ANGHELA M. ALIZA
BSIT 3B

Midterm - Lab_Activity2_The Javascript Basics

The main difference is that Java is an object-oriented, statically typed, and compiled language, while JavaScript is a dynamically typed, interpreted scripting language. This distinction affects how each language is written and executed.

	Java	**JavaScript**

Function Syntax	• Functions are always defined **inside classes** as methods.
	• Each method must have an **access modifier** (public, private, etc.), a **return type** (int, String, void), and **parameter types**.
	• Java requires explicit data types for both parameters and return values.
	• Methods are enclosed in curly braces {} and cannot exist outside a class.
	• Supports **method overloading**, allowing methods with the same name but different parameters.
	• Java is **statically typed**, meaning errors are checked during compilation.
	
	Example:
	public static void greet(String name) {
	System.out.println("Hello " + name);
	}
		• Functions can exist **outside of classes** and do not require access modifiers.
		• Declared using the function keyword, **function expressions**, or **arrow functions** (=>).
		• No need to declare data types for parameters or return values.
		• JavaScript is **dynamically typed**, so types are determined at runtime.
		• Functions are **first-class objects** they can be stored in variables, passed as arguments, or returned by other functions.
		• More flexible and less strict than Java in syntax and structure.
		
		Example:
		function greet(name) {
		console.log("Hello " + name);
		}

Calling Methods	• Methods are called using an **object** or **class name**. • Syntax: object.methodName(args); or ClassName.staticMethod(args); • The number and type of arguments must exactly match the method definition. • Java enforces **compile-time type checking**, preventing mismatched data types. • Code execution must occur within methods standalone code is not allowed. **Example:** Person p = new Person(); p.greet("Tasha");	• Functions can be called **directly by name** or as methods of an object. • Syntax: functionName(args); or object.method(args); • JavaScript allows **flexible arguments** extra ones are ignored, missing ones become undefined. • Functions can use .call() or .apply() to change the context (this). • Code can run **outside functions**, such as directly in the browser console or script. **Example:** greet("Tasha");

ANGHELA M. ALIZA
BSIT 3B

Statements	• Requires **explicit data types** for variable declarations (e.g., int, String, double). Example: int age = 25; • Statements must exist **inside methods** or class blocks. • Uses common control flow statements: if, else, switch, for, while, do-while. • Supports enhanced for-each loops. • Java is **statically and strongly typed**, so variables cannot change types once declared. • Supports **multithreading** for parallel execution of tasks. .	• Variables are declared using var, let, or const **no type declaration** needed. Example: let age = 25; • Code can run globally (outside of functions or objects). • Uses the same control flow structures as Java: if, else, switch, for, while. • Variables declared with let and const are **block-scoped**; var is **function-scoped**. • **Dynamically typed** variable types can change during runtime. • Follows an **event-based concurrency model** using callbacks, promises, and async functions.

Source:

https://www.geeksforgeeks.org/javascript/difference-between-java-and-javascript
https://www.coursera.org/ca/articles/java-vs-javascript