Programming in JavaScript

Section 1: JavaScript Basics

1.1 Introduction

The HTML <script> element tag defines an inline and external JavaScript. Traditionally, we can also refer to scripts other than JavaScript e.g. PHP. In HTML5 the <script> tag assumes the script to be JavaScript.

Below is an example code for an inline JavaScript code:

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Inline JavaScript Example</title>
    </head>
    <body>
        <script>
            windows.alert('Hello, World');
        </script>
    </body>
</html>
```

Note: This code displays the string Hello, World! in the browser's alert window.

console.log('Hello, World!'); on the other hand will print to the browser's JavaScript console within the developer tools.

While it is possible to write scripts this way, we would typically want to make the script file separate from the HTML. There are a number of advantages of having an external JavaScript file such as it can speed up performance if multiple pages refer to the script file because the browser can cache request to the same file, it avoids code duplication, easy to use tools such as linters if the file is separate and easy to edit/maintain/reuse the code.

To create a external JavaScript file we would typically create a external script js directory and store all of our JavaScript files in this directory. The JavaScript file will have an extension of .js for example script.js and this file can be loaded into our HTML file using the <script src=""> tag pointing to the relative path from the .html file. bBelow is an example:

```
js > JS index.js
  1    windows.alert('Hello, World');
```

```html
<!DOCTYPE html>
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>External JavaScript Example</title>
    </head>
    <body>
        <script src="./js/index.js"></script>
    </body>
</html>
```

Disclaimer: The <script> tag can be
places in either the <head> or <body> element tags but for performance reasons when loading synchronous scripts it is often encouraged to place the script tag at the bottom of the <body> element. That way the page content loads and is parsed before waiting for the script to download and execute.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>External JavaScript Example</title>
    <script src="./js/index.js" async></script>
</head>
<body>

</body>
</html>
```

If the script tag is placed in the <header> element we can add the async attribute which will allow the ability to load external scripts asynchronously while the page content downloads.

To download the script while the page content is downloading and execute the script after the HTML has parsed we can use the defer attribute. This will behave and treated as if the script was added to the bottom of the <body> element tags.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>External JavaScript Example</title>
    <script src="./js/index.js" defer></script>
</head>
<body>

</body>
</html>
```

Disclaimer: Assuming the web server supports loading scripts in parallel we would potentially get a slight performance boost but it is recommended to stick to the <script> tag for external JavaScript files to be placed at the bottom of the <body> element tag.