JavaScript Introduction

Course Code: CSC 3215 Course Title: Web Technologies

Dept. of Computer Science
Faculty of Science and Technology

Lecture No:	**10**	**Week No:**	**10**	**Semester:**	**Summer 19-20**
Lecturer:	*Tanvir Ahmed ; tanvir.ahmed@aiub.edu*				

Lecture Outline

1. Introduction to JavaScript
2. Usage of JavaScript and How to Use JavaScript
3. JS Variables and Data Types
4. JS Syntax, Output, Statements
5. JS Operators and Arithmetic operations
6. JS Events
7. JS Functions
8. JS Control flow (If else, loops)

Introduction to JavaScript

JavaScript is a client side programming language which was designed to add Interactivity to static HTML pages.

- JavaScript is a scripting Language
- A scripting language is lightweight programming language
- JavaScript is interpreted not compiled (means that script execute without preliminary compilation)
- JavaScript is also known as ECMASCRIPT
- Developed by ***Brandan Eich*** in September 1995.
- JavaScript and Java have almost nothing in common.

Usage of JavaScript

Topic sub heading..

- Great thing about JavaScript is that you will find tons of frameworks and Libraries already developed which can be used.
- JavaScript usage has now extended to mobile app development, desktop app development, and game development.
- Application of JavaScript is also very easy as it is pre installed in every modern web browsers.
- Once you learn about JS it can help you in front-end development as well as back-end development.

Where and How to Use JS

Where to Use

- In HTML JavaScript code can be inserted in between `<script></script>` tags

```
<script>
     Some JS Codes
</script>
```

- In some old examples you may see `<script type= "text/javascript" ></script>.` Where type is not required as JavaScript is the default scripting language ins HTML.
- Scripts can be placed in the `<body>`, or in the `<head>` section of an HTML page, or in both.
- Placing scripts at the bottom of the `<body>` element improves the display speed, because script interpretation slows down the display.

Where and How to Use JS

Where to Use

- JavaScript can also be placed in external files.
- Its better to put JavaScript codes in an external files.
- You can place any number of scripts in an HTML document.
- External script files are also useful when same code is used in many different web pages.
- External files must have the extension **.js**
- To use external js file you need to add a `src` attribute in the `<script>` tag. Where the value of src will be the path of the external JS file.

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

- The external file can be added both in `<head>` and `<body>` section.
- This will behave as if the codes are located in the `<script>` tag.
- **Remember external scripts cannot contain `<script>` tag only the JS codes.**

Benefits of External JS

- It separates the HTML and JavaScript so increased modularity.
- Many HTML pages can use same codes so increased reusability.
- Modification of code is easier so increased maintainability.
- Modern browsers cached JS files so speed up the page loads.
- JS files residing in another server can also be used.

Note: As browsers cached up the external JS files to speed up the execution you need to clear cache (ctrl +f5) of the page to reflect the change during development.

How to use JS

- JavaScript in <head>

```
<html>
	<head>
		<script>

		</script>
	</head>
	<body>
		<h1>A Web Page</h1>
	</body>
</html>
```

- JavaScript in <body>

```
<html>
	<head>
	</head>
	<body>
		<h1>A Web Page</h1>
		<script>

		</script>
	</body>
</html>
```

- External JavaScript
  - Can also be put in both head and body

```
<html>
	<head>
		<script src="myScript.js"></script>
	</head>
	<body>
		<h1>A Web Page</h1>
	</body>
</html>
```

- External JavaScript residing in another server
  - Put the URL of the file

```
<html>
	<head>
		<script
	src="https://www.w3schools.com/js/myScript1.js"></sc
	ript>
	</head>
	<body>
		<h1>A Web Page</h1>
	</body>
</html>
```

JS Variables and Data Types

JS Variables

- JavaScript is a loosely typed language.
- Loosely typed means the variable is not bound to store a specific type of data like strongly typed languages. If x is a variable it can hold 10 (integer) and `"ten" (string).`
- Creating variable in JavaScript is called declaring a variable.
- JavaScript variables are can be declared using `var` keyword. You can also create variable without `var`.
- Declared variable has the default value `undefined`
- Like other languages variable value can be assigned at the time of declaration using (=) assignment operator.

JavaScript Variable Naming Rules

- All variables in a must be identified by a unique name.
- These unique names can be called as variables or identifiers.
- Variables can be declared with or without `var` keyword.

```
var name="AIUB";
var number=10;
```

- Name can contain letters, digits, **underscore(_)** and **dollar($)** sign.
- Name must begin with a letter **or underscore or dollar sign**.
- **Names are case sensitive**.
- Reserved keyword can not be used as a variable name.
- Variable values can be initialized with = operator.
- String values are written with double quotation ("") or single quotation (').
- Numbers are written without quotation.
- **Re-declaration of variable will not loss the value.**

```
var name="AIUB";
var name;
//This will not lose the value of name. In next instructions
//value of name will remain "AIUB";
var name="AIUB";
var name ="BUET";
```

What Will Happen?

JavaScript Data Types

- As discussed earlier JavaScript is a loosely typed language, this can hold many data types.
- JavaScript has **4 primitive data** types and **2 complex data** type
  - Boolean (Can have only 2 values **true** or **false**)
  - Number
  - String
  - Undefined
  - Object
  - Function
- JavaScript has dynamic types.

```
var x;           // Now x is undefined
x = 5;           // Now x is a Number
x = "John";      // Now x is a String
```

- `typeof(x)` keyword returns the type of a variable.

Data Type Number

- JavaScript has only one type of number that is float.
- JavaScript numbers can be written with or without decimals.

```
var x = 3.14;     // A number with decimals
var y = 3;        // A number without decimals
```

- JavaScript numbers are always stored as 64-bit floating point.
- Integers (numbers without a period or exponent notation) are accurate up to 15 digits.
- The maximum number of decimals is 17, but floating point arithmetic is not always 100% accurate.

```
var x = 0.2 + 0.1;          // x will be 0.30000000000000004
```

- This can be solved by multiply and divide.

```
var x = (0.2 * 10 + 0.1 * 10) / 10;       // x will be 0.3
```

- NaN is a JavaScript reserved word indicating that a number is not a legal number. Trying to do arithmetic with a non-numeric string will result in NaN (Not a Number)

Must Read: https://www.w3schools.com/js/js_numbers.asp

Data Type String

- Strings are used for storing texts
- Strings can be written with single and double quotation.

```
var carName1 = "Volvo";  // Double quotes
var carName2 = 'Volvo';  // Single quotes
```

- To write string with quotations like *My name is 'Jon'* you can write as below

```
var name= "My name is /"Jon/"";
```

- In other case of My name is *My name is "Jon"* you can do as follows

```
var name= 'My name is "Jon"';
```

- By default string acts an object. So it has some properties and methods by default.
- (.) dot operator is used to access its properties and methods.
- Length of the string can be get by length property.

```
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
var sln = txt.length;
```

Must Read: https://www.w3schools.com/js/js_string_methods.asp

Data Type NULL VS Undefined

- Both of them means nothing.
- Both of them are same in terms of value but different in type.
- When a variable in JavaScript is declared its value is `undefined`
- Null value must be assigned programmatically otherwise its undefined

```
var a = null; //this one is null
var b; //this one is undefined
```

- You can also explicitly set a variable to equal undefined.

JS Syntax, Output, Statements

JS Syntax

- Fixed values are called literals and others are called variables.
  - `Sum = x + y * 8` → Literals; Variables
- Each instruction is called a statement. Each statement is ended with a semi colon.

```
var x = 10.5;  //this a statement
var x = 10.4; var y = 7; // 2 statements
```

- JavaScript ignores multiple spaces. You can add white space to your script to make it more readable.

```
var person = "Hege";
var person="Hege";
//both are same
```

- An expression is a combination of values, variables, and operators, which computes to a value.
  - `Sum = x + y * 8 //expression`

JS Syntax

- Comments are those line of codes which are not interpreted by the interpreter.
- In JavaScript the comments are of 2 types.
  - Single line

```
var x = 10;
//var y = 12;
```

→ Single line comment

  - Multi line

```
var z = 12;
/*
this lines
Will not be
Executed
*/
```

- We will be learning more about syntaxes day by day.

JS Output

- JavaScript can display values in 4 different ways
  - Into HTML element (inside tags)
  - In HTML Document
  - In alert box (Like a pop up message)
  - In console (usually for developers)

- We are about to write our first program using JavaScript. Some points to remember
  - Each HTML element should be uniquely identified by a `id` or `name`.
  - We will use `id` attribute to use JavaScript.
  - `name` can also be used.
  - JavaScript operates with a `document` object which works in a object oriented way.
  - There are some built in functions and attributes of `document` object we will use them.

JS Output [InnerHTML]

```
1. <html>
2.     <body>
3.        <h1>My First Web Page</h1>
4.          <p>My First Paragraph</p>
5.          <p id="demo"></p>
6.          <script>
7.             var para1 = document.getElementById("demo");
8.             para1.innerHTML = "Hello World";
9.          </script>
10.      </body>
11. </html>
```

Output of the code:

My First Web Page
My First Paragraph.
Hello World

- Lets observe the code. **In line 5** the `<p>` tag has `id` attribute. The value of `id` **attribute must be unique** in a web page.
- Now to uniquely identify a particular HTML element JavaScript `document` object has a predefined method `getElementById()` which receives the `id` **value as a parameter**.
- **This method returns the HTML element as a object**.
- **In line 7** `para1` variable holds the reference of `<p>` element as a object.
- `innerHTML` is an attribute of HTML element **which has starting and closing tag**.
- There are others attributes of document object we will be covering in further topics.

JS Output [document]

```
1. <html>
2.     <body>
3.        <h1>My First Web Page</h1>
4.          <p>My First Paragraph</p>
5.          <script>
6.             document.write("Hello World");
7.          </script>
8.      </body>
9. </html>
```

Output of the code:

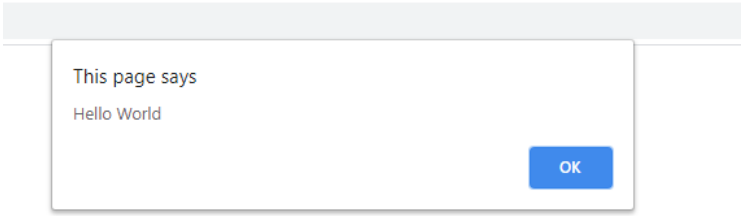
My First Web Page
My First Paragraph.
Hello World

- Lets observe **line 6**, `document` object has a method named `write()` which writes directly in the HTML page. Where ever you put the script it will execute there.

JS Output [alert]

```
1. <html>
2.     <body>
3.        <h1>My First Web Page</h1>
4.          <p>My First Paragraph</p>
5.          <script>
6.             alert("Hello World");
7.          </script>
8.      </body>
9. </html>
```

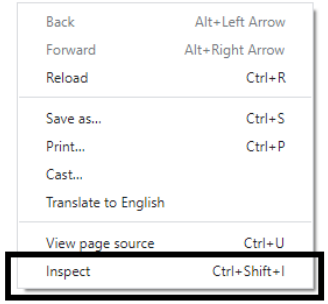
Output of the code:

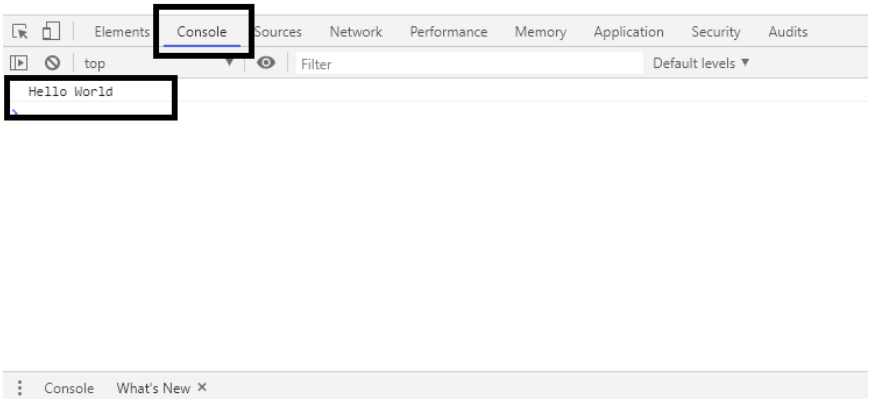


JS Output [console]

```
1. <html>
2.     <body>
3.        <h1>My First Web Page</h1>
4.        <p>My First Paragraph</p>
5.        <script>
6.           console.log("Hello World");
7.        </script>
8.      </body>
9. </html>
```

- Lets observe **line 6**, `console.log()` function which writes directly developer console.
- Right click on the HTML page →inspect





JS Operators and Arithmetic operations

Operators

Arithmetic Operator

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation
/	Division
%	Modulus (Division Remainder)
++	Increment
--	Decrement

Assignment Operator

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
**=	x **= y	x = x ** y

- The + operator can also be used to add (concatenate) strings.
- The += assignment operator can also be used to add (concatenate) strings.

JS Operators

Comparison Operator

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

Logical Operator

Operator	Description
&&	logical and
||	logical or
!	logical not

- `===` Operator is used for checking **both the value and type**

```
var a = 10;
var b = "10";
```

Both `a` and `b` holds the value of 10 but `a` is of **number** type and `b` is of **string** type.

`a==b` will result **true** as both the **value are same.**

`a===b` will check value and type. This will **return false**.

Arithmetic Operations

- Arithmetic operations are like other programming languages.

```
var a = 10;
var c = a + b * 10;
```

- As in traditional school mathematics, the multiplication is done first.
- Multiplication (*) and division (/) have higher precedence than addition (+) and subtraction (-).
- And (as in school mathematics) the precedence can be changed by using parentheses

```
var c = a + b * 10;
```

- When many operations have the same precedence (like addition and subtraction), they are computed from left to right.

```
var c = a + b – 10;
```

Must Read: https://www.w3schools.com/js/js_arithmetic.asp

Books

1. W3Schools Online Web Tutorials; URL: http://www.w3schools.com
2. PHP Documentation; URL: http://www.php.net/docs.php
3. Sams Teach Yourself Ajax JavaScript and PHP All in One; Phil Ballard and Michael Moncur; Sams Publishing; 2010
4. JavaScript Phrasebook; Christian Wenz; Sams Publishing; 2007
5. PHP and MySQL Web Development, 4/E; Luke Welling and Laura Thomson; AddisonWesley Professional; 2009
6. JavaScript for Programmers Paul J. Deitel and Harvey M. Deitel; Prentice Hall; 2009
7. Beginning PHP5, Apache, and MySQL Web Development; Elizabeth Naramore, Jason Gerner, Yann Le Scouarnec, Jeremy Stolz and Michael K. Glass; Wiley Publishing; 2005
8. XML in a Nutshell, 3/E; Elliotte Rusty Harold and W. Scott Means; O'Reilly Media; 2004

References

1. https://www.w3schools.com/js/
2. https://www.springboard.com/blog/history-of-javascript/