Lecture 08: Client-Side Scripting
ICT1153

Buddhika Gayashani Jayaneththi
Department of ICT
Faculty of Technology
University of Ruhuna

Objectives

- Understand what is a scripting language
- Identify the basics of JavaScript
- Understand how to use JavaScript in HTML document

Scripting Language

- A **script** or **scripting language** is a computer language with a series of commands within a file that is capable of being executed without being compiled.
- Using html we can create **static** web pages (display the exact same information whenever anyone visits it).
  - Ex: Every visitor of that page will be greeted by the exact same text
- But using scripting language we can create a web site more **dynamic (**capable of producing different content for different visitors from the same source code file**)** and interactive.

Scripting Language....

- In contrast to programming languages that are compiled first before running, scripting languages do not compile the file and execute the file without being compiled.
- Scripting languages are often written to facilitate enhanced features of Web sites.
- **Java Script** is one of the most popular scripting languages.

Types of Scripting Languages

1. **Server side scripting**
   - Server-side scripting is used at the backend, where the source code is not viewable or hidden at the client side (browser).
   - They run on the server and generate results which are sent to the user.
   - Server-side scripting is useful in customizing the web pages and implementing the dynamic changes in the websites.
   - Server-side scripting is more secure than client-side scripting as the server-side scripts are usually hidden from the client end.

Ex: PHP, Python, Perl

Types of Scripting Languages

2. **Client side scripting**
   - Client-side scripting is used at the front end which users can see from the browser.
   - Client-side scripts rely on the user's computer.
   - It doesn't interact with the server to process data.
   - Interpreted by the browser.
   - It helps reduce the load on the server.
   - It is considered to be less secure in comparison to client-side scripting.

   Ex: JavaScript

Features of JavaScript

- **Developed by Netscape**
- **Light Weight Scripting Language**

  JavaScript is a lightweight scripting language because it is made for data handling at the browser only. Also, it has a limited set of libraries.

- **Dynamic Typing**

  JavaScript supports dynamic typing which means types of the variables are defined based on the stored value. For example, if you declare a variable x then you can store either a string or a Number type value or an array or an object.

- **Case Sensitive**
- **Usually embedded directly into HTML pages**

Features of JavaScript

- **Platform Independent**

  In JavaScript you can simply write the script once and run it anywhere and anytime.

  In general, you can write your JavaScript applications and run them on any platform or any browser without affecting the output of the Script.

- **Interpreted Language**

  The script written inside JavaScript is processed line by line.

  These Scripts are interpreted by JavaScript interpreter which is a built-in component of the Web browser.

Features of JavaScript

- **Client-side Validations**

  This is a feature which is available in JavaScript since forever and is still widely used because every website has a form in which users enter values, and to make sure that users enter the correct value, we must put proper validations in place, both on the client-side and on the server-side. JavaScript is used for implementing client-side validations.

- **More control in Browser**

  JavaScript being a client-side language provides many features that help developers to divide processing between browser and server hence reducing the load on servers by having basic processing, validations, temporary data saving using cookies, etc on the browser itself.

How to Use JavaScript with HTML..??

- JavaScript can be placed in the **<body>** or the **<head>** sections of an HTML page
- JavaScript code block should be written in between **<script> and </script>** tags
- Older examples may use a type attribute:
  - **<script type="text/javascript">**
- But type attribute is not required
  - JavaScript is the default scripting language in HTML

JavaScript in <head> Tag

```
<!DOCTYPE html>
<html>
    <head>
        <script>
            document.write("First JavaScript Example");
        </script>
    </head>
    <body>
    </body>
</html>
```

JavaScript in <body> Tag

- It is a good idea to place scripts at the bottom of the <body> element.
- This can improve page load, because script compilation can slow down the display.

```
<!DOCTYPE html>
<html>
    <body>
        <h1>JavaScript in body section</h1>
        <script>
            document.write("JavaScript in body section")
        </script>
    </body>
</html>
```

JavaScript as an External File

- JavaScript programs can also be uploaded from external files.
- External scripts are practical when the same code is used in many different web pages.
- JavaScript files have the file extension **.js.**
- To use an external script, put the name of the script file in the **src (source)** attribute of a <script> tag

Syntax:

- **<script src="externalFile.js"> </script>**
- Can place an external script reference in <head> or <body>
- External scripts do not contain <script> tags

JavaScript as an External File

HTML File

```
<html>
    <body>
        <h2>External JavaScript</h2>
        <script src="myScript.js"></script>
    </body>
</html>
```

JavaScript File

```
document.write("This is an external JS file");
```

JavaScript as an External File

- To add several script files to one page - use several script tags

```
<script src="myScript1.js"></script>
<script src="myScript2.js"></script>
```

External References

- An external script can be referenced in 3 different ways:
  - ✓ With a full URL (a full web address)
  - ✓ With a file path (like /js/)
  - ✓ Without any path

JavaScript as an External File

External References

- ✓ With a full URL (a full web address)

```
<script src="https://www.w3schools.com/js/myScript.js">
</script>
```

- ✓ With a file path (like /js/)

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

- ✓ Without any path

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

Advantages of Using an External JS File

- Separates HTML and JavaScript code.
- It makes HTML and JavaScript easier to read and maintain.
- Cached JavaScript files can speed up page loads.

JavaScript Basics

Display Outputs Using JavaScript

JavaScript can display outputs in different ways,

- Display using an alert box
  - **window.alert("message")**
- Write output using
  - **document.write("message")**
- Write output into an html element using
  - **innerHTML**
- Display output in browser console using
  - **console.log("message")**

Display Outputs Using JavaScript

Using innerHTML

- To access an HTML element, JavaScript can use the **document.getElementById(id)** method.
- The **id** attribute defines the **HTML element**.
- The **innerHTML** property defines the **HTML content**.

```
<html>
<body>
<h2>My First Web Page</h2>
<p>My First Paragraph.</p>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = 5 + 6;
</script>
</body>
</html>
```

Display Outputs Using JavaScript

Using document.write()

- For testing purposes, it is convenient to use document.write():

```
<html>
<body>
<h2>My First Web Page</h2>
<p>My first paragraph.</p>
<button type="button" onclick="document.write(5 + 6)">Try it</button>
</body>
</html>
```

Display Outputs Using JavaScript

Using window.alert()

- You can use an alert box to display data:

```
<html>
<body>
<h2>My First Web Page</h2>
<p>My first paragraph.</p>
<script>
window.alert(5 + 6);
</script>
</body>
</html>
```

Display Outputs Using JavaScript

Using console.log()

- For debugging purposes, you can call the console.log() method in the browser to display data.

```
<html>
<body>
<h2>Activate Debugging</h2>
<p>F12 on your keyboard will activate debugging.</p>
<p>Then select "Console" in the debugger menu.</p>
<p>Then click Run again.</p>
<script>
console.log(5 + 6);
</script>
</body>
</html>
```

JavaScript Comments

- JavaScript comments can be used to explain JavaScript code, and to make it more readable.
- JavaScript supports two style formats for comments.

  - **Single line comments** – Starts with //
    - Ex. // this is a single line comment
  - **Multi line comments** – write between /* & */
    - Ex. /*This is a block comment */

JavaScript Variables

- Variables are used to store data.
- Variable names are case sensitive.
- Use variable naming conventions when declaring a variable.

  ✓Names can contain letters, digits, underscores, and dollar signs
  ✓Names must begin with a letter
  ✓Names can also begin with $ and _
  ✓Names are case sensitive (y and Y are different variables)
  ✓Reserved words (like JavaScript keywords) cannot be used as names

JavaScript Variables

- Always declare JavaScript variables with **var**, **let**, or **const**.
- The **var** keyword is used in all JavaScript code from 1995 to 2015.
- The **let** and **const** keywords were added to JavaScript in 2015.
- If you want your code to run in older browser, you must use **var**.

Example :

- var x = 5;
- var y = 6;
- var z = x + y;

- x,y,z are the variables

JavaScript Variables

Declaring a variable

- JavaScript variables can be declared with **var, let, const** keyword

  Ex: **var** *name*;

- After the declaration, the variable has no value.

Assign a value to the variable

- To assign a value to the variable, use the equal sign and then the value:

  name="Peter";

- Also can assign a value to a variable at the same time when declaring the variable

  var *name*="Peter";

JavaScript Variables...

- You can declare many variables in one statement

ex: **var** *person* = "John Doe", *carName* = "Volvo", *price* = 200;

JavaScript Data Types

- JavaScript can handle many types of data.
  - Ex: Numbers, Strings, Arrays
- Strings are written inside double or single quotes.
- Numbers are written without quotes.
- If you put quotes around a number, it will be treated as a text string.

JavaScript Data Types - Example

```
<html>
<body>
<h2>JavaScript Data Types</h2>
<p id="demo"></p>
<script>
var num1 = 16;
var answer1 = "It's alright";
var answer2 = "He is called 'Anne'";
var answer3 = 'He is called "Anne"';
const cars = ["Saab", "Volvo", "BMW"];
document.getElementById("demo").innerHTML =
num1 + "<br>" + answer1 + "<br>" +
answer2 + "<br>" +
answer3 + "<br>" + cars;
</script>
</body>
</html>
```

JavaScript Operators

- There are several operators in JavaScript,
  - Arithmetic operators
  - Assignment operators
  - Comparison & Logical operators

Arithmetic Operators

- Arithmetic operators are used to perform arithmetic operations on numbers.

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulus
++	Increment
--	Decrement

Assignment Operators

- Assignment operators assign values to JavaScript variables.

Operator	Example	Same As
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y

Assignment Operators

```
<html>
<body>
<p id="demo"></p>
<script>
var x = 10;
var y = 5;
x += y;
document.getElementById("demo").innerHTML = x;
</script>
</body>
</html>
```

Comparison & Logical Operators

Operator	Description
==	equal to
!=	not equal
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

Comparison Operators - Example

```
<html>
<body>
<h2>JavaScript Comparison</h2>
<p>Assign 5 to x, and display the value of the comparison (x == 8):</p>
<p id="demo"></p>
<script>
let x = 5;
document.getElementById("demo").innerHTML = (x == 8);
</script>
</body>
</html>
```

- You can change == comparison operator to other types (Ex:!=,<,>,>=,<=) and see the results.

Logical Operators

- Logical operators are used to determine the logic between variables or values.
- Assume that x = 6 and y = 3 :

Operator	Description	Example
&&	and	(x < 10 && y > 1) is true
||	or	(x == 5 || y == 5) is false
!	not	!(x == y) is true

Ternary Operator

- Assigns a value to a variable based on some condition.
- The only JavaScript operator that takes three operands.
- A substitute for an *if….else* statement

```
if (condition)
   result = 'something';
else
   result = 'somethingelse';
```

```
result = (condition) ? 'something' : 'somethingelse';
```

Ternary Operator - Example

```
<html>
<body>
<h2>JavaScript Ternary Operator</h2>
<p id="demo"></p>
<script>
  var age = 16;
  var voteable = (age < 18) ? "Too young":"Old enough";
  document.getElementById("demo").innerHTML = voteable + " to vote.";
</script>
</body>
</html>
```

JavaScript String Concatenation

- ‘+’ operator is used to concatenate strings

Example :

- var txt1=“Hello”;
- var txt2=“John”;
- var txt3=txt1 + " " + txt2;
- Output = Hello John

Questions…