Introduction to JavaScript

Anthoniraj Amalanathan

What is JavaScript?

- JavaScript is an interpreted programming language commonly used in web development to add interactivity and dynamic features to websites.
- JavaScript was created by `Brendan Eich` in 1995 while he was working at Netscape Communications Corporation. Originally named "Mocha," it was later renamed "LiveScript," and finally, "JavaScript".
- JavaScript code can be executed in web browsers, server-side using platforms like Node.js, or even in mobile app development frameworks.

Variables:

Variables in JavaScript are used to store data values. They are declared using keywords like `var`, `let`, or `const`.

Example:

```
// Variable declaration and initialization
var x = 5;
let y = 10;
const PI = 3.14;
```

Log messages to the console (Similar to print)

`console.log()` is a method in JavaScript used to log messages to the console. It's commonly used for debugging purposes to output information about the state of a program or to inspect the values of variables at specific points in the code.

```
let name = "Kamal";
let age = 20;
console.log("Name:", name, "Age:", age); //Name: John Age: 30
```

Operators:

JavaScript supports various operators including arithmetic, assignment, comparison, logical, and uninary. These operators are used to perform operations on variables and values.

Example:

```
// Arithmetic operators
let sum = 5 + 3;
let product = 5 * 3;

// Comparison operators
let isEqual = (sum === product);

// Logical operators
let isValid = (sum > 0 && product < 20);
```

JavaScript Functions:

- JavaScript functions are blocks of reusable code designed to perform a particular task.
- Functions can accept input parameters (arguments), perform operations, and return a value.
- Functions in JavaScript can be declared using the `function` keyword or as arrow functions introduced in ES6.

Example:

```javascript
// Function declaration
function greet(name) {
    return "Hello, " + name + "!";
}

// Function call
let greeting = greet("Anthoniraj");
console.log(greeting); // Outputs: "Hello, Anthoniraj!"
```

Running JavaScript using Node.js:

- Node.js is a runtime environment that allows executing JavaScript code server-side. It provides various built-in modules and APIs, making it possible to build scalable and high-performance applications outside the browser environment.
- Visit the official Node.js website at https://nodejs.org, and download the installer for your operating system (Windows, macOS, or Linux).

Internal and External JavaScript:

- **Internal JavaScript:** JavaScript code written directly within the HTML document using `<script>` tags.
- **External JavaScript:** JavaScript code stored in separate files with a `.js` extension and linked to HTML documents using `<script src="filename.js"></script>` tags.

Example (`external.js`):

```javascript
// External JavaScript file
function greet() {
    alert('Hello, world!');
}
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>External JavaScript Example</title>
    <script src="external.js"></script>
</head>
<body>
    <button onclick="greet()">Click me</button>
</body>
</html>
```

var vs let:

- `var`: Variables declared with `var` have function-level scope. They can be re-declared and updated throughout the function or global scope.

Example:

```
function exampleVar() {
    var message = "Hello";
    if (true) {
        var message = "Goodbye";
    }
    console.log(message); // Outputs: "Goodbye"
}
```

- **`let`**: Variables declared with `let` have block-level scope. They are limited to the block (statements enclosed in curly braces) in which they are defined, and they cannot be re-declared in the same scope.

Example:

```
function exampleLet() {
    let message = "Hello";
    if (true) {
        let message = "Goodbye";
    }
    console.log(message); // Outputs: "Hello"
}
```

document and window:

- `document`: The `document` object represents the HTML document loaded in the browser window. It provides methods and properties to interact with the content of the document, such as accessing elements and modifying their attributes or content.

Example:

```
// Accessing an HTML element by its ID
let element = document.getElementById('myElement');

// Changing the text content of the element
element.textContent = "New content";
```

- `window`: The `window` object represents the browser window or a frame. It provides methods and properties for controlling the browser window, such as navigating to a new URL, resizing the window, or displaying alerts.

Example:

```
// Displaying an alert
window.alert("Hello, world!");
```

Differences of using JS inside `<head>`, `<body>`, outside `<body>` tags:

- **Inside `<head>`**: JavaScript placed inside the `<head>` section of an HTML document gets executed before the document's content is rendered. It is often used for script initialization or loading external scripts.

Example:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>JavaScript in Head Example</title>
    <script>
        // JavaScript code in the head section
        console.log("This script runs before the body content is rendered.");
    </script>
</head>
<body>
    <!-- Body content -->
</body>
</html>
```

- **Inside `<body>`**: JavaScript placed inside the `<body>` section gets executed as the document is parsed, allowing it to interact with HTML elements defined above it. It's commonly used for event handling or manipulating DOM elements.

Example:

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <title>JavaScript in Body Example</title>
</head>
<body>
    <!-- Body content -->
    <script>
        // JavaScript code in the body section
        console.log("This script runs after the body content is rendered.");
    </script>
</body>
</html>
```