**Javascript Study Notes:**

JavaScript is a general purpose, multi paradigm programming language with dynamic typing.

Dynamic typing means that the type of a variable can change at run time depending on what is in the variable. Strings can change to numbers and so forth.

Possible Paradigms of JavaScript include:

Event driven. JavaScript can be made to respond to what the user does in the page like clicking and scrolling.

Functional. In JavaScript you can write pure functions, meaning that they always produce the same output for a given set of arguments and never produce side effects.

JavaScript also has First-Class functions, meaning that functions can be treated like any other variable for example being passed to an argument or being returned by another function.

JavaScript also has Higher-Order functions: a higher-order function is a function that accepts functions as parameters and/or returns a function.

**Objects**

Objects: In JavaScript, an object is an unordered collection of key-value pairs. Each key-value pair is called a property.

The key of a property can be a string. And the value of a property can be any value, e.g., a string, a number, an array, and even a function.

Objects can be made to inherit from each other.

Imperative or Declarative: Imperative code tells the computer how to do things and declarative code focuses on what you want from the computer. JavaScript can do either.

Primitives: In JavaScript, a primitive is data that is not an object and has no methods or properties.

There are 7 primitive data types: string, number, boolean, undefined, and null, symbol and bigint.

**Variables and Scoping:**

Let is a keyword for defining block scoped variables that cannot be accessed before they are initialized.

Var is a keyword for defining function scoped variables that is automatically initialized to undefined and hoisted. Generally don’t use var.

Javascript has three types of scoping.

Block Scope

Function Scope

Global Scope:

Let and Const have block scope.

Variables declared with let or const from inside a { } block cannot be accessed from outside the block:

{

let x = 2;

}

// x can NOT be used here

Function Scope: Variables defined inside a function are not accessible (visible) from outside the function.

Example:

function myFunction() {

let carName = "Volvo";

// Function Scope

}

Global Scope: A global variable has Global Scope:

All scripts and functions on a web page can access it.

let x = 2; // Global scope and can be used anywhere.

In general don’t use var, use let or const.

Const does not mean that the value of the variable cannot be changed. It just means that the variable identifier cannot be reassigned.

For example:

const SomeOb = {

id: 233,

name: null,

};

// allowed

SomeOb.name = "John Doe";

// not allowed as variable cannot be reassigned

SomeOb = 10;

Hoisting: In JavaScript var defined variables are hoisted to the top and given a value of undefined before the line on which they are declared:

// using test before declaring

console.log(test);

// this results in undefined

var test;

If I tried that with let it wouldn’t work. However if I defined a variable by just using something like x = 5 it would behave as if I defined it using var.x

Arrays

Arrays in JavaScript can be defined as

const arr = [1, 2, 3]

//bhv I am switching back to the Udemy Colt Steele course from the Algo Expert Course.

The order of operations can be remembered with the acronym PEMDAS (parenthesis, exponents, multiplication, division, addition and subtraction.)

The modulo is written with the % sign. So 9 % 2 (read nine mod 2) returns 1. Two goes into nine four times, with a remainder of 1. Modulo is used to determine if a number is even or odd.

To raise a number to a power in Javascript 2 \*\* 4 that is 2^4.

JavaScript treats negative numbers, positive numbers and decimals all alike.

NaN = Not a Number

This is technically considered a number. This comes from things like 0/0 or anything that isn’t a number when you are trying to do math with it. NaN \* 24 results in NaN.

What does 4 + 3 \* 4 / 2 evaluate to? Think PEMDAS. Answer 10. First 3 is multiplied by 4 then divided by 2 then 4 is added = 10.

Question: What does 200 + 0/0 evaluate to?

Answer: NaN (or Not a Number.)

Q. How would your increase the value of the number of hens stored in numHens by one?

A: numHens = numHens + 1;

Variables and Let

let wifeBirthYear = 1984;

let numChildren = 2;

If you define a variable called numAnimals = numHens + numFoxes and then later update numHens, what will happen to numAnimals?

Nothing. numAnimals is a snapshot in time and won’t automatically be updated.

If I wanted to track a game where you earned points, I could define the variable, let points = 10; later to add 10 to that I could write points = points + 10;

What's a shorter way of subtracting 10 from the numCallGirls variable than "numCallGirls = numCallGirls - 10;"

numCallGirls -= 10;

What’s a simpler way of subtracting one from the numMusketeers variable than numMusketeers -= 1;

You could simply write numMusketeers--; and obviously you could add a Musketeer by typing numMusketeers++;

Another way to define a variable is const, which is an abbreviation of constant. One way to use this is for things that don’t change like the number of centimeters in an inch which is defined by law at 2.54 cm / in . So to use this I would write const numCmPerInch = 2.54;

Can a const variable containing a simple number be changed later?

No.

Var is the original way of writing variables. Generally you should not use var use let.

Use const to define the number of months in a year. Const monthsPerYear = 12;

Booleans:

Booleans can only be true or false in lower case in JavaScript. No other possibilities. It’s never True and never False.

Use a Boolean to write that he or she loves you. Let heSheLovesMe = true;

Can variables change their type in JavaScript? Yes, they can. They can store strings and then later change to storing numbers or whatever primitive you want to store.

Variable Naming Conventions:

When you are naming a variable using any of the methods, use what is called camel case. So you write mySpecialVariable, capitalizing the first letter of the second and further words, but lower casing everything else.

Boolean naming advice: Often adding the word “is” to the beginning of the Boolean variable name can make it clearer. isHellFrozen = false; is a good clear variable name.

Strings:

Strings are textual information. They must be wrapped in “quotes”. You can use ‘single quotes’ or “double quotes.” Be consistent in whichever one you choose.

if (num % 2 ==0) {

console.log("even");

}