**The <script> element**

The <script> element is a non empty element which contains or links a JavaScript code.

* The <script> element containing the JS code can be inserted in the head element or the body element.

Illustration :

|  |  |
| --- | --- |
| <head> | <body> |
| <!DOCTYPE html>  <html>  <head>  <script>  function myFunction() {  document.getElementById("demo").innerHTML = "Paragraph changed.";  }  </script>  </head>  <body>  <h1>A Web Page</h1>  <p id="demo">A Paragraph</p>  <button type="button" onclick="myFunction()">Try it</button>  </body>  </html> | <!DOCTYPE html>  <html>  <body>  <h1>A Web Page</h1>  <p id="demo">A Paragraph</p>  <button type="button" onclick="myFunction()">Try it</button>  <script>  function myFunction() {  document.getElementById("demo").innerHTML = "Paragraph changed.";  }  </script>  </body>  </html> |
| From : https://www.w3schools.com/js/js\_whereto.asp | |

In this example, the code is only executed when the function is invoked.

* Aside from containing the JS code inside the script element, an external JS code may be used using the script element. Note that the following can also be included in the <head> and <body> element.

Illustration :

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

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

<script src="https://www.domainname.com/js/myScript.js"></script>

The use of external script are more effective if different web pages makes use of the same code. It also separates the content from the behavioral aspect of the page which makes the code cleaner

**Embedded script**

Upon reaching the script element, the rendering of the page is blocked until the JS is retrieved.

Thus, it is recommended to put the script element at the bottom of the body element to improve the display speed.

|  |
| --- |
| **var** html = "<h1>"+data.title+"</h1>" html += "<ul>" for(**var** i=0; i<data.supplies.length; i++) {  html += "<li><a href='supplies/"+data.supplies[i]+"'>"  html += data.supplies[i]+"</a></li>" } html += "</ul>"  From : http://www.embeddedjs.com/getting\_started.html |

*This is an example of an embedded code inside a function. The code within the function are executed as they encountered.*

***The no script***

The <noscript> is used as a disclaimer in case a JS type is not supported o is disabled byt the browser.

The lifetime of of a script is per page.

**defer vs async**

The defer and async are attributes of the script element which is used to indicate how the script should be executed.

Note : These attributes should not be used when the src attribute is present.

The defer attribute is used to tell the browser to execute the script only when page is finishe parsing.

On the other hand, the async is used to tell the browser to execute the script asynchronously as soon as it is available.

Asynchronous execution makes it possible for the browser to execute two things at a time, that is, while the browser fetches resources, the browser can also render the page. This means that even if the browser is fetching the script, you can still interact with the UI.

Note : Asynchronous fetch of resources is preferred over Synchronous fetch as synchronous fetch doesn’t allow the user to interact with the page while it is fetching the scripts.

Since asynchronous allows parallel fetching and rendering of page, the problem lies in knowing when the resource is ready for execution.The solution in this is the use of callback function