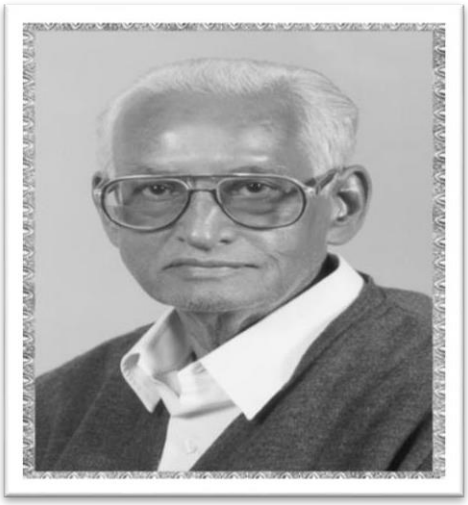
Shree H. N. Shukla College of It. & Mgmt.
(Affiliated To Saurashtra University)



Lt. Shree Chimanbhai Shukla

MSCIT SEM-2 REACTJS

Shree H.N.Shukla college2
vaishali nagar
Near Amrapali Under Bridge,
Raiya road
Rajkot
Ph No:-0281 2440478

Shree H.N.Shukla college3
vaishali nagar
Near Amrapali Under Bridge,
Raiya road
Rajkot
Ph No:-0281 2440478

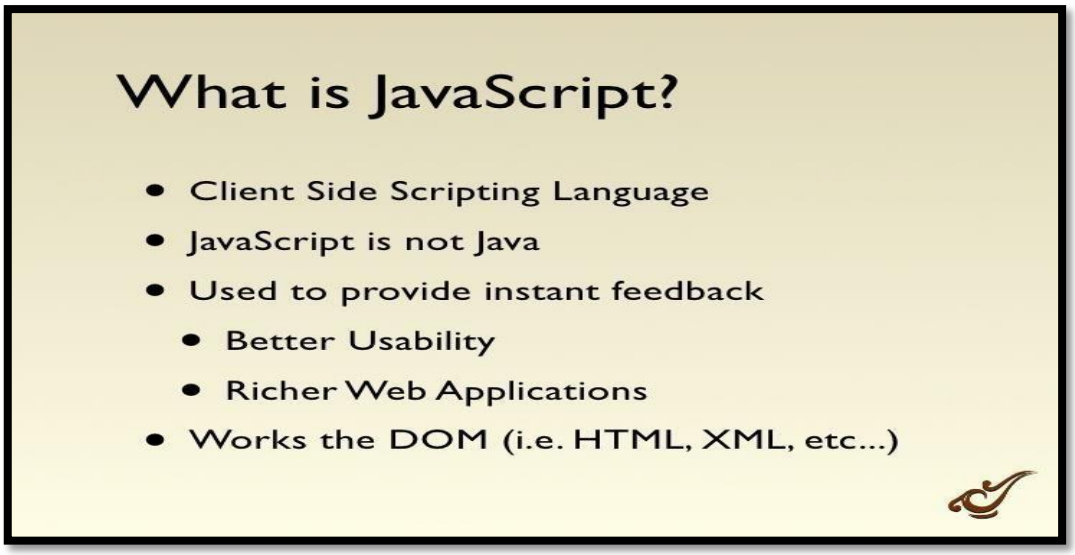
Shree H. N. Shukla College of It. & Mgmt.
(Affiliated To Saurashtra University)

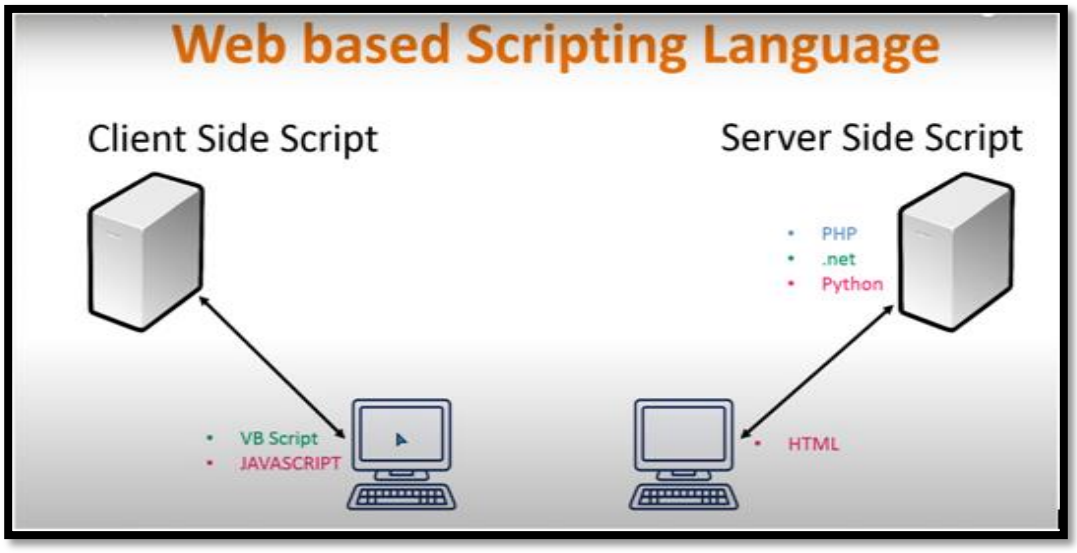
Unit : 1

Express JS & Java Script

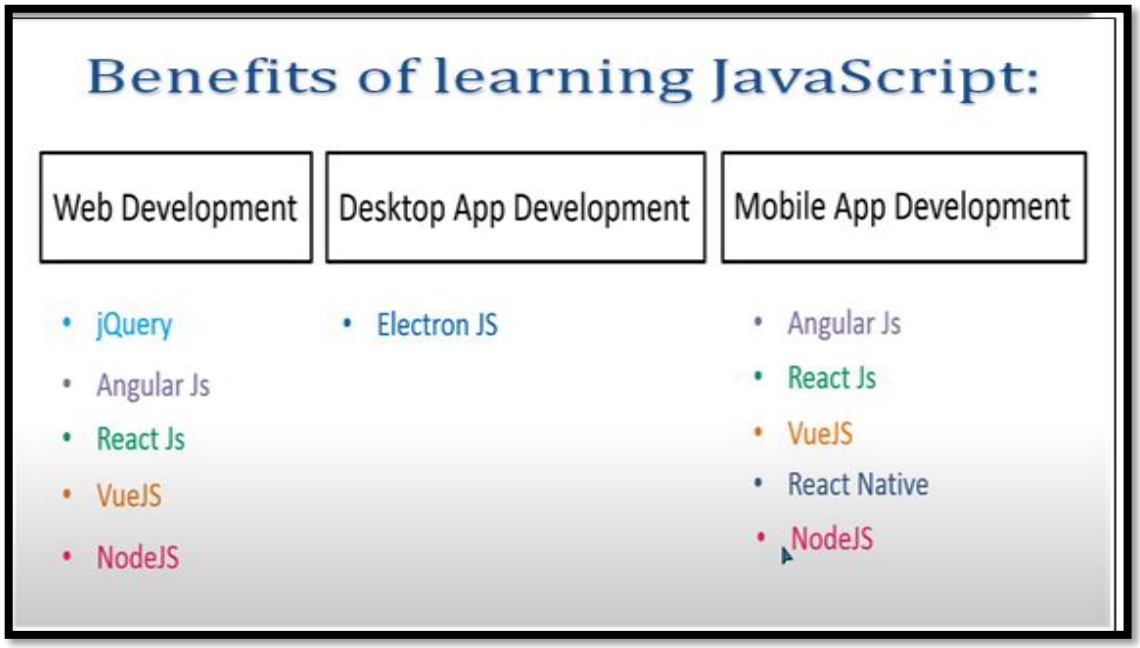
- **Java Script**
- Java Script Overview & Basics
- Variable, Conditional Statements, Loops in JS,
- Functions, Arrays & Events in JS
- ES6 Overview & Basics
- ES6 Classes, functions & Promises
- **Express JS**
- Setting up an app with ExpressJS, Routing in ExpressJS, Connecting views with templates, configurations and error handling.

Java Script





Shree H. N. Shukla College of It. & Mgmt.
(Affiliated To Saurashtra University)



- 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.
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation).
- 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.

Shree H. N. Shukla College of It. & Mgmt.
(Affiliated To Saurashtra University)

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 executed depends onuser's requirement.

Syntax:

```
<HTML>
        <BODY>
        <SCRIPT TYPE = "TEXT/JAVASCRIPT">
. . . .                                          ]  Between
these
tags All
JavaScript code willWritten
        </SCRIPT>
        </BODY>
</HTML>
```

You can also write like **<SCRIPT LANGUAGE = "JAVASCRIPT">**
The **document.write()** command is used to display the messages on the screen.

Shree H. N. Shukla College of It. & Mgmt.
(Affiliated To Saurashtra University)

For Example:

This code will display hello word on browser.

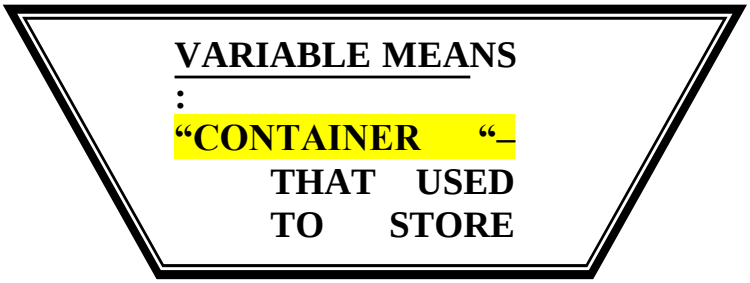
```
<html>
        <Body>
                <script language ="JavaScript">
                document.write('hello word');
                </script>
        </body>
</html>
```

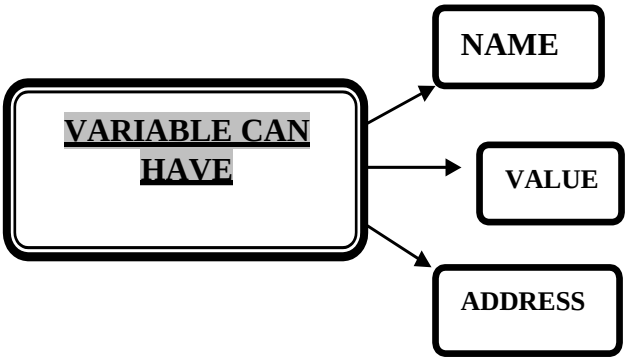
1 MARKS Question – Answer

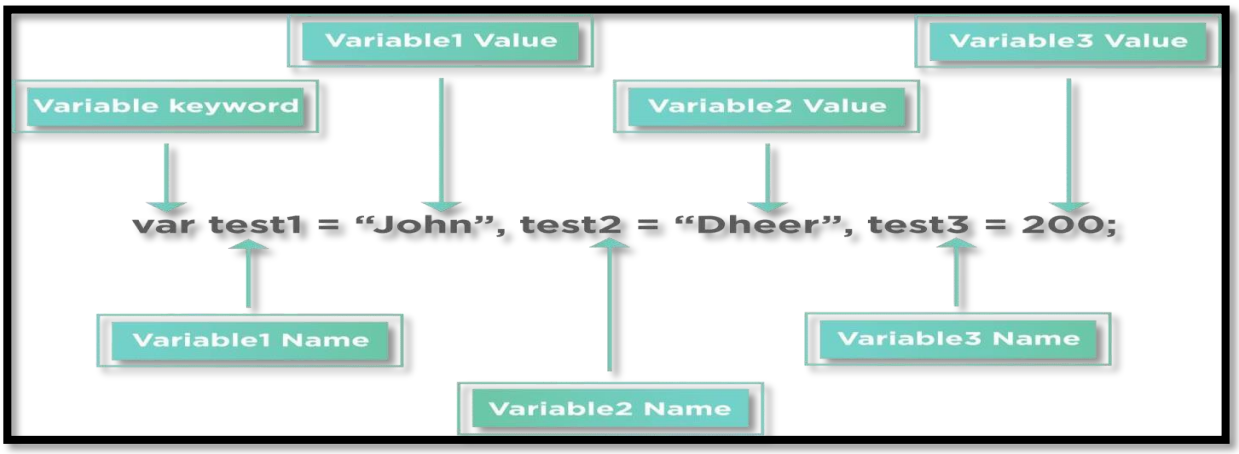
SR.NO.	QUESTION	ANSWER
1	WHAT IS JAVASCRIPT?	CLIENT SIDE S CRIPTING LANGUAGE
2	WHY WE USE JAVASCRIPT	ADD CONTENT TO A WEBPAGE DYNAMICALLY
3	WHICHTAG ISUSE TOINSERT JAVASCRIPT CODE INHTML?	<SCRIPT> TAG
4	IS JAVASCSRIPT IS LIGHTWEIGHT?(YES OR NO)	YES
5	IS JAVASCRIPT IS FREE FOR ALL?(YES ORNO)	YES

Shree H. N. Shukla College of It. & Mgmt.
(Affiliated To Saurashtra University)

EXPLAIN JAVASCRIPT VARIABLES







- 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>

Shree H. N. Shukla College of It. & Mgmt.
(Affiliated To Saurashtra University)

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 willautomatically 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)
```

- A variable declared within a JavaScript function becomes **LOCAL** and can only be accessedwithin that function. (The variable has local scope).
- You can have local variables with the same name in different functions, because local variables are only recognized by the function in which they are declared.
- Local variables are destroyed when you exit the function.
- Variables declared outside a function become **GLOBAL**, and all scripts and functions onthe web page can access it.
- Global variables are destroyed when you close the page.
- If you declare a variable, without using "var", the variable always becomes **GLOBAL.**

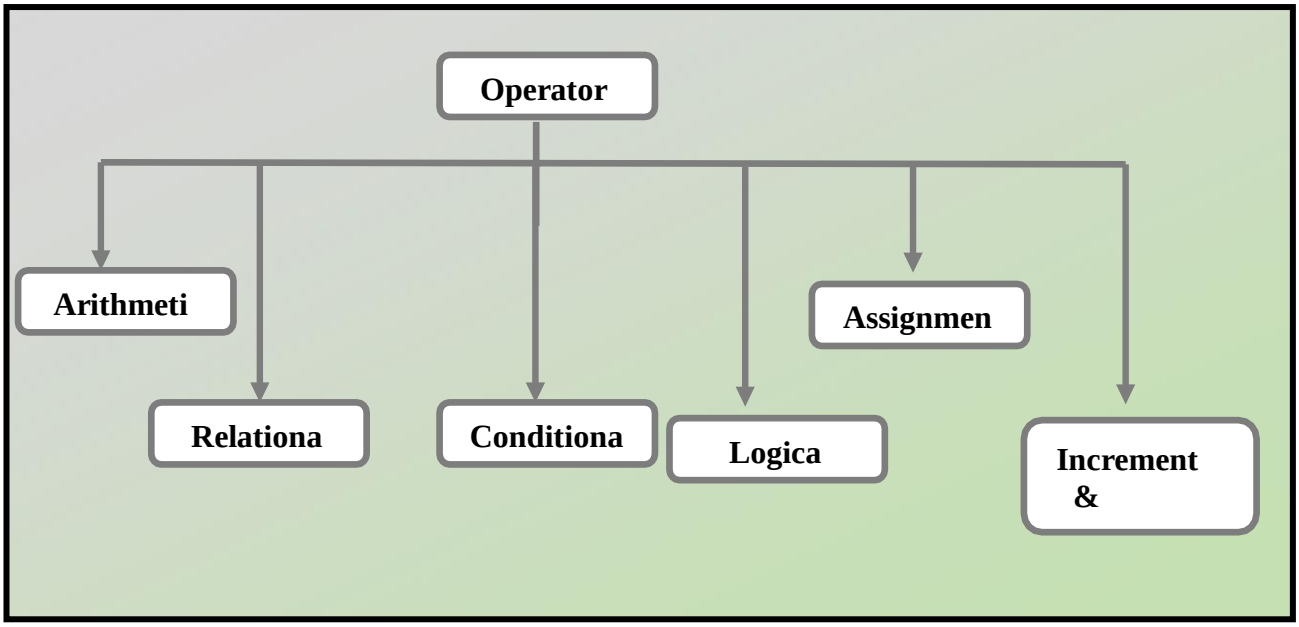
1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	Variable is used to store different types of	values
2	Variable can store on different _____	Memory Location
3	Variable can have _____	Unique Address

Shree H. N. Shukla College of It. & Mgmt.
(Affiliated To Saurashtra University)

- **EXPLAIN JAVASCRIPT OPERATOR**

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



Shree H. N. Shukla College of It. & Mgmt.
(Affiliated To Saurashtra University)

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:

Operator	Example	Same As	Result
=	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

Shree H. N. Shukla College of It. & Mgmt.
(Affiliated To Saurashtra University)

COMPARISION OPERATOR:

- Comparison operators are used in logical statements to determine equality ordifference 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 Yong");}

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

Shree H. N. Shukla College of It. & Mgmt.
(Affiliated To Saurashtra University)

CONDITIONAL OPERATORS:

- JavaScript also contains a conditional operator that assigns avalue to avariable 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”

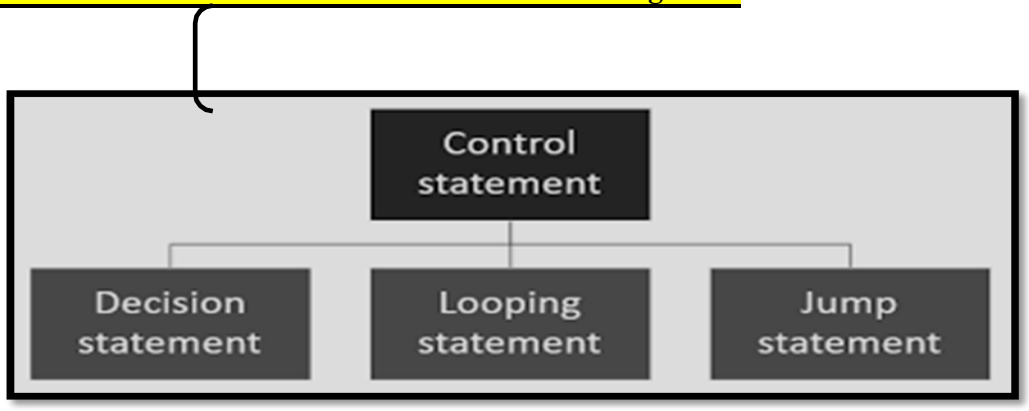
1 Word Question – Answer

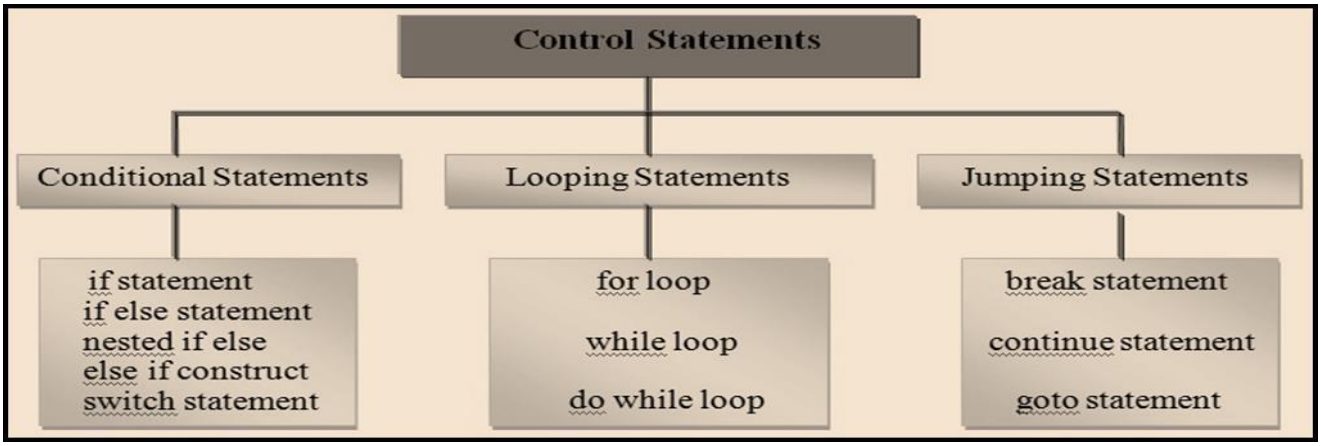
SR.NO.	QUESTION	ANSWER
1	Conditional Operator in c language can be and	? and :
2	_____operator can find how many bytes anoperand can occupies	Sizeof
3	_____operator can used to link relatedexpression together.	Comma
4	_____operator is known as assignment operator	=(equal to)
5	If condition is false , logical operator can return _	0
6	Precedence of operator is known as _____	Hierarchy

Shree H. N. Shukla College of It. & Mgmt.
(Affiliated To Saurashtra University)

Q.4. EXPLAIN CONDITIONAL STATEMENTS

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





If Statement :-

" To check condition & returnresult "

IF types / Flavors :-

Shree H. N. Shukla College of It. & Mgmt.
(Affiliated To Saurashtra University)

Conditional statements in JavaScript are used to perform different actions based ondifferent conditions.

- Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.
- 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) willgenerate a *JavaScript* error!

Shree H. N. Shukla College of It. & Mgmt.
(Affiliated To Saurashtra University)

IF...ELSE STATEMENT:

If you want to execute some code if a condition is true and another code if the 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 executionof 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;}
```

Shree H. N. Shukla College of It. & Mgmt.
(Affiliated To Saurashtra University)

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>
```

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	How Many Control Structures available in JS ? Givename.	5 (two) If Statement Switch Statement
2	If any condition become false ,statement following _____will be execute.	Else
3	In nested if first of all___condition will be checked.	Outer
4	How mamy flavors/types of If statement.Give name.	4(four) Simple if If else If…else..if…elseNested if
5	Which indicate easy way to represent multiple conditions at the same time?	Else…if

Shree H. N. Shukla College of It. & Mgmt.

(Affiliated To Saurashtra University)

➢ SWITCH STATEMENT:

SWITCH CASE :-
" MULTI WAY DECISION

CASE :- WE CAN CREATE *ANY NUMBER OF CASE VALUES* INSIDE SINGLE

DEFAULT :- IT EXECUTE WHEN NO ANY CASE VALUE

- 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.

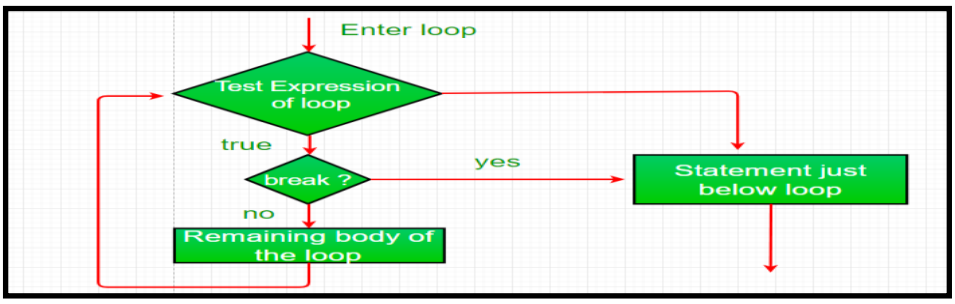
- The value of the expression is then compared with the values for each case in the structure.
- If there is a match, the block of code associated with that case is executed.
- Use break to prevent the code from running into the next case automatically.

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	Switch represent _____	Multiway Decision Statement
2	If no any case value match with condition thenstatement following_will execute.	Default
3	_____statement is used to terminate particular case.	Break
4	In switch case statement , any case value will befollowed by____sign.	: (Colon)
5	Write down syntax to represent switch statement	Switch(Expression)

Shree H. N. Shukla College of It. & Mgmt.
(Affiliated To Saurashtra University)

- **EXPLAIN JAVASCRIPT BREAK & CONTINUE STATEMENTS**



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 this tutorial. It was used to "jump out" of a switch() statement.

The break statement can also be used to jump out of a loop.

The **break statement** breaks the loop and continues executing the code after

the loop (if any):
Since the if statement has only one single line of code, thebraces can beomitt

Example

```
for (var i = 0; i < 10; i++)
{
if (i == 3)
{
break;
}
text += "The number is " + i + "<br>";
}
for (var i = 0; i < 10; i++)

{if (i == 3) break;
text += "The number is " + i + "<br>";}
```

Shree H. N. Shukla College of It. & Mgmt.

(Affiliated To Saurashtra University)

➢ EXPLAIN DIALOG BOXES IN JAVASCRIPT

In JavaScript we can create three kinds of popup boxes:

Alertbox, 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 beforeentering 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”);

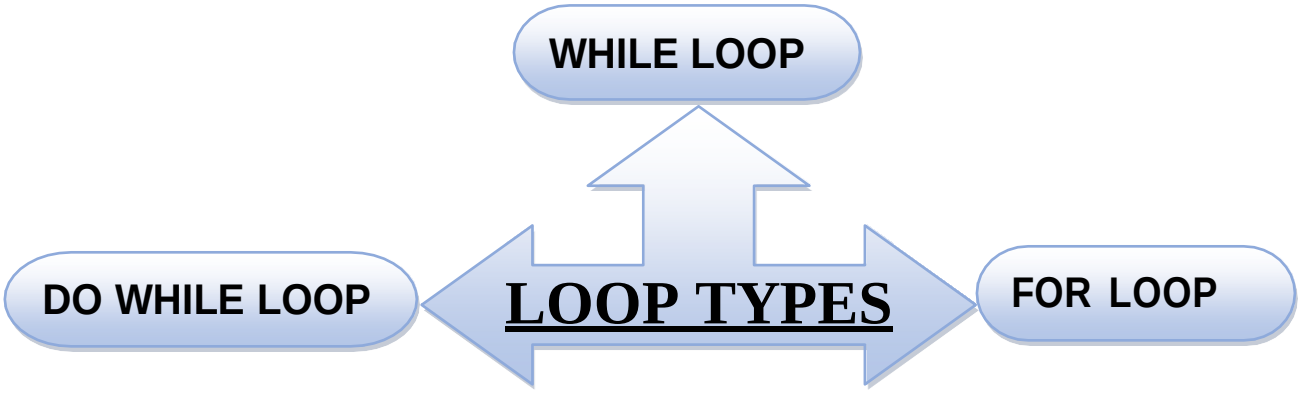
Example:

```
<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

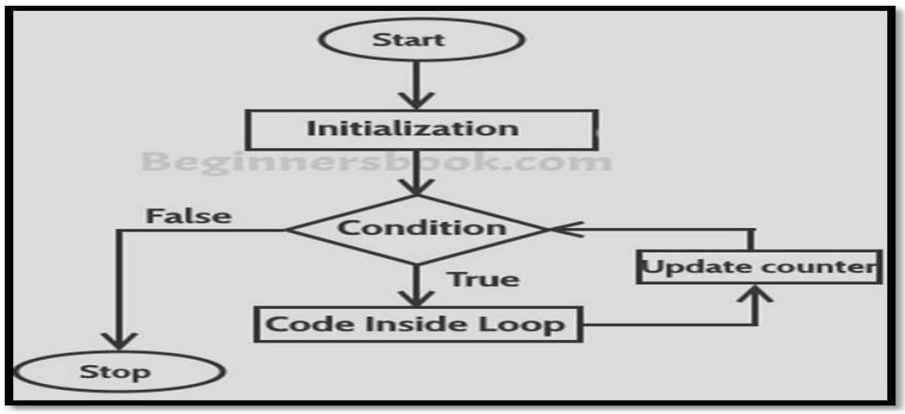
Loop Means “ to execute block of code repeatedly”.

Looping statements also called “Iterative statements”.

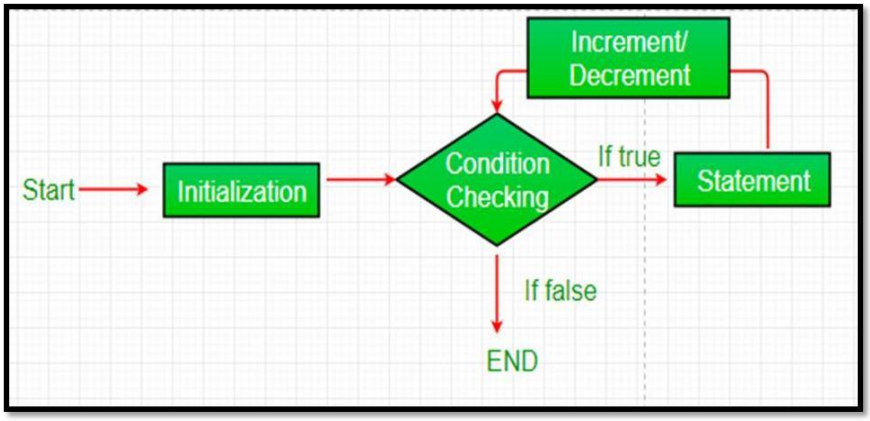


Shree H. N. Shukla College of It. & Mgmt.
(Affiliated To Saurashtra University)

LOOP EXECUTION :-



- 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 certainconditions 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.



Shree H. N. Shukla College of It. & Mgmt.

(Affiliated To Saurashtra University)

FOR LOOP:

Syntax

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

- Expression1 sets up a counter variable and assigns the initial value.
- Condition specifies the final value for the loop to fire (i.e. the loop fires tillcondition evaluates to true).
- Expression2 specifies how the initial value in expression1is incremented.

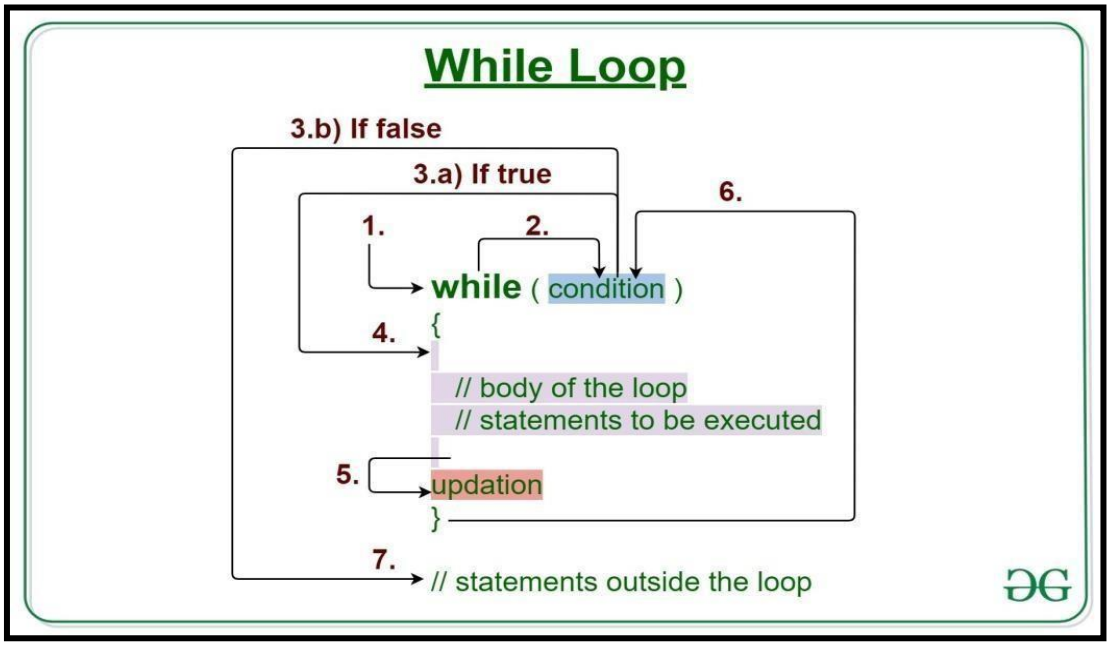
Example:

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

Shree H. N. Shukla College of It. & Mgmt.

(Affiliated To Saurashtra University)

WHILE LOOP:



Syntax

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

- Where, the **condition** is a valid JavaScript expression that evaluates to a Booleanvalue.
- The JavaScript commands execute as long as the condition is true.

Shree H. N. Shukla College of It. & Mgmt.
(Affiliated To Saurashtra University)

Example

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

- **Give difference between While loop & Do .. While loop.**

While Loop	**Do.While Loop**
1)While loop is known as EntryControlled Loop.	1)Do..While loop is known as Exitcontrolled Loop.
2) In While loop if condition become falsethen no any output will be given.	2)n Do..While loop if the conditionbecome true than atleast one loop will be execute &output will be display.
3)There is no terminating semicolon(;) atthe end of loop.	3)There is terminating semicolon(;) at the end of do..while loop.

4)Syntax :
While (<condition>
{<Statements>
}
	4)Synt
	ax :do
	{<Statements>
	} While (<condition>;

Shree H. N. Shukla College of It. & Mgmt.

(Affiliated To Saurashtra University)

Give difference between While loop & For loop

While Loop	For Loop
1)While loop is bit slower than for loop.	1)For loop is faster than while loop.
2) While loop is not suitable for simple Initialization.	2)For loop is more suitable when there is Simple initialization.

3)Syntax :
While (<condition>
{
<Statements>
}
	3)Syntax :
	For(<Initialization>;<Condition>;
	<Inc./Dec.>)
	{
	<Statements>
	}

Give difference between Break & Continue.

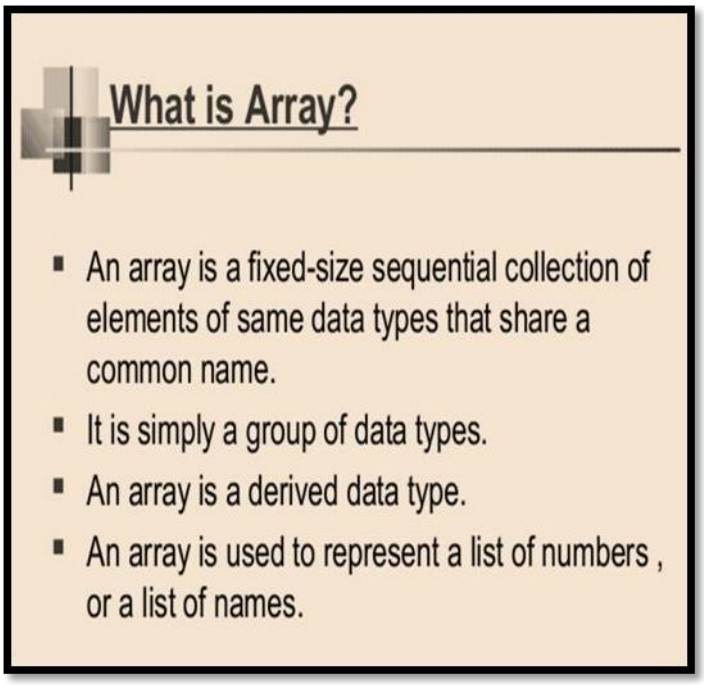
WhileLoop	ForLoop
1) Break is used to terminate the block &get the control out of the loop.	1) Continue is used to get the control to the next iteration of theloop.
2) Break statement can be used in bothswitch case & loop.	2) Continue statement can be only used with looping statement.
3)Syntax : <break>;	3)Syntax : <continue>;

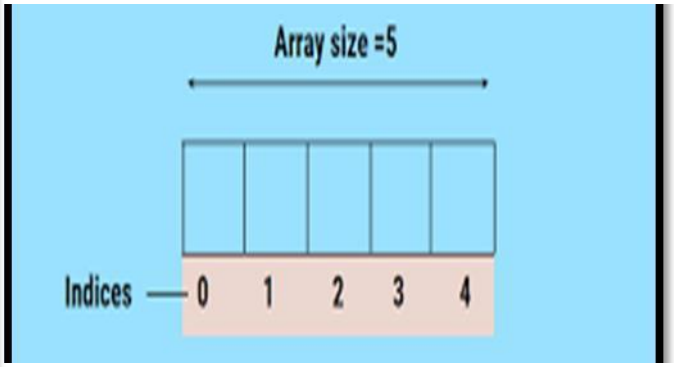
1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	Looping statements are also known as _____ statements	Iterative
2	_____Loop is known as Entry Control Loop.	While
3	_____Loop is known as Exit control Loop.	Do While
4	In Do while Loop condition will be terminated with_____.	Semi Colon(;)
5	_____Loop is known as faster loop in c.	For

Shree H. N. Shukla College of It. & Mgmt.
(Affiliated To Saurashtra University)

EXPLAIN ARRAY IN JAVA SCRIPT





- 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), storingthe 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. Andyou 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.
- An array can be defined in three ways.

The following code creates an Array object called myCars:

```
varmyCars=new Array();
    //regular array
    myCars[0]="Saab";
    myCars[1]="Volv";
    myCars[2]="BMW";

varmyCars=new
Array("Saab","Volvo","BMW");                                    //
condensed array

var myCars=["Saab","Volvo","BMW"];
 // literal array
```

Access an Array

- You can refer to a particular element in an array by referring to the name ofthe array and the index number.
- The index number starts at 0.

EXAMPLE: document.write(myCars[0]);

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	What is Array?	Group of Elements having samename and type.
2	Array is_____datatype.	Derived
3	Array is used to represent _____	Collection
4	Types of array can be_____& _____	Single/One dimension& Multi/Two dimension

Function

- Functions offer the ability to group together JavaScript program code that performs a specific 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 thefunction when it was declared.
- Functions can accept information in the form of arguments and can return results.
- Appropriate syntax needs to be followed for declaring functions,invoking them,passing them values and accepting their return values.

Declaring Function:

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

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

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

Place of Declaration:

i) Functions can be declared anywhere within an HTML file.

ii) Preferably, functions are created within the <HEAD>…<HEAD> tags of the HTMLfile.

iii) This ensures that all functions will be *parsed* before they are invoked or called.

iv) If the function is called before it is declared and parsed, it will lead to an errorcondition.

```
<html>
   <head>
 <script language="javascript">
                        function printnm() {
              var user="shri";
              document.write("name is : ");
                                document.write(user);

                    }
</script></head>
   <body onLoad="printnm();">

   </body></html>
```

Passing Parameters:

- Values can be passed to function parameters when a 'parameterized' function is called. Values are passed to the function by listing them in the parentheses following the function name.
- Multiple values can be passed, separated by commas provided that the function has been coded to accept multiple parameters.

The return Statement

- The return statement is used to specify the value that is returned from thefunction.
- So, functions that are going to return a value must use the return statement.

Example:

```
function prod(a, b)
{
        x=a*b;
        return x;
 }
Product=prod (2, 3);
```

Now variable product contains the 6;

1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	FULL FORM OF UDF	USER DEFINE FUNCTION
2	USE OF UDF_____	CREATE OUR OWN FUNCTION
3	WHICH KEY WORD IS USED TO CREATE UDF?	FUNCTION KEYWORD

Shree H. N. Shukla College of It. & Mgmt.

(Affiliated To Saurashtra University)

BUILT IN FUNCTION IN JAVASCRIPT

STRING FUNCTION:

1) **charAt()**

Syntax: <string>.charAt(num);

Purpose:

- This function returns the character located at position passed in to theargument.
- This indexing is done from left to right starting with the 0 (zero) position.
- If the num passed is not a valid index in the string, -1 is returned.

1) **2 charAt()**

Syntax: <string>.charAt(num);

Purpose:

- This function returns the character located at position passed in to theargument.
- This indexing is done from left to right starting with the 0 (zero) position.
- If the num passed is not a valid index in the string, -1 is returned.

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.
- In argument of this function user can give more than one string.

Example:

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

3) indexOf():

Syntax: <string>.indexOf(<string or char to search>, [index]);

Purpose:

- This method is used to find the position of specified string or charstarting from specified index.
- This method starts to find string from starting position (left to right).
- Here index in argument is optional if it is not defined that start form zero position.

Example:

```
<script
        language="JavaScript
        ">x = "Hello World!";
        document.write("index of l is from 5th position
"+x.indexOf("l",5)+'<br>');
        document.write("index of l is "+x.indexOf("l"));
</script>
```

Output: index of l is from 5th position 9 index of l is 2

4) lastIndexOf():

Syntax: <string>.lastIndexOf(<string or char to search>, [index]);

Purpose:

- This method is used to find the position of specified string or charstarting from specified index.
- This method starts to find string from ending position (right to left).

Example:

```
<script
        language="JavaScript
        ">x = "Hello World!";
        document.write("index of l is from 5th position
"+x.lastIndexOf("l",5)+'<br>');
        document.write("index of l is "+x.lastIndexOf("l"));
</script>
```

Output: index of l is from 5th position 3 index of l is 9

5) 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 withnew 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

6) Search:

Syntax: <string>.search(<string to be search> or <regular expression>);

Purpose:

- This method is used to find the specified string position of main string.
- If string is match then it will return the position.
- If string is not match then it will return -1.

Example:

```
<script language="JavaScript1.2">
        x = "This is main string to be
        replace"y = x.search("string");
        document.write("the position is
        "+y);
</script>
```

Output: the position is 13

7) substr():

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

Purpose:

- This method returns the string from position specified in num1 with noof 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 fromlast 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
```

8) substring():

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

Purpose:

- The substring() method returns the characters starting with the indexed position num1 and ending with the character before num2.
- The string's first character is in position 0.
- If you pass num1 as a negative number, it will be treated as 0.
- If you pass num2 as a value greater than the string.length property, it willbe treated as string.length.
- If num1 equals num2, an empty string is returned.
- It is also possible to pass a single index location to the method.

```
Example:<script language="JavaScript1.1">
 x = "This is a test";
          y = x.substring(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
```

9) 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

10) toLowerCase():

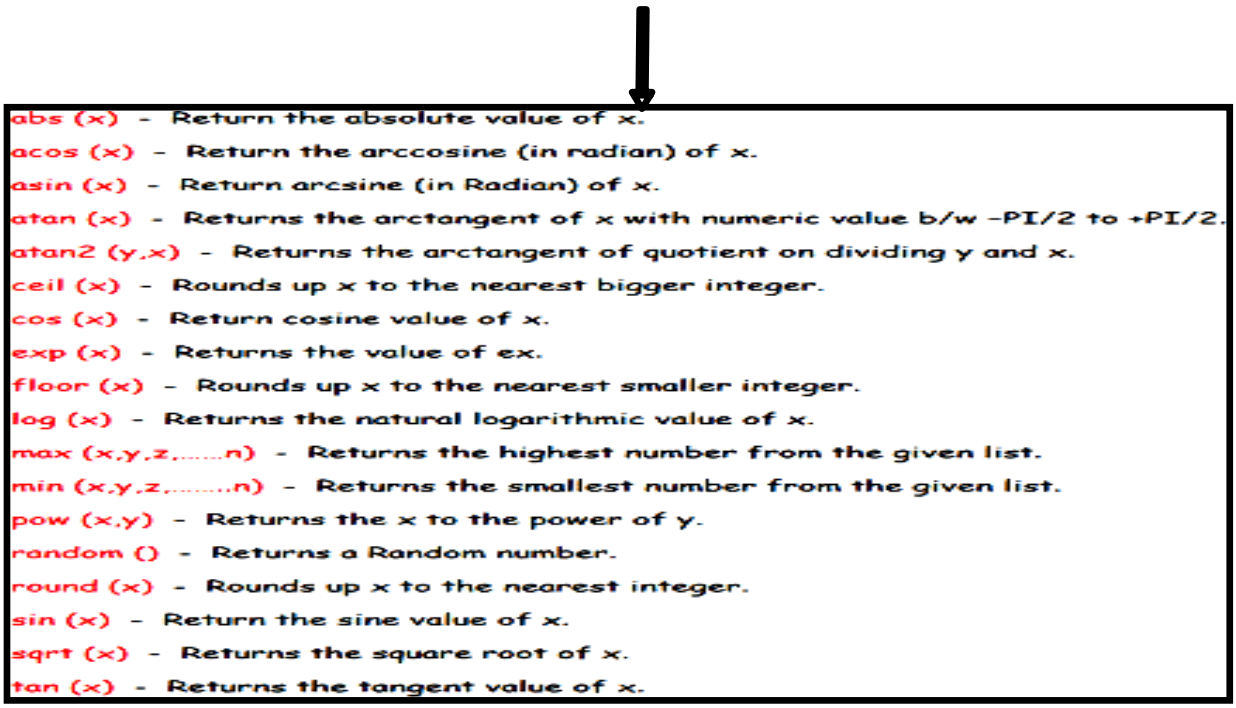
Syntax: <string>.toLowerCase();

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



1) abs()

Syntax: Math.abs(num)

Purpose:

o 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:

o Get the largest integer number, which is equivalent toor less thanthe 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 arandom 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 roundedto the next highest integer.
- If the fractional portion of number is less than .5, the result is rounded to thenext 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 twoparameters 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 ofthe 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:

Name	Description
getDate()	The day of the month as an integer from 1 to 31
getDay()	The day of the week as an integer where 0 is Sunday and 1 is Monday
getHours()	The hour as an integer between 0 and 23
getMinutes()	The minutes as an integer between 0 and 59
getMonth()	The month as an integer between 0 and 11 where 0 is January and 11 is December
getSeconds()	The seconds as an integer between 0 and 59
getTime()	The current time in milliseconds where 0 is January 1, 1970, 00:00:00
getYear()	The year, but this format differs from browser to browser

Syntax: <variable name> = new Date()

1) 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
- Sunday is starting from 0 and onwards.

Example:

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

Output: 6

4) getMonth():

Syntax: <date object>.getMonth();

Purpose:

o The getMonth() method returns the month portion of the Date

o bject expressed as an integer from 0 (January) to 11 (December).

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

5) getYear():

Syntax: <date object>.getYear();

Purpose:

o 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:

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

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

3) getMinutes():

Syntax: <date object>.getMinutes();

Purpose:

This method returns the current minute of the date expressed as an integerfrom 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 Dateobject 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 dateexpressed as an integer from 0 to 999.

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

8) setDate():

Syntax: <date object>.setDate(Day);

Purpose:

- The setDate() method sets the day of the month in the Date object to the argument day, an integer from 1 to 31.
- The method returns an integer representing the number of milliseconds between midnight January 1, 1970 (GMT) to the date and time specified in the Date object after the day of the month has been adjusted.

```
Example:
<script language="JavaScript">x=
        new Date();
        document.write(x.setDate(17));
</script>
Output: 1303051102710
```

9) setMonth():

Syntax: <date object>.setMonth(month);

Purpose:

- The setMonth() method sets the month in the Date object.
- The argument month is an integer from 0 (January) to 11 (December).
- The method returns an integer representing the number of milliseconds between midnight January 1, 1970 (GMT) to the new date.

Example:

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

Output: 1309618547995

10) setYear():

Syntax: <date object>.setYear(Year);

Purpose:

- The setYear() method sets the year in the Date object to the argument year. The argument can be either a four-digit or two-digit integer.
- The method returns an integer representing the number of milliseconds between midnight January 1, 1970 (GMT) to the new date.

Example:

```
<script language="JavaScript">
        x= new Date();
        document.write(x.setYear(1987));
</script>
```

Output: 546793593059

Syntax: <date object>.setFullYear(year);

11) setFullYear():

Purpose:

- The setFullYear() method sets the year in the Date object to the argumentyear, a four-digit integer.
- The method returns an integer representing the number of milliseconds between midnight January 1, 1970 (GMT) to the new date after setting.

Example:

```
<script
language="JavaScript">
x= new Date();
        document.write(x.setFullYear(
        1987));
```

12) setHours():

Syntax: <date object>.setHours(Hours);

Purpose:

- The setHours() method sets the hour in the Date object to the argumenthours, an integer from 0 (12:00 a.m. midnight) to 23 (11:00 p.m.).
- The method returns an integer representing the number of milliseconds between midnight January 1, 1970 (GMT) to the new date.

Example:

```
<script language =
        "javascript">x = new
        Date(); x.setHours(4);
        document.write(x);
</script>
```

Output: Sat Apr 30 2011 04:50:07 GMT+0530 (India Standard Time)

13) setMinutes():

Syntax: <date object>.setMinutes(minutes);

Purpose:

- The setMinutes() method sets the minutes in the Date object to theargument minutes, an integer from 0 to 59.

The method returns an integer representing the number of milliseconds between midnight January 1, 1970 (GMT) to the new date.

Example:
```
<script language =
        "javascript">x = new
        Date(); x.setMinutes(4);
        document.write(x);
</script>
```
Output: Sat Apr 30 2011 04:58:07 GMT+0530 (India Standard Time)

Syntax: <date object>.setSeconds(seconds);

14) setSeconds():

Purpose:

- The setSeconds() method sets the seconds in the Date object to theargument seconds, an integer from 0 to 59.

The method returns an integer representing the number of millisecondsbetween

Example:
```
<script language =
        "javascript">x = new
        Date();
        x.setSeconds(30);
        document.write(x);
</script>
```
Output: Sat Apr 30 2011 04:58:30 GMT+0530 (India Standard Time)

1) join()

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

Purpose:

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

- The reverse () method reverses the order of the elements in the arrayaccording 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'slength 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.
```

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

4) push()

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 thelast 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
```

5) shift()

Syntax: <array object>.shift ();

Purpose:

- The shift () method deletes and returns the first element of the array.
- Once deleted, all the remaining elements are shifted down one spot.
- The first position is filled by the element that was previously in the secondposition.

```
Example:
<script language = "javascript">
        x = new Array("A","B","C");
        document.write("before shift:
        ",x[0],"<br>");y = x.shift();
        document.write("after shif: ",x[0],"<br>");
</script>
        Output: before shift: A  after shift: B
```

6) sort():

Syntax: <array object>.sort([sort function]);

Purpose:

- The sort () method rearranges the elements of the array based on a sortingorder.
- If the method has no parameters, JavaScript attempts to convert all the elements of the array to strings and then sort them alphabetically.
- If array contains numeric value then following function is needed for sortingan array.

```
Function x (a, b)     // this is for ascending order
{
return a - b;
}
```

OR

```
Function x (a, b)     // this is for descending order
{
return b - a;
}
```

Example:

```
<script type="text/javascript">
        function x(a,b)
        {
                return a - b;
        }
        var n = ["10", "5", "40", "25", "100", "1"];
        document.write(n.sort(x));
</script>
```

Output: 1, 5, 10, 25, 40, 100

String

String Object Methods

Method	Description
charAt()	returns the character at the specified index
concat()	joins two or more strings, and returns a copy of the joined strings
indexOf()	returns the position of the first found occurence of a specified value in a string
lastIndexOf()	returns the position of the last found occurence of a specified value in a string
replace()	searches for a match between a substring (or regex) and a string, and replaces the matched substring with a new substring
search()	searches for a match between a regex and a string, and returns the position of the match
slice()	extracts a part of a string and returns a new string
split()	splits a string into an array of substrings
substr()	extracts the characters from a string, beginning at a specified start position, and through the specified number of character
substring()	extracts the characters from a string, between two specified indices
toLowerCase()	converts a string to lowercase letters
toUpperCase()	converts a string to uppercase letters
trim()	removes whitespace from both ends of a string
valueOf()	returns the primitive value of a String object
toString()	returns the value of a String object

Math Object Methods

Method	Description
abs(x)	returns the absolute value of x
round(x)	rounds x to the nearest integer
ceil(x)	returns x, upwards to the nearest integer
floor(x)	returns x, rounded downwards to the nearest integer
exp(x)	returns the value of E^x
log(x)	returns the narural logarithm (base E) of x
max(x, y, z, ...)	returns the number with the highest value
min(x, y, z, ...)	returns the number with the lowest value
pow(x, y)	returns the value of x to the power of y (x^y)
sqrt(x)	returns the square root of x
random()	returns a random number between 0 and 1
sin(x)	returns the sine of x (x in radians)
cos(x)	returns the cosine of x (x in radians)
tan(x)	returns the tangent of an x

Date Object Methods

Method	Meaning	Units
getFullYear()	4-digit year	2000, 2001, ...
getMonth()	month no.	0 - 11
getDate()	dat of month	0 - 31
getHours()	hour no.	0 - 23
getMinutes()	minute no.	0 - 59
getSeconds()	second no.	0 - 59

Array Object Methods

Method	Description
concat()	joins two or more arrays, and returns a copy of the joined array
indexOf()	search the array for an element and returns its position

Array Object Methods (cont)

join()	joins all elements of an array into a string
lastIndexOf()	search the array for an element, starting at the end, and returns its position
pop()	removes the last element of an array, and returns that element
push()	adds new elements to the end of an array, and returns the new length
reverse()	reverses the order of the elements in an array
shift()	removes the first element of an array, and returns that element
sort()	sorts the elements of an array
toString()	converts an array to a string, and returns the result
valueOf()	returns the primitive value of an array

Number Object Methods

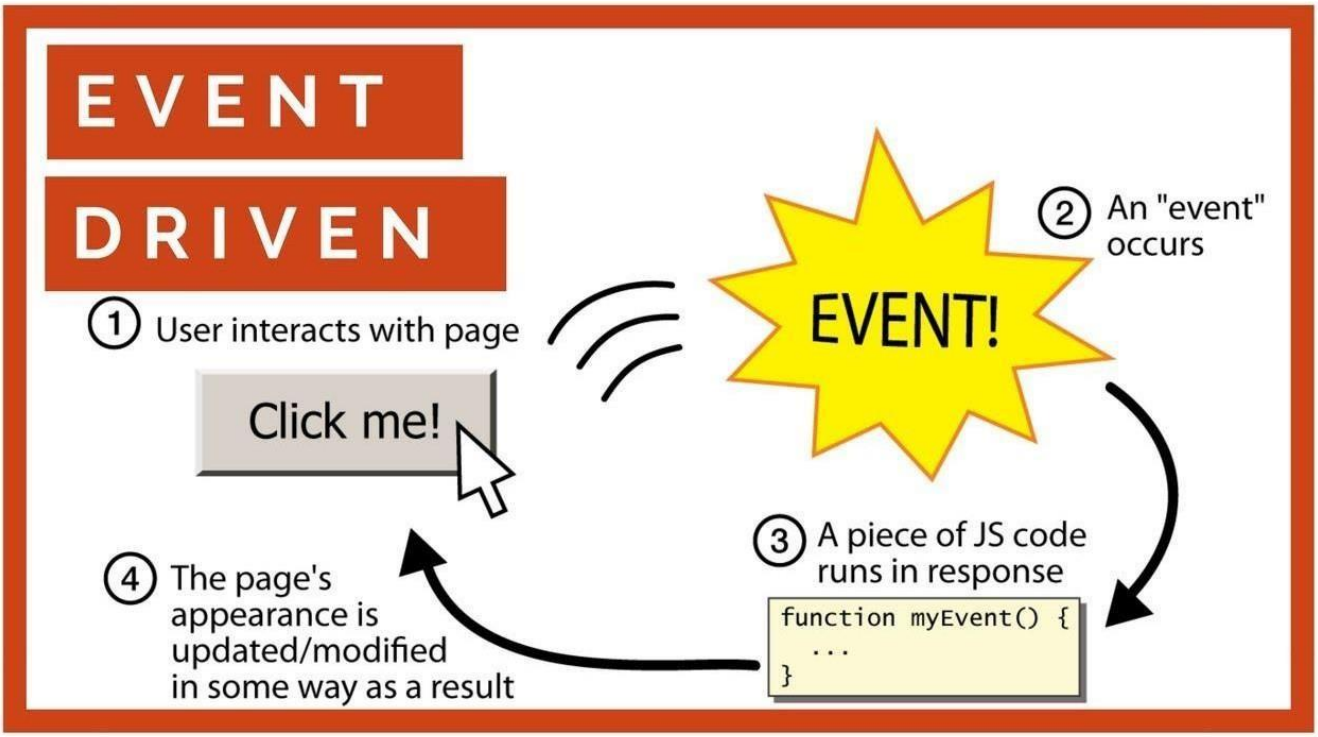
Method	Description
toExponential(x)	converts a number into an exponential notation
toFixed(x)	formats a number with x numbers of digits after the decimal point
toPrecision(x)	formats a number to x length
toString()	converts a Number object to a string
valueOf()	returns the primitive value of a Number object

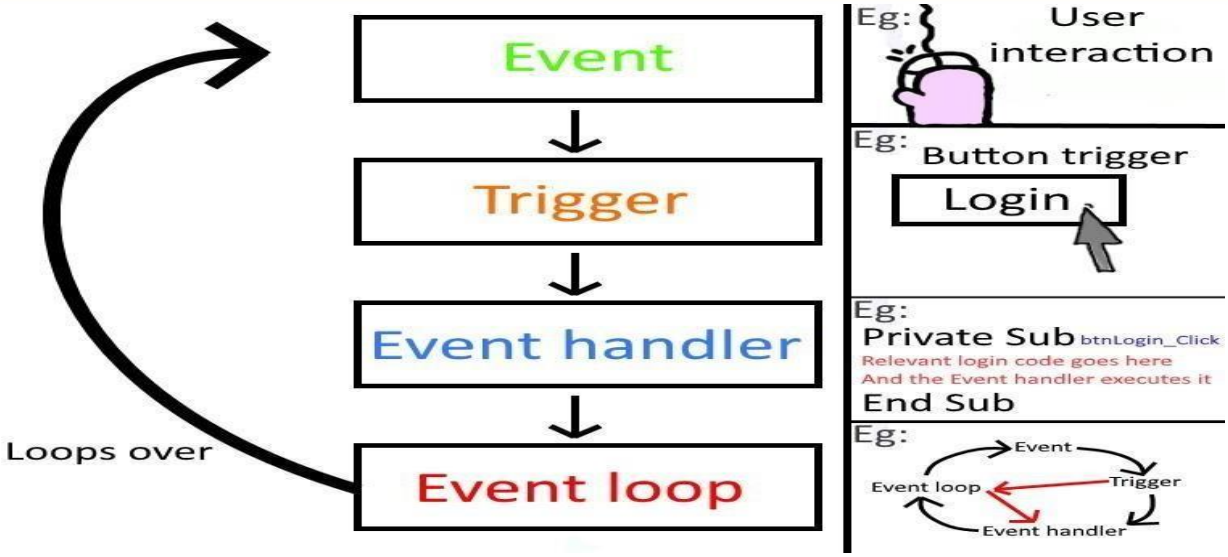
SHREE H. N. SHUKLA COLLEGE OF I.T. & MGMT.

(AFFILIATED TO SAURASHTRA UNIVERSITY)

EXPLAIN EVENTS IN JAVA SCRIPT

HOW EVENTS WORKS





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. onblur Event : When a user leaves an input field

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

function myFunction() {
 var x=document.getElementById("fname");

  x.value=x.value.toUpperCase ();

</script>

                    </head>
                     <body>
                     Enteryourname:<input  type="text"id="fname"
                     onblur="myFunction()">
                           <p>When you leave the input field, a function is triggered

                     whichtransforms the input text to upper case.</p>

                     </body>
                     </html>
```

Note : When you leave the input field, a function is triggered which transforms theinput textto upper case.

2. onblur Event : When a user changes the content of an input field

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction()
{
var x = document.getElementById("fname"); x.value =
x.value.toUpperCase();
}
</script>
</head>
<body>
Enter your name:<input type="text"id="fname"onblur="myFunction()">
<p>When you leave the input field,afunction is triggered whichtransforms the input text to upper case.</p>
</body>
</html>
```

When you leave the input field, a function is triggered which transforms the input text to uppercase.

3. onchange Event : When a user selects a dropdown value

```
<!DOCTYPE html>
<html>
    <head>
        <script>
function preferedBrowser()
{
prefer = document.forms[0].browsers.value;
alert("You prefer browsing internet with " +prefer);
}
</script>
</head>
    <body>
    <form>
Choose which browser you prefer:
<select id="browsers" onchange="preferedBrowser()">
<option value="Chrome">Chrome</option>
<option value="InternetExplorer">InternetExplorer</option>
<option value="Firefox">Firefox</option>
</select>
</form>
    </body>
</html>
```

4. onfocus Event : When an input text field gets focus

```
<!DOCTYPE html>

<html>
    <head>
        <script>
            function  myFunction(x)
{

            x.style.background = "yellow";
}
        </script>
    </head>
    <body>
Enter your name: <input type="text" onfocus="myFunction(this)">
        <p>When the input field gets focus, a function is triggered which changesthe background-color.</p>
    </body>
</html>
```

5. onsubmit Event : When a user clicks the submit button

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

<p>Use the addEventListener() method to attach a "submit" event to a form element.</p>

<p>When you submit the form, a function is triggered which alerts some text.</p>

<form id="myForm" action="/action_page.php">
  Enter name: <input type="text" name="fname">
  <input type="submit" value="Submit">
</form>

<script>
document.getElementById("myForm").addEventListener("submit", myFunction);

function myFunction() {
  alert("The form was submitted");
}
</script>

</body>
</html>
```

6. onreset Event : When a user clicks the reset button

```
<!DOCTYPE html><html><head>
        <script> function message() {
          alert("This alert box was triggered by the onreset event handler");
        }
</script></head>
<body>
    <form onreset="message()">
        Enter your name: <input type="text" size="20">
        <input type="reset">
    </form>
</body>
</html>
```

7. onkeypress Event : When a user is pressing / holding down a key

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
alert("You pressed a key inside the input field");
}
</script>
<</head>
<body>
<p>A fu function is triggered when the user is pressing a key inthe inputfield.</p>
<  <input type="text" onkeypress="myFunction()">
< </body>
<</html>
```

Note : A function is triggered when the user is pressing a key in the inputfield

8. 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 theinput 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.

9. 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>
```

10. 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>
```

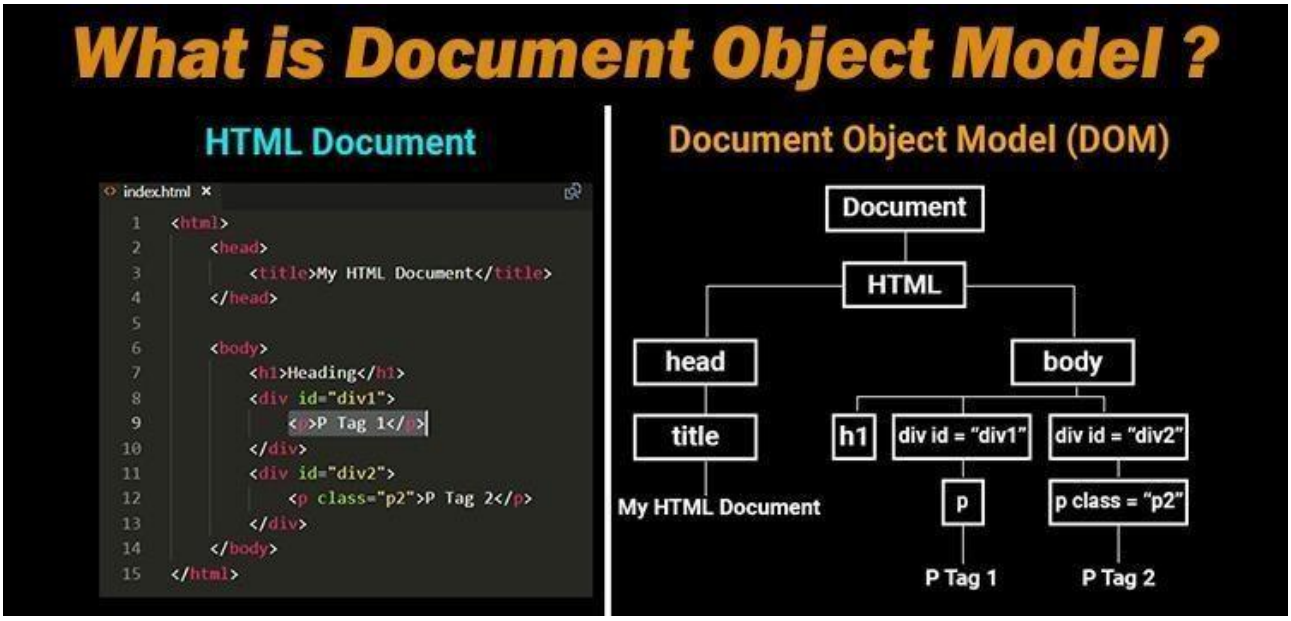
11. onload Event : When the page has been loaded

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction()
{
alert("Page is loaded");
}
</script>
</head>
<body onload="myFunction()">
<h1>Hello World!</h1>
</body>
</html>
```

12. onunload Event : When the browser closes the document

```
<!DOCTYPE html>
<html>
    <head>
        <script>
function myFunction() {
alert("Thank you for visiting SS!");
}
</script>
</head>
<body onunload="myFunction()">
    <h1>Welcome to my Home Page</h1>
    <p>Close this window or press F5 to reload the page.</p>
</body></html>
```

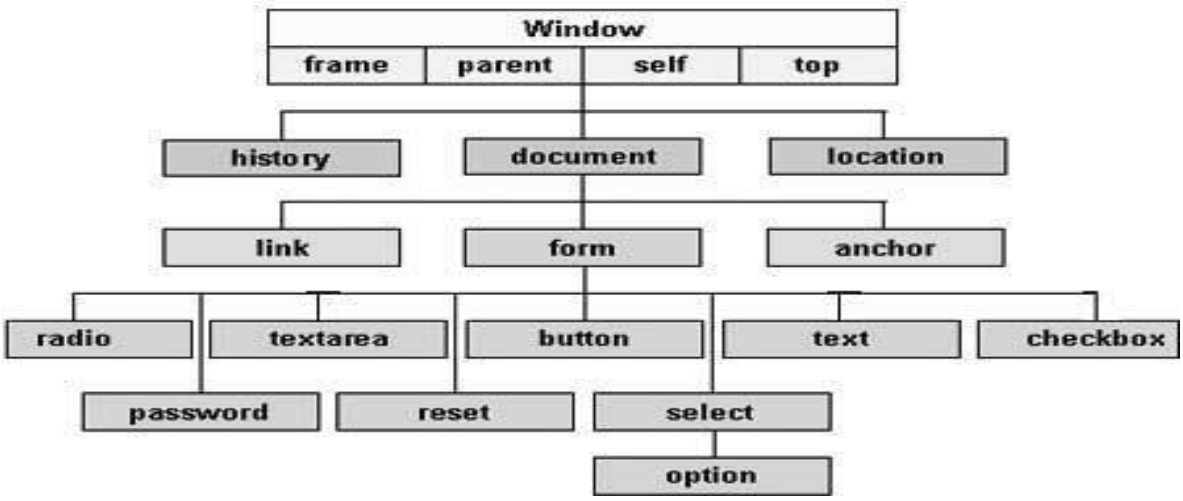
➢ EXPLAIN DOCUMENT OBJECT



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

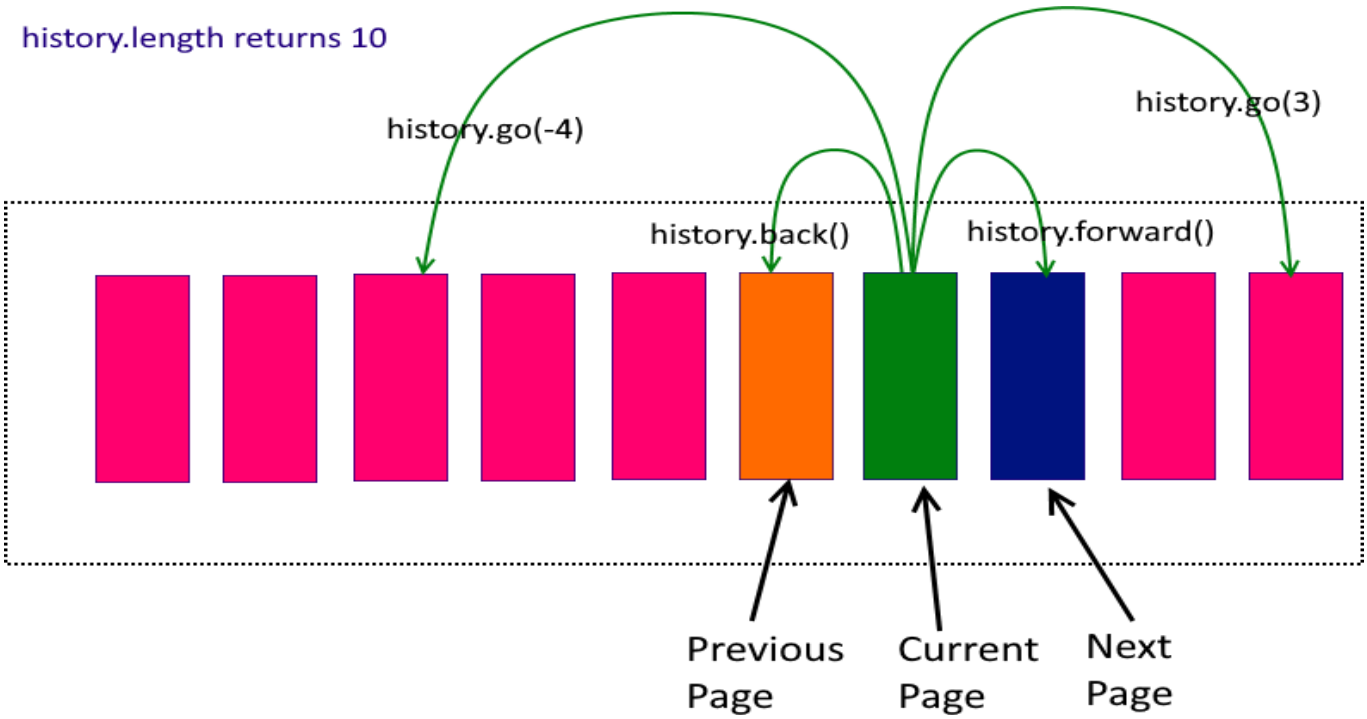
- A Document object represents the HTML document that is displayed in that window. The Document object has various properties that refer to otherobjects which allow access to andmodification of document content.
- The way that document content is accessed and modified is called the **Document Object 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 thatobject such as text fields, buttons, radio buttons, and checkboxes.

- Here is a simple hierarchy of few important objects:



1 Word Question – Answer

SR.NO.	QUESTION	ANSWER
1	FULL FORM OF DOM_____	DOCUMENT OBJECT MODEL
2	LIST OUT ALL OBJECTS OF DOM OBJECT	1. WINDOW 2. DOCUMENT 3. FORM 4. FORM CONTROL
3	WHAT IS THE USE OF DOM OBJECT?	A DOCUMENT OBJECT REPRESENTS THE HTML DOCUMENT THAT IS DISPLAYED IN THAT WINDOW



➢ EXPLAIN HISTORY OBJECT

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 thehistory 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 aspressing 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 presentin 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 gomethod 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 ofhistory object. The forward method seen above is the same as history.go(1)

1 MARKS Question – Answer

SR.NO.	QUESTION	ANSWER
1	WHAT IS HISTORY OBJECT?	HISTORY OBJECT USED TO WORK WITH BROWSER HISTORY
2	LIST OUT PROPERTY OF HISTORY OBJECT	1. LENGTH 2. CURRENT 3. NEXT 4. PREVIOUS
3	LIST OUT METHODS OF HISTORY OBJECT	1. BACK 2. FORWARD 3. GO

➢ EXPLAIN NAVIGATOR OBJECT.

- The Navigator object of JavaScript returns useful information about the visitor'sbrowser 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 Firefoxonly.**
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 EXPLAIN EMAIL VALIDATION & FORM VALIDATION

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.

- **Example of valid email id**

  - my.ownsite@ourearth.org
  - mysite@you.me.net

▪ Example of invalid email id

- mysite.ourearth.com [@ is not present]
- mysite@.com.my [tld (Top Level domain) can not start with dot "."]
- @you.me.net [No character before @]
- mysite123@gmail.b [".b" is not a valid tld]
- mysite@.org.org [tld can not start with dot "."]
- .mysite@mysite.org [an email should not be start with "."]
- mysite()*@gmail.com [here the regular expression only allows character, digit, underscore and dash]mysite..1234@yahoo.com [double dots are not allowed]

JavaScript code to validate an email id

```
function ValidateEmail(mail)
{
if(/^\w+([\.-]?\w+)*@\w+([\.-
]?\w+)*(\.\w{2,3})+$/.test(myForm.emailAddr.value))
{
```

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 will be 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 andpassword 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.

```
<script>
        function validateform()
        {
var name=document.myform.name.value;
    var password=document.myform.password.value;
                if (name==null || name=="")
                {
                        alert("Name can't be
                        blank");return false;
                }else if(password.length<6)
                {
        alert("Password must be at least 6 characters long.");
        return false;
                }
        }
        </script>
    <body>
        <form name="myform" method="post" action="abc.jsp"
        onsubmit="return validateform()" >
                Name: <input type="text" name="name"><br/>
                Password: <input type="password" name="password"><br/>
                <input type="submit" value="register">
</form>
```

JavaScript Number Validation

- Let's validate the textfield for numeric value only. Here, we are using isNaN() function.

```
<script>
function validate()
{
    var num=document.myform.num.value; if (isNaN(num))
    {
    if (isNaN(num))
    {
    document.getElementById("numloc").innerHTML="Enter Numeric value only";
    return false;
    }
    else
    {
    return true;
    }
  }
}
</script>
<form name="myform" onsubmit="return validate()" >
    Number: <input type="text" name="num"><span id="numloc"></span><br/>
    <input type="submit" value="submit">

</form>
```

JavaScript Retype Password Validation

```
<script type="text/javascript">function matchpass()

    {
  var firstpassword=document.f1.password.value;
  var
                                        secondpassword=doc
  ument.f1.password2.value;
  if(firstpassword==secondpassword)
  {}
  else
  {}
}
  return true;


  alert("password
  must be same!");
  return false;


  </script>
  <form name="f1" action="register.jsp"
  onsubmit="return matchpass()">
  Password:<input type="password"
  name="password" /><br/>
  Re-enter Password:<input type="password"
  name="password2"/><br/>
  <input type="submit">

  </form>
```

Shree h. N. Shukla College of i.t. & mgmt.
(Affiliated To Saurashtra University)

ES6 Overview & Basics REACT JS

ES6 Overview:

ECMAScript 2015, commonly known as ES6, brought significant enhancements to JavaScript, making the language more expressive and powerful. Some key features of ES6 include:

1)Let and Const Declarations:

- let allows you to declare block-scoped variables.
- const declares constants with block scope.

❖ **javascript**

```
let variableName = "some value";

const PI = 3.14159;
```

2)Arrow Functions:

- A concise way to write functions.

❖ javascript

```
const add = (a, b) => a + b;
```

3)Template Literals:

- A more readable way to concatenate strings.

❖ **javascript**

```
const name = "John";

const greeting = `Hello, ${name}!`;
```

4)Destructuring Assignment:

- Extract values from objects or arrays into distinct variables.

❖ **javascript**

```
const person = { name: "Alice", age: 30 };

const { name, age } = person;
```

5)Classes:

- **A more straightforward way to create constructor functions and prototype-based inheritance.**

❖ javascript

```
class Animal
{
  constructor(name)
{
  this.name = name;
 }

 speak() {
  console.log(`${this.name} makes a sound.`);
 }
}
```

6)Modules:

- Import and export functionality between different JavaScript files.

❖ **javascript**

```
// export.js

export const PI = 3.14159;

// import.js

import { PI } from './export';
```

Shree h. N. Shukla College of i.t. & mgmt.
(Affiliated To Saurashtra University)

React.js Basics:

React.js is a JavaScript library for building user interfaces, developed by Facebook. Here are some fundamental concepts:

1)Components:

- Building blocks of a React application.
- Can be functional or class-based.

❖ **Javascript**

```
// Functional Component
const MyComponent = () => {
  return <div>Hello, World!</div>;
};

// Class Component
class MyClassComponent extends React.Component {
  render() {
    return <div>Hello, World!</div>;
  }
}
```

2)JSX (JavaScript XML):

- Syntax extension for JavaScript, used with React to describe what the UI should look like.

❖ **javascript**

```
const element = <h1>Hello, JSX!</h1>;
```

3)Props:

- Data passed from a parent component to a child component.

❖ **javascript**

```
const Greeting = (props) => {
```

```
 return <div>Hello, {props.name}!</div>;
};
```

4)State:

- Internal data storage for a component.
- Changes to state trigger a re-render of the component.

❖ **javascript**

```
class Counter extends React.Component {
  constructor() {
        super();
    this.state = { count: 0 };
  }
  render() {
    return <div>Count: {this.state.count}</div>;
  }
}
```

5)Lifecycle Methods:

•Methods that are called at different points in the life cycle of a component.

❖ **javascript**

```
class MyComponent extends React.Component { componentDidMount() {
// Called after the component is rendered to the DOM
}
componentWillUnmount() {
 // Called just before the component is removed from the DOM
 } }
```

6)Handling Events:

- React uses camelCase for event names.

❖**javascript**

```
const Button = () => {
  const handleClick = () => {
    console.log("Button clicked!");
  };
  return <button onClick={handleClick}>Click Me</button>;
};
```

These are just the basics of ES6 and React.js. Both ES6 and React have many more features and concepts that you can explore as you become more familiar with them.

Shree h. N. Shukla College of i.t. & mgmt.
(Affiliated To Saurashtra University)

ES6 Classes, functions & Promises REACTJS

ES6 Classes

ES6 introduced a more convenient syntax for creating classes in JavaScript, providing a cleaner and more structured way to implement object-oriented programming. In React, classes are often used to define components. Here's a basic example:

- **javascript**

```
// Class Component
class MyComponent extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      message: "Hello, React!"
    };
  }
  render() {
    return <div>{this.state.message}</div>;
  }
}
```

In this example, MyComponent is a class component that extends React.Component. The constructor method is where you initialize the component's state, and the render method returns the JSX to be rendered.

Functions:

In ES6, arrow functions provide a concise syntax for writing functions, especially when used in the context of React. Here's a simple example:

- **javascript**

```
// Functional Component using Arrow Function
const FunctionalComponent = (props) => {
  return <div>{props.message}</div>;
};
```

Arrow functions automatically bind this, which can be beneficial when dealing with event handlers or callbacks within a React component.

Promises:

Promises are a pattern for handling asynchronous operations in a more readable and manageable way. In React, Promises are often used when dealing with data fetching or other asynchronous tasks. Here's a basic example:

❖ **javascript**

```
// Asynchronous Data Fetching using Promises
const fetchData = () => {
  return new Promise((resolve, reject) => {
    // Simulating a network request
    setTimeout(() => {
      const data = { message: "Data fetched successfully!" };
      resolve(data);
    }, 2000);
  });
};
// Using the Promise in a React component
class DataComponent extends React.Component {
  componentDidMount() {
    fetchData()
      .then((data) => {
        console.log(data.message);
        // Update state or perform other actions with the data  });
      .catch((error) => {
        console.error("Error fetching data:", error);
      }); }
  render() {
    return <div>Loading data...</div>;
  }}
```

In this example, the fetchData function returns a Promise, and the DataComponent class component uses this Promise to handle asynchronous data fetching within the componentDidMount lifecycle method.

These ES6 features contribute to cleaner and more maintainable code in React applications, making it easier to manage state, handle events, and deal with asynchronous

operations.

Express JS

Setting up an app with ExpressJS

Setting up a full-stack web application with Express.js for the backend and React.js for the frontend involves a few steps. Below is a basic guide on how you can set up a simple project. Ensure that you have Node.js and npm (Node Package Manager) installed on your machine.

1. Create the Express.js Backend:

Step 1: Initialize a new Node.js project
Open your terminal and run the following commands:

```bash
mkdir your-project-name
cd your-project-name
npm init -y
```

Step 2: Install Express.js

```bash
npm install express
```

Step 3: Create a basic Express.js server
Create a file named server.js:

```javascript
const express = require('express');
const app = express();
const port = 5000;

app.get('/', (req, res) => {
  res.send('Hello from Express!');
});

app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

Step 4: Start the Express.js server

Run the following command in your terminal

```bash
node server.js
```

This will start your Express.js server at http://localhost:5000.

2. Create the React.js Frontend:

Step 1: Initialize a new React.js project

Open a new terminal and navigate to the root folder of your project:

```bash
cd your-project-name
npx create-react-app client
```

Step 2: Start the React development server

Navigate to the client folder:

```bash
cd client
npm start
```

This will start your React development server at http://localhost:3000.

3. Connect Frontend and Backend:

By default, the React development server runs on http://localhost:3000, and the Express server runs on http://localhost:5000. To connect them, you can use a proxy.

In the client folder, add a package.json file if it doesn't exist, and add the following line:

```json
"proxy": "http://localhost:5000",
```

Now, when you make API requests from your React components, they will be automatically proxied to the Express server.

4. Test the Setup:

Update your client/src/App.js to make a request to your Express server:

```javascript
import React, { useEffect, useState } from 'react';
import './App.css';

function App() {
  const [message, setMessage] = useState('');

  useEffect(() => {
    fetch('/api')
      .then((res) => res.json())
      .then((data) => setMessage(data.message))
      .catch((error) => console.error('Error fetching data:', error));
  }, []);

  return (
    <div className="App">
      <h1>{message}</h1>
    </div>
  );
}

export default App;
```

Now, when you run both the Express.js server and the React.js development server, you should see "Hello from Express!" displayed on your React app.

This is a basic setup. Depending on your project requirements, you might need additional configurations, middleware, and folder structures. For production, you may consider setting up a build process and deploying your application to a hosting service.

Express.js Routing

Express.js provides a simple and effective way to define routes for your server. You can use the express.Router to organize your routes into modular components. Here's an example:

1)Create a routes folder:

Create a new folder named routes in your project's root directory.

2)Define routes in Express:

Inside the routes folder, create a file named api.js:

javascript

```javascript
// routes/api.js
const express = require('express');
const router = express.Router();

router.get('/api', (req, res) => {
  res.json({ message: 'Hello from the Express API!' });
});

// Add more routes as needed

module.exports = router;
```

3)Use the routes in your Express app:

In your main server.js file, import and use the router:

```javascript
const express = require('express');
const app = express();
const port = 5000;

const apiRoutes = require('./routes/api');
app.use('/api', apiRoutes);

app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

Now, your Express app will respond to requests at http://localhost:5000/api.

- **React.js Routing:**

React.js uses the ‘react-router-do’ library to handle client-side routing. Here's a basic setup:

1)Install ‘react-router-dom’:
In your ‘client’ directory, install the ‘react-router-dom’ library:

```bash
cd client
npm install react-router-dom
```

2)Define routes in React:
Update your client/src/App.js to use react-router-dom:

```javascript
import React from 'react';
import { BrowserRouter as Router, Route, Link } from 'react-router-dom
import Home from './components/Home';
import About from './components/About';

function App() {
  return (
    <Router>
      <div>
        <nav>
          <ul>
            <li>
              <Link to="/">Home</Link>
            </li>
            <li>
              <Link to="/about">About</Link>
            </li>
          </ul>
        </nav>

        <hr />

        <Route exact path="/" component={Home} />
        <Route path="/about" component={About} />
      </div>
    </Router>
  );
}

export default App;
```

3)Create components for each route:

In the **client/src/components** folder, create **Home.js** and **About.js** components:

javascript

```javascript
// components/Home.js
import React from 'react';

function Home() {
  return <h2>Home</h2>;
}

export default Home;
```

javascript

```javascript
// components/About.js
import React from 'react';

function About() {
  return <h2>About</h2>;
}

export default About;
```

Now, when you run your application, you can navigate between the Home and About pages on the client side. The URLs will be handled by the React Router.

Remember to customize the routes and components based on your application's requirements. Additionally, for more complex applications, you might want to explore nested routes, route parameters, and other features provided by **react-router-dom**.

Shree h. N. Shukla College of i.t. & mgmt.
(Affiliated To Saurashtra University)

Connecting views with templates

In React.js, the concept of "templates" is typically referred to as JSX (JavaScript XML). JSX is a syntax extension for JavaScript that looks similar to XML or HTML but allows you to write React components in a more declarative way. JSX code is then transpiled to JavaScript before being executed in the browser.

Here's a basic example of connecting views with JSX templates in React.js:

Assuming you have a React component in a file named **MyComponent.js**:

```jsx
// MyComponent.js
import React from 'react';

const MyComponent = () => {
  return (
    <div>
      <h1>Hello, React!</h1>
      <p>This is a simple React component.</p>
    </div>
  );
};

export default MyComponent;
```

In this example, the **MyComponent** function returns JSX, which represents the structure of the component. You can include HTML-like tags and create a hierarchy of components. JSX is expressive and makes it easy to visualize the UI structure.

To use this component in another file (e.g., **App.js**), you would import it and include it in the JSX code:

```jsx
// App.js
import React from 'react';
import MyComponent from './MyComponent';

const App = () => {
  return (
    <div>
      <h1>Welcome to My React App</h1>
      <MyComponent />
    </div>
  );
};

export default App;
```

In this example, **MyComponent** is imported and used within the **App** component's JSX. This is a common pattern in React applications where you compose your UI by combining smaller, reusable components.

Remember to use proper file paths when importing components. Also, be aware that React components should start with an uppercase letter to distinguish them from regular HTML elements.

This approach allows you to create modular and maintainable code by breaking down your UI into smaller, reusable components. Each component can have its own JSX template and logic, making it easier to manage and update your application.

Shree h. N. Shukla College of i.t. & mgmt.
(Affiliated To Saurashtra University)

configurations and error handling.

Configurations and error handling are crucial aspects of building robust and maintainable React applications. Here's a guide on some common configurations and best practices for error handling in React.js.

Configurations:

1. **Babel Configurations:**

Babel is used to transpile JSX and ES6+ JavaScript code into browser-compatible code. You typically have a **.babelrc** file in your project for Babel configurations:

```json
{
  "presets": ["@babel/preset-env", "@babel/preset-react"],
  "plugins": ["@babel/plugin-proposal-class-properties"]
}
```

2. ESLint Configurations:

ESLint is a static code analysis tool used to find and fix problems in your JavaScript code. It helps enforce coding standards and identify potential issues. A common ESLint configuration might look like this:

```
Json

{
  "extends": ["eslint:recommended", "plugin:react/recommended"],
  "plugins": ["react"],
  "rules": {
    "react/prop-types": 0, // Disable prop-types rule for functional components
    "no-console": "warn"
  },
  "parserOptions": {
    "ecmaVersion": 2020,
    "sourceType": "module",
    "ecmaFeatures": {
      "jsx": true
    }
```

3. Webpack Configurations:
Webpack is a module bundler used to bundle JavaScript files and other assets. Here's a simplified example of a webpack.config.js file:

```javascript
const path = require('path');

module.exports = {
  entry: './src/index.js',
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist')
  },
  module: {
    rules: [
      {
        test: /\.(js|jsx)$/,
        exclude: /node_modules/,
        use: {
          loader: 'babel-loader'
        }
      }
    ]
  }
};
```

Shree h. N. Shukla College of i.t. & mgmt.
(Affiliated To Saurashtra University)

Error Handling:

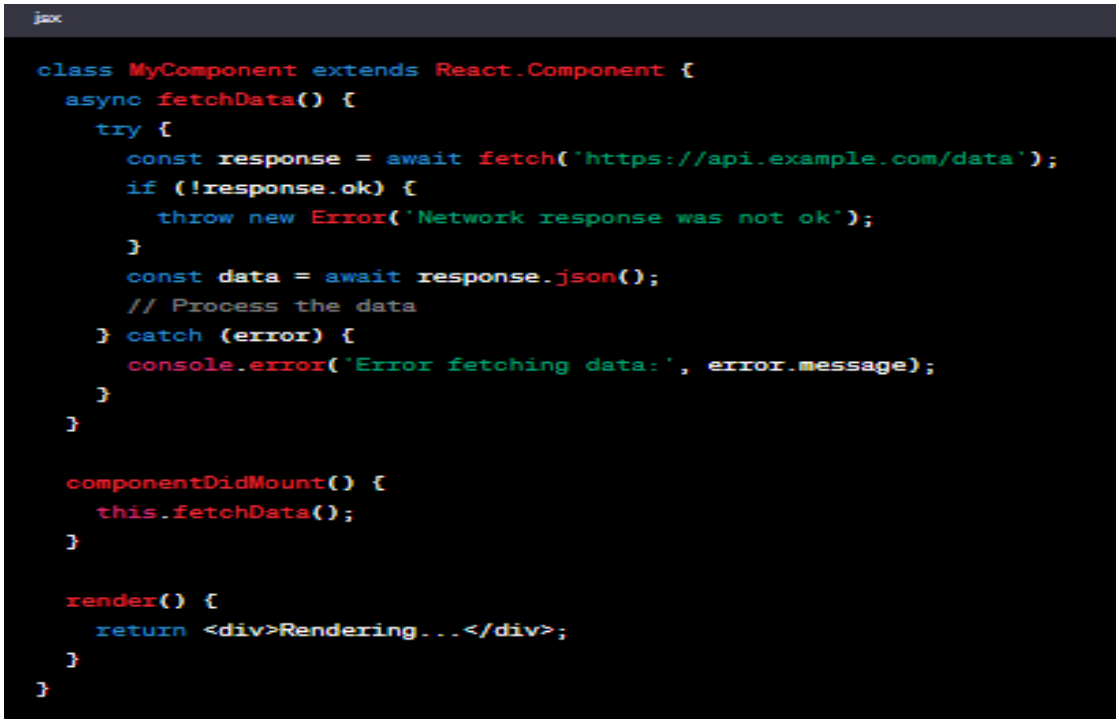
1. Error Boundaries:

React provides a concept called "Error Boundaries" to handle JavaScript errors anywhere in a component tree. You can define an error boundary component using the componentDidCatch lifecycle method:

```jsx
class ErrorBoundary extends React.Component {
  constructor(props) {
    super(props);
    this.state = { hasError: false };
  }

  componentDidCatch(error, errorInfo) {
    this.setState({ hasError: true });
    // Log the error to an error reporting service
    console.error(error, errorInfo);
  }

  render() {
    if (this.state.hasError) {
      return <div>Something went wrong.</div>;
    }

    return this.props.children;
  }
}

// Usage
<ErrorBoundary>
  <YourComponent />
</ErrorBoundary>
```

2. Handling HTTP Errors:

When making asynchronous requests in React, handle HTTP errors gracefully using promises or the async/await syntax. For example:



```jsx
class MyComponent extends React.Component {
  async fetchData() {
    try {
      const response = await fetch('https://api.example.com/data');
      if (!response.ok) {
        throw new Error('Network response was not ok');
      }
      const data = await response.json();
      // Process the data
    } catch (error) {
      console.error('Error fetching data:', error.message);
    }
  }

  componentDidMount() {
    this.fetchData();
  }

  render() {
    return <div>Rendering...</div>;
  }
}
```

These are just some basic configurations and error handling practices. Depending on your project setup and requirements, you may need additional tools and practices, such as setting up a testing framework (e.g., Jest), configuring a state management library (e.g., Redux), or integrating with specific APIs and services for error reporting. Always tailor your approach to the specific needs and scale of your React application.