SUMMARY OF COURSERA COURSE

HTML, CSS AND JAVASCRIPT FOR WEB DEVELOPERS

RATINGS: 4/5

WEEK 1 – INTRODUCTION TO HTML 5

UNIT 1: Course Introduction

- It is essential to have the skills to code up a modern frontend application
- All essential and fundamental website coding skills
- Bootstrap is the most used CSS framework
- Part of the Ruby on Rails specialization
- Take us on a journey of coding real stuff

UNIT 2: How Grading and Being Late on Assignment Works

- Peer graded assignments, have to wait for assignments to be graded
- New cohorts very 2 weeks
- You can always resubmit failed assignments

UNIT 3: Recommended Books

- List of recommended books (screenshoted)

UNIT 4: Check out my site

- https://clearlydecoded.com/
- The site has not been updated in a while

UNIT 5: Example Code

- https://github.com/jhu-ep-coursera/fullstack-course4
- Download “fullstack-course4.zip”

UNIT 6: Optional Practice Quiz

UNIT 7: Introduction to HTML5

- His name is Yakov Heiken, he works with both frontend and backend

UNIT 8: Development Environment Setup Part 1

- Sign up for an account of GitHub
- You will need Google Chrome Browser
- You need a code editor (I will be using VSCode), he recommends Sublime
- You will need Git
- Browser Sync, we need to install Node.js

UNIT 9: Development Environment Setup Part 2 – MacOS

- Google search “git download”, to download git terminal
- Google search “download nodejs”
- Google search “browser sync”
- Browser sync helps to update our code immediately to the browser, similar to “live server” extension on VSCode

UNIT 10: Development Environment Setup Part 2 – Windows

- He will be using MacOS for the rest of the course
- “npm”: node package manager
- We will learn how to host our website on GitHub

UNIT 11: Optional Practice Quiz

UNIT 12: Development Environment Setup Part 3: GitHub and Browser Sync

- How to create your own repository on GitHub, using some git command
- “Git Book”: it is a free book on git, he recommends
- On GitHub Repository: “Setting” > “GitHub Pages”
- You can have several branches on your Git Repository
- You set your password automatically on Git, so you can push (git push) without typing password very time
- Most servers default to “index.html”
- I will use my “live server” extension on VSCode instead of “Browser Sync”

UNIT 13: Resources for Asking Questions

- “Stackoverflow”: it has a very large developer network
- “jsfiddle”: site that allows you to write html, css and javascript codes, then share your code as a link
- “codepen”: similar to jsfiddle, has large community

UNIT 14: What is HTML?

- HyperText Markup Language = HTML
- “HyperText”: is text that contains links to other texts
- HTML is human readable
- Core Technologies that drive the web: HTML, CSS, Javascript
- HTML provides the structure, annotates content
- CSS provides the style
- JS provides behavior, functionality

UNIT 15: Relevant History of HTML

- Before 1997 there were no “standards”, so each browser had its own specific markup elements
- After “World Wide Web Consortium” (W3C) started applying some standards
- Now “WHATWG” (Web Hypertext Application Technology Group) formed by web vendors banding together are the drivers now
- From 2007, “W3C” and “WHATWG” started working together
- “W3C” handles HTML5 (standard), ”WHATWG” handles HTML (evolving)
- To keep track of the changes in HTML visit: “caniuse.com”, “w3c webpage”, “validator by w3c”
- It is good to check out browser statistic to know the most used browsers to target (w3schools browser stats)
- Google is your friend

UNIT 16: Optional Quiz

UNIT 17: Anatomy of an HTML Tag

- At the core of HTML is the “HTML tag”
- Usually have a “opening” and “closing” tag
- Tag and Element are used synonymously
- Attribute of Tag: a name value pair assigned to the tag
- It is best practice to put “attribute values” in quotes

UNIT 18: Optional Quiz

UNIT 19: Basic HTML Document Structure

- Always use the “<!doctype html>” version/page declaration to make the web browser render your page as following HTML5 standard
- “<meta>”: a self-closing tag, communicates information to the browser
- We are always nesting one HTML tag into another HTML tag
- The browser renders the HTML file from top to bottom

UNIT 20: Optional Quiz

UNIT 21: HTML Content Models

- Which elements are allowed to be nested inside which other element = “Content Model”
- “Block-Level Elements”: render to begin on a new line, may nest inline or block-level elements, roughly translates to “Flow Content”
- “Inline Elements”: render on the same line, can only nest other inline elements, roughly translates to “Phrasing Content”
- “div” element is the most generic Block-Level element
- “span” element is the most generic Inline Element

- Visit “w3.org” search for kind of HTML content

UNIT 22: Optional Quiz

UNIT 23: Heading Elements (and some new HTML5 semantic elements)

- “Semantic HTML element”: element that implies some meaning to the content, they may help in SEO
- “h1” to “h6”: are the html heading elements from most important to the least important
- Well chosen “h1” element is crucial to SEO
- “header”: contains the company name, logo, navigations
- “section”: it is usually for the “article” element to be nested in it
- “aside”, “footer”

UNIT 24: Optional Quiz

UNIT 25: Lists

- Very useful HTML structure, they provide a natural commonly used way of grouping items
- Very often, lists are used for structuring navigation portions of the webpage
- “ul” tag: unordered list
- “li” tag: list items, embedded in the “ul” tag
- Text not allowed in “ul” tag, only “li” tag are allowed in the “ul” tag
- “ol” tag: ordered list

UNIT 26: HTML Character Entity References

- “<”: use “&lt;” instead
- “>”: use “&gt;” instead
- “&”: use “&amp;” instead
- “&copy;”: copyright symbol
- “&nbsp;”: non breaking space, to allow a space not to be broken
- “&quot;”: to add quote brackets
- HTML Entities help to avoid issues, provide characters not available on a keyboard

UNIT 27: Optional Quiz

UNIT 28: Creating Links

- Links pretty much make the web what it is
- Links: external (linking to other websites), internal (linking to other pages in the site), fragment (#***, point to sections on our page)
- “<a href = # title=** target= ”_blank”> ** </a>”
- Relative and absolute URL
- “a” tag is both an inline and block level element
- Used for navigation in SPA (Single Page Applications)

UNIT 29: Optional Quiz

UNIT 30: Displaying Images

- “<!-- **** -->”: how to comment on HTML
- “img” tag: helps to render images on the webpage, its an inline element
- <img src=*** alt=***>
- He says it is good to specify the “width” and “height” of our image, whenever possible

UNIT 31: FAQ

- https://github.com/jhu-ep-coursera/fullstack-course4/blob/master/FAQ.md

- https://github.com/jhu-ep-coursera/fullstack-course4/blob/master/FAQ.md#q-how-to-ask-and-get-your-question-answered-a-must-read

UNIT 32: Optional Quiz

UNIT 33: Connect with me

UNIT 34: Module 1 Wrap-Up

- Module 1 and Module 2 help us build foundation

UNIT 35: Module 1 Quiz

WEEK 2 – INTRODUCTION TO CSS 3

UNIT 1: Welcome to Module 2 – Intro to CSS3

- Foundations of CSS, learn media queries, cover Twitter Bootstrap

UNIT 2: Recommended Books

UNIT 3: New Tutorial – here is how

UNIT 4: Power of CSS

- On the web content is king, user experience of content matters
- How users absorb and relate to that content largely depends on our experience of consuming the presentation
- www.csszengarden

UNIT 5: Anatomy of a CSS Rule

- CSS works by associating rules with HTML elements
- “p {property: value;}”, property/value pair
- Collection of CSS rules is called a stylesheet
- Every browser comes with default style it applies to some HTML elements, it is your job to reset this default style

UNIT 6: Optional Quiz

UNIT 7: Element, Class and ID Selectors

- The browser uses it selector API to transverse the DOM
- A lot of JavaScript libraries out there use the browser selector API to attach behavior and data to HTML elements, much in the same way that CSS applies style to those elements
- Types of Selector: “element”, “class”, “id”
- “.class_name”: class selector
- “#id_name”: id selector
- You can group selectors into one rule, we separate selectors with “,” (comma)
- The id selector is the least reusable selectors because “id” have be unique in HTML

UNIT 8: Optional Quiz

UNIT 9: Combining Selectors

- Combining selectors is a powerful technique that allows you to more precisely target DOM elements
- “p.big”: all p elements with class name “big” (element with class selector)
- “article > p”: every p element that is a direct child of the article element (child selector)
- “article p”: every p element that is inside (at any level) of the article element (descendant selector)
- The combined selectors are not limited to element selectors
- Adjacent sibling selector (selector + selector) and General Sibling selectors (selector ~ selector) check them out

UNIT 10: Optional Quiz

UNIT 11: Pseudo-Class Selectors

- Pseudo-class selectors address targeting only the structures that cannot be targeted by simple combinations of regular selectors, or targeting the ability to style based on user interaction with the page
- “selector:pseudo-class { }”: how to specify a pseudo class selector
- Many pseudo-class selectors that exist, pseudo selector can be combined
- “hover”: moves is mouse over the element, “active”: clicks the element but has not released the element

- Make sure your selector is still readable. Simple readable > Complicated Tricky

UNIT 12: Optional Quiz

UNIT 13: Style Placement

- Style declarations override other style declarations; they have a hierarchy
- “Inline Styling” it is the least preferred method of styling; it is the least reusable
- In real world website most CSS are in “External CSS”
- “Head styles” are usually there to override “External Styles”

UNIT 14: Optional Quiz

UNIT 15: Conflict Resolution Part 1

- “Cascading is a fundamental feature of CSS”, it is an algorithm defining how to combine properties values originating from different sources
- “Origin Precedence Rule”: the last declaration wins, HTML is processed sequentially
- “Declarations Merge”: it merges properties from different origin if they don’t conflict
- “Inheritance”: Child/grandchild of an element will inherit the properties of its parent element

UNIT 16: Conflict Resolution Part 2

- “Specificity”: most specific selector combination wins
- Specificity Score = “style=” ”” > “ID” > “class, pseudo-class” > “no of elements”
- “!important”: it means I want to override everything, it can become a maintenance nightmare, use sparingly

UNIT 17: Optional Quiz

UNIT 18: Styling Text Part 1

- Tons of CSS properties that affect the way text is displayed
- “font-family”, “color” you will want to use a hexadecimal value (#*****)
- “font-style”, “font-weight”, “font-size” default font size is usually 16px
- “text-transform”, “text-align”

UNIT 19: Styling Text Part 2

- “%” and “em” are relative measurement, they have accumulative effects
- “%”: calculate the percentage of the default font-size (120% = 1.2*16px)
- “em”: 2em = 2 * parent font-size
- It is best to keep things consistent

UNIT 20: Connect with me

UNIT 21: Optional Quiz

UNIT 22: The Box Model Part 1

- In HTML every element is considered a box
- Box Model: content, padding, border, margin, width, height
- “padding: top right bottom left”
- “box-sizing” at default it is set to “content-box”, so if you specify a width it is only for the pure “content”
- With “box-sizing: border-box”, the “width” includes the “padding” and the “border” sizes
- All modern frameworks like Bootstrap use “border-box”

UNIT 23: The Box Model Part 2

- “box-sizing” is a property that cannot be inherited
- “*”: is a selector that selects all the elements in the HTML specifically (universal selector)
- “* {box-sizing: border-box}”
- “margin collapse”, for elements stacked together vertical the larger margin is applied, it is not added together

UNIT 24: The Box Model Part 3

- “overflow”: helps to deal with overflow content
- “overflow: visible/hidden/auto/scroll”
- Users usually hate double scrolling
- Box model is essential to understand

UNIT 25: Optional Quiz

UNIT 26: The Background Property

- “background-color/image/repeat”
- “background-repeat: repeat-x/repeat-y/no-repeat”
- “background-position: right/left/bottom”
- You can combine all this sub-property into one using “property: ***”

UNIT 27: Optional Quiz

UNIT 28: Positioning Elements by Floating

- I will stick to my flex boxes
- When you float elements it takes them out of the regular document flow
- We use the “clear” property to restore the regular document flow
- “float: left/right”, “clear: left/right/both”
- When float elements can’t fit on the same line they drop to the next line
- Floats are positioned at the top-right or top-left of containing elements

UNIT 29: Optional Quiz

UNIT 30: Relative and Absolute Element Positioning Part 1

- “Static Positioning”: normal document flow, default for all
- “Relative Positioning”: element is positioned relative to its position in normal document flow, element is not taken out of normal document flow (if moved its original spot is preserved), it takes CSS (offset) properties like “top/bottom/left/right”
- “top: **px”: for position: relative it means “from top”
- “Absolute Position”: all offsets (top, bottom, left, right) are relative to the position of the nearest ancestor which has positioning set on it, other than static, element is taken out of normal document flow
- If a container element is offset, all the elements in it at offset with it

UNIT 31: Relative and Absolute Element Positioning Part 2

- Static positioning is default for all elements except html

UNIT 32: Optional Quiz

UNIT 33: Media Queries Part 1

- Media queries allow you to group styles together and target them to devices based on some criteria
- Without media queries responsive web design will not be possible
- @media (media_feature) {***}
- If the media_feature translates to “true” the style in the “{ }” are executed
- Most common media_features are “min-width: **” and “max-width: **”
- We can combine media_features with logical operators like “and”, “or using ‘,’ ”
- Start with base styles > apply media queries to adjust the base styles > make sure the media queries range don’t overlap

UNIT 34: Media Queries Part 2

- For him large devices have a width of at least “1200px”
- Point between media queries is called “breakpoints”

UNIT 35: Optional Quiz

UNIT 36: Responsive Design Part 1

- Responsive web design was born primarily out of the need to deal with the explosion of mobile devices that started being able to browse the web
- “Responsive Web Site”: is a site designed to adapt its layout to the viewing environment by using fluid, proportion-based grids, flexible images and CSS3 media queries
- “Content should be like water”
- Most common responsive layout is the 12-columns Grid
- We can have Nested Grids

UNIT 37: Responsive Design Part 2

- The viewport <meta> tag is very important for responsive web design
- Use % for fluid width

UNIT 38: Optional Quiz

UNIT 39: Introduction to Twitter Bootstrap Part 1

- Bootstrap one of the most popular HTML/CSS/JS framework for developing responsive mobile first projects on the web
- Contains mostly CSS classes, has some JS using "JQuery"
- Plan mobile from the start
- Bootstrap is too big/bloated, lot of features you will not use
- You can write your own that's more targeted/smaller (it will take lot longer)

UNIT 40: Introduction to Twitter Bootstrap Part 2

- We can download Bootstrap on to our local machine or we can use the CDN server version
- We place the link to our style below that of bootstrap so that we can customize/overwrite some of it
- Bootstrap JS depends on JQuery

UNIT 41: The Bootstrap Grid System Part 1

- One of the most central components of Bootstrap that allows developers to easily create responsive layouts is the grid system
- Aligning the edges of different components on your web page is basic design principle
- Bootstrap gird system must always be inside a "container/container-fluid" wrapper, followed by the "row" class it creates a horizontal groups of columns, then followed by "col-SIZE-SPAN" (Bootstrap Column class name format)
- "col-SIZE (screen width range identifier, columns will collapse below that width unless another rule applies)-SPAN (how many columns the element should span)"

UNIT 42: The Bootstrap Grid System Part 2

- Remember the Bootstrap Grid system structured "container" > "row" > "col"
- We can use our custom style to overwrite some default Bootstrap styles
- You can always check out the Bootstrap documentations

UNIT 43: Optional Quiz

UNIT 44: FAQ

UNIT 45: Wrap Up

- Finished about half the course, now we know the fundamentals of HTML/CSS

UNIT 46: Coding Assignment

- My Solution in folder “module-2-solution”

UNIT 47: Review Your Peer

WEEK 3 - CODING THE STATIC RESTAURANT SITE

UNIT 1: Welcome to Module 3 – Coding the Static Restaurant Site

- This module is a real journey where we will code a real website for a real client from scratch

UNIT 2: Recommended Books

UNIT 3: Visit with the Client Part 1

- Most clients have no idea what they want on the website
- It is your job to ask questions to figure it out, you can bring web site examples of similar businesses
- Less (information) is more, encourage your client not to cram too many info on the site
- Find a way for the client to invest in the project
- Have client designate one person responsible for decisions
- Limit number of revisions UPFRONT, if it is a paying job limit the number of free revisions
- Google for “web development client questionnaire”
- Involve others if needed, like designers, photographers
- Get an idea of what the client has right now

UNIT 4: Visit with the Client (Field Trip) Part 2

- Don’t ever code on an empty stomach

UNIT 5: New Tutorial here is how

UNIT 6: Design Overview

- Don’t design and code immediately, make the designs as mockups and show to the client
- Designer provides the color used, font used
- Try to stick with Google fonts
- Professional designer use Adobe Photoshop, you can always start with simple tools just have an idea of how the website will look

UNIT 7: Some Ground Rules and Overview of Setup (lecture 29)

- Source code is available
- We will build up the site piece by piece

- Certain things will be done off-screen to save time
- Switch between code and browser a lot
- BrowerSync similar to LiveServer
- Set up some global styles like Background color, font family, text color

UNIT 8: Optional Quiz

UNIT 9: Coding Basics of Navbar Header Part 1 (lecture 30)

- Try to constrain the width of the browser
- Always Check out the Boostrap Nav bar documentations

UNIT 10: Coding Basics of Navbar Header Part 1(lecture 30)

- "pull-left": is bootstrap class name for "float: left"
- "visible-md": only visible when the screen is medium size

UNIT 11: Optional Quiz

UNIT 12: Coding Button for Future Collapsible Menu (lecture 31)

- Always check out the bootstrap documentations

UNIT 13: Optional Quiz

UNIT 14: Coding Nav Menu Buttons Part 1 (lecture 32)

- Use unordered list <ul> to display navigation items
- You can get "glyphicons" from Bootstrap

UNIT 15: Coding Nav Menu Buttons Part 2 (lecture 32)

- Using of negative margin values

UNIT 16: Fixing Navbar Layout, Text and Dropdown Menus Part 1 (lecture 33)

- You can load up different logo sizes for different screen width
- 1vw = 1% of viewport width, it is a responsive font-size style

UNIT 17: Fixing Navbar Layout, Text and Dropdown Menus Part 2 (lecture 33)

- 1em: 1 * font-size of parent element

UNIT 18: Coding the Jumbotron (lecture 34)

- "jumbotron"
- "img-response": it makes the image response
- One way to load a different size image depending on the browser width is to use "background-image" CSS property and load a different "url" in different media queries
- Be friendly with the Browser Developer Tool for debugging

UNIT 19: Optional Quiz

UNIT 20: Coding Navigation Tiles Part 1 (lecture 35)

- Remember every Bootstrap Grid has to be in a “container” class
- “col-md-4” > “col-sm-6” > “col-xs-12”
- “<iframe>” tag used to embed links/sites to the webpage
- “overflow: hidden”

UNIT 21: Coding Navigation Tiles Part 2 (lecture 35)

- Check out “placeholdit” website
- Check out maps.google.com to get embedded maps

UNIT 22: Coding the Footer Part 1 (lecture 36)

- It is good to end the div and add an HMTL comments to describe the div
- “class = panel-footer”
- “<hr>” tag: horizontal rule

UNIT 23: Coding the Footer Part 2 (lecture 36)

UNIT 24: Optional Quiz

UNIT 25: Coding the Menu Categories Part 1 (lecture 37)

UNIT 26: Coding the Menu Categories Part 2 (lecture 37)

- The smallest level inn Bootstrap is “col-xs-**”
- You can always checkout how Bootstrap style in order to create a similar custom class
- With JavaScript we can dynamically input some elements

UNIT 27: Coding the Single Menu Category Page Part 1 (lecture 38)

- A Grid within a Grid
- Bootstrap “active” class
- Nested Grids are possible in Bootstrap

UNIT 28: Coding the Single Menu Category Page Part 2 (lecture 38)

UNIT 29: Coding the Single Menu Category Page Part 3 (lecture 38)

- Bootstrap “clearfix” class help to fix overflow of content that affects the float items
- Done with our layout

UNIT 30: Testing the Mobile Version on a Real Phone

UNIT 31: FAQ

UNIT 32: Module 3 Wrap up

UNIT 33: Module 3 Coding Assignment

- My Solution in folder “module-3-solution”

UNIT 34: Review Your Peer

WEEK 4 – INTRODUCTION TO JAVASCRIPT

UNIT 1: Welcome to Module 4: Introduction to JavaScript

- All about the JavaScript Language, most used programming language
- Lot of frameworks that simplify JavaScript Programming are now available

UNIT 2: Recommended Books

UNIT 3: Adjusting Development Environment for JavaScript Development

- Loading our intended folder
- Chrome developer tool is a great tool to have and to use
- The “console tab” is where we will run JavaScript and see some of the messages

UNIT 4: Where to Place JavaScript Code

- We could have <script> tag in the <head> tag,
- <script> tag requires a closing tag; you can link an external JS code “src = ***”
- The JavaScript engine in the browser is single threaded not multi-threaded, it is also sequential
- We can define <script> tag also in the <body> tag
- “console.log()”

UNIT 5: Defining Variables, Function and Scope Part 1

- “var *name* = ”: defining a variable
- No types are declared, it is dynamically typed language
- “function **name** () { }”: defining a function
- “*function_name* ()”: execution of a function
- Functions arguments don’t require “var”
- All arguments defined in a function are optional
- Scope: “Global” and “Lexical/Function” scopes
- Lexical: depends on where a variable is physical defined, in JS functions define a new scope, variable/ functions defined here are available only within this function
- Global: variables and function defined here are available everywhere
- Scope Chain: everything is executed in an “Execution Context”, Funtions invocation creates a new Execution Context
- Each Execution Context: has its own variable environment, special “this” object, reference to its Outer Environment

- Referenced (not defined) variable will be searched for in its current scope. If not found, the outer reference will be searched. If not found, the outer reference's outer reference will be searched, etc. This will keep going until the Global Scope. If not found in Global Scope, the variable is "undefined"

UNIT 6: Defining Variables, Function and Scope Part 2 (lecture 41)

- Variables are dynamically typed in JS
- JS code runs within an Execution Context
- Scope chain is used to retrieve variables from Outer Variable Environments, for variables not found in your Execution Context

UNIT 7: Optional Quiz

UNIT 8: JavaScript Types Part 1 (lecture 42)

- A type is a particular data structure, built-in types can be used to build other data structures
- JS has 7 built-in types: 6 primitive and 1 object type
- Object is a collection of name/value pairs
- Primitive type represents a single, immutable value, values become read only
- Primitive Types: Boolean (true, false), Undefined (signifies that no value has ever been set), Null (signifies lack of value, it is ok to explicitly set a variable to null), Number (only numerical type in JS), String (used to represent text), Symbol (is new to ES6)

UNIT 9: JavaScript Types Part 2 (lecture 42)

- "undefined": the variable has been defined but no value has been set to it

UNIT 10: Optional Quiz

UNIT 11: Common Language Constructs Part 1 (lecture 43)

- String Concatenation, Regular Math Operators, Equality ("==" and "===" (strict equality))
- "NaN": not a number

UNIT 12: Common Language Constructs Part 2 (lecture 43)

- "| |" : used as OR operator
- "false", "null", "undefined", "empty_string", 0, "NaN" are interpreted as FALSE by the IF statement (conditional statement)
- "Boolean (***)": returns the Boolean value of ***
- "&&": used as AND operator
- "true", "none empty string", non-zero number, are interpreted as TRUE

UNIT 13: Common Language Constructs Part 3 (lecture 43)

- Always put semi-colon at the end as a best practice
- Best practice for "{ }": always put curly brackets on the same line
- "for (initialization; condition; increment) { }"
- "i++" = "i = i + 1"
- "Type Coercion"
- Typical precedence holds for Math Operator

UNIT 14: Handling Default Values (lecture 44)

UNIT 15: Optional Quiz

UNIT 16: Create Objects Using "new Object ()" Syntax Part 1 (lecture 45)

- "var company = new Object()": creating an object type (dictionary)
- Different ways of creating Object type in JS
- We can use "[**]" or ".**" notations to work with Object type, he thinks bracket notation is better

UNIT 17: Create Objects Using Object Lateral Syntax (lecture 45)

- Using an Object Literal Notation "{**: ***}" is a better way to create objects, it is more readable and easier to type
- It is good to put all your programming ending at the end first before adding the middleware

UNIT 18: Optional Quiz

UNIT 19: Functions Explained (lecture 46)

- Functions are First-Class Data Types (whatever you can do to a variable and an object you can do to first-class data types), they are also Objects
- Every function has an Inherent functions to it ".toString()"
- Functions are special type of objects, functions make JS very flexible and powerful

UNIT 20: Optional Quiz

UNIT 21: Passing Variables by Value vs by Reference Part 1 (lesson 47)

- Passing (Copying) by Value: Given "b = a", passing by value means changing copied value in "b" does not affect the value stored in "a" and vice versa
- Passing by Reference: Given "b = a", passing by value means changing copied value in "b" does affect the value stored in "a" and vice versa
- In JS, primitives are passed by value, objects are passed by reference
- Under the hood everything is actually passed by value

UNIT 22: Passing Variables by Value vs by Reference Part 2 (lesson 47)

- Important to understand this concept

UNIT 23: Optional Quiz

UNIT 24: Function Constructors, prototype, and the “this” keyword (lecture 48)

- “this”: point to the Global Object by default
- “methods”: are functions set on property of objects
- Don’t forget “new” keyword when using a “Function Constructor”, try to capitalize the “Function Constructor” names
- You can use Function Constructor to get new Objects (Function Constructor instance)
- “prototype”: can use this keyword to assign a function to the Function Constructor outside

UNIT 25: Object Laterals and the “this” keyword (lecture 49)

- Some complex shit

UNIT 26: New Tutorials here is how

UNIT 27: Optional Quiz

UNIT 28: Arrays Part 1 (lecture 50)

- Arrays: are just a collection of data
- “var list = new Array()”: to initialize an array
- Because of dynamic typing you can store different type of types in an Array
- Short hand array creation: “var list = [**, **, **]”, similar to object literal
- Arrays in JS can be sparse, we can set value to any index value while leaving the other indexes undefined

UNIT 29: Arrays Part 1 (lecture 50)

- Arrays are just Objects in JS

UNIT 30: Optional Quiz

UNIT 31: Closures (lecture 51)

- Closure is an essential topic in JS, without closure something like AJAX would not really be possible

UNIT 32: Fake Namespace (lecture 52)

- Namespace help to prevent functions in different script from stepping on each other

UNIT 33: Immediately Invoked Function Expressions (IIFEs) (lecture 52)

- IIFE helps to run the function object immediately, it helps to place code into its own execution context not to conflict with the global scope

UNIT 34: Optional Quiz

UNIT 35: FAQ

UNIT 36: Module 4 Wrap-Up

- Have a solid background in JS, start applying JS to our website

UNIT 37: Coding Assignment

- Module-4-solution folder

WEEK 5 – USING JAVASCRIPT TO BUILD WEB APPLICATIONS

UNIT 1: Welcome to Module 5

- Using JS to manipulate elements within our webpage and as well as use AJAX to pull real data from a server

UNIT 2: Recommended Books

UNIT 3: DOM Manipulation Part 1 (lecture 53)

- DOM: Document Object Model
- “document” is a property of the “window”
- “document object” represent the whole HTML page
- “document.getElementById”
- Best to place the JS file that affect the DOM at the end of the <body> tag
- “document” is an instance of “HTMLDocument”, therefore it inherits all its methods

UNIT 4: DOM Manipulation Part 2 (lecture 53)

- “document.getElemenyById(**).textContent/innerHTML”
- “document.querySelector(**)”: its uses CSS selectors (if you want to select an id for instance you write “#idname”)

UNIT 5: Handling Events (lecture 54)

- Event Handlers are basically functions that you bind using specific methods to certain events that happen in the browser
- One of the easiest way to add an event to an element is to use the “on*something*” attribute, side effect it kinds of dirty your HTML with them
- “Unobstructive Event Handling”: HTML knows nothing about the event, it provides a little bit more flexibility

- “.addEventListener(“event”, function)”
- “.on_something = function”
- “document.addEventListener(“DOMContentLoaded”, function), helps to make sure that our script only run after the DOM elements have been loaded

UNIT 6: The “event” Argument (lecture 55)

UNIT 7: Stay in Touch

UNIT 8: Optional Quiz

UNIT 9: HTTP Basics

- AJAX is a client-server communication technique
- HTTP: HyperText Transfer Protocol, it is based on request/response stateless protocol, client (usually the browser) opens connection to server > client sends HTTP request for a resource > server sends HTTP response to the client (w/resource) > client closes connection to server
- URN: Uniform Resource Name, uniquely identifies resource or name of resource, does not tell us how to get the resource
- URI: Uniform Resource Identifier, uniquely identifies resource or location of resource, does not necessarily tell us how to get the resource
- URL: Uniform Resource Locator, is a form of URI that provides info on how to get resource
- HTTP Request Structure (GET): GET/ URI_string?query_string HTTP_version, only issued when the browser is connected to the server
- HTTP Methods: “GET” – retrieves the resource, data is passed to server as part of the URI (query string); “POST” – send data to server in order to be processed, data is sent in the message body, no query string
- HTTP Response Structure: “HTTP/1.1 200 OK”
- “200 OK”: means ok, here is the content you requested
- “404 Not Found”: server cannot find the resources requested
- “403 Forbidden”; unauthenticated client tried to access a secure resource
- “500 Internal Server Error”: some unhandled error was raised on the server

UNIT 10: Ajax Basics Part 1

- AJAX: Asynchronous JavaScript and XML, very few apps use XML nowadays JSON is now more commonly used
- Faster response, nicer experience for user
- Synchronous Execution: execution of one instruction at a time
- Asynchronous Execution: execution of more than one instruction at a time
- “callback funtion”: it handles server response

UNIT 11: Ajax Basics Part 2 (lecture 57)

- I am quite lost

UNIT 12: Ajax Basics Part 3 (lecture 57)

- AJAX calls are asynchronous

UNIT 13: Optional Quiz

UNIT 14: Processing JSON (lecture 58)

- JSON: JavaScript Object Notation, light weight data-interchange format, easy for humans to read and write, easy for machines to parse and generate, completely independent of any language
- It is a subset of JS object literal syntax, property names must be in double quote, and any string value must be in double string
- JSON is just a string, it is NOT a JS object lateral
- "var obj = JSON.parse(jsonString)": converts from JSON string to object
- "var str = JSON.stringify(obj)": converts from object to JSON string
- JSON is a great format for passing data from server to client and back

UNIT 15: Optional Quiz

UNIT 16: Fixing Mobile Nav Menu Automatic Collapse (lecture 59)

- Jquery function name is "$" sign

UNIT 17: Dynamically Loading Home View Content (lecture 60)

- Single Page Web Application is now popular now and we need to dynamically load the content using AJAX
- "ajax.load.com": generates loading gifs

UNIT 18: Dynamically Loading Menu Categories View Part 1 (lecture 61)

- Used Ruby on Rails for backend
- CORS: Cross Origin Resource Sharing
- Used Heroku to serve the JSON data for categories

UNIT 19: Dynamically Loading Menu Categories View Part 1 (lecture 61)

- "{{property_placeholder}}"
- Well that what a lot of confusing functions

UNIT 20: Dynamically Loading Single Category View (lecture 62)

- I am quite lost, similar workflow as above

- He was flying through the code

UNIT 21: Changing “active” Button Style Through JS

- Fixing the “active” using DOM

UNIT 22: Check out the final deployed site

- https://www.davidchuschinabistro.com/index.html#/
- The site was re-implemented in AngularJS, the styles and HTML is the same but the way it was broken up and the way we reach the server is different

UNIT 23: FAQ

UNIT 24: New Tutorials

UNIT 25: Course Wrap Up

- With frameworks you can achieve lot of functionality with little code

UNIT 26: Stay in Touch

UNIT 27: Coding Assignment

UNIT 28: Peer Review