SUMMARY OF UDACITY COURSE

JAVASCRIPT AND DOM (4 LESSONS)

RATING: 4.5/5

LESSON 1 - THE DOCUMENT OBJECT MODEL

UNIT1: Introduction

- How do HTML/CSS connect with JavaScript
- Learn about the DOM, modifying page content, browser events, performance

UNIT2: The DOM

- DOM: Document Object Model
- DOM: A tree structure that captures the content, properties of the HTML and all the relationship between the nodes
- DOM: Full parse representation of the HTML Markup
- DOM is not part of the JavaScript Language
- DOM is constructed from the browser
- DOM is globally accessible by JavaScript code using the "document" object

UNIT3: Selecting Page Elements with CSS Selectors

- Use "." to select class names
- Use "#" to select ID

UNIT4: Select Page Element by ID

- "document.getElementById(***)":
- We can store the output of "document.getElementById()" in a variable

UNIT5: Select Page Elements by Class or Tag

- "document.getElementsByClassName()", "document.getElementsByTagName()"
- Don't forget this methods have a "s"
- "<pre> </pre>": the pre tag
- It returns an "HTML Collection", not an array

UNIT6: Nodes, Elements, and Interfaces

- Characters > Tags > Tokens > Nodes > DOM
- Node: is like a blueprint of a building, similar to "Class" in OOP, it contains "Properties(data)" and "Methods(functionality)", also an Interface
- "node": Objects built from the Interface/Blueprint/Node
- Interfaces on MDN: https://developer.mozilla.org/en-US/docs/Web/API

- Element Interface: is a blueprint for creating elements, it is a descendant of Node Interface
- Lot of different interfaces and method check out the link above

UNIT7: More Ways To Access Elements

- One of the main purposes of the Jquery Library was to abstract the way the differences between different browsers, it determines which browser was running in and use the right code for that browser.
- Since most Browsers now use the Standard methods Jquery popularity has diminished
- "document.querySelector("#header/.header/header")": id, class name, tag, it returns only one element(the first)
- "document.querySelectorAll()": returns a list of elements

UNIT8: Lesson Summary

- Summary of units

LESSON 2 – CREATING CONTENT WITH JAVASCRIPT

UNIT1: Introduction

- We use JavaScript and DOM to manipulate DOM Elements
- We will see the power of combining JavaScript and DOM
- Download the course project: https://github.com/udacity/course-JS-and-the-DOM

UNIT2: Update Existing Page Content

- ".innerHTML", ".textContent", ".innerText"
- ".innerHTML": it is a property, get an element's HTML Content, Set an element's HTML content, returns a DOMString (technical name)
- ".outerHTML": represents the element itself and all its children
- ".textContent": set/return the text content of an element and all its descendants, it does not recognize HTML everything is displayed as a String/Text
- ".innerText": similar to ".textContent", returns Text as it would be seen on the webpage unlike "textContent" which returns the pre text without CSS formats

UNIT3: Add New Page Content

- ".createElement()", ".createTextNode()", ".appendChild()", ".insertAdjacentHTML()"
- https://developer.mozilla.org/en-US/docs/Web/API/Document/createElement
- ".createElement()": is a method on the "document" objects, it creates and returns an element
- ".appendChild()": is a method used to add an element to the page, it needs to be called o another element not the "document" object

- https://developer.mozilla.org/en-US/docs/Web/API/Node/appendChild
- ".createTextNode()": it creates new text nodes
- ".insertAdjacentHTML()": called by two arguments(location of the HTML, HTML text that is going to be inserted), needs to called on a parent element
- https://developer.mozilla.org/en-US/docs/Web/API/Element/insertAdjacentHTML

UNIT4: Removing Page Content

- ".removeChild()", ".remove()", ".firstElementChild", ".parentElement"
- ".removeChild()": removes child element, requires parent element and the child element that will be removed
- ".remove()": can be called directly on the element to delete "mainheading.remove()"

UNIT5: Style Page Content

- ".style.<prop>", ".cssText()", ".setAttribute()", ".className", ".classList"
- ".style.<prop>": can only modify one CSS style at time, ".style.color/fontSize"
- ".style.cssText": allows us to set multiple CSS styles at once
- ".setAttribute()": is not just for "styling" any other attribute can be set
- Let us just make our JavaScript control the className
- ".className": returns a string of all of the elements classes, convert the string into an array using JavaScript string method ".split(' ')", we can set a new className with it
- ".classList": returns a list, much nicer than ".className"
- Reading documentations one of the best ways to learn how a property or method works

UNIT6: Lesson Summary

LESSON 3 – WORKING WITH BROWSER EVENTS

UNIT1: Introduction

- What events are, how to respond and so on
- What is an event: like an announcement, so JS listens for this events and responds
- "monitorEvents()": A chrome browser function that lets us see different events as they are occurring, used mainly for development/testing purposes, not meant to be used for Production code

UNIT2: Respond to Events

- The "Node Interface" inherits from the "EventTarget Interface"
- Both the "document object" and any "DOM element" can be an event target
- "EventTarget Interface" has only 3 methods: ".addEventListener()", ".removeEventListener()" and ".dispatchEvent()"
- ".addEventListener()": lets us to listen for events and respond to them, it needs 3 things; the target, type of event to listen for, function to run when the events occurs (the listener)

- JS files are usually loaded at the bottom of the <body>
- https://developer.mozilla.org/en-US/docs/Web/Events: documentation

UNIT3: Remove an Event Listener

- ".removeEventListener()": to remove an event listener
- When working with JS Objects we have to think if they are two separate objects or different names referring to the same object
- ".removeEventListener()": needs three things; the target, type of event to listen for, the function to remove
- The function used by the "addEventListener" and that of the "removeEventListener" msut be equal (object/function equality)

UNIT4: Quiz – Phases of an Event

- Phases in the lifecycle of an event: "capturing" phase, "at target" phase and the "bubbling" phase
- "addEventListeners()" runs by default in "bubbling phase", setting the third argument to "true" makes it to run earlier at the "capturing phase"
- "Event Object", "event.preventDefault()": prevents default actions from occurring

UNIT5: Avoid too many Events

- "event.target": gives us access to the element that was clicked
- "event.target.nodeName(.toLowerCase())"
- Event delegation: is the process if delegating to a parent element the ability to manage events for the child elements, we used ".target" property for this

UNIT6: Know When The DOM is Ready

- HTML is received and converted into tokens and built into the DOM through a SEQUENTIAL PROCESS
- The placement of the JS file matters
- "DOMContentLoaded" event, the target should be the "document" object
- The "Network" pane of the DevTools shows the "DOMContentLoaded" time
- It would still be better to move the JS code to the bottom if the HTML file
- When to use "DOMContentLoaded"; when you want the JS code in the <head> to run before the one in the <body>

UNIT7: Lesson Summary

LESSON 4 – PERFORMANCE

UNIT1: Introduction

- How to measure the speed of your code, how to write efficient and performance code

UNIT2: Add Page Content Efficiently

- The standard way to measure how long it takes code to run is by using “performance.now()”
- “performance.now()”: returns a timestamp that is measured in milliseconds
- “DocumentFragment”: “.createDocumentFragment()”, it like a light weight DOM tree, it does not cause “reflow”

UNIT3: Reflow and Repaint

- “Reflow”: is the process of the browser laying out the page. Happens again every time something could change the layout. It is a fairly expensive (slow) process.
- “Repaint”: happens after reflow as the browser draws the new layout to the screen, it is fairly quick but we still want to limit how often it happens.
- “Virtual DOM”: what React and co uses to load pages faster
- We want to always reduce the number of reflows as much as possible

UNIT4: The Call Stack

- “Single-Threaded”: is simply the processing of one command at a time
- “Call Stack”: is basically a list of the functions that are running, helps JS to keep track of what functions are running

UNIT5: Code Synchronicity

- “synchronous”: existing or occurring at the same time
- “event listeners” are kinda “not synchronous”, they are executed at a later point in time
- “The JavaScript Event Loop”: “if some JS is running, let it run until it is finished. If no JS is running, run any pending event handler an repeat” > Rules of the Concurrency Model
- “setTimeout” and functions passed to “.addEventListeners” are Asynchronous codes

UNIT6: setTimeout

- “setTimeout()”: takes a function to run at some later time, the number of milliseconds the code should wait before running the function
- SetTimeout helps us to break up up Long running codes
- Check out documentations

UNIT7: Lesson Summary

- Both the code you write and how it performs are important

UNIT8: Course Summary

- Practice by building projects to demonstrate DOM events, put online share with others