JavaScript

Day 4

JavaScript HTML DOM EventListener

📌 addEventListener() Method

✅ Example:

Add an event listener that fires when a user clicks a button:

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

📘 What is addEventListener()?

- The addEventListener() method attaches an **event handler** to a specified element.
- It does **not overwrite** existing event handlers.
- You can add **multiple event handlers** to the **same element**.
- You can add **multiple handlers of the same type**, e.g., two "click" events.
- You can attach event listeners to **any DOM object**, not just HTML elements (e.g., the window object).

🔨 Add an Event Handler to an Element

✅ Example: Using an anonymous function

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

✅ Example: Using a named function

```
element.addEventListener("click", myFunction);

function myFunction() {
  alert("Hello World!");
}
```

➕ Add Multiple Event Handlers to the Same Element

You can add multiple events without overwriting existing ones.

✅ Example: Multiple "click" events

```
element.addEventListener("click", myFunction);
element.addEventListener("click", mySecondFunction);
```

✅ Example: Different types of events

```
element.addEventListener("mouseover", myFunction);
element.addEventListener("click", mySecondFunction);
element.addEventListener("mouseout", myThirdFunction);
```

🌐 Add an Event Handler to the window Object

The addEventListener() method can be used with the **window object**, **document**, and other DOM-supported objects.

✅ Example: Resize the window

```
window.addEventListener("resize", function() {
  document.getElementById("demo").innerHTML = sometext;
});
```

📦 Passing Parameters to Event Handlers

To pass parameters, use an **anonymous function** that calls your function with parameters.

✅ Example:

```
element.addEventListener("click", function() {
  myFunction(p1, p2);
});
```

❌ removeEventListener() Method

The removeEventListener() method removes event handlers attached with addEventListener().

✅ Example:

```
element.removeEventListener("mouseover", myFunction);
```

🌐 JavaScript HTML DOM Navigation

📚 DOM Nodes

According to the **HTML DOM standard**, everything in an HTML document is a **node**:

- The entire document is a **document node**
- Every HTML element is an **element node**
- The text inside HTML elements is a **text node**

- Every HTML attribute is an **attribute node** (deprecated)
- All comments are **comment nodes**

🌳 DOM HTML Tree

With the HTML DOM:

- All nodes in the node tree can be accessed via JavaScript.
- You can **create**, **modify**, or **delete** nodes dynamically.

👨‍👩‍👦 Node Relationships

Nodes in the DOM tree have hierarchical relationships. Key terms:

- **Root node**: Topmost node (usually <html>)
- **Parent**: Node that contains another node
- **Child**: Node contained by another
- **Siblings**: Nodes that share the same parent

✅ Example Structure:

```
<html>
  <head>
    <title>DOM Tutorial</title>
  </head>
  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>
</html>
```

Node Relationship Breakdown:

- <html> is the **root node**
- <html> has no parents
- <html> is the parent of <head> and <body>
- <head> is the **first child** of <html>
- <body> is the **last child** of <html>

Further:

- <head> has one child: <title>
- <title> has one child (a text node): "DOM Tutorial"
- <body> has two children: <h1> and <p>
- <h1> has one child: "DOM Lesson one"
- <p> has one child: "Hello world!"
- <h1> and <p> are **siblings**

🧭 Navigating Between Nodes

You can navigate between nodes using these properties:

- parentNode
- childNodes[nodenumber]
- firstChild
- lastChild
- nextSibling
- previousSibling

🔍 Child Nodes and Node Values

A common mistake is expecting an element to contain text directly—it actually contains a **text node**.

✅ **Example:**

<title id="demo">DOM Tutorial</title>

Accessing the value:

let myTitle = document.getElementById("demo").innerHTML;

Equivalent ways:

myTitle = document.getElementById("demo").firstChild.nodeValue;

myTitle = document.getElementById("demo").childNodes[0].nodeValue;

🧪 **Practical Examples**

✅ **Example 1:**

<html>

<body>

<h1 id="id01">My First Page</h1>

<script>

document.getElementById("id02").innerHTML = document.getElementById("id01").innerHTML;

</script>

</body>

</html>

✅ **Example 2:**

<html>

```
<body>

<h1 id="id01">My First Page</h1>
<p id="id02"></p>

<script>
document.getElementById("id02").innerHTML =
document.getElementById("id01").firstChild.nodeValue;
</script>

</body>
</html>
```

✅ **Example 3:**

```
<html>
<body>

<h1 id="id01">My First Page</h1>
<p id="id02">Hello!</p>

<script>
document.getElementById("id02").innerHTML =
document.getElementById("id01").childNodes[0].nodeValue;
</script>

</body>
```

</html>

🧠 innerHTML

In this tutorial, we use innerHTML to retrieve element content.

However, learning firstChild, childNodes, etc., is useful for understanding the **DOM tree structure**.

📂 DOM Root Nodes

Two special properties provide full document access:

- document.body – Returns the document's body
- document.documentElement – Returns the full HTML document

✅ Example 1:

<html>

<body>

<h2>JavaScript HTMLDOM</h2>

Displaying document.body

<script>

document.getElementById("demo").innerHTML = document.body.innerHTML;

</script>

</body>

</html>

✅ Example 2:

<html>

```
<body>

<h2>JavaScript HTMLDOM</h2>
<p>Displaying document.documentElement</p>

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

<script>
document.getElementById("demo").innerHTML =
document.documentElement.innerHTML;
</script>
</body>
</html>
```

🏷️ The nodeName Property

The nodeName property returns the name of a node.

- It is **read-only**
- For element nodes: returns the tag name (e.g., H1)
- For attribute nodes: returns the attribute name
- For text nodes: returns #text
- For the document node: returns #document

✅ Example:

```
<h1 id="id01">My First Page</h1>
<p id="id02"></p>
<script>
```

```
document.getElementById("id02").innerHTML =
document.getElementById("id01").nodeName;
</script>
```