CMPS 165 **Spring 2018**

Web Fundamentals: HTML (including DOM), CSS, Javascript, and SVG

Resources

Chapter 3 of the textbook on "Web Technology Fundamentals"
www.w3schools.com
https://developer.mozilla.org/en-US

Purpose of education is to learn how to learn.

In this handout, we will learn some basics of web fundamentals and learn how to learn more as needed. One could easily spend a whole quarter or more on learning web fundamentals. However, in this course we will spend only 1 to 1-1/2 class and empower ourselves to learn more about them as we go along. The objective is to learn enough about these fundamentals so that we can start working with D3 to generate cool interactive visualizations.

WEB

Web = Browser + Search Engine + Communication + ….

User-Client-Server-Provider Communication

Most descriptions of world-wide-web emphasize client-server communication. However, in my view, this description misses out two important entities. They are user and provider.

User=You (Human Being)
Browser=Client (Software Code)
Server (Software Code residing somewhere typically Apache)
Provider=Person or Institution (who provides the data and code describing content and appearance)

This communication is described on Pages 15-17 of Chapter 3 of the book. We will discuss this in the class along with the 4 URL properties:
1. Communication protocol
2. Domain Name
3. Port Number
4. Additional Path or parameters

In this course*, you are the provider*. You will be providing data and code to interested users so that they can view interactive visualizations. So the communication above is as follows:

User=Someone interested in your visualization (Human Being): lets call him Aarav;
Browser=Browser that Aarav uses (let us hope he uses Chrome; what if he doesn't?);
Server= Software Code residing somewhere typically Apache); don't worry about it;
Provider=YOU (provide this data to Aarav describing content and appearance of interactive visualization).

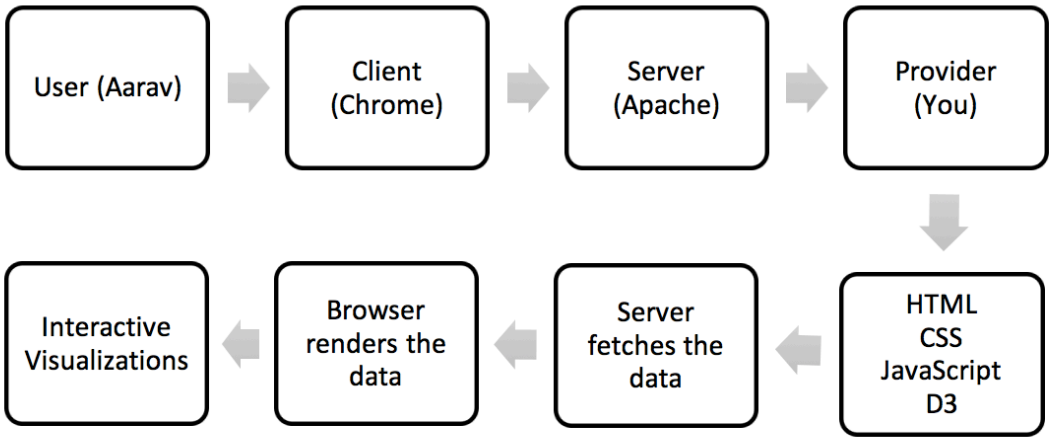

FIGURE: Web Communication: User -> Client (Browser) -> Server -> Provider

Search Engine
You can combine any browser with any search engine. Examples of popular browsers include Chrome, Firefox, Internet Explorer, and Safari. Examples of popular search engines include Google, Bing, Yahoo.

Since many common users are not aware of this distinction or that they can exercise a choice, companies can make a *lot* of money. For example, Firefox auctioned its search engine and switched from Google as its *default* search engine to Yahoo. Of course, if you don't like yahoo, you can always switch to Google, but do you know how?

In this class, our focus is not search engine but Browser. So, I will be using the term "Web" almost synonymously with "Browser" in this class unless otherwise mentioned. As discussed in the previous handout, we will use Chrome Browser.

Browser
A browser is software code that takes certain files such as html+css and converts (or *renders*) them into a display that opens up in a window as described in the Figure above.

HTML = *content* of the web
CSS = *appearance* of the web
CSS files are embedded inside the HTML files to describe what the appearance of the content should be.

Since we are dealing with visualizations, we will need to learn how to change css files based on data. This is known as data binding; you bind the data to documents (html and css) so that the content and the appearance change.

We will not learn anything about the internals of browsers and treat them as "black box". Since internals (software that runs the browser) differ, different browsers may yield different results for the code (html+css+….) you create. As stated in Handout 1, we will focus only on Chrome Browser and hope that it works with other browsers.

HTML

HTML = Hyper-**T**ext **M**arkup **L**anguage
Hyper-Text = Text+Images+Video+Audio+Web Links+ ……
HTML = content of the web

The core function of HTML is to "mark-up" content thereby giving it structure by providing hierarchy and relationships. This hierarchy gives meaning to content, referred to as semantic structure. Hierarchy and relationship will be described using tree terminology such as parent, sibling, and child (that you should be familiar with).

We will follow pages 19-25 of Chapter 3 of the textbook to learn the following about HTML:

Tags create *Elements*

1. Comments: <! Comments go here >
2. Paired tags: html, head, title, body, h1, h2, h3, h4, p, ul, ol, li, a, span, div
3. Paired nested tags: em, strong
4. stand-alone tags: <!DOCTYPE html>, img

Attributes are properties of elements. For example, href is a property of element *a* to create a hyperlink. Other attributes are Classes and ID, discussed below.

DOM = **D**ocument **O**bject **M**odel
Hierarchical Structure of HTML

We will follow only the very brief description on page 24 of Chapter 3 of the textbook on DOM. Web browsers parse DOM (hierarchical structure of HTML) to make sense of page contents. If you are lazy and do not provide DOM, browsers may provide default DOM with unintended consequences. In this class, we will assume that you will always provide HTML and the DOM associated with the HTML.

We need to create hierarchy using *div*, *class*, and *id*.
div is a tag to create an element that defines as arbitrary division within the document used for grouping.
Class is an attribute assigned to several elements. (example: awesome, sky)
id is attribute assigned to only one element. (example: button)

For example, we do not want to make all elements blue but we should know how to select only *div* with *class sky* to make them blue. Similarly, we do not want to make all elements clickable but only *div* with *id button1* clickable to perform action1.

For further learning and excellent tutorials and reference, we recommend the two websites mentioned at the beginning of this handout.
For further info, https://developer.mozilla.org/en/HTML/Element

CSS

CSS = **C**ascading **S**tyle **S**heet
CSS = *appearance* of the web
CSS is used to create visual representation of contents (presented in HTML) using *selectors* and *property-value pairs.*

We will follow pages 30-36 of Chapter 3 of the textbook to learn the following about CSS:

Selectors
Selectors allow you to *select* only certain parts of the content so that you can apply specific *properties* to only the selected content. The book (page 30) discusses how to select using following different types of selectors:
Type, Descendant, Class, ID, Combinations of multiple Class with multiple ID
http://mzl.la/V27Mcr

Examples:

Type:	h1, p, strong, em, div
Descendant:	h1 em, div p
Class:	.caption, .label, .axis
ID:	#header, #nav, #export
Combinations:	.bar.highlight, .axis.x, div.sidebar, #button.on

Property-Value Pairs
Property-Value pairs allow you to assign desired values to specific properties of selected content.

Examples

Margin:	10px;
Padding:	25px;
Background-color:	yellow;
Color:	pink;
Font-family:	Helvetica, Arial, sans-serif*;*

There are different ways to describe colors:
Name, RGB value, Hex value, RGBA where alpha stands for transparency

Linking CSS files to HTML files
There are three ways:
(i) Create a css file and link it within html using link command:
<link rel=”stylesheet” href=”cssfilename.css”>
(ii) include CSS commands inside HTML rather than using a separate file,
(iii) using inline styles.
First method is preferred over the second method because it helps to keep the content and style separate. Later, we will learn how to apply inline styles programmatically with d3 to affect appearances of specific content.

Inheritance
Children inherit the style of parents.

Cascading
Style rules cascade. The latest rule overrides the earlier rules.

For further learning and excellent tutorials and reference, we recommend the two websites mentioned at the beginning of this handout.
For further info, https://developer.mozilla.org/en/CSS/CSS_Reference

JavaScript

HTML and CSS files are sufficient to render *static* content and appearance. However, if you want the content and appearance of the web page to change *dynamically* based on certain events (such as the user *interaction*), you will need to learn Javascript. Javascript is a scripting language. Some components of most browsers are written in Javascript. By embedding Javascript code inside html and css documents, you are giving instructions to the Browser to act differently based on certain conditions/events.

We will follow pages 36-52 of Chapter 3 of the textbook to learn the following about JavaScript:

Comments			/*	*/		
Console.log						
Variables	int	float	Boolean	string		
Arrays						
Objects						
Arrays of Objects						

Mathematical Operators:	+	-	*	/		
Comparsion Operators:	==	!=	<	>	<=	>=
Control Structures:	if()	for()				
Functions						

Linking Javascript with HTML

Javascript code can either be embedding inside HTML files between two *script* tags as follows:

<script type="text/javascript"> console.log("Hello, world!")</script>

or stored in a separate file with a .js suffix and linked as follows:

<script type="text/javascript" src="javascriptfilename.js"></script>

Top Four Javascript Gotchas!

1. Dynamic Typing (loosely typed language: no variable declaration required; typeof)
2. Variable Hoisting (variable is hoisted up to the top of the function context)
3. Function-level scope (variables are available anywhere within the function and not just the block inside the curly braces)
4. Global Namespace (new variable gets added to the global object *window*; so organize your code to avoid variable name conflicts)

Javascript Output to HTML

(This section is not in the book).

Ref: www.w3schools.com/js/js_output.asp

Read "Using innerHTML"

To access an HTML element, JavaScript can use the **document.getElementById(id)** method.

The **id** attribute defines the HTML element. The **innerHTML** property defines the HTML content:

```
<!DOCTYPE html>
<html>
<body>
<h1>My First Web Page</h1>
<p>My First Paragraph</p>
<p id="demo"></p>
<script>
var txt= “Hello”;
document.getElementById("demo").innerHTML = txt;
</script>
</body>
</html>
```

The code above will print out “Hello” on the browser. There are other mechanisms (javascript codes) that can be used to print out contents to the browser. You can use any of these mechanisms. For further learning and excellent tutorials and reference, we recommend the two websites mentioned at the beginning of this handout.

SVG = **S**calar **V**ector **G**raphics

Back to the good old SVGs (based on lines) rather than Raster Graphics (based on pixels)!

We will follow pages 52-60 of Chapter 3 of the textbook to learn the following about SVG:

<svg width= height= ></svg>

Geometry elements and coordinate system (upper left corner):
Circle (cx, cy, r)
Rect
Ellipse
Line
path

Style elements:
Fill
Stroke
Stroke-width
Opacity

Color Specifications
rgb
rgba (transparency)