SUMMARY OF UDEMY COURSE

JAVASCRIPT ESSENTIALS FOR ABSOLUTE BEGINNERS (2 SECTIONS)

RATINGS: 4/5

SECTION 1 – JAVASCRIPT ESSENTIALS (16 UNITS)

UNIT 1: Introduction

- A scripting language for the web
- Slack, Skype, Visual Code, Netflix, PayPal uses JavaScript in some capacity
- Table of Content
- About himself, coding since 1988
- JS is not a “strongly typed” programming language

UNIT 2: Variables, Data Types and More (first.js)

- We can write JS code on browsers using “developer tools”
- “console.log()”: to print
- “*** + **” : to concatenate strings
- “//”: used to comment
- Variables: pieces of data that sit on memory (RAM)
- Never use “var” (it does not have scoping), use “const” and “let” instead
- Use “const” as much has possible
- You cannot reassign “const” variable
- A “const” must have a valued (it cannot be undefined)
- You use “let” in Loops
- “` ** `”: the tick mark > `Amanda is ${variable name} years old` instead of
- “Amanda is ” + variable name + “years old”
- Data Types: number, string, Boolean, double/float, undefined (unassigned variables), null
- “prompt()”: to get input, it is a browser based function

UNIT 3: Conditions (conditionals.js)

- Execute code if a condition is True or False
- “if / else”
- “Comparison Operators”: “===” - equal, “!==” - Not equal, “< / > / <= / >=”
- Use “===” as much as possible (checks the value and the data type), instead of “==” (checks only the value)

UNIT 4: Arrays (arrays.js)

- Is a consecutive list of stuffs, that are group into one object
- “const names = [******]”
- Names[index_number], names.push() > to add to the array, names.pop() > to remove
- Names.lengh > number of stuff in the array
- Strings are kind of an array
- Array.from(***) > to convert *** to an array

UNIT 5: JSON Objects (objects.js)

- “const user = { }”: a legit JSON Object, “user.name”, “user.age”
- An object is nothing but a collection of key: values, similar to a dictionary
- The key is almost always a string
- You can make an array of objects

UNIT 6: Functions (functions.js)

- Grouping of a chunk of code into a function of work
- “function *** (params) { …return }”: function syntax
- Use to “Shift + Enter” for multiline in browser console
- Any code after “return” will not be “run”
- “return (number % 2 === 0 ? true: false)” > a way of return statement
- “new Date()”: to get current date time
- “setTimeout(function, time)”: is a function that executes a code/function after a specific time
- Use “setTimeout” in a function to create a sort of recursion

UNIT 7: Arrow Functions (arrowfunctions.js)

- They are compact and easily readable, used for function with just one line of code
- “const isEven2 = (params) => number % 2 === 0”
- “(params) => return statement”

UNIT 8: Loops

- For, For each
- “for (initialization > execute only once; conditional > checked after each loop; Incremental i++) { }”
- “for (let i = 0; i < names.lenght; i++) { if () { } }”
- “break”: to stop the loop, “return”: will also break the loop
- “names.forEach( )”: an array function, usually used with arrow functions
- “names.forEach(console.log)”: Prints everything, gives more info

UNIT 9: Dates (date.js)

- "new Date()": ".getTime()"
- Create a function with ".getTime()"

UNIT 10: Exercise (Exercise.js)

- Built a function "isPrime()", "getPrimeLessThan()"

UNIT 11: Array Maps (array_map.js)

- Mapping is simply to change the state of something to another state
- "users.forEach(u => console.log(u.name)}"
- "const userNames = users.map(e => e.name)": create list of name from the object

UNIT 12: Array Find (array_find.js)

- "users.find(e => e.name === "Hussein")": returns the first element in the array that satisfies the condition in the arrow function
- "users.filter(e => e.name === "Hussein")": returns all the elements in the array that satisfy the condition in the arrow function

UNIT 13: Sync vs Async (sync_async.js)

- Sync: you call a function and you wait for it to execute step by step
- "function isPrimeAsync(number, callback) {}"
- Async allows us to work on other stuff while code is still running
- JavaScript is a "single threaded" application

UNIT 14: Promises (promises.js)

- Promise is similar to callback but it is way better
- "A function with Promise always immediately 'return' a result"
- "function isPrimePromise ()"
- "new Promise((resolve, reject) => {resolve (***), reject = (***)})"
- "He consumes promises (using built in Promise Functions like "fetchAPI") more than he writes promise"
- "isPrimePromise().then(v => console.log(r)").catch(j => console.log(***+ JSON.stringify(e))
- ".then": deals with the resolve cases
- ".catch": deals with the reject cases

UNIT 15: Async/await (async_await.js)

- To give a feel of Async, and make it more readable
- We still need to add ".catch()" to Async/Await

UNIT 16: Fetch API

- Is a promised based API that allows to make a request/function to a web browser and gives back results
- "fetch(****)"
- Promises are chainable: fetch (***).then(arrow_function).then (arrow_function).***

SECTION 2 – BUILDING A CALCULATOR WITH JAVASCRIPT (9 UNITS)

UNIT 1: Getting Started (Calculator.html)

- We are building a Calculator

UNIT 2: Building User Interface (Calculator.html)

- Trying to mimic a simple calculator
- Have a goal then use the tool to reach/achieve the goal
- In HTML the table rows are defined first "<tr>" inside which we now define the columns as "<td>"
- By default in HTML "tables" don't have borders
- <table border = 1>
- "Going to grow to love Copy and Paste"

UNIT 3: DOM, Events & Functions (Calculator.html)

- GitHub Page for code: https://github.com/hnasr/javascript-by-example/blob/L1-Episode02/calc.html
- You sometimes need a unique identifier (ID) for each HTML elements to allow JS trigger an event on the object
- Event: something that happens then triggers lines of code
- The "id" attribute, attributes in HTML don't have to be in sequence
- "document.getElementById()", "alert()": is simply a browser popup box
- ".textContent"
- ".addEventListerner(event, function)": specify what event you are listening on, and what to do after the event is activated
- "Bug": an undesired functionality

UNIT 4: Arrow Functions (Calculator.html)

- "(params) => code"

UNIT 5: Evaluate Expressions (eval) (Calculator.html)

- "eval(**str***)": a built in JS function to evaluate a string

UNIT 6: Conditions (Calculator.html)

- If (***) {***}

UNIT 7: Running on Mobile (Calculator.html)

- Added “(” and “)”

UNIT 8: CSS, Arrays and Loops (Calculator.html)

- <style> *** </style> is usually in the <head> **** </head> for CSS
- Use a for loop for the addEventListeners()

UNIT 9: Debugging (Calculator.html)

- “getElementsByClassName(“symbol”)”
- “Adding breakpoint using Chrome Developer Tools”