Java vs JavaScript

|  |  |
| --- | --- |
| Java | JavaScript |
| Ham | Hamster |
| Compiled | Interpreted |
| OOP | Cake-Salad (mostly functional but also OOP) |
| Access Modifiers | No Access Modifiers |
| Multi-threaded | Asynchronous |
| Strongly Typed | Loosely Typed |
| Compiled/non-scripting | Scripting |
| Backend | Mostly frontend |
| Classical Inheritance | Prototypal Inheritance |
| No Global Scope | Global Scope |
| Automatic Memory management | Automatic memory management (done by browser) |
| Universal JVM | Interpreter |
| Extensive Exception handling | Simple try catch blocks |
| Different primitives | Different primitives |

JS types and type coercion

* JS does have primitives
  + boolean
  + number
  + string
  + null (Explicitly given to mean it does not exist)
  + undefined (Default value of everything in JS)
  + symbol (We do not cover)
* JS has aggressive type coercion
  + JS will always find a way to compare and perform operations between your variables
* Truthy falsy
* All values in JS are either inherently true or false
  + Everything is truthy except certain falsy values
  + Imply some kind of emptiness or non-existence
    - 0
    - “”
    - null
    - undefined
    - false
    - []
    - NaN

Scopes in JS

* Scopes are defined using keyword (or lack thereof)
  + Global (no key word)
  + Function (var)
    - Friends do not let friends use var
  + Local (let or const)

Objects

* Objects are just collections of key value pairs
* Every key value pair is called a property
* You do not need a class to create an object
  + Object Literal let adam = {name:”Adam”, age : 19};
* All objects inherit from Object
* A method is a function attached to an object
* Use keyword in to iterate over an object

Functions

* Functions are objects!!!!!
* Functions are first class objects/citizens
* Special types of functions
  + Callback function
    - Function passed as a parameter into another function
  + Higher order function
    - Function that takes in a call back function
  + Anonymous function
    - Function created without a name
    - Not using let x = function() or function hello()
  + IIFE (Immediately Invoked Function Expression)
    - Aka self-invoking function
    - Provides a tool to create encapsulation
  + Arrow notation function
    - () =>{ code}
    - Alternate syntax to function keyword
    - Binds the this to the parent execution context

Arrays

* Arrays can contain anything
* They are dynamically sized
* Use keyword of to iterate through an array easily
* Key functions associated with arrays
  + Push (adds to end)
  + Pop (removes last element)
  + forEach (takes in a callback to perform for each element)
  + filter
  + map
  + slice
  + reduce

Closures

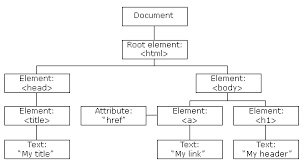
1. Closures occur when you have a function within a function
2. The inner function when returned will have access to all variables within the outer function
3. Even though the outer function has finished executing

Prototypal inheritance

* JS does not support classical inheritance
* Objects can inherit properties directly from another object
* Every object has a \_\_proto\_\_ property that points to their prototype object to inherit from
* Custom Objects inherit from Object
* Custom Functions => Function => Object
* Custom Arrays => Array=>Object

DOM (Document Object Model)

* DOM is a representation of your html page as a tree
* Aka a system of nodes connected to each other



* JS was invented to manipulate the DOM
* This makes your webpage dynamic