**Javascript (client side)**

1. A Script is placed in a separate file. (external) It can be placed anywhere on the HTML file
2. It can also be embedded in the HTML file (internal)
3. Inline script (not desirable)

**TYPES**

* Top-level/global code - executes on the fly, non-member of a function code
* Function - has to be called to execute

**External Attribute**

* Defer - fetch and continue rendering the page
* Async - execution becomes asyncronous, compiles the script after it’s being fetched
* No Script - notifies users if the browser doesn’t support scripting
* Window - global object, access to other objects
* Document object - gain access to any elements, allows interaction between JS and HTML
* Outer text
* Outer HTML
* Inner HTML - allows html elements to be parsed
* Inner vs outer - outer replaces the whole markup

**Document**

* getElementByID - only global
* getElementByClassName
* getElementByName
* getElementByTagName

Query - Specify a css selector

Matches - sees if the object matches the selector

Query selector/all - for retrieving

- pass CSS selector tags

* .matches - to check if the body matches the head
* .style - changes the style

- values are always a string

Node Interface - the primary data type for the entire DOM

\* node value is always null, can also be used for changing node values

**Check Node Types**

* firstElementChild
* lastElementChild
* nextSiblingElement
* previousElementSibling
* parentElement

**Dynamic Content Creation**

* createElement()
* insertBefore()
* replaceChild
* removeChild
* shallowClone
* createDocumentFragment

Java Script (the defacto client - side scripting language)

- is an imperative programming language

- weakly typed

**Variables** - ECMA script 2016

* Var - the same inside a function, becomes a property of the global object

- only exists inside the function its declared on

* Let - varies depending on which bracket

- only exists inside the block it declared in.

- if declared in the top - level, variable is just standard

* Const - cannot be reassigned/ needs initialization

- if declared in the top - level, variable is just standard

**Data types**

Simple/primitive

* boolean = simplest

“falsy” = not really false but equivalent to false

“truthy” = not really true but equivalent to true

\*0 is considered a false

\* null, undefined or empty string is false

* Numbers - are only considered numbers
* Strings - no char type

- use backslash to escape a quote

- template literals [` `] for multiple lines

- string interpolation

* Undefined
* Null

Standard Objects - all reference types begin with object

**FUNCTIONS**

* Procedural
* All functions return a value
* Type coercion

- functions can be recursive

n! = n \* (n-1)

0! = 1

\*”throw” can throw anything

* - Functions can be nested
* - “this”, this object which you are invoking

“arguments”, arguments you use to call, organized like arrays.

- only exists in the function

* Functions can have default values
* Function rest parameter

**Structures**

* Array
* Wrapper Classes

**ARRAYS**

var emptyArray = new Array ();

var alsoEmptyArray = [];

Var arrayWithLengthFIve = new Array(5);

Var arrayWithOneElementWithValue5 = [5];

Var array = new Array(5, 10, 15);

Var sameArray = [5, 10, 15];

Var mixedElementType = [10, true, ‘hello’, new Date()];

Array destructing - extract element and assign to different values

Var array = [1, 2, 3, 4, 5];

Var[a,b,c,d,e] = array;

Va[m,n,… others] = array;

Var[, x,, y] = array;

- array indices can be non - contiguous

Var array = [1,2,3,4,5]

Array [10] = 10;

Array Methods

Mutator methods - change the array value “mutate”

Iteration methods - iterate through methods one by one and apply a function.

Objects - based on a prototype of an object

Var emptyObj = new object();

Var alsoEmptyObj = {};

Ex: student.idno = ‘2150387’;

Student[‘name’] = ‘Juan Dela Cruz’;