CSE 3002
INTERNET AND WEB PROGRAMMING

Module : 3

JavaScript Introduction

Introduction

- **JavaScript scripting language**
  - Enhances functionality and appearance
  - Client-side scripting
    - Makes pages more dynamic and interactive
  - Web browser contains a JavaScript interpreter

- **JavaScript is an Object Oriented Programming (OOP) language.**
  - Attributes (data) and behaviors (methods) are associated with the object.

JavaScript vs Java

- JavaScript is interpreted while Java is compiled
  - But server-side JavaScript is compiled
- JavaScript is object-based while Java is object-oriented
  - Object-based languages can utilize pre-defined objects, but you are limited in terms of creating your own objects
- JavaScript has loose data typing, while Java has strong data typing
  - Loose data typing means that a variable can hold any kind of data
- JavaScript has dynamic binding, while Java has static binding
  - Names bound at runtime
- JavaScript can access browser objects and functionality, while Java cannot

What is JavaScript?

- **Scripting language (object-oriented)**
  - Lightweight programming language developed by Netscape
  - Interpreted, not compiled

- **Designed to be embedded in browsers**
  - Ideal for adding interactivity to HTML pages
  - Detect browser versions
  - Work with info from user via HTML forms
  - Create cookies
  - Validate form data
  - Read and write HTML elements

- **Supported by all major browsers**

What is JavaScript?

- Syntax is similar to Java, but it’s not Java form.
- Usually JavaScript code is embedded within HTML code using the <script> tag:
- Can have more than one pair of script tags in a page

Client-Side JavaScript

- Detect whether the browser supports a certain plug-in
- Control a plug-in
- **Validate user form input**
- **Prompt a user for confirmation**
- **Perform post-processing of information retrieved from server-side JavaScript scripts**

Client-Side JavaScript

- It was designed to provide a quicker and simpler language for enhancing web pages and servers.
- Allows client browser to generate HTML on the fly.
- Java script provides increased functionality to the web browsers in the form of Java applets, Plugins and HTML elements.
- Provides a fairly complete set of Built in functions and commands to perform Mathematical ,string functions & etc..

<noscript> tag

- Html uses this <noscript> tag to display an alternative text if the browser does not support scripting languages.

Programming Notes

- Javascript can be located in the head, body or external file
  - **Head section**
    - Ensures script is loaded before trigger event
  - **Body section**
    - Script executes when body loads
  - **External**
    - Allows scripts to run on several pages

JavaScript Program : Syntax

- Inline scripting
  - Written in the **<body>** or **<head>** of a document
  - **<script>** tag
  - JavaScript Single-line comment symbol: //
  - Example:

```
<script type= "text/javascript">
   <! - -
            script code here
   - - >
</script>
```

Programming with JavaScript (1)

- **document** object:the HTML document the browser is currently displaying
- A statement should end with a semicolon **(;)**
- JavaScript is ***case sensitive***.
- **writeln**method writes a line in the document

```
document.writeln("<h1>Welcome</h1>");
```

JavaScript - Variables

Variables in JavaScript are defined by using the **var** operator (short for variable), followed by the variable name, such as:

```
var test = "hi";
```

In this example, the variable test is declared and given an initialization value of "hi" (a string)

You can also define two or more variables using the same var statement:

```
var test = "hi", test2 = "hola";
```

Variables using the same var statement don't have to be of the same type:

```
var test = "hi", age = 25;
```

variables in JavaScript do not require initialization:

```
var test;
```

Javascript : Naming Variables

- The first character of the name must be a letter or an underscore (_).
- All characters following the first character can be **letters, underscore, or digits**.
- Letters can be either **upper or lowercase**.
- JavaScript does distinguish between the two cases.
- All the following variable names are legal:

var test;

var $test;

var $1;

var _$te$t2;

Javascript Variables : More

- Variables can hold different types of values at different times
- Variables don't have to be declared before being used
- Define variable and assign value
- Reassigning value to Variables
- Working With JavaScript Variables
- Declare two variables in the same line
- Reference a variable without declaration
- Assign one variable to another variable
- Do simple calculation
- Braces indicate code blocks

Refer demo

Javascript :Variable Scope

- A variable can be either global or local in JavaScript.
- All variables are global unless they are declared in a function.
- Variables declared in a function is local to that function.
- It is possible for two variables with the same name to exist if one is global and the other is local.
- When accessing the variable within the function, you are accessing the local variable.
- If the variable is accessed outside the function, the global variable is used.
- Always use the **var** keyword to declare local variables in functions.

Refer Demo

JavaScript Output

- JavaScript Display Possibilities
- JavaScript can "display" data in different ways:

  - HTML element, using innerHTML
  - HTML output using document.write()
  - Alert box, using window.alert()
  - Browser console, using console.log()

Define message in one block and output it in another

```
<html>
<head>
<title>Define message in one block and output it in another</title>
<script language="javascript" type="text/javascript">
<!--
var msg = "Message in javascript";
//-->
</script>
</head>
<body>
<h1>Define message in one block and output it in another</h1>
<script language="javascript" type="text/javascript">
<!--
document.write("<P>" + msg + "</p>");
//-->
</script>
</body>
</html>
```

Reference javascript source file stored outside

```
<script src="path-to-file/fileName.js"></script>
```

JavaScript Can Change HTML Content

```
document.getElementById("demo").innerHTML
```

The **Document** method **getElementById**() returns an Element object representing the element whose id property matches the specified string. Since element IDs are required to be unique if specified, they're a useful way to get access to a specific element quickly.

Refer Demo

The Element property **innerHTML** gets or sets the HTML or XML markup contained within the element.

Where ids are unique to the element, means it can be used only to that particular tag and no other tag should have same **id**

Javascript - innerHTML

- The **innerHTML** property can be used to write the dynamic html on the html document.
- It is used mostly in the web pages to generate the dynamic html such as registration form, comment form, links etc.

Syntax

```
const content = element.innerHTML;

element.innerHTML = htmlString;
```

Javascript - outerHTML

- The outerHTML attribute of the Element DOM interface gets the serialized HTML fragment describing the element including its descendants.
- It can also be set to replace the element with nodes parsed from the given string.

<u>Syntax</u>

```
var content = element.outerHTML;
element.outerHTML = htmlString;
```

<u>Example</u>

```
var d = document.getElementById("d");
console.writeline(d.outerHTML);
```

To only obtain the HTML representation of the contents of an element, or to replace the contents of an element, use the innerHTML property instead

Number() Object

- Every number in JavaScript is treated as a floating-point number.
- JavaScript does support integers, octal, hexadecimal, and so on.
- At the lowest level, JavaScript sees numbers as floating-point numbers.

Syntax

```
var variable = new Number(value)
```

- The Number object represents numeric value types.
- You can create a Number object by specifying a value in the parameter for the number constructor.
- Comparison of Number objects using == operator compares Number objects and not the values.

Properties and Methods of the Number Object

MAX_VALUE	Specifies the largest value a number can have.
MIN_VALUE	Specifies the smallest value a number can have without being equal to 0.
NaN	Stands for Not a Number. Represents a value that is not equal to any numeric value.
NEGATIVE_INFINITY	A special value that represents a negative infinity value.
POSITIVE_INFINITY	A special value that represents a positive infinity value.
prototype	Represents the prototype for the number class.
toSource()	Returns a string representation of the number object.
toString()	Returns a string representing the specified number object.
valueOf()	Returns the primitive value of a number object as a number data type.

Number Methods

Method	Description
toExponential(fractionDigits)	Returns exponential value as a string. **Example:** var num = 100; Num.toExponential(2); // returns '1.00e+2'
toFixed(fractionDigits)	Returns string of decimal value of a number based on specified fractionDigits. **Example:** var num = 100; Num.toFixed(2); // returns '100.00'
toLocaleString()	Returns a number as a string value according to a browser's locale settings. **Example:** var num = 100; Num.toLocaleString(); // returns '100'
toPrecision(precisionNumber)	Returns number as a string with specified total digits. **Example:** var num = 100; Num.toPrecision(4); // returns '100.0'
toString()	Returns the string representation of the number value. **Example:** var num = 100; Num.toString(); // returns '100'
valueOf()	Returns the value of Number object. **Example:** var num = new Number(100); Num.valueOf(); // returns '100'