Module 3

- DOM AND EVENT HANDLING
  - Document Object Model in JavaScript
  - Handling Strings and working with Window object
  - Handling events - Mouse events
  - Keyboard events
  - form events and
  - window/document events
  - using the addEventListener() method.

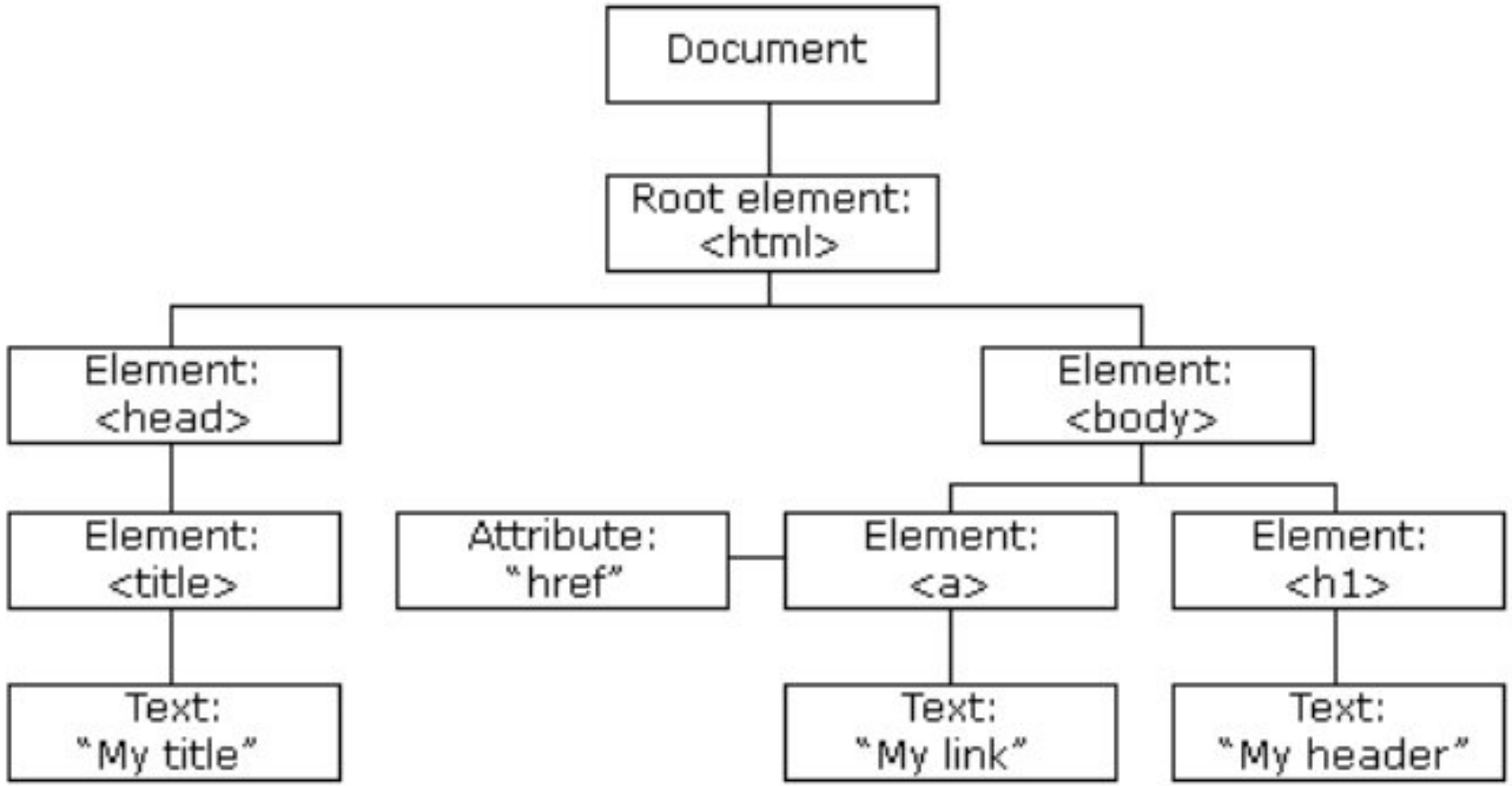
Document Object Model

- The **Document Object Model** (**DOM**) is a cross-platform and language-independent interface that treats an HTML or XML document as a tree structure wherein each node is an object representing a part of the document.
- The DOM represents a document with a logical tree. Each branch of the tree ends in a node, and each node contains objects.
- DOM methods allow programmatic access to the tree; with them one can change the structure, style or content of a document.
- The DOM defines a standard for accessing documents:
- *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*

- In HTML DOM (Document Object Model), every element is a node:[4]
- A document is a document node.
- All HTML elements are element nodes.
- All HTML attributes are attribute nodes.
- Text inserted into HTML elements are text nodes.
- Comments are comment nodes.

- With the object model, JavaScript gets all the power it needs to create dynamic HTML:
- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page
-

The HTML DOM Tree of Objects



- The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:
- The HTML elements as **objects**
- The **properties** of all HTML elements
- The **methods** to access all HTML elements
- The **events** for all HTML elements
- In other words: **The HTML DOM is a standard for how to get, change, add, or delete HTML elements.**
- HTML DOM methods are **actions** you can perform (on HTML Elements).
- HTML DOM properties are **values** (of HTML Elements) that you can set or change.

Example:

```
<html>
<body>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

- In the example above, getElementById is a method, while innerHTML is a property.

- The getElementById Method
- The most common way to access an HTML element is to use the id of the element.
- In the example above the getElementById method used id="demo" to find the element.

- The innerHTML Property
- The easiest way to get the content of an element is by using the innerHTML property.
- The innerHTML property is useful for getting or replacing the content of HTML elements.
- The innerHTML property can be used to get or change any HTML element, including <html> and <body>.

String handling in JavaScript

- **JavaScript strings** are the sequence of characters.
- They are treated as **Primitive data types**.
- In JavaScript, strings are automatically converted to string objects when using **string methods** on them.
- This process is called **auto-boxing.**
- The following are methods that we can call on strings.

- slice() extracts a part of the string based on the given stating-index and ending-index and returns a new string.
- substr() This method returns the specified number of characters from the specified index from the given string. It extracts a part of the original string.
- replace() replaces a part of the given string with another string or a regular expression. The original string will remain unchanged.
- replaceAll() returns a new string after replacing all the matches of a string with a specified string or a regular expression. The original string is left unchanged after this operation.
- charAt() returns the character at the specified index.
- toUpperCase() converts all the characters present in the String to upper case and returns a new String with all characters in upper case. This method accepts single parameter **stringVariable** string that you want to convert in upper case.

- toLowerCase() converts all the characters present in the so lowercase and returns a new string with all the characters in lowercase.
- trim() is used to remove either white spaces from the given string. This method returns a new string with removed white spaces. This method is called on a String object. This method doesn't accept any parameter.
- split() splits the string into an array of sub-strings. This method returns an array. This method accepts a single parameter **character** on which you want to split the string.
- at() The at() method returns the character at a specified index (position) in a string. The at() method is supported in all modern browsers since March 2022
  - The length property returns the length of a string:
  - let text = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    let length = text.length;

- **Window Object**
- In JavaScript, the Window object represents the browser window that contains a DOM document.
- The Window object offers various properties and methods that enable interaction with the browser environment, including manipulating the document, handling events, managing timers, displaying dialog boxes, and more.
- Some of the methods of window object are:
- Window.open(), close(), alert(), prompt(), confirm(), blur(), focus(), print() etc.
- Some of the properties of window object are:
- Window.document, console, location, history, navigator etc.

```
<!DOCTYPE html>
<html>
<body>
<h1>The Window Object</h1>
<h2>The addEventListener() Method</h2>
<p>Click anywhere in the window to display "Hello World!".</p>
<p id="demo"></p>
<script>
window.addEventListener("click", myFunction);
function myFunction() {
  document.getElementById("demo").innerHTML = "Hello World";
}
</script>
</body>
</html>
```

Adding Events Handlers

Method	Description
document.getElementById(*id*).onclick = function() {*code*}	Adding event handler code to an onclick event

Changing HTML Elements

Property	Description
element.innerHTML = *new html content*	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.style.*property = new style*	Change the style of an HTML element
Method	**Description**
element.setAttribute*(attribute, value)*	Change the attribute value of an HTML element

The addEventListener() method

- Add an event listener that fires when a user clicks a button:
- The addEventListener() method attaches an event handler to an element without overwriting existing event handlers.
- You can add many event handlers to one element.
- You can add many event handlers of the same type to one element, i.e two "click" events.
- You can add event listeners to any DOM object not only HTML elements. i.e the window object.
- The addEventListener() method makes it easier to control how the event reacts to bubbling.
- When using the 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 markup.
- You can easily remove an event listener by using the removeEventListener() method.

Ex: document.getElementById("myBtn").addEventListener("click", displayDate);

```
<html>
<body>
<h2>JavaScript addEventListener()</h2>
<p>This example uses the addEventListener() method to attach a click event to a button.</p>
<button id="myBtn">Try it</button>
<script>
document.getElementById("myBtn").addEventListener("click", function() {
  alert("Hello World!");
});
</script>
</body>
</html>
```

This example uses the addEventListener() method to execute a function when a user clicks on a button.

```
<html>
<body>
<h2>JavaScript addEventListener()</h2>
<p>This example uses the addEventListener() method to execute a function when a user clicks on a button.</p>
<button id="myBtn">Try it</button>
<script>
document.getElementById("myBtn").addEventListener("click", myFunction);
function myFunction() {
  alert ("Hello World!");
}
</script>
</body>
</html>
```