '#000')
.on('click', function() {})
 .append('div');
```

- jQuery is generally used to manipulate the DOM, d3.js allows for data-binding
- You could write whole books on how to use these libraries.
- Search the documentation/Google on what you're looking for. This is an important skill!

Getting Started with Homework 1

Downloading the files

1. Go to https://github.com/CS171/HW1
2. In a location of your choice:

```
git clone https://github.com/CS171/HW1.git
```

Note: please create repository named cs171-hw1-lastname-firstname and copy the files over when you submit.

Downloading the files

1. Go to https://github.com/CS171/HW1
2. In a location of your choice:

```
git clone https://github.com/CS171/HW1.git
```

Note: please create repository named cs171-hw1-lastname-firstname and copy the files over when you submit.

Running a webserver

1. Navigate to the ‘HW1’ directory which you just created
2. Start a local web server with the following command:

```
python -m SimpleHTTPServer
```

Note: you need Python installed for this. (You probably already have this installed; if not, ask one of us for help)

Runninng a webserver

1. Navigate to the ‘HW1’ directory which you just created
2. Start a local web server with the following command:

```
python -m SimpleHTTPServer
```

Note: you need Python installed for this. (You probably already have this installed; if not, ask one of us for help)

Understanding table.html

1. Navigate to the ‘HW1’ directory which you just created
2. Visiting the following address in your browser:

http://localhost:8000/table.html

Understanding table.html

- d3.**tsv**(*url*[, *accessor*][, *callback*])
  - Issues an HTTP GET request for the tab-separated values (CSV) file at the specified *url*.

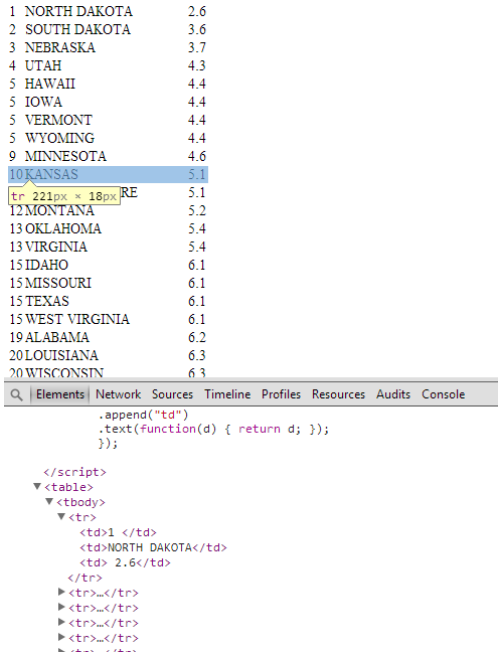
```
var table = d3.select("body").append("table"),
      tbody = table.append("tbody");

      var rows = tbody.selectAll("tr")
      .data(data)
      .enter()
      .append("tr");
```

Understanding table.html (2)

```
var table = d3.select("body").append("table"),
     tbody = table.append("tbody");
     var rows = tbody.selectAll("tr")
     .data(data)
     .enter()
     .append("tr");
     var cells = rows.selectAll("td")
     .data(function(row) {
       return d3.range(Object.keys(row).length).map(function(column, i) {
         return row[Object.keys(row)[i]];
         });
       })
     .enter()
     .append("td")
     .text(function(d) { return d; });
     });
```

Understanding table.html (3)

```
var table = d3.select("body").append("table"),
    tbody = table.append("tbody");
    var rows = tbody.selectAll("tr")
    .data(data)
    .enter()
    .append("tr");
    var cells = rows.selectAll("td")
    .data(function(row) {
      return d3.range(Object.keys(row).length).map(function(column, i) {
        return row[Object.keys(row)[i]];
        });
      })
    .enter()
    .append("td")
    .text(function(d) { return d; });
    });
```



Summary

- HTML, CSS, and Javascript are the building blocks of the web experience.
- SVG and d3.js make beautiful data visualization possible.

Questions?

Addendum 1: d3 vs jQuery

- http://www.macwright.org/presentations/dcjq/