* Starting JavaScript
  + Has a .js file tag
  + Is a loosely typed scripting language
    - Loosely means you don’t declare the type of the variable when declaring it
    - Dynamically typed
      * The type of the variable is determined at runtime
      * The variable can change datatypes
      * Uses implicit conversion
  + Is run client side by the browser
  + Provides the functionality of our web pages
  + Can be run by all major browsers
  + Interpreted not compiled
  + Cannot be run outside the browser
    - Except Node.js
  + Object-oriented, functional or procedural programming
  + Was created initially to interact with the DOM and increase the functionality of our web pages
  + The interpreter actually runs through the file multiple times before running it
    - This will be important for why things get hoisted
  + To print out to the console we use console.log()
  + Can write JavaScript in 3 different places
    - Inline
      * Written in the event button itself
    - Internal
      * In the html file in a script tag
    - External
      * Written in an external file
      * To reference, you can link in the head or body
        + Always put js in the bottom of your page
        + Set <script src=’path’ > </script>

Do not make it self closing

* + Is case sensitive
  + Variables
    - Container used to store values and are not declared with data types
    - Declared with 1 of 3 scope defining keywords
      * Var
        + Use var to declare a variable exists in global or function scope
      * Let
        + Can be used to declare a variable as having global function or block scope
      * Const
        + Defines a constant valued variable with block scope
    - Big difference between var and let
      * Do not use var
      * Can reference before use
  + Hoisting
    - The interpreter actually runs through the file a few times before it runs it
    - When it does so, It pulls all global variables defined with var to the top of the page
    - Will also do so with some functions
    - Var we can use in block scope but it wont be enforced
      * The var declared variables will be hoisted into function scope
  + Scope
    - Global
      * Anything outside of a function, just in the file
    - Function scope
      * Anything declared inside of a function
    - Block
      * Anything declared within that block of code
  + Data types
    - String
    - Number
    - Boolean
    - Undefined
    - Symbol
      * Kind of an object
      * Represents a unique identifier
    - Null
  + Undefined vs null
    - Undefined means that a variable has yet to be declared with a value
    - Null is used to represent a value where it intentionally does not have a value
  + You can use a typeof() to get the type of a variable
  + Anything that is not a primitive is an object
    - Includes collections, arrays, objects or functions
  + Lists and arrays are almost identical in javascript
    - Arrays are 0-indexe lusts of elements that contain comma-separated values
    - You can use different data types and they are dynamically sized
    - let myVar = [];
    - let myOtherVar = [], ‘hello’, null, 12];
    - Collections in js are arrays and objects
    - Object members are accessed via name
      * myotherVar[1];
  + Objects
    - Key-value pairs with the object literal, curly braces, and optional comma-seperated key value paris
      * let obj = {};
      * let obj2 = [

key1: 1,

key2: “Some val”,

key3: undefinined

};

* + - * To access a key you defined with a string, use obj2[“Key 3”]
  + Functions
    - A bit of code you can call based off its name
    - Can be defined in several different ways
      * Using the function keyword followed by the name of the function and parentheses containing the parameters
      * Can be assigned to a variable
        + We will see this more later
        + You reference them by the variable name assigned to them
        + Called anonymous functions
        + var func1 = function(){

console.log(‘This is anon’);

}

To call,

func1();

* s
  + - * + You can run the function as well as assigning it’s value to something else
  + Truthy vs Falsy values
    - Truthy values are values that equal true
    - Falsy values are values that equal false
    - This is done with implicit conversion
    - Everything that isn’t a falsy is a truthy
    - Falsy
      * 0
      * Null
      * False
      * An empty string
      * Undefined
      * NaN
    - == compare value and not type
      * There is implicit type conversion
    - === compares value and type
      * Will not implicitly convert data types
  + Flow control statements
    - If-else
    - If-else if-else
      * Both of these look standard
    - Switch statements
      * Use break after each switch
      * Identical to c++
    - For loops
      * Several different types
      * For(let I = 0; I < 3; I++){
      * For(val in obj2){
        + Loops through the properties of an object
        + Use obj[val] to reference that value where val is the key
      * For(v of myArray){
        + Loops through values of an integrable object
        + Like the advanced for loop in apex
    - While loop
      * Identical to apex
    - Do while loop
      * Same as apex
  + DOM
    - Our way of interaction with the DOM or webpage
    - Document object model
      * Document: the html page
      * Object: the elements on the page
      * Model: the way we interact with, change, edit and add our elements
    - Model:
      * A picture containing diagram

        Description automatically generated
    - How we interact
      * We can access them by using a few different methods
      * getElementsById()
      * getElementsByClassName()
      * getElementsByTagName()
      * querySelector()
  + Events and event listeners
    - Events are actions that happen inside of the browser
    - Event listeners wait for and provide a response for an event and can dynamically respond to them
    - Document.getElementById(‘test’).addEventListener(‘EventName’, functionName)
  + Event propagation
    - The order in which elements are able to handle events
    - This can be done with bubbling or capturing
    - 3 phases
      * Capturing
        + The event travels from the top of the DOM down to the element that triggered the event
        + If we propagate events in the capturing phase they will be triggered from parent to child
      * Bridge
        + Event is at the source element
      * Bubbling
        + Source element up the DOM tree
        + This is the default phase of propagation
      * Can use stopPropogation
      * stopImmediatePropogation()
        + Stop all actions that may occour
  + AJAX
    - Asynchronous javascript and xml
    - How we communicate from client side and server
    - Window.onLoad()
      * When the window finishes loading will execute whatever you bind to it
    - Steps
      * Instantiate the XMLHttpRequest object
        + This has an onreadustatechange property that holds a function that is called when the xhr’s ready state changes
        + let xhr = new XMLHttpRequest();
      * Check for ready state change
        + 0 means not initialized
        + 1 means set up
        + 2 means request was sent
        + 3 means request is in progress
        + 4 means request is complete
        + xhr.onreadystatechange = function(){
        + Check for this.readystate == 4 && this.status == 200
      * Open our request
        + xhr.open(‘GET’,’URLendpointGoesHere’)
      * Send the request
        + xhr.send()
      * Parse the data
        + let tex = JSON.parse(this.response)
        + Can use tex as the return object since this will return an object