DOM Manipulation

Web Development Essentials - Session 11

Session Overview

Learning Goals for Today:

- Understand the Document Object Model (DOM) and its role in web development.
- Learn how to select and modify HTML elements using JavaScript.
- Introduction to event handling (clicks, keypresses, etc.).

What is the DOM?

- **Definition**: The **Document Object Model (DOM)** is a programming interface for web documents. It represents the structure of an HTML document as a tree of objects that can be manipulated using JavaScript.
- **Why is the DOM Important?**
  - Allows developers to interact with and manipulate the structure, style, and content of a webpage.
  - When the DOM is updated, the changes are reflected on the page instantly.
- **Visual Representation**:
  - HTML elements (like `<div>`, `<p>`, `<button>`) are nodes in the DOM tree.
  - JavaScript can traverse, add, modify, or remove these nodes.

The Structure of the DOM

Key DOM Objects:

- **Document**: Represents the entire webpage.
- **Element**: Represents individual HTML elements (like `<p>`, `<h1>`, `<div>`).
- **Text**: Represents the text inside an element.

DOM Tree Example:

```
<body>
  <h1>Hello, World!</h1>
  <p>This is a paragraph.</p>
</body>
```

Corresponding DOM tree:

- `<body>`
  - `<h1>` → "Hello, World!"
  - `<p>` → "This is a paragraph."

Selecting HTML Elements in JavaScript

How to Access DOM Elements:

- Use **selectors** to find elements in the DOM.

Common Methods:

1. `document.getElementById()`: Selects an element by its ID

```
let title = document.getElementById("main-title");
console.log(title.textContent);
```

1. `document.getElementsByClassName()`: Selects elements by class name.
2. `document.querySelector()`: Selects the first element that matches a CSS selector
3. `document.querySelectorAll()`: Selects all elements that match a CSS selector.

```
let heading = document.querySelector("h1");
```

Modifying HTML Elements

Modifying Text Content:

- Use `textContent` or `innerHTML` to change the content of an element.

```
let heading = document.querySelector("h1");
heading.textContent = "New Title"; // Changes the text of the heading
```

Changing Element Attributes:

- Use `setAttribute()` to modify attributes like `src`, `href`, or `class`.

```
let image = document.querySelector("img");
image.setAttribute("src", "new-image.jpg"); // Changes the image source
```

Adding and Removing Classes:

- Use `classList.add()`, `classList.remove()`, or `classList.toggle()` to manipulate an element's CSS classes.

```
let button = document.querySelector("button");
button.classList.add("active"); // Adds the "active" class to the button
```

Adding and Removing DOM Elements

Creating New Elements:

- Use `document.createElement()` to create a new element.

```
let newParagraph = document.createElement("p");
newParagraph.textContent = "This is a new paragraph!";
document.body.appendChild(newParagraph); // Adds the paragraph to the body
```

Removing Elements:

- Use `remove()` to delete an element from the DOM.

```
let elementToRemove = document.querySelector("#old-element");
elementToRemove.remove();
```

Introduction to Event Handling

What is an Event?

- An event is an action that occurs on a webpage, like a button click, keypress, or page load.

Why Use Events?

- Event handling allows developers to create interactive websites, responding to user actions in real-time.

Common Events:

1. **Click Event**: Fired when an element is clicked.
2. **Mouseover Event**: Fired when the mouse is moved over an element.
3. **Keypress Event**: Fired when a key is pressed.

Event Listeners in JavaScript

Adding an Event Listener:

- Use the `addEventListener()` method to attach a function (called a **callback**) to an element that executes when the event occurs.

```
element.addEventListener("event", function() {
  // Code to run when the event occurs
});
```

Example

```
let button = document.querySelector("button");
button.addEventListener("click", function() {
  alert("Button was clicked!");
});
```

Handling Different Events

Click Event Example:

```
let button = document.querySelector("#myButton");
button.addEventListener("click", function() {
  console.log("Button clicked!");
});
```

Keypress Event Example:

```
document.addEventListener("keypress", function(event) {
  console.log("Key pressed: " + event.key);
});
```

Mouseover Event Example:

```
let image = document.querySelector("img");
image.addEventListener("mouseover", function() {
  image.setAttribute("src", "hover-image.jpg");
});
```

Hands-On Activity

Goal: Create a simple webpage with DOM manipulation and event handling.

- Select an element by ID and change its text.
- Add a click event to a button that changes the color of a heading.

Instructions:

1. Use `getElementById()` to select a heading.
2. Change the heading's text using `textContent`.
3. Add a button that triggers a color change when clicked.

```
<h1 id="title">Original Title</h1>
<button id="changeTitle">Change Title</button>

<script>
  let title = document.getElementById("title");
  let button = document.getElementById("changeTitle");
```

```
  button.addEventListener("click", function() {
    title.textContent = "New Title";
    title.style.color = "blue"; // Changes the title color
  });
</script>
```

Common Mistakes in DOM Manipulation

Selecting Non-existent Elements: Make sure the element you are selecting actually exists in the DOM.

Overwriting Existing Content: When using `innerHTML`, be cautious as it can overwrite existing content.

Forgetting to Attach Event Listeners: Always ensure that event listeners are added after the elements are loaded.

Debugging DOM Manipulation

Using the Console: Use the browser's developer console to inspect the DOM and test JavaScript code.

Check for Errors: Look for common errors like `null` or `undefined` when selecting elements.

```
console.log(document.querySelector("#myElement"));
// Use console.log() to check if elements are correctly selected.
```

Summary

What We Learned Today:

- Introduction to the DOM and its structure.
- How to select, modify, and remove HTML elements using JavaScript.
- Event handling (e.g., clicks, keypresses) to add interactivity to webpages.

Questions?

Q&A Session

- Any questions before we wrap up?

Thank You & See You in the Next Class!