Java Networking

- Java Networking is a concept of connecting two or more computing devices together so that we can share resources.
- Java Socket programming provides facility to share data between different computing devices.

Advantages :→

1. Sharing Resources.
2. Centralize software management.

Java Networking Terminology :→

1. IP Address
2. Protocol
3. Port Number
4. Connection-oriented and Connection-less protocol.
5. Socket.

1) IP Address :→

IP Address is a unique number assigned to a node of a network e.g. 192.168.0.1.

- It is composed of octets that range from 0 to 255.
- It is logical address that can be changed.

2) Protocol :→

A protocol is a set of rules basically that is followed for communication.

Ex: TCP, FTP, Telnet, SMTP, POP etc.

3) Port Number :→

The port number is used to uniquely identify different applications. It acts as a communication endpoint between applications.

- The port number is associated with the IP address for communication between two applications.

4) Connection-less and Connection-oriented protocol :-

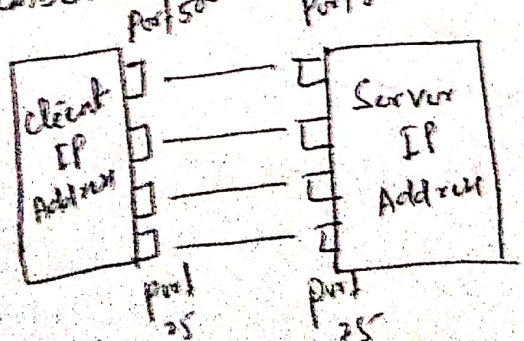
- In Connection-oriented protocol, acknowledgement is sent by the receiver.
- It is reliable but slow.
- Example : TCP

- In Connection-less protocol, acknowledgement is not sent by the receiver.
- It is not reliable but fast.
- Example : UDP.

5) Socket :→

A Socket is one endpoint of a two way communication link between two programs running on the network.

- An endpoint is a combination of an IP Address and a port number.



that Java Networking used to write programs execute across multiple devices (Computer present in remote locations/ same locations connected by a network).

- The java.net package provides many classes to deal with networking application in Java. It uses two protocols i.e, TCP (Transmission Control protocol and UDP (User Datagram Protocol).

Java URL :→

The Java URL class represents an URL. URL stands for Uniform Resource Locator. It points to a resource on the World wide web.

Example:

http://www.javapoint.com/java-tutorial

protocol — Host Name — File

URL Contains many information such as

1. Protocol: In this case, http is the protocol.
2. Server name or IP Address: In this case, www.javapoint.com is the server name.
3. Port Number: It is an optional attribute,
   ex: http://www.javapoint.com:80/abc/, '80' is port number. If port number is not mentioned in the URL, it returns -1.
4. File Name or directory name

Example :→

```
import java.net.*;
public class URLDemo{
    public static void main(String [] args)
    { try{
      URL u = new URL ("http://www.it.com:80/
                                    index.html");
      System.out.println("Protocol :"+ u.getProtocol());
      System.out.println("Host Name:"+ u.getHost());
      System.out.println("Port Number"+ u.getPort());
      System.out.println("File Name"+ u.getFile());
       }
      catch(Exception e)
       {
           System.out.println(e);
       }
    }
}
```

Note: There are certain information that we can parse from a URL. Here we will see how to parse the information from URL using URL class

O/P

protocol: http
Host Name: www.it.com
Port Number: 80
File Name: index.html.

Java URLConnection class :→

- The Java URLConnection class represents a communication link between the URL and the application.
- This class can be used to read and write data to the specified resource referred by the URL.

How to get the object of URLConnection class :→

The openConnection() method of URL class returns the object of URLConnection class.

Syntax :

```
public URLConnection openConnection() throws
                                    IOException{}
```

The URLConnection class provides many methods, we can display all the data of a webpage by using the getInputStream() method.

- The getInputStream() method returns all the data of the specified URL in the stream that can be read and displayed.

Example :→

```
import java.io.*;
import java.net.*;
public class URLConnectionDemo{
    public static void main(String[] args)
    {
        try{
            URL u = new URL("http://www.abc.com/a.txt");
            URLConnection ucon = u.openConnection();
            InputStream st = ucon.getInputStream();
            int i;
            while((i = st.read()) != -1)
            {
                System.out.println((char)i);
            }
        } catch(Exception e)
            { System.out.println(e); }
    }
}
```

Present

→ we required specified URL.

→ Connection Established

st will Contain the stream of a.txt file.

Java InetAddress class: →

Java InetAddress class represents an IP address.

- The java.net.InetAddress class provides methods to get the IP of any host name.

  ex: www.google.com, www.facebook.com etc.

- An IP address is represented by 32-bit or 128-bit unsigned number.
- An instance of InetAddress represents the IP address with its corresponding host name.

Commonly used methods of InetAddress class: →

method	Description
① public static InetAddress getByName(String host) throws UnknownHostException	- It returns the instance of InetAddress containing LocalHost IP and name.
② public static InetAddress getLocalHost() throws UnknownHostException	- It returns the instance of InetAddress Containing local host name and address.
③ public String getHostName()	- It returns the host name of the IP address.
④ public String getHostAddress()	- It returns the IP address in string format.

Example

```
import java.io.*;
import java.net.*;
public class InetExample{
    public static void main(String [] args)
    {
        try{
    InetAddress ip= InetAddress.getByName("www.
                                    ipec.org");
    System.out.println("Host Name"+ip.getHostName());
    System.out.println("IP Address" + ip.getHostAddress());
        }
        Catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

O/P

Host Name : www.ipec.org

IP Address : 64.207.145.203

Socket Programming :→

Socket programming in java is used for communication between the applications that are running on different JRE. It can be either Connection-oriented or Connectionless.

- A Socket programming is a way of Connecting two nodes on a network to communicate with each other.
- A Socket is a way to establish a Connection between a client and a Server.
- A Socket is simply an endpoint for communication between the machines. The Socket class can be used to create a Socket..

Important Methods of Socket class :→

	Method	Description
1)	public InputStream getInputStream()	- returns the InputStream attached with this socket.
2)	public OutputStream getOutputStream()	- returns the OutputStream attached with this socket.
3)	public synchronized void close()	- closes this Socket.

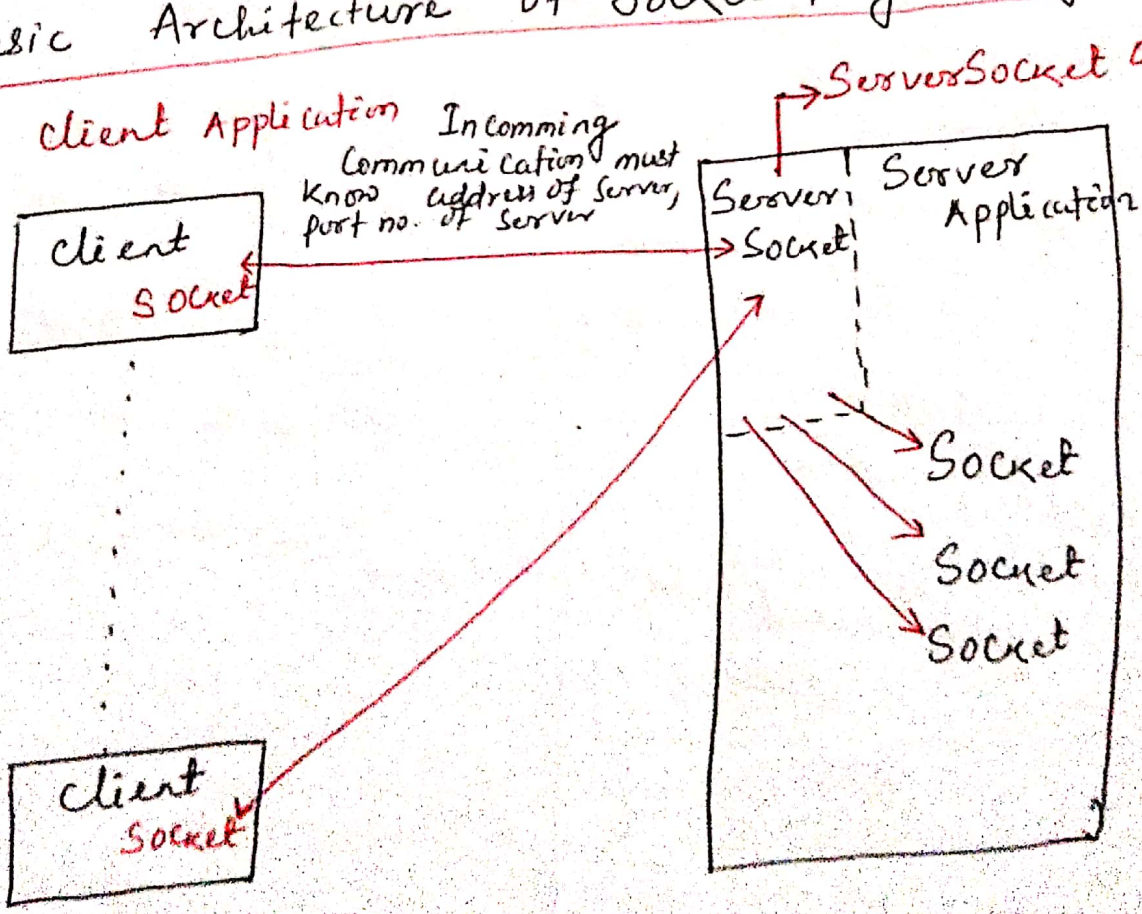
- The ServerSocket class can be used to create a server socket. This object is used to establish communication with the client.

Important methods of ServerSocket :→

method	Description
1) public Socket accept()	- returns the socket and establish a connection between server and client.
2) public synchronized void close()	- close the ServerSocket.

Socket and ServerSocket classes are used for Connection-oriented socket programming.

Basic Architecture of Socket Programming :→



Basic Steps in Establishing a TCP Connection :→

① Server instantiate ServerSocket object [port no]

② Server invokes → accept() method of ServerSocket.

③ Client → Socket object (Server name ex: localhost, Port no ex: 6666)

④ Attempt by the client to connect.

A Simple Server Program Steps

1. Open the Server Socket.

   ServerSocket s = new ServerSocket (PORT);

2. Wait for client Request

   Socket c = s.accept();

3. Create I/O stream for Communicating to the client

4. Perform Communication with client.

5. close socket.

A Simple client Program Steps

1. Create Socket object

   Socket c = new Socket (Server, port);

2. Create I/O stream for Communication with the Server.

3. Perform I/O or Communication with the server.

4. close socket.

Example

ServerExample.java

```
import java.io.*;
import java.net.*;
public class ServerExample{
public static void main(String
                    [] args)
{
  try{
ServerSocket ss = new
          ServerSocket(6666);
Socket s = ss.accept();
↳ establish connection
DataInputStream dis = new
    DataInputStream(s.getInput
                    Stream());
String str = (String) dis.
                    readUTF();
System.out.println("message"
                + str);
  ss.close();
}
Catch (Exception e)
  {  e.printStackTrace();
  }
}
}
```

O/P

Hello

ClientExample.java

```
import java.io.*;
import java.net.*;
public class ClientExample{
public static void main(String
                    [] args){
try{
Socket s = new Socket("localhost",
                    6666);
DataOutputStream dout = new
    DataOutputStream(s.getOutput
                    Stream());
dout.writeUTF("Hello");
 dout.flush();
dout.close();
s.close();
  }
Catch (Exception e)
  {
     System.out.println(e);
  }
}
}
```

JavaScript

A script is a small program which is used with HTML to make web pages more attractive, dynamic, and interactive such as an alert, pop-up window on mouse click. Currently most popular scripting language which is JavaScript.

- Javascript is a client side scripting language.
- Javascript is an object-based scripting language which is lightweight and cross-platform.
- Javascript is interpreted programming language.
- Javascript was first known as livescript but Netscape changed its name to Javascript.

Features of Javascript :→

- All popular web browsers support Javascript as they provide built in execution environment.
- Follows syntax and structure of C programming language. Thus it is a structured programming language.
- It is a weakly typed language.
- Object-oriented programming language.
- Light weighted and Interpreted language.
- It is a case-sensitive language.
- It is cross-platform language.

- Javascript is a high level, untyped and interpreted programing language.

Applications of Javascript :→

1. Client-side validation
2. Dynamic drop-down menus.
3. Displaying date and time.
4. Displaying pop-up windows and dialog boxes.
5. Displaying clocks etc.
6. Communicating with server and read write HTML elements.

Why study Javascript :→

Javascript is one of the 3 languages all developers must learn :-

HTML - to define the content of web pages.

CSS - to specify the layout of web pages.

Java script - to program the behavior of web pages.

Java Script Example :→

—Javascript example is easy to code. Javascript provides 3 places to put the Javascript code - within body tag, within head tag and external Javascript file.

```
<script type="text/javascript">
    document.write("Java Script is a object based
        programming language");
</script>
```

<script> - tag specifies that we are using Javascript
text/javascript - is the content type that provides information to the browser about the data.

document.write() - is used to display dynamic content through Javascript.

places to put

Placing the Javascript Code :→

There are two ways to place the Java script code:

(1) Embedded/Inline JavaScript :- Java script code can be placed either in the HEAD or in the BODY section of a HTML document.

Java script Example - code between the body tag :-

```
<html>
<body>
<script type="text/javascript">
        alert("Hello Javascript!");
</script>
</body>
</html>
```

Java script Example - code between the head tag -

```
<html>
<head>
<script type="text/javascript">
    function msg()
    { alert("Hello javascript!");
    }
</script>
</head>
```

```
<body>
<form>
    <input type="button" value="Click"
                onclick="msg()"/>
</form>
</body>
</html>
```

(2) External Javascript - It provides code-reusability because single javascript file can be used in several html pages. Javascript code in external files having extensions as ".js". To use external Javascript file we will use "src" attribute in <script> tag to indicate the link for the source Javascript file.

Example

test.html

```
<html>
<head>
<script type="text/
            javascript
    src="message.js">
</script>
</head>
<body> <form>
<input type="button value="Click" onclick="msg()"/>
</form></body></html>
```

message.js

```
function msg()
{
    alert("Hello Javascript!");
}
```

Statements in Javascript :→

Statements are the commands or instructions given to the Javascript interpreter to take some actions as directed.

ex

alert ('Hello, world!') → shows the message

- Any no. of statements can be there in the code.
- Statements can be separated with a semicolon.

alert ('Hello'); alert ('word');

- Semicolon may be omitted when a line break exists.

Javascript statements are composed of: values, operators, Expressions, keywords and comments.

Comments -

- Single line Comment — //
- Multiple lines comment - /* — */

Literals :— Literals refer to the constant values, which are used directly in JavaScript code.

Ex:

```
a = 10;
b = 5.7;
document.write ("Hello");
```

literals

Identifiers :— Identifiers refer to the name of variables, functions, arrays etc.

→ Identifier

ex: var RollNo;

Constants :— To declare a constant (unchanging) variables, 'Const' keyword is used.

```
Const a = 10;
a = 20; Can't be reassigned.
```

Datatypes :—

There are two types of Data types in Java script

① primitive
② Non-primitive.

Primitive Data Type

① String : represents sequence of characters.

```
var country = "India";
```

② Number : represents numeric values

```
a = 10
```

③ Boolean : represent boolean value either false or true.

④ Undefined : represents undefine value. → `var x; document.write(x);` o/p undefined

⑤ Null : represent null i.e, no value at all.

```
var distance = new object();
distance = null;
```

Non-primitive Datatype

① Object : represents instance through which we can access members:

```
ObjectName.propertyName.
```

② Array : represents group of similar values.

```
var a = new a()
```

③ RegExp : represents regular Expression.

Javascript Variables :-

Variable is a name of storage location 'var' keyword is used to create variables in Javascript.

Syntax :-

```
Var   variablename = value;
```

ex

```
Var   x = 10
Var _name = "javascript";
```

Valid variable names	Invalid variable names
f_name	10_number
total123	rate-%
_sumof	my name

There are two types of variables :-

1. Local variable - is declared inside block or function.
2. Global Variable - is declared out side the function.

Note :- To declare Javascript global variables inside function, window object is used.

```
window.value = 10;
```

Ex

```
function msg()
{   window.value = 10; //declaring global variable.
}
function access()
{   alert(window.value); // accessing global variables.
```

Tools needed for writing and Running Javascript Code

Following tools are needed for working with Javascript code.

(a) Text Editors: We can choose any text editor or word processor (i.e, Notepad, Notepad++, Wordpad etc

(b) Browser: Browser interprets Javascript code and shows the output on browser's document window.

Javascript Operators :→

Javascript operators are symbols that are used to perform operations on operands.

```
Var sum = 10+20;
```

operator

Types of operator

1. Arithmetic Operator
2. Comparision (Relational) operator
3. Bitwise operator
4. Logical Operator
5. Assignment operator
6. Special operator.

Arithmetic Operators :→

Operator	Description	Example
+	Addition	10+20 =30
−	Subtraction	20−10 =10
*	Multiplication	10*20 =200
/	Division	20/10 =2
%	Modulus (Remainder)	20%10 =0
++	Increment	var a=10; a++ a=11
−−	Decrement	Var a=10; a−−, Now a=9
**	Exponentiation	var a=2; var b=5; a**b ; ⇒32

Note '+' ⇒ this can be used as Concatenation operator
Var x = "Hello" + " " + "Student" ⇒ Hello Student

Javascript Comparision Operators :→

Operator	Description	Example
==	Is equal to	var x="2"] x==y ⇒true var y=2;]
===	Identical(equal and of same type)	var x="2";] x===y ⇒ false var y=2;]
!=	Not equal to	10!=20 ⇒ true
!==	Not Identical	20!==20 ⇒ false
>	Greater than	20>10 ⇒ true
>=	Greater than equal to	20>=10 ⇒ true
<	Less than	20<10 ⇒ false
<=	Less than or equal to	20<=10 ⇒ false.

Javascript Bitwise operator :→

The bitwise operators perform bitwise operations on operands.

Operator	Description	Example
&	Bitwise AND	(10==20 & 20==33) ⇒ false
|	Bitwise OR	(10==20 | 20==33) ⇒ false
^	Bitwise XOR	(10==20 ^ 20==33) ⇒ false
~	Bitwise NOT	(~10) ⇒ -10
<<	Bitwise Left shift	(10<<2) ⇒ 40
>>	Bitwise Right shift	(10>>2) ⇒ 2

Javascript Logical operators :→

Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) ⇒ false
||	Logical OR	(10==20) || (20==33) ⇒ false
!	Logical NOT	!(10==20) ⇒ true

Java script Assignment operator :→

Operator	Description	Example
=	Assign	Var x=20;
+=	Add assign	Var a=10; a+=20; ⇒ a=30
-=	Subtract assign	Var a=10; a-=5; ⇒ a=5
*=	Multiply assign	var a=10, a*=2; ⇒ a=20
/=, %=	Divide assign, modulus assign	Var a=10; a/=2; a⇒5 Var a=10; a-%=2; a=0.

Javascript Special operator :→

Operator	Description
(?:)	Conditional operator returns value based on the Condition. (a>b)? 10 : 20;
,	Comma operator a=10, b=20
delete	Delete operator deletes a property from the object
in	In operator checks if object has the given property.
instance of	checks if the object is an instance of given type
new	creates an instance (object).
typeof	Checks the type of object
void	return value.

Javascript if-else :→

The javascript if-else statement is used to execute the code whether condition is true or false.

(1) If statement
(2) If else statement
(3) if else if statement.

Javascript if Statement :→

```
if (expression)
{
    // statements to be evaluated
}
```

ex:

```
<html><head>
<script>
var a = 20;
if (a>20){
  document.write("value of a is greator than 10");
}
</script></head></html>
```

Javascript if-else statement :→

```
if(expression)
{
    // statement to be evaluated
}
else{
    // statement to be evaluated
}
```

ex:-

```
<script>
var a = 20;
if((a%2)==0)
{   document.write("a is even");
}
else{   document.write("a is odd");
}
</script>
```

Javascript If ... else if Statement :→ (else if ladder)

```
if(expression)
{
    // statement to be evaluated
}
else if(expression2)
{
    // statement to be evaluated
}
else if(expression3)
{
    // statement to be evaluated
}
else
{
    // statement to be evaluated.
}
```

ex

```
<script>
var a=20;
if (a==10)
  {  document.write("a is equal to 10");
  }
else if(a==15)
  {  document.write("a is equal to 15");
  }
else if(a==20)
  {  document.write("a is equal to 20");
  }
else { document.write("a is not equal to 10 15 or 20"); } </script>
```

Note : Javascript offers almost the same control statements as we have in C language.

Javascript switch :→

```
switch(expression)
{
    Case value1:
            statements;
            break;
    Case value2:
            statements;
            break;
    ---------
    ---------
    default:
            statements;
}
```

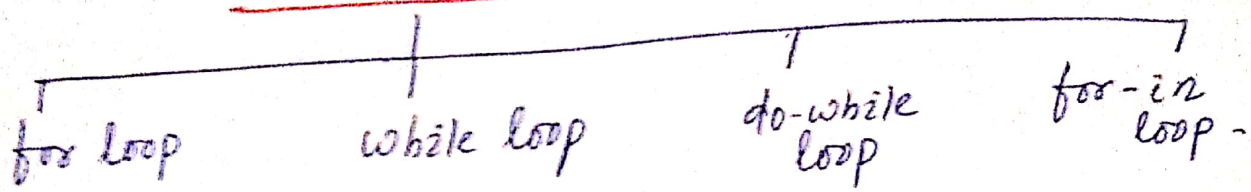
Note :- The switch statement is fall-through i.e, all the cases will be evaluated if you don't use break statement.

```
<script>
var grade = 'B';
var result;
switch(grade){
        Case 'A':
            result = "A Grade";
            break;
        Case 'B':
            result = "B Grade";
            break;
        Case 'C':
            result = "C Grade";
            break;
        default:
            result = "No Grade";
}
document.write(result);
</script>
```

Javascript Loops :→

Javascript loops are used to iterate the piece of code loop.

Types of Loops



1) Javascript for loop :→

```
for(initialization; condition; increment/decrement)
{
   //statements
}
```

ex

```
<script>
var i;
for(i=1; i<=5; i++)
{
   document.write(i + "<br/>");
}
</script>
```

O/P

1
2
3
4
5

2) Javascript while loop :→

```
while(condition)
{
   // statements
}
```

ex!

```
<script>
var i;
  i=1;
while(i<=5)
{
document.write(i + "<br/>");
  i++;
}
</script>
```

O/P

1
2
3
4
5

3) Javascript do-while loop :→

```
do{
   //statements;
}while(condition);
```

ex:-

```
<script>
var i=1;
do
{
   document.write(i + "<br/>");
   i++;
}while(i<=5);
```

o/p

1
2
3
4
5

4) Javascript for-in loop :→

It is used to iterate the properties of an object. Loop through objects properties.

- In each iteration one property from object is assigned to variable and the loop terminates when all properties evaluated.

Syntax

```
for(variable in object)
{
   //statements
}
```

ex!

```
<script>
var i;
for(i in navigator){
   document.write(i + "<br/>");
}
</script>
```

object

o/p

appCodeName
appName
appMinorVersion
platform
plugins
appVersion
systemLanguage
userAgent
appVersion
CookieEnabled
mimeTypes

Javascript Popup Boxes (Dialog Boxes) :→

Javascript, popup boxes are used to display the message or notification to the user. In Javascript, three kinds of popup boxes - Alert box, Confirm box, and Prompt box can be created using three methods of window objects.

- Alert Box
- Confirm Box
- Prompt Box

Alert Box :→ It is used when a warning message is needed to be produced. When the alert box is displayed the user, the user needs to press OK and proceed.

Syntax

```
[window].alert("Text to be displayed on the popup box");
```

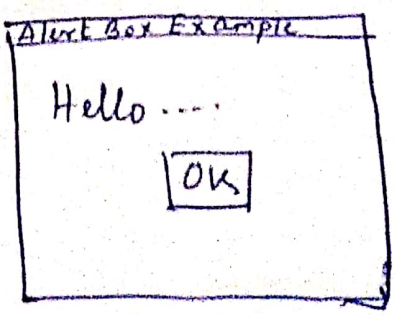
(window is optional)

Ex

```
window.alert("I am to alert you about...");
        OR
alert("I am to alert you about...");
```

Example Program :→

```
<html>
<head> <title> Alert Box Example </title>
</head>
<body>
  <script>
    alert("Hello....");
  </script>
</body> </html>
```



Confirm Box: → It is a type of pop up box which is used to get the authorization or permission from the user. The user has to press the "OK" or "Cancel" button to proceed. Confirm box is used if we want the user to verify and confirm the information.

Syntax

```
[window].confirm("Text to be confirmed");
```

↓ optional

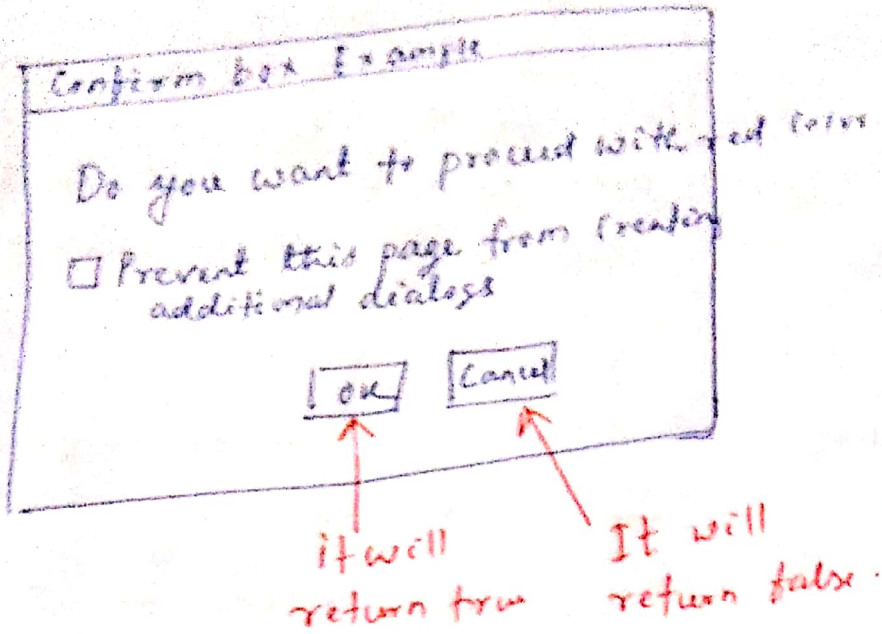
Note: If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

ex:

```
Confirm("Do you want to quit now?");
```

Example Program: →

```
<html>
<head><title> Confirm box Example </title>
</head>
<body>
<script>
    var x = Confirm("Do you want to proceed with red color");
                ↳ if true
    if(x)
    {
        document.getElementsByTagName("Body")[0].
                style.backgroundcolor = "red";
    }
</script>
<h1> Welcome to the page </h1>
</body> </html>
```



Prompt Box :→

Prompt box allows getting input from the user. We can specify the default text for the text field. The information submitted by the user from prompt() can be stored in a variable. It is a type of pop up box which is used to get the user input for further use. After entering the required details user have to click ok to proceed next stage else by processing the cancel button user returning null value.

Syntax prompt("Message", [, "default value in the text field"]);

↓

optional.

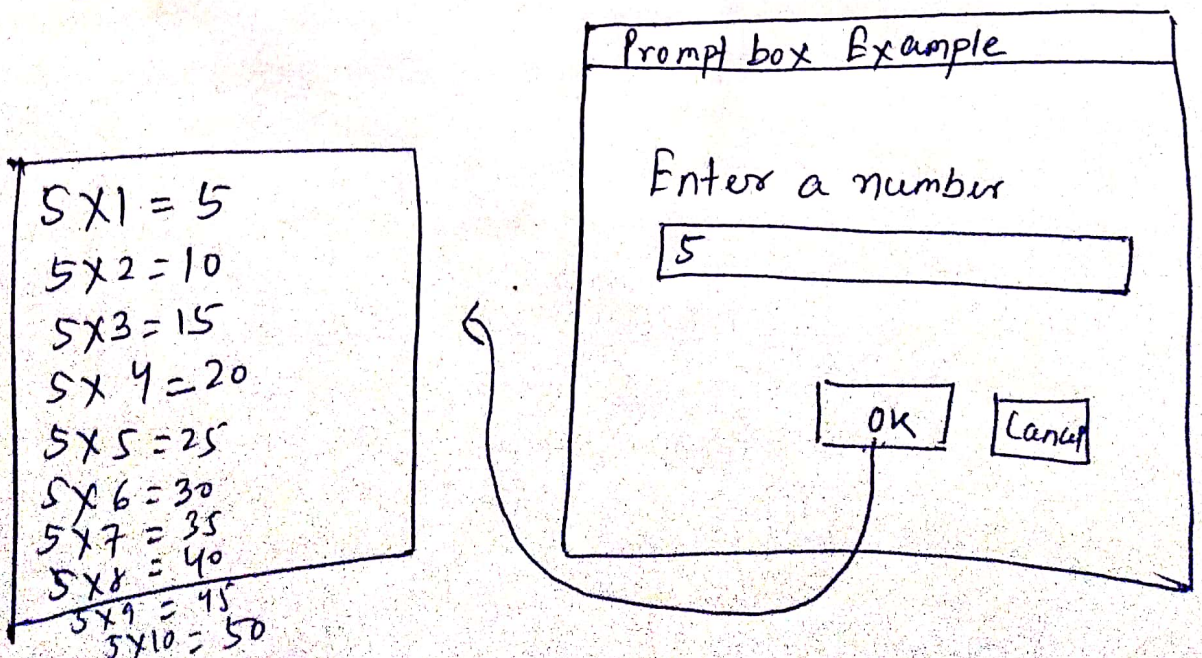
Ex

```
var name = prompt("what's your name?",
                  "your name please..");
```

Example program :→

```
<html>
 <head>
 <title> Prompt box Example </title>
</head>
 <body>
  <script>
     var x = prompt("Enter a number", "5");
    if (x!=null)
     {
          for(i=1; i<=10; i++)
              document.write("<br/>"+ x +" x" +i+"="
                                    + x *i);
     }
 </script></body>
</html>
```

O/P



Java Script Functions

Function is a named block of statements which can be executed again and again simply by writing its name and can return some value.

- Function is a block of code designed to perform certain action.
- It is useful in making a program modular and understandable.

Advantages of Javascript Function :→

1. Code reusability - We can call a function several times so it save coding.
2. Less coding - It makes our program compact. We don't need to write many lines of codes each time to perform a common task.

Defining a Function :→

Function can be defined using the following

syntax: → keyword

```
function <function-name>([arg1, arg2, ....argN])
{
    // Code to be executed
}
```

Example

```
function welcome()
{
    alert("Welcome to IPEC...");
}
```

Javascript Function with Arguments / Parameters :→

The arguments received by the function in a list of corresponding values are called parameters. These values can be assigned to local variables within the function.

Example

```
function welcome(name)
{
    alert("Welcome to IPEC" + name);
}
```

Example Program :→

```
<html>
<head>
<title> Function Example </title>
<script>
function message(name)
{
    alert("Welcome to IPEC" + name + "!");
}
</script>
</head>
<body>
<script>
    var nm = prompt("What is your name?", "your nam
                    please...");
    message(nm);
</script>
</body> </html>
```

Calling the Function :→

A function can be called by writing the name of the function along with the list of arguments. A function call can also be used in an event handler code also.

Syntax

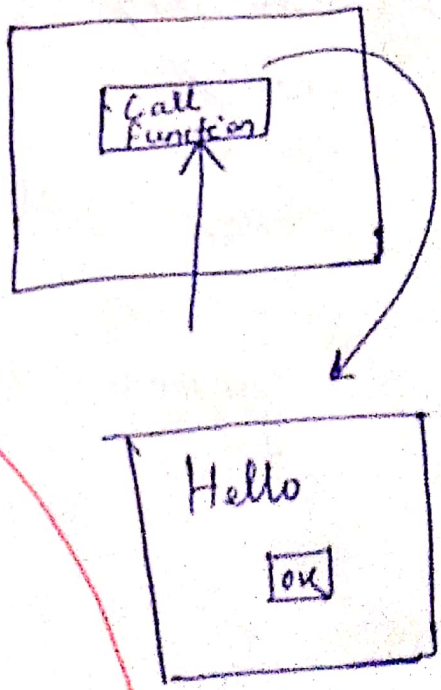
<functionName> ([parameterlist])

ex

Welcome (" Rohit");

Calling a Function from an Event :→

Once a function is defined, it may be used with events like on click event.



Example

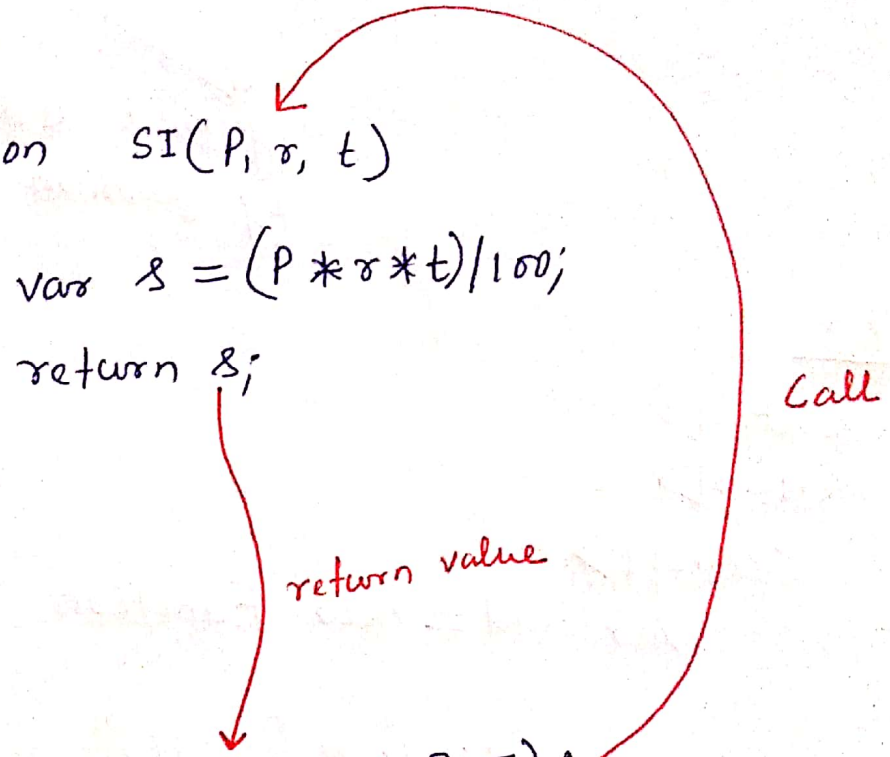
```
<html>
<head>
<script>
   function msg()
   {    alert (" Hello ... ");

   }
</script>
</head>
<body> <center>
   <input type=" button"  onclick=" msg()"
                          value = " Call Function" />
</center>
</body>
</html>
```

Function with Return value :→

We can call function that returns a value and use it in our program.

Example : To calculate Simple Interest.

```
<html>
<head>
<title> Function Returning value </title>
<script>
    function SI(P, r, t)
    {
        var s = (P*r*t)/100;
        return s;
    }
</script>
</head>
<body>
<script>
    var result = SI(1000, 5, 7);
    document.write("Result =" + result);
</script>
</body>
</html>
```



Javascript Function object :→

In, Javascript, the purpose of Function constructor is to create a new Function object. It executes the Code globally.

Syntax

```
new Function ([arg1, [arg2 [, ..argn]],] function body)
```

arguement used by function

function definition

Example

```
<html>
<head>
  <script>
    var add = new Function("num1", "num2", "
                return num1 + num2");
    document.write(add(2,5));
  </script>
</head>
</html>
```

argument

function definition

function call

Function with Default Value :→

If a parameter is not provided, then its value becomes undefined.

```
function showMessage(from, text)
{
  alert(from + ": " + text);
}
showMessage("John");
```

undefined → text

only parameter provided.

O/P

John: undefined

```
function showMessage(
  from, text = "Hello")
{
  alert(from + ": " + text);
}
showMessage("John");
```

Default value

O/P

John: Hello

Differentiate between Java and Javascript :→

Java	Javascript
- Java is strongly typed language and variable must be declared first to use in program. In Java the type of a variable is checked at compile-time.	- Javascript is weekly typed language and have more relaxed syntax and rules.
Java is an object oriented programming language.	- Java script is an object based scripting language.
Java applications can run in any Virtual machine(JVM) or browser.	- Javascript code run on browser only as Javascript is developed for browser only

- Objects of java are class based even we can't make any program in java without class	- Javascript objects are prototype based.
- Java program has file extension ".java" and translates source code into bytecodes which is executed by JVM.	- Javascript file has file extension ".js" and it is interpreted but not compiled, every browser has the Javascript interpreter to execute JS code.
- Java is a standalone language.	- Contained within a web page and integrates with its HTML content.
- Java program uses more memory.	- Javascript requires less memory ~~there~~ therefore it is used in web pages.
- Java has a thread based approach to Concurrency.	- Javascript has event based approach to Concurrency.

Javascript Event handling :→

Changing the state of an object is known as an event. Events are actions that can be detected by Javascript. The process of reacting over the events is called Event handling. Thus, js handles the HTML events via Event handler.

- When a user clicks the mouse
- When a web page has loaded.
- When an image has been loaded.
- When the mouse moves over an element.
- When an input field is changed.
- When an HTML form is submitted.
- When a user stokes a key.

Sometimes we want to execute a Javascript when an event occurs, such as when a user clicks a button.

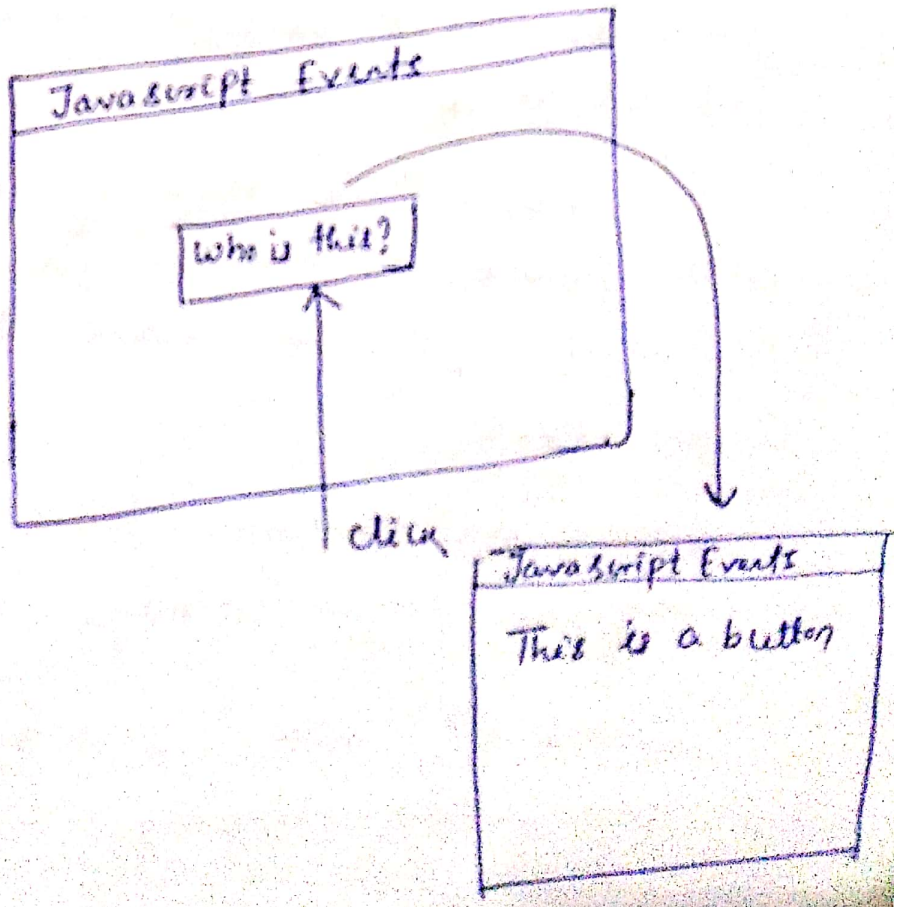
Events are normally used in combination with functions, and the function will not be executed before the event occurs.

Common Events :

- onload and onunload - when page is visited and left
- onfocus, onblur, onchange - events pertaining to form elements.
- onsubmit - when a form is submitted.
- onmouseover, onmouseout - for menu effects.

Example : Mouse click event

```
<html>
<head> <title> Javascript Events </title> </head>
<body>
<script>
    function clickevent()
    {
        document.write(" This is a button");
    }
</script>
<form> <center>
  <input type="button" onclick="clickevent()"
                       value="who is this?" />
</center>
</form>
</body>
</html>
```



Example 2

```
<html>
 <head>
  <script>
    function effect()
    {
        var x = document.getElementById("para1");
         x.style.backgroundColor = "Red";
    }
    function effectback()
    {
        var x = document.getElementById("para1");
         x.style.backgroundColor = "green";
    }
</script></head>
<body>
<p id="para1" onmouseover="effect()"
        onmouseout="effectback()"> This is a
        paragraph </p>
</body>
</html>
```

Javascript Form Validation :→

It is important to validate the form submitted by the user because it can have inappropriate value. So, validation is must to authenticate user.

Javascript provides facility to validate the form on the client-side so data processing will be faster than server-side validation.

Through, Javascript, we can validate name, password, email, date, mobile number etc.

Example : Javascript Example to validate name and password. : →

```
<html>
<head>
<script>
function validate()
{  var name =document.myform.name.value;
   var pass = document.myform.password.value;
   var repass = document.myform.password2.value;
   if (name ==null || name =="")
     {   alert ("Name can't be blank");
         return false;
     }
```

```
else if (pass == "" || pass.length < 8)
 {
    alert("password must be at least 8
                    characters long");
     return false;
 }
 else if(pass!= repass)
  {
     alert ("password must be same");
      return false;
  }
 return true;
}
</script>
<body>
<form name="myform" method="post" action="
                success.html" onsubmit=
                "return validate()" >
Name : <input type="text" name="name"> <br/>
Password: <input type="password" name="password"> <br/>
Re-enter Password <input type="password" name="password1">
<input type="submit" value="login">
</form>
</body>
</html>
```

Example 2: To validate phone number : →

To validate the text field for numeric value only, here we are using "isNaN()" function.

```
<html>
<head>
<script>
 function validate()
 {
     var phno = document.myform.ph.value;
      if (isNaN(phno))
      {
          alert("Enter numeric value only");
          return false;
      }
      else
      { return true;
      }
 }
</script>
<body>
  <form name="myform" onsubmit="return validate()">
phno: <input type="text" name="ph">
     <input type="submit" value="submit">
</form>
  </body></html>
```

Example3: Email Validation :→

There are many criteria that need to validate the email such as:

(1) email id must contain the "@" and "." character.

(2) There must be at least one character before and after "@".

(3) There must be at least two characters after dot (.).

```
<html>
<head>
<script>
function validate()
 {
      var x = document.myform.email.value;
      var atposition = x.indexof("@");
      var dotposition = x.lastIndex(".");
      if(atposition < 1 || dotposition < atposition+2||
                    dotposition+2 >= x.length)
       {
          alert (" Please enter valid email");
          return false;
       }
      return true;
 }
</script>
</head>
```

```
<body>
<form name="myform" method="post"
      action="success.html" onsubmit="return
                                validate()">
Email: <input type="text" name="email"> <br>
  <input type="submit" value="register">
  </form>
  </body>
  </html>
```

Javascript Objects :→

A javascript object is an entity having state and behavior (properties and method).

- Javascript is an object-based language. Everything is an object in Javascript.
- Javascript is template based not class based. Here, we don't create class to get the object. but we directly create objects.

Creating objects in Javascript :→

Objects in Javascript can be created in 3 different ways :

(1) By object literal.
(2) By creating instance of object directly (by using new keyword).
(3) By using an object constructor (using new keyword.

(1) Javascript object by Object literal :→

Syntax

```
object = { property1 : value1, property2 : value2 ....
                                    propertyN : valueN }
```

Ex

```
<script>
emp = {id:102, name:"Ram", salary:40000}
document.write(emp.id + " " + emp.name + " "
                                    + emp.salary);
</script>
```

(2) By creating instance of Object :→

Syntax

```
var objectname = new Object();
```

Ex

```
<script>
  var emp = new Object();
    emp.id = 101;
    emp.name = "Ram";
    emp.salary = 50000;
  document.write(emp.id + " " + emp.name + " " +
                                    emp.salary);
</script>
```

(3) By using an object Constructor :→

Here, we need to create function with argument. Each argument value can be assigned in the current object by using "this" keyword.

- "this" keyword refers to current object.

Ex

```
<script>
  function emp(id, name, salary)
  {   this.id = id;
      this.name = name;
      this.salary = salary;
  }
  e = new emp(103, "Ram", 40000);
  document.write(e.id+" "+e.name+" "+e.salary
</script>
```

Document object Model (DOM) :→

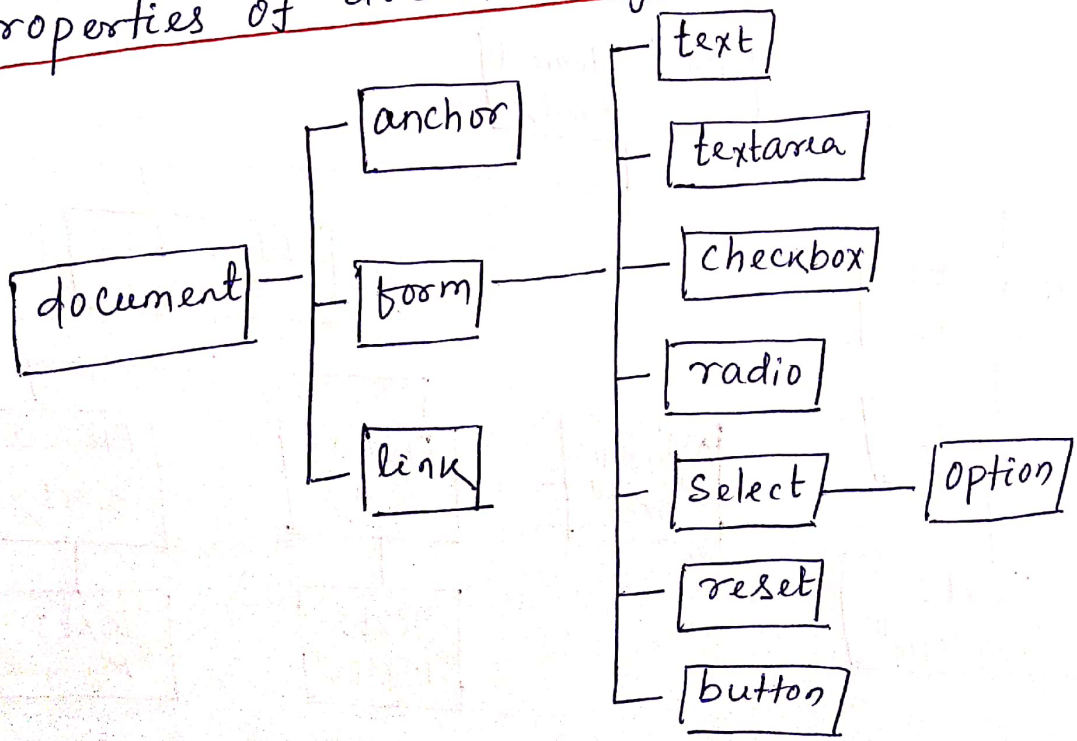
The document object represents the whole html document. When html document is loaded in the browser, it becomes a document object.

- DOM is a specification that determines a mapping between programming language objects and the elements of an HTML document.
- The Document object is one of the parts of the Window object.

~~window object~~ → window.document or document

Note : According to W3C - "The W3C DOM is a platform and language-neutral interface that allows programs and script to dynamically access and update the content, structure and style of document".

Properties of document object :→

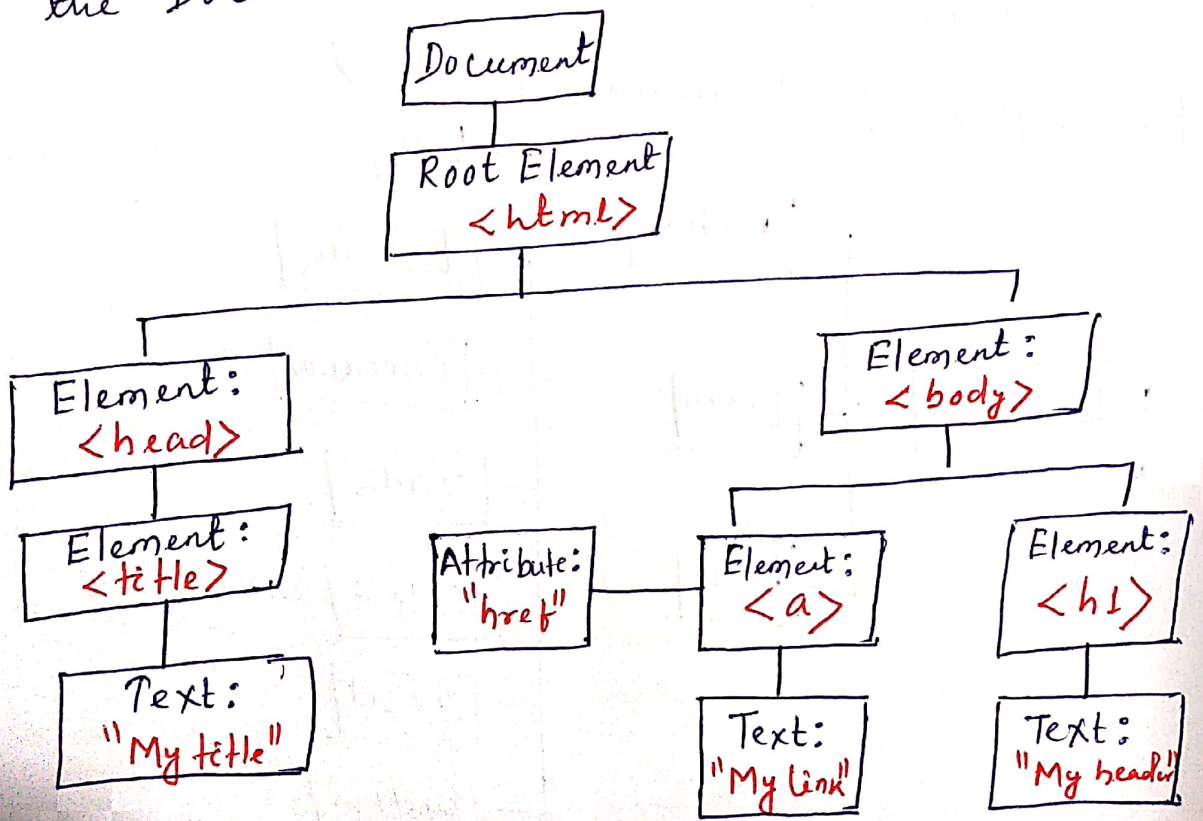


Methods	Purpose
open()	— Opens a document for writing.
write()	— Writes string / data to a document.
writeln()	— Writes string / data followed by a newline character to a document.
close()	— close a document stream for writing.

HTML DOM : The Document Tree :→

Accessing elements and changing their properties lets us do simple things like form validation, data transfer etc.

- HTML DOM lets us do much more
- We can create, delete and modify parts of the HTML document. For this we need to understand the Document Tree.



Navigating Document Tree / Accessing field Value by Document object :→

Methods	Description
- getElementById()	returns the element having the given Id value.
- getElementsByName()	returns all the elements having the given name value.
- getElementsByTagName()	returns all the elements having the given tag name.
- getElementsByClassName()	returns all the elements having the given class name.

Example : Accessing field value using getElementById().

```
<html>
<head>
<script>
    function getResult()
    {
        var num = document.getElementById("num").
                  value;
        alert(num * num * num);
    }
</script>
<body><form>
Enter Number: <input type="text" id="num" /><br/>
<input type="button" value="cube" onclick="getResu
()" />
</form></body></html>.
```

Built-In JavaScript objects :→

Some basic objects are built-in to Java Script such as:

- String
- Date
- Array
- Boolean
- Math

Javascript Array :→

Javascript array is an object that represents a collection of similar type of elements.

There are 3 ways to constructor array in Javascript

1. By array literal
2. By creating instance of Array directly. (using new keyword)
3. By using an Array constructor (using new keyword).

1. By array literal :→

Syntax:

```
var arrayname=[value1, value2, ... valuen];
```

Example

```
<script>
var emp = ["Ram", "Hari", "Rita"];
for(i=0; i<emp.length; i++)
  document.write(emp[i]+"<br/>");
</script>
```

(2) By Array directly (using new keyword) :→

Syntax

```
var arrayname = new Array();
```

Ex

```
<script>
var i;
var emp = new Array();
  emp[0] = "Ram";
  emp[1] = "Hari";
  emp[2] = "Rita";
  for(i in emp){
  document.write(emp[i]+"<br/>");
  }
</script>
```

(3) By array Constructor (new keyword) :→

Ex

```
<script>
  var emp = new Array("Ram", "Hari", "Rita");
    for(i=0; i<emp.length; i++)
      document.write(emp[i]+"<br/>");
</script>
```

Javascript String :→

The Javascript String is an object that represents a sequence of characters. There are 2 ways to Create string.

(1) By String literal
(2) By string object (using new keyword).

(1) By using String literal :→

Syntax

```
var stringname = " String value";
```

ex

```
<script>
    var str = " Thi is a string";
    document.write(str);
</script>
```

(2) By using String object :→

Syntax

```
var stringname = new String(" String value");
```

Ex

```
<script>
    var str = new String(" Hello IPEi");
    document.write(str);
</script>.
```

Java script String methods :→

(1) charAt(index) : returns the character at the given index.

ex
```
var str = "javascript";
document.write(str.charAt[2]); → v
```

(2) concat(str) : concatenates or join two strings

ex:
```
var s1 = "ipec";
var s2 = "college";
var s3 = s1.concat(s2);
document.write(s3); → ipec college.
```

(3) indexof(str) : returns the index position of the given string.

ex
```
var s1 = "My name is John";
           0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
var n = s1.indexOf("name");
document.write(n); → 3
```

(4) lastIndexOf(str) : returns the last index position of the given string.

ex
```
var s1 = "javascript vs java program";
          0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
var n = s1.lastIndexOf("java");
document.write(n); → 14
```

(5) toLowerCase() – to convert lowercase letter
toUpperCase() – to convert uppercase letter

ex
```
var s1 = "Good Morning";
var s2 = s1.toLowerCase(); → good morning
var s3 = s1.toUpperCase(); → GOODMORNING
```

(6) substr(start, length): fetch the part of the given string and return the new string.

ex
```
var str = "JavaScript";
document.write(str.substr(0,4));
```
→ Java

(7) slice(beginIndex, endIndex): returns the parts of string from given beginIndex to endIndex.

ex
```
var s1 = "abcdefgh";
var s2 = s1.slice(2,5); → cde.
```

Javascript Date Object :→

The Javascript date object can be used to get year, month and day. This object is used to set and manipulate date and time.

4 variant of Date constructor to create date object.

1. Date()
2. Date(milliseconds)
3. Date(dateString)
4. Date(year, month, day, hours, minutes, seconds, milliseconds).

Ex
```
<script>
var date = new Date();
var day = date.getDate();
var month = date.getMonth()+1
var year = date.getFullYear();
document.write("Date :" + day+"/"+month
+"/"+year);
</script>
```

O/P
Date: 9/4/2020

Javascript Math :→

The Javascript Math object provides several Constants and methods to perform mathematical operation.

Properties	Description
Math.PI	Returns the value of π.
Math.E	Euler's constant
Math.LN2	Natural logarithm of 2.
Math.LN10	Natural logarithm of 10.
SQRT2	Square root of 2.

Javascript Math methods :→

(1) Math.sqrt(n) : returns square root of given number.

ex Math.sqrt(17); → 4.123105625617661

(2) Math.random() : returns random number bet 0 to 1.

ex Math.random();
→ 0.6082866817138679

(3) Math.pow(m,n) : returns the m to the power of n i.e, m^n.

ex Math.pow(3,4); → 81.

(4) Math.floor(n) : returns the lowest integer for the given number.

ex Math.floor(4.6); → 4.

(5) Math.ceil(n) : returns the largest integer for the given number.

ex Math.ceil(4.6); → 5

(6) Math.round(n) : returns the rounded integer nearest for the given number.

ex Math.round(4.3); → 4

Math.round(4.7); → 5

(7) Math.abs(n) : returns the absolute value for the given number.

ex Math.abs(-4); → 4.

Javascript Number object :→

The Javascript number object enables you to represent a numeric value.

— By the help of Number() constructor, we can create number object.

```
Var n = new Number(value);
```

ex
```
Var n = new Number(16);
```
↳ Integer value by number object.

Javascript Boolean object :→

The Javascript Boolean is an object that represents value in two states: true or false.

— Javascript Boolean object created by Boolean Constructor.

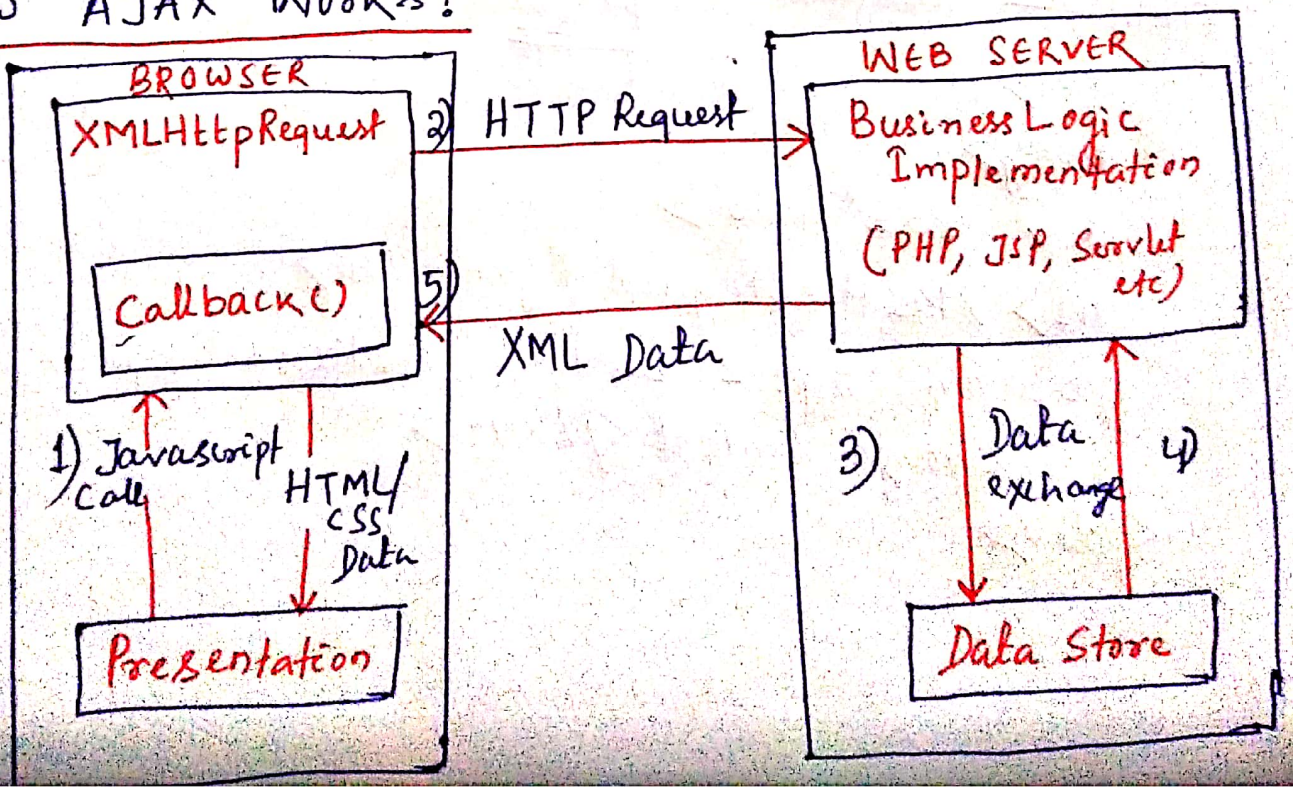
```
Boolean b = new Boolean(value);
```

ex
```
document.write(10<20); → true.
```

AJAX

- AJAX stands for Asynchronous Javascript and XML.
- It is a group of inter-related technologies like Javascript, DOM, XML, HTML, CSS, XMLHttpRequest etc.
- AJAX is not a Programming Language.
- AJAX allows you to send and receive data asynchronously without reloading the web page. So it is fast.
- AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. It is possible to update parts of a webpage, without reloading the whole page.
- AJAX uses a Combination of:
  - A browser built-in XMLHttpRequest Object (to request data from a web server).
  - Javascript and HTML DOM (to display or use the data.

How AJAX Works?



AJAX communicates with the Server using XMLHttpRequest object. XMLHttpRequest object plays a important role.

1. User sends a request from the UI and a javascript call to XMLHttpRequest object.
2. HTTP Request is sent to the Server by XMLHttpRequest object.
3. Server interacts with the database using JSP, PHP, Servlet, ASP.net etc.
4. Data is retrieved.
5. Server sends XML data or JSON data to the XMLHttpRequest callback function.
6. HTML and CSS data is displayed on the browser.

Example: AJAX with Javascript :→

P.html

```
<html>
<body>
<div id="demo">
<h1> XMLHttpRequest Object </h1>
<button type="button" onclick="func()">
    Change Content </button>
</div>
```

```
<script>
function fun()
{
    var req = new XMLHttpRequest();
    req.onreadystatechange = function()
    {
        if (req.readyState == 4 && req.status
                                    == 200)
        {
document.getElementById('demo').innerHTML
            = req.responseText;
        }
    };
    req.open("GET", "ajaxinfo.txt", true);
    req.send();
}
</script>
</body>
</html>
```

ajaxinfo.txt

Hello IPEC

Properties of XMLHttpRequest object :→

Property	Description
① onreadystatechange	It is called whenever readystate attribute changes. It must not be used with asynchronous requests.
② readyState	represents the state of the request. It is ranges from 0 to 4 0 - unopened, open() is not called. 1 - Opened, open() is called but send() is not called. ~~2 - HEA~~ (Server Connection established) 2 - request received. 3 - Processing request. 4 - request finished, response is ready.
③ responseText	returns response as text.
④ responseXML	returns response as XML.
⑤ Status	200 : "OK" 403 : "forbidden" 404 : "Page not found"

Methods of XMLHttpRequest Object :→

Method	Description
① void open(method, URL)	– opens the request specifying get or post method and URL
② void open(method, URL, async)	– same as above but Specifies asynchronous or not
③ void open(method, URL, async, username, password)	– same as above but specifies username and password.
④ void send()	– sends get request.
⑤ void send(string)	– sends post request.
⑥ setRequestHeader(header, value)	– it adds request headers.