ual: creative computing institute

Introduction to JavaScript

BY - RICHA SHARMA
MSC IN COMPUTER SCIENCE

Agenda:

Introduction to JavaScript

JavaScript Basics : Variables and Conventions

JavaScript and Accessibility

Integrating JavaScript with HTML and CSS

The Document Object Model (DOM)

Exercises and Hands-On Lab

What is JavaScript?

Definition: A high-level, interpreted scripting language.

History: Created by Brendan Eich in 1995.

Importance: Essential for web development to create interactive and dynamic web pages.

JavaScript vs. Other Languages

HTML: Structures web content.

CSS: Styles web content.

JavaScript: Adds interactivity to web content.

Front-end scripting language used alongside HTML and CSS.

JavaScript : Case sensitivity

JavaScript pays attention to whether letters are uppercase or lowercase in variable and function names.

For instance, "myvar" and "Myvar" are treated as different names.

JavaScript Variables

What is a Variable?

A variable is like a container that holds information. Think of it as a labeled box where you can store a value, such as a number or a piece of text. You can then use the label to access or change the value inside the box.

Rules for Naming Variables:

- A variable name must start with a letter (A-Z, a-z), an underscore (_), or a dollar sign ($).
- After the first character, variable names can also include numbers (0-9).

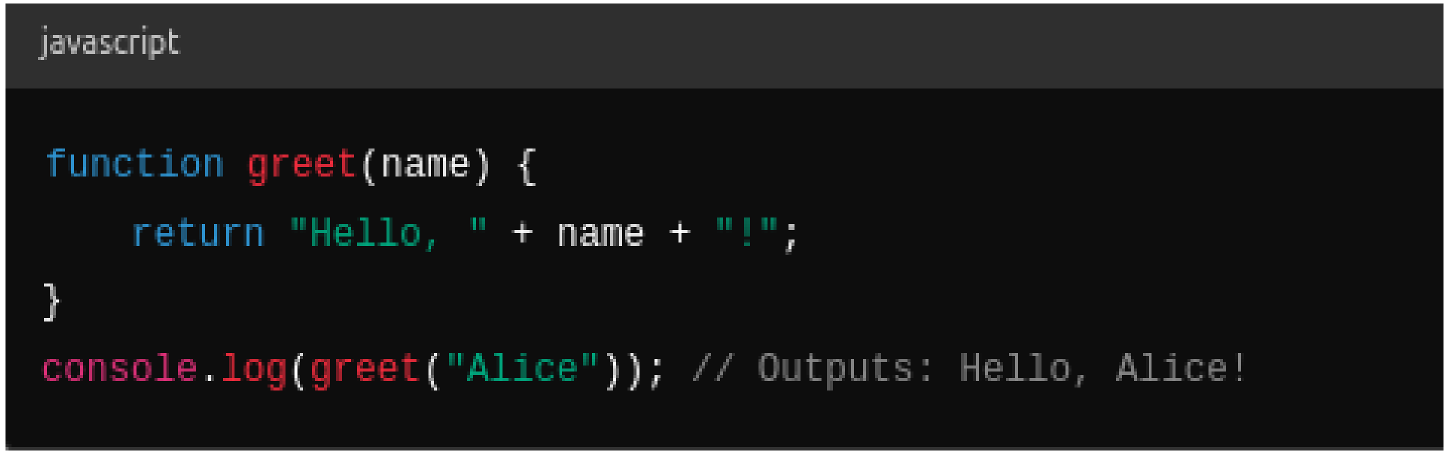
How to Declare Variables:

In JavaScript, you can declare a variable using one of three keywords: ***var, let, and const.***

var	let	const
Function-scoped, meaning it is available throughout the function it is declared in.	Block-scoped, meaning it is only available within the block (e.g., inside {}) where it is declared.	Block-scoped, similar to let. It is constant which means once assigned, its value cannot be changed.
var name = "Alice";	let age = 25;	const birthYear = 1998;

What is a Function?

A function is a reusable block of code designed to perform a particular task. You can think of it as a machine that takes some input, does something with it, and then produces an output.


javascript

```javascript
function greet(name) {
    return "Hello, " + name + "!";
}
console.log(greet("Alice")); // Outputs: Hello, Alice!
```

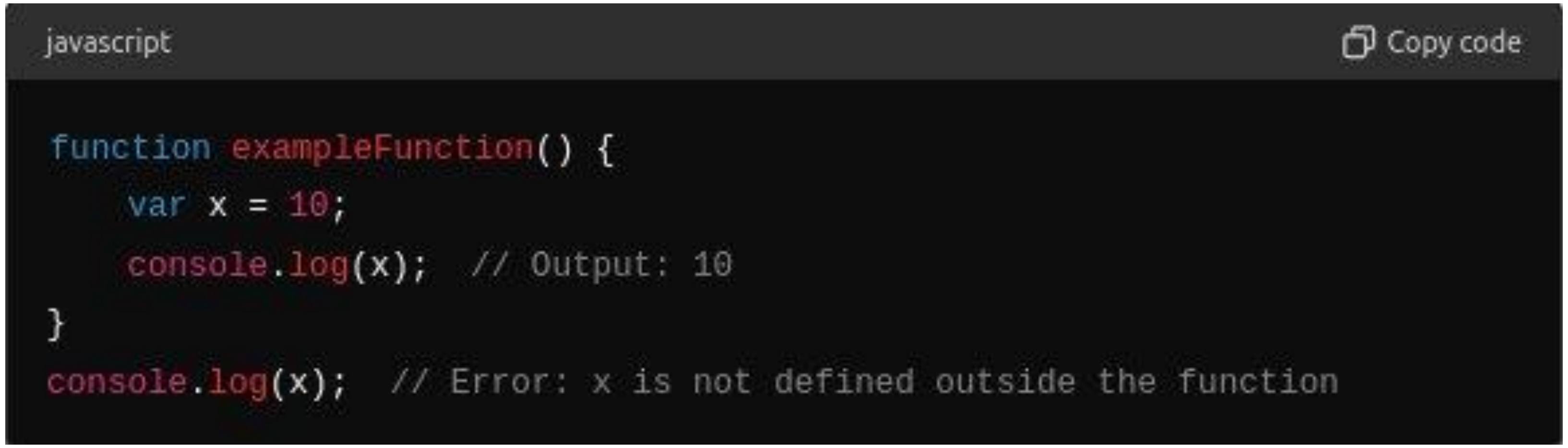
Scope in JavaScript:

Scope determines where variables and functions are accessible in your code.

Global Scope	Function Scope	Block Scope
Variables declared outside any function or block are in the global scope and can be accessed from anywhere in the code.	Variables declared with var inside a function are only accessible within that function.	Variables declared with let or const inside a block (e.g., inside {}) are only accessible within that block.

Function Scope

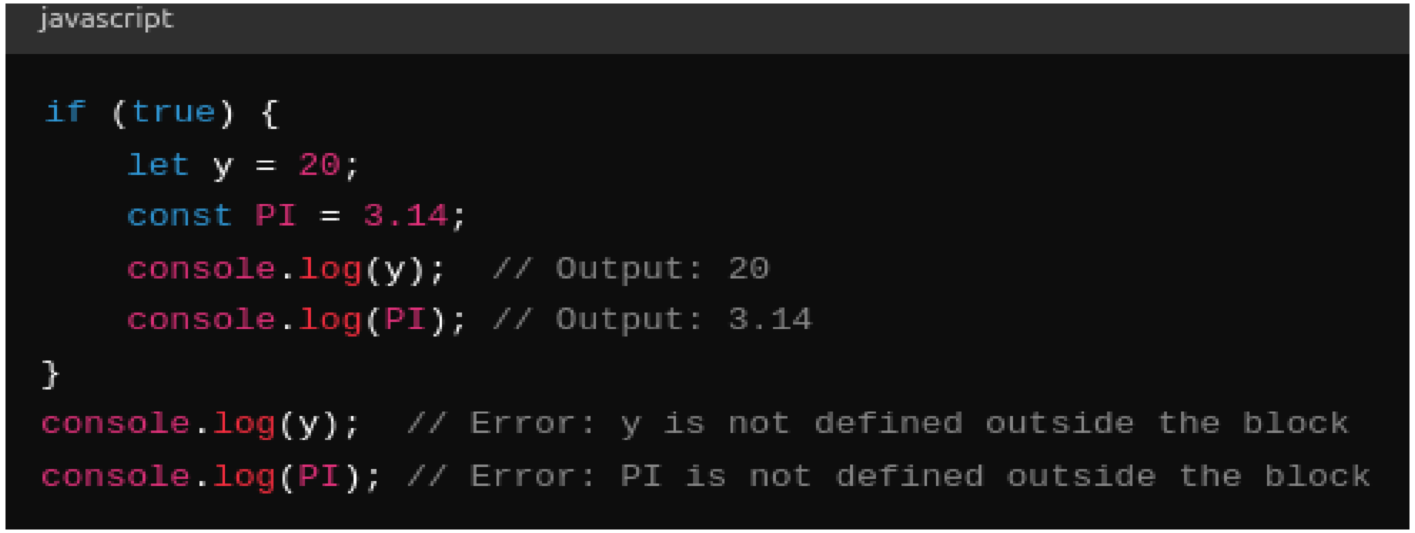
- **var**: Variables declared with var are function-scoped.
  - They are accessible within the function in which they are declared.



```javascript
function exampleFunction() {
    var x = 10;
    console.log(x);  // Output: 10
}
console.log(x);  // Error: x is not defined outside the function
```

Block Scope

- **let and const:** Variables declared with let and const are block-scoped.
  - They are accessible only within the block ({ }) in which they are declared.


javascript

```javascript
if (true) {
    let y = 20;
    const PI = 3.14;
    console.log(y);  // Output: 20
    console.log(PI); // Output: 3.14
}
console.log(y);  // Error: y is not defined outside the block
console.log(PI); // Error: PI is not defined outside the block
```

Literals & Expressions

In JavaScript, literals represent fixed values that are not variables.

1. **Array Literals:** Represent lists of data enclosed in square brackets [ ].
2. **Boolean Literals:** Represent true or false values: true or false.
3. **Integer Literals:** Represent whole numbers: 42, -10, 0.
4. **Floating-Point Literals:** Represent decimal numbers: 3.14, -0.01.
5. **Object Literals:** Represent key-value pairs enclosed in curly braces { }.
6. **String Literals:** Represent sequences of characters enclosed in single ' ' or double " " quotes.

Operators in JavaScript

JavaScript includes various types of operators for performing operations on variables and values.

1. Arithmetic Operators

   Addition (+), Subtraction (-), Multiplication (*), Division (/)

2. Comparison Operators

   Equal to (==), Not equal to (!=), Strict equal to (===), Greater than (>), Less than (<)

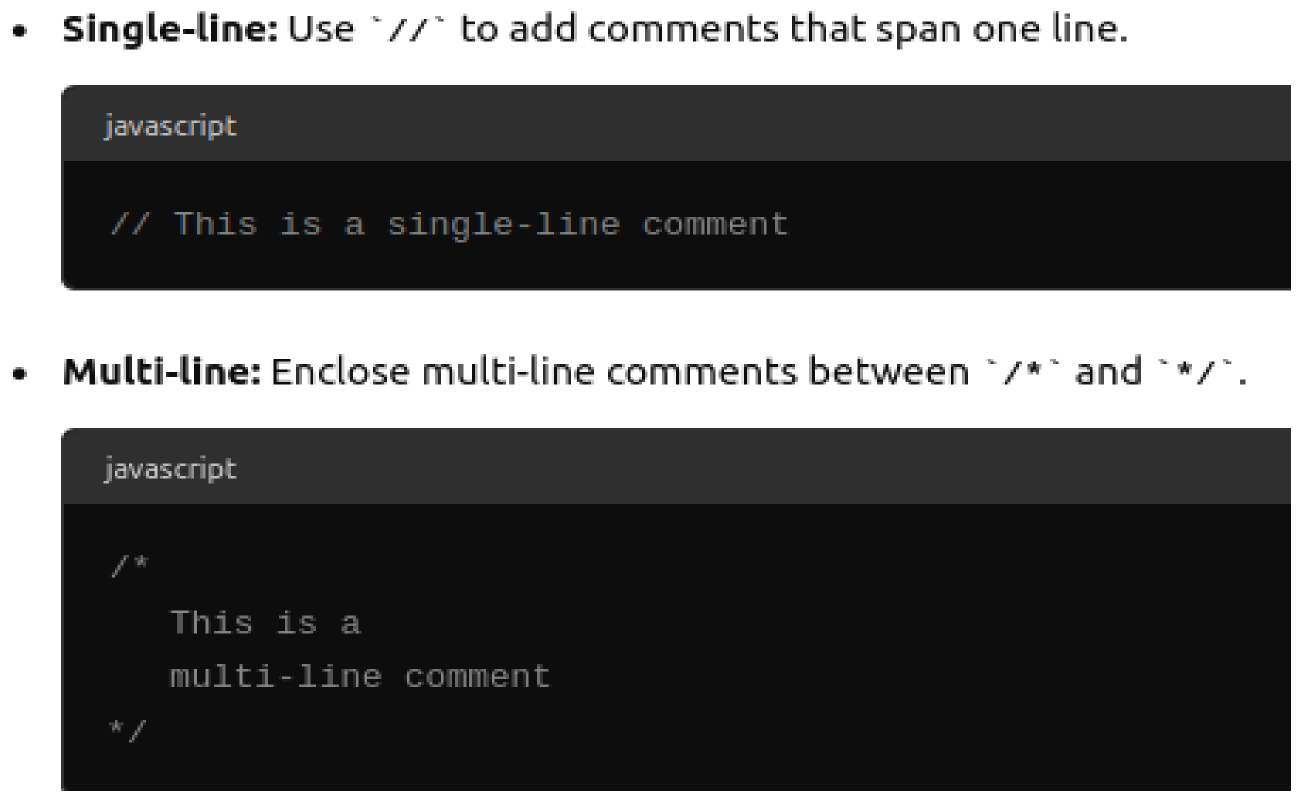
3. Logical Operators

   Logical AND (&&), Logical OR (||), Logical NOT (!)

JavaScript Syntax

JavaScript syntax defines the rules for writing code in JavaScript. Understanding these rules is essential for creating functional and readable programs.

1. Comments



- **Single-line:** Use `//` to add comments that span one line.

javascript

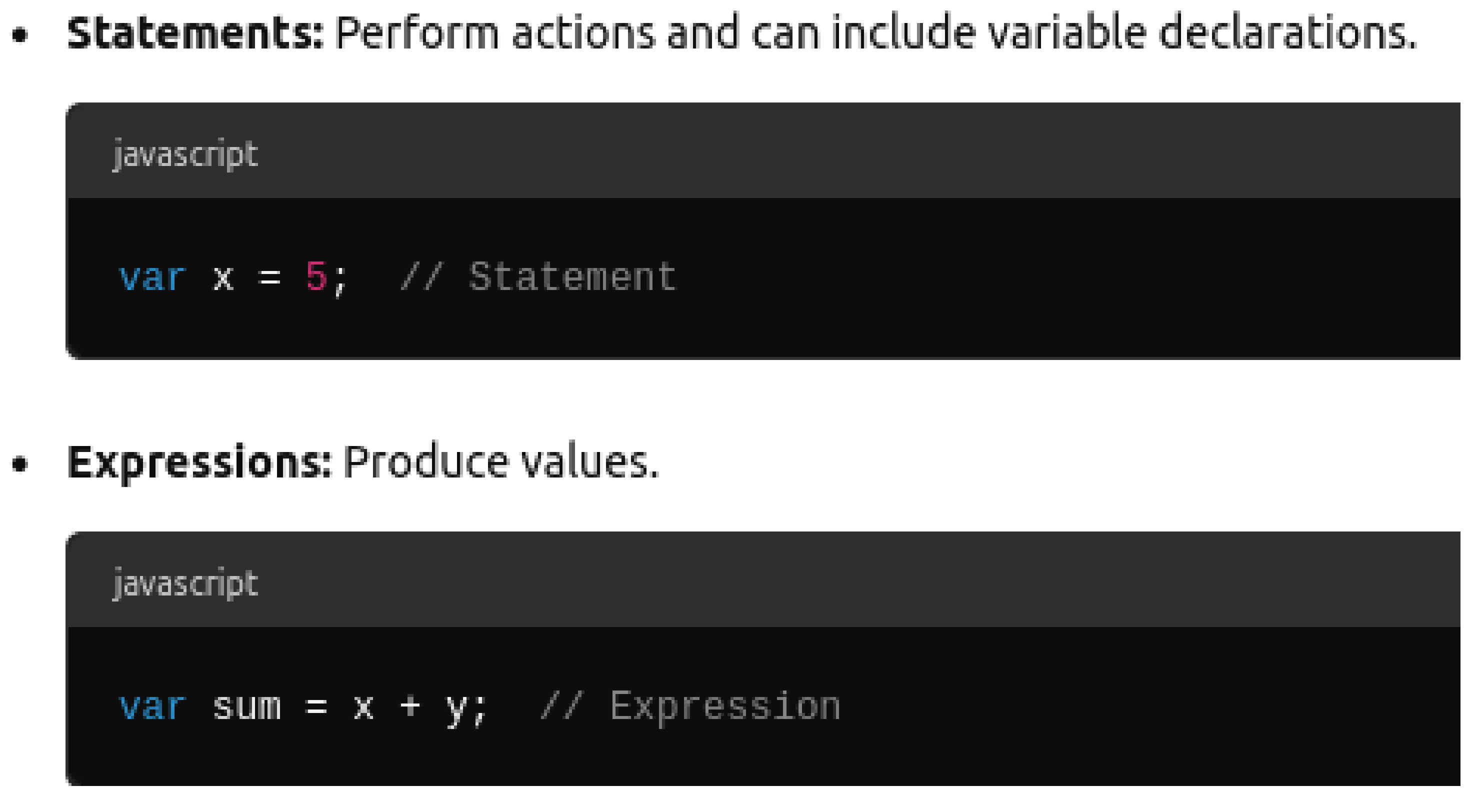
```javascript
// This is a single-line comment
```

- **Multi-line:** Enclose multi-line comments between `/*` and `*/`.

javascript

```javascript
/*
   This is a
   multi-line comment
*/
```

2. Statements and Expressions



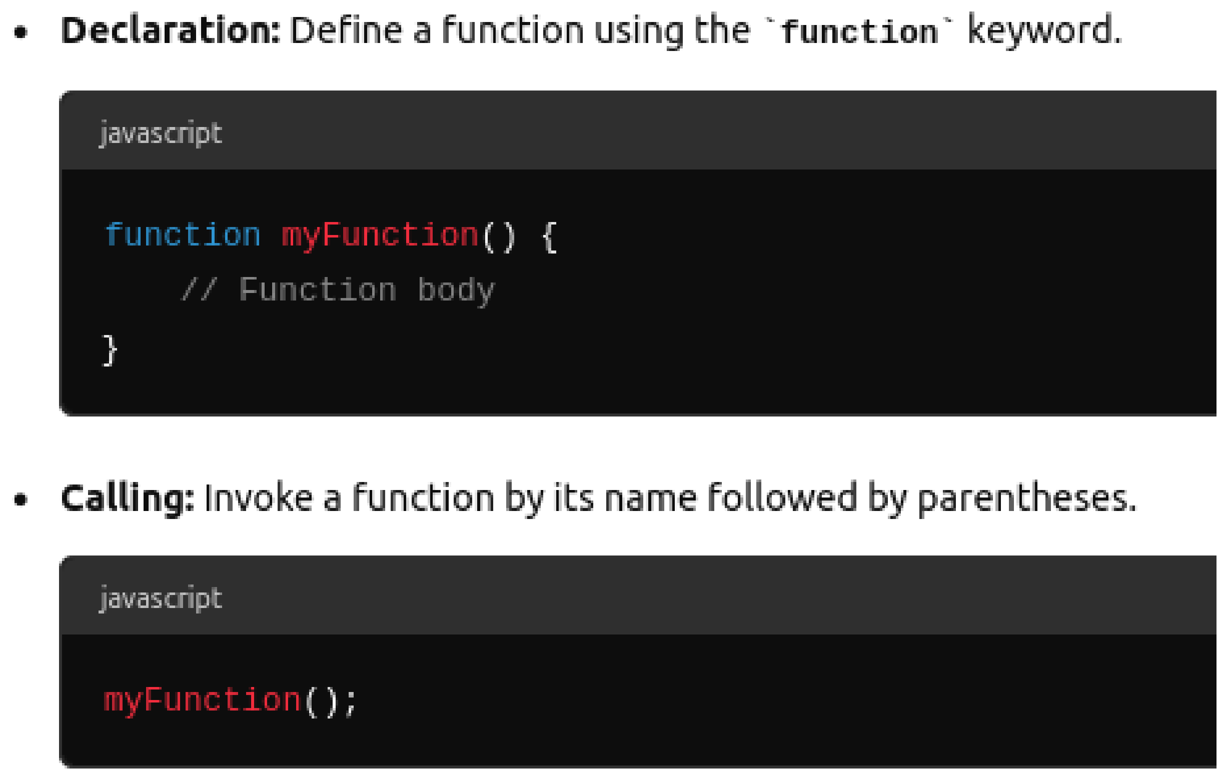
- **Statements:** Perform actions and can include variable declarations.

javascript
```javascript
var x = 5;  // Statement
```

- **Expressions:** Produce values.

javascript
```javascript
var sum = x + y;  // Expression
```

3. Functions



- **Declaration:** Define a function using the `function` keyword.

javascript

```javascript
function myFunction() {
    // Function body
}
```

- **Calling:** Invoke a function by its name followed by parentheses.

javascript

```javascript
myFunction();
```

JavaScript and Accessibility

JavaScript can greatly enhance the accessibility of web applications, ensuring they are usable by everyone, including individuals with disabilities.

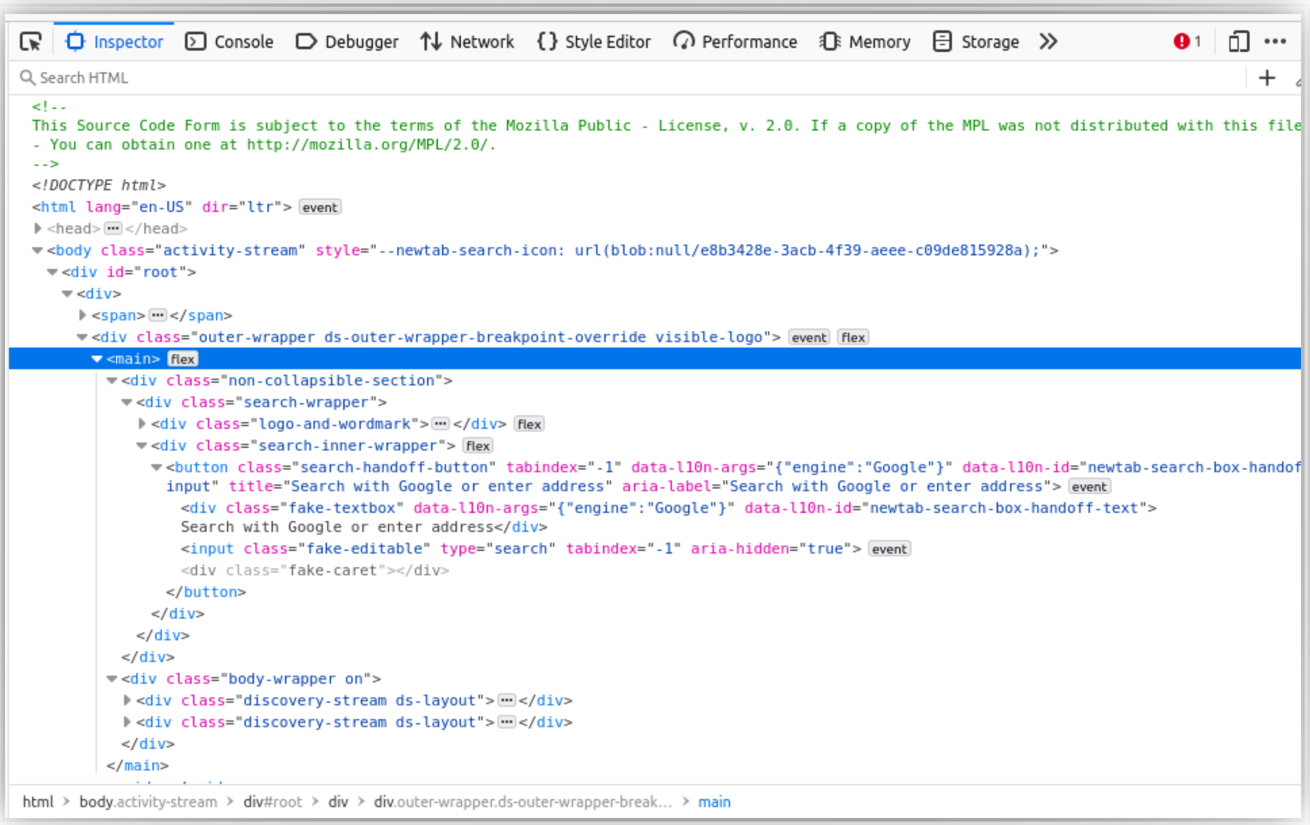
1. **Keyboard Navigation:** Ensure all interactive elements, like buttons and links, are accessible via the keyboard, not just the mouse. This is crucial for users who rely on keyboard navigation.
2. **Form Validation Messages:** Provide clear and accessible error messages when form validation fails. Ensure that these messages are easily perceivable by assistive technologies.

Integrating JavaScript with HTML and CSS

Linking JavaScript to HTML	Inline and External Styles
Use the <script> tag.	Use JavaScript to modify CSS.
Link external JavaScript files.	Change styles dynamically.

The Document Object Model

The Document Object Model (DOM) represents the structure of a web page as a hierarchical tree of objects. JavaScript can manipulate this structure to dynamically change the content and style of web pages.



```
<!--
This Source Code Form is subject to the terms of the Mozilla Public - License, v. 2.0. If a copy of the MPL was not distributed with this file
- You can obtain one at http://mozilla.org/MPL/2.0/.
-->
<!DOCTYPE html>
<html lang="en-US" dir="ltr"> event
 <head>…</head>
 <body class="activity-stream" style="--newtab-search-icon: url(blob:null/e8b3428e-3acb-4f39-aeee-c09de815928a);">
  <div id="root">
   <div>
    <span>…</span>
    <div class="outer-wrapper ds-outer-wrapper-breakpoint-override visible-logo"> event flex
     <main> flex
      <div class="non-collapsible-section">
       <div class="search-wrapper">
        <div class="logo-and-wordmark">…</div> flex
        <div class="search-inner-wrapper"> flex
         <button class="search-handoff-button" tabindex="-1" data-l10n-args="{"engine":"Google"}" data-l10n-id="newtab-search-box-handof
         input" title="Search with Google or enter address" aria-label="Search with Google or enter address"> event
          <div class="fake-textbox" data-l10n-args="{"engine":"Google"}" data-l10n-id="newtab-search-box-handoff-text">
          Search with Google or enter address</div>
          <input class="fake-editable" type="search" tabindex="-1" aria-hidden="true"> event
          <div class="fake-caret"></div>
         </button>
        </div>
       </div>
      </div>
      <div class="body-wrapper on">
       <div class="discovery-stream ds-layout">…</div>
       <div class="discovery-stream ds-layout">…</div>
      </div>
     </main>
```

html > body.activity-stream > div#root > div > div.outer-wrapper.ds-outer-wrapper-break… > main

Selecting DOM Elements

JavaScript provides several methods to select and access HTML elements for manipulation.

1. *getElementById:* Selects an element by its ID.
2. *getElementsByClassName:* Selects elements by their class name.
3. *getElementsByTagName:* Selects elements by their tag name.
4. *querySelector:* Selects the first element that matches a CSS selector.
5. *querySelectorAll:* Selects all elements that match a CSS selector.

Thank You!

Email : r.sharma0520231@arts.ac.uk
Github : https://github.com/14Richa