WEB230: JavaScript 1

Module 4: JavaScript and the Browser

URL

- each document on the web is identified by a unique URL (Uniform Resource Locator)

```
https://www.example.com/products/widget.html
|    |                |                      |
protocol      server           path
```

Protocol

- how to get the document
- web pages use either:
  - **HTTP** - HyperText Transfer Protocol
  - **HTTPS** - HTTP Secure - same as HTTP but with TLS encryption to prevent eavesdropping
- browsers can deal with other protocols or pass off to other applications
- if missing, it will default to the same protocol as the page

Server

- identifies the server on the internet
- can be a name or IP address
- names are looked up using DNS to get an IP address
- if missing, it will default to the same server as the page

Path

- where to find the document on the server
- if it starts with a slash or the server name is given
  - absolute path to document
- if it does not start with a slash
  - path starts in the same directory (folder) as the document

URL Example

```
https://www.example.com/products/widget.html
  |      |                  |       |
  |      |                  |       +-- document
  |      |                  +-- directory
  |      + - server
  + - protocol
```

URL Example Continued ...

```
/products/widget.html
|          |
|          +-- document
+-- directory
```

```
widget.html
|
+-- document
```

HTML

- HyperText Markup Language
- the first document loaded by a browser
- has references for other documents used
  - CSS
  - JavaScript
  - images
  - links, etc.

HTML and JavaScript

- HTML can include JavaScript code in `<script>` tags
- can appear anywhere in the `<head>` or the `<body>`
- by deault, will run as soon as the browser encounters the `<script>` tag

```
<h1>Testing alert</h1>
<script>
   alert('hello!');
</script>
```

JavaScript in Attributes

- some attributes can contain JavaScript

```
<button onclick="alert('Boom!');">DO NOT PRESS</button>
```

- this is an old way to do it
- mixes JavaScript into HTML
- it is discouraged
- **don't use it!**

Separating JavaScript

- large amounts of JavaScript clutter up the HTML
- the `<script>` tag can be used to import code from a separate file
- do not put the `<script>` tags in the external JavaScript file
- the closing `</script>` tag is still required

```
<h1>Testing alert</h1>
<script src="code/hello.js"></script>
```

Attributes

- `src` - location of JavaScript file
- `type` - script type - default `application/javascript`
- `charset` - character encoding - default is `UTF-8`
- `async` - download file in the background, run once downloaded
- `defer` - download file in the background, run it after page is loaded

Strict Mode

- run JavaScript in a stricter mode
  - requires better code
  - more errors, fewer bugs!
- include `"use strict";` at the beginning of your file
  - can also be included at the beginning of a function, scoped only to that function

Best Practice

- the `<script>` tag should go in the `head` section
- use an external JS file
- use the `defer` attribute to prevent blocking
- do **not** include the `type` attribute (JS is default)
- never use `document.write()` (it doesn't work with external JavaScript files!)
- always use strict mode

HTML File

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="utf-8" />
      <title>WEB230</title>
      <script src="script.js" defer></script>
   </head>

   <body>
      <h1>WEB230 - JavaScript</h1>
      <p>I hate JavaScript</p>
   </body>
</html>
```

JavaScript File

```
'use strict';

let message = 'JavaScript is fun!';
console.log(message);
alert('ਜਾਵਾ ਸਕ੍ਰਿਪਟ ਮਜ਼ੇਦਾਰ ਹੈ!');
console.log('😊');

document.querySelector('p').textContent = message;
```

In the Sandbox

- running downloaded code is dangerous
- JavaScript runs the code in a "sandbox"
- browser prevents it from doing dangerous things
  - can't access files
  - can't access other pages in your browser
  - network access is limited

Compatibility and the Browser Wars

- browsers compete for market
- in the early years they did this by introducing features that did not work in the competing browsers
- now they work together to try to make web pages compatible and introduce new features in a compatible way
- they compete on speed and features that do not effect page compatibility

Reference

Most of the material presented is not from the chapter.

- EJS Chapter 13
- W3Schools - the script tag
- MDN - more detail on script tag