Introduction to JavaScript

What is JavaScript?

JavaScript (JS) is a programming language that allows you to create dynamic and interactive content on web pages. It runs in browsers and is widely used for:

- Web development (frontend and backend with Node.js)
- Game development
- Mobile applications
- Server-side applications

Setting Up JavaScript

There are two primary ways to run JavaScript:

1. **In the Browser**: Open the Developer Console (F12 or Right-click > Inspect > Console)
2. **Using Node.js**: Install Node.js and run JavaScript files using node filename.js in the terminal.

Your mentor will decide which method to use for your learning process.

Console.log

The console.log () method is used to display output.

- **Browser**: Open the console in Developer Tools.
- **Node.js**: Run the script using Node.js, and logs appear in the terminal.

Example:

```
console.log ("Hello, World!");
```

Variable Declaration and Assignment

Variables in JavaScript store values and can be declared using var, let, or const. Each has different behaviour regarding scope and reassignment.

var (Function-scoped)

- var declarations are **function-scoped**, meaning they are only available within the function in which they are declared.
- var **does not support block-scoping**, meaning a variable declared inside an if block or a loop is still accessible outside that block.
- Variables declared with var **can be redeclared** within the same scope.
- Variables declared with var **are hoisted**, meaning they are moved to the top of their scope during execution but remain undefined until assigned a value.

Example:

```
function exampleVar() {
  if (true) {
    var x = 10;
  }
  console.log(x); // Works, because var is function-scoped
}
exampleVar();
```

***Avoid using var** unless necessary due to its unpredictable behavior.*

let (Block-scoped, allows reassignment)

- let is **block-scoped**, meaning it is only accessible within the block {} where it is defined.
- It **can be reassigned** but **cannot be redeclared** within the same scope.
- It is also **hoisted**, but unlike var, it is not initialized, leading to a ReferenceError if accessed before declaration.

Example:

```
function exampleLet() {
  if (true) {
    let y = 20;
    console.log(y); // Works
  }
  // console.log(y); // Error: y is not defined outside the block
}
exampleLet();
```

***Use let** when you need to change the value of a variable.*

const (Block-scoped, cannot be reassigned)

- const is **block-scoped**, similar to let.
- It **must be assigned a value upon declaration** and **cannot be reassigned**.
- It **cannot be redeclared** in the same scope.
- const does **not make objects immutable**—only the variable binding is constant, not the contents of an object or array.

Example:

```
const test = 1;
test = 2; // Error: Assignment to a constant variable
```

***Best practice**: Always use const unless you need to reassign the variable.*

const with Arrays and Objects

```
const arr = [1, 2, 3];
arr.push(4); // Allowed, modifying the contents of the array
console.log(arr); // [1, 2, 3, 4]
arr = [5, 6, 7]; // Error: Assignment to constant variable
const obj = { name: "Alice" };
obj.name = "Bob"; // Allowed, modifying object properties
console.log(obj.name); // Bob
obj = { age: 25 }; // Error: Assignment to constant variable
```

Variable Types

JavaScript has several data types:

String

```
const text = "Hello";
console.log(text.length);
console.log(text.toLowerCase());
console.log(text[0]);
```

Number

```
let num = 42;

console.log(typeof num);

console.log(num + 3.5); // JS treats integers and floats the same
```

Boolean

```
const isTrue = true;

console.log(typeof isTrue);
```

Array

```
const shoppingList = ["egg", "apples", "flour", 3, false];

console.log(shoppingList[1]);
```

Undefined

```
let undefinedVar;

console.log(undefinedVar);
```

Comparison Operators

Equality Operators

```
console.log(7 == "7");  // true (loose equality, type coercion)

console.log(9 === "9"); // false (strict equality, no type conversion)
```

Relational Operators

```
console.log(10 > 5);  // true

console.log(10 >= 10); // true

console.log(5 < 2);   // false
```

Arithmetic Operators

```
console.log(10 + 5);  // 15

console.log(10 - 5);  // 5

console.log(10 * 2);  // 20

console.log(10 / 2);  // 5

console.log(10 % 3);  // 1 (modulo)
```

Debugging Errors

Learning how to read and fix errors is crucial.

```
const test = 1;

test = 2; // Error: Assignment to a constant variable
```

Exercises

Variable Types

Write down what these statements will log before running them:

```
    console.log(typeof 3);

    console.log(typeof "3");

    console.log(typeof [3]);

    console.log(typeof true);
```

Follow-up Exercises

```
Create a variable that is 24 * 55.

Declare a const with your name.

Log the first character of your name.

Create an array with three strings, three numbers, and three booleans.

Log the 4th element of the array.

Optional: Log the last character of your name.
```

Fix the Errors

Identify and fix the issues in this script:

```
const name = "benjamin";

name = "benjamin-better"; // Error: Assignment to constant variable

const pizzaPrice = 78;

const pizzaPriceDiscounted = pizzaprice - 10; // Error: Undefined variable

const users = ["peter", "Johnny", "Børge"];

const lastUser = users[3]; // Error: Out-of-bounds indexing
```

Practice Project: Burger Order System

Part 1

1. Create a new folder called "burger-order".
2. Inside the folder, create an HTML file called "index.html".
3. Also, create a JavaScript file called "burger.js".
4. Link the JavaScript file to the HTML file using a <script> tag.
5. Add a log statement to verify that the script is running:

console.log("Welcome to the Burger Order System!");

6. Create variables to store:
    - ✓ The name of your favorite burger.
    - ✓ The price of the burger.
    - ✓ A statement for the burger chef indicating the order details.

Example:

```
const burgerName = "Cheeseburger";

const burgerPrice = 8.99;

console.log(` New burger order: ${burgerName}. Price: $${burgerPrice}` );
```

Part 2

1. Add variables for:
    - ✓ The number of burgers ordered.
    - ✓ Whether the burger is a "combo meal" (if true, increase the price by 50%).
2. Calculate the total price.
3. Modify the log statement to display:
    - ✓ The quantity of burgers ordered.
    - ✓ Whether it’s a combo meal.
    - ✓ The total price.

Example:

```
const quantity = 3;

const isCombo = true;

let totalPrice = burgerPrice * quantity;

if (isCombo) {

   totalPrice *= 1.5;

}

console.log(` Order: ${quantity} ${isCombo ? "Combo" : "Regular"} ${burgerName}. Total Price: $${totalPrice}` );
```

Final Project: Pizza Order System

Part 1

1. Create a folder called "pizza-exercise".
2. Inside, create "index.html" and "pizza.js".
3. Link the JavaScript file in the HTML file.
4. Log a statement: console.log("I love pizza");.
5. Create variables for:
    - ✓ Pizza name.
    - ✓ Price.
    - ✓ Log a statement: New pizza order: <pizza name>. Price: <price>.

Part 2

1. Add variables for:
    - ✓ Number of pizzas.
    - ✓ Whether it’s family size (if true, double the price).
2. Calculate totalPrice.
3. Modify the log statement to include:
    - ✓ The quantity.
    - ✓ Whether it’s family size.

- ✓ **The total cost.**