

CS-ELECTIVE -1

RCS-E12: WEB TECHNOLOGIES

Unit -3

Scripting: Java script: Introduction, documents, forms, statements, functions, objects; introduction to AJAX,

Networking : Internet Addressing, InetAddress, Factory Methods, Instance Methods, TCP/IP Client Sockets, URL, URL Connection, TCP/IP Server Sockets, Datagram.

DHTML Technologies -

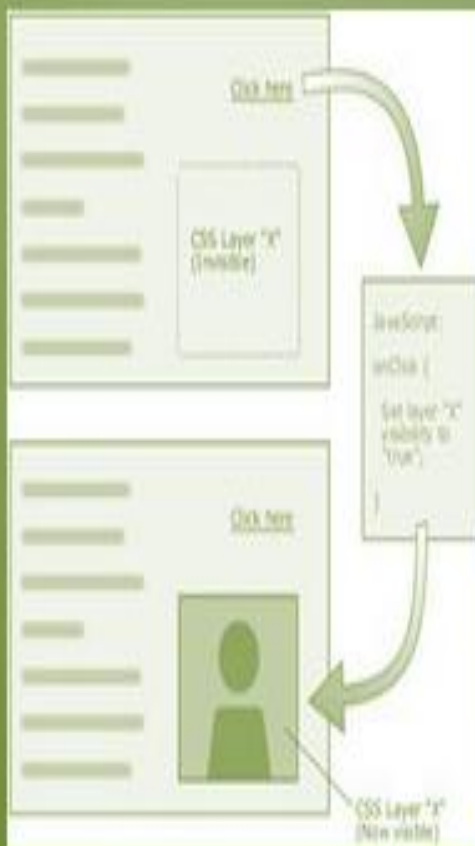
- ♦ What is DHTML?
- ♦ DHTML Technologies
 - ♦ XHTML, CSS, JavaScript, DOM





DHTML

Dynamic Behavior at the Client Side



What is DHTML ?

- ♦ Dynamic HTML (DHTML)

- ♦ Makes possible a Web page to react and change in response to the user's actions

- ♦ $\text{DHTML} = \text{HTML} + \text{CSS} + \text{JavaScript}$



✂telerik

javascript
for
beginner

```
String.prototype =  
function ()  
{  
    return this  
    .replace("Pm", "PM")  
    .replace("a$", "A");  
}
```

.js



Js

Introduction to JavaScript

if javascript

DOM

Style

Events



JavaScript

- ♦ JavaScript is a front-end scripting language developed by Netscape for dynamic content
 - ♦ Lightweight, but with limited capabilities
 - ♦ Can be used as object-oriented language
- ♦ Client-side technology
 - ♦ Embedded in your HTML page
 - ♦ Interpreted by the Web browser
- ♦ Simple and flexible
- ♦ Powerful to manipulate the DOM

JavaScript Advantages

- ♦ JavaScript allows interactivity such as:
 - ♦ Implementing form validation
 - ♦ React to user actions, e.g. handle keys
 - ♦ Changing an image on moving mouse over it
 - ♦ Sections of a page appearing and disappearing
 - ♦ Content loading and changing dynamically
 - ♦ Performing complex calculations
 - ♦ Custom HTML controls, e.g. scrollable table
 - ♦ Implementing AJAX functionality

What can JavaScript Do?

- ♦ Can handle events
- ♦ Can read and write HTML elements and modify the DOM tree
- ♦ Can validate form data
- ♦ Can access / modify browser cookies
- ♦ Can detect the user's browser and OS
- ♦ Can be used as object-oriented language
- ♦ Can handle exceptions
- ♦ Can perform asynchronous server calls (AJAX)

First JavaScript Program

first-script.html

```
<html>

<body>
  <script type="text/javascript">
    alert('Hello JavaScript!');
  </script>
</body>

</html>
```



Another small Example

small-example.html

```
<html>

<body>
  <script type="text/javascript">
    document.write('JavaScript rulez!');
  </script>
</body>

</html>
```



Using javaScript Code

- ♦ The JavaScript code can be placed in:
 - ♦ `<script>` tag in the head
 - ♦ `<script>` tag in the body – not recommended
 - ♦ External files, linked via `<script>` tag the head
 - ♦ Files usually have .js extension

```
<script src="scripts.js" type="text/javascript">  
<!-- code placed here will not be executed! -->  
</script>
```

- ♦ Highly recommended
- ♦ The .js files get cached by the browser

JavaScript – When is executed ?

- ♦ JavaScript code is executed during the page loading or when the browser fires an event
 - ♦ All statements are executed at page loading
 - ♦ Some statements just define functions that can be called later
- ♦ Function calls or code can be attached as "event handlers" via tag attributes
 - ♦ Executed when the event is fired by the browser

```


```


Calling a JavaScript Function from Event Handler - Example

```
<html>
<head>
<script type="text/javascript">
  function test (message) {
    alert(message);
  }
</script>
</head>

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

image-onclick.html

A screenshot of a web browser window titled "Javascript - onclick Event". The address bar shows "image-onclick.html". On the left side of the browser, there is a green circular button with the text "Click Me!". An alert dialog box titled "Javascript Alert" is displayed in the center of the browser window, containing the text "clicked!" and an "OK" button at the bottom right.

Using External Script Files

Using external script files:

```
<html>
<head>
  <script src="sample.js" type="text/javascript">
  </script>
</head>
<body>
  <button onclick="sample()" value="Call JavaScript
    function from sample.js" />
</body>
</html>
```

external-JavaScript.html

External JavaScript file:

```
function sample() {
  alert('Hello from sample.js!')
}
```

sample.js



Sum of Numbers – Example

sum-of-numbers.html

```
<html>

<head>
  <title>JavaScript Demo</title>
  <script type="text/javascript">
    function calcSum() {
      value1 =
        parseInt(document.mainForm.textBox1.value);
      value2 =
        parseInt(document.mainForm.textBox2.value);
      sum = value1 + value2;
      document.mainForm.textBoxSum.value = sum;
    }
  </script>
</head>
```

Sum of Numbers – Example 2

sum-of-numbers.html (cont.)

```
<body>
  <form name="mainForm">
    <input type="text" name="textBox1" /> <br/>
    <input type="text" name="textBox2" /> <br/>
    <input type="button" value="Process"
      onclick="javascript: calcSum()" />
    <input type="text" name="textBoxSum"
      readonly="readonly"/>
  </form>
</body>

</html>
```



JavaScript Prompt Example

prompt.html

```
price = prompt("Enter the price", "10.00");  
alert('Price + VAT = ' + price * 1.2);
```





The JavaScript Object Model

Document Object Model (DOM)

HTML DOM Event Model

- ♦ JavaScript can register event handlers

- ♦ Events are fired by the Browser and are sent to the specified JavaScript event handler function
- ♦ Can be set with HTML attributes:

```

```

- ♦ Can be accessed through the DOM:

```
var img = document.getElementById("myImage");  
img.onclick = imageClicked;
```


The HTML DOM Event Model 2

- ♦ All event handlers receive one parameter
 - ♦ It brings information about the event
 - ♦ Contains the type of the event (mouse click, key press, etc.)
 - ♦ Data about the location where the event has been fired (e.g. mouse coordinates)
 - ♦ Holds a reference to the event sender
 - ♦ E.g. the button that was clicked

Common DOM Events

- ♦ Mouse events:

- ♦ onclick, onmousedown, onmouseup
- ♦ onmouseover, onmouseout, onmousemove

- ♦ Key events:

- ♦ onkeypress, onkeydown, onkeyup
- ♦ Only for input fields

- ♦ Interface events:

- ♦ onblur, onfocus
- ♦ onscroll

Common DOM Events

♦ Form events

- ♦ onchange – for input fields
- ♦ onsubmit
 - ♦ Allows you to cancel a form submission
 - ♦ Useful for form validation

♦ Miscellaneous events

- ♦ onload, onunload
 - ♦ Allowed only for the <body> element
 - ♦ Fires when all content on the page was loaded / unloaded

Onload Event - Example



Objects

- JavaScript is designed on a simple object-based paradigm. An object is a collection of properties, and a property is an association between a name (or *key*) and a value. A property's value can be a function, in which case the property is known as a method. In addition to objects that are predefined in the browser, you can define your own objects. This chapter describes how to use objects, properties, functions, and methods, and how to create your own objects.
- Using objects follows the syntax of C++/Java:

`objectname.attributeName`

`objectname.methodname ()`

Objects are Variables - W

- JavaScript variables can contain single values:

Example

```
var person = "John Doe";
```

- Objects are variables too. But objects can contain many values.
- The values are written as **name : value** pairs (name and value separated by a colon).

Example

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```


Built-in Browser Objects

- ♦ The browser provides some read-only data via:
 - ♦ window
 - ♦ The top node of the DOM tree
 - ♦ Represents the browser's window
 - ♦ document
 - ♦ holds information the current loaded document
 - ♦ screen
 - ♦ Holds the user's display properties
 - ♦ browser
 - ♦ Holds information about the browser

Opening New Window - Example

♦ window.open()

window-open.html

```
var newWindow = window.open("", "sampleWindow",  
    "width=300, height=100, menubar=yes,  
    status=yes, resizable=yes");
```

```
newWindow.document.write(  
    "<html><head><title>  
    Sample Title</title>  
    </head><body><h1>Sample  
    Text</h1></body>");  
newWindow.status =  
    "Hello folks";
```



The Navigator Object

```
alert(window.navigator.userAgent);
```



Document and Location

♦ document object

- ♦ Provides some built-in arrays of specific objects on the currently loaded Web page

```
document.links[0].href = "yahoo.com";  
document.write(  
    "This is some <b>bold text</b>");
```

♦ document.location

- ♦ Used to access the currently open URL or redirect the browser

```
document.location = "http://www.yahoo.com/";
```

Form Validation - Example

form-validation.html

```
function checkForm()
{
    var valid = true;
    if (document.mainForm.firstName.value == "") {
        alert("Please type in your first name!");
        document.getElementById("firstNameError").
            style.display = "inline";
        valid = false;
    }
    return valid;
}

...
<form name="mainForm" onsubmit="return checkForm()">
    <input type="text" name="firstName" />
    ...
</form>
```


The Math Objects

- ♦ The Math object provides some mathematical functions

math.html

```
for (i=1; i<=20; i++) {  
    var x = Math.random();  
    x = 10*x + 1;  
    x = Math.floor(x);  
    document.write(  
        "Random number (" +  
        i + ") in range " +  
        "1..10 --> " + x +  
        "<br/>");  
}
```

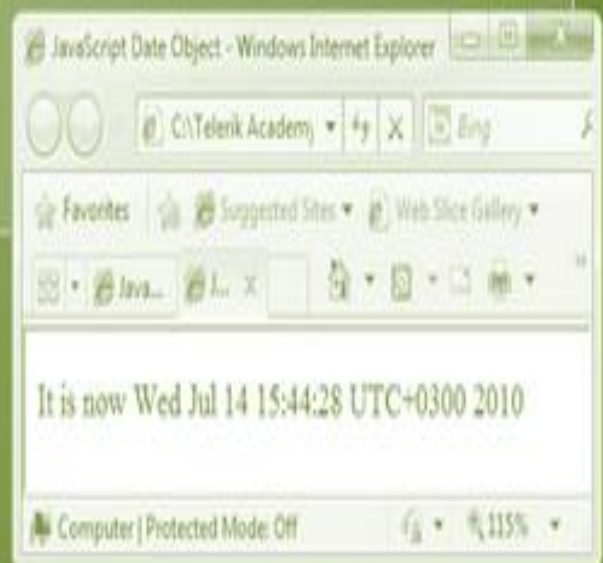


The Date Object

- ♦ The Date object provides date / calendar functions

dates.html

```
var now = new Date();  
var result = "It is now " + now;  
document.getElementById("timeField")  
    .innerText = result;  
...  
<p id="timeField"></p>
```



Timer - Example

timer-demo.html

```
<script type="text/javascript">
  function timerFunc() {
    var now = new Date();
    var hour = now.getHours();
    var min = now.getMinutes();
    var sec = now.getSeconds();
    document.getElementById("clock").value =
      "" + hour + ":" + min + ":" + sec;
  }

  setInterval('timerFunc()', 1000);
</script>

<input type="text" id="clock" />
```

Debugging JavaScript



Debugging JavaScript

- ♦ Modern browsers have JavaScript console where errors in scripts are reported
 - ♦ Errors may differ across browsers
- ♦ Several tools to debug JavaScript
 - ♦ Microsoft Script Editor
 - ♦ Add-on for Internet Explorer
 - ♦ Supports breakpoints, watches
 - ♦ JavaScript statement debugger; opens the script editor

Firebug

- ♦ Firebug – Firefox add-on for debugging JavaScript, CSS, HTML
 - ♦ Supports breakpoints, watches, JavaScript console editor
 - ♦ Very useful for CSS and HTML too
 - ♦ You can edit all the document real-time: CSS, HTML, etc
 - ♦ Shows how CSS rules apply to element
 - ♦ Shows Ajax requests and responses
 - ♦ Firebug is written mostly in JavaScript

JavaScript Console Object

- ♦ The `console` object exists only if there is a debugging tool that supports it
 - ♦ Used to write log messages at runtime
- ♦ Methods of the `console` object:
 - ♦ `debug(message)`
 - ♦ `info(message)`
 - ♦ `log(message)`
 - ♦ `warn(message)`
 - ♦ `error(message)`

Introduction to AJAX

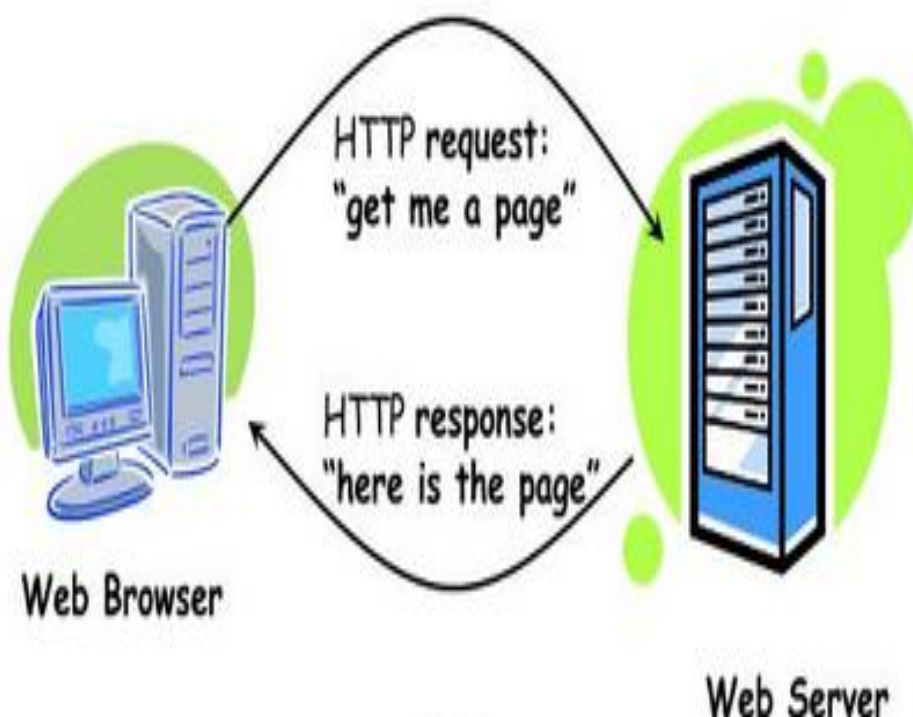
Asynchronous JavaScript And XML

- Ajax isn't a new technology or programming language.
- It is a technique used to develop interactive web applications that are able to process a user request immediately.
- Ajax can selectively modify a part of a page displayed by the browser, and update it without the need to reload the whole document with all images, menus, etc.

HTTP is a transaction-based communication protocol

Each HTTP request generates a response

Typically the browser requests a (HTML) page, the server responds by sending it, and the browser then displays it

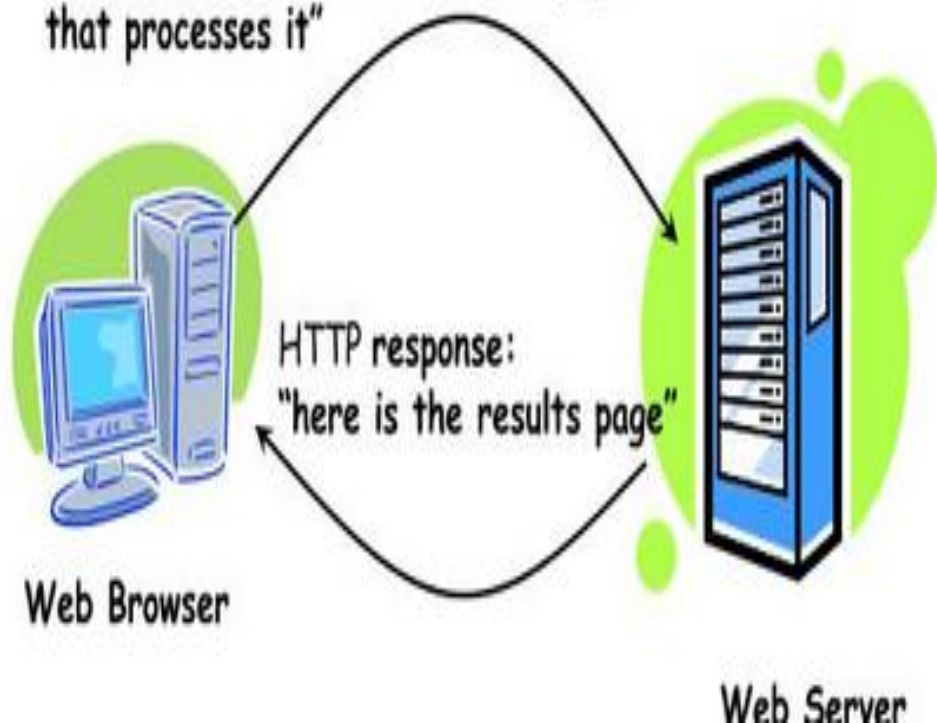


Subsequent transactions are typically requests for other web pages

For example, the browser requests another page to process the form data entered in the currently displayed page

HTTP request:

"here's some data; load another page that processes it"



Ajax is a methodology for requesting a response from the server that does not involve a change in the current web page

For example, the browser requests just the information needed to validate username/password part of a form

AJAX request:

"here's is a username/password to be validated"

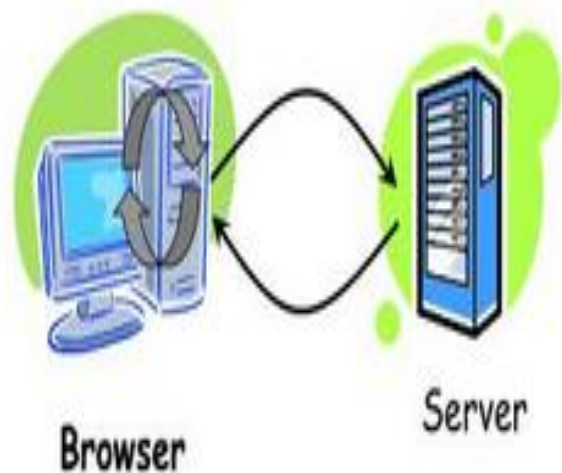


After receiving the response, the browser incorporates the results into the current page being displayed (using JavaScript)

Ajax requests can be *Synchronous* or *Asynchronous*

- The browser can continue to execute (JavaScript) while awaiting the results of an **Asynchronous** request

- Use if results take a long time to generate
- Call setup is more complex



- **Synchronous** requests cause the browser to stop and wait for a request

- OK if browser responds quickly
- Call setup is simple



Ajax request/response is implemented in JavaScript



- The XMLHttpRequest object is the workhorse of Ajax

```
var xhr = new XMLHttpRequest();
```

- You use this object (and it's methods and attributes) from within Javascript

IP Address

- An **Internet Protocol address (IP address)** is a numerical label assigned to each device (e.g., computer, printer) participating in a computer network that uses the Internet Protocol for communication.
- An IP address serves two principal functions:
 - host or network interface identification
 - location addressing.

IP Address

- IP addresses format
 - binary numbers,
 - usually stored in text files and displayed in human-readable notations, such as 172.16.254.1(IPv4)
- IPv4
 - 32-bit number
 - still in use today.
- IPv6
 - 128-bits number
 - developed in 1995.

IP Address


- ▶ What is an IP address...?

- An IP address is a unique global address for a network interface

- is a **32 bit long** identifier

- encodes a network number (**network prefix**) and a **host number**

10000000	10001111	10001001	10010000
1 st Byte	2 nd Byte	3 rd Byte	4 th Byte
= 128	= 143	= 137	= 144



128.143.137.144

Journey to IP Versions...

- ▶ IPV(1–3) : were not formally assigned.
- ▶ IPV4 : TCP/IP , 32bit IP address currently used.
- ▶ IPV5 : Internet Stream Protocol (SP)
 - Experimental Protocol
 - Never Introduced for public use.
- ▶ IPV6 : Designed to replace IPV4 , 128bit IP address

IPv4 Supporting Devices..

- PCs
- Servers
- Modems
- Routers
- Printers
- Cameras
- Smart Phones
- Tablets & Gaming Systems
- Just about anything else connecting to the Internet



Java InetAddress Class

InetAddress

Java InetAddress Class is used to encapsulate the two thing.

1. **Numeric IP Address**
2. **The domain name for that address.**

You interact with this class by using the name of an IP host, which is more convenient and understandable than its IP address. The InetAddress class hides the number inside.

InetAddress can handle both IPv4 and IPv6 addresses.

Factory Methods

The `InetAddress` class has no visible constructors. To create an `InetAddress` object, you have to use one of the available factory methods.

Factory methods are merely a convention where by static methods in a class return an instance of that class. This is done in lieu of overloading a constructor with various parameter lists when having unique method names makes the results much clearer.

Method	Description
<code>public static InetAddress getByName(String host) throws UnknownHostException</code>	It returns the object of <code>InetAddress</code> containing <code>LocalHost</code> IP and name.
<code>public static InetAddress getLocalHost() throws UnknownHostException</code>	It returns the object of <code>InetAddress</code> containing local host name and address.
<code>public static InetAddress[] getAllByName(String <i>hostName</i>) throws UnknownHostException</code>	It returns an array of <code>InetAddresses</code> that represent all of the addresses that a particular name resolves to.

Syntax of these Methods

(1) **getLocalHost()**: It returns the `InetAddress` object that represents the local host contain the name and address both. If this method unable to find out the host name, it throw an `UnknownHostException`.

Syntax:

```
Static InetAddress getLocalHost() throws UnknownHostException
```

(2) **getByName()**: It returns an InetAddress for a host name passed to it as a parameter argument. If this method unable to find out the host name, it throw an UnknownHostException.

Syntax:

Static InetAddress getByName(String host_name) throws
UnknownHostException

(3) **getAllByName()**: It returns an array of an InetAddress that represent all of the addresses that a particular name resolve to it. If this method can't find out the name to at least one address, it throw an UnknownHostException.

Syntax:

Static InetAddress[] getAllByName(String host_name) throws
UnknownHostException

Instance Methods

- The **InetAddress** class also has several other methods, which can be used on the objects returned by the Factory methods

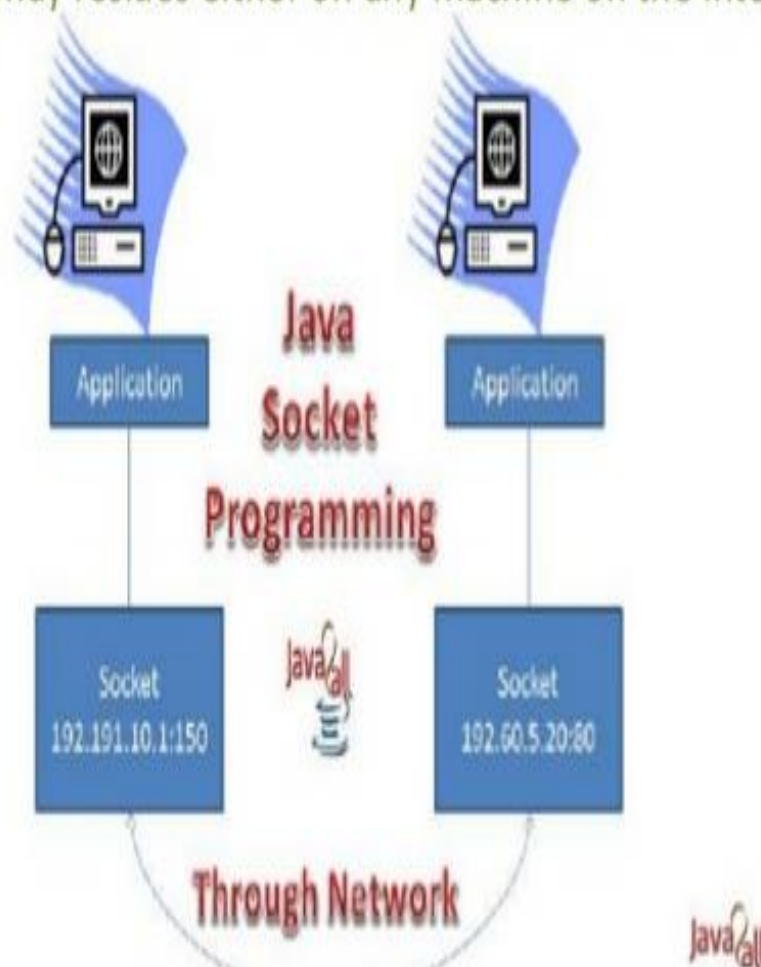
Method	Description
<code>boolean equals(Object <i>other</i>)</code>	Returns true if this object has the same Internet address as <i>other</i> .
<code>byte[] getAddress()</code>	Returns a byte array that represents the object's Internet address in network byte order.
<code>String getHostAddress()</code>	Returns a string that represents the host address associated with the InetAddress object.
<code>String getHostName()</code>	Returns a string that represents the host name associated with the InetAddress object.
<code>boolean isMulticastAddress()</code>	Returns true if this Internet address is a multicast address. Otherwise, it returns false .
<code>String toString()</code>	Returns a string that lists the host name and the IP address

socket programming in java

socket programming in java is very important topic and concept of network programming.

Java network Programming supports the concept of socket. A socket identifies an endpoint in a network. The socket communication take place via a protocol.

A socket can be used to connect JAVA Input/Output system to other programs that may resides either on any machine on the Internet or on the local machine.



TCP/IP Sockets:

TCP/IP sockets are used to implement point-to-point, reliable, bidirectional, stream-based connections between hosts on the Internet.

There are two types of TCP sockets available in java:

(1) **TCP/IP Client Socket**

(2) **TCP/IP Server Socket**

(1) **TCP/IP Client Socket:**

The Socket class (available in java.net package) is for the Client Socket. It is designed to connect to server sockets and initiate protocol exchange. There are two constructors used to create client sockets type object.

(a) **Socket**(String host_name,int port) throws **UnknownHostException,IOException** it creates a socket that is connected to the given host_name and port number.

(b) **Socket(InetAddress ip,int port)** throws **IOException** it creates a socket using a pre-existing **InetAddress** object and a port number.

(2) TCP/IP Server Socket:

The **ServerSocket** class (available in **java.net** package) is for the Server. It is designed to be a “listener”, which waits for clients to connect before doing anything and that listen for either local or remote client programs to connect to them on given port.

When you create **ServerSocket** it will register itself with the system as having an interest in client connection.

Syntax:

ServerSocket(int port) throws **IOException**

Program: Write down a program which demonstrate the Socket programming for passing the message from server to client.

Client.java:

```
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;
import java.net.UnknownHostException;
public class Client {
    public static void main(String[] args) {
        System.out.println("Sending a request.....");
        try {
            Socket s = new Socket("127.0.0.1",1564);
            System.out.println("connected successfully.....");
            BufferedReader br = new BufferedReader(new
            InputStreamReader(s.getInputStream()));
            System.out.println("response from server...");
            System.out.println("Client side : "+br.readLine()); s.close();
        }
        catch (UnknownHostException e) {
            System.out.println("Not find the IP-ADDRESS for :"+e);
        }
        catch (IOException e) {
            System.out.println("Not Found data for Socket : "+e);
        }
    }
}
```


Server.java:

```
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;
public class Server
{
    public static void main(String[] args)
    {
        try
        {
            ServerSocket ss = new ServerSocket(1564);
            System.out.println("waiting for request....");
            Socket s = ss.accept();
            System.out.println("Request accepted");
            PrintStream ps = new PrintStream(s.getOutputStream());
            BufferedReader br = new BufferedReader(new
            InputStreamReader(System.in));
            System.out.println("Input the data at server : ");
            ps.print(br.readLine());
            s.close();
            ss.close();
        }
        catch (Exception e)
        {
            System.out.println("Not Found data for Socket : "+e);
        }
    }
}
```



For Output follow the below step:

(1) **Run server.java**

Console:

waiting for request....

(2) **Run Client.java**

Console:

waiting for request....

Request accepted

Input the data at server:

(3) **Now enter the message at console**

Input the data at server:

welcome at server

(4) **Then press Enter.**



<http://www.java2all.com>

(5) Sending a request.....

connected successfully.....

response from server...

Client side: welcome at server

Uniform Resource Locator (URL)

The Web is a loose collection of higher-level protocols and file formats, all unified in a web browser. One of the most important aspects of the Web is that Tim Berners-Lee devised a scaleable way to locate all of the resources of the Net.

Once you can reliably name anything and everything, it becomes a very powerful paradigm. The Uniform Resource Locator (URL) does exactly that.

The URL provides a reasonably intelligible form to uniquely identify or address information on the Internet.

URLs are everywhere, every browser uses them to identify information on the Web. Within Java's network class library,

All URLs share the same basic format Here are two examples:

http://www.osborne.com/ and http://www.osborne.com:80/index.htm.

A URL specification is based on four components.

- 1. Protocol:** to use, separated from the rest of the locator by a colon (:). Common protocols are HTTP, FTP,
- 2. Host name or IP address of the host to use:** this is delimited on the left by double slashes (//) and on the right by a slash (/) or optionally a colon (:).
- 3. Port number:** is an optional parameter, delimited on the left from the host name by a colon (:) and on the right by a slash (/).
- 4. File path.** Most HTTP servers will append a file named **index.html** or **index.htm** to **URLs that refer directly to a directory resource.**

Thus, **http://www.osborne.com/** is the same as
http://www.osborne.com**index.htm.**

URL Connection

- 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.
- To create object of URLConnection class the `openConnection()` method of URL class is used.
- Syntax: `public URLConnection openConnection()` throws `IOException`
- `getInputStream()` method is used to display all the data of a webpage. It returns all the data of the specified URL in the stream that can be read and displayed.

Datagrams

TCP/IP-style networking is appropriate for most networking needs. It provides a serialized, predictable, reliable stream of packet data. This is not without its cost, however. TCP includes many complicated algorithms for dealing with congestion control on crowded networks, as well as pessimistic expectations about packet loss. This leads to a somewhat inefficient way to transport data. Datagrams provide an alternative.

Datagrams are bundles of information passed between machines. They are some what like a hard throw from a well-trained but blindfolded catcher to the third baseman. Once the datagram has been released to its intended target, there is no assurance that it will arrive or even that someone will be there to catch it. Likewise, when the datagram is received, there is no assurance that it hasn't been damaged in transit or that whoever sent it is still there to receive a response.

Java implements datagram's on top of the UDP protocol by using two classes:

DatagramPacket object is the data container,

DatagramSocket is the mechanism used to send or receive the Datagram Packets.