**Javascript Lesson Plan**

**Lesson 1: Introduction to Javascript**

JavaScript was created by Brendan Eich in 1995 during his time at Netscape Communications. It was inspired by Java, Scheme and Self.

Javascript is not related to Java.

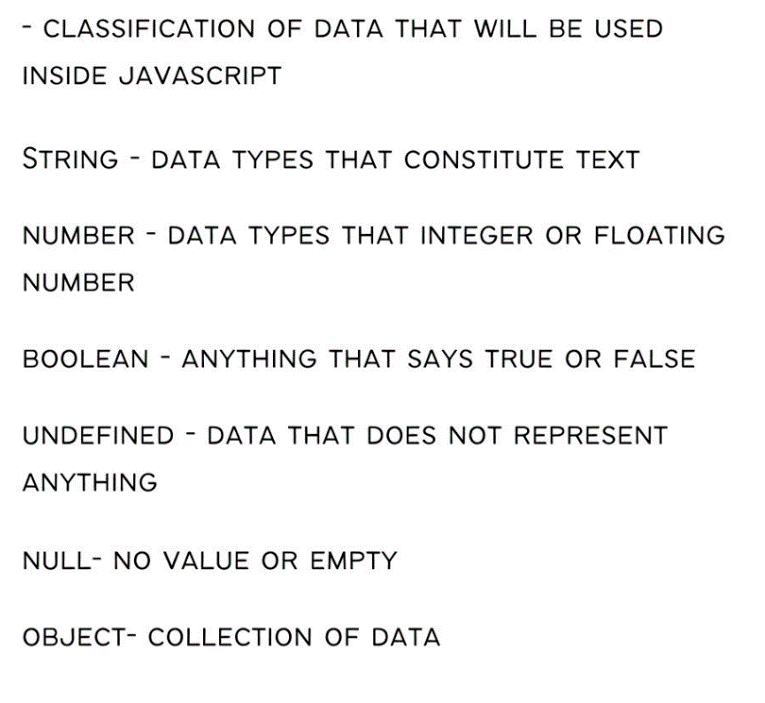
Some say that it’s a “toy” programming language before.

Internal JS – External Js

**Lesson 2: Variables and Data Types**

**Declaring variables** with **var (non-block scope), let,** and **const –** They are the ones that holds the data

**Data types** – This are the categories of data inside javascript

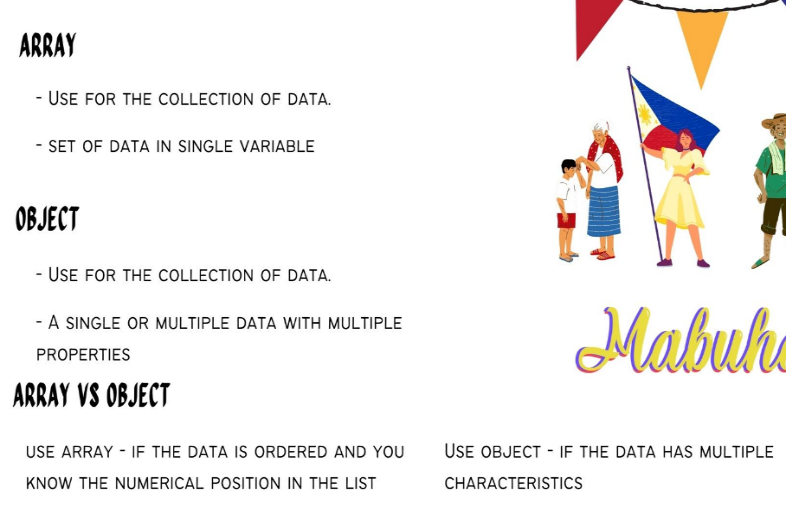


Date - for date

**Changing types**

**ParseInt, ParseFloat**

**String**

 **Arrays and Objects**

**Lesson 3: Operators and Expressions**

**Arithmetic Operations** – The application of mathematical operations (+, -, \*, /, %)

**Comparison operators** - (>, <, >=, <=, ==, ===, !==, !=)

**Logical operators** - (&&, ||,!)

**String Manipulation**  -

Concat – combine string

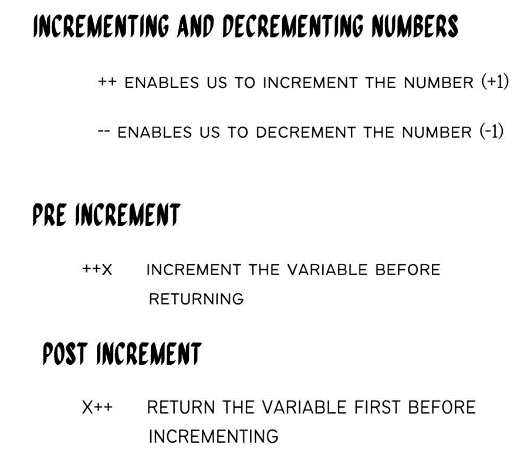
Template Literal – show the exact code with variables

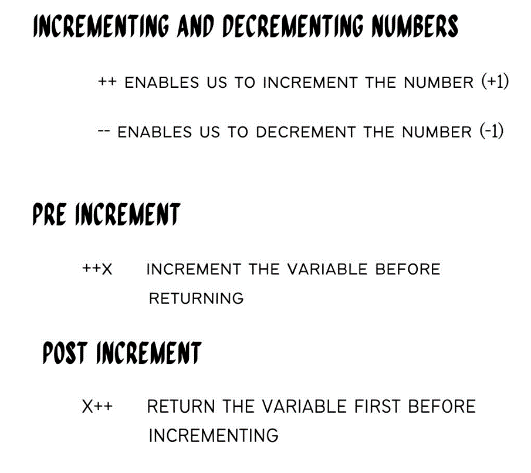
Slice - It allows you to extract a portion of an array or string without modifying the original data.

Split - used to divide a string into an array of substrings based on a specified separator.

Substring - used to extract a portion of a string based on specified start and end indexes.

Convert to lower/upper case – converting to capital letter / small letter





**Lesson 4: Control Flow Statements**

**Conditional statements:** if, else if, else

**Ternary operator** for conditional expressions

Looping with **for and while loop**

Breaking **loop execution**

**Acivity1: Draw an Asterisk Tree using loop**

**Lesson 5: Functions and Scope**

**Javascript function** - a set of statements that performs a task or calculates a value

**Function parameters and arguments** –

Parameters – names listed in function

Arguments – real values passed in the function

**Function scope**

Local scope– variable declared inside a function

Global scope – variables declared outside function

**Return** – use to return a data to the caller of function

**function declarations / Function expressions / IIFE**

Function declaration – just an ordinary function. / Traditional – Non hoisting

Function expression – a function inside a variable. (arrow functions) - Hoisting

* Modern way

IIFE (Immediately Invoked Function Expressions) – functions that execute immediately

**Lesson 6: Array Methods**

**Array methods:**

**Push** - adds new items **to the end** of an array.

**Unshift** - adds new elements to **the beginning** of an array.

**Pop** - removes (pops) **the last element** of an array.

**Shift** - method removes **the first item** of an array

**Iterating over arrays:**

Iterating over an array means going through each element in the array one by one and performing some operation or action on each element.

For loop – get the data by looping and based the array index on the current loop

**Higher-order functions**

A higher-order function is a special kind of function that can either accept other functions as inputs or produce functions as outputs.

forEach(): Calls a provided function once for each element in an array, but doesn't create a new array.

map(): Creates a new array by applying a function to each element of an existing array.

filter(): Creates a new array containing elements that pass a specified test (provided as a function).

splice - changes the contents of an array by removing or replacing existing elements and/or adding new elements in place

slice - returns a [shallow copy](https://developer.mozilla.org/en-US/docs/Glossary/Shallow_copy) of a portion of an array into a new array object selected from start to end (end not included) where start and end represent the index of items in that array. The original array will not be modified.

sort(): Sorts the elements of an array . (use reverse for desc) (String based)

every(): Checks if all elements in an array pass a test (provided as a function). It returns a Boolean value.

some(): Checks if at least one element in an array passes a test ( It doesn't modify the array.).

find(): Returns the first element in an array that passes a specified test (provided as a function).

findIndex(): Returns the index of the first element in an array that passes a specified test (provided as a function).

isArray(): Checks if a value is an array.

` concat(): Combines two or more arrays.

**Lesson 7: Working with Objects**

**Accessing and modifying object properties**

Add, delete and modify properties

**Adding methods to objects**

Inserting a function in object

**Lesson 8: Dom Manipulation**

The JavaScript capability to control the elements

**Accessing DOM Elements**

**Selecting elements using different methods:**

getElementById – Selecting the element by id (specific)

getElementsByClassName - Selecting the element by classname

getElementsByTagName – Selecting element based on html tag

querySelector - returns the first [Element](https://developer.mozilla.org/en-US/docs/Web/API/Element) within the document that matches the specified selector, or group of selectors.

const paragraph = document.querySelector("p");

const highlightedElement = document.querySelector(".highlight");

const header = document.querySelector("#header");

querySelectorAll – returns all the element that you selected

**Manipulating DOM Elements**

The DOM can change almost everything about HTML element such as:

InnerHTML , CSS, className, value, checked ,hidden, readonly

**Appending, Creating, and removing elements**

**Eventlisteners / Events**

Event listeners are used to attach functions (event handlers) to specific events, allowing you to execute custom code when those events occur.

**click**: Triggered when an element is clicked.

**submit**: Triggered when a form is submitted.

**keydown** / **keyup**: Triggered when a keyboard key is pressed or released.

**mouseover** / **mouseout**: Triggered when the mouse pointer enters or leaves an element.

**change**: Triggered when the value of an input or select element changes.

**Input** – Triggering when a value in inputted.

**Lesson 9: Synchronous vs Asynchronous**

**Synchronous –** Each task is completed before the next one starts, creating a clear and predictable flow. However, if a task takes a long time to finish, it can block the entire program's execution, leading to potential responsiveness issues.

**Asynchronous -** programming is a way of executing tasks without waiting for the previous task to complete. It enables other tasks to continue running while certain operations are being performed.

**Examples of asynchronous functions / Scenarious**

1. SetTimeout
2. Set Interval / Clear Interval
3. Event handlers
4. File uploading
5. Ajax

Techniquest to handle asynchronous code (Waiting functions)

**Callbacks -** function passed as an argument to another function.

Callback hell - The situation where callbacks are nested within other callbacks several levels deep, potentially making it difficult to understand and maintain the code.

**Promises -** represents the eventual completion (or failure) of an asynchronous operation and its resulting value.

**async await**

**Lesson 10: Ajax Request**

**What is Ajax?**

**JSON OVERVIEW**

**How To make a request**

**Activity: Try to submit the data from the registration form going to PHP and displaying it back to a table**