SUMMARY OF UDACITY COURSE

OBJECT ORIENTED PROGRAM

RATINGS: /5

LESSON 1 – OBJECTS IN DEPTH

UNIT 1: Introduction

- JS helps to create dynamics in the web
- Delegations and Inheritance
- Recap of Arrays and Objects

UNIT 2: Creating and Modifying Properties

- Creating objects: using “object literal notation” or using “new Object ()” constructor function
- Snap shot
- To delete from an Object: “delete object.prop_names”
- Passing a Primitive: Snap shot, Primitives are immutable values
- Passing an Object: JS Objects are passed by reference, snap shot
- Comparing two objects: it will only return “true” when you compare two references to exactly the same object
- Objects are commonly created with “literal notation”

UNIT 3: Invoking Object Methods

- In an Object another name for a function property is a “method”
- Calling an object method: “object.prop_name(arg)”
- Snap shot
- A method can access the Object it was called on using “this” keyword, snap shot
- Depending on how a function is called, “this” can be set to different values
- A “method” is a function property of an object

UNIT 4: Beware of Globals

- “How the function is invoked determines the value of “this” inside the function”
- For methods, the value of “this” is whatever is “left of the dot” at invocation
- When a regular function is invoked, the value of “this” is the “global window object”
- The “window” object: snap shot
- Since the “window” object is at the highest level, every variable declaration that is made at the global level (outside of a function) automatically becomes a property on the “window object”
- Snap shots, only declaring variables with “var” will add them to the window object

- Global function are methods on the “window” object
- “Global variables and functions are not ideal”
- “Tight Coupling”: when codes are joined together in a way where changing one unintentionally alters the functioning of some other code, snap shot
- “Name Collisions”: occurs when two functions depend on a variable with the same name, snap shot
- “You should write as few global variables as possible”
- “window” object is provided by the browser and is not part of the JS language or specification

UNIT 5: Object Methods

- The “Object ( )” function actually includes a few methods of its own, these methods are “Object.keys(object_name)” and “Object.values(object_name)”
- At its core an object is just a collection of key/value pairs
- Snap shot
- We can use the “for…in” loop on the object to get the keys
- Support for “Object.keys(object_name)” has been around for quite a long time while
- “Object.values(object_name)” is significantly newer, it was added in 2017, all browsers might not support it yet

UNIT 6: Lesson Summary

LESSON 2 – FUNCTIONS AT RUNTIME

UNIT 1: Introduction

- Are functions objects or function objects??
- Learn about Immediately invoked functions

UNIT 2: First Class Functions

- Functions in JS are first class functions, it can be stored in variables, be returned from a function, be passed as arguments into another function
- Functions in JS can be used like the other elements such as strings
- A function that returns another function is known as “Higher Order Function”

UNIT 3: Callbacks

- We can pass functions as arguments because they are “first class functions”
- A function that is passed as in argument into another function is called a “callback function”
- Callbacks lead to cleaner and more efficient code
- Snap shot

- Functions are commonly passed into array methods and called on elements within an array
- ".forEach()": it takes a callback as an argument, snap shots
- ".map()": similar to "forEach" it involves a callback function for each element in an array, but it returns a new array, snap shot
- ".filter()": similar to map(), it also takes a callback argument and it returns a new array, the difference is that the function passed to "filter()" is used as a test, only items in the array that pass the test are included in the new array
- Snap shots

UNIT 4: Scope

- A functions runtime scope describes the variables available for use inside a given function
- A function has access to: its own arguments, local variables declared within the function, variables from its parent scope, to global variables
- Snap shot
- Variables in JS is function scoped, snap shot
- "Scope Chain": happens when our code attempts to access a variable during a function call
- Snap shot
- "Variable Shadowing": occurs when a variable with a local scope has same name with a global scope variable, the variable with the local scope will temporarily "shadow" the variable in the outer scope
- If there are naming overlaps between variables in different contexts, they are all resolved by moving through the scope chain from inner to outer scope

UNIT 5: Closures

- "Closure": is the process of a function retaining access to its scope. It also a combination of a function and the lexical environment within which that function was declared
- Scope and Closure are very closely related
- Every time a function is defined, closure is created for that function
- JS manages memory with automatic "garbage collection", that is when data is no longer referable it is garbage collected
- Closure is very useful for nested functions

UNIT 6: Immediately Invoked Function Expressions

- "Function declaration" defines a function and does not require a variable to be assigned it while "Function expression" does return a value, they are commonly assigned to variables as well. Snap shot
- IIFE: is a function that is called immediately after it is defined, snap shot
- IIFE help to create a private scope that protect variables or methods

- IIFE help to prevent creation of extra global variables, it is best for one-time task (initialization of an application)

UNIT 7: Lesson Summary

LESSON 3 – CLASSES AND OBJECTS

UNIT 1: Introduction

- Almost everything in JS is an Object
- Object in JS can represent real-life things
- Classes can be referred to as “categories of objects”
- Object-Oriented JS is a way to automatically classify Objects

UNIT 2: Constructor Functions

- Objects can be created using a “constructor function”
- A constructor function always starts with a “new” keyword, the function name starts with Uppercase, snap shot
- Constructor functions in JS should not have an explicit return value, that there should be no return statement, snap shot
- We can use the same constructor function to create multiple objects
- Constructor functions can have “parameters”
- “typeof”
- “object_name instanceof constructor_name” = true/false
- JS class system is built directly on using functions and objects

UNIT 3: The “this” Keyword

- Object, Function and “this” keyword are all interconnected in JS
- The value of “this” depends on how the function was called
- Snap shot
- The value assigned to “this” is based on the object that invokes the method where “this” is defined
- Calling a function on its own will set “this” to “window” global object, snap shot

UNIT 4: Setting Our Own “this”

- We can set the value of “this” by ourselves using “call ()”, “apply ()”, “bind ()”
- “call ()”: is a method directly invoked unto a function, snap shot
- “apply ()”: just like the call method, but the arguments are put in an array. Snap shot
- Snap shot
- “bind ()”: is also called on a function, but unlike call and apply, which both invoke the function right away, it returns a new function that when called has “this” set to the value we give it. Snap shot

UNIT 5: Prototypal Inheritance

- Inheritance in JS is when one object is based on another object
- JS uses “prototype” to manage inheritance
- “An object is secretly linked to its constructor’s prototype”, snap shot
- Prototype chain: snap shot
- “Dog.prototype.dog = function() {….}”
- “Object_name.hasOwnProperty(property_name)”: used to check for the origin a particular property, snap shot
- “isPrototypeOf()”: snap shot
- Object.getPrototypeOf(object_name): snap shot
- “constructor” property: snap shot
- Snap shot

UNIT 6: Prototypal Inheritance – Subclasses

- One of the benefits of inheritance is that it allows us to reuse existing code
- “subclass”: is to have a “child” object take on most or all of a “parent” object’s properties while retaining unique properties of its own, snap shot
- “it is highly discouraged to reassign the __proto__ property, or even use it in any code you write”
- Snap shot
- “Object.create(object_name)”: takes in a single object as an argument, and returns a new object with its “__proto__” property set to what argument is passed into it, snap shot
- “Object.create()”: gives us a clean method of establishing prototypal inheritance in JS, we can easily extend the prototype chain this way, and we can have objects inherit from just about any object we want
- Avoid mutating the prototype directly
- Snap shot

UNIT 7: Lesson Summary

UNIT 8: Course Outro

- I think there is still one more section, probably just updated

LESSON 4 – OBJECT-OREINTED SEDIGN PATTERNS

UNIT 1: Introduction

- Design patterns to create new objects, extend an object without involving its prototype
- “Mixins”, “Module Pattern”, “Revealing Module Pattern”

UNIT 2: Mixins/ Extending Object Functionality with Mixins

- JS only supports single inheritance
- “Mixin”: is a technique that takes the properties and methods from one object and copies them over to another object.
- “Object.assign(target, sources)”: Is the simplest way to implement the mixin pattern, snap shots, the order in which we pass the source object matters (overriding similar properties)
- Mixin was introduced in ES6 so check for browser compatibility
- Mixin does not set up prototype chain

UNIT 3: Functional Mixins

- Constructor function recap, snap shot
- “Factory Functions”: is a function that returns an object, but isn’t itself a class of constructor, snap shots
- “Functional Mixins”: is a composable factory function that receives a “mixin” as an argument, copies properties and methods from that mixin, and returns a new object
- Snap shots

UNIT 4: The Module Pattern

- Sometimes we want to make Object properties private/ non accessible form the outside
- Snap shot
- JS has no concept of private properties, there is no special syntax or keyword we can use to protect certain properties from being accessed
- We can use “scope” and “closures” to create a private state, snap shot
- “Module Pattern”: it leverages scope and closures and IIFE to create “privacy”, snap shots
- A function in a function creates a new scope
- Snap shots

UNIT 5: The Revealing Module Pattern

- It’s a slight variation to the “Module Pattern”
- We can choose to reveal certain properties or functions
- Key components are: IIFE, Module content, return Object Literals
- Snap shots

UNIT 6: Lesson Summary

UNIT 7: Course Outro