SARDAR PATEL EDUCATION CAMPUS

(MGD BY: TIRUPATI FOUNDATION TRUST)

SARDAR PATEL COLLEGE OF ADMINISTRATION AND MANAGEMENT
www.spec.edu.in 02692-235300

BACHELOR OF COMPUTER APPLICATION

(FY BCA)

SEMESTER – II

US02CBCA23: WEB APPLICATION DEVELOPMENT – II (W.E.F JUNE 2018)

FACULTY NAME: RUTUL PATEL MO.NO: 9737781143

INDEX

UNIT-1

❖ INTRODUCTION TO SCRIPTING

- Programming languages can be divided in two categories: **compiled and interpreted**.
- In **compiled languages** (e.g. C) **the source code goes through a compiler and produces an executable file** that can be run on a compatible machine.
- scripting language requires an interpreter
- Scripting languages originated as *job control languages*

"A scripting language is a programming language designed for integrating and communicating with other programming languages."

- Script is a sequence of commands written as plain text and run by an interpreter.
- Some of the most widely used scripting languages are JavaScript, VBScript, PHP, Perl, Python, Ruby, ASP and Tcl.
- A scripting language is normally used in conjunction with another programming language, they are often found alongside HTML, Java or C++.

Major advantages of scripting languages include:

- Easy To Learn And Use
- Minimum Programming Knowledge Or Experience Required
- Allow Complex Tasks To Be Performed In Relatively Few Steps
- Allow Simple Creation And Editing In A Variety Of Text Editors
- Allow The Addition Of Dynamic And Interactive Activities To Web Pages
- Editing And Running Code Is Fast.

❖ CLIENT SIDE SCRIPTING AND SERVER SIDE SCRIPTING

Explain client side & server side scripting.

Client-Side:

- It is an important part of the Dynamic HTML (DHTML).
- JavaScript is the main client-side scripting language for the web.
- The scripts are interpreted by the browser.
- It is used to make web pages change after they arrive at the browser.
- It is useful for making the pages a bit more interesting and user-friendly.
- It provides useful gadgets such as calculators, clocks etc.
- It enables interaction within a webpage.
- These scripting languages include JavaScript.
- It is affected by the processing speed of the user's computer.

Operation of Client-Side:

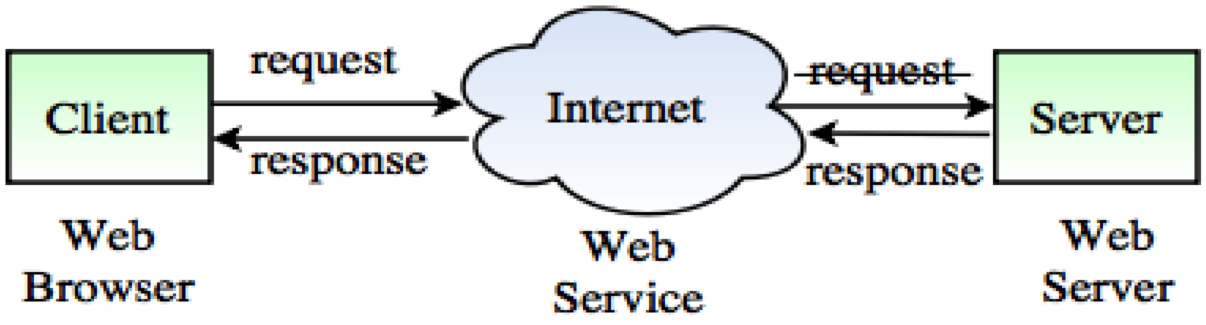


Fig. Client-Side Scripting

- The user requests a web page from the server.
- The server finds the page and sends it to the user.
- The page is displayed on the browser with any scripts running during or after the display.
- It is used to make web pages change after they arrive at the browser.
- These scripts rely on the user's computer. If the computer is slow, then they may run slow.
- These scripts may not run at all if the browser does not understand the scripting language.

<u>Server-Side:</u>

- It is a technique used in web development.
- The server-side environment that runs a scripting language is a web server.
- It is used to provide interactive web sites that interface to databases or other stores on the server.
- A user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages.
- It is different from client-side scripting where scripts are run by the viewing web browser, usually in JavaScript.
- It tends to be used for allowing users to have individual accounts and providing data from databases.
- Server-side allows a level of privacy, personalization and provision of information that is very powerful.
- These scripting languages include ASP.NET and PHP.
- It does not rely on the user having specific browser or plug-in.

- It is affected by the processing speed of the host server.

Operation of Server-Side:

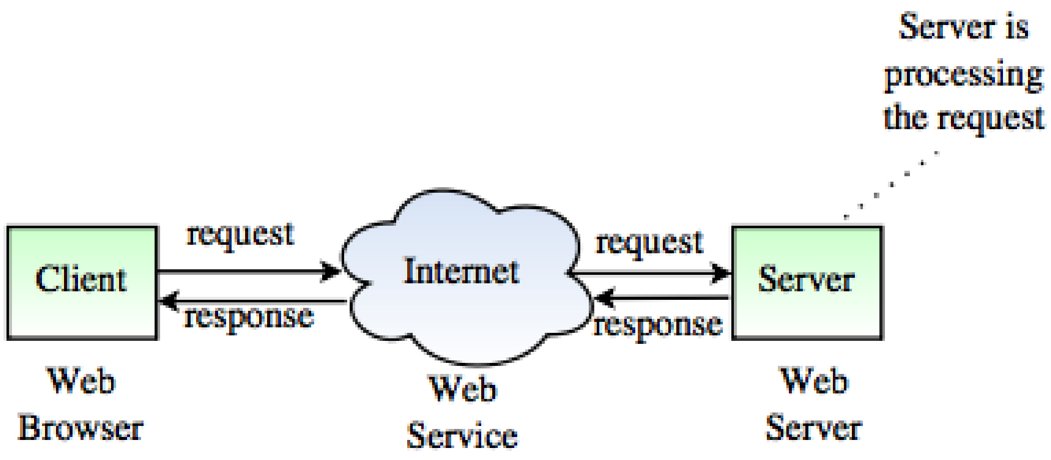


Fig. Server-Side Scripting

- The user (client) requests a web page from the server.
- The script in the page is interpreted by the server, creating or changing the page content to suit the user (client) and the passing data around.
- The page in its final form is sent to the user(client) and then cannot be changed using server-side scripting.
- It tends to be used for allowing users to have individual accounts and provides data from databases.
- Server-side scripting allows a level pf privacy, personalization and provision of information that is very useful.
- These scripts are never seen by the user.
- Server-side script runs on the server and generates results which are sent to the user.
- Running all the scripts puts a lot of load onto a server but not on the user's system.

	Client-side Scripting	**Server-side Scripting**
1	Client side script executed at browser.	Server side script executed at server
2	Client-side script is taking action on the user's (the client's) computer.	Server-side script means that the action takes place on a web server.
3	Client-side programming is run on the user's computer	Server-side scripts are run on the server.
4	Client side scripting is possible to be blocked by the user.	Server side scripting can't be blocked by the user.
5	Client side scripting cannot be used to connect to the databases on the web server.	Server side scripting is used to connect to the databases that reside on the web server.
6	Response from a client-side script is faster as compared to a server-side script because the scripts are processed on the local computer.	Response from a server-side script is slower as compared to a client-side script because the scripts are processed on the remote computer.
7	Examples of Client side scripting languages : Javascript, VB script, etc.	Examples of Server side scripting languages : PHP, ASP.NET, etc.

❖ INTRODUCTION TO JAVASCRIPT

- JavaScript is the **Programming Language** for the Web
- **JavaScript** is a object-based scripting language and it is light weighted.
- It is first implemented by Netscape (with help from Sun Microsystems).
- JavaScript was created by **Brendan Eich** at Netscape in 1995 for the purpose of allowing code in web-pages (performing logical operation on client side).
- JavaScript can update and change both **HTML** and **CSS**
- JavaScript can **calculate**, **manipulate** and **validate** data
- JavaScript was designed to add interactivity to HTML pages
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Netscape first introduced a JavaScript interpreter in Navigator 2.
- The interpreter was an extra software component in the browser that was capable of interpreting JavaScript source code inside an HTML document. This means that web page developer no needs other software other than a text editor of develop any web page.
- Everyone can use JavaScript without purchasing a license
- JavaScript is most commonly used as a client side scripting language.
- This means that JavaScript code is written into an HTML page.
- When a user requests an HTML page with JavaScript in it, the script is sent to the browser and it's up to the browser to do something with it.

- The fact that the script is in the HTML page means that scripts can be seen and copied by whoever views page.

What can a JavaScript do?

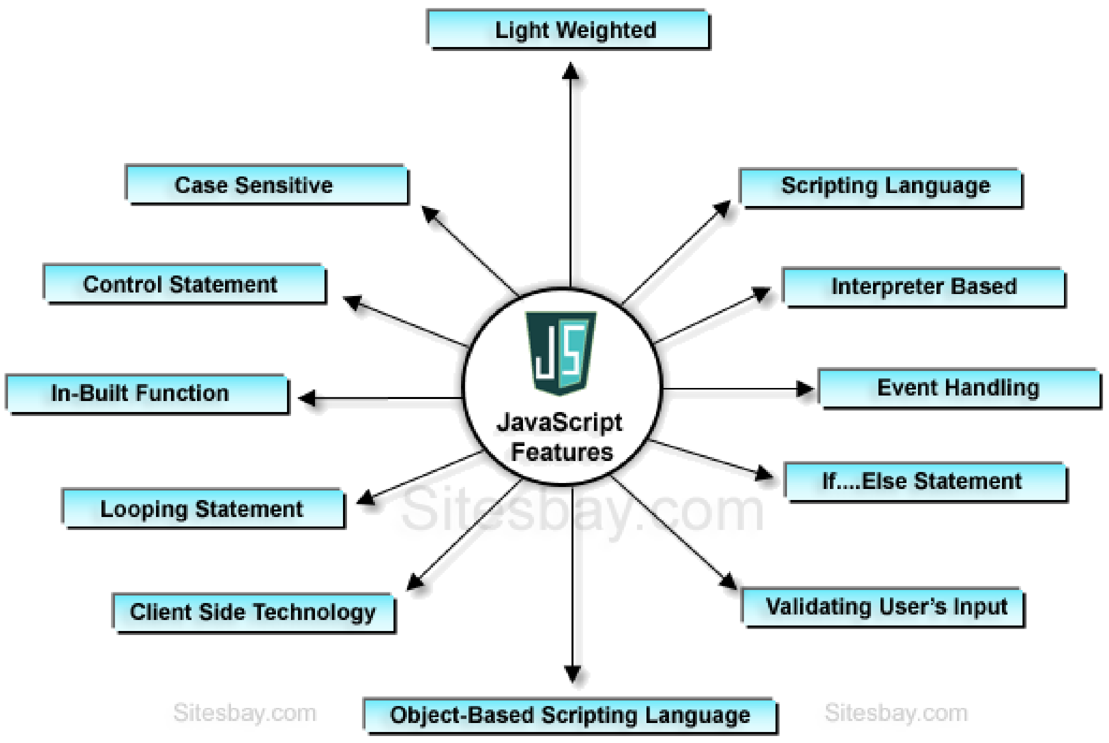
- **JavaScript gives HTML designers a programming tool -** HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages
- **JavaScript can put dynamic text into an HTML page -** A JavaScript statement like this: document. write("<h1>" + name + "</h1>") can write a variable text into an HTML page
- **JavaScript can react to events -** A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element
- **JavaScript can read and write HTML elements -** A JavaScript can read and change the content of an HTML element
- **JavaScript can be used to validate data -** A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing
- **JavaScript can be used to detect the visitor's browser** - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser
- **JavaScript can be used to create cookies** - A JavaScript can be used to store and retrieve information on the visitor's computer

Uses of JavaScript

- Using HTML we can only design a web page but you cannot run any logic on web browser like addition of two numbers, check any condition, looping statements (for, while), decision making statement (if-else) at client side. All these are not possible using HTML so for perform this entire task at client side you need to use JavaScript.
- Client-side validation
- Dynamic drop-down menus
- Displaying date and time
- Validate user input in an HTML form before sending the data to a server.
- Build forms that respond to user input without accessing a server.
- Change the appearance of HTML documents and dynamically write HTML into separate Windows.
- Open and close new windows or frames.
- Manipulate HTML "layers" including hiding, moving, and allowing the user to drag them around a browser window.
- Build small but complete client side programs.
- Displaying popup windows and dialog boxes (like alert dialog box, confirm dialog box and prompt dialog box)
- Displaying clocks etc.

❖ Features of JavaScript

- JavaScript is a client side technology, it is mainly used for gives client side validation, but it have lot of features which are given below;



- JavaScript is a object-based scripting language.
- Giving the user more control over the browser.
- It Handling dates and time.
- It Detecting the user's browser and OS,
- It is light weighted.
- JavaScript is a scripting language and it is not java.
- JavaScript is interpreter based scripting language.
- JavaScript is case sensitive.
- JavaScript is object based language as it provides predefined objects.
- Every statement in javascript must be terminated with semicolon (;).
- Most of the javascript control statements syntax is same as syntax of control statements in C language.
- An important part of JavaScript is the ability to create new functions within scripts. Declare a function in JavaScript using function keyword.

❖ Limitations of JavaScript

- Client-side JavaScript does not allow the reading or writing of files.
- It cannot be used for networking applications because there is no such support available.
- It doesn't have any multithreading or multiprocessor capabilities.

❖ Advantages of JavaScript:

- **An interpreted language**: JavaScript is an interpreted language, which requires no compilation steps. This provides an easy development process. The syntax is completely interpreted by the browser just as it interprets html tags.
- **Embedded within html:** JavaScript does not require any special or separate editor for programs to be written, edited or compiled. It can be written in any text editor like notepad along with appropriate html tags. And saved as filename.html html files with embedded JavaScript commands can then be read and interpreted by any browser that is JavaScript enabled.
- **Minimal syntax:** easy to learn: by leaning just a few commands and simple rules of syntax, complete applications can be built using JavaScript
- **Quick development:** JavaScript does not require compilation; script can be developed in a short period of time. This is enhanced by the fact that many gui interface features, such as alerts, prompts, confirm boxes and other gui elements, are handled by client side JavaScript, the browser and html code.
- **Design for simple, small programs:** it is well suited to implement simple small programs such programs can be easily written and executed at an acceptable speed using JavaScript.
- **Performance:** java script can be written such that the html files are fairly compact and quite small. This minimizes storage requirements on the web server and downloads time for the client.
- **Procedural capability:** every programming language are usually needs to support facilities such as condition checking, looping and branching. JavaScript provides syntax, which can be used to add such procedural capabilities to web page
- **Easy debugging and testing:** being an interpreted language, script in JavaScript tested line by line and error is also listed as they are encountered. it is thus easy to locate error, make changes, and test it again without the overhead and delay of compiling.
- **Platform independence:** JavaScript is a programming language that is completely independent of the hardware on which it works. It is a language by any JavaScript enabled browser. Thus JavaScript application work on any machine that has an appropriate JavaScript enabled browser installed. This machine can be anywhere on the network

❖ Disadvantages of JavaScript:-

- **Security:** Because the code executes on the user’s computer, in some cases it can be exploited for malicious purposes. This is one reason some people choose to disable JavaScript.
- **Reliance on End User**: JavaScript is sometimes interpreted differently by different browsers. Whereas server-side scripts will always produce the same output,client-side scripts can be a little unpredictable. Don't be overly concerned by this though - as long as you test your script in all the major browsers you should be safe.

❖ USING JAVASCRIPT ON A WEBPAGE

There are three places to put the JavaScript code.

1. Between the <head> </head> tag of html (Internal JavaScript)

- To write a JavaScript code In <head> section we can use <script> tag.
- The <script> tag is used to define a client-side script (JavaScript).
- The <script> element either contains scripting statements, or it points to an external script file through the src attribute.
- The <script> tag alerts the browser program to start interpreting all the text between these tags as a script.
- A simple syntax of your JavaScript will appear as follows.

SYNTAX

```
<script language=”JavaScript” type="text/JavaScript">

 // JavaScript code here

</script>
```

- **Language** − This attribute specifies what scripting language you are using. Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.
- **Type** − This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

Example

```
<html>
  <head>
    <script language="javascript" type="text/javascript">

        document.write("Hello World!")

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

2. Between the <body> </body> tag of html (Inline JavaScript)

- You can place the <script> tags, containing your JavaScript, anywhere within your web page

SYNTAX

```
<script language="JavaScript" type="text/JavaScript">

 // JavaScript code here

</script>
```

- **Language** − This attribute specifies what scripting language you are using. Typically, its value will be javascript. Although recent versions of HTML (and XHTML, its successor) have phased out the use of this attribute.
- **Type** − This attribute is what is now recommended to indicate the scripting language in use and its value should be set to "text/javascript".

Example

```
<html>

<body>

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

<script>

document.getElementById("demo").innerHTML = "Hello JavaScript!";

</script>

</body>

</html>
```

3. In .js file (External JavaScript)

- If you want to run the same JavaScript on several pages in a web site, you should create an external JavaScript file, instead of writing the same script over and over again.
- Save the script file with a .js extension, and then refer to it using the src attribute in the <script> tag.
- External scripts are practical when the same code is used in many different web pages.
- JavaScript files have the file extension **.js**.
- To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:

SYNTAX

<script src="*URL*">

URL: The URL of the external script file.

Example

1. Html file

```
<html>

<head>

<script src="demo.js" />

</head>

</html>
```

2. JavaScript file(demo.js)

```
Function abc ()
{
Document. Write (“this is external JavaScript”);
}
```

UNIT-2

❖ Data Types:

- JavaScript variables are containers for storing data values.
- One of the most fundamental characteristics of a programming language is the set of data types it supports.
- These are the type of values that can be represented and manipulated in a programming language.
- JavaScript provides different data types to hold different types of values. There are two types of data types in JavaScript.
  1) Primitive data type
  2) Non-primitive (reference) data type
- JavaScript is a dynamic type language, means you don't need to specify type of the variable because it is dynamically used by JavaScript engine.
- You need to use var here to specify the data type. It can hold any type of values such as numbers, strings

For example:

1) var a=40;//holding number
2) var b="Rahul";//holding string

JavaScript allows you to work with three primitive data types −

1) **Numbers,** eg. 123, 120.50 etc.
2) **Strings** of text e.g. "This text string" etc.
3) **Boolean** e.g. true or false.

1) Number

- There is only one type of Number in JavaScript. Numbers can be written with or without a decimal point. A number can also be +Infinity, -Infinity, and NaN (not a number).
- The Number is a primitive data type in JavaScript.
- Number type represents integer, float, hexadecimal, octal or exponential value.
- First character in a Number type must be an integer value and it must not be enclosed in quotation marks.

Syntax:

```
var int = 100;
var float = 100.5;
var hex = 0xfff;
var exponential = 2.56e3;
var octal = 030;
```

Example

```
<html>

<body>
```

```
  <h1>Demo: Number in JavaScript</h1>

  <p id="p1"></p>

  <p id="p2"></p>

  <p id="p3"></p>

  <p id="p3"></p>

  <p id="p4"></p>

  <p id="p5"></p>

  <script>

          var int = 100;

          var float = 100.5;

          var hex = 0xfff;

          var exponential = 2.56e3;

          var octal = 030;

          document.getElementById("p1").innerHTML = int;

          document.getElementById("p2").innerHTML = float;

          document.getElementById("p3").innerHTML = hex;

          document.getElementById("p4").innerHTML = exponential;

          document.getElementById("p5").innerHTML = octal;

   </script>

</body>

</html>
```

2) **<u>Strings</u>** :

- JavaScript strings are used for storing and manipulating text.
- String is a primitive data type in JavaScript. A string is textual content. It must be enclosed in single or double quotation marks.
- Strings are used for storing text.
- Strings must be inside of either double or single quotes.

Syntax:

```
var str1 = 'hello, it is me';
var str2 = "hello, it's me";
```

Example

```
<html>

<body>

    <h1>Demo: JavaScript String</h1>

    <p id="p1"></p>

    <p id="p2"></p>

    <script>

        var str1 = "Hello World";

        var str2 = 'Hello World';

        document.getElementById("p1").innerHTML = str1;

        document.getElementById("p2").innerHTML = str2;

 </script>

</body>

</html>
```

OUTPUT

```
Demo: JavaScript String

Hello World

Hello World
```

3) Boolean

- A JavaScript Boolean represents one of two values: **true** or **false**.
- Boolean is a primitive data type in JavaScript.
- Boolean can have only two values, true or false.
- It is useful in controlling program flow using conditional statements like if..else, switch, while, do..while.

Boolean Values

Very often, in programming, you will need a data type that can only have one of two values, like

1) YES / NO
2) ON / OFF
3) TRUE / FALSE

For this, JavaScript has a **Boolean** data type. It can only take the values **true** or **false**.

Syntax:

```
var YES = true;

var NO = false;
```

Example

```
<html>
<body>
            <h1>Demo: JavaScript Boolean</h1>
            <script>
              var YES = true;
              var NO = false;

              if(YES)
              {
                      alert("This code block will be executed");
              }

              if(NO)
              {
                      alert("This code block will not be executed");
              }
  </script>
</body>
</html>
```

Example 2:

```
<html>
<body>
 <h1>Demo: JavaScript Boolean</h1>
 <script>
        alert(1 > 2); // false

        alert(10< 9); // false
```

```
            alert(5 == 5); // true
    </script>
  </body>
  </html>
```

❖ Variables:

- Before you use a variable in a JavaScript program, you must declare it. Variables are declared with the **var** keyword as follows.
- JavaScript uses reserved keyword **var** to declare a variable.
- A variable must have a unique name.
- You can assign a value to a variable using **equal to** (=) operator when you declare it or before using it.

Syntax 1:

```
var <variable-name>;

var <variable-name> = <value>;
```

Syntax 2:

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

    var money;
    var name;

</script>
```

You can also declare multiple variables with the same **var** keyword as follows −

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

      var money, name;

</script>
```

- A JavaScript variable is simply a name of storage location. There are two types of variables in JavaScript : local variable and global variable.

JavaScript local variable

- A JavaScript local variable is declared inside block or function. It is accessible within the function or block only. For example:

```
<script>
function abc()
{
var x=10;//local variable
}
</script>
```

OR,

```
<script>
If(10<13)
{
var y=20;//JavaScript local variable
}
</script>
```

JavaScript global variable

- A JavaScript global variable is accessible from any function. A variable i.e. declared outside the function or declared with window object is known as global variable. For example:

```
<script>

var data=200;//gloabal variable

function a(){

document.writeln(data);

}

function b(){

document.writeln(data);

}

a();//calling JavaScript function

b();

</script>
```

Rules for creating variable:

The general rules for constructing names for variables (unique identifiers) are:

- Names can contain letters, digits, underscores, and dollar signs.
- Names must begin with a letter
- Names can also begin with $ and _ (but we will not use it in this tutorial)
- Names are case sensitive (y and Y are different variables)
- Reserved words (like JavaScript keywords) cannot be used as names

```
var x;                    // Declaration and initialization
```

```
x = "Hello World";              // Assignment

Or

var y = "Hello World";        // all in one
```

Variables lifetime and scope

A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.The completion of a function deletes the local variable.

A global variable has a global scope which means it can be defined anywhere in your JavaScript code. Global variables delete when the web browser is closed. However if a new page is loaded in the same browser window, then it remains.

Here's the usage of global variables:

You can try to run the following code to learn how to work with a scope of variables in JavaScript

Example

```
<html>
  <body onload = checkscope();>
    <script>
      <!--
      var myVar = "global";   // Declare a global variable
      function checkscope( ) {
        var myVar = "local"; // Declare a local variable
        document.write(myVar);
      }
      //-->
    </script>
  </body>
</html>
```

❖ Operators

Let us take a simple expression 4 + 5 is equal to 9. Here 4 and 5 are called operands and '+' is called the operator. JavaScript supports the following types of operators.

1. Arithmetic Operators
2. Comparison Operators
3. Logical (or Relational) Operators
4. Assignment Operators
5. Conditional (or ternary) Operators

1. Arithmetic Operators:-

JavaScript supports the following arithmetic operators −

Assume variable A holds 10 and variable B holds 20, then −

Sr.No.	Operator & Description
1	**+ (Addition)** Adds two operands **Ex:** A + B will give 30
2	**- (Subtraction)** Subtracts the second operand from the first **Ex:** A - B will give -10
3	*** (Multiplication)** Multiply both operands **Ex:** A * B will give 200
4	**/ (Division)** Divide the numerator by the denominator **Ex:** B / A will give 2
5	**% (Modulus)** Outputs the remainder of an integer division **Ex:** B % A will give 0
6	**++ (Increment)** Increases an integer value by one **Ex:** A++ will give 11
7	**-- (Decrement)** Decreases an integer value by one **Ex:** A-- will give 9

Note − Addition operator (+) works for Numeric as well as Strings. e.g. "a" + 10 will give "a10".

Example

The following code shows how to use arithmetic operators in JavaScript.

```
<html>
  <body>

    <script type = "text/javascript">
      <!--
        var a = 33;
        var b = 10;
        var c = "Test";
        var linebreak = "<br />";

        document.write("a + b = ");
        result = a + b;
        document.write(result);
        document.write(linebreak);

        document.write("a - b = ");
        result = a - b;
        document.write(result);
        document.write(linebreak);

        document.write("a / b = ");
        result = a / b;
        document.write(result);
        document.write(linebreak);

        document.write("a % b = ");
        result = a % b;
        document.write(result);
        document.write(linebreak);

        document.write("a + b + c = ");
        result = a + b + c;
        document.write(result);
        document.write(linebreak);

        a = ++a;
        document.write("++a = ");
        result = ++a;
        document.write(result);
        document.write(linebreak);

        b = --b;
        document.write("--b = ");
        result = --b;
        document.write(result);
        document.write(linebreak);
      //-->
    </script>

    Set the variables to different values and then try...
  </body>
</html>
```

Output

```
a + b = 43
a - b = 23
```

```
a / b = 3.3
a % b = 3
a + b + c = 43Test
++a = 35
--b = 8
Set the variables to different values and then try...
```

2. Comparison Operators

JavaScript supports the following comparison operators −
Assume variable A holds 10 and variable B holds 20, then −

Sr.No.	Operator & Description
1	**= = (Equal)** Checks if the value of two operands are equal or not, if yes, then the condition becomes true. **Ex:** (A == B) is not true.
2	**!= (Not Equal)** Checks if the value of two operands are equal or not, if the values are not equal, then the condition becomes true. **Ex:** (A != B) is true.
3	**> (Greater than)** Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true. **Ex:** (A > B) is not true.
4	**< (Less than)** Checks if the value of the left operand is less than the value of the right operand, if yes, then the condition becomes true. **Ex:** (A < B) is true.
5	**>= (Greater than or Equal to)** Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true. **Ex:** (A >= B) is not true.
6	**<= (Less than or Equal to)** Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true. **Ex:** (A <= B) is true.

Example

The following code shows how to use comparison operators in JavaScript.

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var a = 10;
        var b = 20;
        var linebreak = "<br />";
```

```
            document.write("(a == b) => ");
            result = (a == b);
            document.write(result);
            document.write(linebreak);

            document.write("(a < b) => ");
            result = (a < b);
            document.write(result);
            document.write(linebreak);

            document.write("(a > b) => ");
            result = (a > b);
            document.write(result);
            document.write(linebreak);

            document.write("(a != b) => ");
            result = (a != b);
            document.write(result);
            document.write(linebreak);

            document.write("(a >= b) => ");
            result = (a >= b);
            document.write(result);
            document.write(linebreak);

            document.write("(a <= b) => ");
            result = (a <= b);
            document.write(result);
            document.write(linebreak);
         //-->
      </script>
      Set the variables to different values and different operators and then try...
   </body>
</html>
```

Output

```
(a == b) => false
(a < b) => true
(a > b) => false
(a != b) => true
(a >= b) => false
a <= b) => true
Set the variables to different values and different operators and then try...
```

3. Logical Operators

JavaScript supports the following logical operators −

Assume variable A holds 10 and variable B holds 20, then −

Sr.No.	Operator & Description
1	**&& (Logical AND)** If both the operands are non-zero, then the condition becomes true. **Ex:** (A && B) is true.
2	**|| (Logical OR)** If any of the two operands are non-zero, then the condition becomes true. **Ex:** (A || B) is true.
3	**! (Logical NOT)** Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false. **Ex:** ! (A && B) is false.

Example

Try the following code to learn how to implement Logical Operators in JavaScript.

Live Demo

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var a = true;
        var b = false;
        var linebreak = "<br />";

        document.write("(a && b) => ");
        result = (a && b);
        document.write(result);
        document.write(linebreak);

        document.write("(a || b) => ");
        result = (a || b);
        document.write(result);
        document.write(linebreak);

        document.write("!(a && b) => ");
        result = (!(a && b));
        document.write(result);
        document.write(linebreak);
      //-->
    </script>
    <p>Set the variables to different values and different operators and then try...</p>
  </body>
</html>
```

Output

```
(a && b) => false
(a || b) => true
!(a && b) => true
Set the variables to different values and different operators and then try...
```

4. Bitwise Operators

JavaScript supports the following bitwise operators −

Assume variable A holds 2 and variable B holds 3, then −

Sr.No.	Operator & Description
1	**& (Bitwise AND)** It performs a Boolean AND operation on each bit of its integer arguments. **Ex:** (A & B) is 2.
2	**| (BitWise OR)** It performs a Boolean OR operation on each bit of its integer arguments. **Ex:** (A | B) is 3.
3	**^ (Bitwise XOR)** It performs a Boolean exclusive OR operation on each bit of its integer arguments. Exclusive OR means that either operand one is true or operand two is true, but not both. **Ex:** (A ^ B) is 1.
4	**~ (Bitwise Not)** It is a unary operator and operates by reversing all the bits in the operand. **Ex:** (~B) is -4.
5	**<< (Left Shift)** It moves all the bits in its first operand to the left by the number of places specified in the second operand. New bits are filled with zeros. Shifting a value left by one position is equivalent to multiplying it by 2, shifting two positions is equivalent to multiplying by 4, and so on. **Ex:** (A << 1) is 4.
6	**>> (Right Shift)** Binary Right Shift Operator. The left operand's value is moved right by the number of bits specified by the right operand. **Ex:** (A >> 1) is 1.
7	**>>> (Right shift with Zero)** This operator is just like the >> operator, except that the bits shifted in on the left are always zero. **Ex:** (A >>> 1) is 1.

Example

Try the following code to implement Bitwise operator in JavaScript.

Live Demo

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var a = 2; // Bit presentation 10
        var b = 3; // Bit presentation 11
        var linebreak = "<br />";
```

```
            document.write("(a & b) => ");
            result = (a & b);
            document.write(result);
            document.write(linebreak);

            document.write("(a | b) => ");
            result = (a | b);
            document.write(result);
            document.write(linebreak);

            document.write("(a ^ b) => ");
            result = (a ^ b);
            document.write(result);
            document.write(linebreak);

            document.write("(~b) => ");
            result = (~b);
            document.write(result);
            document.write(linebreak);

            document.write("(a << b) => ");
            result = (a << b);
            document.write(result);
            document.write(linebreak);

            document.write("(a >> b) => ");
            result = (a >> b);
            document.write(result);
            document.write(linebreak);
        //-->
      </script>
      <p>Set the variables to different values and different operators and then try...</p>
   </body>
</html>
```

output

```
(a & b) => 2
(a | b) => 3
(a ^ b) => 1
(~b) => -4
(a << b) => 16
(a >> b) => 0
Set the variables to different values and different operators and then try...
```

5. Assignment Operators

JavaScript supports the following assignment operators –

Sr.No.	Operator & Description
1	**= (Simple Assignment)** Assigns values from the right side operand to the left side operand **Ex:** C = A + B will assign the value of A + B into C
2	**+= (Add and Assignment)** It adds the right operand to the left operand and assigns the result to the left operand. **Ex:** C += A is equivalent to C = C + A
3	**−= (Subtract and Assignment)** It subtracts the right operand from the left operand and assigns the result to the left operand. **Ex:** C -= A is equivalent to C = C - A
4	***= (Multiply and Assignment)** It multiplies the right operand with the left operand and assigns the result to the left operand. **Ex:** C *= A is equivalent to C = C * A
5	**/= (Divide and Assignment)** It divides the left operand with the right operand and assigns the result to the left operand. **Ex:** C /= A is equivalent to C = C / A
6	**%= (Modules and Assignment)** It takes modulus using two operands and assigns the result to the left operand. **Ex:** C %= A is equivalent to C = C % A

Note − Same logic applies to Bitwise operators so they will become like <<=, >>=, >>=, &=, |= and ^=.

Example

Try the following code to implement assignment operator in JavaScript.

Live Demo

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var a = 33;
        var b = 10;
        var linebreak = "<br />";

        document.write("Value of a => (a = b) => ");
        result = (a = b);
        document.write(result);
        document.write(linebreak);

        document.write("Value of a => (a += b) => ");
        result = (a += b);
        document.write(result);
        document.write(linebreak);

        document.write("Value of a => (a -= b) => ");
        result = (a -= b);
        document.write(result);
        document.write(linebreak);
        document.write("Value of a => (a *= b) => ");
        result = (a *= b);
        document.write(result);
        document.write(linebreak);

        document.write("Value of a => (a /= b) => ");
        result = (a /= b);
        document.write(result);
        document.write(linebreak);

        document.write("Value of a => (a %= b) => ");
        result = (a %= b);
        document.write(result);
        document.write(linebreak);
      //-->
    </script>
    <p>Set the variables to different values and different operators and then try...</p>
  </body>
</html>
```

Output

```
Value of a => (a = b) => 10
Value of a => (a += b) => 20
Value of a => (a -= b) => 10
Value of a => (a *= b) => 100
Value of a => (a /= b) => 10
Value of a => (a %= b) => 0
Set the variables to different values and different operators and then try...
```

6. Miscellaneous Operator

We will discuss two operators here that are quite useful in JavaScript: the **conditional operator** (? :)

Conditional Operator (? :)

The conditional operator first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation.

Sr.No.	Operator and Description
1	**? : (Conditional)** If Condition is true? Then value X : Otherwise value Y

Example

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var a = 10;
        var b = 20;
        var linebreak = "<br />";

        document.write ("((a > b) ? 100 : 200) => ");
        result = (a > b) ? 100 : 200;
        document.write(result);
        document.write(linebreak);

        document.write ("((a < b) ? 100 : 200) => ");
        result = (a < b) ? 100 : 200;
        document.write(result);
        document.write(linebreak);
      //-->
    </script>
    <p>Set the variables to different values and different operators and then try...</p>
  </body>
</html>
```

Output

```
((a > b) ? 100 : 200) => 200
((a < b) ? 100 : 200) => 100
Set the variables to different values and different operators and then try...
```

❖ User interaction through dialog boxes

❖ Alert Dialog Box

- An alert dialog box is mostly used to give a warning message to the users.
- For example, if one input field requires to enter some text but the user does not provide any input, then as a part of validation, you can use an alert box to give a warning message.
- Nonetheless, an alert box can still be used for friendlier messages.
- Alert box gives only one button "OK" to select and proceed.

Example

```
<html>
  <head>
    <script type = "text/javascript">
      <!--
        function Warn() {
          alert ("This is a warning message!");
          document.write ("This is a warning message!");
        }
      //-->
    </script>
  </head>

  <body>
    <p>Click the following button to see the result: </p>
    <form>
      <input type = "button" value = "Click Me" onclick = "Warn();" />
    </form>
  </body>
</html>
```

❖ Confirmation Dialog Box

- A confirmation dialog box is mostly used to take user's consent on any option.
- It displays a dialog box with two buttons: **OK** and **Cancel**.
- If the user clicks on the OK button, the window method **confirm()** will return true. If the user clicks on the Cancel button, then **confirm()** returns false.
- You can use a confirmation dialog box as follows.

Example

```
<html>
  <head>
    <script type = "text/javascript">
      <!--
        function getConfirmation() {
          var retVal = confirm("Do you want to continue ?");
          if( retVal == true ) {
            document.write ("User wants to continue!");
            return true;
          } else {
            document.write ("User does not want to continue!");
            return false;
          }
        }
      //-->
    </script>
  </head>

  <body>
    <p>Click the following button to see the result: </p>
    <form>
      <input type = "button" value = "Click Me" onclick = "getConfirmation();" />
    </form>
  </body>
</html>
```

❖ Prompt Dialog Box

- The prompt dialog box is very useful when you want to pop-up a text box to get user input. Thus, it enables you to interact with the user.
- The user needs to fill in the field and then click OK.
- This dialog box is displayed using a method called **prompt()** which takes two parameters: (i) a label which you want to display in the text box and (ii) a default string to display in the text box.
- This dialog box has two buttons: **OK** and **Cancel**. If the user clicks the OK button, the window method **prompt()** will return the entered value from the text box. If the user clicks the Cancel button, the window method **prompt()** returns **null**.

Example

```
<html>
  <head>
    <script type = "text/javascript">
      <!--
        function getValue() {
          var retVal = prompt("Enter your name : ", "your name here");
          document.write("You have entered : " + retVal);
        }
      //-->
    </script>
  </head>

  <body>
    <p>Click the following button to see the result: </p>
    <form>
      <input type = "button" value = "Click Me" onclick = "getValue();" />
    </form>
  </body>
</html>
```

❖ Flow Control statements: Decision- Making and Looping

❖ Decision- Making

JavaScript supports conditional statements which are used to perform different actions based on different conditions

1. if statement
2. if...else statement
3. if...else if... statement.

1. if statement

- The **if** statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.

Syntax

```
if (expression)

{

  Statement(s) to be executed if expression is true

}
```

- Here a JavaScript expression is evaluated. If the resulting value is true, the given statement(s) are executed.
- If the expression is false, then no statement would be not executed. Most of the times, you will use comparison operators while making decisions.

Example

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var age = 20;

        if( age > 18 ) {
          document.write("<b>Qualifies for driving</b>");
        }
      //-->
    </script>
    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

Output

Qualifies for driving
Set the variable to different value and then try...

2. if...else statement

- The **'if...else'** statement is the next form of control statement that allows JavaScript to execute statements in a more controlled way.

Syntax

```
if (expression)

 {

  Statement(s) to be executed if expression is true

}

else

 {
  Statement(s) to be executed if expression is false
}
```

- Here JavaScript expression is evaluated. If the resulting value is true, the given statement(s) in the 'if' block, are executed.
- If the expression is false, then the given statement(s) in the else block are executed.

Example

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var age = 15;

        if( age > 18 ) {
          document.write("<b>Qualifies for driving</b>");
        } else {
          document.write("<b>Does not qualify for driving</b>");
        }
      //-->
    </script>
    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

Output

Does not qualify for driving
Set the variable to different value and then try...

3. if...else if... statement

- The **if...else if...** statement is an advanced form of **if…else** that allows JavaScript to make a correct decision out of several conditions.

Syntax

```
if (expression 1)
 {
  Statement(s) to be executed if expression 1 is true
}
else if (expression 2)
 {
  Statement(s) to be executed if expression 2 is true
}
else if (expression 3)
 {
  Statement(s) to be executed if expression 3 is true
}
```

```
else
{
  Statement(s) to be executed if no expression is true
}
```

- There is nothing special about this code. It is just a series of **if** statements, where each **if** is a part of the **else** clause of the previous statement. Statement(s) are executed based on the true condition, if none of the conditions is true, then the **else** block is executed.

Example

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var book = "maths";
        if( book == "history" ) {
          document.write("<b>History Book</b>");
        } else if( book == "maths" ) {
          document.write("<b>Maths Book</b>");
        } else if( book == "economics" ) {
          document.write("<b>Economics Book</b>");
        } else {
          document.write("<b>Unknown Book</b>");
        }
      //-->
    </script>
    <p>Set the variable to different value and then try...</p>
  </body>
<html>
```

Output

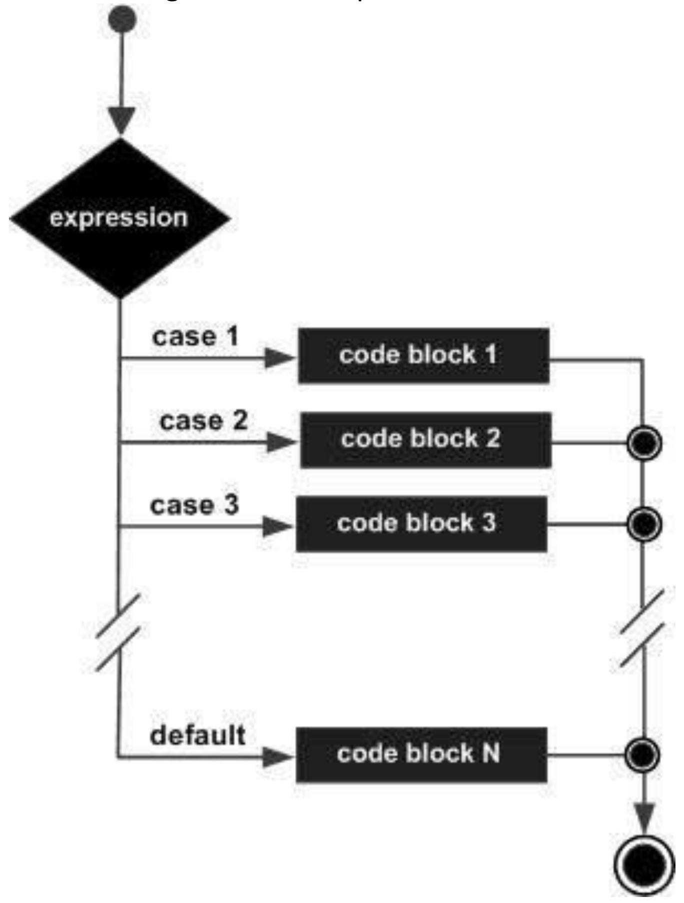
Maths Book
Set the variable to different value and then try...

❖ SWITCH statement

you can use a **switch** statement which handles exactly this situation, and it does so more efficiently than repeated **if...else if** statements.

Flow Chart

The following flow chart explains a switch-case statement works.



Syntax

- The objective of a **switch** statement is to give an expression to evaluate and several different statements to execute based on the value of the expression.
- The interpreter checks each **case** against the value of the expression until a match is found. If nothing matches, a **default** condition will be used.

```
switch (expression)
{
  case condition 1: statement(s)
  break;

  case condition 2: statement(s)
  break;
  ...

  case condition n: statement(s)
  break;

  default: statement(s)
}
```

- The **break** statements indicate the end of a particular case.
- If they were omitted, the interpreter would continue executing each statement in each of the following cases.

Example

Try the following example to implement switch-case statement.

Live Demo

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var grade = 'A';
        document.write("Entering switch block<br />");
        switch (grade) {
          case 'A': document.write("Good job<br />");
          break;

          case 'B': document.write("Pretty good<br />");
          break;

          case 'C': document.write("Passed<br />");
          break;

          case 'D': document.write("Not so good<br />");
          break;

          case 'F': document.write("Failed<br />");
          break;

          default:  document.write("Unknown grade<br />")
        }
        document.write("Exiting switch block");
      //-->
    </script>
    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

Output

```
Entering switch block

Good job

Exiting switch block

Set the variable to different value and then try...
```

❖ <u>Looping statement</u>

1. For…loop

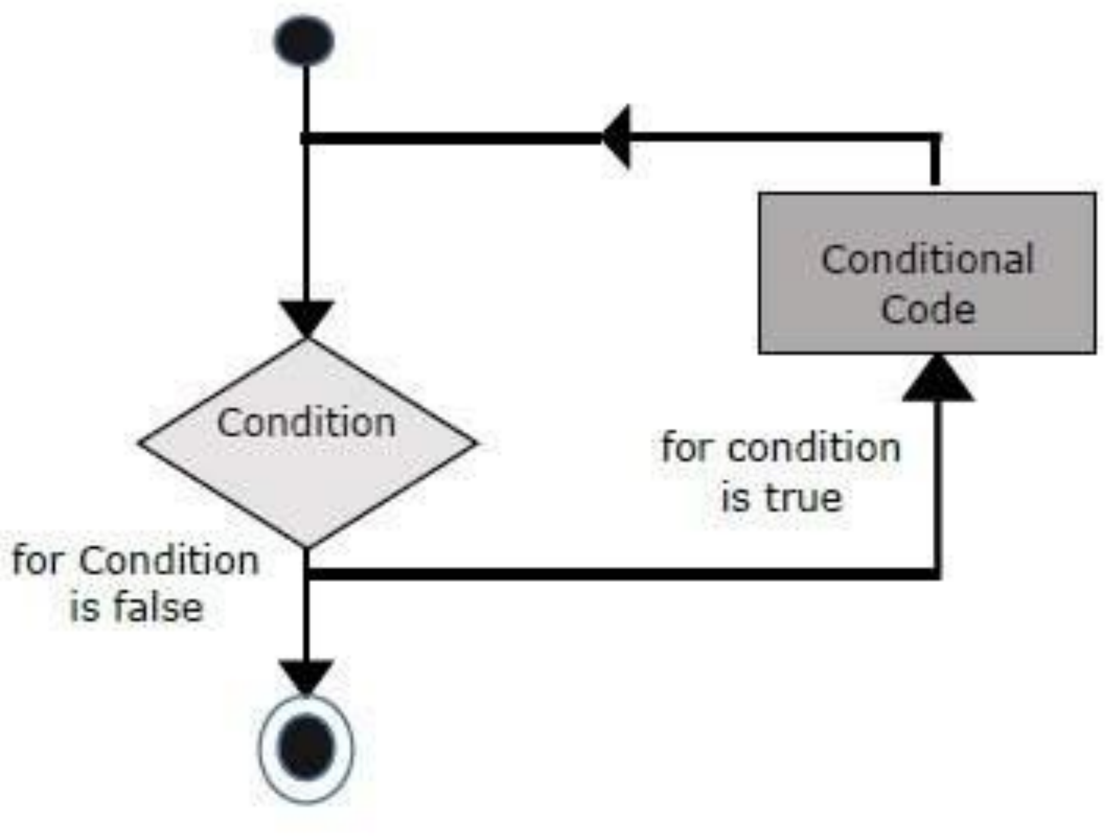
The '**for**' loop is the most compact form of looping. It includes the following three important parts −

- The **loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
- The **test statement** which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
- The **iteration statement** where you can increase or decrease your counter.

You can put all the three parts in a single line separated by semicolons.

Flow Chart

The flow chart of a **for** loop in JavaScript would be as follows −



Syntax

The syntax of **for** loop is JavaScript is as follows −

```
for (initialization; test condition; iteration statement)
 {
  Statement(s) to be executed if test condition is true
}
```

Example

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var count;
        document.write("Starting Loop" + "<br />");

        for(count = 0; count < 10; count++) {
          document.write("Current Count : " + count );
          document.write("<br />");
        }
        document.write("Loop stopped!");
      //-->
    </script>
    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

Output

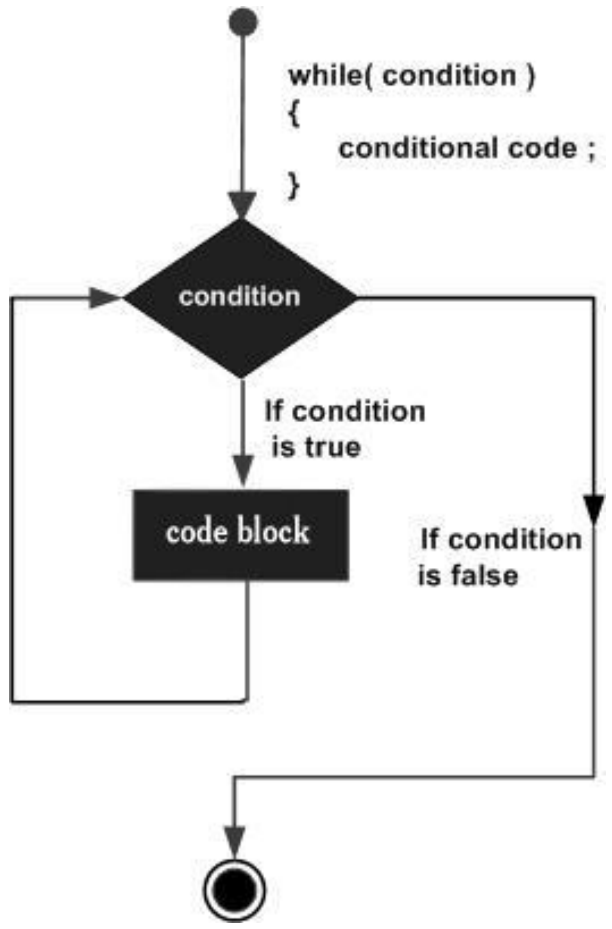
```
Starting Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Current Count : 5
Current Count : 6
Current Count : 7
Current Count : 8
Current Count : 9

Loop stopped!
```

2. The while Loop

- The most basic loop in JavaScript is the **while** loop which would be discussed in this chapter.
- The purpose of a **while** loop is to execute a statement or code block repeatedly as long as an **expression** is true. Once the expression becomes **false,** the loop terminates.

Flow Chart

The flow chart of **while loop** looks as follows −



Syntax

The syntax of **while loop** in JavaScript is as follows −

```
while (expression)
 {
  Statement(s) to be executed if expression is true
}
```

Example

```
<html>
  <body>

    <script type = "text/javascript">
      <!--
        var count = 0;
        document.write("Starting Loop ");

        while (count < 10) {
          document.write("Current Count : " + count + "<br />");
          count++;
        }

        document.write("Loop stopped!");
      //-->
    </script>

    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

Output

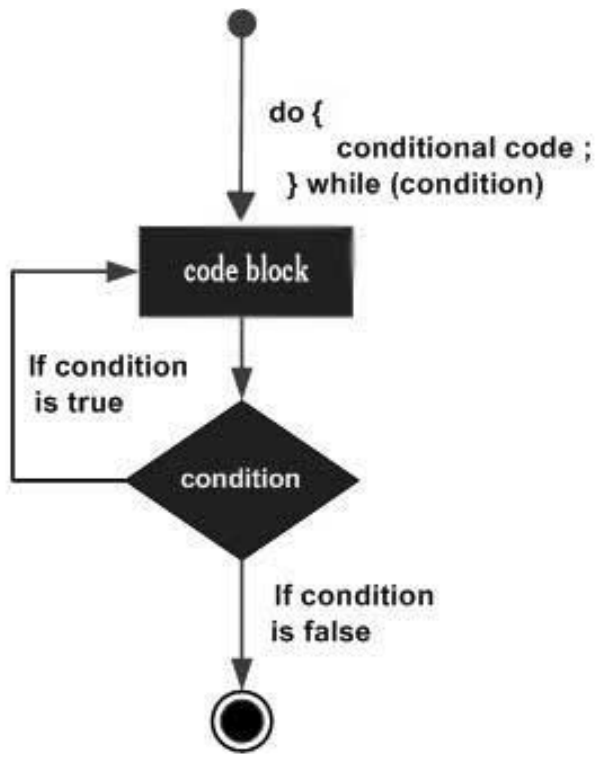
```
Starting Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Current Count : 5
Current Count : 6
Current Count : 7
Current Count : 8
Current Count : 9
Loop stopped!
Set the variable to different value and then try...
```

3. The do...while Loop

- The **do...while** loop is similar to the **while** loop except that the condition check happens at the end of the loop.
- This means that the loop will always be executed at least once, even if the condition is **false**.

Flow Chart

The flow chart of a **do-while** loop would be as follows −



Syntax

The syntax for **do-while** loop in JavaScript is as follows −

```
Do
 {
  Statement(s) to be executed;
}
while (expression);
```

Note − Don’t miss the semicolon used at the end of the **do...while** loop.

Example

```
<html>
  <body>
    <script type = "text/javascript">
      <!--
        var count = 0;

        document.write("Starting Loop" + "<br />");
        do {
          document.write("Current Count : " + count + "<br />");
          count++;
        }

        while (count < 5);
        document.write ("Loop stopped!");
      //-->
    </script>
    <p>Set the variable to different value and then try...</p>
  </body>
</html>
```

Output

```
Starting Loop
Current Count : 0
Current Count : 1
Current Count : 2
Current Count : 3
Current Count : 4
Loop Stopped!
Set the variable to different value and then try...
```

UNIT-3

❖ Arrays

- Arrays are JavaScript objects that are capable of storing a sequence of values.
- These values are stored in indexed locations within the array.
- The length of an array is the number of elements that an array contains.
- The individual elements of an array are accessed by using the **name of the array** followed by the index value of the array element enclosed in square brackets [ ]
- The array element index starts with **0**.
- The last array index number is **one less** than of the length array.

Declare/create an array:

- **Syntax:**

  var array name=new Array (size);

  var array name=new Array();

- **Example:**

  var spcam = new Array(5)

  spcam[0] = bca;

  array name= name of the array variable

  size= number of elements value to be store.

- The first method will created with length size (for example new Array(5) size= 5).
- The second array with 0 size. (For example new Array() size=0).

Assign values to an array:

- An array can be assigned values using the = operator. Each element must be referenced using its index value.

Syntax:

Array name[index] = value;

Example:

var spcam = new Array(3)

spcam[0] =” BCA”;

spcam[1] = “BBA”;

spcam[2] = “BCOM”;

- It will assign BCA to the 1 element with the index [0], BBA to the 2 element with index [1] and BCOM to the 3 element with the index [2].

- **Initialize/Dense Arrays:**
  - A dense array is an array that has been created with each of its elements being assigned a specific value.
  - Dense arrays are used exactly in same manner as other array.
  - They are declared and initialized at the same time.

.

Syntax:

var array name= new Array(value0,value1…..value n)

Example:

var spcam=new Array(10,20,”BCA”)

In this array, the element count start from 0 to n, the array length is n+1

❖ User-defined functions

- Function is group of JavaScript program code that perform a specific task into a Single unit that can be used repeatedly whenever required in a JavaScript program.

<u>Function definition/declare/create</u>:-

- Functions are declaring and created using the **function** keyword.
- A name for the function.
- A list of parameters (arguments).
- A block of JavaScript code. (Function body)

Syntax:

Function function name(parameter1,parameter2,…..)

{

Block of JavaScript code

}

Function parameter passing:

- A list of parameters (arguments) that will accept values passed to the function when called.

Function call:-

- The function can be called by using the function name followed by list of paraeteters.

Example:

Calling function from an on click event handler

```
<html>
<head>
<script type=”javascript”>
Function showmsg(msg)          //function declaration & parameter pass
{

    Alert(“the button sent sent:” +msg);   //function body
}
</script>
</head>
<body>
<form>
<input type=”button” value =”click me” onclick=”showmsg(‘The button has been clicke!’)”>
//function is call
</form>
</body>
</html>
```

Return values:-

- User-define function can return values. Such values can be return using the **return** statement.

Example:

```
 Function cube(number)
{
    var cube =number*number*number;
    return cube;
 }
```

❖ String Object

- The String object is used to manipulate a stored piece of text.
- String objects are created with **new** String ().

Syntax:

```
var variable name = new String("string");
```

Example:

```
var a = new String(”spcam”);
 alert(a);
```

OUTPUT:

```
spcam
```

1. Length Property

- The length property returns the number of characters in a string.
- It counts spaces and punctuation symbol as characters.

Syntax:

```
Objectname(variable name).length
```

Example:

Return the number of characters in a string:

```
<script type="text/javascript">
var a = "spcam";
alert(a.length);
</script>
```

OUTPUT: 5

String Object Methods

1. charAt

- The charAt() method returns the **character** at the specified **index position** in a string.
- The index of the **first character** is **0** and the index of the **last character** in a string **variable name.length-1.**

Syntax:

variablename.charAt(index position) // check the **first character** index position.

variablename.charAt(variable name.length-1) // check the **last character** index position

- **Example:**

```
<script type="text/javascript">
var str = "Hello world!";
document.write("First character: " + str.charAt(0) );
</script>
```

The output of the code above will be:

```
First character: H
```

2. indexOf

- The index of () method returns the **character position** of a specified value in a string.
- The character positions start at **0**.
- if the search characters no match in string, this method return **-1**.

Syntax:

variable name. index Of (search string, start)

Example:

```
<script type="text/javascript">
var str="Hello world!";
```

```
document.write(str.indexOf("d") + "<br />");
document.write(str.indexOf("world"));
</script>
```

The output of the code above will be:10, 6

3. substr

- The substr() method extracts(cut) the characters from a string and returns the new sub string.
- These methods specify index position for **start point** of string value and index position for **ending point (length of string)** of string value.
- It counts spaces and punctuation symbol as characters.

Syntax:

Variable name.substr (start, length)

Example:

```
<script type="text/javascript">
 var str="Hello world!";
 document.write(str.substr(3)+"<br />");
 document.write(str.substr(3,4));
 </script>
```

The output of the code above will be:

```
lo world!
lo w
```

4. toLowerCase

- The toLowerCase() method converts a string to lowercase letters.

Syntax:

- variablename.toLowerCase ()

Example:

```
<script type="text/javascript">
var str="Hello World!";
document.write(str.toLowerCase());
</script>
```

5. toUpperCase

- The toUpperCase() method converts a string to uppercase letters.

Syntax:

- variable name.toUpperCase()

Example:

```
<script type="text/javascript">
 var str="Hello world!";
document.write(str.toUpperCase());
</script>
```

❖ Math Object

- The Math object allows you to perform mathematical tasks.
- The Math objects are created with **new Math()**

Syntax:

```
var a = new Math();
```

Example:

```
var a=100,b;
var b = Math.sqrt(a);
alert(b);
```

PI Property:

- The PI property returns the ratio of a circle's area to the square of its radius, approximately 3.14159.

Syntax:

```
Math.PI
```

Example:

```
<script type="text/javascript">
     document.write("PI: " + Math.PI);
     </script>
```

The output of the code above will be:

```
PI: 3.141592653589793
```

1. Abs()

- The abs() method returns the **absolute value**(positive value) of a number.

Syntax:

- Math.abs(value/variable name)

Example:

```
<script type="text/javascript">
 document.write(Math.abs(7.25));
document.write(Math.abs(-7.25));
document.write(Math.abs(null) );
document.write(Math.abs("Hello"));
document.write(Math.abs(7.25-10));
</script>
```

The output of the code above will be:

```
7.25,7.25,0,NaN,2.75
```

2. Ceil()

- The ceil() method rounds a number **UPWARDS**(round up) to the nearest integer, and returns the result.

Syntax:

Math.ceil(value/variable name)

Example:

```
<script type="text/javascript">
document.write(Math.ceil(0.60));
document.write(Math.ceil(0.40) );
document.write(Math.ceil(5) );
document.write(Math.ceil(5.1));
document.write(Math.ceil(-5.1));
document.write(Math.ceil(-5.9));
</script>
```

The output of the code above will be:

```
1,1,5,6,-5,-5
```

3. Floor()

- The floor() method rounds a number **DOWNWARDS**(round down) to the nearest integer, and returns the result.

Syntax:

Math.floor(value/variable name)

Example:

```
<script type="text/javascript">
document.write(Math.floor(0.60));
document.write(Math.floor(0.40) );
document.write(Math.floor(5));
document.write(Math.floor(5.1));
document.write(Math.floor(-5.1));
document.write(Math.floor(-5.9));
</script>
```

The output of the code above will be:

```
0,0,5,5,-6,-6
```

4. Max()

- The max() method returns to find the number with the **highest(maximum)** value in the list.

Syntax :

Math.max(x,y,z,...,n)

Example:

```
<script type="text/javascript">
document.write(Math.max(5,10));
document.write(Math.max(0,150,30,20,38));
document.write(Math.max(-5,10));
document.write(Math.max(-5,-10));
document.write(Math.max(1.5,2.5));
</script>
```

The output of the code above will be:

```
10,150,10,-5,2.5
```

5. Min()

- The min() method returns to find the number with the **lowest(minimum)** value in the list.

Syntax :

```
Math.min(x,y,z,...,n)
```

Example:

```
<script type="text/javascript">
document.write(Math.min(5,10));
document.write(Math.min(0,150,30,20,38));
document.write(Math.min(-5,10));
document.write(Math.min(-5,-10));
document.write(Math.min(1.5,2.5));
</script>
```

The output of the code above will be:

```
5,0,-5,-10,1.5
```

6. Round()

- The round () method return the **round** a number to the nearest integer.
- It **round up** only if the decimal part is 5 or greater than other wise it return **round down**

Syntax:

```
Math.round(value/variable)
```

Example:

```
<script type="text/javascript">
document.write(Math.round(0.60));
document.write(Math.round(0.50) );
document.write(Math.round(0.49) );
document.write(Math.round(-4.40));
document.write(Math.round(-4.60));
</script>
```

The output of the code above will be:

```
1,1,0,-4,-5
```

❖ Date Object

- ➢ The Date object is used to work with dates and times.
- ➢ Date objects are created with **new Date().**

◊ GET methods

1. getDate()

- The getDate() method returns the **day of the month** (from 1 to 31) for the specified date, according to local time.

Syntax:

Variable name.getDate()

Example:

```
<script type="text/javascript">
var d = new Date();
document.write(d.getDate());
</script>
```

The output of the code above will be:18

2. getDay()

- The getDay() method returns the **day of the week** (from 0 to 6) for the specified date, according to local time.

Syntax:

Variable name.getDay()

Example:

```
<script type="text/javascript">
var d = new Date();
document.write(d.getDay());
</script>
```

The output of the code above will be:3

3. getFullYear()

- The getFullYear() method **returns the year** (four digits) of the specified date, according to local time

Syntax:

Variable name .getFullYear()

Example:

```
<script type="text/javascript">
var d = new Date();
document.write(d.getFullYear());
</script>
```

The output of the code above will be:2012

4. getMonth()

- The getMonth() method **returns the month** (from 0 to 11) for the specified date, according to local time.

Syntax:

Variable name.getMonth()

Example:

```
<script type="text/javascript">
var d = new Date();
document.write(d.getMonth());
</script>
```

The output of the code above will be:0

5. getTime()

- The getTime() method returns the number of **milliseconds** since midnight of January 1, 1970 and the specified date.

Syntax:

Variable name.getTime()

Example:

```
<script type="text/javascript">
    var d = new Date();
   document.write(d.getTime() + " milliseconds since 1970/01/01");
   </script>
```

Output: 1326898266603 milliseconds since 1970/01/01

6. getHours()

- The getHours() method returns the **hour** (from 0 to 23) of the specified date and time, according to local time.

Syntax:

Variable name.getHours()

Example :

```
<script type="text/javascript">
 var d = new Date();
document.write(d.getHours());
</script>
```

The output of the code above will be:20

7. getMinutes()

- The getMinutes() method returns the **minutes** (from 0 to 59) of the specified date and time, according to local time.

Syntax:

Variable name.getMinutes()

Example:

```
<script type="text/javascript">
var d = new Date();
document.write(d.getMinutes());
</script>
```

The output of the code above will be:24

8. getSeconds()

- The getSeconds() method returns the **seconds** (from 0 to 59) of the specified date and time, according to local time.

Syntax:

Variable name.getSeconds()

Example:

```
<script type="text/javascript">
var d = new Date();
document.write(d.getSeconds());
</script>
```

Output: 21

◊ SET methods

1. setDate()

- The setDate() method sets the day of the month (from 1 to 31), according to local time

Syntax:

```
Variable name.setDate(day)
```

Example:

```
<script type="text/javascript">
    var d = new Date();
    d.setDate(15);
    document.write(d);
  </script>
```

The output of the code above will be:

```
Sun Jan 15 20:30:03 UTC+0530 2012
```

2. setFullYear()

- The setFullYear() method sets the year (four digits), according to local time.

Syntax:

```
Variable name.setFullYear(year,month,day)
```

Example:

```
<script type="text/javascript">
     var d = new Date();
     d.setFullYear(2020);
      document.write(d);
     </script>
```

The output of the code above will be:

```
Sat Jan 18 20:31:32 UTC+0530 2020
```

3. setMonth()

- The set Month () method sets the month (from 0 to 11), according to local time.

Syntax:

Variable name.setMonth(month,day)

Example:

```
<script type="text/javascript">
      var d = new Date();
      d.setMonth(0);
      document.write(d);
     </script>
```

The output of the code above will be:

```
Wed Jan 18 20:33:45 UTC+0530 2012
```

4. setTime()

- The set Time() method sets a date and time by adding or subtracting a specified number of milliseconds to / from midnight January 1, 1970.

Syntax :

Variable name.setTime(millisec)

Example :

```
<script type="text/javascript">
var d = new Date();
d.setTime(77771564221);
document.write(d);
</script>
```

The output of the code above will be:

```
Mon Jun 19 05:12:44 UTC+0200 1972
```

5. setHours()

- The setHours() method sets the hour (from 0 to 23), according to local time.

Syntax:

Variable name.setHours(hour,min,sec,millisec)

Example:

```
<script type="text/javascript">
var d = new Date();
d.setHours(15);
document.write(d);
</script>
```

The output of the code above will be:

```
Wed Jan 18 15:36:04 UTC+0530 2012
```

6. setMinutes()

- The setMinutes() method sets the minutes (from 0 to 59), according to local time.

Syntax :

Variable name.setMinutes(min,sec,millisec)

Example:

```
<script type="text/javascript">
var d = new Date();
d.setMinutes(1);
document.write(d);
</script>
```

The output of the code above will be:

```
Wed Jan 18 20:01:59 UTC+0530 2012
```

7. setSeconds()

- The setSeconds() method sets the seconds (from 0 to 59), according to local time.

Syntax :

Variable name.setSeconds(sec,millisec)

Example:

```
<script type="text/javascript">
 var d = new Date();
  d.setSeconds(1);
  document.write(d);
    </script>
```

The output of the code above will be:

```
Wed Jan 18 20:39:01 UTC+0530 2012
```

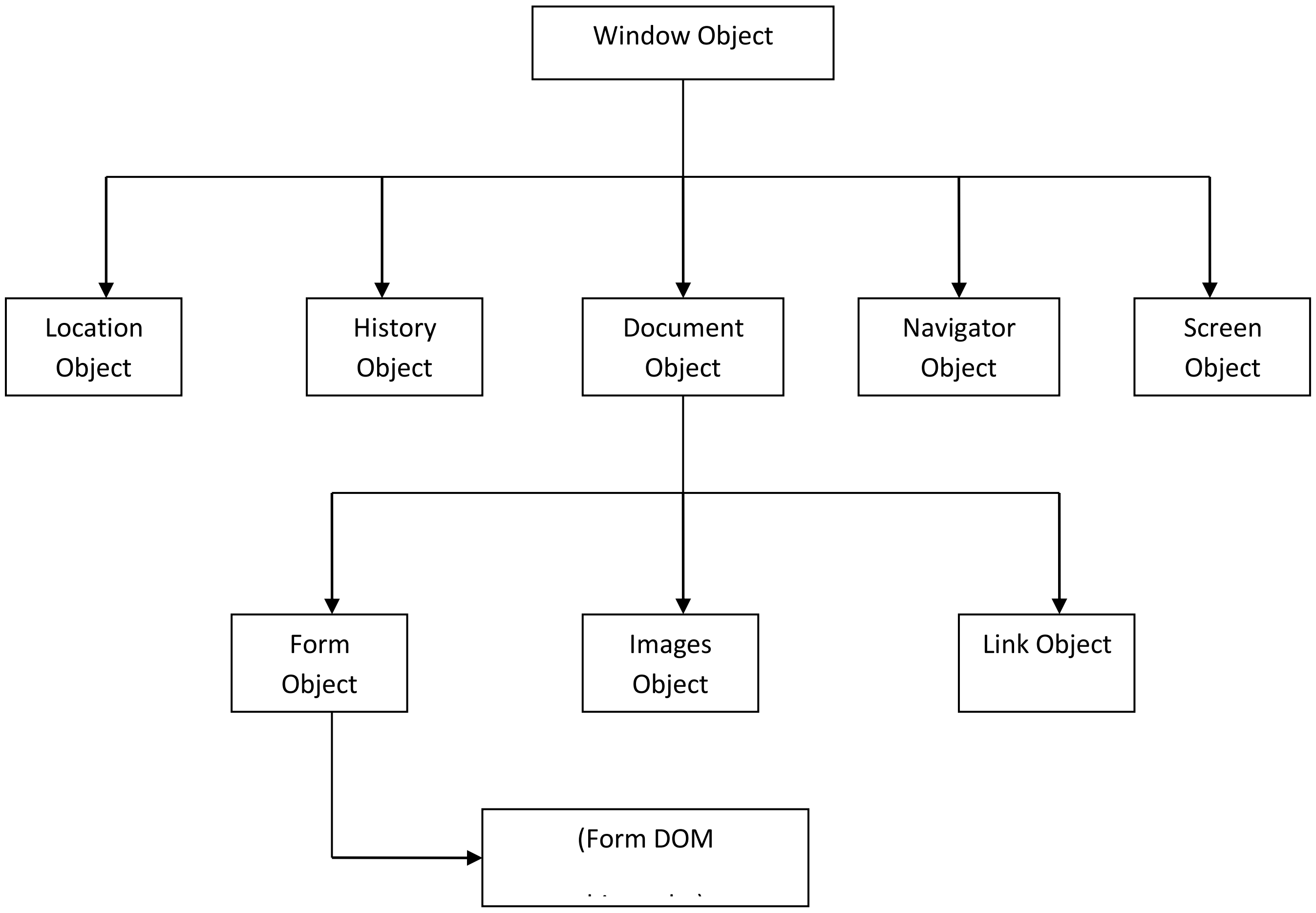
UNIT-4

❖ Introduction to DOM

- The Document Object Model (DOM) is a programming API for HTML and XML documents.
- It defines the logical structure of documents and the way a document is accessed and manipulated.
- The way document content is accessed and modified is called the Document Object Model, or DOM.
- With the Document Object Model, programmers can create and build documents, navigate their structure, and add, modify, or delete elements and content.
- Anything found in an HTML or XML document can be accessed, changed, deleted, or added using the Document Object Model.
- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- With the HTML DOM, JavaScript can access and change all the elements of an HTML document.

❖ DOM Hierarchy and Understanding objects & Collections in DOM

- Dom Hierarchy it forms like structure, the top most objects is called the root object.
- The root object is the window object, all the objects below are called child object.
- An object above child object is called the parents object.



1. Window Object:-

- Window object is a global object. Some of the properties of this object like document, history, screen and location are also objects which own properties or method.
- Window object allow you can find out what browser is running page of users has visited, the size of browser window, the size of user screen, resolution of user screen.

➢ **Syntax**

window. Properties name
window.method name(parameters)

➢ **Window Method/property:**

1. Alert Method
2.Prompt Method
3.Default status property

- **Alert Method:-**

➢ An alert box can be used to display a message (string passed to the alert method) or display some information

- **Syntax:**

window. alert("text value");

➢ **Example:**

window. alert("window object");

- **Prompt Method:-**

➢ The Prompt box displays a predefine message.

➢ The prompt box also displays a single data entry field (textbox), which accepts user input.

➢ **Syntax**

window.method name(parameter);

Example

var a;
a.window.prompt("Enter the number");
alert("hello"+a);

- **Default Status property:-**

➢ This property use to change default message in the window status bar

- **Syntax**

window.defaultstatus="Some Value"

- **Example**

```
window.defaultstatus="welcome"
```

2. History Object:

- History object contain the history of page visited by the users
- Uses to click the browsers back a forward button to revisited page

- **Syntax**

```
window.history.propertyname
window.history.methodname(parameter)
```

- **Go method**
  - This method specify how far forward or backward in the history stack you want go.
  - **Syntax**

    ```
    window.history.go(some number);
    ```

  - **Example**

    ```
    window.history.go(2); // Forward page
    window.history.go(-2); //Backward page
    ```

- **Back Method**
  - This method displays the previous (back) page the user had visited in the browser.
  - **Syntax**

    ```
    window.history.back();
    ```

  - **Example**

    ```
    window.history.back();
    ```

- **Forward Method:**
  - This method display the next (forward) page the user had visited in the browser.
  - **Syntax**

    ```
    window.history.forward();
    ```

  - **Example**

    ```
    window.history.forward();
    ```

3. Location Object

- Location object contain useful information about the current page.
- Location includes URL for the page. Server hosting the page/port number of the server connection and protocol is used.

➢ **Syntax**

window.location.property name;

window.location.method name(parameters)

- **href properties**
  - ➢ This property allows navigating to another page.
  - ➢ This property using add new page to the top of the history stack
  - ➢ It can be used to fetch the current path from the address bar as well as change the page which is currently displayed to another page.

➢ **Syntax**

window.location.**href**=”Some URL”;

➢ **Example**

window.location.href=”http\\google.com”;

- **Replace Method**
  - ➢ These methods remove the current page from history stack and add the new page to the top at history stack.

➢ **Syntax**

window.location.**replace**(“some URL”);

➢ **Example**

window.location.replace(“http:\\www.google.com”);

4. Screen Object

- This object contains a lot of information about the display capability of client site machine.
- Screen object property like height, width color depth which indicate the screen sizes (pixel) and the number of bits used to color value.

➢ **Syntax**

window.screen.property/method name.

- **Height/Width property:**
  - ➢ This indicates vertical & horizontal range of the screen.

➢ **Syntax**

window.screen.height

- **Color Depth property:**
  - ➢ It tells us the number of bits used for color on the client screen.
  - ➢ **Syntax**

    window.screen.colordepth

5. Document object:

- Document object has a number of properties associated with it, which are also arrays.
- The main ones are the forms, images and links arrays.
- A number of other array properties such as the **all collection property** which is an array of all the elements represent by object in the page.

➢ **Syntax:**

window.document.property name

window.document.method name(parameters/value)

- **Write method:**

➢ These method uses display the content as the page load in the browser.

➢ **Syntax:**

window.document.**write**(some value);

➢ **Example:**

window.document.**write**("spcam");

- **bgcolor property:**

➢ It changes the background color of the web page after it is loaded.

➢ **Syntax:**

window.document.bgcolor=color value;

➢ **Example:**

window.document.bgcolor=red;

- **Image collection(using ARRAY):-**

➢ All the <img> tags are used in the html page are accessible using the **images collection**

- Each <img> tag creates an **Img object**, and group of **img objects** from the **images collection**, which is the property of the document object.
- **Images** collection can be used as an **images** array with the first <img> tag at the index position **0** or the name of image.
- The individual objects can be accessed using either the index position or the name of image.

- **Syntax:**

  window.document.images[Index/Name].method Name[(parameter)]

  OR

  window.document.images.method Name[(parameter)]

- **Syntax:**

  window.document.images[Index/Name].property name

  OR

  window.document.images.property name

◊ **Length Property:-**

- This property returns the number of images present in the images collection.

- **Syntax:**

window.document.images.length

- **Example:-**

```
var a;
a= window.document.images.length
alert (“there are ” + a +” image in page”);
```

◊ **src Property:-**

- This property is used to fetch the src attributes value for any image
- This property is used to display a new image in place of the old image
- It is also used to change the source of any image and create the effect of swapping images.

- **Syntax:-**

window.document.images.[index/name].**src**

- **Example:-**

```
var a;
a= window.document.images.[0].src;
  alert(a);
window.document.images.[0].src=”sunset.jpg”
```

- **Link Collection(using ARRAY):-**
  - ➢ All the <a> tags used in the html pages are accessed using **links collection**.
  - ➢ Each <a> tag, create and **a object**, and group of **a objects** form the links collection, which is the property of the document object.
  - ➢ The link collection can then be used as links array with the first <a> tag of the **index position 0** in the link array

  - ➢ **Syntax:-**

    window.document.links[index.].method name(parameter)

    window.document.links[index.].property name

◊ **href Property:-**

- ➢ Using this property you can find out where the link points to and this can be used fetch or change the destination URL of a link.

- ➢ **Syntax:-**

  window.document.link[index].**href**=.”URL address”;

  window.document.link[0].href=http://www.yahoo.com;

- **Form Collection:-**
  - ➢ All the <form> tags used in html page are accessible using the **forms** collection.
  - ➢ Each<form> creates a “**form object**” and group of form objects creates the **forms** collection which is the property of the document object.
  - ➢ The forms collection can be used from array with the first<form> tag at the **index position 0** in the form array

  - ➢ **Syntax:**

  window.document.forms.[index].propertyname

  OR

  window.document.forms.[“formname”].propertyname

  OR

  window.document.formname.propertyname

- **All collection:-**
  - ➢ This collection is the group of all elements in html.it is accessed through the **all** array.
  - ➢ This property use access group of any elements by using **ID** identifier in the form.
  - ➢ Id identifier use to access any element that has the assigned as its name of the elements in index.

- **Syntax:-**

window.document.**all**[“id_name”].propertyname.

- **Example:-**
  var a;
  window.document.**all**[“id_btn”].value=”changed”
  a= window.document.**all** [“id_text”].value;

- **getElementById:**

  - This collection is the group of all elements in html.it is accessed through the **getElementBy** array
  - This property use access group of any elements by using **ID** identifier in the form.
  - **Id** identifier use to access any element that has the assigned as its name of the elements.

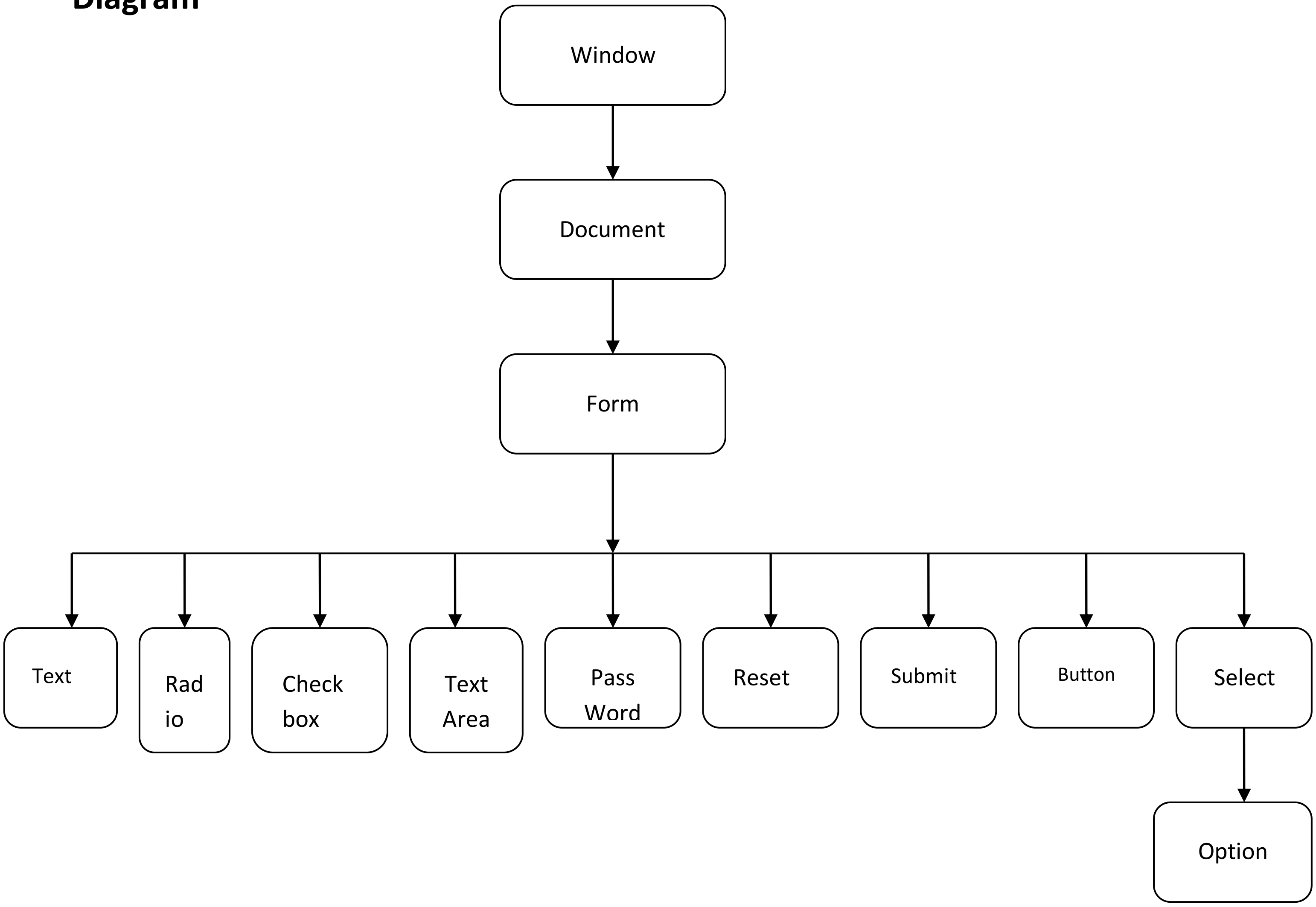
  - **Syntax:-**
  window.document. **getElementBy** [“id_name of element”].propertyname.

  - **Example:-**
  var a;
  window.document. **getElementBy**.[“id_btn”].value=”changed”
  a= window.document. **getElementBy.** [“id_text”].value;

❖ HTML Form Hierarchy

- Any documents can various html objects such as images, hyperlinks, frames, form with various form elements, etc.
- Form Dom hierarchy represent form with various form elements control such as
- Text element, radio element, button element, submit element, reset element etc.
- These form objects (element) have a method and properties that can be get and set using java script.
- Inside in the form access any elements using **form object** or form name and **elements property** or elements name and its property, method, event.
- Form element using **array property** to access all elements at the form using index. Ex text [0], text [1], text [2], text[n].
- All the form elements must be named and can also be accessed by their names in java script.
- Each form element can also be accessed as **form object**.
- The form element objects have properties like name, value, type, checked select index etc.
- The form elements object have methods/event like click(),focus(),bluer(),load() etc.

Diagram



Syntax:
window.document.forms name. elements [ ].property/method/event
OR
window.document.forms[ ].elements [ ].property/method/event
OR
window.document.forms [“name”].elements [ ].property/method/event
OR
window.document.formname.element name.property/method/event
OR
window.document.forms [ ].element name.property/method/event
OR
window.document.forms [“name”].element name.property/method/event

example:

```
window.document.frm.text1.value;
window.document.frm.radio1.checked;
window.document.frm.checkbox1.checked;
window.document.frm.button1.value;
window.document.frm.reset1.value;
```

window.document.frm.submit1.value;
window.document.frm.password1.value;

❖ Form:

- The forms array is used to access all forms within a single document.
- A form can referenced using the forms array or using the form name.
- Form has three properties.

1. **Name property:**

- This property returns the name of the form which is used to reference the form in script.

- **Syntax:**

Some_var=window.document.forms[index].**name** ;
or
Some_var=window.document.forms[“form name”].**name** ;

- **Example:**

```
var a;
a=  window.document.forms[0].name ;
alert(a);
```

OUTPUT:frm

2. **Method property:**

- This property returns the method of transmission used to send data captured by the various form elements back to the web server.

- **Syntax:**

Some_var=window.document.forms[index].**method** ;
or
Some_var=window.document.forms[“form name**”].method** ;
or
Some_var=window.document.formname**. method** ;

- **Example**

```
var a;
 a=  window.document.forms[0].method ;
alert(a);
```

OUTPUT: get/post

3. **Action property:**

- These properties return the entire file path of submission or return the URL address of a program on the web server.

- **Syntax:**

Some_var=window.document.forms[index].**action** ;

or
Some_var=window.document.forms["form name"].**action** ;
or
Some_var=window.document.formname**. action**;

- **Example:**
  var a;
  a= window.document.forms[0].action ;
  alert(a);

OUTPUT:c\html\e1.html

❖ Accessing Form elements (Text, Radio, Checkbox, Dropdown, Button)

1. Text control:

- This element access through the forms using index or form name by using the name of the element.
- It is used to enter a single line of text.

- **Syntax:**

  window.document.forms[index].**textbox name**
  or window.document.forms["formname"].**textbox name**
  or window.document.formname.**textboxname**

- **example:**
  var a;
  a= window.document.forms[0].txt1.value;
  alert(a);

- **value property:**

  - This property return string value entered in the textbox by the user.
  - This can be used to read or write value in the textbox.

  - **syntax:**
    var_name=window.document.forms[index].textboxname.**value()**;
    var_name=window.document.forms["formname"].textboxname.**value;**
    var_name=window.document.formname.textboxname.**value**;

- **example**:
  var a;
  a= window.document.frm.text1.value;
  alert(a);

- **focus() method:**

➢ This event is used to set the focus onto a textbox using script.
When an object has focus, the text insertion pointer will be linking in that control.

➢ **Syntax:**

var_name=window.document.forms[index].textboxname.**focus()**;
var_name=window.document.forms[“formname”].textboxname.**focus();**
var_name=window.document.formname.textboxname.**focus()**;

➢ **example:**

```
var a;
 a= window.document.frm.text1.value;
 if(a==” ”)
   window.document.frm.text1.focus();
```

2. radio control:

➢ This element access through the forms using index or form name by using the name of the element.
➢ They can access be accessed as an array with the first index [ ] as 0.
➢ Entire group of radio buttons allows accessing value or check which one is selected.
➢ Entire group of radio buttons must have same name.

➢ **Syntax:**

window.document.forms[index].**radiogroupname**[index].checked/value;

OR

window.document.forms[“forename”].**radiogroupname**[index].checked/value;

OR

window.document.formname.**radiogroupname**[index].checked/value;

➢ **Example:**

```
var a;
if (window.document.frm.rd[0].checked=true)
  alert(“first radio select”)
```

- **checked property:**

➢ This property return **true or false** depending on whether that radio button is checked or not.
➢ All the other radio button have unchecked this property return value false or set to false.

➢ **Syntax:**

window.document.forms[index].radiogroupname[index]**.checked**;

OR

window.document.forms[“formname”].radiogroupname[index]**.checked**;

OR

window.document.formname.radiogroupname[index]**.checked**;

- **Example:**

```
var a;
if(window.document.frm.rd[0].checked==true)
alert("first radio button is selected");
```

OUTPUT: first radio button is selected

- **value property:**
  - This property returns the **values** of the radio button which is checked in the group that means to find out which radio button was selected.

  - **Syntax:**

varname=window.document.forms[index].radiogroupname[index]**.value**;

OR

varname=window.document.forms["formname"].radiogroupname[index]**.value**

OR

varname=window.document.formname.radiogroupname[index]**.value**;

Example:

```
var a, b;
if(window.document.frm.rd[0].checked==true)

        a= window.document.frm.rd[0].value;
         alert("first radio select")
else
        b= window.document.frm.rd[1].value;
          alert("second radio select");
```

OUTPUT: second radio button is selected

3. checkbox control:

- This element access through the forms using index or form name by using the name of the element.
- It indicates that the checkbox is true if it is selected, otherwise false.
- Entire group of checkbox must have different name.
- They can be accessed using an array with first index as [0] and all the checkbox in group must have the same name**.(using array)**

- **Syntax:**

window.document.forms[index].**checkboxgroupname[index]**.value/checked;

OR

window.document.forms["formname"]. **checkboxgroupname [index]**.value/checked;

OR

window.document.formname**. checkboxgroupname [index].**value/checked;

- **example:**

```
var a;
if(window.document.frm.chk1[0].checked==true)
alert("first check box is selected");
```

- **checked property:**
- This property returns **true or false** depending on whether the checkbox button is checked or not.

- **Syntax:**

window.document.forms[index].checkboxgroupname[index].**checked;**

OR

window.document.forms["formname"]. checkboxgroupname [index].**checked;**

OR

window.document.formname. checkboxgroupname [index**].checked;**

- **example:**

```
var a;
if(window.document.frm.chk1[0].checked==true)
alert("first check box is selected");
```

OUTPUT: first check box is selected

- **value property:**
- this property return the **value** associated with that checkbox that means
- It will return value checkbox selected or not if checkbox is selected it will return value otherwise not return value .

- **Syntax:**

window.document.forms[index].checkboxgroupname[index].**value;**

or

window.document.forms["formname"]. checkboxgroupname [index].**value;**

or

window.document.formname. checkboxgroupname [index**].value;**

- **example:**

```
var a;
if(window.document.frm.chk1[0].checked==true)
{
        a= window.document.frm.chk1.value;
        alert(a);
}
```

OUTPUT: male

4. Select/dropdown control:

- This element access through the forms using index or form name by using the name of the element.
- It appears as a drop-down list or a scrollable list or selectable items.

- **Syntax:**

  window.document.forms[index].**select name**

  OR

  window.document.forms[“formname”].**select name**

  OR

  window.document.formname.**selectname**

- **select index property:**
  - This property returns an integer (starting from 0 for the 1 option) indicate the selected list items index value.

  - **Syntax:**

    var_name=window.document.forms[index].selectname**.selectindex;**

    OR

    var_name= window.document.forms[“formname”].select name. **select index;**

    OR

    var_name= window.document.formname.selectname. **selectindex;**

  - **Example:**

    var a;

    a= window.document.frm.sel.selectindex;

    alert(a);

- **options[ ]:**
  - it is used to access all option element within the entire select object.
  - Each list item is including as an array element starting at [0] index.

  - **Syntax:**

    var_name=window.document.forms[index].selectname**.options[index].**text/value**;** **OR**

    var_name=window.document.forms[“formname”].selectname.**options[index].**text/value **;** **OR**

    var_name=window.document.formname.selectname.**options[index].**text/value**;**

  - **example:**

    var a,b;

    a= window.document.frm.sel.selectindex ;

    b= window.document.formname.selectname.options[a]**.**text;

    alert(b);

5. Button:

- The Input Button object represents an HTML <input> element with type="button".
- You can access an <input> element with type="button" by using getElementById():

Example

```
var x = document.getElementById("myBtn");
```

- **value Property**

  - The value property sets or returns the value of the value attribute of an input button.
  - The value attribute defines the text that is displayed on the button.

Syntax

Return the value property:
buttonObject.value

Set the value property:
buttonObject.value = text

Property Values

Value	Description
text	The text displayed on the button

Example

```
<script>

function myFunction() {

  var x = document.getElementById("myBtn").value;

  document.getElementById("demo").innerHTML = x;

}

</script>
```

❖ Event handling

❖ Event:

- Events occur when something in particular happens such as call a particular function or code when that event occurs.

- For example, the user clicking on **the** on a hyperlink, or moving the mouse pointer over some text all cause events to occur.

❖ Event Handling:

- ➢ Event handler is a code can be executed directly when the event is occurs.
- ➢ A web page event is associated with the action of the user with the mouse cursor or the keyboard on the web page. **Such example** on click event fire when mouse-click on an object on web page.
- ➢ Event handlers are created by adding **on** word before the event name.
- ➢ JavaScript event handlers divided into two types

- **Interactive:**
  - ➢ it depends on user interaction with an html page.
  - ➢ e.g:onclick, onmouseover, onmouseout etc.

- **Non interactive:**
  - o this event does not need user interaction.
  - o e.g onload event

1. onclick Event

➢ The on click event occurs when the user clicks on an element.

➢ **Syntax**

<input type=element onclick="SomeJavaScriptCode">

➢ **Example:**

< input type=button onclick="copyText()">Copy Text</button>

2. onblur Event

➢ The onblur event occurs when an object loses focus.

➢ **Syntax**

< input type=element onblur="SomeJavaScriptCode">

➢ **Example**

<input type="text" id="fname" onblur="upperCase()">

3. onfocus Event

➢ The on focus event occurs when an element gets focus.

➢ **Syntax:**

< input type=element onfocus="SomeJavaScriptCode">

➢ **Example**

<input type="text" id="fname" onfocus="setStyle(this.id)">

4. onload Event

- The onload event occurs when an object has been loaded.

- **Syntax:**
  < input type=element onload="SomeJavaScriptCode">

- **Example**
  < input type=element onload="SomeJavaScriptCode">

  <body onload="load()">

5. onmousedown Event

- The onmousedown event occurs when a user presses a mouse button over an element.

- **Syntax**

< input type=element onmousedown="SomeJavaScriptCode">

- **Example**

<p onmousedown="mouseDown()">Click the text!</p>

6. onmouseup Event

- The onmouseup event occurs when a user releases a mouse button over an element.

- **Syntax**

< input type=element onmouseup="SomeJavaScriptCode">

- **Example**

<p onmouseup="mouseUp()">Click the text!</p>

7. onmouseout Event

- The onmouseout event occurs when a user moves the mouse pointer out of an element.

- **Syntax**

< input type=element onmouseout="SomeJavaScriptCode">

- **Example**

<img onmouseout="normalImg(this)" src="smiley.gif" alt="Smiley" />

8. onmouseover Event

- The onmouseover event occurs when a user mouse over an element.

- **Syntax**

```
< input type=elementonmouseover="SomeJavaScriptCode">
```

- **Example**

```
<img onmouseout="normalImg(this)" src="smiley.gif" alt="Smiley" />
```

9. onReset event

- The on Reset event handler is used to execute specified JavaScript code whenever the user resets a form by clicking a Reset button.

- **Syntax**

```
< input type=element onreset="SomeJavaScriptCode">
```

- **Example**

```
<FORM NAME="form1" onReset="alert('Reset to Book of the Month.')">
```

10.onSubmit event

- The on Submit event handler is used to execute specified JavaScript code whenever the user submits a form by clicking a submit button.

- **Syntax:**

```
< input type=element onSubmit ="SomeJavaScriptCode">
```

- **Example**

```
<FORM onSubmit="submitEvent()">
<FORM onSubmit="return validate(this)">
```

11.onunload Event

- The on unload event occurs before the browser closes the document.

- **Syntax**

```
< input type= element onunload="SomeJavaScriptCode">
```

- **Example**

```
<body onunload="OnUnload()">
    ...
    ...
</body>
```

SARDAR PATEL UNIVERSITY
Vallabh Vidyanagar
FYBCA (SEMESTER-II)
WEB APPLICATION DEVELOPMENT-II (US02CBCA23)
QUESTION BANK (w.e.f. March 2018)

UNIT-I: MULTIPLE CHOICE QUESTIONS

1.	__________ scripts are executed by the Browser.	
	a. Client-side	**b.** Server-side
	c. CSS	**d.** HTML
	ANSWER: A	

2.	Client-side scripts are written in __________	
	a. HTML	**b.** CSS
	c. DOM	**d.** JavaScript
	ANSWER: D	

3.	The main use of Client-side scripts is __________	
	a. Data input in form	**b.** Data validation
	c. Display messages	**d.** Generate HTML pages
	ANSWER: B	

4.	The code which is used to connect to the database must be __________	
	a. written in server-side scripts	**b.** written in client-side scripts
	c. written in CSS	**d.** executed by browser.
	ANSWER: A	

5.	The code written in __________ is visible to the visitor of the site.	
	a. CSS	**b.** server-side scripting language
	c. client-side scripting language	**d.** ASP
	ANSWER: C	

6.	The code written in __________ is not visible to the visitor of the site.	
	a. CSS	**b.** server-side scripting language
	c. client-side scripting language	**d.** HTML
	ANSWER: B	

7.	You need a __________ to execute a server-side script.	
	a. Browser	**b.** Client
	c. Server	**d.** Web server
	ANSWER: D	

8.	The scripts which executes based on user action or some condition are written in ________	
	a. CSS	**b.** JavaScript
	c. PHP	**d.** ASP
	ANSWER: B	

9.	Where can you write the JavaScript within your HTML code?

	a. In <HEAD> only	**b.** In <Body> only
	c. In <HEAD> and <BODY> both	**d.** None of these
	ANSWER: C	

10.	Which attribute of <SCRIPT> tag is used to specify the language of script?	
	a. language or Type	**b.** Name
	c. Id	**d.** None of these
	ANSWER: A	

11.	Javascript is an __________	
	a. Object-created language	**b.** Object-based language
	c. Object-interpreted language	**d.** None of these
	ANSWER: B	

UNIT-I: SHORT QUESTIONS

1.	What is Scripting?
2.	What is Client-side Scripting?
3.	What is Server-side Scripting?
4.	Explain the difference between Client-side and Server-side scripting in terms of its place of execution
5.	List two languages used in Client-side scripting and Server-side scripting
6.	Write a note on <script> tag.
7.	List 5 Applications of Javascript.
8.	List 5 Advantages of Javascript.

UNIT-I: LONG QUESTIONS

1.	Write a note on Scripting
2.	Write a note on Client-side scripting
3.	Write a note on Server-side scripting
4.	Differentiate between Server-side and Client-side scripting
5.	Discuss advantages of JavaScript in brief.
6.	Write a note on Applications of JavaScript

UNIT-II: MULTIPLE CHOICE QUESTIONS

1.	Which keyword is used to declare a variable as an integer?	
	a. int	**b.** num
	c. var	**d.** None of these
	ANSWER: C	
2.	Which of the following is an invalid data type?	
	a. Number	**b.** Boolean
	c. String	**d.** VarChar
	ANSWER: D	
3.	Which of the following is a valid variable name?	
	a. With	**b.** My%variable
	c. 99variable	**d.** Mystr
	ANSWER: D	
4.	How many operands are required for operater ++?	
	a. One	**b.** Two
	c. Three	**d.** Four
	ANSWER: A	
5.	Which one of the following is not Comparison operator?	
	a. ==	**b.** !=
	c. <	**d.** +=
	ANSWER: D	
6.	Which operator is used to concatenate two strings?	
	a. +	**b.** ++
	c. +=	**d.** None of these
	ANSWER: A	
7.	NaN means __________	
	a. Not allowed Number	**b.** Not a Number
	c. Non allowed Number	**d.** None of these
	ANSWER: B	
8.	Which of the following is not true for variables?	
	a. Variables can contain underscore	**b.** Variables can contain digit characters
	c. Variables are case-insensitive	**d.** Variables can be of any length
	ANSWER: C	
9.	When two operands of different types are used in an expression, they may be converted into a type applicable for the operator. This is called as __________	
	a. Literals	**b.** Conversion
	c. Tokens	**d.** Type Casting
	ANSWER: D	
10.	Fixed or constant values in an expression is called __________	
	a. Literals	**b.** Conversion
	c. Tokens	**d.** Type Casting
	ANSWER: A	
11.	Which of the following functions is used to convert the string into Integer number?	
	a. Integer()	**b.** intParse()
	c. parseInt()	**d.** None of these
	ANSWER: C	
12.	Which of the following sentence is true for isNaN() function?	
	a. It returns today's date.	**b.** It returns Yes or No.

	c. It returns True or False.	**d.** None of these
	ANSWER: C	
13	To display a message to the user, __________ can be used.	
	a. Alert()	**b.** Prompt()
	c. alert()	**d.** prompt()

UNIT-II: SHORT QUESTIONS

1.	List out primitive data types.
2.	Write down the rules for defining variable
3.	Write a note on Assignment operator.
4.	Write a note on Increment operator.
5.	Write a note on Concatenation operator.
6.	Write a note on **new** operator.
7.	Explain the % operator with an example.
8.	Explain about Prompt dialog box. (with example)
9.	Explain about Alert dialog box. (with example)
10.	Write a note on Decrement operator.
11.	Explain the parseInt() method
12.	Explain the Number() method
13.	Explain the isNaN() method
14.	Explain the alert() function
15.	Explain the prompt() function
16.	Write a note on Assignment operator.
17.	Write a note on IF statement.

UNIT-II: LONG QUESTIONS

1.	Write a note on Operators.
2.	Write a note on Variables.
3.	Explain the basic data types.
4.	Explain Comparison operators
5.	Explain about Arithmetic operators
6.	Explain Logical operator.
7.	Write a note on Conditional Constructs. (if, if-else, else-if, switch)
8.	Write a note on Built-in functions.
9.	Write a note on Types of Dialog Boxes.
10.	Write a note on SWITCH statement
11.	Write a note on FOR looping statement

UNIT-III: MULTIPLE CHOICE QUESTIONS

1.	In JavaScript created objects using _______ operator (keyword.	
	a. new	**b.** Arithmetic
	c. logical	**d.** Assignment
	ANSWER: A	
2.	The ______ property returns the number of characters in a string.	
	a. charAt	b. length
	c. indexOf	d. substr
	ANSWER: B	
3.	The charAt() method returns the _____ at the specified index position in a string.	
	a. Unicode(ASCII)	b. character position
	c. character	d. none of these
	ANSWER: C	
4.	The indexOf() method returns the ______ of the specified value or string.	
	a. Unicode(ASCII)	b. character position
	c. character	d. none of these
	ANSWER: D	
5.	The ____method extracts (cut) the characters from a string and returns the new substring.	
	a. substr()	b. indexOf()
	c. charAt()	d. toUpperCase()
	ANSWER: A	
6.	The _______ method converts a string to Small case letters.	
	a. substr()	b. indexOf()
	c. charAt()	d. toLowerCase()
	ANSWER: D	
7.	The_______ method converts a string to Capital letters.	
	a. substr()	b. indexOf()
	c. charAt()	d. toUpperCase()
	ANSWER: D	
8.	The _____ object allows you to perform mathematical tasks.	
	a. String	b. Array
	c. Math	d. Date
	ANSWER: C	
9.	The_____ method returns the absolute value (positive value) of a number.	
	a. abs()	b. ceil()
	c. floor()	d. round()
	ANSWER: A	
10.	The _____ method rounds a number to the nearest larger integer, and returns the result.	
	a. abs()	b. ceil()
	c. floor()	d. round()
	ANSWER: B	
11.	The _____ method rounds a number to the nearest smaller integer, and returns the result.	
	a. abs()	b. ceil()
	c. floor()	d. round()
	ANSWER: C	
12.	The ______method returns the largest value from the list.	
	a. max()	b. abs()
	c. min()	d. large()

	ANSWER: A	
13.	The ____ method returns the smallest value from the list.	
	a. max()	b. abs()
	c. min()	d. large()
	ANSWER: C	
14.	The ______ method returns the rounded number to the nearest integer.	
	a. abs()	b. ceil()
	c. floor()	d. round()
	ANSWER: D	
15.	The _______ method returns the day of the month for the specified date, according to local time.	
	a. getDate()	b. getDay()
	c. getMonth()	d. getFullYear()
	ANSWER: A	
16.	The ______ method returns the day of the week for the specified date, according to local time.	
	a. getDate()	b. getDay()
	c. getMonth()	d. getFullYear()
	ANSWER: B	
17.	The ______ method returns the year for the specified date, according to local time.	
	a. getDate()	b. getDay()
	c. getMonth()	d. getFullYear()
	ANSWER: D	
18.	The ______ method returns the month for the specified date, according to local time.	
	a. getDate()	b. getDay()
	c. getMonth()	d. getFullYear()
	ANSWER: C	
19.	The _____ method returns the number of milliseconds since midnight of January 1, 1970 for the specified date.	
	a. getTime()	b. getHours()
	c. getMinutes()	d. getSeconds()
	ANSWER: A	
20.	The ____ method returns hours from the specified date and time, according to local time.	
	a. getTime()	b. getHours()
	c. getMinutes()	d. getSeconds()
	ANSWER: B	
21.	The ___ method returns minutes from the specified date and time, according to local time.	
	a. getTime()	b. getHours()
	c. getMinutes()	d. getSeconds()
	ANSWER: C	
22.	The ___ method returns seconds from the specified date and time, according to local time.	
	a. getTime()	b. getHours()
	c. getMinutes()	d. getSeconds()
	ANSWER: D	

UNIT-III: SHORT QUESTIONS

1.	Explain how to declare an Array with syntax and example.
2.	Explain the LENGTH property of String object with its syntax and example.

3.	Explain the CHARAT() method with its syntax and example.
4.	Explain the INDEXOF() method with its syntax and example.
5.	Explain the SUBSTR() method with its syntax and example.
6.	Explain the TOUPPERCASE() method with its syntax and example.
7.	Explain the TOLOWERSCASE()method with its syntax and example.
8.	Explain the PI property of Math object with its syntax and example.
9.	Explain the ABS() method of Math object with its syntax and example.
10.	Explain the CEIL() method of Math object with its syntax and example.
11.	Explain the FLOOR() method of Math object with its syntax and example.
12.	Explain the ROUND() method of Math object with its syntax and example.
13.	Explain the MAX() method of Math object with its example.
14.	Explain the MIN() method of Math object with its example.
15.	Explain the GETDATE() method of Date object with its example.
16.	Explain the GETDAY() method of Date object with its example.
17.	Explain the GETFULLYEAR() method of Date object with its example.
18.	Explain the GETMONTH() method of Date object with its example.
19.	Explain the GETTIME() method of Date object with its example.
20.	Explain the GETHOURS() method of Date object with its example.
21.	Explain the GETMINUTES() method of Date object with its example.
22.	Explain the GETSECONDS() method of Date object with its example.
23.	Explain the SETFULLYEAR() method of Date object with its example.
24.	Explain the SETDATE() method of Date object with its example.
25.	Explain the SETMONTH() method of Date object with its example.
26.	Explain the SETTIME() method of Date object with its example.
27.	Explain the SETMINUTES() method of Date object with its example.
28.	Explain the SETHOURS() method of Date object with its example.
29.	Explain the SETSECONDS() method of Date object with its syntax and example.
30.	Write a note on Dense Arrays.

UNIT-III: LONG QUESTIONS

1.	Explain how can you Creating, assigning, initializing Arrays in JavaScript with its syntax and example.
2.	Explain how can you Creating (declaring), parameter passing, Function call, Return values in user define functions in JavaScript.
3.	Explain any 5 methods of String object with its syntax and example.
4.	Explain any 5 methods of Math object with its syntax and example.
5.	Explain Get methods in Date object with its syntax and example.
6.	Explain Set methods in Date object with its syntax and example.
7.	Write a note on User-defined functions, giving all its terminologies.
8.	Write a note on User-defined functions. Explain the concept of ([No] Parameter passing and [No] return value) with an example.

UNIT-IV: MULTIPLE CHOICE QUESTIONS

1.	The topmost object in the DOM is __________	
	a. History	**b.** Navigator
	c. Document	**d.** Window
	ANSWER: D	
2.	The collection of objects in the browser for use in JavaScript is called the______.	
	a. DOM	b. Document
	c. Navigator	d. Window
	ANSWER: A	
3.	_____ property is used to change default message in the window status bar.	
	a. length	b. src
	c. href	d. defaultStatus
	ANSWER: D	
4.	______ object contains the history of pages visited by the user.	
	a. History	b. Navigator
	c. Document	d. Window
	ANSWER: A	
5.	_____ method specifies how far forward or backward in the history stack you want go.	
	a. back()	b. go ()
	c. forward()	d. None of these
	ANSWER: B	
6.	______ method displays the previous page from the history stack the user had visited in the browser.	
	a. back()	b. replace()
	c. forward()	d. None of these
	ANSWER: A	
7.	______ method displays the next page from the history stack the user had visited in the browser.	
	a. back()	b. replace()
	c. forward()	d. None of these
	ANSWER: C	
8.	_____ object contains information about the URL, Server hosting, Port number of the server and Protocol used in current page.	
	a. Location	b. History
	c. Window	d. Document
	ANSWER: A	
9.	______ removes the current page entry and adds a new page in the history stack.	
	a. back()	b. replace()
	c. forward()	d. href
	ANSWER: B	
10.	______ object contains information about the display capability like height, width, color depth, screen sizes (pixel) etc of client machine.	
	a. History	b. Navigator
	c. Screen	d. Window
	ANSWER: C	
11.	_____ Property is used to change the background color of the web page after it is loaded.	
	a. Bgcolor	b. BgColor
	c. bgcolor	d. bgColor

	ANSWER: D	
12.	______ property returns the number of images present in the images collection.	
	a. src	b. length
	c. href	d. image
	ANSWER: B	
13.	______ property returns the string value entered in the textbox by the user.	
	a. readonly	b. length
	c. src	d. value
	ANSWER: D	
14.	_____ property returns true or false depending on whether a radio/checkbox is checked or not.	
	a. value	b. selected
	c. selectIndex	d. checked
	ANSWER: D	
15.	________ event occurs when the user clicks on an element.	
	a. onclick	b. onblur
	c. onfocus	d. onload
	ANSWER: A	
16.	______ event occurs when an object loses focus	
	a. onclick	b. onblur
	c. onfocus	d. onload
	ANSWER: B	
17.	_______ event occurs when an element gets focus	
	a. onclick	b. onblur
	c. onfocus	d. onload
	ANSWER: C	
18.	______ event occurs when a user presses a mouse button over an element.	
	a. onmousedown	b. onmouseup
	c. onmouseout	d. onmouseover
	ANSWER: A	
19.	_____ event occurs when a user releases a mouse button over an element.	
	a. onmousedown	b. onmouseup
	c. onmouseout	d. onmouseover
	ANSWER: B	
20.	_______ event occurs when a user moves the mouse pointer out of an element.	
	a. onmousedown	b. onmouseup
	c. onmouseout	d. onmouseover
	ANSWER: C	
21.	_____ event occurs when a user moves mouse over an element.	
	a. onmousedown	b. onmouseup
	c. onmouseout	d. onmouseover
	ANSWER: D	
22.	_____ event occurs before the browser closes the document.	
	a. onunload	b. onblur
	c. onfocus	d. onload
	ANSWER: A	

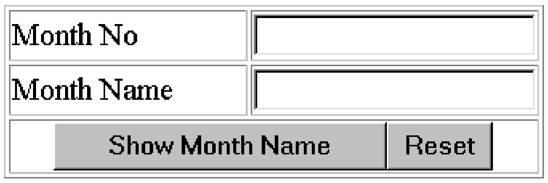
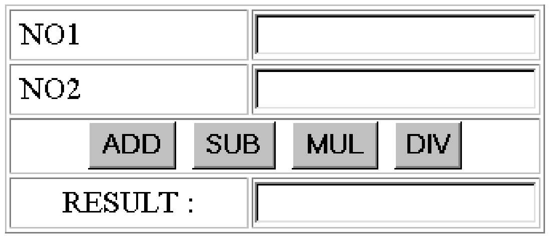
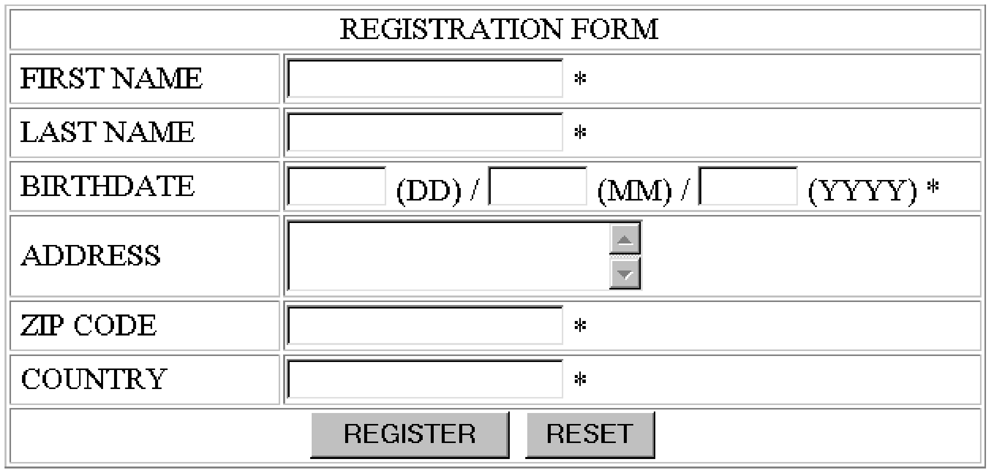
UNIT-IV: SHORT QUESTIONS

1.	Explain the DOM in short.
2.	Explain the **defaultStatus** property of Window object in short.
3.	Explain the **back** and **forward** methods of History object in short
4.	Explain the **href** property of Location object in short.
5.	Explain the **go** and **replace** methods of History object in short.
6.	Explain in Screen object in short.
7.	Explain the write() method of Document object with an example.
8.	Explain the **bgColor** property of Document object in short.
9.	Explain the **images** collection in short.
10.	Explain the **links** collection in short.
11.	Explain the **forms** collection in short.
12.	Explain the **all** collection and **getElementById** in short.
13.	What is event handler? List out how many types of event handler?
14.	Explain “onclick” event with an example.
15.	Explain “onblur” event with an example.
16.	Explain “onfocus” event with an example.
17.	Explain “onload” event with an example.
18.	Explain “onmousedown” event with an example.
19.	Explain “onmouseup” event with an example.
20.	Explain “onmouseout” event with an example.
21.	Explain “onmouseover” event with an example.
22.	Explain “onreset” event with an example.
23.	Explain “onsubmit” event with an example.
24.	Explain “onunload” event with an example.

UNIT-IV: LONG QUESTIONS

1	Draw and explain the structure of DOM Hierarchy in detail [marks:10]
2	Explain Window object in detail.
3	Explain Document Object in detail. [marks:10]
4	Explain History Object in detail.
5	Explain Location Object in detail.
6	Draw and explain the FORM DOM HIERARCHY in detail [marks:10]
7	What is Event Handling? Explain any 5(five) with an example.
8	Explain the TEXT control with its property in short.
9	Explain the RADIO BUTTON control with its property in short.
10	Explain the CHECKBOX control with its property in short.
11	Explain the DROPDOWNLIST control with its property in short.
12	Explain the BUTTON control with its property in short.

	CSS PROGRAMS
1	**Make mystyle.css file having following type of specification and and link this file in html file** ***[HTML TAGS*** ***H1 TO H6, HR, <P>, <B>, <I>, <U>, <FONT>, <TABLE>, <TR>, <TD>, <FORM>, <BODY>, <OL>, <UL>, <LI>, <IMG>,*** ***<INPUT TYPE=TEXTBOX/RADIO/CHECK/DROPDOWN/BUTTON>]*** **Specification of CSS File** 1. Apply different **font attributes** to different HTML tags using CSS. 2. Apply different **color and background attributes** to different HTML tags. 3. Apply different **font, color, background, text, border and margin attributes**. 4. Apply above all attributes and **List-attributes**. 5. Apply **class** concept in the style sheet and use different classes with the HTML tags. 6. Apply **<SPAN>** tag to display a particular word or a set of words in specific style. 7. Use **<DIV>** tag to display different divisions at different locations within web-page using position, left, top and width attributes. Use **<LAYER>** tag. *(Create two layers One with Red and another with Blue background and display next to each other horizontally and vertically both)*
2	**Make mystyle.css file having following type of specification and and link this file in html file** 1. Body : background ivory 2. Paragraph : 12point, arial, brown font 3. Heading : 14-24point, arial, red font 4. Horizontal Rule : color=green / background image **5.** Table : backgroundcolor=pink; border=2; text-align=right; color=brown;

	JAVASCRIPT PROGRAMS
3	Display an alert box giving message “Onclick event on button” using ONCLICK button click event.
4	Create “CLICK HERE TO ENTER NAME” button, onclick event accept user name and print it with “Hello! How are you nnnn”. Where nnnn is user entered name.
5	Display a alert box giving message “Enter your name” using onload event. Find the length of the inputted string and display the message “Length of your name is nn” where nn is the length of the string.
6	Write an html page to find the month name from the inputted month number. 
7	Write an html page to print result according to selected choice (ADD,SUB,MUL,DIV, MOD) 
8	Write an html page for registration form with all the data validation.  REGISTRATION FORM FIRST NAME * LAST NAME * BIRTHDATE (DD) / (MM) / (YYYY) * ADDRESS ZIP CODE * COUNTRY * REGISTER RESET Validations: 1. First Name must begin with Capital alphabet 2. Last Name must be having 4 character minimum witdth 3. DD must between 1-31 4. MM must be a valid no within 1-12

	5. YYYY must be four digited 6. Zip code must be number 7. Country Either INDIA, USA, AUSTRALIA, UK
9	Create the following Form and perform the following validations and functions User Name : [ ] Password : [ ] **Login** **Validations:** 1. Both fields are compulsory & it should not contain blank space. 2. User name must contain only characters. 3. Password begins with an alphabet only. 4. Length of password is minimum 8 characters and maximum 16 characters.
10	**CALCULATION BASED ON TEXTBOX ENTRY** 10.Calculate Total marks, percentage and class of the student. Directly enter your name in the textbox. **NAME** : [ ] ORACLE : [ ] C++ : [ ] E-COMMERCE : [ ] SAD : [ ] STATISTIC : [ ] SE : [ ] TOTAL : [ ] PERCENTAGE : [ ] CLASS : [ ] **Note:** **35 – 50% Pass Class, 51 – 75 Second Class, >75 First Class**

11	**<u>User login form</u>** Registration No: ☐ Name : ☐ Gender: ◯ Male ◯ Female English Ability: ☐ Read ☐ Write ☐ Speak CLICK SUBMIT RESET ☐ To create the following form. When press the 'click' button in the last text box, display the selected field like "Hello Sir/Madam _____, your registration no ____ has been selected due to your English ability ____".
12	**Write a JavaScript program to read a string from user and perform the following** 1. *Find whether the first character is a vowel or not* 2. *Check whether the last character is same as the first character or not* 3. *Check whether it contains a space any where* 4. *Replace all the "a" with "e"* 5. *Extract the last three characters and check if it is "com" or not.* **Note: Display all these in individual message boxes**
13	**Write a JavaScript program to read a number from user and perform the following, if entered value is Number then only perform the following other wise generate error message** 1. *Print the positive equivalent of the number* 2. *Check if the no is Integer or Not* 3. *Find the Square root of the number (If the no is +ve)* 4. *Find the Cube of the number* **Note: Display all these in individual message boxes**
14	**Write a JavaScript program to read a date from user and perform the following, if entered value is date then perform the following** 1. *Display the day no from the date* 2. *Display the year from the date* 3. *Display the month from the date* 4. *Change the date to next day* 5. *Change the year to next year* 6. *change the month to next month* **Note: Display all these in individual message boxes**

15	**Write a JavaScript program to read a time from user and perform the following, if entered value is time then perform the following** *1. Display the hour from the time* *2. Display the minute from the time* *3. Display the second from the time* *4. Change the hour to next hour* *5. Change the minute to next minute* *6. change the second to next second* **Note: Display all these in individual message boxes**

Note: getDay() is not to be covered in the theory

SC

[39/A-8]

SEAT No.________

No. of Pages: 02

SARDAR PATEL UNIVERSITY
F. Y. BCA (Semester – II) (CBCS) EXAMINATION
DATE: 01/04/2019, Monday TIME: 10:00a.m to 1:00p.m

US02CBCA04: WEB APPLICATION DEVELOPMENT [NC]

Total Marks: 70

Note: **1.** All the questions are compulsory. **2.** Figures to the right indicate marks.
3. Start a new question from a new page.

Q.1 Choose the correct answer: **[10]**

1. You need a _________ to execute a server-side script.
   a. Browser **b.** Client **c.** Server **d.** Web Server
2. The main use of Client-side scripts is _________
   a. Data input in form **b.** Data validation
   c. Display messages **d.** Generate HTML pages
3. To have an inline style sheet in HTML file, _________ must be used
   a. <script> **b.** <style> **c.** <link> **d.** style attribute
4. _________ is used to give bold effect to text
   a. font-family **b.** font-size **c.** font-style **d.** font-weight
5. A layered effect is only possible if _________ property is used
   a. z-index **b.** visibility **c.** position **d.** <div>
6. How many operands are required for operator ++?
   a. One **b.** Two **c.** Three **d.** Four
7. Which operator is used to concatenate two strings?
   a. + **b.** ++ **c.** += **d.** None of these
8. Which one of the following is not Comparison operator?
   a. == **b.** != **c.** < **d.** +=
9. ________ statement allows the loop to exit prematurely.
   a. break **b.** continue **c.** switch **d.** } (closing bracket)
10. To access each item in the dropdown list, _________ is used
   a. text **b.** value **c.** options[] **d.** selectedIndex

Q.2 Attempt (any Ten) of the following: **[20]**

1. What is Scripting?
2. List two languages used in Client-side scripting and Server-side scripting.
3. List 5 uses of DHTML
4. Explain the text-align CSS property taking an example.
5. Explain the background-repeat CSS property taking an example.
6. Explain the font-size CSS property taking an example.
7. Explain immediate-if.
8. Write a note on **new** operator.
9. Explain toUpperCase( ) and toLowerCase( ) methods.
10. Draw the DOM hierarchy.
11. Explain the all collection in short.
12. What is an event? List different events available.

(P.T.O)

Q.3

a. Write a short note on Components of DHTML. **[05]**

b. What are Client-side scripts? Explain the advantages of Client-side scripting. **[05]**

OR

Q.3

a. List and explain the uses of DHTML. **[05]**

b. What are Server-side scripts? Explain the advantages of Server-side scripting. **[05]**

Q.4

a. Write a note on FONT properties in CSS with example. **[10]**

OR

Q.4

a. Write a note on TEXT properties in CSS with example. **[10]**

Q.5

a. Explain the Increment & Decrement operators with examples. **[05]**

b. Write a note on any 2 Date Object methods with syntax and example for each. **[05]**

OR

Q.5

a. What are Variables? Give the rules and syntax of declaring variables. **[05]**

b. Write a note on any 2 Math object Get methods with syntax and example for each. **[05]**

Q.6

a. Explain History object in detail. **[05]**

b. Write a note on working with RADIO control in Javascript **[05]**

OR

Q.6

a. Write a note on working with CHECKBOX control in Javascript **[05]**

b. Explain Location object in detail. **[05]**

—X—

********* All The Best *********

NO.OF.PAGES: 02

SARDAR PATEL EDUCATION CAMPUS-II

SARDAR PATEL COLLEGE OF ADMINISTRATION & MANAGEMENT

F.Y.BCA (Semester-II) Prelim Exam-2019

DATE:07/03/2019, THURSDAY **TIME:**10:30 TO 1:30 P.M

US02CBCA23: WEB APPLICATION DEVELOPMENT – II

Q-1 MULTIPLE CHOICE QUESTIONS (10 MARKS)

1 _________ Scripts are executed by the Browser.

A. Client-side B. Server-side
C. CSS D. HTML

2 The main use of Client-side scripts is _________

A. Data input in form B. Data validation
C. Display messages D. Generate HTML pages

3 The code written in _________ is visible to the visitor of the site.

A. CSS B. server-side scripting language
C. client-side scripting language D. ASP

4 Which keyword is used to declare a variable as an integer?

A. int B. num
C. var D. None of these

5 Which of the following is a valid variable name?

A. With B. My%variable
C. 99variable D. Mystr

6 NaN means _________

A. Not allowed Number B. Not a Number
C. Non allowed Number D. None of these

7 The length property returns the number of characters in a string (T/F)________________.

8 The ____ method returns the smallest value from the list.

A. max() B. abs()
C. min() D. large()

9 ______ object contains the history of pages visited by the user.

A. History B. Navigator
C. Document D. Window

10 Onclick event occurs when the user clicks on an element (T/F) ________________.

P.T.O

Q-2 GIVES THE ANSWER FOR FOLLOWING QUESTIONS (ANY 10). 20 MARKS

1 What is Scripting?
2 What is Client-side Scripting?
3 Explain the difference between Client-side and Server-side scripting.
4 Write down the rules for defining variable
5 Explain about Prompt dialog box.
6 Explain the parseInt() method
7 Explain how to declare an Array with syntax and example.
8 Explain the LENGTH property of String object with its syntax and example.
9 Explain the TOUPPERCASE() method with its syntax and example
10 Explain the DOM in short.
11 Explain “onclick” event with an example.
12 Explain the go and replace methods of History object in short.

Q-3 **A)** Write a note on Scripting. List the type of scripting and explain any one in details. 10

OR

Q-3 **A)** Discuss advantages of JavaScript in brief. 5
B) Write a note on Applications of JavaScript 5

Q-4 **A)** Write a note on Conditional statements. Explain any one in details. 5
B) Write a note on Variables. 5

OR

Q-4 **A)** Write a note on Operator. List all types of operators and explain any two in details. 10

Q-5 **A)** Explain how can you Creating, assigning, initializing Arrays in JavaScript with its syntax and example 10

OR

Q-5 **A)** Explain any 5 methods of String object with its syntax and example. 5
B) Explain any 5 methods of Math object with its syntax and example. 5

Q-6 **A)** What is Event Handling? Explain any five with an example. 5
B) Explain the TEXT control with its property in short. 5

OR

Q-6 **A)** Draw and explain the structure of DOM Hierarchy in detail 10

SARDAR PATEL EDUCATION CAMPUS-II

SARDAR PATEL COLLEGE OF ADMINISTRATION & MANAGEMENT

UNIT TEST-2019

SUBJECT : US02CBCA23: WEB APPLICATION DEVELOPMENT – II

CLASS: BCA(2ND SEM) **DATE:** 18 /01/2019

TIME: 10:30 A.M TO 01:30 P.M **TOTAL MARKS:** 70 MARKS

Q-1 MULTIPLE CHOICE QUESTIONS (10 MARKS)

1 The main use of Client-side scripts is _________

A. Data input in form **B.** Data input in form

C. Display messages **D.** Display messages

2 The code which is used to connect to the database must be _________

A. written in server-side scripts **B.** written in server-side scripts

C. written in CSS **D.** written in CSS

3 The code written in _________ is visible to the visitor of the site.

A. CSS **B.** CSS

C. client-side scripting language **D.** client-side scripting language

4 The scripts to follow based on user action or some condition are written in _________

A. CSS **B.** CSS

C. PHP **D.** PHP

5 Where can you write the JavaScript within your HTML code?

A. In <HEAD> only **B.** In <HEAD> only

C. In <HEAD> and <BODY> both **D.** In <HEAD> and <BODY> both

6 Which of the following is an invalid data type?

A. Number **B.** Number

C. String **D.** String

7 NaN means Not a Number(T/F)________________

8 Which one of the following is not Comparison operator?

A. == **B.** ==

C. < **D.** <

9 _________ returns a true or false depending on the user communication.

A. alert() **B.** alert()

C. confirm() **D.** confirm()

10 You need a web server to execute a server-side script.(T/F)_______________.

Q-2 GIVES THE ANSWER FOR FOLLOWING QUESTIONS (ANY 10) 20 MARKS

1 What is Scripting?
2 What is Client-side Scripting?
3 Differentiate between Server-side and Client-side scripting
4 List two languages used in Client-side scripting and Server-side scripting
5 Write down the rules for defining variable.
6 List out all operator.
7 Write a note on JavaScript.
8 Explain about Arithmetic operators
9 Explain Logical operator.
10 List the advantages of Client-side scripting
11 Explain scope of variable.
12 Explain about Prompt dialog box. (with example)

Q-3 GIVES THE ANSWER FOR FOLLOWING QUESTIONS 40 MARKS

1. Write a note on Client-side scripting stating the advantages and disadvantages of it 8

OR

Write a note on Server-side scripting stating the advantages and disadvantages of it

2. What is JavaScript? Discuss advantages and disadvantages of JavaScript in brief. 10

3. Write a note on users and features of JavaScript 7

OR

How to use JavaScript on webpage. Explain with syntax and example.

4. What is data type? list and explain all type of data type with example. 7

OR

What is variable? How to Declaration and initialization of variable.

5. List the entire operator and explain any two operators with example. 8

OR

Explain following.

1. if statement
2. if...else statement
3. if...else if... statement.