Lesson 6: Manipulating the DOM with JavaScript

The Document Object Model (DOM) is a programming interface that represents the structure of HTML and XML documents as a tree-like structure. Each element in an HTML document, such as headings, paragraphs, images, and buttons, is represented as a node in the DOM tree. JavaScript provides powerful methods and properties to manipulate the DOM, allowing you to dynamically update the content, structure, and styles of web pages. In this section, we will explore how to use JavaScript to interact with the DOM effectively.

Accessing Elements:

JavaScript provides methods to select and access elements in the DOM. These methods allow you to target specific elements or groups of elements based on various criteria:

getElementById: Retrieves an element by its unique ID attribute. This method returns a single element that matches the specified ID.

```
var element = document.getElementById("myElement");
```

getElementsByClassName: Retrieves a collection of elements based on their class name. This method returns an HTMLCollection or a NodeList of elements that have the specified class.

```
var elements = document.getElementsByClassName("myClass");
```

getElementsByTagName: Retrieves a collection of elements based on their tag name. This method returns an HTMLCollection or a NodeList of elements that have the specified tag.

```
var elements = document.getElementsByTagName("div");
```

querySelector: Retrieves the first element that matches a specific CSS selector. This method returns the first element that matches the specified selector.

```
var element = document.querySelector(".myClass");
```

Modifying Elements:

Once you have selected an element or a group of elements, JavaScript allows you to modify their properties, content, and styles:

Changing Content: You can update the text or HTML content of an element using the ***textContent*** or ***innerHTML*** properties, respectively. The ***textContent*** property sets or returns the text content of an element, while the ***innerHTML*** property sets or returns the HTML content.

```
element.textContent = "Hello, world!";
element.innerHTML = "<strong>Welcome</strong> to my website!";
```

Modifying Attributes: JavaScript provides methods to modify element attributes. You can use the ***getAttribute*** method to retrieve the value of an attribute, the ***setAttribute*** method to set the value of an attribute, and the ***removeAttribute*** method to remove an attribute from an element.

```
var src = image.getAttribute("src");
link.setAttribute("href", "https://www.example.com");
button.removeAttribute("disabled");
```

Adjusting Styles: You can change the appearance of an element by modifying its CSS styles using the ***style*** property. The ***style*** property allows you to access and modify individual style properties, such as ***color***, ***backgroundColor***, ***fontSize***, etc. You can assign values to these properties to change the styling of an element.

```
element.style.color = "red";
element.style.backgroundColor = "yellow";
```

Creating and Appending Elements:

JavaScript enables you to create new elements dynamically and add them to the DOM:

Creating Elements: Use the ***createElement*** method to create a new HTML element. You can specify the element type by passing the tag name as an argument to the

method. After creating the element, you can set its attributes and content using various DOM methods.

```
var newElement = document.createElement("div");
newElement.setAttribute("id", "myNewElement");
newElement.textContent = "New Element";
```

Appending Elements: You can add the newly created element to the DOM by appending it to an existing element using the ***appendChild*** or ***insertBefore*** method. The ***appendChild*** method appends the new element as the last child of the parent element, while the ***insertBefore*** method inserts the new element before a specified reference element.

```
var container = document.getElementById("container");
container.appendChild(newElement);
```

Handling Events:

JavaScript allows you to respond to user interactions and create interactive web functionality through event handling:

Event Listeners: Use the ***addEventListener*** method to attach event listeners to elements. Event listeners allow you to define actions to be performed when a specific event occurs, such as a mouse click, key press, or form submission.

```
element.addEventListener("click", function() {
  // Perform actions when the element is clicked
});
```

Common Events: JavaScript provides access to a wide range of events that can be listened to and responded to. These events include mouse events (***click, mouseover, mouseout***), keyboard events (***keydown, keyup***), form events (***submit, change***), and many more. You can attach event listeners to elements based on the desired interaction.

```
element.addEventListener("mouseover", function() {
  // Perform actions when the mouse is over the element
```

```
});
```

By manipulating the DOM with JavaScript, you can dynamically update web page content, modify element attributes and styles, create new elements, and respond to user interactions. This powerful capability allows you to build interactive and engaging web applications that respond to user actions and provide a rich user experience.