

BSc CsIt SEM - V

Web technology

Unit 4: Client-Side Scripting with JavaScript

Syllabus:

Unit 4: Client Side Scripting with JavaScript (9 Hrs.)

Structure of JavaScript Program; Variables and Data Types; Statements: Expression, Keyword, Block; Operators; Flow Controls, Looping, Functions; Popup Boxes: Alert, Confirm, Prompt; Objects and properties; Constructors; Arrays; Built-in Objects: Window, String, Number, Boolean, Date, Math, RegExp, Form, DOM; User Defined Objects; Event Handling and Form Validation, Error Handling, Handling Cookies, jQuery Syntax; jQuery Selectors, Events and Effects; Introduction to JSON

Introduction: JavaScript

JavaScript is a compact, object-based scripting language for developing client and server Internet applications. JavaScript can be divided into three parts; the core, the client and the server side.

The **core** is heart of the language including its operator, expressions, statements and sub-programs. **Client-side** JavaScript is a collection of objects that supports a control of browser and interactions with users. **Server-side** java script is collection of objects that makes the language useful on a web server like communication with database management system. Server-side JavaScript is used less frequently than client-side JavaScript.

Every collection of JavaScript code is referred as a **script** and an HTML document can include any number of embedded scripts. The script can be embedded in HTML document by using `<script>` tag with type as attribute which is set to "text/javascript" in head portion. This way of embedding script in HTML document is **explicit embedding** i.e. JavaScript code physically resides in HTML document.

Syntax:

```
<script type = "text/javascript"> </script>
```

Java script code can also be placed in its own file separating from HTML document. This way of embedding is known as **implicit embedding**. For this src attribute is included in script tag whose value is the name of a file that contains the script.

Syntax:

```
<script type = "text/javascript" src = "hello.js"> </script>
```

Note: it is recommended to use JavaScript code separately with that of html

Advantages of External Script

- External scripts separate HTML document and JavaScript code which makes the document easier to read and manipulate.
- Hides the script from the browser user.

Disadvantage of Internal Script:

- Mixing JavaScript and HTML code i.e. two completely different kinds of notation in same document will makes the document difficult to read.

BSc CsIt SEM - V

Web technology

- In some case, the person maintaining HTML document can be different from person maintaining JavaScript code which may lead to confusion.

Where to Put the JavaScript?

Scripts in a page will be executed immediately while the page loads into the browser. This is not always what we want. Sometimes we want to execute a script when a page loads, other times when a user triggers an event.

- **Scripts in the head section:** Scripts to be executed when they are called, or when an event is triggered, go in the head section. When you place a script in the head section, you will ensure that the script is loaded before anyone uses it.

```
<html>
  <head>
    <script language="Javascript"> some statements </script>
  </head>

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

- **Scripts in the body section:** Scripts to be executed when the page loads go in the body section. When you place a script in the body section it generates the content of the page.

```
<html>
  <head>
</head>
  <body>
    <script language="Javascript"> some statements </script>
  </body>
</html>
```

- **Scripts in both the body and the head section:** You can place an unlimited number of scripts in your document, so you can have scripts in both the body and the head section.

```
<html>
  <head>
    <script language="Javascript"> some statements </script>
  </head>
  <body>
    <script language="Javascript"> some statements </script>
  </body>
</html>
```

BSc CsIt SEM - V

Web technology

Structure of JavaScript Program

Whenever we write JavaScript program, we make use of the 'script' tag. We know what a normal HTML document is made up of. The 'script' tag is used to write JavaScript code and this 'script' tag is either placed inside the 'head' tag or inside the 'body' tag of Html document.

Example:

```
<!DOCTYPE HTML>
<html>
  <head>
    <title></title>
    <!-- Script tag can also be placed here -->
  </head>

  <body>
    <p>Before the script...</p>
    <!-- Script tag inside the body -->
    <script>
      // write the javascript code inside it
    </script>
    <p>...After the script.</p>
  </body>
</html>
```

We can put the script tag inside the 'head' or 'body' tag. Though it should be noted that each choice of putting the 'script' tag has its own consequences. For now, we can put the script tag anywhere as we want.

```
<!DOCTYPE html>
<html>
  <body>
    <h2>JavaScript in Body</h2>
    <p id="demo"></p>
    <script>
      document.getElementById("demo").innerHTML = "My First
      JavaScript";
    </script>
  </body>
</html>
```

Output:

JavaScript in Body

My First JavaScript

JavaScript code is inserted between <script> and </script> tags. 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.

BSc CsIt SEM - V

Web technology

Screen Output and Keyboard Input:

JavaScript can display data in different way using innerHTML, document.write(), and alert().

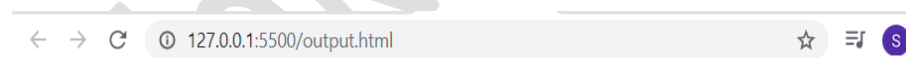
innerHTML

innerHTML is used to show output on HTML element and also used to manipulate content inside HTML element. For innerHTML, HTML element should be access by some way like using tag name, id, class name etc. The following example will illustrate use of innerHTML.

Example 1:

```
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=`, initial-scale=1.0">
  <title>screen Outputs</title>
</head>
<body>
  <h1 style="color: orange;text-align: center;">Example of Showing Output Using Inner
  HTML</h1>
  <p id="p1">This is first normal paragraph using html tag. second paragraph is shown using
  javascript inner html</p>
  <!-- in p2, output produce by using inner html is shown -->
  <p id="p2"></p>
  <script type="text/javascript">
    document.getElementById("p2").innerHTML="this is second paragraph created through
    use of inner html";
  </script>
</body>
</html>
```

Output:



Example of Showing Output Using Inner HTML

This is first normal paragraph using html tag. second paragraph is shown using javascript inner html

this is second paragraph created through use of inner html

BSc CsIt SEM - V

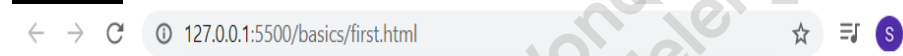
Web technology

Example 2:

```
<!DOCTYPE html>
<html>
<head>
<title> use of innerHTML </title>
<script type = "text/javascript">
function change(){
document.getElementById("p1").innerHTML = "this is second paragraph ";
//this will change content of paragraph to: this is second paragraph.
}
</script>
</head>

<body>
<h1 style="color: blueviolet; text-align: center;">Example of Showing Output Using Inner
HTML</h1>
<p id = "p1"> this is first paragraph which will be change if a button is clicked</p>
<form>
<input type = "button" value = "Change Text" onclick= "change()" />
</form>
</body>
</html>
```

Output:



Example of Showing Output Using Inner HTML

this is first paragraph which will be changed if button is clicked

Change Text

Figure: output before clicking on button

Example of Showing Output Using Inner HTML

first paragraph is changed to second paragraph

Change Text

Figure: output after clicking on button

Document.write

Document.write is used to create output which is dynamically created HTML document content. this content is specified in the parameter of write. Write is used to create markup so, the only useful punctuation in its parameter is in the form of HTML tags. Therefore, parameter of write include HTML tag like
. Document.write () should only be used for

BSc CsIt SEM - V

Web technology

testing because it will delete all the existing HTML contents after an HTML document is loaded.

Example1: shows change in content of paragraph when button is clicked.

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title> screen output</title>
  <script type="text/javascript">

    function change() {

      document.write("this is second paragraph");

    }
  </script>
</head>
<body>
  <h1 style="color: rgb(6, 56, 25);text-align: center;">Example of Showing Output Using Document.Write</h1>
  <p id="p1">this is first paragraph which will be changed if button is clicked</p>

  <!-- compare output with innerHTML to explore difference on document.write and innerHTML -->
  <form>

    <input type="button" name="c1" value="Change Text" onclick="change()"/>
  </form>
</body>
</html>
Output:

```

Example of Showing Output Using Document.Write

this is first paragraph which will be changed if button is clicked

Change Text

Figure: before clicking on button

this is second paragraph

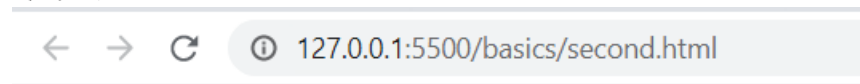
Figure: After clicking on button

BSc CsIt SEM - V

Web technology

Example 2: shows how HTML tags can be added on Document.write

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>screen output</title>
</head>
<body>
  <script type="text/javascript">
    document.write("<h1>This is web 1</h1><p> Here we will study: <br/> html <br/> css
<br/> javascript </p>");
  </script>
</body>
</html>
```



This is web 1

Here we will study:

html

css

javascript

Figure: example of using document.write

alert method

The **alert** method opens as a dialog window with “OK” button and displays its parameter in that window. The string parameter of alert is not HTML code i.e. it produces plain text. Therefore, string parameter of alert will not include HTML tags.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>screen output</title>
</head>

<body>
  <h3>Example of Alert</h3>
  <script type="text/javascript">
    alert("hello this is web programming 1");
  </script>
</body>
```

BSc CsIt SEM - V

Web technology

</html>

Output:

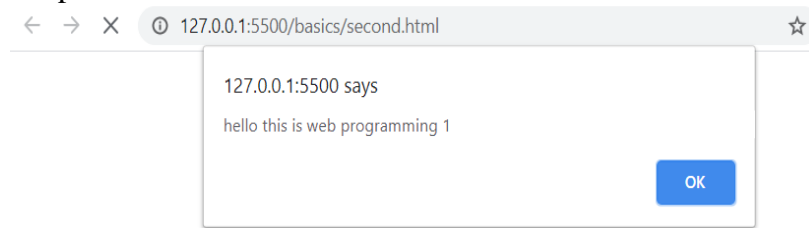


Figure: example of alert

Console.log

Console.log is also used for displaying data mostly for debugging purposes. It produces an output in browser console. For example:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>screen output</title>
</head>
<body>
  <h3 style="color: blue;text-align: center;">Example of console.log </h3>
  <h3 style="color: red;text-align: center;">To see output press F12 and click on
  console</h3>

  <script type="text/javascript">
    var x,y,z;
    x=10, y=20.5, z=false;
    console.log("the suum of x and y is :", x + y);
    console.log("value of z is ",z);
  </script>
</body>
</html>
```

Output:

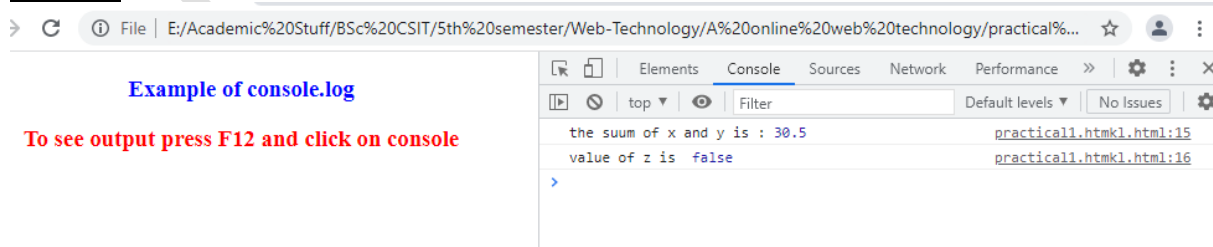


Figure: Example of console.log

BSc CsIt SEM - V

Web technology

Variables:

Variables are used to store data temporarily i.e. they are like container for storing data. In java script, variables are dynamically typed that means a variable can be used for anything or their types are not determined. Any variable can have any values i.e. values can be of primitive type like integer, string, Boolean or it can be reference to object. The type of the value contained by particular variable can be determined by interpreter and in many cases, interpreter can convert the type of values of variable to whatever it is needed for the context.

For example: if variable contains numeric value like 23 then interpreter implement it as a numeric type, if variable contains decimal values like 4.5 then interpreter implement it as a floating type, if variable contains values in double inverted comma like "hello" then interpreter implement it as a String type.

A variable in JavaScript can be declared either by using reserved word **var** keyword or by simply assigning a value. A variable that has been declared but not assigned any value will have undefined value i.e. its value will be undefined. A variable can also be declared using let keyword.

Example :

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>screen output</title>
</head>
<body>
  <h3 style="color: blue;">Variable </h3>
  <script type="text/javascript">
    var x,y,z;
    x=10, y=20.5, z=false;
    document.write("the suum of x and y is :", x + y);
    document.write("<br>");
    document.write("value of z is ",z);
  </script>
</body>
</html>
```

Output:

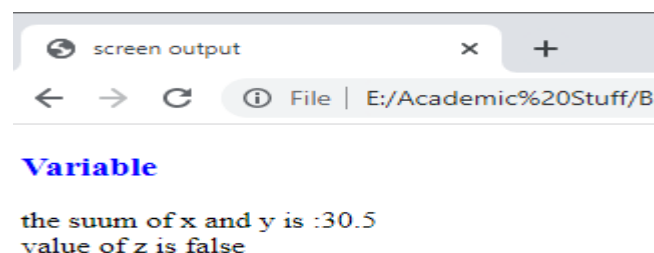


Figure: declaration of variables using different keyword

BSc CsIt SEM - V

Web technology

Rules for declaring variables:

- Variable name should not be reserved keyword like let, new, if, try etc.
- Name of the variable should be meaning full like firstName, lastName etc.
- Variable name cannot start with number like 1name i.e. variable name should start with letter.
- Whitespace and hyphen are not allowed but underscore can be used.
- Variable are case sensitive. For example, first and First are different.

Constant:

Constant is used to declare fixed value that does not change during the execution of program. In JavaScript, constant is declared using const keyword. It is used in situation where we don't want value to be changed i.e. fixed value. If we attempt to change the value of constant then error will be occurred.

For example:

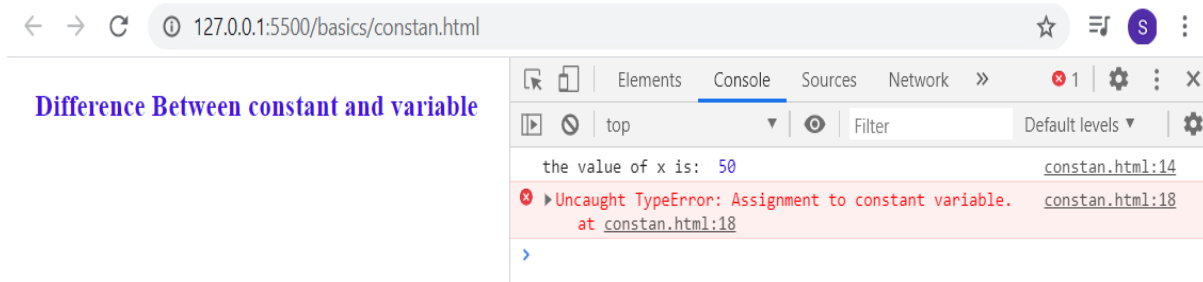
This example will show the difference between normal variable and constant:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Constant example</title>
  </head>
  <body>
    <h3 style="color: rgb(71, 15, 224);text-align: center;">Difference Between constant
    and variable </h3>
    <script type="text/javascript">
      var x= 40;
      x=50;
      // the value of x will be change from 40 to 50
      console.log ("the value of x is: ",x);
      //declaring constant
      const pi = 3.1415;
      //attempt to change this constant will produce error
      pi = 1.5;
      console.log("the new value of pi is: ",pi);
    </script>
  </body>
</html>
```

Output:

BSc CsIt SEM - V

Web technology



Data Types:

There are two data types in JavaScript: primitive/value type and object or reference type

Primitive Data Type:

JavaScript has five primitive data types: **Number, String, Boolean, Undefined and Null.**

The Number Data Type

The number data type is used to represent positive or negative numbers with or without decimal place, or numbers written using exponential notation e.g. $1.5e-4$ (equivalent to 1.5×10^{-4}).

Example

```
var a = 25;      // integer
var b = 80.5;    // floating-point number
var c = 4.25e+6; // exponential notation, same as 4.25e6 or 4250000
var d = 4.25e-6; // exponential notation, same as 0.00000425
```

The Number data type also includes some special values which are: Infinity, -Infinity and NaN. Infinity represents the mathematical Infinity ∞ , which is greater than any number. Infinity is the result of dividing a nonzero number by 0, as demonstrated below:

Example

```
alert(16 / 0); // Output: Infinity
alert(-16 / 0); // Output: -Infinity
alert(16 / -0); // Output: -Infinity
```

While NaN represents a special Not-a-Number value. It is a result of an invalid or an undefined mathematical operation, like taking the square root of -1 or dividing 0 by 0, etc.

Example

```
alert("Some text" / 2); // Output: NaN
alert("Some text" / 2 + 10); // Output: NaN
alert(Math.sqrt(-1)); // Output: NaN
```

The String Data Type

The string data type is used to represent textual data (i.e. sequences of characters). Strings are created using single or double quotes surrounding one or more characters, as shown below:

Example

```
var a = 'Hi there!'; // using single quotes
var b = "Hi there!"; // using double quotes
```

BSc CsIt SEM - V

Web technology

The Boolean Data Type

The Boolean data type can hold only two values: true or false. It is typically used to store values like yes (true) or no (false), on (true) or off (false), etc. as demonstrated below:

Example

```
var isReading = true; // yes, I'm reading
var isSleeping = false; // no, I'm not sleeping
```

The Undefined Data Type

The undefined data type can only have one value-the special value undefined. If a variable has been declared, but has not been assigned a value, has the value undefined.

Example

```
var a;
var b = "Hello World!"

alert(a) // Output: undefined
alert(b) // Output: Hello World!
```

The Null Data Type

This is another special data type that can have only one value-the null value. A null value means that there is no value. It is not equivalent to an empty string ("") or 0, it is simply nothing. A variable can be explicitly emptied of its current contents by assigning it the null value.

Example

```
var a = null;
alert(a); // Output: null

var b = "Hello World!"
alert(b); // Output: Hello World!

b = null;
alert(b) // Output: null
```

The Object Data Type (Reference Type):

Reference type consist of object, array and function. An object is something that exist with the problem domain and can be identifies by data or behavior. Object can be view as object in real life for example a book is an object which has its name, writer name, page number etc. which has its own identity and distinguish from other.

In JavaScript, when we need to deals with similar type of multiple variables, we can put this in an object. In java script object is created same like creating a variable but a curly brace (object literals) is used and between its curly braces object's properties and behavior are determined.

Syntax:

```
Let book = {};
```

BSc CsIt SEM - V

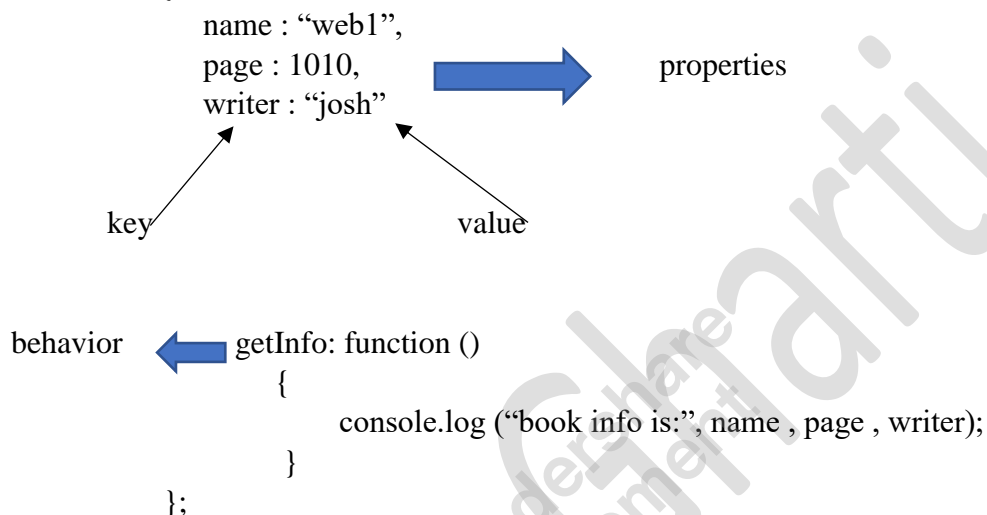
Web technology

Between curly braces, properties (data) are determined using key value pair and behavior (what it can do) is determined by creating respective function.

For example: book can have its name, writer name, page number and so on which determines its properties and certain function that determines its properties:

Syntax:

Let book = {



Function and properties can be accessed by: name of object. its properties or function:

```

book.name = "java"; //this changes name of book to java
book.getInfo(); //calling getInfo() function

```

Following example shows creating an object and accessing its properties and function:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Object type example</title>
</head>
<body>
  <h3 style="color: green;">Example of creating object, properties and behavior </h3>
  <script >
    //creating object
    let book = {
      //creating properties
      name : "web",
      page : "1010",
      writer : "josh",
      //creating behavior or function

```

BSc CsIt SEM - V

Web technology

//this determines what an object can do.

```
getInfo: function () {
    //showing on console
    console.log("the name of book is:",this.name);
    console.log("book contains: ",this.page,"pages");
    console.log("the book is written by: ",this.writer);
    //showing on browser main page
    document.write("the name of book is:"+this.name+"<br>")
    document.write("book contains:"+this.page+"<br>")
    document.write("the book is written by: "+this.writer+"<br>")
}
};
```

book.getInfo();//accessing function or behavior using dot operator

book.name ="java"; // accessing property name. this will change name into java;

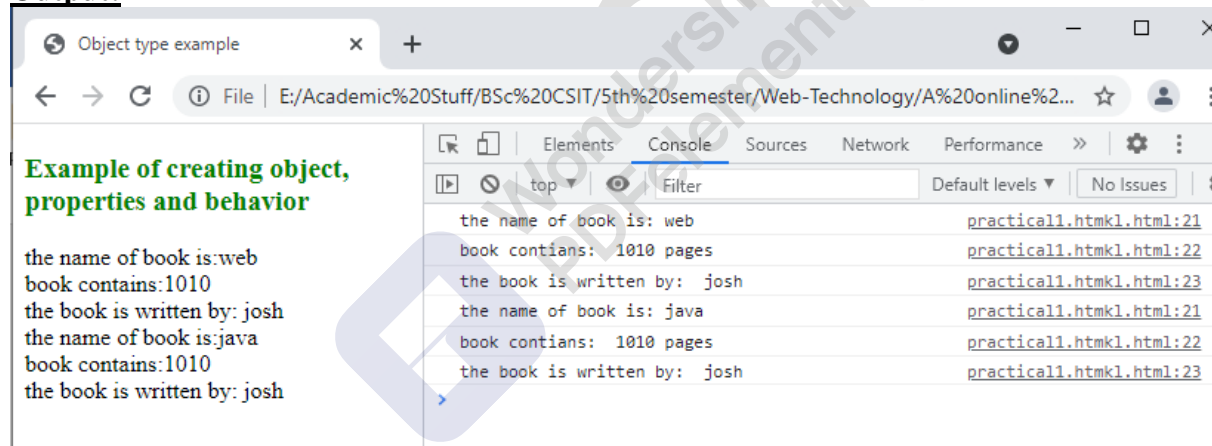
book.getInfo();

</script>

</body>

</html>

Output:



Example 2:

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript Objects</h2>

<p>Objects can also have methods</p>

<p id="demo"></p>

<script>

// Create an object:

```
const student = {
    firstName: "Teksan",
    lastName: "gharti",
```

Teksan Gharti

BSc CsIt SEM - V

Web technology

```
rollno: 10,  
function_fullName: function() {  
    return this.firstName + " " + this.lastName;  
}  
};  
// Display data from the object:  
document.getElementById("demo").innerHTML = student.function_fullName();  
</script>  
  
</body>  
</html>
```

Output:

JavaScript Objects

Objects can also have methods

Teksan gharti

Statements: Expression, Keyword, Block;

Statement

A computer program is a list of "instructions" to be "executed" by a computer. the programming instructions in a programming language, are called statements.

JavaScript statements are composed of: Values, Operators, Expressions, Keywords, and Comments. JavaScript statements are the commands to tell the browser to what action to perform. Statements are separated by semicolon (;).

Example:

```
document.getElementById("demo").innerHTML = "Welcome";
```

This statement tells the browser to write "welcome" inside an HTML element with id="demo":

Keyword

Keywords are reserved words that are part of the syntax in the programming language. Here, const is a keyword that denotes that a is a constant. ... Keywords cannot be used to name identifiers.

Example: - const, continue, do, for, enum etc

Blocks

JavaScript statements can be grouped together in code blocks, inside curly brackets {...}.



BSc CsIt SEM - V

Web technology

Example:

```
function myFunction()
{
    document.getElementById("demo1").innerHTML = "Hello Dolly!";
    document.getElementById("demo2").innerHTML = "How are you?";
}
```

JavaScript Operators

Operators are used to operate on values.

- 1) Arithmetic Operators
- 2) Assignment Operators
- 3) Comparison Operators
- 4) Logical Operators
- 5) String Operator

Arithmetic Operators			
Operator	Description	Example	Result
+	Addition	x=2 and x+2	4
-	Subtraction	x=2 and 5-x	3
*	Multiplication	x=4 and x*5	20
/	Division	15/5	3
%	Modulus (remainder)	5%2 10%2	1 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

BSc CsIt SEM - V

Web technology

Assignment Operators		
Operator	Example	Is The 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

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

Logical Operators

Operator	Description	Example
&&	and	X=6 , y=3 then (x < 10 && y > 1) returns true
	or	X=6 , y=3 then (x==5 y==5) returns false
!	not	X=6 , y=3 then !(x==y) returns true

String Operator

To stick two or more string variables together, use the + operator. To add a space between two string variables, insert a space into the expression, or in one of the strings.

txt1="What a very" txt2="nice day!" txt3=txt1+txt2	Or	txt1="What a very" txt2="nice day!" txt3=txt1+" "+txt2	or	txt1="What a very " txt2="nice day!" txt3=txt1+txt2
output		output		Output
"What a verynice day!"		"What a very nice day!"		"What a very nice day!"

BSc CsIt SEM - V

Web technology

Conditional Operator / Ternary Operator

JavaScript also contains a conditional operator that assigns a value to a variable based on some condition.

Syntax: <code>variablename=(condition)?value1:value2</code>
--

Example: <code>greeting=(visitor=="PRES")?"Dear President ":"Dear "</code>

If the variable visitor is equal to PRES, then put the string "Dear President " in the variable named greeting. If the variable visitor is not equal to PRES, then put the string "Dear " into the variable named greeting.

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <script>
      function myFunction()
      {
        let age = document.getElementById("age").value;
        let voteable = (age < 18) ? "Too young":"Elegible for voting";
        document.getElementById("demo").innerHTML = voteable + " to vote.";
      }
    </script>
  </head>
  <body>
    <h2>Ternary operator</h2>
    <p>Input your age and click the button:</p>
    <input id="age"/>

    <button onclick="myFunction()">Click</button>
    <p id="demo"></p>
  </body>
</html>
```

Output:

Ternary operator

Input your age and click the button:

BSc CsIt SEM - V

Web technology

Ternary operator

Input your age and click the button:

Too young to vote.

Ternary operator

Input your age and click the button:

Elegible for voting to vote.

There are additional assignment operators for bitwise operations:

- $x \ll= y$ means $x = x \ll y$
- $x \gg= y$ means $x = x \gg y$
- $x \gg\gg= y$ means $x = x \gg\gg y$
- $x \&=$ means $x = x \& y$
- $x \wedge=$ means $x = x \wedge y$
- $x |=$ means $x = x | y$

Operator Precedence

The *precedence* of operators determines the order they are applied when evaluating an expression. You can override operator precedence by using parentheses. The precedence of operators, from lowest to highest is as follows:

- comma ,
- assignment $=$ $+=$ $-=$ $*=$ $/=$ $\% =$ $\ll=$ $\gg=$ $\gg\gg=$ $\&=$ $\wedge=$ $|=$
- conditional $?$:
- logical-or $||$
- logical-and $\&\&$
- bitwise-or $|$
- bitwise-xor \wedge
- bitwise-and $\&$
- equality $==$ $!=$
- relational $<$ $<=$ $>$ $>=$
- shift \ll \gg $\gg\gg$
- addition/subtraction $+$ $-$
- multiply/divide $*$ $/$ $\%$
- negation/increment $!$ \sim $-$ $++$ $--$

BSc CsIt SEM - V

Web technology

Flow Control

Flow Control allows the execution of code only under certain conditions. In JavaScript, we used, if statements, if/else statements, if/else if/else statements, ternary operators, and switch statements. They are also called Conditional statements.

JavaScript Conditional Statements

Conditional statements in JavaScript are used to perform different actions based on different conditions. Very often when you write code, you want to perform different actions for different decisions.

In JavaScript we have three conditional statements:

- **if statement** - use this statement if you want to execute a set of code when a condition is true
- **if...else statement** - use this statement if you want to select one of two sets of lines to execute
- **switch statement** - use this statement if you want to select one of many sets of lines to execute

If Statement

You should use the “if statement”, if you want to execute some code if a condition is true.

Syntax

```
if (condition)
{
    code to be executed if condition is true
}
```

Example:

```
<!DOCTYPE html>
<html>
  <head>
  <script>
    function myFunction()
    {
      let age = document.getElementById("age").value;
      if(age < 18)
      {
        document.getElementById("demo").innerHTML = " You are not eligible for
voting";
      }
    }
  </script>
  </head>
  <body>
    <h2>If statement</h2>
    <p>enter your age:</p>
    <input id="age"/>
    <button onclick="myFunction()">Click</button>
    <p id="demo"></p>
```

BSc CsIt SEM - V

Web technology

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

Output:

If statement

enter your age:

If statement

enter your age:

You are not eligible for voting

If...else Statement:

You use the if...else statement if you want to execute some code if a condition is true and another code if a condition is false. Use this statement if you want to execute one set of code if the condition is true and another set of code if the condition is false.

Syntax

If (condition)

```
{
    code to be executed if condition is true
}
else
{
    code to be executed if condition is false
}
```

Example:

```
<!DOCTYPE html>
<html>
  <head>
  <script>
    function myFunction()
    {
      let age = document.getElementById("age").value;
      if(age < 18)
      {
        document.getElementById("demo").innerHTML = " You are not eligible for
voting";
      }
    }
  </script>
</html>
```

BSc CsIt SEM - V

Web technology

```

else
{
    document.getElementById("demo").innerHTML = " You are eligible for voting";
}

}
</script>
</head>
<body>
<h2>If statement</h2>
<p>enter your age:</p>
<input id="age"/>
<button onclick="myFunction()">Click</button>
<p id="demo"></p>
</body>
</html>

```

Output:

If statement

enter your age:

If statement

enter your age:

You are eligible for voting

Switch Statement

You should use the Switch statement if you want to select one of many blocks of code to be executed.

Syntax

```

switch (expression)
{
    case label1:
        code to be executed if expression = label1
        break;
    case label2:
        code to be executed if expression = label2
        break;
    default:
        code to be executed if expression is different from both label1 and label2
}

```

First, we have a single expression (most often a variable), that is evaluated once. The value of the expression is then compared with the values for each case in the structure. If there is a match, the block of code associated with that case is executed. Use **break** to prevent the code from running into the next case automatically.

BSc CsIt SEM - V

Web technology

Example:

```
<!DOCTYPE html>
<html>
  <head>
<script>
  function myFunction()
  {
    let dayno = document.getElementById("day").value;
    switch(dayno)
    {
      case 1:
        document.getElementById("demo").innerHTML = " Sunday";
        break;
      case 2:
        document.getElementById("demo").innerHTML = " Monday";
        break;
      case 3:
        document.getElementById("demo").innerHTML = " Tuesday";
        break;

      case 4:
        document.getElementById("demo").innerHTML = " Wednesday";
        break;
      case 5:
        document.getElementById("demo").innerHTML = " Thursday";
        break;
      case 6:
        document.getElementById("demo").innerHTML = " Friday";
        break;
      case 7:
        document.getElementById("demo").innerHTML = " Saturday";
        break;
      default:
        document.getElementById("demo").innerHTML = " Invalid day";
    }
  }
</script>
  </head>
  <body>
<h2>switch case</h2>
<p>Enter day number:</p>
<input id="day"/>
<button onclick="myFunction()">Click</button>
<p id="demo"></p>
  </body>
</html>
```

BSc CsIt SEM - V

Web technology

switch case

Enter day number:

Invalid day

JavaScript Looping

Looping statements in JavaScript are used to execute the same block of code a specified number of times. Very often when you write code, you want the same block of code to run a number of times. You can use looping statements in your code to do this. In JavaScript we have the following looping statements:

- **while** - loops through a block of code while a condition is true
- **do...while** - loops through a block of code once, and then repeats the loop while a condition is true
- **for** - run statements a specified number of times

while loop

The while statement will execute a block of code while a condition is true

Syntax

```
while (condition)
{
    code to be executed
}
```

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <script>
      function myFunction()
      {
        let n=document.getElementById("day").value;
        var i=0;
        while (i<=n)
        {
          document.write("Iteration " + i + "<br>");
          i++;
        }
      }
    </script>
  </head>
</body>
```


BSc CsIt SEM - V

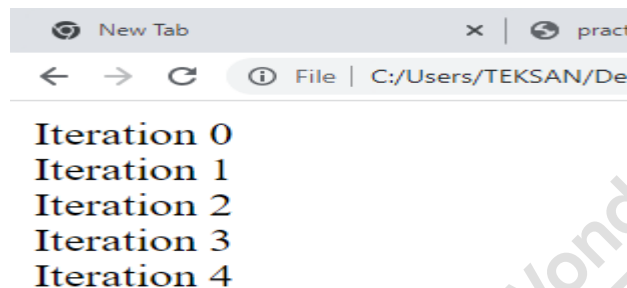
Web technology

```
<h2>whiile loop</h2>
<p>how many times you want to run the loop:</p>
<input id="day"/>
<button onclick="myFunction()">Click</button>
<p id="demo"></p>
</body>
</html>
```

Output:

while loop

how many times you want to run the loop:



do...while Loop

This statement will execute a block of code once, and then it will repeat the loop while a condition is true. Use a Do While loop to run the same block of code while or until a condition is true. This loop will always be executed once, even if the condition is false, because the statements are executed before the condition is tested

Syntax:

```
do
{
    code to be executed
} while (condition)
```

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <script>
      function myFunction()
      {
```

BSc CsIt SEM - V

Web technology

```

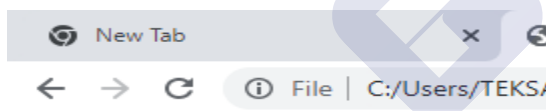
let n=document.getElementById("day").value;
var i=1;
do
{
    document.write("Iteration " + i + "<br>");
    i++;
} while (i<=n)
}
</script>
</head>
<body>
    <h2>Do while loop</h2>
    <p>how many times you want to run the loop:</p>
    <input id="day"/>
    <button onclick="myFunction()">Click</button>
    <p id="demo"></p>
</body>
</html>

```

Output:

Do while loop

how many times you want to run the loop:



Iteration 1
Iteration 2
Iteration 3

for Loop

The for statement will execute a block of code a specified number of times

Syntax:

```

for (initialization; condition; increment)
{
    code to be executed
}

```



BSc CsIt SEM - V

Web technology

Example

```
<!DOCTYPE html>
<html>
  <head>
    <script>
      function myFunction()
      {

        let n=document.getElementById("day").value;
        var i;
        for (i = 1; i <= n; i++)
        {
          document.write("Iteration " + i + "<br>");
        }

      }
    </script>
  </head>
  <body>
    <h2>Forloop</h2>
    <p>how many times you want to run the loop:</p>
    <input id="day"/>
    <button onclick="myFunction()">Click</button>
    <p id="demo"></p>
  </body>
</html>
```

Forloop

how many times you want to run the loop:

← → ↺ ⓘ File | C:/Users/TEKSAN,

Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5

BSc CsIt SEM - V

Web technology

Functions

A function contains some code that will be executed by an event or a call to that function. A function is a set of statements. You can reuse functions within the same script, or in other documents. You define functions at the beginning of a file (in the head section), and call them later in the document.

This is JavaScript's method to alert the user.

```
alert("This is a message"); // It is similar to msgbox in VB
```

How to Define a Function

To create a function, you define

- its name,
- any values ("arguments") and
- some statements:

A function with arguments and parentheses

```
function myfunction(argument1,argument2,etc)
{
    some statements
}
```

A function with no arguments and parentheses

```
function myfunction()
{
    some statements
}
```

Arguments are variables used in the function. The variable values are values passed on by the function call. By placing functions in the head section of the document, you make sure that all the code in the function has been loaded before the function is called. Some functions return a value to the calling expression

```
function result(a,b)
{
    c=a+b;
    return c;
}
```



BSc CsIt SEM - V

Web technology

How to Call a Function?

A function is not executed before it is called. You can call a function containing arguments or without arguments as below:

1. myfunction(*argument1,argument2,etc*)
2. myfunction()

The return Statement

Functions that will return a result must use the "return" statement. This statement specifies the value which will be returned to where the function was called from. Say you have a function that returns the sum of two numbers:

```
function total(a,b)
{
    result=a+b;
    return result;
}
```

When you call this function you must send two arguments with it; sum=total(2,3) The returned value from the function **that is 5** will be stored in the variable called sum.

How to call a function.

```
<!DOCTYPE html>
<html>
  <head>
    <script>
      function myfunction()
      {
        alert("HELLO") ;
      }
    </script>
  </head>
  <body>
    <h2>Calling function</h2>
    <form>
      <input type="button" onclick="myfunction()" value="Call function">
    </form>
    <p>By pressing the button, a function will be called. The function will alert a
message.</p>
  </body>
</html>
```

BSc CsIt SEM - V

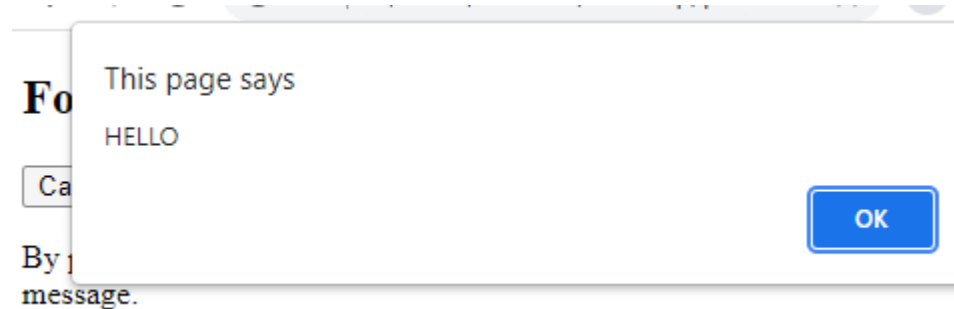
Web technology

Output:

5 Forloop

Call function

By pressing the button, a function will be called. The function will alert a message.



Function with argument:

```
<!DOCTYPE html>
<html>
  <head>
    <script>

      function myfunction(txt)
      {
        alert(txt);
      }
    </script>
  </head>
  <body>
    <h2>Calling function</h2>
    <form>
      <input type="button" onclick="myfunction('Good Morning!')" value="In the morning">
      <input type="button" onclick="myfunction('Good Evening!')" value="In the Evening">
    </form>

    <p>By pressing the button, a function with an argument will be called. That will alert
    this argument.</p>

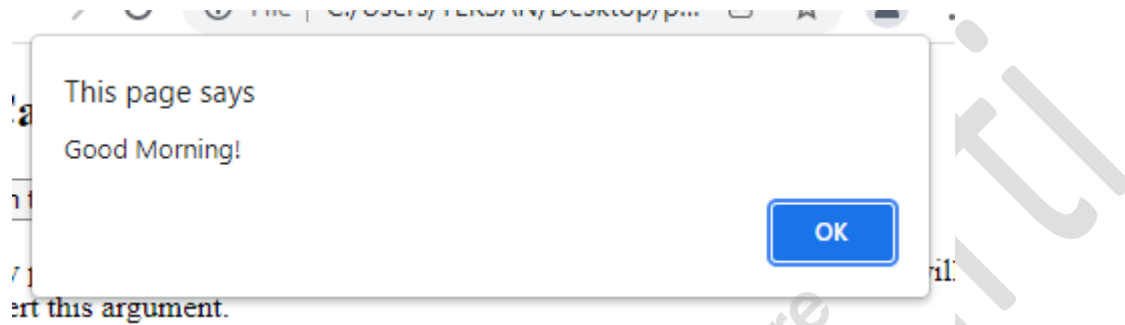
  </body>
</html>
```

BSc CsIt SEM - V

Web technology

Calling function

By pressing the button, a function with an argument will be called. That will alert this argument.



Function that returns a value:

```
<!DOCTYPE html>
<html>
  <head>
    <script>

      function myFunction()
      {
        return ("Hello, have a nice day!");
      }

    </script>
  </head>
  <body>
    <h2>Calling function</h2>
    <script>
      document.write(myFunction());
    </script>
    <p>The script in the body section calls a function.<br>The function returns a text.</p>

  </body>
</html>
```

BSc CsIt SEM - V

Web technology

Calling function

Hello, have a nice day!

The script in the body section calls a function.
The function returns a text.

A function with arguments, that returns a value:

```
<!DOCTYPE html>
<html>
  <head>
    <script>
      function total(numberA,numberB)
      {
        return numberA + numberB;
      }
    </script>
  </head>
  <body>
    <h2>Calling function</h2>
    <script>
      document.write(total(2,3));
    </script>
    <p>The script in the body section calls a function with two arguments, 2 and 3.</p>
    <p>The function returns the sum of these two arguments.</p>
  </body>
</html>
```

← → ↻ ⓘ File | C:/Users/TEKSAN/Desktop/p... ⌂ ☆ 👤 ⋮

Calling function

5

The script in the body section calls a function with two arguments, 2 and 3.

The function returns the sum of these two arguments.

BSc CsIt SEM - V

Web technology

Form Validation and Error Handling

Steps to be followed while validating form fields

1. First make form with appropriate id and values
2. Retrieve value of form's fields inside script tag.
3. Validate for empty field i.e., check for empty field
4. validate for radio, checkbox and select button
5. validate for condition like username should be between 5-20, password should be greater than 5 etc.

Following example shows complete form validation:

- checking for emptiness of text box
- checking for selection of radio, checkbox and drop-down menu
- validation of username: username should be between 3 to 25 character
- validation of password: password should be greater than 5 and password and repass word should be same
- validation of email: position of @ should not be on first and after position of '.' There should be only 2 to 3 character.
- Phone number validation: there should not be any character and there should be exactly 10 digits

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>form validation</title>
  <script type="text/javascript">
    function validate(){
      //retriving value of form field
      var username = document.getElementById("uname").value;
      var password = document.getElementById("pass").value;
      var repassword = document.getElementById("repass").value;
      var email = document.getElementById("email").value;
      var phone = document.getElementById("ph").value;
      var rb1 = document.getElementById("rb1");
      var rb2 = document.getElementById("rb2");
      var j1 = document.getElementById("j1");
      var j2 = document.getElementById("j2");
      var country = document.getElementById("country").value;
      //cheking for empty field also known as basic validation
      //checking for emptiness of text box
```

BSc CsIt SEM - V

Web technology

```

if(username.trim()==""||password.trim()==""||repassword.trim()==""||email.trim()
()==""||phone.trim()=="){
    alert("username or password or repassword or email or phone number
field is emptyl. please fill");
    return false;
}
//checking for emptiness of radio button
else if(rb1.checked==false && rb2.checked==false){
    alert("plz select one option");
    return false;

}
//checking for emptiness of checkbox.
else if(j1.checked==false && j2.checked==false){
    alert("plz selct one checkbx");
    return false;
}
else if(country=="-1"){
    alert("please select any option");
    return false;

}
//checking for condition:
//username should be between 3 to 25 character
else if(username.trim().length<3||username.trim().length>25){
    alert("username should be between 3 to 25 character");
    return false;
}
//password should be grater than 5 and password and repassword should be
same
else if(password.trim().length<5){
    alert("password too short");
    return false;

}
else if(password!=repassword){
    alert("password and repassword doesnot match");
    return false;
}
//email validation: position of @ should not be on first and after "." only two or
//three letter should be present
//eg: csit@gmail.com

```

BSc CsIt SEM - V**Web technology**

```

else if(email.indexOf('@')<=0||email.charAt(email.length-4)!='.'){
    alert("please insert in correct format");
    return false;

}
//phone number validation: there should not be any character and length should
//be equal to 10
else if(isNaN(phone)){
    alert("characters are not allowed only numbers are allowed for phone
number");
    return false;

}
else if(phone.length!=10){
    alert("length of phone number should be exactly 10 digits");
    return false;
}
else{
    alert("all are correct. form submitted");
    return true;
}
}

```

```

</script>
</head>
<body>
    <h1> making form and validating it for empty field and for some
condition</h1>
    <form onsubmit="return validate();" action="">
        Username:<input type="text" id="uname" placeholder="Username"/><br>
        Password:<input type="password" id="pass"
placeholder="password"/><br>
        Re-Password: <input type="password" id="repass"
placeholder="repassword"/><br>
        Email: <input type="text" id="email" placeholder="email"/><br>
        Phone:<input type="text" id="ph" placeholder="phone number"/><br>
        Gender: <input type="radio" name="gender" value="male" id="rb1"/>Male
        &nbsp;
        <input type="radio" name="gender" value="female" id="rb2"/>Female
    <br/>
        course:<input type="checkbox" name="course" value="java" id="j1"/>Java
        &nbsp;

```

BSc CsIt SEM - V

Web technology

```
<input type="checkbox" name="course" value="C" id="j2"/>C<br/>
```

```
Select country: <select id="country">
```

```
  <option value="-1">Select here</option>
```

```
  <option value="Nepal">Nepal</option>
```

```
  <option value="USA">USA</option>
```

```
  <option value="Canada">Canada</option>
```

```
</select>
```

```
<br/>
```

```
<input type="submit" value="submit"/>
```

```
</form>
```

```
</body>
```

```
</html>
```

Output:

If any text field is empty:

← → ↻ ⓘ File | E:/form/formval.html?gender=male&course=java

making form and validating it for empty

Username:

Password:

Re-Password:

Email:

Phone:

Gender: ☒ Male ☐ Female

course: ☒ Java ☐ C

Select country:

This page says
username or password or repassword or email or phone number field is empty. please fill

OK

If no any option is selected from radio button:

← → ↻ ⓘ File | E:/form/formval.html

making form and validating it for empty

Username:

Password:

Re-Password:

Email:

Phone:

Gender: ☐ Male ☐ Female

course: ☒ Java ☐ C

Select country:

This page says
plz select one option of radio button

OK

If no any option is selected from checkbox:

BSc CsIt SEM - V

Web technology

← → ↻ ⓘ File | E:/form/formval.html

making form and validating it for empty

Username:
Password:
Re-Password:
Email:
Phone:
Gender: ☒ Male ☐ Female
course: ☐ Java ☐ C
Select country:

This page says
plz select one checkbox

OK

If no any option is selected from drop down menu

← → ↻ ⓘ File | E:/form/formval.html

making form and validating it for empty

Username:
Password:
Re-Password:
Email:
Phone:
Gender: ☒ Male ☐ Female
course: ☒ Java ☒ C
Select country:

This page says
please select any option

OK

If username is not between 3 to 25 character

← → ↻ ⓘ File | E:/form/formval.html

making form and validating it for empty

Username:
Password:
Re-Password:
Email:
Phone:
Gender: ☒ Male ☐ Female
course: ☒ Java ☒ C
Select country:

This page says
username should be between 3 to 25 character

OK

If password is not greater than 5

BSc CsIt SEM - V

Web technology

← → ↻ ⓘ File | E:/form/formval.html

making form and validating it for empty

Username:
 Password:
 Re-Password:
 Email:
 Phone:
 Gender: ☒ Male ☐ Female
 course: ☒ Java ☒ C
 Select country:

This page says
password too short

OK

If password and re-password is not same

← → ↻ ⓘ File | E:/form/formval.html

making form and validating it for empty

Username:
 Password:
 Re-Password:
 Email:
 Phone:
 Gender: ☒ Male ☐ Female
 course: ☒ Java ☒ C
 Select country:

This page says
password and repassword doesnot match

OK

If email is not on exact format:

← → ↻ ⓘ File | E:/form/formval.html

making form and validating it for empty

Username:
 Password:
 Re-Password:
 Email:
 Phone:
 Gender: ☒ Male ☐ Female
 course: ☒ Java ☒ C
 Select country:

This page says
please insert in correct format

OK

If phone number is not number

← → ↻ ⓘ File | E:/form/formval.html

making form and validating it for empty

Username:
 Password:
 Re-Password:
 Email:
 Phone:
 Gender: ☒ Male ☐ Female
 course: ☒ Java ☒ C
 Select country:

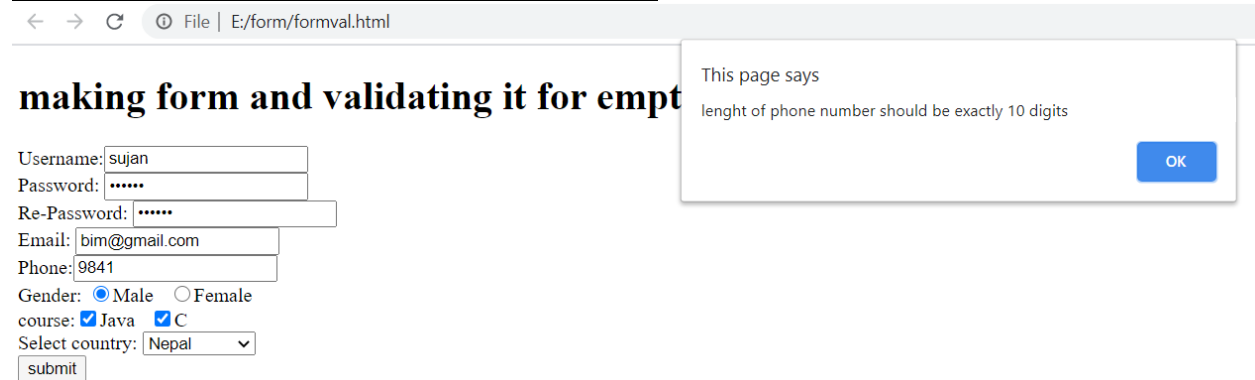
This page says
characters are not allowed only numbers are allowed for phone number

OK

BSc CsIt SEM - V

Web technology

If phone number is not equal to 10 digits



The screenshot shows a web browser window with the address bar displaying "E:/form/formval.html". The page title is "making form and validating it for empty". The form contains the following fields: Username (sujan), Password (masked with dots), Re-Password (masked with dots), Email (bim@gmail.com), Phone (9841), Gender (Male selected, Female unselected), course (Java selected, C unselected), and Select country (Nepal). A submit button is at the bottom. A validation error message box is displayed, stating "This page says" and "length of phone number should be exactly 10 digits". An OK button is present in the message box.

If all fields are in correct format:

Popup Boxes: Alert, Confirm, Prompt;

JavaScript offers built-in global functions that are used to display messages for a variety of purposes. The various popup boxes that are available in JavaScript can be used to notify, warn users as well as to display a simple message or take the user's confirmation or user's input. Popup boxes aren't utilized excessively as they halt other parts of the program until the pop-up is closed.

There are three different popup boxes available in JavaScript; alert, Prompt, and Confirm Box and we will explore each of the popup boxes in this post.

Alert box

The alert box is a box shown at the top center of the browser window and it is mainly used to show warnings to the user. Other parts of the program would stop executing unless the user closes the alert box. To close the alert popup box, simply click on the ok button present in the alert box.

Syntax:

```
alert("Warning");
```

BSc CsIt SEM - V

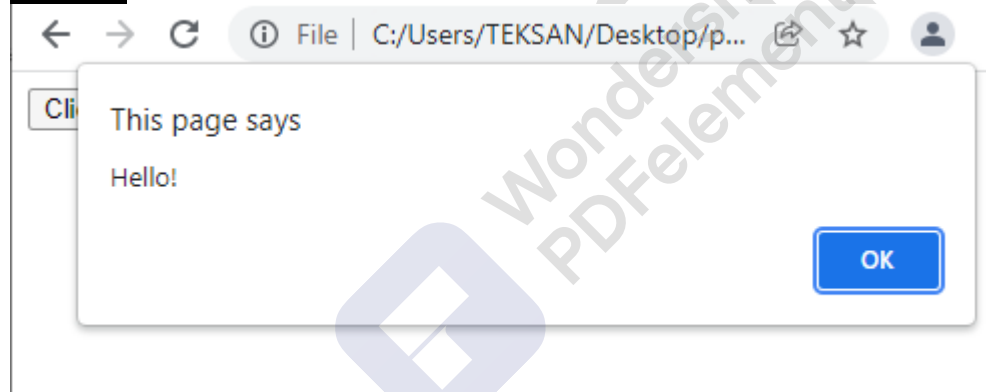
Web technology

Example:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Alert Example</title>
  </head>

  <body>
    <button onclick="myFunc()">Click Me</button>
    <script>
      function myFunc()
      {
        alert("Hello!");
      }
    </script>
  </body>
</html>
```

Output



Prompt Box

The prompt box is mainly used to get input from a user and appears at the top center of the browser when invoked. The input data is also available after we close the prompt box and if you leave the input field in the prompt box empty then it will return a null value.

Syntax:

```
prompt("Prompt Message Here");
```

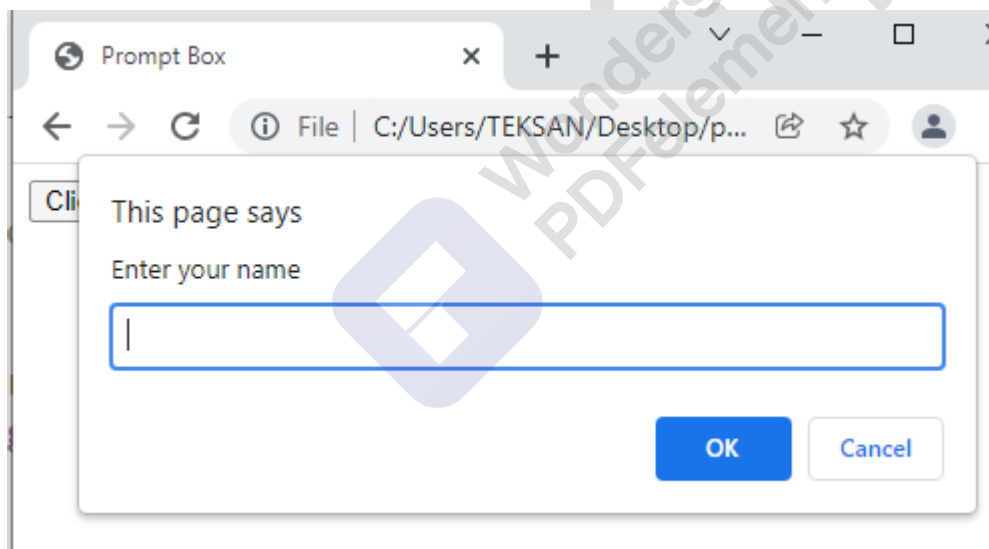

BSc CsIt SEM - V

Web technology

Example

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Prompt Box</title>
    <script>
      // function to show alert box
      function myFunc()
      {
        var name=prompt("Enter your name");
        console.log(name);
      }
    </script>
  </head>
  <body>
    <button onclick="myFunc()">Click Me</button>
  </body>
</html>
```

Output:



Confirm Box

A confirm box can be used for the various purposes like verification or acceptance of some task. When a confirm box pops up, the user is given an option to accept or cancel it according to the message written in the confirm box.

Similar to the prompt box, if the user clicks on OK, the box returns true else the box returns false. A confirm box can be made active by providing confirm() method for any click or action in a web browser.

BSc CsIt SEM - V

Web technology

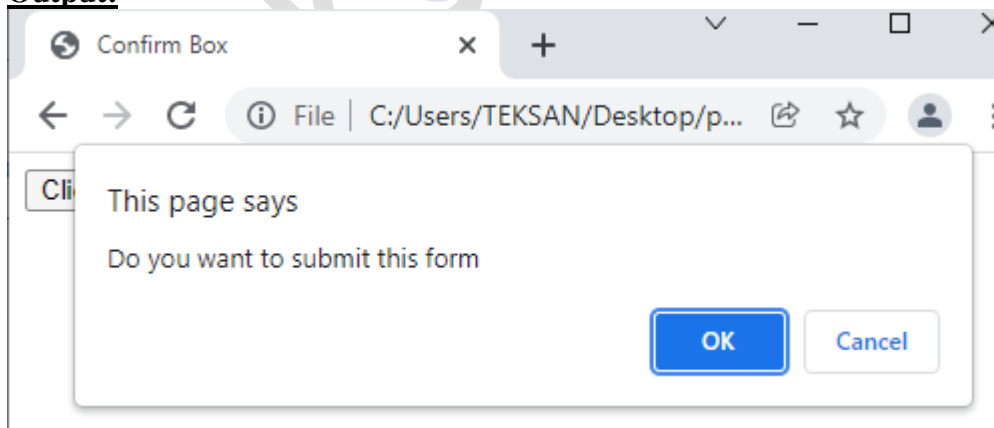
Syntax:

```
confirm("Permission goes here");
```

Example:

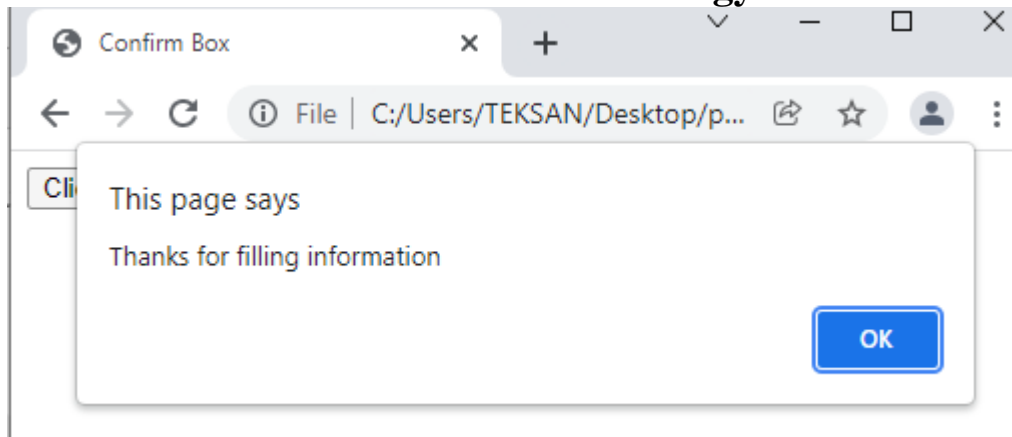
```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Confirm Box</title>
    <script>
      // function to show alert box
      function myFunc()
      {
        var myValue;
        // check whether user clicks on the ok button or cancel
        if (confirm("To proceed press a button!") == true)
        {
          myValue = "OK pressed!";
        }
        else {
          myValue = "Cancel!";
        }
        console.log(myValue);
      }
    </script>
  </head>
  <body>
    <button onclick="myFunc()">Click Me</button>
  </body>
</html>
```

Output:



BSc CsIt SEM - V

Web technology



Objects and properties

An object is a collection of properties, and a property is an association of name (or key) and a value. A property's value can be a function i.e. method. In addition to objects that are predefined, you can define your own objects.

Objects in JavaScript, just as in many other programming languages, can be compared to objects in real life. The concept of objects in JavaScript can be understood with real life, tangible objects.

In JavaScript, an object is a standalone entity, with properties and type. Compare it with a cup, for example. A cup is an object, with properties. A cup has a color, a design, weight, a material it is made of, etc. The same way, JavaScript objects can have properties, which define their characteristics.

So, JavaScript object is a non-primitive data-type (Number, String, Boolean, null, undefined and symbol) that allows you to store multiple collections of data.

Loosely speaking, objects in JavaScript may be defined as an unordered collection of related data, of primitive or reference types, in the form of “key: value” pairs. These keys can be variables or functions and are called properties and methods, respectively, in the context of an object.

Here is an example of a JavaScript object.

```
// object
const student =
{
  firstName: 'ram',
  class: 10
};
```

Here, student is an object that stores values such as strings and numbers.

BSc CsIt SEM - V

Web technology

In JavaScript, an object can be created in two ways:

- 1) using Object Literal
- 2) using the Object() Constructor function with the new keyword.

Create Object using Object Literal

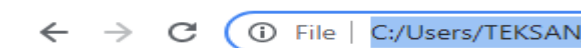
Syntax:

```
Var <object-name> = {  
    key1: value1,  
    key2: value2,  
    .....  
};
```

Example:

```
<!DOCTYPE html>  
<html>  
  <body>  
  
    <h2>JavaScript Objects</h2>  
  
    <p id="p1"></p>  
    <p id="p2"></p>  
    <p id="p3"></p>  
  
    <script>  
      // Create an object:  
      const person = { name:"lalit", address:"ktm", age:20};  
  
      // Display data from the object:  
      document.getElementById("p1").innerHTML = "name of person " + person.name;  
      document.getElementById("p2").innerHTML = "Address " + person.address;  
      document.getElementById("p3").innerHTML = "Age " + person.age;  
    </script>  
  
  </body>  
</html>
```

Output:



JavaScript Objects

name of person lalit

Address ktm

Age 20

BSc CsIt SEM - V

Web technology

Using the Object() Constructor function with the new keyword.

Example:

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
  <h1>JavaScript Object</h1>
  <p id="p1"></p>
  <p id="p2"></p>
  <p id="p3"></p>

  <script>
    var person = new Object();

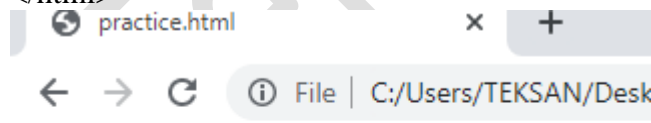
    // Attach properties and methods to person object
    person.firstName = "lalit";
    person.lastName = "singh";
    person.age = 25;

    person.getFullName = function ()
    {
      return this.firstName + ' ' + this.lastName;
    };

    // access properties & methods
    document.getElementById("p1").innerHTML = person.firstName;
    document.getElementById("p2").innerHTML = person.lastName;

    document.getElementById("p3").innerHTML = person.getFullName();

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



JavaScript Object

lalit

singh

lalit singh

BSc CsIt SEM - V

Web technology

Example2

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <h1>JavaScript constructor</h1>
    <script>
      function Person ()
      {
        this.name = 'lalit',
        this.age = 23,
        this.greet = function ()
        {
          console.log('hello');
        }
      }

      // create objects
      var person1 = new Person();
      // access properties
      console.log(person1.name);
    </script>
  </body>
</html>
```

Constructors

A constructor is a special method of a class that creates an instance of a class which is typically called an “object”. In JavaScript, a constructor gets called when you declare an object using the new keyword.

The purpose of a constructor is to create an object and set values if there are any object properties present.

Syntax

```
constructor([arguments])
{
  ...
}
```

BSc CsIt SEM - V

Web technology

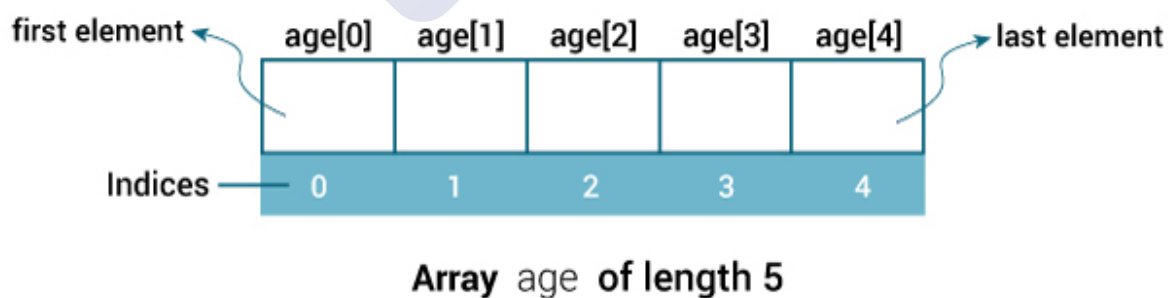
Example1

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <h1>JavaScript constructor</h1>
    <script>
      class Employee
      {
        constructor(id, name)
        {
          this.id = id;
          this.name = name;
        }
      }
      var emp1 = new Employee(1,"Lalit");
      console.log(emp1.name);
    </script>
  </body>
</html>
```

Arrays

As we know that a variable can hold only one value, for example `var i = 1`, we can assign only one literal value to `i`. We cannot assign multiple literal values to a variable `i`. To overcome this problem, JavaScript provides an array.

An array is a special type of variable, which can store multiple values using special syntax. Every value is associated with numeric index starting with 0. The following figure illustrates how an array stores value.



JavaScript array can store multiple elements of different data types as given below...

Value →	12	"Hello"	"world"	34	90	2.3	2	3	87
Index →	0	1	2	3	4	5	6	7	8

It is not required to store value of same data type in an array.

BSc CsIt SEM - V

Web technology

Array Initialization

An array in JavaScript can be defined and initialized in two ways..

- 1) Array literal
- 2) Array constructor syntax.

Array Literal

Array literal syntax takes a list of values separated by a comma and enclosed in square brackets.

Syntax:

var <array-name> = [element0, element1, element2,... elementN];

Example:

```
<!DOCTYPE html>
<html>
<body>
  <h1>JavaScript Array</h1>
  <p id="p1"></p>
  <p id="p2"></p>
  <p id="p3"></p>
  <p id="p4"></p>
  <p id="p5"></p>

  <script>
    var stringArray = ["one", "two", "three"];
    var numericArray = [1, 2, 3, 4];
    var decimalArray = [1.1, 1.2, 1.3];
    var booleanArray = [true, false, false, true];
    var mixedArray = [1, "two", "three", 4];
    document.getElementById("p1").innerHTML = stringArray;
    document.getElementById("p2").innerHTML = numericArray;
    document.getElementById("p3").innerHTML = decimalArray;
    document.getElementById("p4").innerHTML = booleanArray;
    document.getElementById("p5").innerHTML = mixedArray;
  </script>
</body>
</html>
```

Output:

BSc CsIt SEM - V

Web technology

JavaScript Array

one,two,three

1,2,3,4

1.1,1.2,1.3

true,false,false,true

1,two,three,4

Array Constructor

You can initialize an array with Array constructor syntax using new keyword. The Array constructor has following three forms.

Syntax:

- 1) **var arrayName = new Array();**
- 2) **var arrayName = new Array(Number length);**
- 3) **var arrayName = new Array(element1, element2, element3,... elementN);**

As you can see in the above syntax, an array can be initialized using new keyword, in the same way as an object.

Example:

```
<!DOCTYPE html>
<html>
<body>
  <h1>JavaScript Array Object</h1>
  <p id="p1"></p>
  <p id="p2"></p>
  <p id="p3"></p>

  <script>
    var stringArray = new Array();
    stringArray[0] = "one";
    stringArray[1] = "two";
    stringArray[2] = "three";
    stringArray[3] = "four";

    var numericArray = new Array(3);
    numericArray[0] = 1;
    numericArray[1] = 2;
    numericArray[2] = 3;
```

BSc CsIt SEM - V

Web technology

```
var mixedArray = new Array(1, "two", 3, "four");
```

```
document.getElementById("p1").innerHTML = stringArray;  
document.getElementById("p2").innerHTML = numericArray;  
document.getElementById("p3").innerHTML = mixedArray;
```

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

Output:



Built-in Objects:

The object model of JavaScript language forms the foundation of the language. These objects help to provide custom functionalities in the script.

JavaScript has several built-in or core language objects. These objects provide some of the core functionality for working with text, numbers, collections of data, dates, and a whole lot more.

JavaScript treats the primitive data types as objects and provide equivalent object for each of them. For example, if a variable contains a string of characters, then it is treated as String object by JavaScript. Similarly, if a variable contains the value true or false, it is treated as Boolean object.

JavaScript objects are categorized as **built-in objects**, **browser objects**, and **HTML objects**.

- The built-in objects are Static objects Which Can be used to extend the functionality in the script. Some of these objects are: **String, Math, and Date**.
- The browser objects, such as **window**, history, and navigator are used to work with the browser window,
- Whereas the HTML objects, such as **form, anchor**, and so on are used to access elements on a Web page.

BSc CsIt SEM - V

Web technology

Window Object

JavaScript Window Object is global object which represents the currently opened window tab in the browser. It can be said that the window object is nothing but the browser window which is opened in the browser. This Window object is different from the normal object, a window is an object of browser. Every tab opened in the browser will have its window object associated with it and it will be available in every part of the application as it is global. It has multiple values and methods are available with it and we can call these methods directly to perform multiple operations over the browser window.

The window object allows developers to perform tasks such as opening and closing browser windows, displaying alert and prompt dialogs and setting up timeouts (specifying an action to take place after a specified period of time).

All global JavaScript objects, functions, and variables automatically become members of the window object. Global variables are properties of the window object and all global functions are methods of the window object. Even the document object is a property of the window object, `window.document.getElementById("p1")`, is the same as `document.getElementById("p1")`.

Methods of window object

<u>Method</u>	<u>Description</u>
<ul style="list-style-type: none">• <code>alert()</code>	<ul style="list-style-type: none">• displays the alert box containing message with ok button.
<ul style="list-style-type: none">• <code>confirm()</code>	<ul style="list-style-type: none">• displays the confirm dialog box containing message with ok and cancel button.
<ul style="list-style-type: none">• <code>prompt()</code>	<ul style="list-style-type: none">• displays a dialog box to get input from the user.
<ul style="list-style-type: none">• <code>open()</code>	<ul style="list-style-type: none">• opens the new window.
<ul style="list-style-type: none">• <code>close()</code>	<ul style="list-style-type: none">• closes the current window.
<ul style="list-style-type: none">• <code>setTimeout()</code>	<ul style="list-style-type: none">• performs action after specified time like calling function, evaluating expressions etc.

BSc CsIt SEM - V

Web technology

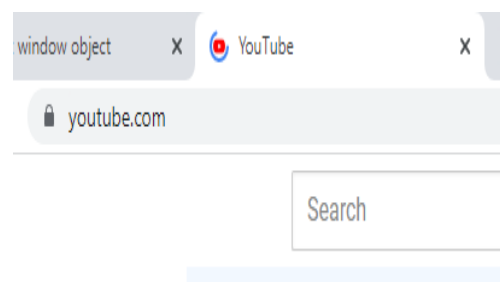
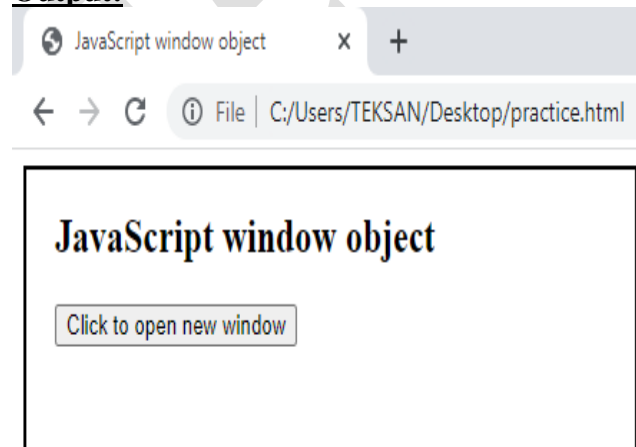
open() Method:

This method will open the new window with the specified URL address.

Code:

```
<!DOCTYPE html>
<html>
  <head>
    <title>JavaScript window object</title>
    <style>
      .div1
      {
        border :black 2px solid;
        text-align : left;
        padding-left : 20px;
        height : 140px;
        width : 30%;
      }
    </style>
  </head>
  <body>
    <div class = "div1">
      <h2> JavaScript window object </h2>
      <input type = "button" value = "Click to open new window" onclick =
      "openTheWindow()">
    </div>
    <script type = "text/javascript">
      function openTheWindow()
      {
        open("https://www.youtube.com/");
      }
    </script>
  </body>
</html>
```

Output:





BSc CsIt SEM - V

Web technology

close() Method

This method will close the window which is opened by the Window object.

Code:

```
<!DOCTYPE html>
<html>
  <head>
    <title> JavaScript window object </title>
    <style>
      .div1
      {
        border : black 2px solid;
        text-align : left;
        padding-left : 20px;
        height : 140px;
        width : 30%;
      }
    </style>
  </head>
  <body>
    <div class = "div">
      <h2> JavaScript window object </h2>
      <input type = "button" value = "open window" onclick =
        "openTheWindow()">
      <br> <br>
      <input type = "button" value = "close window" onclick =
        "closeTheWindow()">
    </div>
    <script type = "text/javascript">
      let a;
      function openTheWindow()
      {
        a = open("https://www.youtube.com/");
      }
      function closeTheWindow()
      {
        a.close();
      }
    </script>
  </body>
</html>
Output:
```

BSc CsIt SEM - V

Web technology

setTimeout() Method

This method will perform the action specified after a time interval which is specified while calling it. We can perform any kind of activity in this way.

Code:

```
<!DOCTYPE html>
<html>
  <head>
    <title> JavaScript window object </title>
    <style>
      .div1
      {
        border : black 2px solid;
        text-align : left;
        padding-left : 20px;
        height : 140px;
        width : 30%;
      }
    </style>
  </head>
  <body>
    <div class = "div1">
      <h2> JavaScript window object </h2>
      <input type = "button" value = "open new window" onclick =
        "openTheWindow()">
      <br> <br>
      <input type = "button" value = "close window after some time
        automatically" onclick = "executeTimeout()">
    </div>

    <script type = "text/javascript">
      let a;
      function openTheWindow()
      {
        a= open("https://www.youtube.com/");
      }
      function closeTheWindow()
      {
        a.close();
      }
      function executeTimeout()
      {
        setTimeout(function()
          {
            closeTheWindow()
          },2500);
      }
    </script>
  </body>
</html>
```

BSc CsIt SEM - V

Web technology

JavaScript Window Object Properties

The window object properties refer to the variables created inside the window object. In JavaScript, all the available data is attached to the window object as properties. We can access window object's properties like:

window.propertyname

Here propertyname is the name of property.

Some window object properties :

<u>Property</u>	<u>Description</u>
closed	returns a boolean value that specifies whether a window has been closed or not
document	specifies a document object in the window.
innerHeight	specifies the inner height of the window's content area.
innerWidth	specifies the inner width of the window's content area.

Document Object

When an HTML document is loaded into a web browser, it becomes a document object. The document object is a property of the window object. The document object is accessed with:

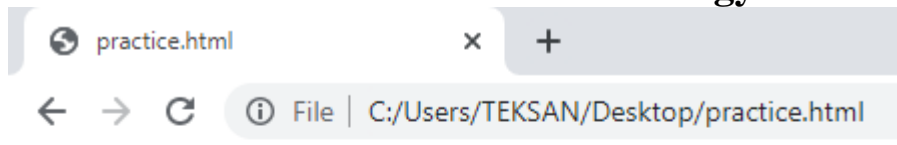
window.document or just document

Example:

```
<!DOCTYPE html>
<html>
  <head></head>
  <body>
    <h1>Window Document Object</h1>
    <h2>The doc property</h2>
    <p id="demo"></p>
    <script>
      let url = window.document.URL;
      document.getElementById("demo").innerHTML = url;
    </script>
  </body>
</html>
```

BSc CsIt SEM - V

Web technology



Window Document Object.

The doc property

file:///C:/Users/TEKSAN/Desktop/practice.html

Example: document,close,inner width and height properties

```
<!DOCTYPE html>
<html>
<body>

<h1>Window Document Object.</h1>
<h2>The doc property</h2>
<p id="demo"></p>

<h2>The closed Property</h2>
<p><button onclick="openTheWindow()">Open</button></p>
<p><button onclick="closeTheWindow()">Close</button></p>

<h2>The innerWidth and innerHeight Properties</h2>
<p id="innerwh"></p>

<script>
//document property
let url = window.document.URL;
document.getElementById("demo").innerHTML = url;

//close property
let a;
function openTheWindow()
{
    a= open("https://www.youtube.com/");
}
function closeTheWindow()
{
    a.close();
}

// inner height property
let w = window.innerWidth;
```

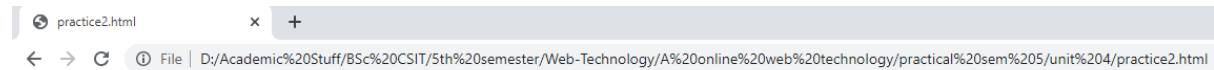

BSc CsIt SEM - V

Web technology

```
let h = window.innerHeight;
document.getElementById("innerwh").innerHTML = "Width: " + w + "<br>Height: " + h;

</script>

</body>
</html>
```



Window Document Object.

The doc property

file:///D:/Academic%20Stuff/BSc%20CSIT/5th%20semester/Web-Technology/A%20online%20web%20technology/practical%20sem%205/unit%204/practice2.html

The closed Property

Open

Close

The innerWidth and innerHeight Properties

Width: 1366
Height: 657

String object

The String object is used to represent and manipulate a sequence of characters. A string is a combination of letters, numbers, special characters (single or double quotations), and arithmetic values. Strings can be created as primitives, from string literals, or as objects, using the String() constructor as follows.

```
const string1 = "A string primitive";
const string4 = new String ("A String object");
```

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <title>String</title>
  </head>
  <body>
    <h2>String Object </h2>
    <script type="text/javascript">
      var msg = new String("String example");
      document.write(msg);

      //comparisons of literal and objects
      document.write("<br>");
      var str1 ="String"; //literal
```

BSc CsIt SEM - V

Web technology

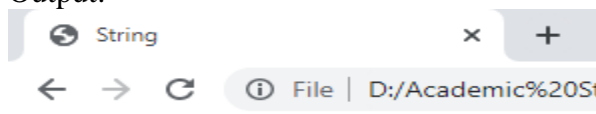
```
var str2 = new String("String"); //object
var str3 = new String("String"); //object
```

```
// comparisons value
(str1==str2)?document.write("Equal"):document.write("Not equal");
document.write("<br>");
```

```
//The comparison of two object
(str2==str3)?document.write("Equal"):document.write("Not equal");
```

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

Output:



String Object

String example
Equal
Not equal

Number Object

The JavaScript Number object represents numerical data which can be integers or floating-point numbers.

We create a Number object using the Number function by writing the following code:

```
new Number('123');
const a = new Number('123');
const b = Number('123');
```

Examble

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Number</title>
  </head>
  <body>
    <h2>Number object Object </h2>
    <script type="text/javascript">
      var num1=new Number(102); //integer value
      var num2=Number(102.7); //Floating point value
      var num3=Number(13e4); //Exponent value
```

BSc CsIt SEM - V

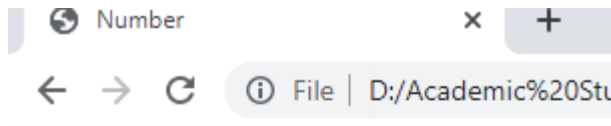
Web technology

```

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

```

Output:



Number object Object

Integer Value: 102
 Floating point Value: 102.7
 Exponent Value: 130000

Boolean Object

JavaScript Boolean object is a member of global objects and a wrapper class. It is used to create a boolean object which holds true or false value, depending upon the value used while creating the Boolean object.

The Boolean object's true and false values are different from the primitive boolean type's true and false values. The default value of JavaScript Boolean object is false.

You can create the JavaScript Boolean object by Boolean () constructor as given below.

Boolean b=new Boolean(value);

Example:

```

<!DOCTYPE HTML>
<html>
  <head>
    <title>Boolean Object </title>
    <style> </style>
  </head>

  <body>
    <script>
      var valueFalse = false;
      var output = valueFalse && true;
      document.write(output + "<br>");

      var notBoolean = new Boolean(false);
      output = notBoolean && true;
      document.write(output + "<br>");
    </script>
  </body>
</html>

```

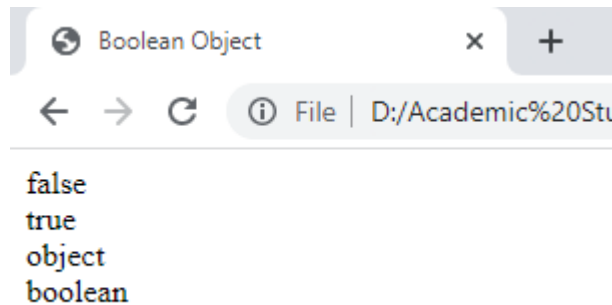
BSc CsIt SEM - V

Web technology

```
document.write(typeof notBoolean + "<br>");
document.write(typeof valueFalse + "<br>");
```

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

Output



Date object

JavaScript Date Object is built-in object that is used to get date and time of user device. Using JavaScript date object, we can detect date, time, timezone, day and also change date time data using js date function.

JavaScript will pick date and time from user machine. The accuracy of date and time totally depends upon device.

You can create an instance of the Date object with the "new" keyword. To store the current date in a variable called "my_date":

```
var my_date=new Date()
```

After creating an instance of the Date object, you can access all the methods of the object from the "my_date" variable. If, for example, you want to return the date (from 1-31) of a Date object, you should write the following:

```
my_date.getDate()
```

There are four different way to declare a date, the basic things is that the date objects are created by the new Date() operator.

Syntax:

1. **new Date():** It creates a new date object with the current date and time:
2. **new Date(milliseconds):** It creates a new date object as zero time plus milliseconds
3. **new Date(dataString):** creates a new date object from a date string
4. **new Date(year, month, date, hour, minute, second, millisecond) :** It creates a new date object with a specified date and time having numbers specify year, month, day, hour, minute, second, and millisecond (in that order).

BSc CsIt SEM - V

Web technology

Example:

```
<!DOCTYPE html>
<html>
  <head> </head>
  <body>
    <h2>JavaScript date object</h2>
    <p> <b>Using new Date():</b> It creates a new date object with the
    current date and time</p>
    <p id="dateob"></p>

    <p><b>Using new Date(milliseconds):</b> creates a new date object
    as January 1, 1970, 00:00:00 Universal Time (UTC) plus the
    milliseconds:</p>
    <p><b>Adding zero milisecond:</b> </p>

    <p id="milliseconds"></p>

    <p><b>Adding 100000000000 milliseconds:</b></p>
    <p id="milliseconds2"></p>

    <p><b>Using new Date(dataString):</b> creates a new date object
    from a date string:</p>
    <p id="dataString"></p>

    <p><b>Using new Date(year, month, date, hour, minute, second,
    millisecond) :</b></p>
    <p id="7number"></p>

    <script>
      // using new Date()
      const d = new Date();
      document.getElementById("dateob").innerHTML =d;

      // Using new Date(milliseconds)
      const d1 = new Date(0);
      document.getElementById("milliseconds").innerHTML =d1;

      const d2 = new Date(1000000000000);
      document.getElementById("milliseconds2").innerHTML =d2;

      // dataString
      const d3 = new Date("October 1, 2020 11:13:00");
      document.getElementById("dataString").innerHTML =d3;

      // using new Date(year, month, date, hour, minute, second,
      millisecond) :
```

BSc CsIt SEM - V

Web technology

```

const d4 = new Date(2020, 11, 24, 10, 33, 30, 0);
document.getElementById("7number").innerHTML = d4;
</script>
</body>
</html>

```

Output:

JavaScript date object

Using **new Date()**: It creates a new date object with the current date and time

Mon Dec 20 2021 12:10:22 GMT+0545 (Nepal Time)

Using **new Date(milliseconds)**: creates a new date object as January 1, 1970, 00:00:00 Universal Time (UTC) plus the milliseconds:

Adding zero milisecond:

Thu Jan 01 1970 05:30:00 GMT+0530 (Nepal Time)

Adding 100000000000 milliseconds:

Sat Mar 03 1973 15:16:40 GMT+0530 (Nepal Time)

Using **new Date(dataString)**: creates a new date object from a date string:

Thu Oct 01 2020 11:13:00 GMT+0545 (Nepal Time)

Using **new Date(year, month, date, hour, minute, second, millisecond)**:

Thu Dec 24 2020 10:33:30 GMT+0545 (Nepal Time)

Example2:

```

<!DOCTYPE html>
<html>
<body>
  <h2>JavaScript date object</h2>
  <script>
    var d = new Date()
    document.write(d.getDate()+"-"+(d.getMonth()+1)+"-"+d.getFullYear());
    document.write("<br>");
    document.write(d.getHours()+":"+d.getMinutes()+":"+d.getSeconds());
  </script>
</body>
</html>

```

Output:

JavaScript date object

20/12/2021
12:21:59



BSc CsIt SEM - V

Web technology

Set date

You can also set the date or time into the date object, with the setDate, setHour etc. Note that in this example, only the FullYear is set.

```
<!DOCTYPE html>
<html>
  <body>
    <h2>Set date</h2>
    <script>
      var d = new Date();
      d.setFullYear("1990");
      document.write(d);
    </script>
  </body>
</html>
```

Output:

Set date

Thu Dec 20 1990 12:26:37 GMT+0545 (Nepal Time)

Math Object

The built-in Math object includes mathematical constants and functions. You do not need to create the Math object before using it.

- To store a random number between 0 and 1 in a variable called "r_number":
r_number=Math.random()
- To store the rounded number of 8.6 in a variable called "r_number":
r_number=Math.round(8.6)

Properties	Explanation
E	Returns the base of a natural logarithm
LN2	Returns the natural logarithm of 2
LN10	Returns the natural logarithm of 10
LOG2E	Returns the base-2 logarithm of E
LOG10E	Returns the base-10 logarithm of E
PI	Returns PI
SQRT1_2	Returns 1 divided by the square root of 2
SQRT2	Returns the square root of 2

BSc CsIt SEM - V

Web technology

	Methods	Explanation
1	abs(x)	Returns the absolute value of x
2	sin(x)	Returns the sine of x
3	cos(x)	Returns the cosine of x
4	tan(x)	Returns the tangent of x
5	ceil(x)	Returns the nearest integer greater than or equal to x
6	floor(x)	Returns the nearest integer less than or equal to x
7	round(x)	Rounds x to the nearest integer
8	exp(x)	Returns the value of E raised to the power of x
9	log(x)	Returns the natural log of x
10	max(x,y)	Returns the number with the highest value of x and y
11	min(x,y)	Returns the number with the lowest value of x and y
12	pow(x,y)	Returns the value of the number x raised to the power of y
13	sqrt(x)	Returns the square root of x
14	random()	Returns a random number between 0 and 1

Round: How to round a specified number to the nearest whole number

```
<!DOCTYPE html>
```

```
<html>
```

```
<body>
```

```
<h2>Rounding the numbers</h2>
```

```
<script>
```

```
document.write(Math.round(7.25));  
document.write("<br>");
```

```
document.write(Math.random()); // a random number between 0 and 1  
document.write("<br>");
```

```
var no=Math.random()*10; // random number from 0 to 9 using the random  
method.
```

```
document.write(Math.floor(no));  
document.write("<br>");
```

```
document.write(Math.max(2,4)); // to return higher values  
document.write("<br>");
```

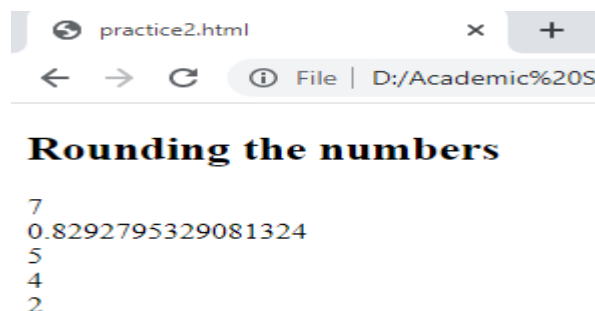
```
document.write(Math.min(2,4)); // to return lower values
```

```
</script>
```

```
</body>
```

```
</html>
```

Output:



BSc CsIt SEM - V

Web technology

RegExp

A regular expression is an object that describes a pattern of characters. The JavaScript RegExp class represents regular expressions, and both String and RegExp define methods that use regular expressions to perform powerful pattern-matching and search-and-replace functions on text.

So...

- A regular expression is a sequence of characters that forms a search pattern.
- When you search for data in a text, you can use this search pattern to describe what you are searching for.
- A regular expression can be a single character, or a more complicated pattern.
- Regular expressions can be used to perform all types of text search and text replace operations.

Syntax

A regular expression could be defined with the RegExp () constructor, as follows...

```
var pattern = new RegExp(pattern, attributes);
```

Or

```
var pattern = /pattern/attributes;
```

Here is the description of the parameters: -

- pattern: - A string that specifies the pattern of the regular expression or another regular expression.
- attributes: - An optional string containing any of the "g", "i", and "m" attributes that specify global, case-insensitive, and multi-line matches, respectively.

Example:

```
<!DOCTYPE html>
```

```
<html>
```

```
  <body>
```

```
    <h2>JavaScript Regular Expressions</h2>
```

```
    <p>Search a string for "sequence", and display the position of the match:</p>
```

```
    <p id="demo"></p>
```

```
    <script>
```

```
      let text = "A regular expression is a sequence of characters that forms a  
      search pattern!";
```

```
      let n = text.search(/sequence/);
```

```
      document.getElementById("demo").innerHTML = n;
```

```
    </script>
```

```
  </body>
```

```
</html>
```

Output:



BSsc CsIt SEM - V

Web technology

Document Object Model (DOM)

Every web page resides inside a browser window which can be considered as an object. A Document object represents the HTML document that is displayed in that window. The Document object has various properties that refer to other objects which allow access to and modification of document content. The way a document content is accessed and modified is called the Document Object Model (DOM).

So, DOM stands for Document Object Model. It is a programming interface that allows us to create, change, or remove elements from the document. We can also add events to these elements to make our page more dynamic.

It is called an Object Model because Documents are modelled using objects, and the model includes not only the structure of a document but also the behaviour of a document and the objects of which it is composed of like tag elements with attributes in HTML.

HTML is used to structure the web pages and JavaScript is used to add behaviour to our web pages. When an HTML file is loaded into the browser, the JavaScript cannot understand the HTML document directly. So, a corresponding document is created (DOM).

DOM is basically the representation of the same HTML document but in a different format with the use of objects. JavaScript interprets DOM easily i.e JavaScript cannot understand the tags(<h1>H</h1>) in HTML document but can understand object h1 in DOM. Now, JavaScript can access each of the objects (h1, p, etc) by using different functions.

Structure of DOM:

DOM can be thought of as a Tree or Forest (more than one tree). The term structure model is sometimes used to describe the tree-like representation of a document. The DOM views an HTML document as a tree of nodes. A node represents an HTML element.

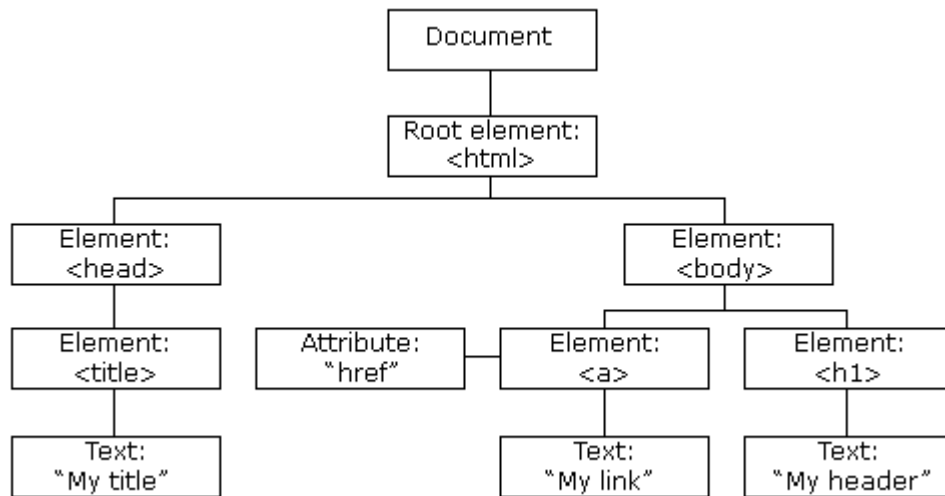
The DOM represents an HTML or XML document as a hierarchy of nodes. Consider the following HTML document:

```
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <a href="https://www.google.com">My link</a>
    <h1>My header</h1>
  </body>
</html>
```

BSc CsIt SEM - V

Web technology

The following tree represents the above HTML document:



We can access these elements in the document and make changes to them using JavaScript. Let's take a look at a few examples of how we can work with the DOM using JavaScript.

To Select Elements in the Document

There are a few different methods for selecting an element in the HTML document.

1. getElementById()
2. querySelector()
3. querySelectorAll()

getElementById ()

In JavaScript, we can grab an HTML tag by referencing the id name using getElementById() method.

```
<p id="para1">This is my first paragraph.</p>
```

```
<script>
```

```
const paragraph1 = document.getElementById("para1");
console.log(paragraph1);
```

```
</script>
```

BSc CsIt SEM - V

Web technology

querySelector ()

You can use this method to find elements with one or more CSS selectors.

In below example If we want to find and print h1 element to the console, then we could use that tag name inside the querySelector().

Example:

```
<!DOCTYPE html>
<html>
  <head>
    <title> JavaScript window object </title>
  </head>
  <body>
    <h1>Fruits</h1>
    <ul class="list">
      <li>Apple</li>
      <li>Orange</li>
      <li>Banana</li>
      <li>Grapes</li>
    </ul>
    <script>
      const h1Element = document.querySelector("h1");
      console.log(h1Element);
    </script>
  </body>
</html>
```

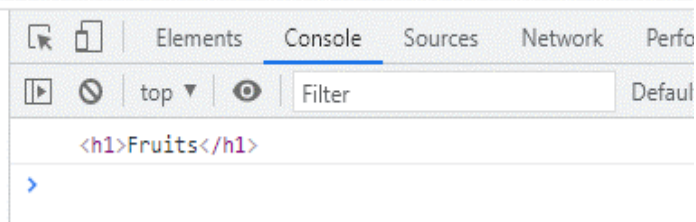
Output:



Fruits

- Apple
- Orange
- Banana
- Grapes

hnology/practical%20sem%205/unit%204/practice2.html



BSc CsIt SEM - V

Web technology

querySelectorAll()

This method finds all of the elements that match the CSS selector and returns a list of all of those nodes.

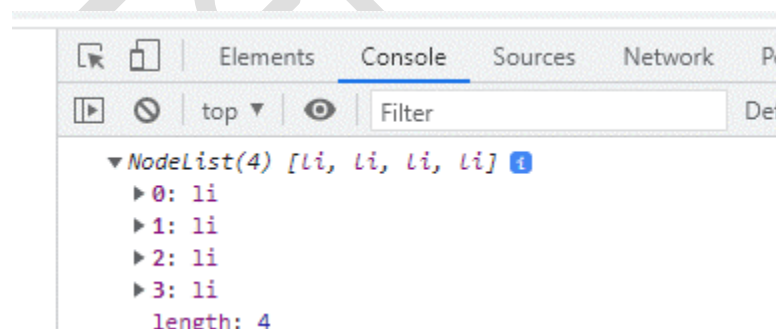
If I wanted to find all of the items in our example, I could use the > operator to find all of the children of the .

```
const listItems = document.querySelectorAll("ul > li");
console.log(listItems);
```

Example:

```
<!DOCTYPE html>
<html>
<head>
  <title> JavaScript window object </title>
</head>
<body>
  <h1>Fruits</h1>
  <ul class="list">
    <li>Apple</li>
    <li>Orange</li>
    <li>Banana</li>
    <li>Grapes</li>
  </ul>
  <script>
    const listItems = document.querySelectorAll("ul > li");
    console.log(listItems);
  </script>
</body>
</html>
```

Output:



BSc CsIt SEM - V

Web technology

To Add New Elements to the Document

To add a new element to the HTML DOM, you must create the element (element node) first, and then append it to an existing element.

createElement()

It is use to add new element to the DOM tree.

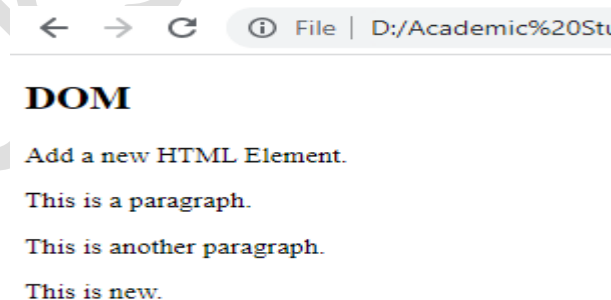
Example:

```
<!DOCTYPE html>
<html>
  <body>
    <h2>DOM</h2>
    <p>Add a new HTML Element.</p>

    <div id="div1">
      <p id="p1">This is a paragraph.</p>
      <p id="p2">This is another paragraph.</p>
    </div>
    <script>
      const para = document.createElement("p");
      const node = document.createTextNode("This is new.");
      para.appendChild(node);

      const element = document.getElementById("div1");
      element.appendChild(para);
    </script>
  </body>
</html>
```

Output:



Consider the another example

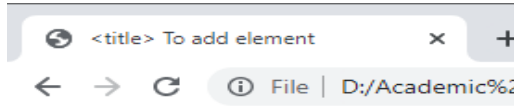
```
<!DOCTYPE html>
<html>
  <head>
    <title> To add element </title>
  </head>
  <body>
```

BSc CsIt SEM - V

Web technology

```
<h1>List of fruits</h1>
</body>
</html>
```

Output:



Lists of fruits

Right now, I just have an <h1> tag on the page. But if we want to add a list of fruits underneath that <h1> tag then...

Example

```
<!DOCTYPE html>
<html>
<head>
<title> <title> To add element </title> </title>
</head>
<body>
<h1>Lists of fruits</h1>

<script>
let unorderedList = document.createElement("ul");
document.body.appendChild(unorderedList);

let listItem1 = document.createElement("li");
listItem1.textContent = "Apple";
unorderedList.appendChild(listItem1);

let listItem2 = document.createElement("li");
listItem2.textContent = "orange";
unorderedList.appendChild(listItem2);

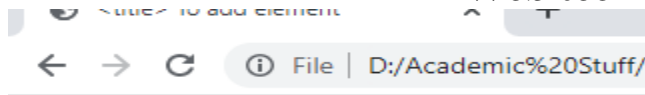
let listItem3 = document.createElement("li");
listItem3.textContent = "banana";
unorderedList.appendChild(listItem3);

let listItem4 = document.createElement("li");
listItem4.textContent = "grapes";
unorderedList.appendChild(listItem4);
</script>

</body>
</html>
```

BSc CsIt SEM - V

Web technology



Lists of fruits

- Apple
- orange
- banana
- grapes

To change the inline CSS styles of elements using the style property.

Example:

```
<!DOCTYPE html>
```

```
<html>
```

```
  <body>
```

```
    <h2>JavaScript HTML DOM</h2>
```

```
    <p>Changing the style:</p>
```

```
    <p id="p1">this is blue colored text</p>
```

```
    <p id="p2">This is Arial font with lager text size</p>
```

```
    <script>
```

```
      document.getElementById("p1").style.color = "blue";
```

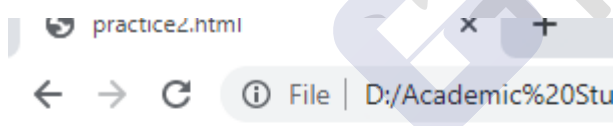
```
      document.getElementById("p2").style.fontFamily = "Arial";
```

```
      document.getElementById("p2").style.fontSize = "larger";
```

```
    </script>
```

```
  </body>
```

```
</html>
```



JavaScript HTML DOM

Changing the style:

this is blue colored text

This is Arial font with lager text size

User Defined Objects

JavaScript provides types of objects: Built-in and User Defined.

- Built-in objects: which are provided by the JavaScript core. Array, Strings, Number, Boolean, RegExp are all built-in objects
- User-defined objects: these are objects which you have created in your program or application.

BSc CsIt SEM - V

Web technology

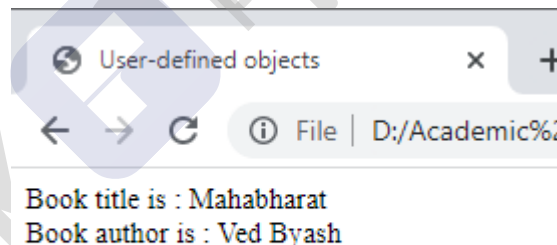
Example: User defined object

```
<html>
<head>
<title>User-defined objects</title>
<script type = "text/javascript">
    function book(title, author)
    {
        this.title = title;
        this.author = author;
    }
</script>
</head>

<body>
<script type = "text/javascript">
    var myBook = new book("Mahabharat", "Ved Byash");

    document.write("Book title is : " + myBook.title + "<br>");
    document.write("Book author is : " + myBook.author + "<br>");
</script>
</body>
</html>
```

Output:



Event Handling

When the page loads, it is called an event. When the user clicks a button, that click too is an event. Other examples are pressing any key, closing a window, resizing a window, etc.

Hence the change in the state of an object is known as an Event. In html, there are various events which represents that some activity is performed by the user or by the browser. When JavaScript code is included in HTML, JavaScript react over these events and allow the execution. This process of reacting over the events is called Event Handling. Thus, js handles the HTML events via Event Handlers.



BSc CsIt SEM - V

Web technology

Mouse events:

<u>Event Performed</u>	<u>Event Handler</u>	<u>Description</u>
click	onclick	When mouse click on an element
mouseover	onmouseover	When the cursor of the mouse comes over the element
mouseout	onmouseout	When the cursor of the mouse leaves an element
mousedown	onmousedown	When the mouse button is pressed over the element
mouseup	onmouseup	When the mouse button is released over the element
mousemove	onmousemove	When the mouse movement takes place.

Keyboard events:

<u>Event Performed</u>	<u>Event Handler</u>	<u>Description</u>
Keydown & Keyup	onkeydown & onkeyup	When the user press and then release the key

Form events:

<u>Event Performed</u>	<u>Event Handler</u>	<u>Description</u>
focus	onfocus	When the user focuses on an element
submit	onsubmit	When the user submits the form
blur	onblur	When the focus is away from a form element
change	onchange	When the user modifies or changes the value of a form element

Window/Document events

<u>Event Performed</u>	<u>Event Handler</u>	<u>Description</u>
load	onload	When the browser finishes the loading of the page
unload	onunload	When the visitor leaves the current webpage, the browser unloads it
resize	onresize	When the visitor resizes the window of the browser



BSc CsIt SEM - V

Web technology

Example

```
<html>
  <head>
    <title>Javascript Events</title>
  </head>
  <body onload="window.alert('Page successfully loaded');">
    <script language="Javascript" type="text/Javascript">

      //Javascript Events
      function clickevent()
      {
        document.write("You clicked mouse");
      }
      function mouseoverevent()
      {
        alert("Mouse hover event occurred");
      }
      function focusevent()
      {
        document.getElementById("input1").style.background=" aqua";
      }
      function keydownevent()
      {
        document.getElementById("input1");
        alert("Pressed a key");
      }

      document.write("The page is loaded successfully");
    </script>
  <form>
    <H2>Form events</H2>
    <input type="text" id="input1" onfocus="focusevent()"/>

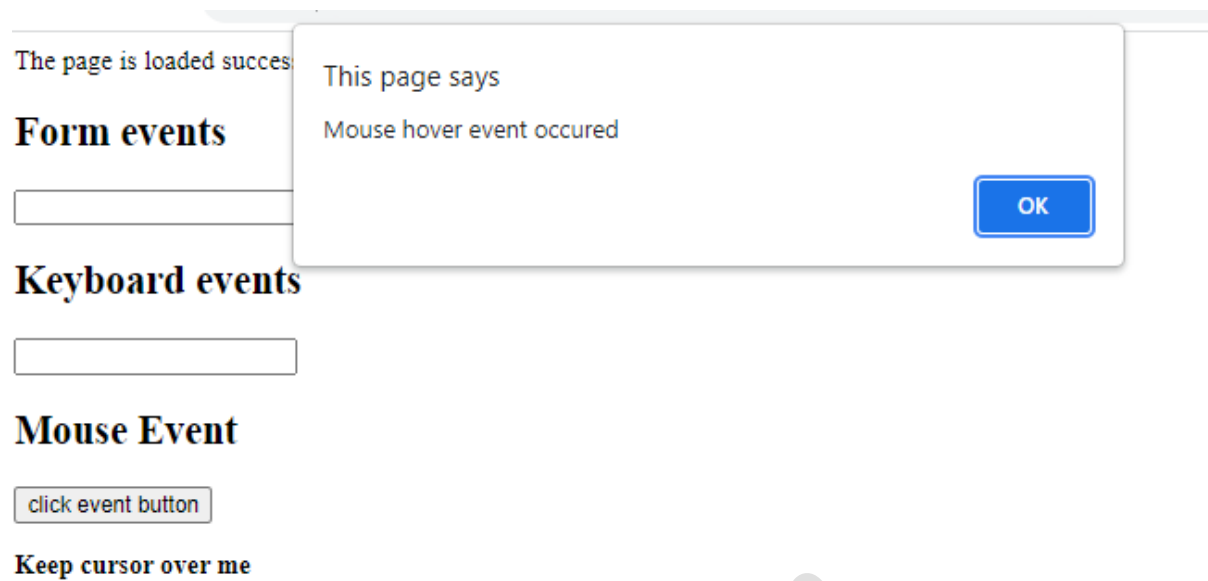
    <H2>Keyboard events</H2>
    <input type="text" id="input1" onkeydown="keydownevent()"/>

    <H2>Mouse Event</h2>
    <input type="button" onclick="clickevent()" value="click event button"/>
    <p onmouseover="mouseoverevent()"> <b>Keep cursor over me</b></p>
  </form>
</body>
</html>
```

BS Sc CsIt SEM - V

Web technology

Output:



Handling Cookies

A cookie is a piece of data that is stored on your computer to be accessed by your browser. You also might have enjoyed the benefits of cookies knowingly or unknowingly. Have you ever saved your Facebook password so that you do not have to type it each and every time you try to login? If yes, then you are using cookies. Cookies are saved as key/value pairs.

The communication between a web browser and server happens using a stateless protocol named HTTP. Stateless protocol treats each request independent. So, the server does not keep the data after sending it to the browser. But in many situations, the data will be required again. Here come cookies into a picture. With cookies, the web browser will not have to communicate with the server each time the data is required. Instead, it can be fetched directly from the computer.

Javascript Set Cookie

You can create cookies using document.cookie property like this.

```
document.cookie = "cookieName=cookieValue"
```

You can even add expiry date to your cookie so that the particular cookie will be removed from the computer on the specified date. The expiry date should be set in the UTC/GMT format. If you do not set the expiry date, the cookie will be removed when the user closes the browser.

```
document.cookie = "cookieName=cookieValue; expires= Thu, 21 Aug 2014 20:00:00 UTC"
```

You can also set the domain and path to specify to which domain and to which directories in the specific domain the cookie belongs to. By default, a cookie belongs to the page that sets the cookie.

BSc CsIt SEM - V

Web technology

```
document.cookie = "cookieName=cookievalue; expires= Thu, 21 Aug 2014 20:00:00 UTC; path=/"
```

//create a cookie with a domain to the current page and path to the entire domain.

JavaScript get Cookie

You can access the cookie like this which will return all the cookies saved for the current domain.

```
var x = document.cookie
```

JavaScript Delete Cookie

To delete a cookie, you just need to set the value of the cookie to empty and set the value of expires to a passed date.

```
document.cookie = "cookieName= ; expires = Thu, 01 Jan 1970 00:00:00 GMT"
```

jQuery

jQuery is a fast, small and feature-rich JavaScript library included in a single .js file. jQuery makes a web developer's life easy. It provides many built-in functions using which you can accomplish various tasks easily and quickly.

jQuery Important Features

- **DOM Selection:** jQuery provides Selectors to retrieve DOM element based on different criteria like tag name, id, css class name, attribute name, value, nth child in hierarchy etc.
- **DOM Manipulation:** You can manipulate DOM elements using various built-in jQuery functions. For example, adding or removing elements, modifying html content, css class etc.
- **Special Effects:** You can apply special effects to DOM elements like show or hide elements, fade-in or fade-out of visibility, sliding effect, animation etc.
- **Events:** jQuery library includes functions which are equivalent to DOM events like click, dblclick, mouseenter, mouseleave, blur, keyup, keydown etc. These functions automatically handle cross-browser issues.
- **Ajax:** jQuery also includes easy to use AJAX functions to load data from servers without reloading whole page.
- **Cross-browser support:** jQuery library automatically handles cross-browser issues, so the user does not have to worry about it. jQuery supports IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+.

BSc CsIt SEM - V

Web technology

Advantages of jQuery

- **Easy to learn:** jQuery is easy to learn because it supports same JavaScript style coding.
- **Write less do more:** jQuery provides a rich set of features that increase developers' productivity by writing less and readable code.
- **Excellent API Documentation:** jQuery provides excellent online API documentation.
- **Cross-browser support:** jQuery provides excellent cross-browser support without writing extra code.
- **Unobtrusive:** jQuery is unobtrusive which allows separation of concerns by separating html and jQuery code.

Disadvantages:

The JQuery javascript file is required to run JQuery commands, while the size of this file is relatively small (25-100KB depending on the server), it is still a strain on the client computer and maybe your web server as well if you intend to host the JQuery script on your own web server.

How to use jQuery?

There are two ways to use jQuery.

Local Installation – You can download jQuery library on your local machine and include it in your HTML code.

CDN Based Version – You can include jQuery library into your HTML code directly from Content Delivery Network (CDN).

Example:

```
<html>
<head>
<title>jQuery</title>
```

//By Local Installation

```
<script type = "text/javascript" src = "D:/Academic Stuff/BSc CSIT/5th
semester/Web-Technology/A online web technology/All doc and pdf/unit 4 overall/js
for local/jquery-3.3.1.min.js">
</script>
```

//By Content Delivery Network(CDN) Based Version

```
<!--
<script type = "text/javascript"
src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
</script>
-->
```

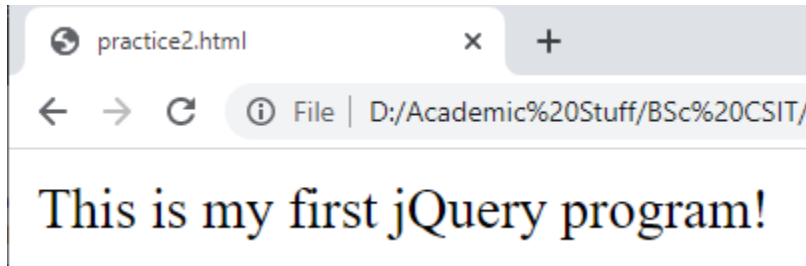
```
<script type = "text/javascript">
$(document).ready(function()
{
document.write("This is my first jQuery program!");
```

BSc CsIt SEM - V

Web technology

```
});  
</script>  
</head>  
  
<body>  
  <h1>Hello</h1>  
</body>  
</html>
```

Output



jQuery Syntax:

jQuery syntax is made by using HTML elements selector and perform some action on the elements are manipulation in Dot sign(.).

```
$(document).ready(function()  
{  
  $(selector).action();  
});
```

Here,

- \$ sign define the jQuery.
- A (selector) defines the Query element's to find in HTML element's.
- And action() to be performed on the given element.

The \$(document).ready Function

It is always a good practice to wait for the document to be load fully and hence be ready before working with it. You can check this using the ready event of the document element. So before starting your jQuery code, write the following lines of code and add all your jQuery code within this function.

```
$(document).ready(function()  
{  
  //add your jQuery code here  
});
```

BSc CsIt SEM - V

Web technology

So, `$(document).ready(function(){})` - Represents the **document ready event** and it is used to execute the jQuery code once the document is fully loaded and ready before working with it.

jQuery Selectors

jQuery Selectors allow you to select and manipulate HTML elements as a single element or group of elements and once the element is selected then you can perform various operation on that.

Syntax

jQuery Selector typically follows this syntax,

`$(selector).Method();`

You can also perform multiple jQuery operations by using jQuery chaining process,

`$(selector).Method1().Method2().Method3();`

jQuery Basic Selectors

<u>Selector</u>	<u>Example</u>	<u>Description</u>
element	<code>\$("p")</code>	Selected All <p> elements.
element, element	<code>\$("p, h3")</code>	Selected All <p> and <h3> elements.
*	<code>\$("*")</code>	Selects all elements
#id	<code>\$("#demo")</code>	Selects element whose id="demo"
.class	<code>\$(".param")</code>	Selects all the elements with class="param"
.class, .class	<code>\$(".param1, .param2")</code>	Selects all the elements with class="param1" and class="param2"

Example:id selector

```

<!DOCTYPE html>
<html>
<head>
  <title>jQuery hide p#div1 element</title>
  <script type = "text/javascript" src = "D:/Academic Stuff/BSc CSIT/5th
semester/Web-Technology/A online web technology/All doc and pdf/unit 4 overall/js
for local/jquery-3.3.1.min.js">
    </script>

```

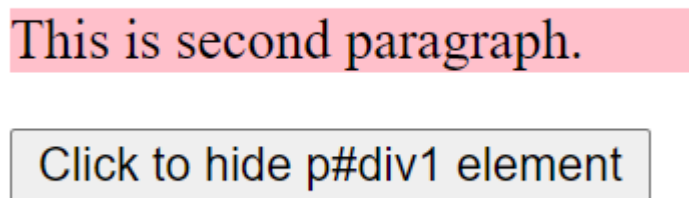
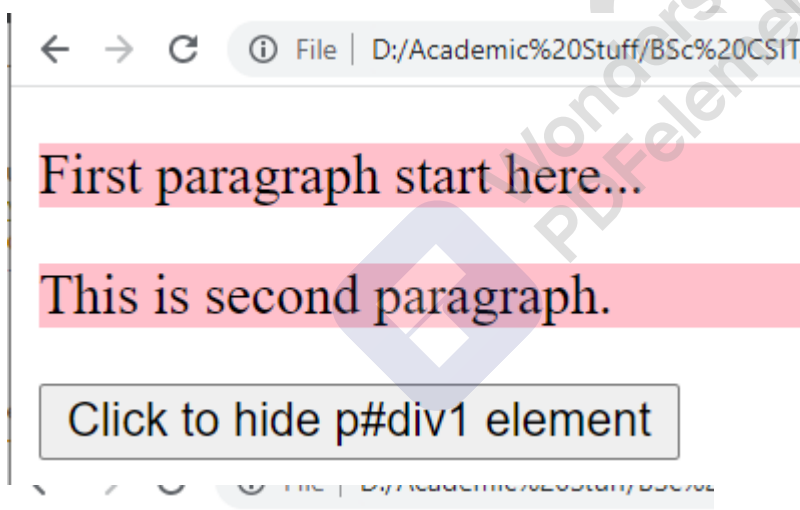

BSc CsIt SEM - V

Web technology

```
<script>
$(document).ready(function() {
    $("button").click(function()
    {
        $("#div1").hide();
    });
    $("p").css("background-color", "pink");
});

</script>
</head>
<body>
    <p id="div1">First paragraph start here...</p>
    <p>This is second paragraph.</p>
    <button>Click to hide p#div1 element</button>
</body>
</html>
```

Output:




BSc CsIt SEM - V

Web technology

Example: class selector

```
<!DOCTYPE html>
<html>
<head>
  <title>Class selector</title>
  <script type = "text/javascript" src = "D:/Academic Stuff/BSc CSIT/5th semester/Web-
Technology/A online web technology/All doc and pdf/unit 4 overall/js for local/jquery-
3.3.1.min.js">
    </script>
  <script>
    $(document).ready(function() {
      $("button").click(function(){
        $(".div1").hide();
      });
    });
  </script>
</head>
<body>
  <p class="div1">First paragraph start here...</p>
  <p class="div1">Second paragraph start here...</p>
  <button>Click here to hide above all paragraph</button>
</body>
</html>
```

Output:



First paragraph start here...

Second paragraph start here...

Click here to hide above all paragraph



Click here to hide above all paragraph

BSc CsIt SEM - V

Web technology

Events and Effects:

An event represents the precise moment when something happens.

Examples:

- moving a mouse over an element
- selecting a radio button
- clicking on an element

Here are some common DOM events:

<u>Mouse Events</u>	<u>Keyboard Events</u>	<u>Form Events</u>	<u>Document/Window Events</u>
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Class selector</title>
<script type = "text/javascript" src = "D:/Academic Stuff/BSc CSIT/5th semester/Web-
Technology/A online web technology/All doc and pdf/unit 4 overall/js for local/jquery-
3.3.1.min.js">
</script>
<script>
//click event
$(document).ready(function(){
    $("#p1").click(function(){
        $(this).hide();
    });
});

//mouse down event
$(document).ready(function(){
    $("#p2").mousedown(function(){
        alert("Mouse down over p1!");
    });
});

// Mouseup Event
```

BSc CsIt SEM - V

Web technology

```
$(document).ready(function(){
    $("#p3").mouseup(function(){
        alert("Mouse up over p1!");
    });
});

// Focus event
$(document).ready(function(){
    $("input").focus(function(){
        $(this).css("background-color", "yellow");
    });
    $("input").blur(function(){
        $(this).css("background-color", "green");
    });
});
// Mouseenter event

$(document).ready(function(){
    $("#pn").mouseenter(function(){
        alert("You entered p1!");
    });
});

</script>
</head>

<body>
<h2>click event</h2>
<p id="p1">After click, will disappear.</p>
<h2>Mouse down event</h2>
<p id="p2">mouse down</p>
<h2>Mouse up event</h2>
<p id="p3">Mouse down event</p>
<h2>Focus event</h2>
Name: <input type="text" name="fullname"><br>
Email: <input type="text" name="email">
<h2>Mouse enter event</h2>
<p id="pn">Enter this paragraph.</p>

</body>
</html>
```

BSc CsIt SEM - V

Web technology

jQuery Effects

"jQuery Effects" is a term that can be used to describe some of the jQuery functions which visibly affect HTML elements. jQuery provides a simple interface for doing various kind of amazing effects. jQuery methods allow us to quickly apply commonly used effects with a minimum configuration.

Some of the jQuery functions which cause HTML "effects"...

- \$(selector).hide() - Hide selected elements
- \$(selector).show() - Show selected elements
- \$(selector).toggle() - Toggle selected elements between hide and show
- \$(selector).slideDown() - Slide-down (show) selected elements
- \$(selector).slideUp() - Slide-up (hide) selected elements
- \$(selector).slideToggle() - Toggle slide-up and slide-down of selected elements
- \$(selector).fadeIn() - Fade in (show) selected elements
- \$(selector).fadeOut() - Fade out (hide) selected elements
- \$(selector).fadeTo() - Fade out selected elements to a given opacity
- \$(selector).animate() - Perform a custom animation (such as a width or height change) on selected elements
- \$(selector).html() - Change the html content (the innerHTML property) of the selected element

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Class selector</title>
  <script type = "text/javascript" src ="D:/Academic Stuff/BSc CSIT/5th
semester/Web-Technology/A online web technology/All doc and pdf/unit 4 overall/js
for local/jquery-3.3.1.min.js">
</script>
  <script>
    //Hide and show Effect
    $(document).ready(function(){
      $("#hide").click(function(){
        $("#hs").hide();
      });
      $("#show").click(function(){
        $("#hs").show();
      });
    });

    // toggle
    $(document).ready(function(){
      $("#toggle").click(function(){
```

BSc CsIt SEM - V

Web technology

```

        $("#togg").toggle();
    });
});
</script>

</head>

<body>
    <p id="hs">If you click on the "Hide" button, I will disappear.</p>
    <button id="hide">Hide</button>
    <button id="show">Show</button>
    <br>

    <p>following paragraph for toggling</p>
    <p id="togg">toggle one</p>
    <button id="toggle">Toggle between </button>
</body>
</html>

```

← → ↻ ⓘ File | D:/Academic%20Stuff/BSc%20CSIT/5th%20

If you click on the "Hide" button, I will disappear.

Hide Show

following paragraph for toggling

toggle one

Toggle between

===== End of Unit-4 =====

Unit 4: Client Side Scripting with JavaScript (9 Hrs.)

Structure of JavaScript Program; Variables and Data Types; Statements: Expression, Keyword, Block; Operators; Flow Controls, Looping, Functions; Popup Boxes: Alert, Confirm, Prompt; Objects and properties; Constructors; Arrays; Built-in Objects: Window, String, Number, Boolean, Date, Math, RegExp, Form, DOM; User Defined Objects; Event Handling and Form Validation, Error Handling, Handling Cookies, jQuery Syntax; jQuery Selectors, Events and Effects; Introduction to JSON