Unit 3

Web Technology

Content

- Introduction to JavaScript
- JS Dialogue Boxes
- JS DOM
- AJAX & its working
- AJAX life cycle
- Socket Programming
- TCP/IP Server socket
- URL class
- Cookies
- Datagrams

Introduction to JavaScript

- JavaScript was created by Brendan Eich in 1995. Today it has far more powerful uses and companies like Google and Facebook use JavaScript to build complex, desktop-like web applications. With the launch of Node.js, It has also become one of the most popular languages for building server-side software.
- JavaScript is a high level, interpreted, programming language used to make web pages more interactive. It is a very powerful client-side scripting language which makes our webpage more lively and interactive.

Where to write JavaScript?

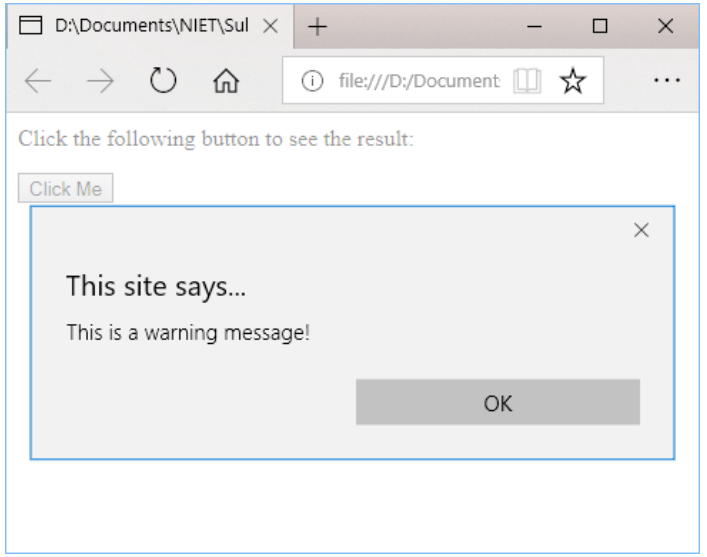
- Scripts in a page will be executed immediately when the page loads. This is not always what we want. Sometimes we want to execute scripts when a page loads, other times when a user triggers an event.
- Physically the script code can be placed at three different places in the html files.
  - Head section
  - Body section
  - External file

JavaScript - Dialog Boxes

- JavaScript supports three important types of dialog boxes.
- These dialog boxes can be used to raise an alert, or to get confirmation on any input or to have a kind of input from the users.

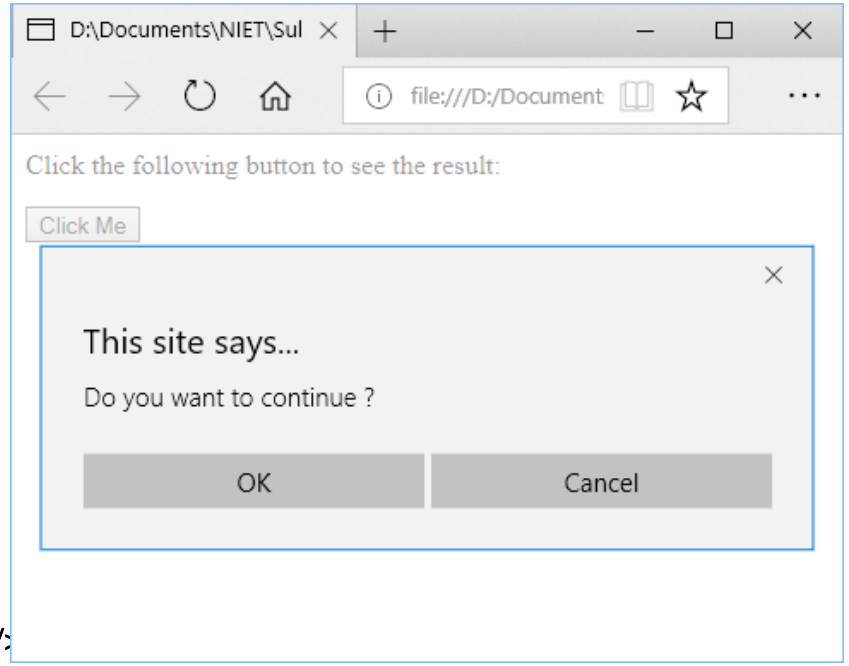
Alert Dialog Box

```
<html>
  <head>
    <script type="text/javascript">
      <!--
        function Warn()
                  {
          alert ("This is a warning message!");
          document.write ("This is a warning message!");
        }
      //-->
    </script>
  </head>
  <body>
    <p>Click the following button to see the result: </p>
    <form>
      <input type="button" value="Click Me" onclick="Warn();" />
    </form>
  </body>
</html>
```



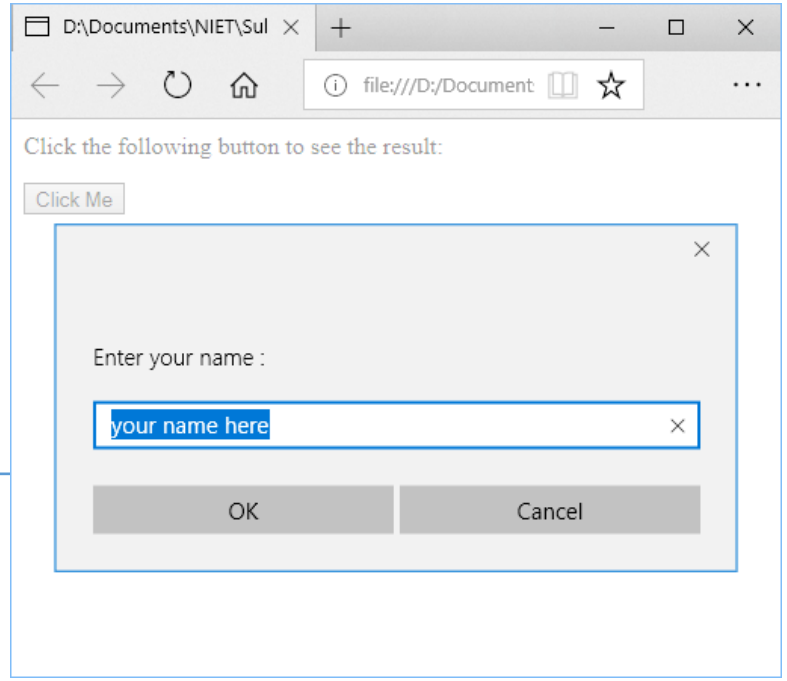
Confirmation Dialog Box

```
<html>
  <head>
    <script type="text/javascript">
      <!--
        function getConfirmation()
                  {
          var retVal = confirm("Do you want to continue ?");
          if( retVal == true ){
            document.write ("User wants to continue!");
            return true;
          }
          else
          {
            document.write ("User does not want to continue!");
            return false;
          }
        }
      //-->
    </script>
  </head>
  <body>
    <p>Click the following button to see the result: </p>
    <form>
      <input type="button" value="Click Me" onclick="getConfirmation();" />
    </form>
  </body>
</html>
```



Prompt Dialog Box

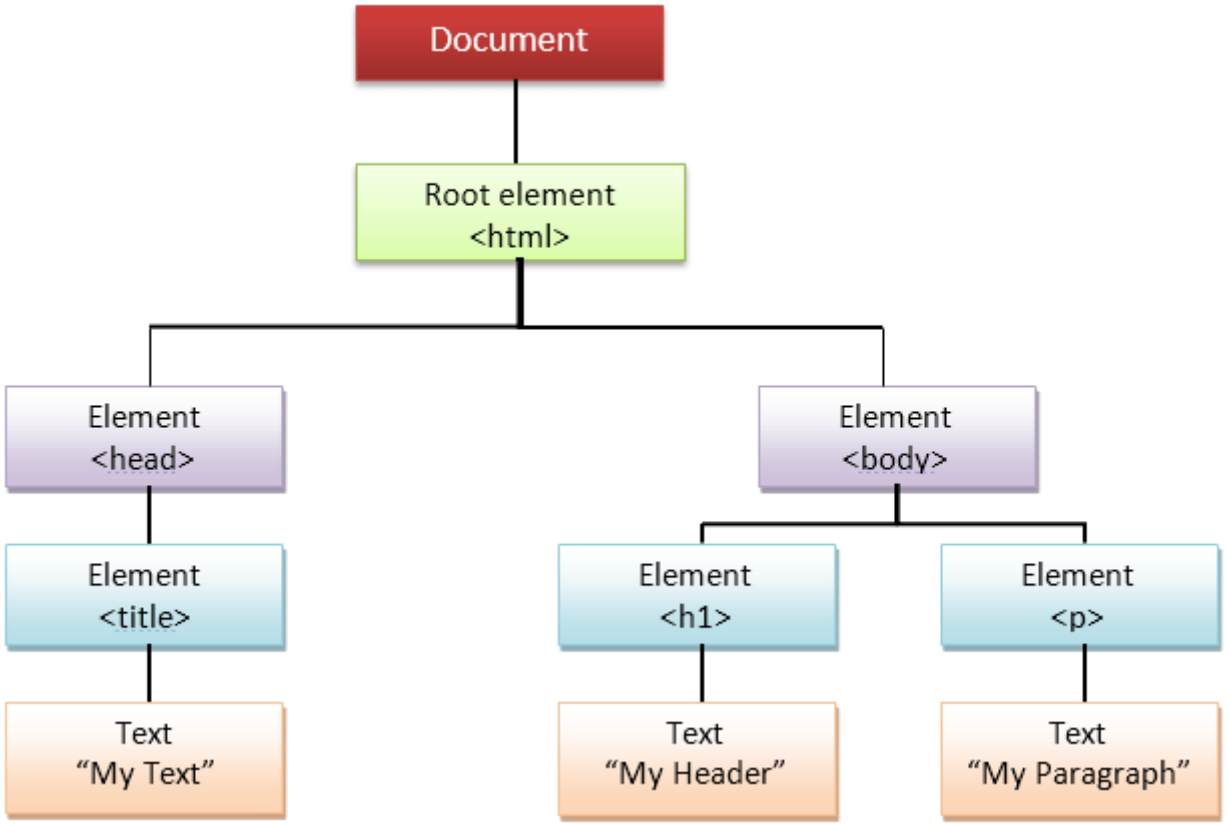
```
<html>
  <head>
    <script type="text/javascript">
      <!--
        function getValue()
                  {
          var retVal = prompt("Enter your name : ", "your name here");
          document.write("You have entered : " + retVal);
        }
      //-->
    </script>
  </head>
  <body>
    <p>Click the following button to see the result: </p>
    <form>
      <input type="button" value="Click Me" onclick="getValue();" />
    </form>
  </body>
</html>
```



JavaScript DOM

- JavaScript can access all the elements in a webpage making use of Document Object Model (DOM). In fact, the web browser creates a DOM of the webpage when the page is loaded.
- The Document Object Model, or DOM for short, is a platform and language independent model to represent the HTML or XML documents. It defines the logical structure of the documents and the way in which they can be accessed and manipulated by an application program.

JavaScript DOM



What is AJAX?

- Asynchronous JavaScript and XML (AJAX) is the art of exchanging data with a server, and updating parts of a web page – without reloading the whole webpage.
- In other words, AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes.
- If application are not using AJAX, then it will have to reload the webpage on every request user made.

How AJAX works?

- It is important to understand that Ajax is not a single technology, but a group of technologies e.g. HTML, CSS, DOM and JavaScript etc.
- HTML and CSS can be used in combination to mark up and style information.
- The DOM is accessed with JavaScript to dynamically display, and allow the user to interact with, the information presented.
- JavaScript and the XMLHttpRequest object provide a method for exchanging data asynchronously between browser and server to avoid full page reloads.

AJAX life cycle

- Normally, an AJAX call to server and getting back response (life cycle events) from server involve following steps:
  1. We type the URL of a webpage in browser's address bar and hit enter. Page is loaded in browser window.
  2. Some action triggers an event, like the user clicking a button.
  3. Event fires the AJAX call, and sends a request to a server using XML or JSON.
  4. The server service takes the input from AJAX/HTTP request, and processes the request. It also prepare the response data if required.
  5. Server sends the data back to the original webpage that made the request.
  6. Another JavaScript function, called a callback function, receives the data, and updates the web page.

Introduction to AJAX

AJAX is an acronym for **Asynchronous JavaScript and XML**. It is a group of inter-related technologies like JavaScript, DOM, XML, HTML /XHTML, CSS, XML HttpRequest etc.

AJAX allows you to send and receive data asynchronously without reloading the web page. So it is fast.

AJAX allows you to send only important information to the server not the entire page. So only valuable data from the client side is routed to the server side. It makes your application interactive and faster.

Where it is used?

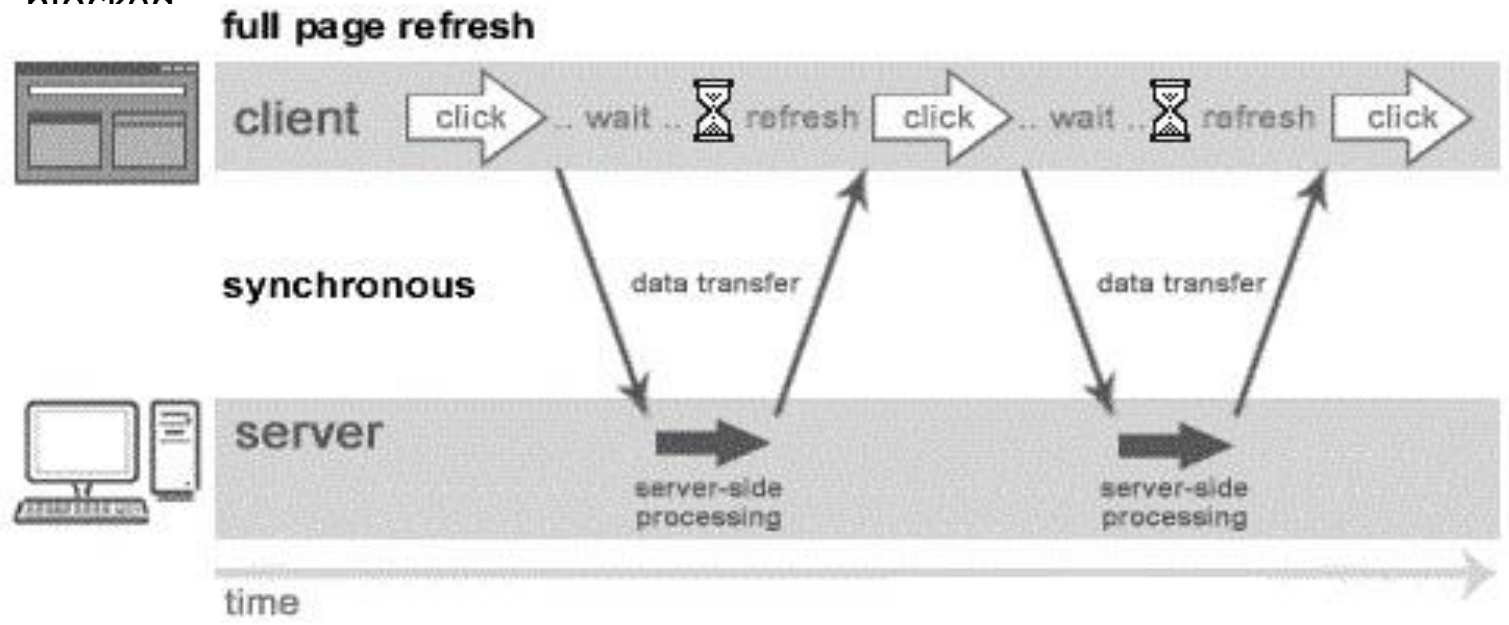
There are too many web applications running on the web that are using ajax technology like **gmail**, **facebook**,**twitter**, **google map**, **youtube** etc.

Understanding Synchronous vs Asynchronous

Before understanding AJAX, let's understand classic web application model and ajax web application model first.

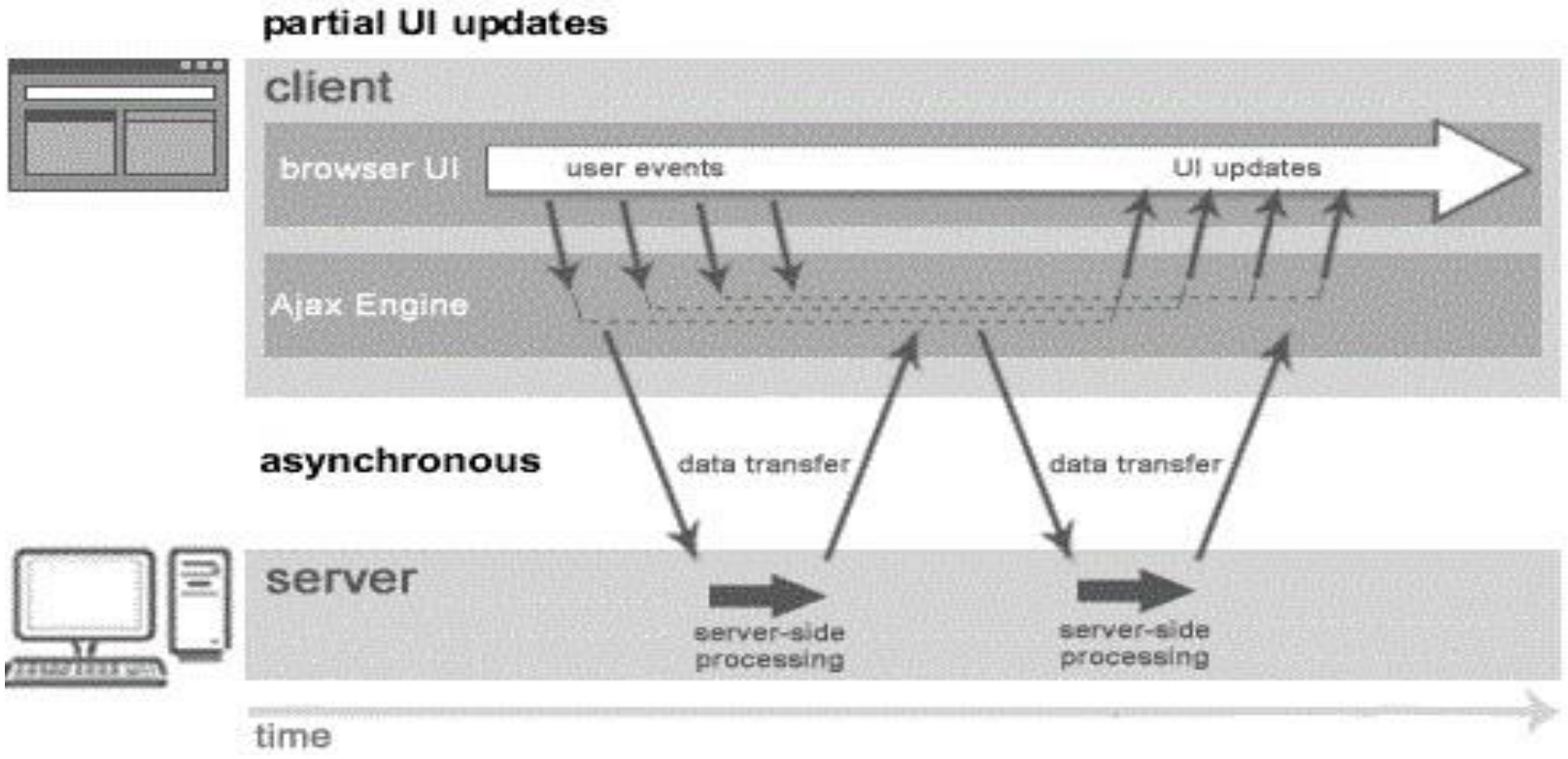
Synchronous (Classic Web-Application Model)

A synchronous request blocks the client until operation completes i.e. browser is unresponsive. In such case, javascript engine of the browser is blocked



Asynchronous (AJAX Web-Application Model)

An asynchronous request doesn't block the client i.e. browser is responsive. At that time, user can perform another operations also. In such case, javascript engine of the browser is not blocked.



AJAX Technologies

HTML/XHTML and CSS

These technologies are used for displaying content and style. It is mainly used for presentation.

DOM

It is used for dynamic display and interaction with data.

XML or JSON

For carrying data to and from server. JSON (Javascript Object Notation) is like XML but short and faster than XML.

XMLHttpRequest

For asynchronous communication between client and server.

JavaScript

It is used to bring above technologies together.
Independently, it is used mainly for client-side validation.

Understanding XMLHttpRequest

An object of XMLHttpRequest is used for asynchronous communication between client and server.

It performs following operations:

- Sends data from the client in the background
- Receives the data from the server
- Updates the webpage without reloading it.

Properties of XMLHttpRequest object

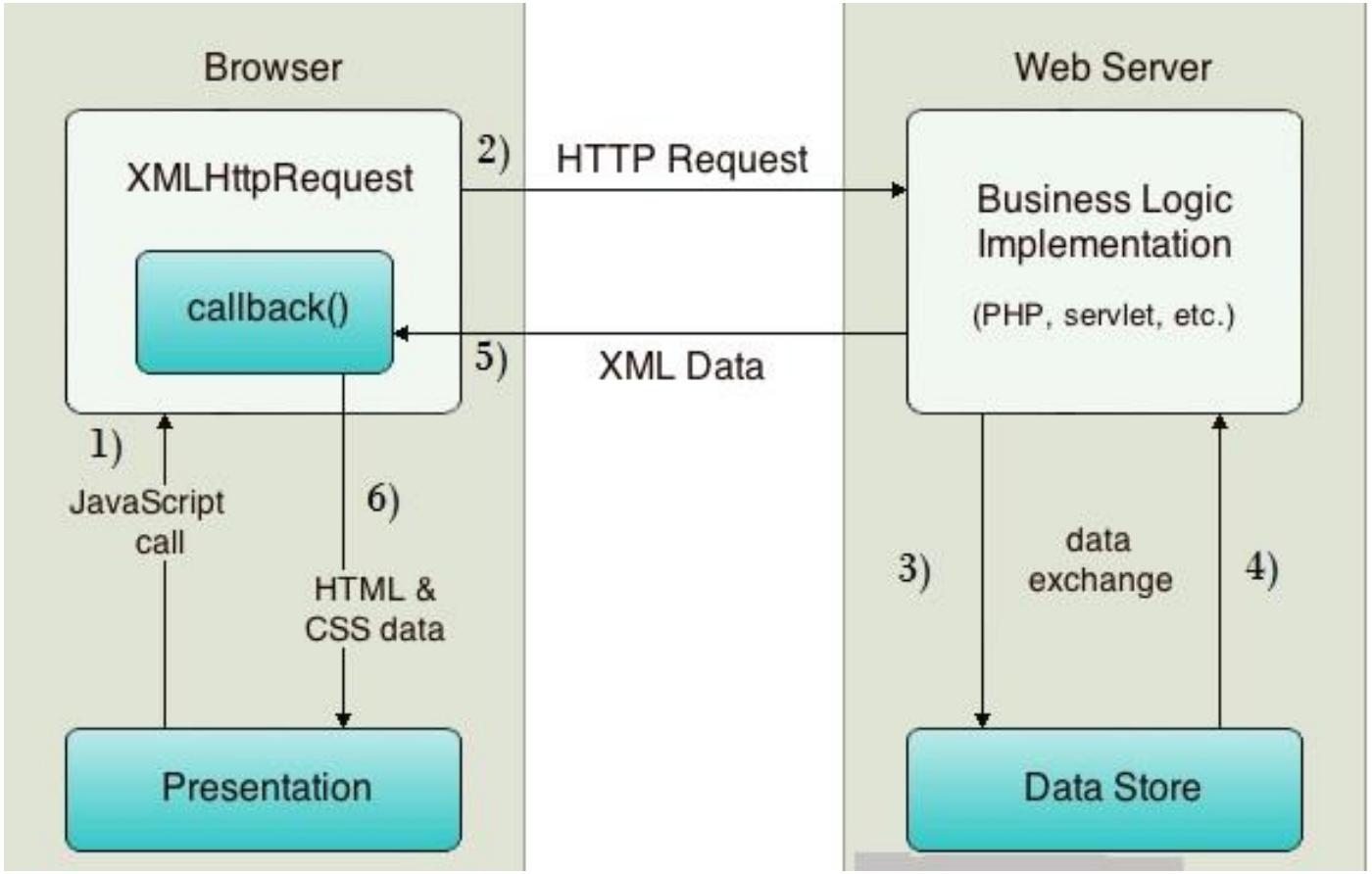
Property	Description
onReadyStateChange	It is called whenever readystate attribute changes. It must not be used with synchronous requests.
readyState	represents the state of the request. It ranges from 0 to 4.**0** UNOPENED open() is not called. **1** OPENED open is called but send() is not called. **2** HEADERS_RECEIVED send() is called, and headers and status are available. **3** LOADING Downloading data; responseText holds the data. **4** DONE The operation is completed fully.
reponseText	returns response as text.
responseXML	returns response as XML

Methods of XMLHttpRequest object

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

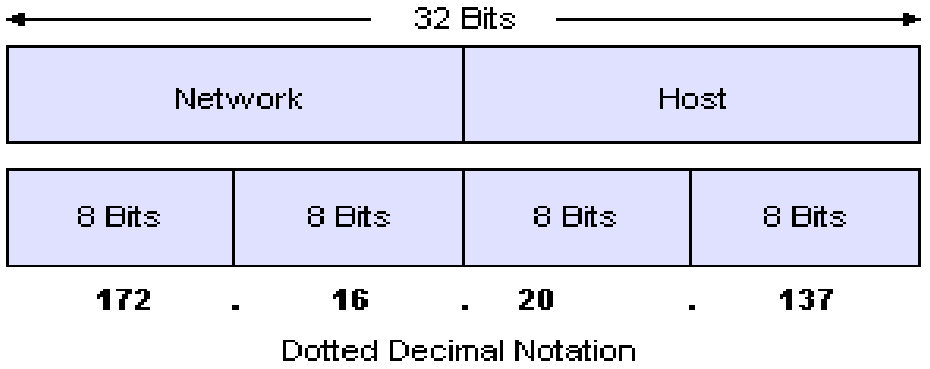
How AJAX works?

AJAX communicates with the server using XMLHttpRequest object. Let's try to understand the flow of ajax or how ajax works by the image displayed below.



Internet Addressing

Each host on a TCP/IP network is assigned a unique 32-bit IP address consisting of a *network number* and a *host number*. The network number identifies a specific network, and must be assigned by the *Internet Network Information Center* (InterNIC) or an accredited registrar. An Internet Service Provider (ISP) can obtain blocks of network addresses from InterNIC and can assign addresses as necessary. The host number identifies a host on a network and is assigned by the network administrator. The IP address is grouped into four binary *octets* (an octet is a group of eight bits) and is represented using *dotted decimal notation*. The minimum value for an octet is 0, and the maximum value is 255. The basic format is illustrated below.



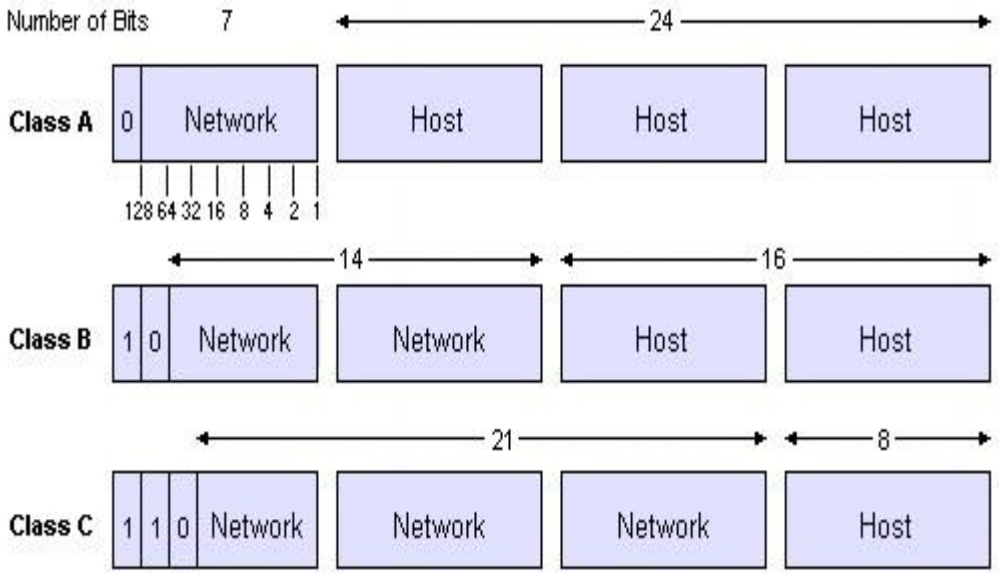
IP addressing supports five address classes - A, B, C, D, and E, of which only classes A, B, and C are available for commercial use. The following table illustrates the IP address classes.

IP Address Classes

Class	Format	Purpose	Address range	Max hosts
A	N.H.H.H	A few large organisations	1.0.0.0 - 126.0.0.0	16,777,214
B	N.N.H.H	Medium-size organisations	128.1.0.0 - 191.254.0.0	65,534
C	N.N.N.H	Relatively small organisations	192.0.1.0 - 223.255.254.0	254
D	N/A	Multicast groups (RFC 1112)	224.0.0.0 - 239.255.255.255	N/A
E	N/A	Experimental	240.0.0.0 - 254.255.255.255	N/A

(N = Network number, H = Host number)

The diagram below illustrates the format of the commercial IP address classes.



The format of the commercial IP address classes

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 *for example* 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. There are two types of addresses: Unicast and Multicast. The Unicast is an identifier for a single interface whereas Multicast is an identifier for a set of interfaces.

- Moreover, InetAddress has a cache mechanism to store successful and unsuccessful host name resolutions.

IP Address

An IP address helps to identify a specific resource on the network using a numerical representation.

Most networks combine IP with TCP (Transmission Control Protocol). It builds a virtual bridge among the destination and the source.

There are two versions of IP address:

1. IPv4

IPv4 is the primary Internet protocol. It is the first version of IP deployed for production in the ARAPNET in 1983. It is a widely used IP version to differentiate devices on network using an addressing scheme. A 32-bit addressing scheme is used to store 2^{32} addresses that is more than 4 million addresses.

Features of IPv4:

It is a connectionless protocol.
It utilizes less memory and the addresses can be remembered easily with the class based addressing scheme.
It also offers video conferencing and libraries.

2. IPv6

IPv6 is the latest version of Internet protocol. It aims at fulfilling the need of more internet addresses. It provides solutions for the problems present in IPv4. It provides 128-bit address space that can be used to form a network of 340 undecillion unique IP addresses. IPv6 is also identified with a name IPng (Internet Protocol next generation).

Features of IPv6:

It has a stateful and stateless both configurations.
It provides support for quality of service (QoS).
It has a hierarchical addressing and routing infrastructure.

Factory Methods

The **InetAddress** class is used to encapsulate both the numerical IP address and 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.

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 whereby static methods in a class return an instance of that class. Three commonly used InetAddress factory methods are shown here:

static InetAddress **getLocalHost**( )
throws UnknownHostException
static InetAddress **getByName**(**String** hostName)
throws UnknownHostException
static InetAddress[ ] **getAllByName**(**String** hostName)
throws UnknownHostException

The **getLocalHost**() method simply returns the InetAddress object that represents the localhost. The **getByName**() method returns an InetAddress for a hostname passed to it. If these methods are unable to resolve the hostname, they throw an UnknownHostException.

The **getAllByName**() factory method returns an array of InetAddresses that represent all of the addresses that a particular name resolves to. It will also throw an UnknownHostException if it can't resolve the name to at least one address. InetAddress also includes the factory method **getByAddress**(), which takes an IP address and returns an InetAddress object.

```
import java.net.*;
class InetAddressTest
{
public static void main(String args[]) throws UnknownHostException
{
InetAddress Address = InetAddress.getLocalHost();
System.out.println(Address);
Address = InetAddress.getByName("webeduclick.com");
System.out.println(Address);
InetAddress SW[] = InetAddress.getAllByName("www.forbes.com");
for (int i=0; i<SW.length; i++)
System.out.println(SW[i]);
}}
```

What is Socket Programming in Java?

- The server forms the listener *socket while* the client reaches out to the server. Socket and ServerSocket classes are used for connection-oriented socket programming.
- Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.
- The client in socket programming must know two information:
  - IP Address of Server
  - Port number

Creation of Sockets

- The creation of a Socket object implicitly establishes a connection between the client and server.
- There are no methods or constructors that explicitly expose the details of establishing that connection.

Socket(String *hostName,* int *port*) throws UnknownHostException,IOException	Creates a socket connected to the named host and port.
Socket(InetAddress *ipAddress,* int *port*) throws IOException	Creates a socket using a preexisting InetAddress object and a port.

What is a Server Socket in Java?

- The **ServerSocket** class is used to create servers that listen for either local or remote client programs to connect to them on published ports.
- **ServerSocket**s are quite different from normal **Socket**s.
- When we create a **ServerSocket,** it will register itself with the system as having an interest in client connections.
- The constructors for **ServerSocket** reflect the port number that we want to accept connections on and, optionally, how long we want the queue for said port to be.

TCP/IP Client Socket

In Java, **TCP/IP Client Socket** used to implement reliable, bidirectional, persistent, point-to-point, stream-based connections between hosts on the Internet. A socket can be used to connect Java's I/O system to other programs that may reside either on the local machine or on any other machine on the Internet.

There are two kinds of TCP/IP Client Socket in Java. One is for **servers**, and the other is for **clients**. The ServerSocket class designed to be a "listener," which waits for clients to connect before doing anything. Thus, ServerSocket is for servers. The Socket class is for clients. It designed to connect to server sockets and initiate protocol exchanges.

Constructor to Create TCP/IP Client Socket:

- **Socket(String hostName, int port) throws UnknownHostException, IOException:** Creates a socket connected to the named host and port.

- **Socket(InetAddress ipAddress, int port) throws IOException:** Creates a socket using a preexisting InetAddress object and a port.

Instance Methods of TCP/IP Client Socket:

InetAddress getInetAddress(): Returns the InetAddress associated with the Socket object. It returns null if the socket is not connected.

int getPort(): It returns the remote port to which the invoking Socket object is connected. It returns 0 if the socket is not connected.

int getLocalPort(): It returns the local port to which the invoking Socket object is bound. It returns –1 if the socket is not bound

I/O Stream of TCP/IP Client Socket:

InputStream getInputStream( ) throws IOException: Returns the InputStream associated with the invoking socket.

OutputStream getOutputStream( ) throws IOException: Returns the OutputStream associated with the invoking socket.

Demonstrate TCP/IP Client Socket Program:

```
import java.net.*;
import java.io.*;
class Whois
{
public static void main(String args[]) throws Exception
{
int c;
Socket s = new Socket("internic.net", 43);
InputStream in = s.getInputStream();
OutputStream out = s.getOutputStream();
String str = (args.length == 0 ? "osborne.com" : args[0]) + "\n";
byte buf[] = str.getBytes();
out.write(buf);
while ((c = in.read()) != -1) {
System.out.print((char) c);
}
s.close();
}
```

TCP/IP Server Socket:

Java has a different socket class that must be used for creating server applications. The **TCP/IP Server Socket** class is used to create servers that listen for either local or remote client programs to connect to them on published ports. ServerSockets are quite different from normal Sockets.

When you create a ServerSocket, it will register itself with the system as having an interest in client connections. The queue length tells the system how many client connections it can leave pending before it should simply refuse connections. The default is 50.

Constructor to Create TCP/IP Server Socket:

ServerSocket(int port) throws IOException: Creates server socket on the specified port with a queue length of 50.

ServerSocket(int port, int maxQueue) throws IOException: Creates a server socket on the specified port with a maximum queue length of the max queue.

ServerSocket(int port, int maxQueue, InetAddress local address) throws IOException: Creates a server socket on the specified port with a maximum queue length of the max queue. On a multihomed host, the local address specifies the IP address to which this socket binds.

URL (Uniform Resource Locator)

- A URL takes the form of a string that describes how to find a resource on the Internet.
- URLs have two main components:
  - the protocol needed to access the resource and
  - the location of the resource.
- Within our Java programs, we can create a URL object that represents a URL address.
- The URL object always refers to an absolute URL but can be constructed from an absolute URL, a relative URL, or from URL components.

URL Class

- Java URL class mainly deals with URL(Uniform Resource Locator) which is used to identify the resources on the internet.
- For Example: https://www.google.com/search
- Here,
  - https: Protocol
  - www.google.com: hostname
  - /search: filename

URL Connection

- URLConnection is a general-purpose class for accessing the attributes of a remote resource.
- Once we make a connection to a remote server, we can use URLConnection to inspect the properties of the remote object before actually transporting it locally.
- These attributes are exposed by the HTTP protocol specification and, as such, only make sense for URL objects that are using the HTTP protocol.

The URI Class

- This class provides constructors for creating URI instances from their components or by parsing their string forms, methods for accessing the various components of an instance, and methods for normalizing, resolving, and relativizing URI instances. Instances of this class are immutable.
- The URI class encapsulates a Uniform Resource Identifier (URI).
- URIs are similar to URLs.
- In fact, URLs constitute a subset of URIs.
- A URI represents a standard way to identify a resource.
- A URL also describes how to access the resource.

URLConnection Class in Java

URL:

The **URL** provides a reasonably intelligible form to uniquely identify or address information on the Internet. URLs are ubiquitous, every browser uses them to identify information on the Web. Within Java's network class library, the URL class provides a simple, concise API to access information across the Internet using URLs.

A URL specification is based on four components. The first is the protocol to use, separated from the rest of the locator by a colon (:). Common protocols are HTTP, FTP, gopher, and file, although these days almost everything is being done via HTTP. The second component is the hostname 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 (:). The third component, the port number, is an optional parameter, delimited on the left from the hostname by a colon (:) and on the right by a slash (/). (It defaults to port 80, the predefined HTTP port; thus, ":80" is redundant.) The fourth part is the actual file path. Most HTTP servers will append a file named index.html or index.htm to URLs that refer directly to a directory resource.

Java's URL class has several constructors; each can throw a MalformedURLException. One commonly used form specifies the URL with a string that is identical to what you see displayed in a browser:

```
URL(String urlSpecifier) throws MalformedURLException
```

The next two forms of the constructor allow you to break up the URL into its component parts:

```
URL(String protocolName, String hostName, int port, String path)
throws MalformedURLException
URL(String protocolName, String hostName, String path)
throws MalformedURLException
```

URLConnection:

It is a general-purpose class for accessing the attributes of a remote resource. Once you make a connection to a remote server, you can use URLConnection to inspect the properties of the remote object before actually transporting it locally. These attributes are exposed by the HTTP protocol specification and, as such, only make sense for URL objects that are using the HTTP protocol.

URLConnection Methods:

int getContentLength(): Returns the size in bytes of the content associated with the resource. If the length is unavailable, –1 is returned.

String getContentType(): Returns the type of content found in the resource. This is the value of the content-type header field.
long getDate(): Returns the time and date of the response represented in terms of milliseconds.

long getExpiration(): Returns the expiration time and date of the resource represented in terms of milliseconds

String getHeaderField(int idx): Returns the value of the header field at index idx. Returns null if the value of idx exceeds the number of fields.

String getHeaderField(String fieldName): Returns the value of header field whose name is specified by fieldName. Returns null if the specified name is not found.

String getHeaderFieldKey(int idx): Returns the header field key at index idx. Returns null if the value of idx exceeds the number of fields.

Map<String, List<String>> getHeaderFields(): Returns a map that contains all of the header fields and values.

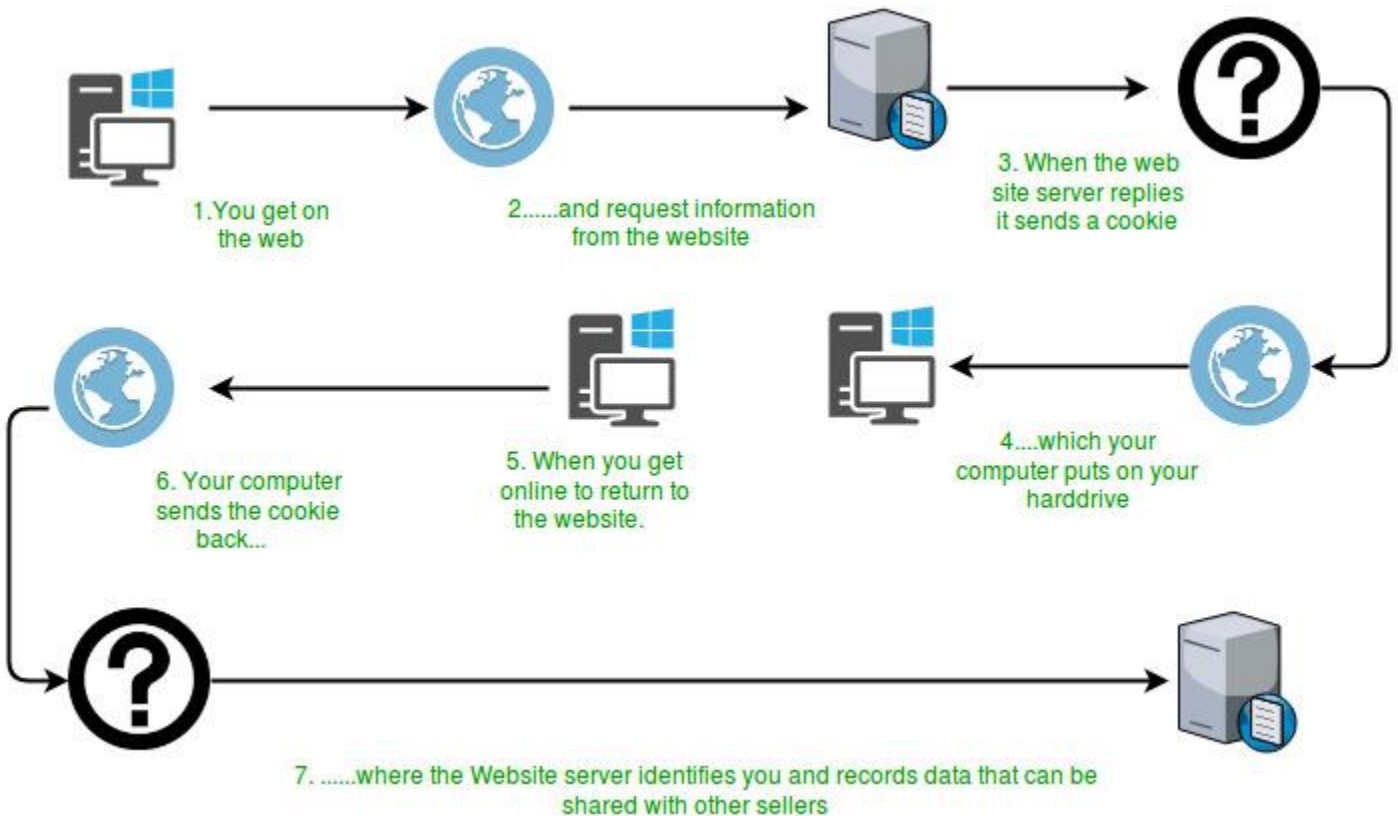
long getLastModified(): Returns the time and date, represented in terms of milliseconds.

InputStream getInputStream() throws IOException: Returns an InputStream that is linked to the resource. This stream can be used to obtain the content of the resource

Cookies

- Cookies can be used by a server to indicate session IDs, shopping cart contents, login credentials, user preferences, and more.
- The **java.net** package includes classes and interfaces that help manage cookies and can be used to create a stateful (as opposed to stateless) HTTP session.
- The classes are **CookieHandler, CookieManager,** and **HttpCookie.**
- The interfaces are **CookiePolicy** and **CookieStore.**

How Cookies work?



Cookies - constructor

Syntax

public Cookie(String name, String value)

Parameters :

name : name of the cookie

value : value associated with this cookie

Datagrams

- *Datagrams* are bundles of information passed between machines.
- Java implements datagrams 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 **DatagramPacket**s.

DatagramSocket

- **DatagramSocket** defines many methods. Two of the most important are **send( )** and **receive( ).**

```
void send(DatagramPacket packet) throws IOException
void receive(DatagramPacket packet) throws IOException
```

- The **send( )** method sends a packet to the port specified by *packet*. The **receive( )** method waits for a packet to be received and returns the result.
- **DatagramSocket** also defines the **close( )**method, which closes the socket.

DatagramPacket

DatagramPacket(byte *data* [ ], int *size*)
DatagramPacket(byte *data* [ ], int *offset*, int *size*)
DatagramPacket(byte *data* [ ], int *size*, InetAddress *ipAddress*, int *port*)
DatagramPacket(byte *data* [ ], int *offset*, int *size*, InetAddress *ipAddress*, int *port*)