NAME: ANDE, JESSICA DANIELLE C. DATE: 0CT 22,2025
SECTION: BSIT-3B

# Differences Between Java and JavaScript in Terms of Function Syntax, Calling Methods, and Statements

**Java and JavaScript, despite sharing part of their names, are two distinct programming languages with key differences in their syntax, method calls, and statements. Below, we'll examine these differences in detail.**

## 1. Function Syntax

**Java:**

In Java, functions are referred to as "methods" and must be defined inside a class. The method syntax requires explicit data types for both parameters and return values. Here's an example of a basic method in Java:

```
public class Example {
  public static int addNumbers(int a, int b) {
    return a + b;
  }

  public static void main(String[] args) {
    System.out.println(addNumbers(3, 5));
  }
}
```

**Key Points:**

The return type must always be declared (int, void, etc.).

Methods must be defined within a class.

The main method serves as the entry point for the application.

**JavaScript:**

In contrast, JavaScript has a more flexible syntax for defining functions. Functions can exist independently (outside of any class) and don’t require explicit data types. JavaScript supports both traditional function declarations and modern arrow functions. Here's an example:

NAME: ANDE, JESSICA DANIELLE C. DATE: 0CT 22,2025
SECTION: BSIT-3B

```
function addNumbers(a, b) {

   return a + b;

}


console.log(addNumbers(3, 5));
```

**Key Points:**

No need to specify return types or data types of parameters.

Functions can be defined anywhere in the code (even outside classes).

JavaScript allows anonymous functions and arrow functions (const addNumbers = (a, b) => a + b;).

**2. Calling Methods**

**Java:**

In Java, methods are invoked on objects (instances of a class) or from the class directly (for static methods). A method call follows the format of specifying the object or class and the method name.

```
Example obj = new Example();

int result = obj.addNumbers(10, 5);
```

**Key Points:**

Methods can be called on instances (for non-static methods) or on the class itself (for static methods).

The method name must always match exactly, including case sensitivity.

**JavaScript:**

JavaScript allows for method calls on objects, and since JavaScript is loosely typed, the method calls can be more flexible. Methods can also be called as part of event handling or passed around as arguments.

```
let obj = {

   addNumbers: function(a, b) {
```

```
return a + b;
    }
};

let result = obj.addNumbers(10, 5);
```

**Key Points:**

Methods can be called on objects, but the syntax is simpler and more flexible.

JavaScript supports dynamic typing and allows functions to be passed around or assigned to variables.

**3. Statements**

**Java:**

Java follows a strict and structured syntax for statements. Every line of code is a statement, and the use of semicolons is mandatory to mark the end of a statement. Additionally, blocks of code within classes, methods, and control structures (like loops) must be enclosed in curly braces {}.

```
if (x > 10) {
    System.out.println("Greater than 10");
}
```

**Key Points:**

Statements in Java end with semicolons.

Code blocks in control structures are always enclosed by braces.

Java requires strict adherence to syntax rules, with little flexibility.

**JavaScript:**

JavaScript is more flexible with statements and often allows for semicolons to be optional, though they are still recommended. JavaScript also uses braces for control structures but can often work with fewer syntactical restrictions.

NAME: ANDE, JESSICA DANIELLE C. DATE: 0CT 22,2025
SECTION: BSIT-3B

```
if (x > 10) {
  console.log("Greater than 10");
}
```

**Key Points:**

Statements in JavaScript may or may not require semicolons (due to automatic semicolon insertion).

JavaScript’s syntax is more forgiving, which allows for more concise code.

JavaScript can also use control structures and loops with fewer constraints on their formatting.