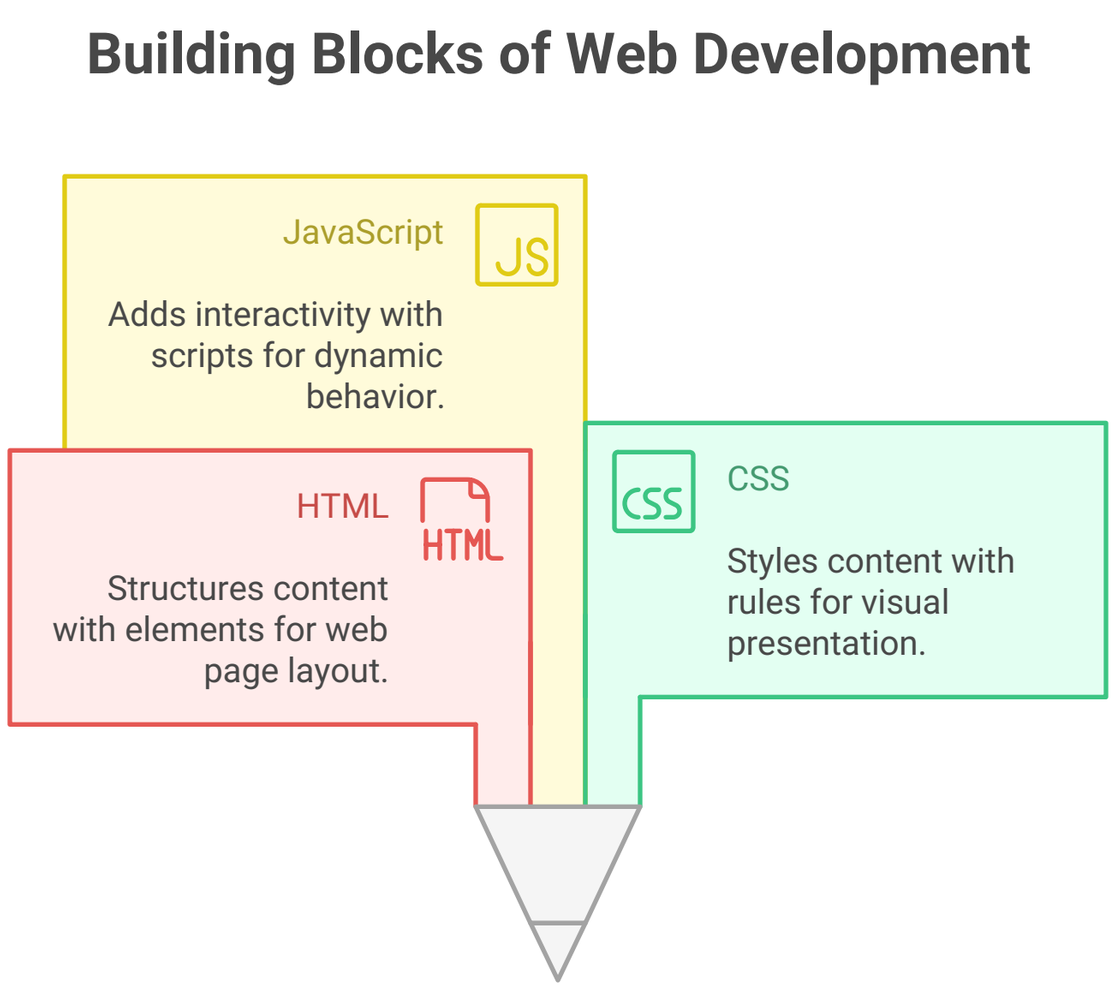
How HTML, CSS, and JavaScript Work Together

This document explains how HTML, CSS, and JavaScript work together to create dynamic and interactive web pages. It details their individual roles and how they interact, and provides a solution to the problem of linking a JavaScript file to an HTML page.

The Roles of HTML, CSS, and JavaScript

HTML, CSS, and JavaScript are the core technologies for building web pages. Each has a distinct role:



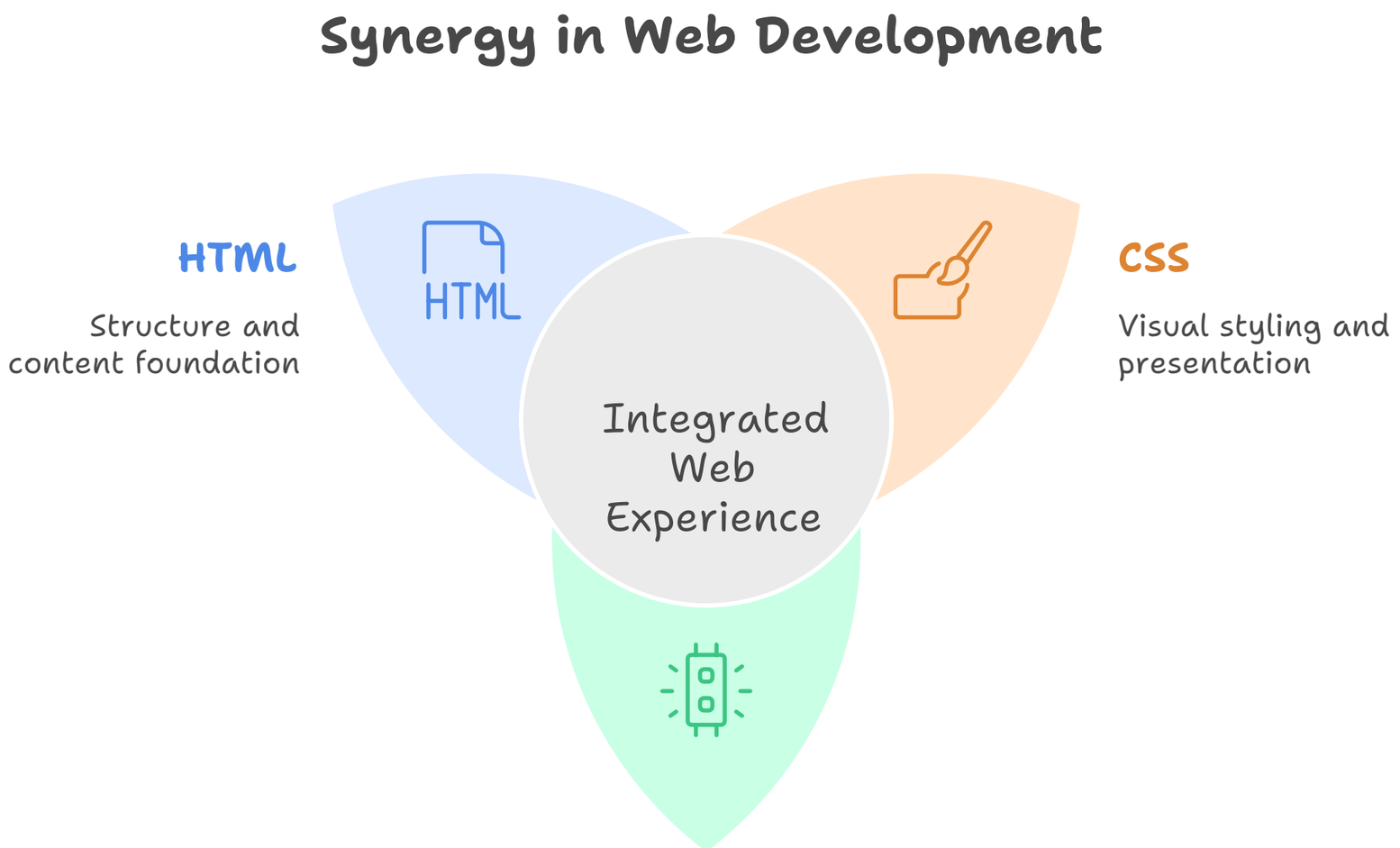
- **HTML (HyperText Markup Language):** Provides the structure and content of a web page. It uses elements (tags) to define headings, paragraphs, images, links, forms, and other content. HTML is the foundation upon which the other technologies build.

- **CSS (Cascading Style Sheets):** Controls the presentation and visual styling of a web page. It defines how HTML elements should be displayed, including colors, fonts, layout, and responsiveness. CSS separates the content from the presentation, making it easier to maintain and update the look and feel of a website.

- **JavaScript:** Adds interactivity and dynamic behavior to a web page. It allows you to manipulate the HTML structure (DOM - Document Object Model), respond to user events (like clicks and form submissions), make asynchronous requests to servers (AJAX), and create animations and other visual effects.

How They Work Together

These three technologies work together in the following way:



1. **HTML provides the structure:** The browser first parses the HTML document to create a DOM tree, which represents the structure of the page.

2. **CSS styles the structure:** The browser then applies CSS rules to the DOM tree to determine how each element should be displayed. CSS rules can be defined in separate CSS files, embedded within the HTML document using the <style> tag, or applied inline using the style attribute.

3. **JavaScript adds interactivity:** Finally, the browser executes JavaScript code, which can manipulate the DOM tree, modify CSS styles, and respond to user events. JavaScript code can be included directly within the HTML document using the <script> tag or linked from external JavaScript files.

Example

Consider a simple example:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Web Page</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <h1>Welcome to My Page</h1>
  <p id="myParagraph">This is a paragraph.</p>
  <button id="myButton">Click Me</button>

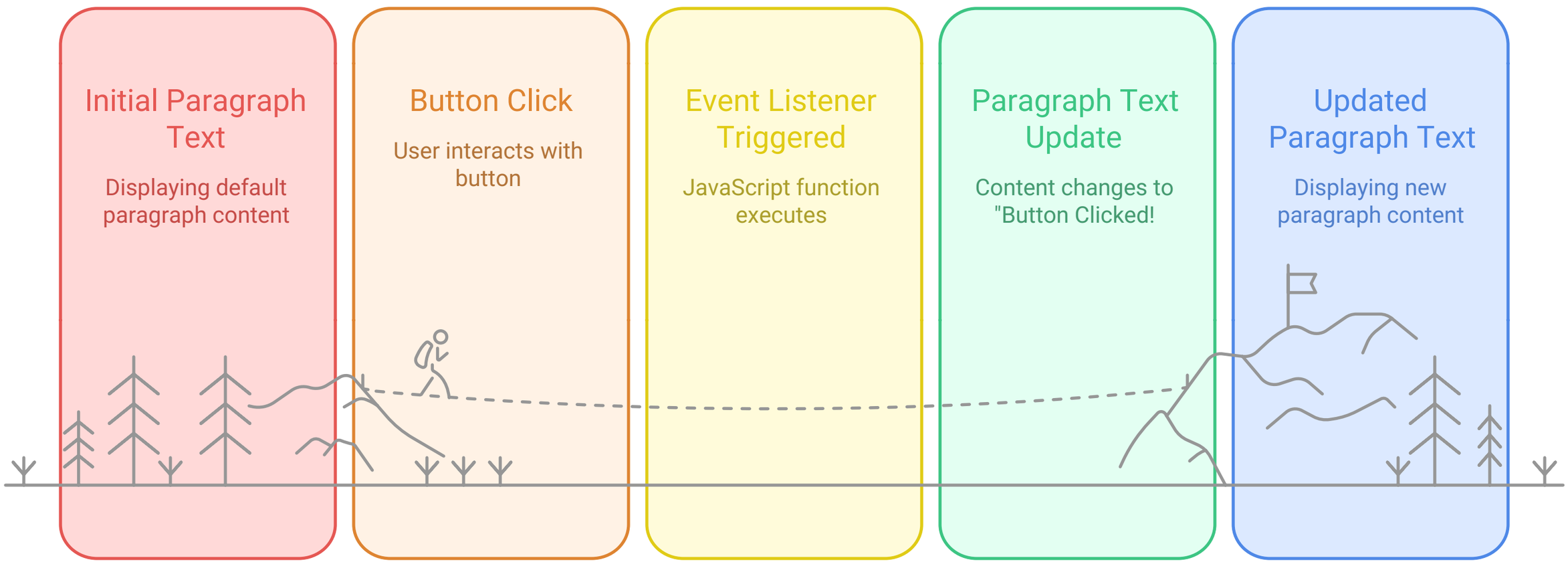
  <script src="script.js"></script>
</body>
</html>
```

```
/* style.css */
body {
  font-family: sans-serif;
  background-color: #f0f0f0;
}

h1 {
  color: blue;
}

#myParagraph {
  font-size: 16px;
}
```

```
// script.js
document.getElementById("myButton").addEventListener("click", function() {
  document.getElementById("myParagraph").textContent = "Button Clicked!";
});
```



In this example:

- The HTML file defines the structure of the page, including a heading, a paragraph, and a button. It also links to an external CSS file (style.css) and an external JavaScript file (script.js).
- The CSS file styles the page, setting the font, background color, heading color, and paragraph font size.
- The JavaScript file adds interactivity. It listens for a click event on the button and, when clicked, changes the text content of the paragraph.

Linking JavaScript to HTML: The Solution

The problem presented is: "Write a JavaScript program to link JavaScript file with the HTML page."

The solution is to use the <script> tag within the HTML document. The <script> tag has a src attribute that specifies the path to the external JavaScript file.

Here's how to link an external JavaScript file to an HTML page:

```
<!DOCTYPE html>
<html>
<head>
  <title>My Web Page</title>
</head>
<body>
  <h1>Welcome</h1>
  <p>This is a sample page.</p>

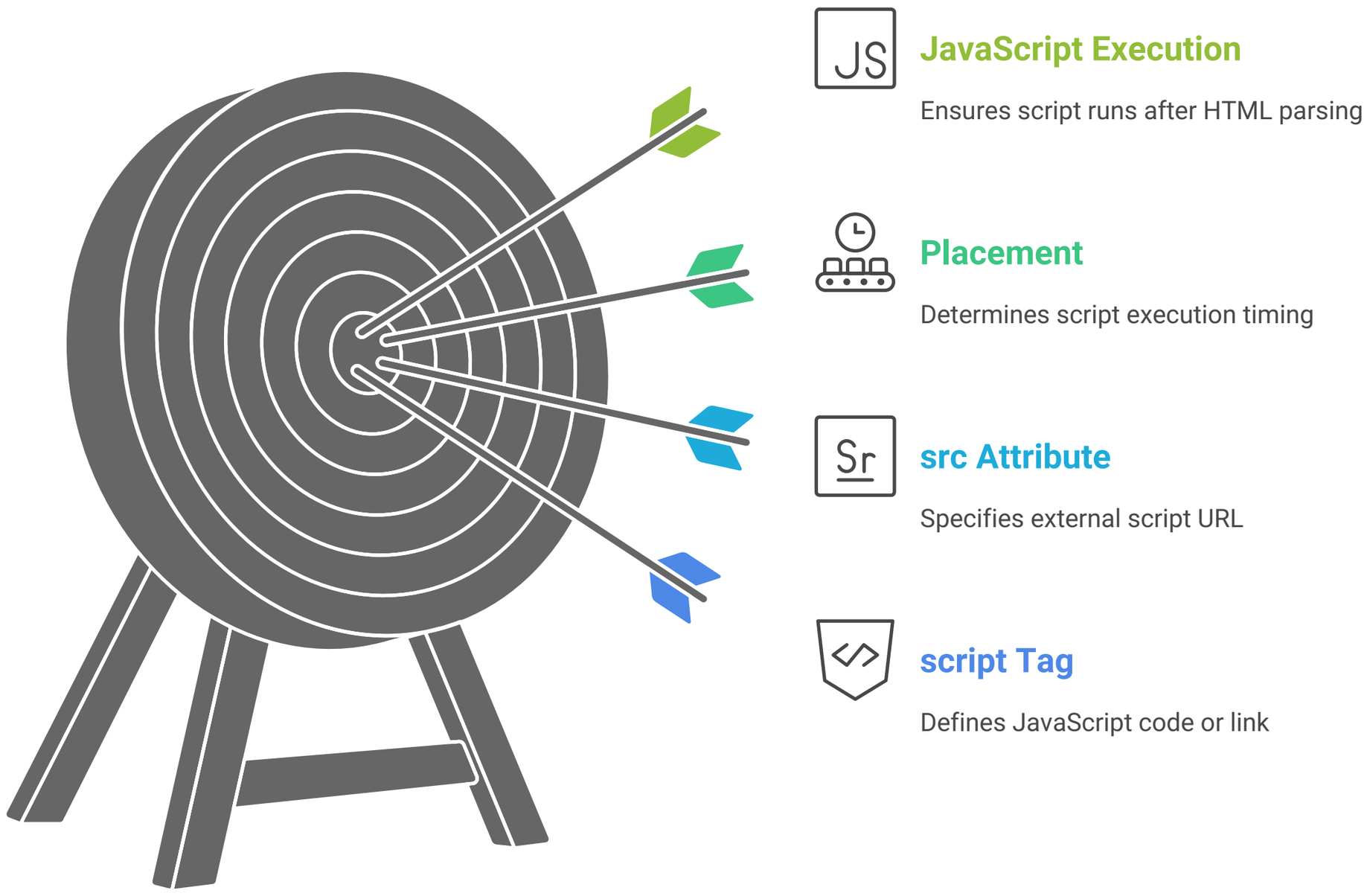
  <script src="myScript.js"></script>
</body>
</html>
```

In this example, the myScript.js file is linked to the HTML page. The browser will download and execute the JavaScript code in myScript.js when the HTML page is loaded.

Explanation:

- <script> **tag:** This tag tells the browser that the content within is JavaScript code (or a link to JavaScript code).
- src **attribute:** The src attribute specifies the URL of the external JavaScript file. The URL can be relative (like in the example above, where myScript.js is in the same directory as the HTML file) or absolute (e.g., https://example.com/js/myScript.js).

- **Placement:** The <script> tag is typically placed just before the closing </body> tag. This ensures that the HTML content is parsed and rendered before the JavaScript code is executed. This is important because the JavaScript code might need to interact with the HTML elements. Placing the script at the end of the body ensures that those elements exist when the script runs. Alternatively, you can place the <script> tag in the <head> section, but you might need to use the defer or async attributes to control when the script is executed.



Example myScript.js**:**

```
// myScript.js
alert("Hello from myScript.js!");
```

When the HTML page is loaded, the alert box will appear, demonstrating that the JavaScript file has been successfully linked and executed.

Alternative Placement and Attributes:

- <head> **with** defer**:**

```
```html
```

```
<!DOCTYPE html>
```

```
<html>
```
```

```
<head>
```

```
  <title>My Web Page</title>
```

```
  <script src="myScript.js" defer></script>
```

```
</head>
```

```
<body>
```

```
  <h1>Welcome</h1>
```

```
  <p>This is a sample page.</p>
```

```
</body>
```

```
</html>
```

```
```

```
The `defer` attribute tells the browser to download the script in the background and execute it after the HTML parsing is complete. This is a good option when the script needs to interact with the DOM but you want to avoid blocking the rendering of the page.
```

- <head> **with** async**:**

```
```html
```

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
  <title>My Web Page</title>
```

```
  <script src="myScript.js" async></script>
```
```

```
</head>
```

```
<body>
```

```
  <h1>Welcome</h1>
```

```
  <p>This is a sample page.</p>
```

```
</body>
```

```
</html>
```

```
```

```
The `async` attribute tells the browser to download the script in the background and execute it as soon as it's available, without waiting for the HTML parsing to complete. This is suitable for scripts that don't depend on the DOM being fully loaded.  However, be aware that the script might execute before the DOM is fully parsed, so you need to handle that case if your script interacts with the DOM.
```

In summary, HTML provides the structure, CSS provides the styling, and JavaScript provides the interactivity. By linking JavaScript files to HTML pages using the <script> tag, you can create dynamic and engaging web experiences. The placement of the <script> tag and the use of defer or async attributes can affect the performance and behavior of your web page, so choose the option that best suits your needs.