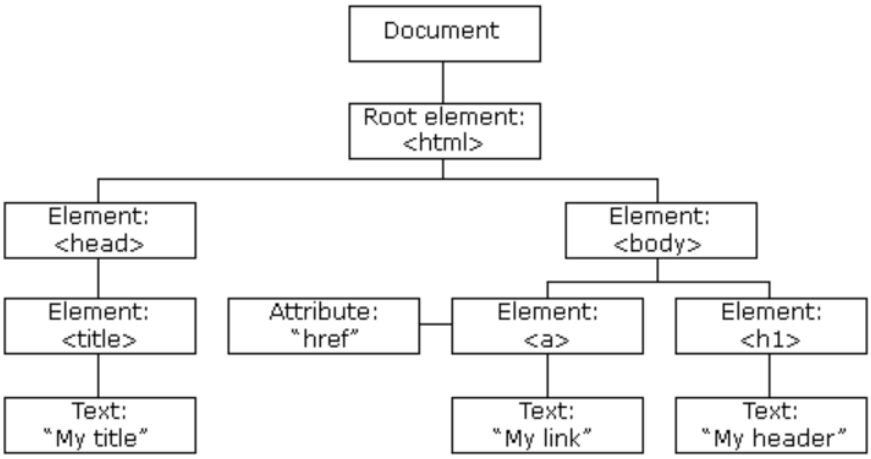
Introduction to JavaScript Interactivity

JavaScript is a powerful language that allows you to create interactive web pages. Interactivity can enhance the user experience by responding to user actions, updating content dynamically, and more. In this section, we'll explore how JavaScript can be used to make web pages interactive.

DOM (Document Object Model)

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page. The DOM represents the structure of an HTML document as a tree of objects. Each element in the HTML is an object in the DOM.



JavaScript can be used to manipulate these objects to change the content, structure, and style of a webpage.

Accessing and Manipulating DOM Elements:

1. **getElementById:** Selects an element by its ID.

```
<div id="myElement">Hello, world!</div>
```

```
let element = document.getElementById("myElement");
console.log(element.textContent); // Output: Hello, world!
```

2. **getElementsByClassName:** Selects all elements with a specified class name. Returns a live HTMLCollection

```
<div class="myClass">First Element</div>
<div class="myClass">Second Element</div>

let elements = document.getElementsByClassName("myClass");
console.log(elements[0].textContent); // Output: First Element
console.log(elements[1].textContent); // Output: Second Element
```

3. **getElementsByTagName:** Selects all elements with a specified tag name. Returns a live HTMLCollection.

```
<p>First paragraph</p>
<p>Second paragraph</p>

let elements = document.getElementsByTagName("p");
console.log(elements[0].textContent); // Output: First paragraph
console.log(elements[1].textContent); // Output: Second paragraph
```

4. **querySelector:** Selects the first element that matches a specified CSS selector.

```
<div class="myClass">First Element</div>
<div class="myClass">Second Element</div>

let element = document.querySelector(".myClass");
console.log(element.textContent); // Output: First Element
```

5. **querySelectorAll:** Selects all elements that match a specified CSS selector. Returns a static NodeList.

```
<div class="myClass">First Element</div>
<div class="myClass">Second Element</div>

let elements = document.querySelectorAll(".myClass");
elements.forEach((element) => {
  console.log(element.textContent);
```

6. **getElementsByName:** Selects all elements with a specified name attribute. Returns a live NodeList.

```
<input type="text" name="myName">
<input type="text" name="myName">

let elements = document.getElementsByName("myName");
console.log(elements.length);
```

Changing Content and Attributes:

```
element.textContent = "New content";
element.innerHTML = "<strong>New content</strong>";
element.setAttribute("src", "newImage.jpg");
```

- document.write() can be used to write directly to the HTML output stream:

  ```
  <p>Hello </p>
  <script> document.write('to') </script>
  <p> World </p>
  ```

DOM Events:

A JavaScript can be executed when an event occurs, like when a user clicks on an HTML element.

To execute code when a user clicks on an element, add JavaScript code to an HTML event attribute:

```
const myDiv = document.getElementById('myDiv').innerText
function displayName() {
   document.getElementById('myDiv').innerText = myDiv + ' Menna'
}
```

DOM Events Listener:

When using addEventListener() method, the JavaScript is separated from the HTML markup, for better readability and allows you to add event listeners even when you do not control the HTML

```
element.addEventListener("click", myFunction);
function myFunction() {
  alert ("Hello World!");
}
```

Event Propagation:

Event propagation is a way of defining the element order when an event occurs. If you have a <p> element inside a element, and the user clicks on the <p> element, which element's "click" event should be handled first?

In bubbling the inner most element's event is handled first and then the outer
In capturing the outer most element's event is handled first and then the inner

```
addEventListener(event, function, useCapture);
```

Remove Event Listener:

```
<div id="myDIV">
 <p>This div element has an onmousemove event handler that displays a random number every time you move your mouse inside this orange field.</p>
 <p>Click the button to remove the div's event handler.</p>
 <button onclick="removeHandler()" id="myBtn">Remove</button>
</div>
<p id="demo"></p>

document.getElementById("myDIV").addEventListener("mousemove ,myFunction);

function myFunction() {
 document.getElementById("demo").innerHTML = Math.random();
}

function removeHandler() {
 document.getElementById("myDIV").removeEventListener("mousemove", myFunction);
}
```

Adding and Removing Elements:

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
const parent = document.getElementById("div1");
const child = document.getElementById("p1");
parent.removeChild(child);
let newElement = document.createElement("div");
document.body.appendChild(newElement);
</script>
```

BOM (Browser Object Model)

The BOM allows JavaScript to interact with the browser.
It includes objects like `window`, `document`, `navigator`, `screen`, `location`, and `history`.

WindowObject:

The window object is supported by all browsers. It represents the browser's window.
Global variables,functions , DOM are properties of window object

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

```
window.confirm("Are you sure?");
```

```
window.prompt("Enter your name:");
```

- **Timing Events:** The window object allows execution of code at specified time intervals
  - ✓ setTimeout(function, milliseconds)
    Executes a function, after waiting a specified number of milliseconds.

```
<button onclick="myVar = setTimeout(myFunction, 3000)">Try it</button>

<button onclick="clearTimeout(myVar)">Stop it</button>
```

  - ✓ setInterval(function, milliseconds)
    Same as setTimeout(), but repeats the execution of the function continuously.

```
<p id="demo"></p>
<button onclick="clearInterval(myVar)">Stop time</button>
<script>
let myVar = setInterval(myTimer ,1000);
function myTimer() {
 const d = new Date();
 document.getElementById("demo").innerHTML =
d.toLocaleTimeString();
}
```

Document Object:

```
document.write("Hello, world!");
```

```
document.title = "New Title";
```

Location Object:

The window.location object can be used to get the current page address (URL) and to redirect the browser to a new page.

- window.location.href returns the href (URL) of the current page

  ```
  document.getElementById("demo").innerHTML =
  "Page location is " + window.location.href;
  ```

- window.location.pathname returns the path and filename of the current page

  ```
  document.getElementById("demo").innerHTML =
  "Page path is " + window.location.pathname;
  ```

- window.location.assign() loads a new document

  ```
  window.location.assign("https://www.google.com")
  ```

History Object:

```
history.back();
history.forward();
history.go(-1);
```

127.0.0.1:5500 says

Your age is: 8

OK

MY FORM

First Name

Menna

Last Name

Osama

Birthdate:

06/07/2016

Display

Hello, My name is Menna Osama and my age is 2