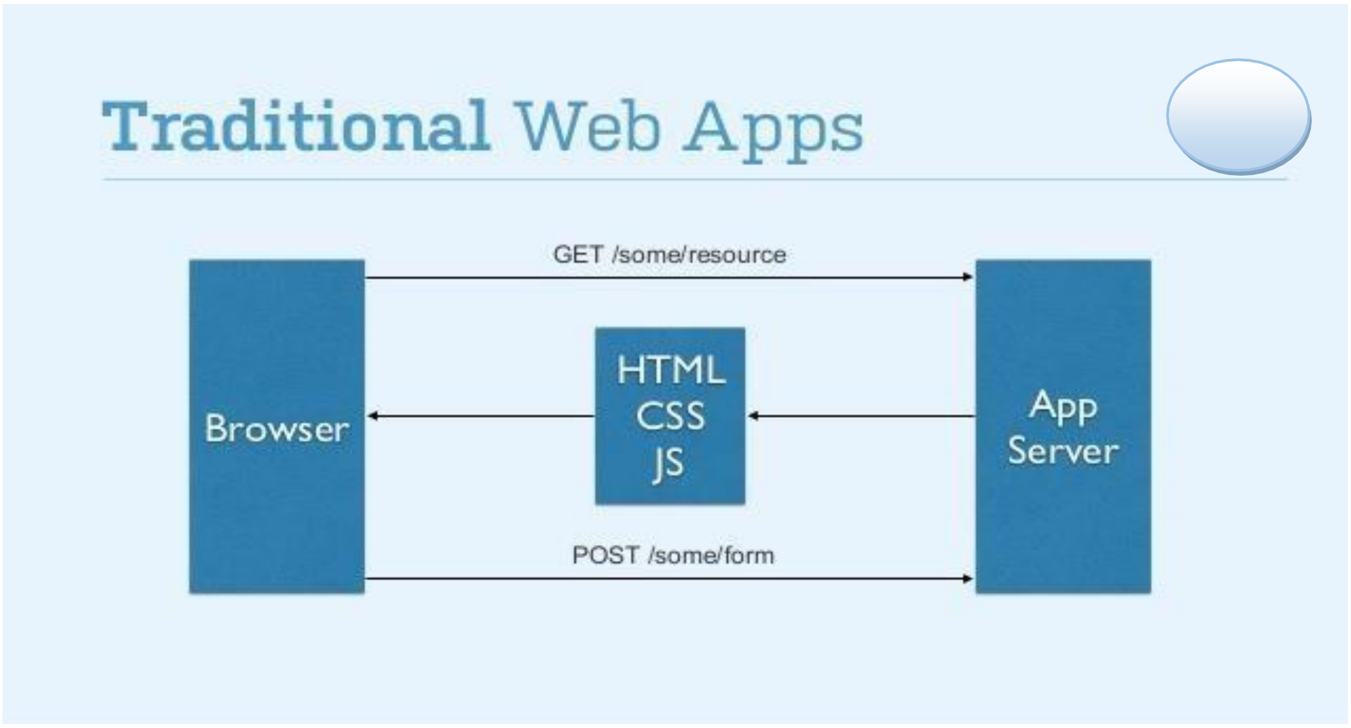
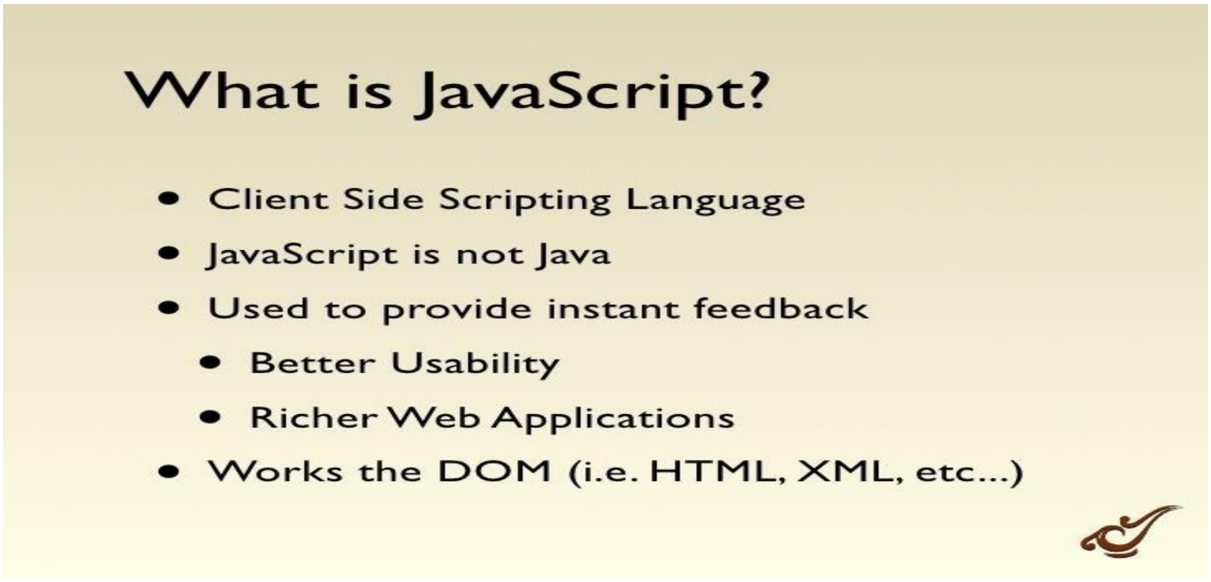
UNIT-4

JAVASCRIPT

- INTRODUCTION TO JAVASCRIPT
- VARIABLES
- OPERATORS
- CONDITIONAL STATEMENTS
- LOOPING STATEMENTS
- BREAK AND CONTINUE STATEMENTS
- DIALOG BOXES
- ARRAYS
- USER DEFINED FUNCTIONS(UDF)
- BUILT IN FUNCTIONS
- EVENTS
- DOM OBJECT
- HISTORY OBJECT
- FORM AND EMAIL VALIDATION

➢ WRITE A NOTE ON JAVASCRIPT. (3 OR 5 MARKS)





What is JavaScript?

- Client Side Scripting Language
- JavaScript is not Java
- Used to provide instant feedback
  - Better Usability
  - Richer Web Applications
- Works the DOM (i.e. HTML, XML, etc...)

- ➢ JavaScript is the most popular scripting language on the internet, and works in all major

  browsers, such as Internet Explorer, Firefox, Chrome, Opera, and Safari.
- ➢ JavaScript is a language that is used to make web pages

interactive.

- It runs on your user's computer and so does not require constant downloads from yourweb site.
- JavaScript was designed to add interactivity to HTML pages.
- JavaScript is a scripting language.
- A scripting language is a lightweight programming language.
- JavaScript is usually embedded directly into HTML pages.
- Everyone can use JavaScript without purchasing a license.
- Using JavaScript user can add events on your website.
- Using JavaScript user can check the data before it is submitted to the server.

How to write JavaScript code into html page?

- The HTML **<script>** tag is used to insert a JavaScript into an HTML page.
- Inside <script> tag t**ype** attribute is used to specify the scripting language.
- JavaScript can write into two sections either between <HEAD> tag or between <BODY>tag.
- If user writes code in <body> section then JavaScript code will be executed immediatelyafter the page has been load.
- If user writes code in <head> section then JavaScript code will be executed depends onuser's requirement.

Syntax:

```
<HTML>
        <BODY>
        <SCRIPT TYPE = "TEXT/JAVASCRIPT">
. . . .
Between these

two tags All
```

JavaScript code will

Be Written

```
        </SCRIPT>
        </BODY>
</HTML>
```

You can also write like **<SCRIPT LANGUAGE = “JAVASCRIPT”>.**

The **document.write()** command is used to display the messages on the screen.

For Example:

This code will display hello word on browser.

```
<Html>
        <Body>
                <script language = "JavaScript">
                        document.write('hello
                        word');
                </script>
        </body>
</html>
```

➢ EXPLAIN JAVASCRIPT VARIABLES (2 OR 3 MARKS)

- ➢ Variable is container for storing the information.
- ➢ Variable names are case sensitive. (x and X are two different variable).
- ➢ Variable names must be started with letter or underscore.
- ➢ To declare variable in JavaScript **VAR** statement is used.
- ➢ Syntax: VAR <variable name>

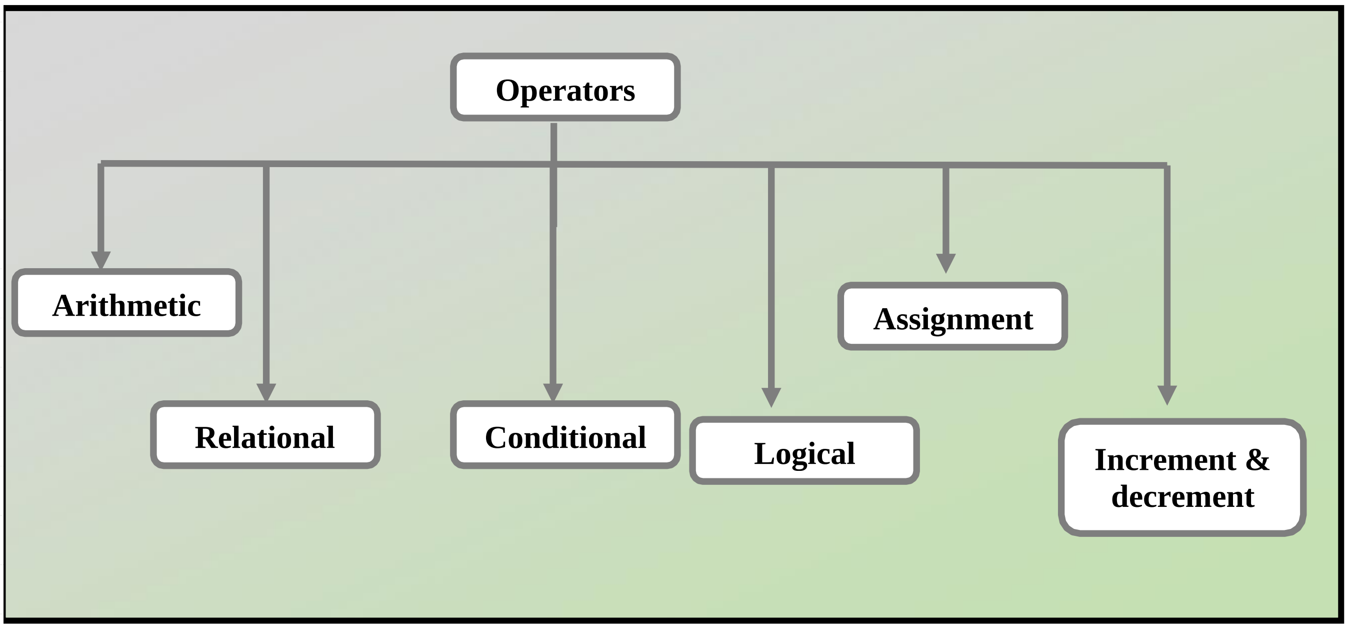
NOTE: it is not compulsory to declare variable in JavaScript before its use.

- If you assign values to variables that have not yet been declared, the variables will automatically be declared.

```
<Html>
        <Body>
                <script language = "JavaScript">
                        var x = 10; <-------variable x is declared.
                        var y = 20; <-------variable y is declared.
                        z = x+y;    < ------- variable z is not declared.
                        document.write('addition is' + z);
                </script>
        </body>
</html> (semicolon in JavaScript is optional)
```

- **EXPLAIN JAVASCRIPT OPERATOR (3 OR 5 MARKS)**

Operator means to "Operate something".
Operator can have different Operand or Values



ARITHMATIC OPERATOR:

- Arithmetic operators are used to perform arithmetic between variables and/orvalues.
- Given that **y=5**, the table below explains the arithmetic operators:

Operator	Description	Example	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3
*	Multiplication	x=y*2	x=10
/	Division	x=y/2	x=2.5
%	Modulus(division remainder)	x=y%2	x=1
++	Increment	x=++y	x=6
--	Decrement	x=--y	x=4

ASSIGNMENT OPERATOR:

- Assignment operators are used to assign values to JavaScript variables.
- Given that **x=10** and **y=5**, the table below explains the assignment operators:

Operato	Example	Same As	Result

r			
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
*=	x*=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

COMPARISION OPERATOR:

- Comparison operators are used in logical statements to determine equality or difference between variables or values.
- Given that **x=5**, the table below explains the comparison operators:

Operator	**Description**	**Example**
==	**is equal to**	**x==8 is false**
===	**is exactly equal to (valueand type)**	**x===5 is true x==="5" is false**
!=	**is not equal**	**x!=8 is true**
>	**is greater than**	**x>8 is false**
<	**is less than**	**x<8 is true**
>=	**is greater than or equal to**	**x>=8 is false**
<=	**is less than or equal to**	**x<=8 is true**

How Can it be Used

Comparison operators can be used in conditional statements to compare values andtake action depending on the result:

For Example:

```
If (age<18)
                    document.write("Too young");
```

LOGICAL OPERATOR:

- Logical operators are used to determine the logic between variables or values.
- Given that **x=6 and y=3**, the table below explains the

logical operators:

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

CONDITIONAL OPERATORS:

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

Syntax:

<Variable name> = <condition>? value1: value2

Example:

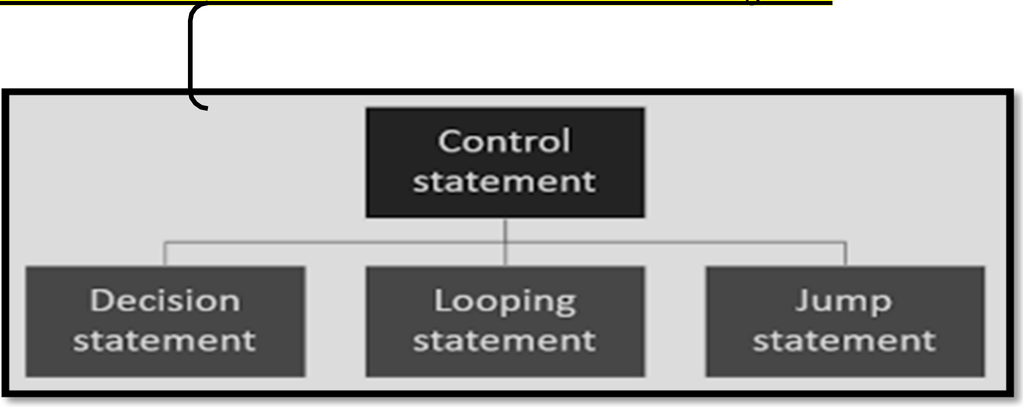
x = (y == 5)? “x is true”:”x is false”

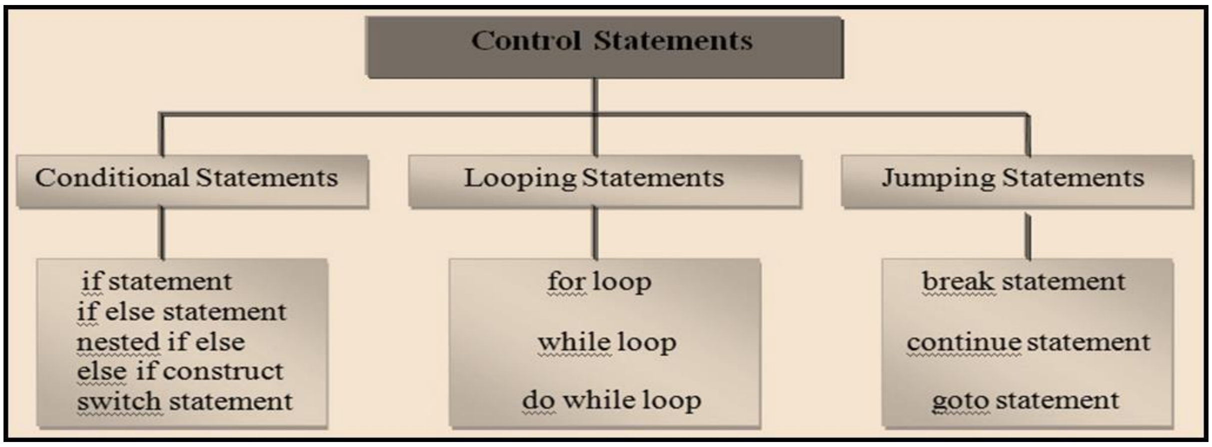
Here if value of y is 5 then value of x = “x is true”

If value of y is not 5 then value of = “x is false”

Q.4. EXPLAIN CONDITIONAL STATEMENTS (3 OR 5 MARKS)

Control Structure :- “It used to control Flow of the Program”





- Conditional statements in JavaScript are used to perform different actions based ondifferent conditions.
- In JavaScript we have the following conditional statements:

  - ✦ **if statement** - use this statement if you want to execute some code only ifa specified condition is true
  - ✦ **if...else statement** - use this statement if you want to execute some codeif the condition is true and another code if the condition is false
  - ✦ **if...else ifelse statement** - use this statement if you want to select one any blocks of code to be executed
  - ✦ **switch statement** - use this statement if you want to select one of manyblocks of code to be executed.

IF STATEMENT:

You should use if statement if you want to execute some code only if aspecified condition is True.

Syntax:

```
        If <condition>{
                JavaScript code executed if condition becomes true. }
```

Example:

```
<script language =
        "javascript">x = 5; y = 6;
        if (x<6){
                document.write(' x is greater'); }
</script>
```

Note that if is written in lowercase letters. Using uppercase letters (IF) will generate a *JavaScript* error!

IF...ELSE STATEMENT:

If you want to execute some code if a condition is true and another code ifthe condition is not true, use if else statement.

```
Syntax
        if (condition){
            Code to be executed if condition is true}
        else{
            Code to be executed if condition is not true }
```

```
Example:
<script language = "javascript">x =
5;
            y = 6;
            if (x > y) {
                document.write('x is greater');
            }else
            {document.write('y is greater');}</script>
```

IF…ELSE IF…ELSE STATEMENT:

It is used when user have two or more condition to check for execution of code.

```
Syntax:
        if <condition1>
        {
            Code if condition 1 is true;
        }else if <condition2>{
            Code if condition 2 is true;
        }else
        {
            Code if no condition becomes true;}
```

Example

```
<script language = "javascript">
          x=5;
          y=6;
          z=7;
          if(x>y && x>z)
          {
                    document.write('x is greater');
          }
          else if(y>x && y>z)
          {
                    document.write('y is greater');
          }
          else
          {
          document.write('z is greater');
          }
          </script>
```

➢ SWITCH STATEMENT:

- use the switch statement to select one of many blocks of code to be executed

```
Syntax
  :
      switch (n)
              case 1:
                      Execute code block
                      1
              case 2:
                      Execute code block
                      2
              default:
                      Code to be executed if n is
                      Different from case 1 and 2}
```

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

➢ EXPLAIN JAVASCRIPT BREAK & CONTINUE STATEMENTS (3 MARKS)

BREAK STATEMENTS

- ➢ The break statement "jumps out" of a loop.

- The continue statement "jumps over" one iteration in the loop.
- **The Break Statement**
- You have already seen the break statement used in an earlier chapter of thistutorial. It was used to "jump out" of a switch() statement.
- The break statement can also be used to jump out of a loop.

CONTINUE STATEMENT

- The **continue statement** breaks one iteration (in the loop), if a specifiedcondition occurs, and continues with the next iteration in the loop.
- This example skips the value of 3:

- **EXPLAIN DIALOG BOXES IN JAVASCRIPT (5 MARKS)(MIMP)**

  In JavaScript we can create three kinds of popup boxes: **Alert box, Confirm box, andPrompt box**.

  ✦ **Alert Box**

  - An alert box is often used if you want to make sure information comesthrough to the user.
  - When an alert box pops up, the user will have to click "OK" to proceed.

Syntax:
```
alert("sometext");
```

Example:
```
<html>
<body>
<script type="text/javascript">
alert("Hello! I am an alert box!");
</script>
</body>
</html>
```

CONFORM BOX

- A confirm box is often used if you want the user to verify or acceptsomething.
- When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
- If the user clicks "OK", the box returns true. If the user clicks "Cancel", thebox returns false.

Syntax:
```
confirm("sometext")
```

Example:

```
<script type="text/javascript">
        a = confirm('press ok
        button');if (a==true)
                alert('you press ok button');
        else
                alert('you press cancel button');
</script>
```

PROMPT BOX

- A prompt box is often used if you want the user to input a value before entering a page.
- When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
- If the user clicks "OK" the box returns the input value. If the user clicks "Cancel" the box returns null.

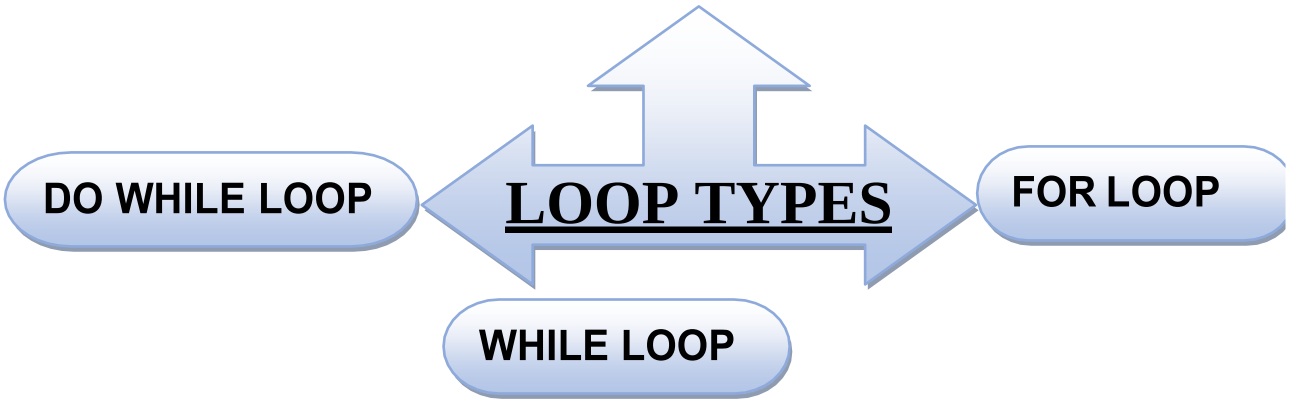
Syntax:

```
prompt (“sometext”, “default value”);
```

Example:

```
<script type="text/javascript">
        name=prompt("Please enter your name","tejas");
        document.write("Hello " + name + "! How are you today?");
</script>
```

- EXPLAIN LOOPING STRUCTURE IN JAVA SCRIPT (3 OR 5 MARKS)(IMP)



- Looping refers to the ability of a block of code to repeat itself.
- This repetition can be for a predefined number of times or it can go until certain conditions are met.
- For instance, a block of code needs to be executed till the value of a variable becomes 20 (Conditional Looping), or a block of code needs to be repeated 7 times.
- For this purpose, JavaScript offers 2 types of loop structures:
  1) For Loops - This loop iterate a specific number of times as specified.
  2) While Loops – This is Conditional Loops, which continue until a condition is met.

FOR LOOP:

Syntax

```
for (expression1; condition;
expression2)
{
            // Javascript commands…
}
```

Example:

```
    <script language =
            "javascript">for(n=10;
            n>=1; n--)
            {
                        document.write(n);
            }
  </script>
```

WHILE LOOP:

Where, the **condition** is a valid JavaScript expression that evaluates to a Boolean value.

- The JavaScript commands execute as long as the condition is true.

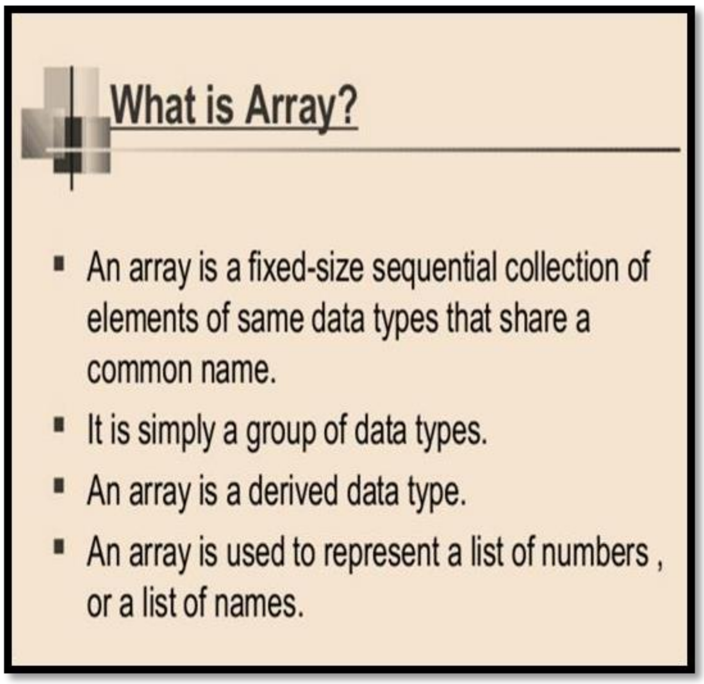
Syntax

```
while (condition)
{
            Javascript
            code…
}
```

Example

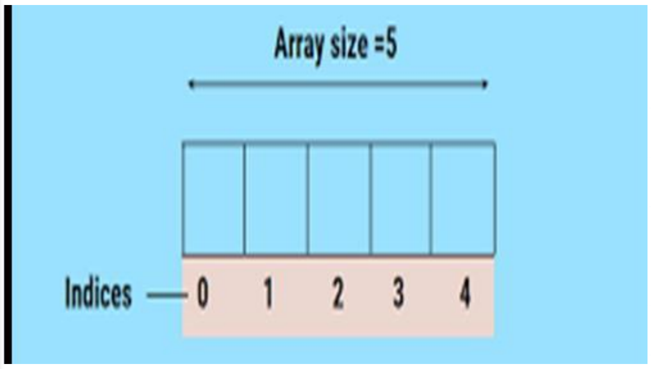
```
        <script language
        ="javascript">
              var n=1;
              while
              (n<=10)
              {
                   document.write(n);
                   n++;
              }
```

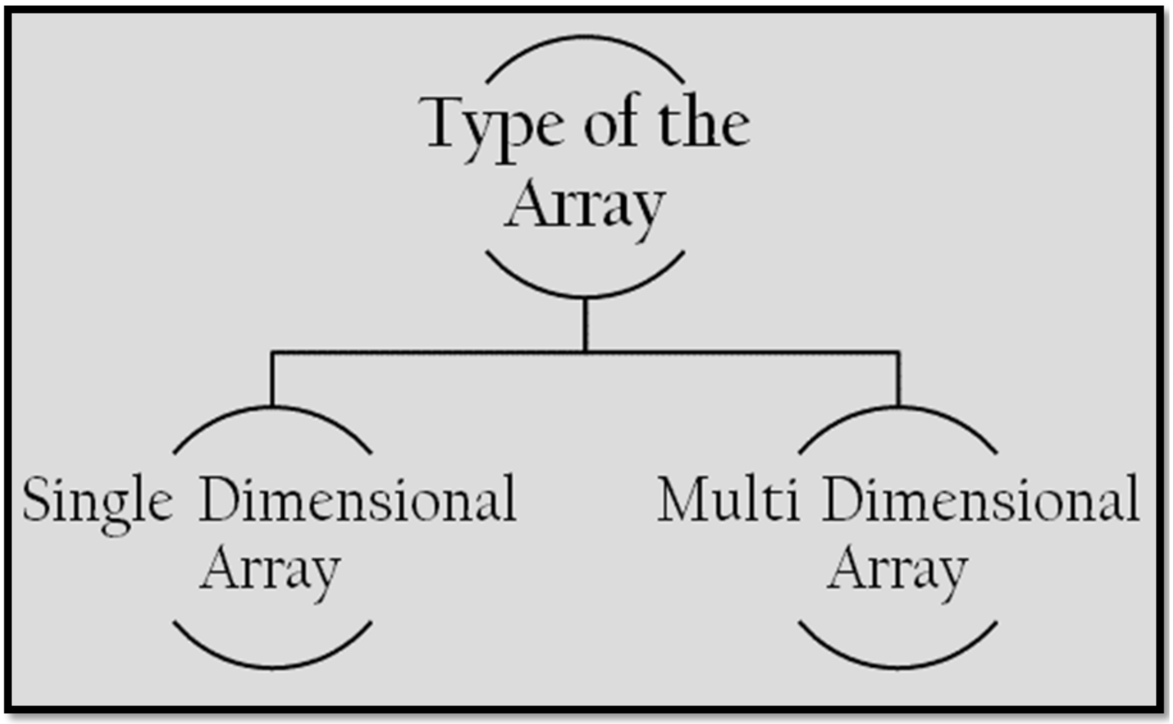
- **EXPLAIN ARRAY IN JAVA SCRIPT (3 MARKS)(IMP)**



What is Array?

- An array is a fixed-size sequential collection of elements of same data types that share a common name.
- It is simply a group of data types.
- An array is a derived data type.
- An array is used to represent a list of numbers , or a list of names.





- ➢ An array is a special variable, which can hold more than one value, at a time.
- ➢ If you have a list of items (a list of car names, for example), storing the cars in singlevariables could look like this:

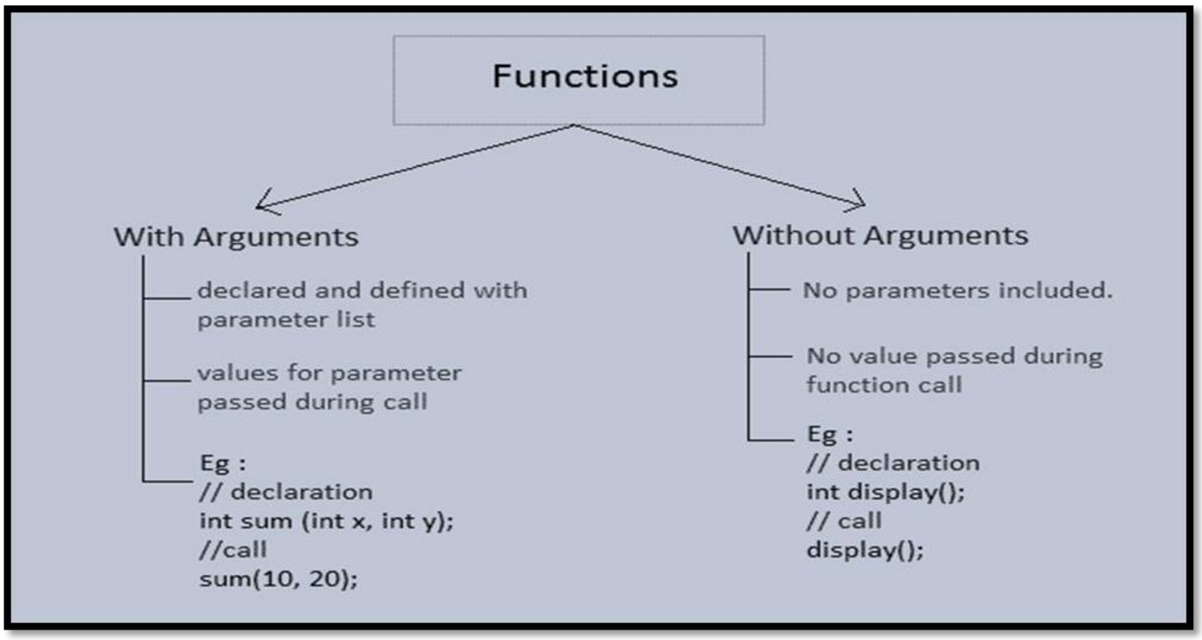
```
var
car1="Saab";
```

var
car2="Volvo";

var
car3="BMW";

- However, what if you want to loop through the cars and find a specific one? And what ifyou had not 3 cars, but 300?
- The best solution here is to use an array!
- An array can hold all your variable values under a single name. And you can access thevalues by referring to the array name.
- Each element in the array has its own ID so that it can be easily accessed.

- **WRITE A NOTE ON USER DEFINED FUNCTION(UDF) (3 MARKS OR 5 MARKS)**



- Functions offer the ability to group together JavaScript program code that performs aspecific task into a single unit that can be used repeatedly whenever required in a

JavaScript program.

- A user defined function first needs to be declared and coded.
- Once this is done the function can be invoked by calling it using the name given to the function when it was declared.
- Functions can accept information in the form of arguments and can return results.

Declaring Function:

- Functions are declared and created using the **function** keyword. A function contain the following:
    - A name for the function.
    - A list of parameters (arguments) that will accept values passed to the function when called.
    - A block of JavaScript code that defines what the function does

Syntax:

```
function <function name>(parameter1,parameter2,..)
{
        // JavaScript code…
}
```

- A <function name> is case sensitive, can include underscore (_), and has to start with a letter.
- The list of **parameters** passed to the function appears in parentheses and commas separate members of the list.

➢ BUILT IN FUNCTION IN JAVASCRIPT (2 , 3 OR 5 MARKS)

- **STRING FUNCTION:**

1) charAt()

Syntax: <string>.charAt(num);

Purpose:

- This function returns the character located at position passed in to the argument.
- This indexing is done from left to right starting with the 0 (zero) position.

.

Example:

```
<script language = "javascript">
        var x = "This is a test, small
        test.";y = x.charAt(5);
        document.write('character at 5th position is '+y);
</script>
```

Output: character at 5th position is i

2) concat()

Syntax: <string>.concat(string2);

Purpose:

- This function is used to join more than one string with string1.

```
Example:
        <script language =
                "javascript">var x =
                "hi";
                var y = "hw r
                u?"; var z =
                x.concat(y);
                document.write('final string is  '+z);
        </script>
Output: final string is hi hw r u?
```

3) **Replace:**

Purpose:

Syntax: <string>.replace(<string to replace> or [regular exp], <new string>);

o This method is used to replace the string from main string with new string

```
Example:
        <script language="JavaScript1.2">
                x = "This is main string to be
                replace"y =
                x.replace("string","text");
                document.write("new string is "+y);
        </script>
Output: new string is This is main text to be replace
```

4) **substr():**

Syntax: <string>.substr([num1], [num2]);

Purpose:

- This method returns the string from position specified in num1 with no of character specified with num2.
- If num1 is not specified then it starts to count character from 0.
- If num1 is negative then num2 is optional and it starts to count from last character of

string.

```
  Exampl:
<script language="JavaScript1.1">
  x = "This is a test";
                         y = x.substr(5,10);
         document.write('the substring from position 5 to 10 is:'+y);
             </script>
  Output: the substring from position 5 to 10 is: is a test
```

5) toUpperCase():

```
Syntax: <string>.toUpperCase();
```

Purpose:

This method returns all the characters of string in to uppercase.

```
Example:
<script language="JavaScript1.1">
x = "This is a test";
y = x.toUpperCase();document.write(y);
</script>
Output: THIS IS A
TEST
```

6) toLowerCase():

```
Syntax: <string>.toLowerCase();
```

Purpose:

This method returns the all characters of string in lower case.

```
Example:
   <script language="JavaScript1.1">
   x = "THIS IS A TEST";
           y = x.tolowerCase();
          document.write(y);
   </script>
```

➢ MATHS FUNCTION

1) abs()

Syntax: Math.abs(num)

Purpose:

- The abs() method is used to calculate the absolute value of num.

```
Example:
  <script
            language="JavaScript1.1"
            >x = Math.abs(-10);
            document.write(x);
 </script>
Output:
10
```

Note: for all math function user have to user Math object before all function.

2) Ceil():

Syntax: Math.ceil(num)

Purpose:

o The ceil() method calculates the smallest integer that is greaterthan or equal to the number passed in as a parameter

Example:
```
        <script
        language="JavaScript">
        x= Math.ceil(2.1);
        document.write(x);
</script>
Output:  3
```

3) floor():

Syntax: Math.floor(num)

Purpose:

- Get the largest integer number, which is equivalent to or less than the number passed as the parameter.

Example:
```
        <script
        language="JavaScript">
        x= Math.floor(2.9);
        document.write(x);
 </script>
  Output: 2
```

4) pow():

Syntax: Math.pow(num1, num2)

Purpose:

- The pow() method of the Math object is used to calculate anexponent power.

Example:
```
 <script
          language="JavaScript"
          >x= Math.pow(2, 3);
          document.write(x);
</script>
Output:        8
```

5) random():

Syntax: Math.random()

Purpose:

- The random() method of the Math object is used to obtain a random number between values 0 and 1.

Example:
```
<script
          language="JavaScript
          ">x= Math.random();
          document.write(x);
</script>
Output: 0.45678993
```

6) round():

Syntax: Math.round(num)

Purpose:

- The round() method rounds a number to its nearest integer value.
- If the fractional portion of the number is .5 or

greater, the result is rounded to the next highest integer.

- If the fractional portion of number is less than .5, the result is rounded to the next lowest integer.

```
Example:
        <script
                language="JavaScript
                ">x=
                Math.round(3.4);
                document.write(x);
         </scrip>
Output: 3
```

7) max():

```
Syntax: Math.max(num1, num2, num3…numn)
```

Purpose:

- The max() method of the Math object gets the given number of the two parameters passed to it.
- The larger value is returned as the result.

```
Example:
<script language="JavaScript">
        x= Math.max(3,4,1,2);
        document.write(x);
</script>
```

8) min():

```
Syntax: Math.min(num1, num2…numn)
```

Purpose:

- The min() method of the Math object gets the minimum number of the given parameters passed to it.

- The larger value is returned as the result.

Example:
```
<script language="JavaScript">
x= Math.min(3,4,1,2);
    document.write(x);
        </script>
```
Output: 1

DATE:

1) Date()

Syntax: <variable name> = new Date()

Purpose:

It is used to get the current system date and time.

Example:
```
<script language="JavaScript">
  x= new Date();
    document.write(x);
      </script>
```
Output: Tue Apr 12 2011 09:04:37 GMT+0530 (India Standard Time)

2) getDate():

Syntax: <date object>.getDate();

Purpose:

- The getDate() method returns the day of the month expressed as an integer from1 to 31.

- To access this function user must have to create date object.
- Date object is created as follows.
- x = new Date (); here we create date object named x.

Example:

```
 <script language="JavaScript">
    x= new Date();
  document.write(x.getDate());
</script>
Output:  30
```

3) getDay():

Syntax: <date object>.getDay();

Purpose:

- This method returns the current day of the week in integer form from 0 to6

Example:

```
<script language="JavaScript">x= new
           Date();
           document.write(x.getDay());
</script>
Output: 6
```

- Sunday is starting from 0 and onwards.

4) getMonth():

Syntax: <date object>.getMonth();

Purpose:

- The getMonth() method returns the month portion of the Date object expressed as an integer from 0

```
Example:
<script language="JavaScript">
x= new Date();
          document.write(x.getMonth());
</script>
Output: 3
```

(January) to 11 (December).

5) getYear():

```
Syntax: <date object>.getYear();
```

Purpose:

- The getYear() method returns the year portion of the Date object.

The year is represented as either a two-digit number or a four-digit number, depending on

```
Example:
<script language="JavaScript">
x= new Date();
          document.write(x.getYear());
</script>
 Output: 11
```

1) getFullYear():

```
Syntax: <date object>.getFullyear();
```

Purpose:

This method returns the year portion of the date with four digitnumber.

Example:

```
<script language="JavaScript">
        x= new Date();
        document.write(x.getFullYear());
</script>
Output: 20
11
```

2) getHours():

Syntax: <date object>.getHours();

Purpose:

o The getHours() method returns the hour portion of the date expressed asan integer from 0 (12:00 a.m. midnight) to 23 (11:00 p.m.).

Example:

```
<script language="JavaScript">
        x= new Date();
        document.write(x.getHours());
</script>
Output:
6
```

3) getMinutes():

Syntax: <date object>.getMinutes();

Purpose:

This method returns the current minute of the date expressed as an integer from 0 to 59.

```
Example:
<script language="JavaScript">
            x= new Date();
            document.write(x.getMinutes());
</script>
Output:
19
```

6) getSeconds():

Syntax: <date object>.getSeconds();

Purpose:

The getSeconds() method returns the seconds portion of the Date object expressed as an integer from 0 to 59.

```
Example:
<script language="JavaScript">
x= new Date();
            document.write(x.getSeconds());
</script>
Output: 29
```

7) getMiliSeconds():

Syntax: <date object>.getMilliSeconds();

Purpose:

- The getMilliseconds() method returns the millisecond portion of the date expressed as an integer from 0 to 999.

```
Example:
<script language="JavaScript">
        x= new Date();
        document.write(x.getMilliSeconds());
</script>
Output: 299
```

➢ ARRAY FUNCTION

1) join()

```
Syntax: <array object>.join(string);
```

Purpose:

- The join () method converts all the elements to strings and then concatenates all the strings into a longer string.
- If an argument is provided in the parameter list, it is used to separate the elements in the string returned by the method.

```
Example:<script language = "javascript">
 fruit = new Array("A","B","C");
aString = fruit.join("-");
        document.write("The fruit array contains: ",aString);
</script>
Output: The fruit array contains: A-B-C
```

2) reverse()

```
Syntax: <array object>.reverse();
```

- The reverse () method reverses the order of the elements in the array according to the array index numbers.

```
Example:
<script language = "javascript">
          x = new Array ("A","B","C");
          x.reverse();
          document.write("x[0]=",x[0],"<br>")
          ;
           document.write("x[1]=",x[1],"<br>");
           document.write("x[2]=",x[2],"<br>");
</scrip>
Output:
x [0]=C
x [1]=B
x [2]=A
```

3) pop():

Syntax: <array object>.pop();

Purpose:

- The pop () method deletes the last element of the array and sets the array's length property to one less than its current value.
- This last element is returned from the method.

```
Example:
<script language = "javascript">
          x = new
          Array("a","b","c");y =
          x.pop();
          document.write(y," was removed from the pile.");
</script>
Output: c was removed from the pile.
```

4) push()

Syntax: <array object>.push(arg1, arg2…argn);

Purpose:

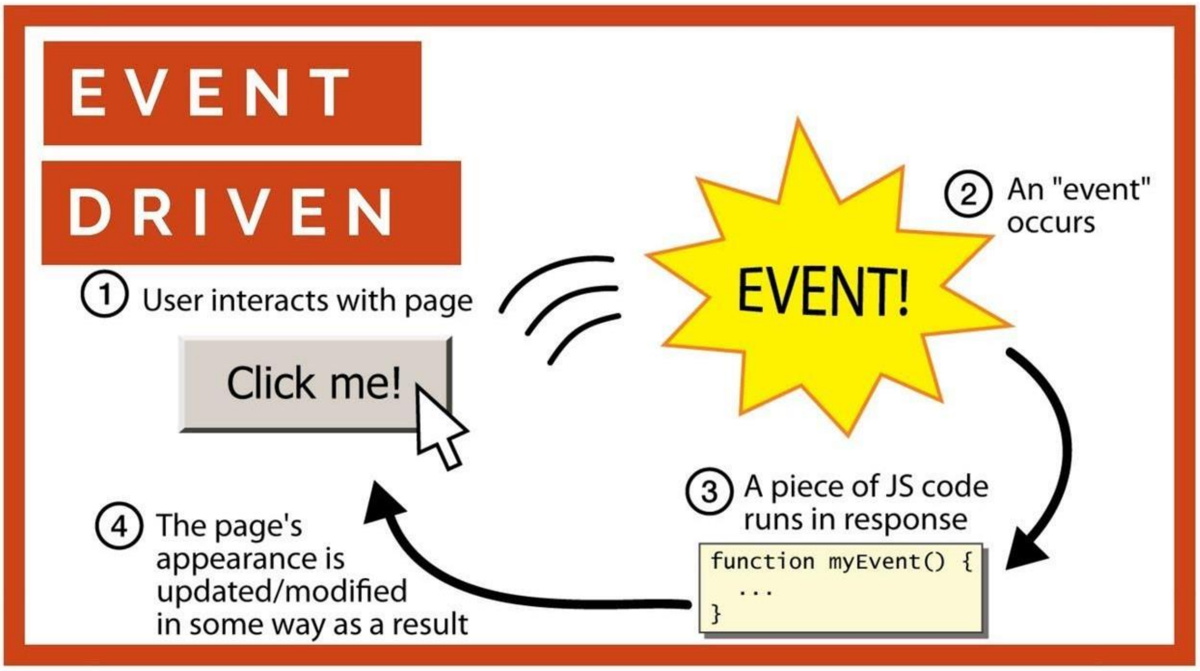
- The push () method "pushes" the elements to the end of the array in theorder they were listed.
- arg1,...argN are elements to be pushed to the end of the array
- It returns the last element added to the end of the array, which is also the last argument in the parameter list.

Example:

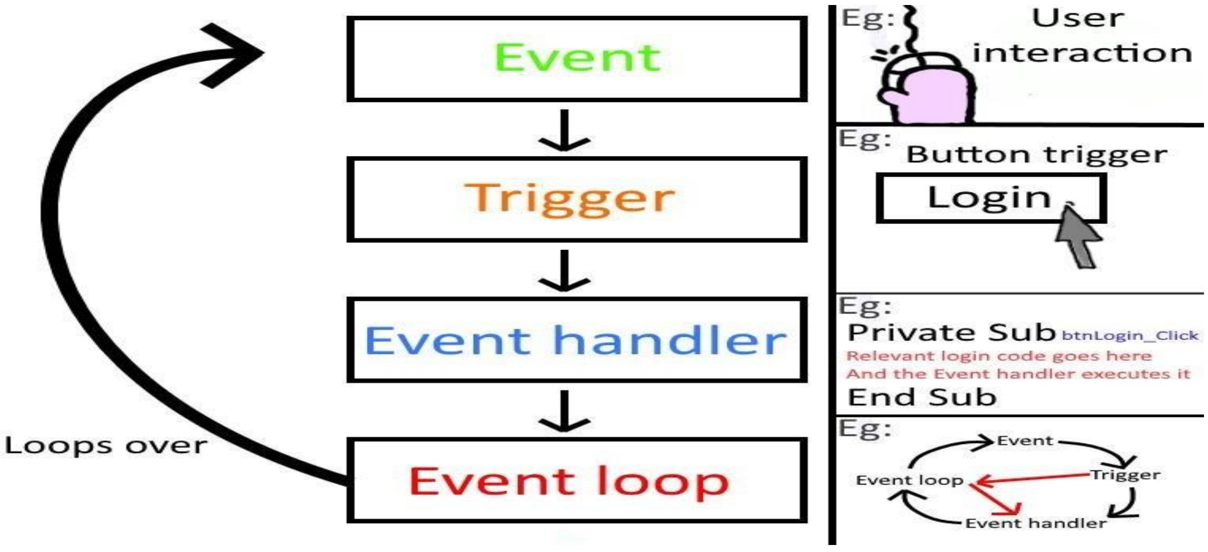
```
<script language =
        "javascript">x = new
        Array("A","B");
         y = x.push("C","D");
        document.write(x[2],"<BR>");
        document.write(x[3]);
</script>
Output: C D
```

- **EXPLAIN EVENTS IN JAVA SCRIPT (5 MARKS)(IMP)(2 OR 3 MARKS EACH)**

HOW EVENTS WORKS

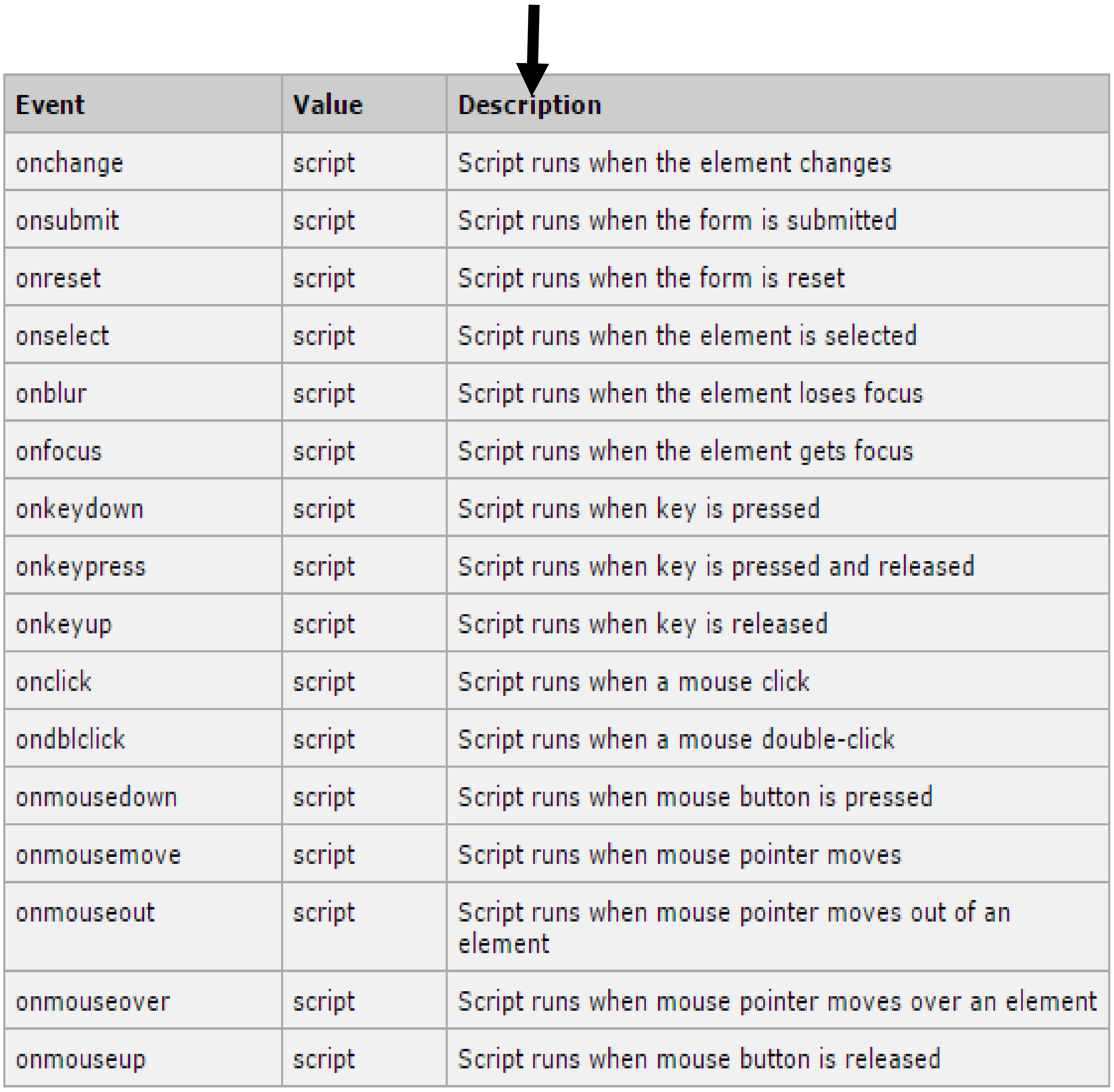

EVENT
DRIVEN
① User interacts with page
Click me!
② An "event" occurs
EVENT!
③ A piece of JS code runs in response
④ The page's appearance is updated/modified in some way as a result

```
function myEvent() {
  ...
}
```



```
Private Sub btnLogin_Click
Relevant login code goes here
And the Event handler executes it
End Sub
```

FOR 1 MARKS



Event	Value	Description
onchange	script	Script runs when the element changes
onsubmit	script	Script runs when the form is submitted
onreset	script	Script runs when the form is reset
onselect	script	Script runs when the element is selected
onblur	script	Script runs when the element loses focus
onfocus	script	Script runs when the element gets focus
onkeydown	script	Script runs when key is pressed
onkeypress	script	Script runs when key is pressed and released
onkeyup	script	Script runs when key is released
onclick	script	Script runs when a mouse click
ondblclick	script	Script runs when a mouse double-click
onmousedown	script	Script runs when mouse button is pressed
onmousemove	script	Script runs when mouse pointer moves
onmouseout	script	Script runs when mouse pointer moves out of an element
onmouseover	script	Script runs when mouse pointer moves over an element
onmouseup	script	Script runs when mouse button is released

1.onkeyup Event : When the user releases the key

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

```
                              function myFunction() {
                    var                  x                  =
                    document.getElementById("fname");
                    x.value = x.value.toUpperCase();
}
                         </script>
                     </head>
                     <body>
                         <p>A function is triggered when the user releases a
                         key in the input field.
                     The function transforms the character to upper case.</p>
                         Enter    your    name:    <input
                                  type="text"      id="fname"
                     onkeyup="myFunction()">
                     </body>
                 </html>
```

Note : A function is triggered when the user releases a key in the input field. The function transforms the character to upper case.

2. onclick Event : When button is clicked

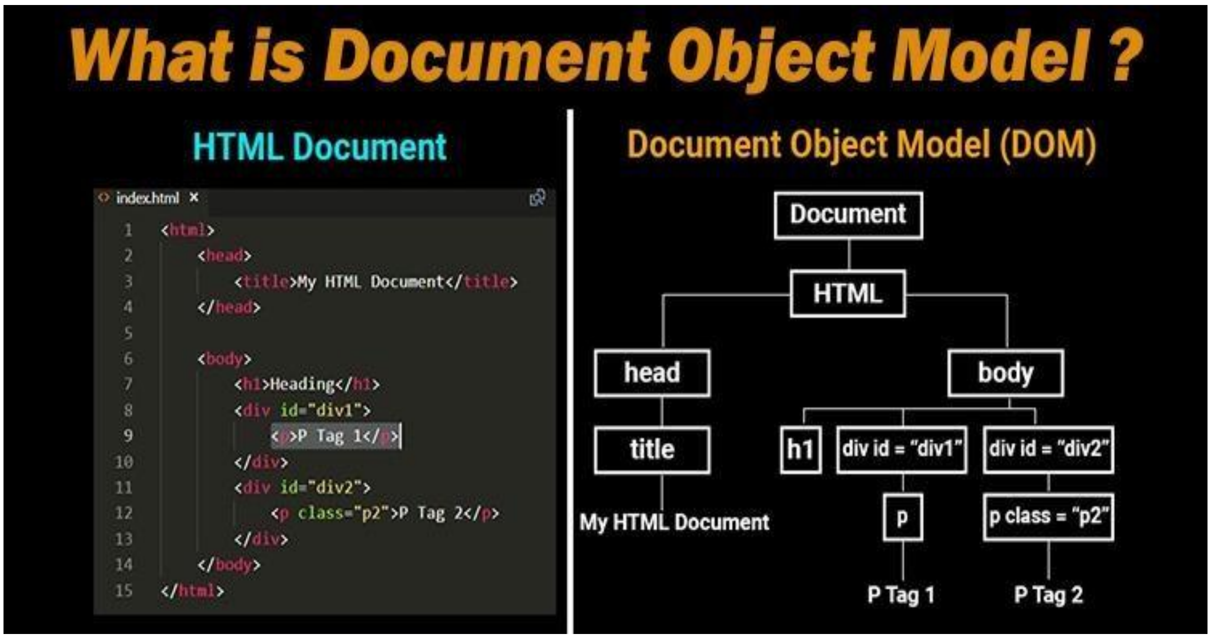
```
     <!DOCTYPE html>
     <html>
     <head>
<script>
 function myFunction()
{ document.getElementById("demo").innerHTML="HelloWorld";
}
                      </script></head>
                      <body>
```

```
                <p>Click the button to trigger a function.</p>
                <button            onclick="myFunction()">Click
                me</button>
                <p id="demo"></p>
            </body>
        </html>
```

3. ondblclick Event : When a text is double clicked

```
        <!DOCTYPE html>
        <html>
            <head>
                <script>
                    function          myFunction()          {
                      document.getElementById("demo")
                      .innerHTML="HelloWorld";
}
</script>
            </head>
            <body>
                <p ondblclick="myFunction()">Doubleclick this
            paragraph to trigger afunction.</p>
                <p id="demo"></p>
            </body>
        </html>
```

➢ EXPLAIN DOCUMENT OBJECT (2 OR 3 MARKS)(IMP)



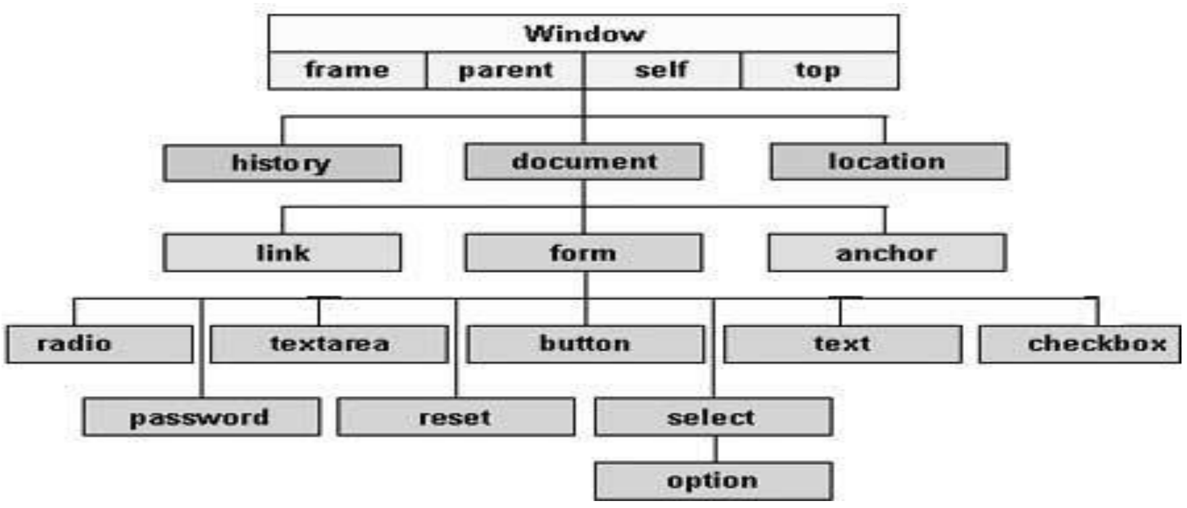
```
<html>
    <head>
        <title>My HTML Document</title>
    </head>

    <body>
        <h1>Heading</h1>
        <div id="div1">
            <p>P Tag 1</p>
        </div>
        <div id="div2">
            <p class="p2">P Tag 2</p>
        </div>
    </body>
</html>
```

Every web page resides inside a browser window which can be considered as an object.

- A Document object represents the HTML document that is displayed in that window. The Document object has various properties that refer to other objects which allow access to and modification of document content.
- The way that document content is accessed and modified is called the **Document Object Model**, or **DOM**. The Objects are organized in a hierarchy. This hierarchical structure applies to the organization of objects in a Web document.
    - **Window object:** Top of the hierarchy. It is the outmost element of the object hierarchy.
    - **Document object:** Each HTML document that gets loaded into a window becomes a document object. The document contains the content of the page.
    - **Form object:** Everything enclosed in the <form>...</form> tags sets the form object.
    - **Form control elements:** The form object contains all

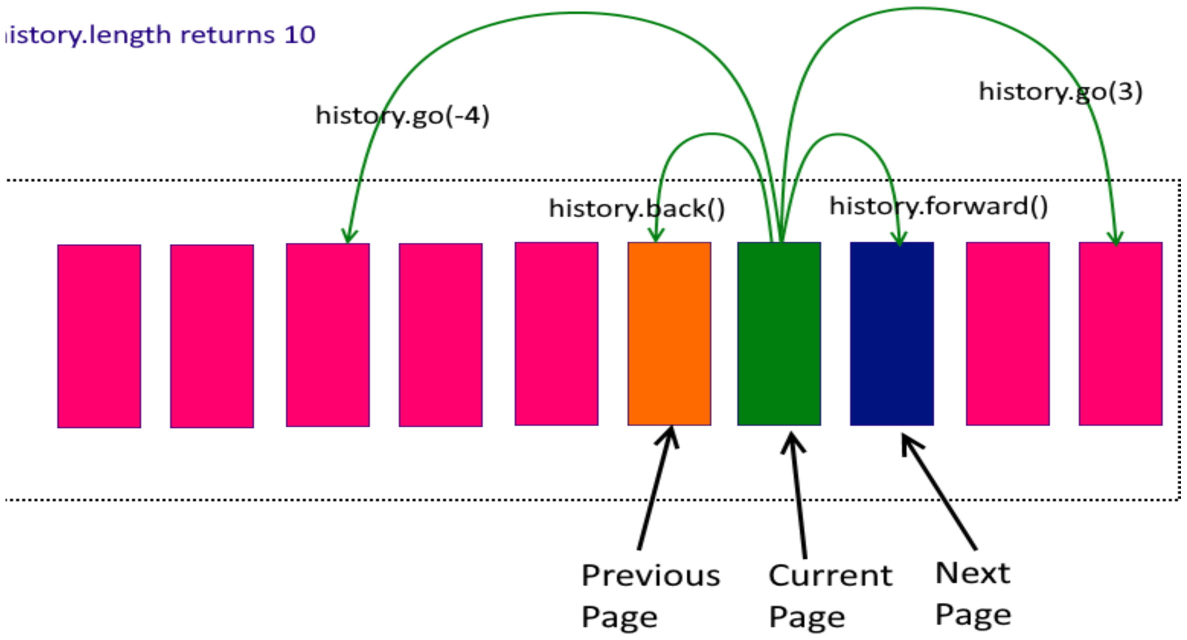
the elements defined for that object such as text fields, buttons, radio buttons, and checkboxes.

- Here is a simple hierarchy of few important objects:



➢ EXPLAIN HISTORY OBJECT(3 MARKS)(MIMP)

HOW HISTORY OBJECT WORKS



- The history property has the return value as history object, which is an array of history items having details of the URL's visited from within that window. Also, note that the History object is a JavaScript object and not an HTML DOM object.
- General syntax of history property of window object:
  - *window.history*
- The JavaScript runtime engine automatically creates this object.
- An introduction on the history object and the properties and methods associated with it was covered in an earlier section.

➢ Property of History Object:

1. **length:**
   - The length property of the history object returns the number of elements in the history list.
   - General syntax of length property of history Object:
     - *history.length*
   - An example to understand the length property of history

object:

2. **current:**
   - This property contains the complete URL of the current History entry.
   - General syntax of current property of history Object:
     - *history.current*
3. **next:**
   - The next property of history object contains the complete URL of the next element in the History list. This functionality or the URL visited is the same as pressing the forward button or menu.
   - General syntax of next property of history Object:
     - *history.next*
4. **previous:**
   - The previous property of history object contains the complete URL of the previous element in the History list. This functionality or the URL visited is the same as pressing the back button or menu.
   - General syntax of previous property of history Object:
     - *history.previous*

➢ **Methods of History Object:**

1. **back():**
   - There may be scenarios where the programmer wishes to load the previous URL present in the history list. In this case, the programmer can make use of the back() method of the history object.

     The back() method of the history object takes the user to the previous page. The functionality results in the same as pressing the back button of the browser.
   - General syntax of back method of

history Object:

history.back()

2. **forward():**

- The forward() method of the history object loads the next URL in the History list. The functionality results are the same as pressing the forward button of the browser.
- General syntax of forward method of history Object:
  - *history.forward()*

3. **go():**

- If the programmer wishes to load a specified URL from the History list, then the go method of history object can be used.
- General syntax of go method of history Object:
  - *history.go(number)*

    or
  - *history.go(URL)*
- The back method seen above is the same as history.go(-1) in terms of go method of history object. The forward method seen above is the same as history.go(1)

- **EXPLAIN NAVIGATOR OBJECT.(2 OR 3 MARKS)(IMP)**
  - The Navigator object of JavaScript returns useful information about the visitor's browser and system.

Properties

Properties	**Description**
appCodeName	The code name of the browser.
appName	The name of the browser (ie: Microsoft Internet Explorer).
appVersion	Version information for the browser (ie: 5.0 (Windows)).
cookieEnabled	Boolean that indicates whether the browser has cookies enabled.
language	Returns the default language of the browser version (ie: en-US). **NS and Firefox only.**
mimeTypes[]	An array of all MIME types supported by the client. **NS and Firefox only.**
platform[]	The platform of the client's computer (ie: Win32).
plugins	An array of all plug-ins currently installed on the client. **NS and Firefox only.**
systemLanguage	Returns the default language of the operating system (ie: en-us). **IE only.**
userLanguage	Returns the preferred language setting of the user (ie: en-ca). **IE only.**

Q.13 <u>EXPLAIN EMAIL VALIDATION & FORM VALIDATION (5 MARKS WITH EXAMPLE OR 3 MARKS)</u>

EMAIL VALIDATION

- Validating email is a very important point while validating an HTML form. In this page we have discussed how to validate an email using JavaScript :
- An email is a string (subset of ASCII characters) separated into two parts by @ symbol. a "personal_info" and a domain, that is personal_info@domain. The length of the personal_info part may be up to 64 characters long and domain name may be up to 253 characters.

- The personal_info part contains the following ASCII characters.
    - Uppercase (A-Z) and lowercase (a-z) English letters.
    - Digits (0-9).
    - Characters ! # $ % & ' * + - / = ? ^ _ ` { | } ~
    - Character. (period, dot or fullstop) provided that it is not the first or last character

      and it will not come one after the other.
    - The domain name [for example com, org, net, in, us, info] part contains letters,digits, hyphens and dots.

FORM VALIDATION

- It is important to validate the form submitted by the user because it can have inappropriate values. So validation is must.
- The JavaScript provides you the facility the validate the form on the client side so processing willbe fast than server-side validation. So, most of the web developers prefer JavaScript form validation.
- Through JavaScript, we can validate name, password, email, date, mobile number etc fields.
- In this example, we are going to validate the name and password. The name can’t be empty and password can’t be less than 6 characters long.
- Here, we are validating the form on form submit. The user will not be forwarded to the next page until given values are correct.