Introduction to The DOM

Software Development Bootcamp

Topic

Document Object Model (DOM), Browser Object Model (BOM)

What Is The DOM

The DOM (**D**ocument **O**bject **M**odel) is a programming interface for HTML documents.

- **Document**: HTML page
- **Object**: HTML elements and comments
- **Model**: Arranging them logically

Key Points

- Access the DOM in JavaScript with the built-in `document` object.
- The browser models the HTML document as objects using a tree data structure.
- Everything in your file becomes a node: elements, text strings, and comments.

What Is The BOM?

The BOM (Browser Object Model) represents additional objects provided by the browser for working with everything except the document.

- The `window` object is the top of the BOM hierarchy
- The `window` object represents the browser window and provides methods to control it.

Window vs. Document

Window

- Represents the whole browser window
- You can use it to control things like opening new tabs, changing the size of the window, or showing alert boxes

Document

- Represents the actual HTML document loaded in the browser
- Lets you access and change HTML elements on your page, like paragraphs, buttons, and images
- You can use it to find elements, or change how elements look and behave

Including JavaScript in HTML

- **Example 1:** Writing JavaScript directly in the HTML file using `<script>` tags
- **Example 2:** Referencing external JavaScript files using `<script src="index.js">`

```
index.html
Example 1
<body>
    <script>
        let message = 'hello world!'
        alert(message)
    </script>
</body>
```

```
index.html
Example 2
<body>
  </script><script src="index.js"></script>
</body>
```

Topic

Accessing And Creating DOM Elements

Finding Elements by TagName

- Returns an **HTMLCollection** of all the specified elements in the document
- Useful when you want to select all elements of a specific type

```
index.html
 <body>
        <h1></h1>
        </script>
        <!-- This script tag is referencing
our index.js file -->
        <script src="index.js"></script>
    </body>
```

```
index.js
let myH1 =
document.getElementsByTagName('h1')
```

Finding Elements by ClassName

- This method returns an HTMLCollection of all elements with the specified class name
- Helpful when you want to select multiple elements that share a common class

```
index.html
  <body>
      <h1 class="my-h1-class"></h1>
      </script>
      <script src="index.js"></script>
    </body>
```

```
index.js
let myH1 =
document.getElementsByClassName('my-h1-class')
```

Finding Elements by Id

- Returns a single element with the specified id. It's useful when you need to select a unique element on the page.

```
index.html
 <body>
   <h1 id="my-h1-id"></h1>
  </script>
  <script src="index.js"></script>
</body>
```

```
index.js
let myH1 = document.getElementById('my-h1-id')
```

Finding Elements by QuerySelector

- This method returns the first element that matches the specified CSS selector
- Can select elements by id, class, or tag name using CSS selector syntax.

```
index.html
<body>
    <h1 id="my-h1-id"></h1>
    <h2 class="my-h2-class"></h2>
    <h3></h3>
    </script><script src="index.js"></script>
</body>
```

```
index.js
let myH1 = document.querySelector('#my-h1-id')
let myH2 =
document.querySelector('.my-h2-class')
let myH3 = document.querySelector('h3')
```

Finding Elements by QuerySelectorAll

- This method returns a NodeList containing all elements that match the specified CSS selector
- It's useful when you need to select multiple elements that match a certain criteria

```
index.html
<body>
    <h1 id="my-h1-id"></h1>
    <h2 class="my-h2-class"></h2>
    <h2 class="my-h2-class"></h2>
    <h2 class="my-h2-class"></h2>
    <h3></h3>
    </script><script src="index.js"></script>
</body>
```

```
index.js
let myH2 =
document.querySelectorAll('.my-h2-class')
```

HTMLCollection VS. NodeList

Key Differences

- Content Type
  - **HTMLCollection**: Contains only element nodes (tags)
  - **NodeList**: Can contain any node type (element nodes, text nodes, comment nodes)
- Method of Creation
  - **HTMLCollection:** Returned by methods like `getElementsByClassName()`, `getElementsByTagName()`
  - **NodeList:** Returned by methods like `querySelectorAll()`
- Array-like Features
  - **HTMLCollection**: Can be accessed by index or name/id, but doesn't have array methods
  - **NodeList**: Can be accessed by index and has some array-like methods (e.g., `forEach()`)

Using an Element

- Find the specific element to change using **`document.getElementById`**
- Use properties like **`textContent`** to modify the element

```
index.html
<body>
   <!-- h1 tag is currently empty -->
   <h1 id="my-h1-id"></h1>
   </script><script
src="index.js"></script>
</body>
```

```
index.js
// Create a variable myH1 that will store
the value of the element
let myH1 =
document.getElementById('my-h1-id')
// Use textContent to change the text
myH1.textContent = 'Hello World'
```

Creating an Element

- To create new elements use `document.createElement()`
- Use `appendChild()` to add the new element to the DOM

```
index.html
<body>
  <div class="my-div">
      <!-- div is empty -->
  </div>
  </script><script src="index.js"></script>
</body>
```

```
index.js
let myDiv =
document.getElementById('my-div')
// Creating a new 'p' tag
let paragraph = document.createElement('p')
// Adding text to the created 'p' tag
paragraph.textContent = "Lorem ipsum dolor
sit amet, consectetur adipiscing elit, sed
do eiusmod tempor incididunt ut labore et
dolore magna aliqua. Ut enim ad minim
veniam, quis nostrud"
// Adding the created 'p' tag to the DOM
myDiv.appendChild(paragraph)
```

Topic

Dev Tools

What Are Dev Tools?

Browser Developer Tools (Dev Tools) are a set of web authoring and debugging tools built into modern web browsers. They allow developers to inspect, debug, and optimize websites and web applications

Key Features of Dev Tools

- Elements Panel
  - Inspect and edit HTML and CSS in real-time
  - View and modify the DOM tree
  - Examine and adjust CSS styles
- Console Panel
  - View JavaScript errors and logs
  - Execute JavaScript code directly in the browser
- Sources Panel
  - Debug JavaScript code
  - Set breakpoints and step through code execution

Key Features of Dev Tools

- Network Panel
  - Monitor network requests and responses
  - Analyze loading performance
  - Inspect HTTP headers and content
- Performance Panel
  - Record and analyze runtime performance
  - Identify bottlenecks and optimization opportunities
- Application Panel
  - Inspect local storage, session storage, and cookies
  - Manage service workers and cache

Exercise

Colorful Boxes