Module 3: Introduction to JavaScript

Injecting JavaScript into HTML

Injecting JavaScript into HTML web pages is a common practice for adding interactivity, manipulating the DOM (Document Object Model), and enhancing the functionality of web applications.

There are two main ways of injecting JavaScript into HTML web pages:

- Inline JavaScript
- External JavaScript

Inline JavaScript

You can include JavaScript code directly in an HTML file by using the <script> tag. You can place it in the <head> or just before the end of the <body> section of an HTML document.

```
<html>
    <head>
        <title>Inline JavaScript Example</title>
    </head>
    <body>
        <button onclick="greet()">Click me</button>
        <script>
        // Your JavaScript code here
        function greet() {
            alert("Hello World!");
        }
    </script>
     </body>
</html>
```

It is recommended that you place the <script> tag just before the <body> tag ends so that the entire DOM is loaded and ready for the JavaScript to interact with it.

External JavaScript

For larger scripts and better code organization, you can create an external JavaScript file with a .js extension and link it to your HTML document by using the <script> tag's src attribute.

```
<html>
    <head>
        <title>External JavaScript Example</title>
    </head>
    <body>
        <button onclick="greet()">Click me</button>
        <script src="script.js"></script>
    </body>
</html>
```

The script.js file contains the JavaScript code:

```
function greet() {
    alert('Hello, World!');
}
```

Please note that both index.html and script.js should be present in the same folder location for this code to work. You'll need to make corresponding changes to the file path in the src attribute of the <script> tag if index.html and script.js are in different folders. This method is preferred for maintaining and reusing JavaScript code.

Modifying HTML Using JavaScript

By injecting JavaScript into HTML web pages, you can modify HTML and its styles, which increases the overall interactivity of the web pages and enhances the user experience.

1. **Changing Element Content**

   You can change the content of an HTML element, such as a paragraph (<p>) or a heading (<h1>), by accessing the element using JavaScript and then updating its innerHTML or textContent property. For example:

```
<p id="myParagraph">This is a paragraph.</p>
<script>
    // Change the content of the paragraph
    var paragraph = document.getElementById("myParagraph");
    paragraph.innerHTML = "This is updated content.";
</script>
```

2. **Modifying Element Attributes**

   You can change element attributes like src, href, class, or style with JavaScript. For example, to change the source of an image (<img>) element:

```
<img id="myImage" src="image.jpg" />
<script>
    // Change the source of the image
    var image = document.getElementById("myImage");
    image.src = "new-image.jpg";
</script>
```

3. **Adding and Removing Elements**

   You can dynamically add or remove HTML elements with JavaScript. For instance, you can add new elements to a container element (e.g., <div>) or remove existing ones. For example:

```
<div id="myContainer"></div>
<script>
    // Create a new paragraph element and add it to the container
    var container = document.getElementById("myContainer");
    var newParagraph = document.createElement("p");
    newParagraph.textContent = "This is a new paragraph.";
    container.appendChild(newParagraph);
    // Remove an element from the container
    var elementToRemove =
document.getElementById("elementToRemove");
    container.removeChild(elementToRemove);
</script>
```

4. **Styling Elements**

   JavaScript can be used to change the style of HTML elements dynamically. You can modify CSS properties such as backgroundColor, fontSize, and display. For example:

```
<p id="myStyledParagraph">This is a styled paragraph.</p>
<script>
    // Change the style of the paragraph
    var styledParagraph =
document.getElementById("myStyledParagraph");
    styledParagraph.style.backgroundColor = "blue";
    styledParagraph.style.color = "white";
    styledParagraph.style.fontSize = "20px";
</script>
```

5. **Event Handling**

   You can use JavaScript to attach event listeners to HTML elements, allowing you to respond to user interactions such as clicks, mouseovers, and keyboard input. This enables dynamic behavior in your web applications.

```
<button id="myButton">Click me</button>
<script>
    // Attach an event listener to the button
    var button = document.getElementById("myButton");
    button.addEventListener("click", function () {
        alert("Button clicked!");
    });
</script>
```