JavaScript Hello World Example

To insert JavaScript into an HTML page, you use the <script> element. There are two ways to use the <script> element in an HTML page:

- Embed JavaScript code directly into the HTML page.
- Reference an external JavaScript code file.

Embed JavaScript code in an HTML page

Placing JavaScript code inside the <script> element directly is not recommended and should be used only for proof of concept or testing purposes.

The JavaScript code in the <script> element is interpreted from top to bottom. For example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>JavaScript Hello World Example</title>
    <script>
        alert('Hello, World!');
    </script>
</head>
<body>
</body>
</html>
```

In the <script> element, we use the alert() function to display the Hello, World! message.

Include an external JavaScript file

To include a JavaScript from an external file:

- First, create a file whose extension is **.js** e.g., **app.js** and place it in the **js** subfolder. Note that placing the JavaScript file in the **js** folder is not required however it is a good practice.

- Then, use the URL to the JavasScript source code file in the **src** attribute of the <script> element.

The following shows the contents of the app.js file:

```
alert('Hello, World!');
```

And the following shows the **helloworld.html** file:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>JavaScript Hello World Example</title>
    <script src="js/app.js"></script>
</head>
<body>

</body>
</html>
```

If you launch the **helloworld.html** file in the web browser, you will see an alert that displays the Hello, World! message.

When you have multiple JavaScript files on a page, the JavaScript engine interprets the files in the order that they appear. For example:

```
<script src="js/service.js"></script>
<script src="js/app.js"></script>
```

In this example, JavaScript engine will interpret the **service.js** and the **app.js** files in sequence. It completes interpreting the service.js file first before interpreting the app.js file.

For the page that includes many external JavaScript files, the blank page is shown during the page rendering phase.

To avoid this, you include the JavaScript file just before the </body> tag as shown in this example:

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>JavaScript Hello World Example</title>
</head>
<body>

    <!-- end of page content here-->
    <script src="js/service.js"></script>
    <script src="js/app.js"></script>
</body>
</html>
```

The async and defer attributes

To change how the browser load and execute JavaScript files, you use one of two attributes of the <script> element async and defer.

These attributes take effect only on the external script files.
The async attribute instructs the web browser to execute the JavaScript file asynchronously. The async attribute does not guarantee the script files to execute in the order that they appear. For example:

```
<script async src="service.js"></script>
<script async src="app.js"></script>
```

The app.js file might execute before the service.js file. Therefore, you must ensure that there is no dependency between them.

The defer attribute requests the web browser to execute the script file after the HTML document has been parsed.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>JavaScript defer demonstration</title>
    <script defer src="defer-script.js"></script>
</head>
<body>
</body>
```

```
</html>
```

Even though we place the <script> element in the <head> section, the script will wait for the browser to receive the closing tag <html> to start executing.

Summary

- Use <script> element to include a JavaScript file in a HTML page.
- The async attribute of the <script> element instructs the web browser to fetch the JavaScript file in parallel and then parse and execute as soon as the JavaScript file is available.
- The defer attribute of the <script> element allows the web browser to execute the JavaScript file after the document has been parsed.