AJAX

AJAX is an acronym for **Asynchronous JavaScript and XML**. It is a group of inter-related technologies ike JavaScript, DOM, XML, HTML, CSS 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 lata from the client side is routed to the server side. It makes your application interactive and faster.

Ajax Applications (Where it is used)

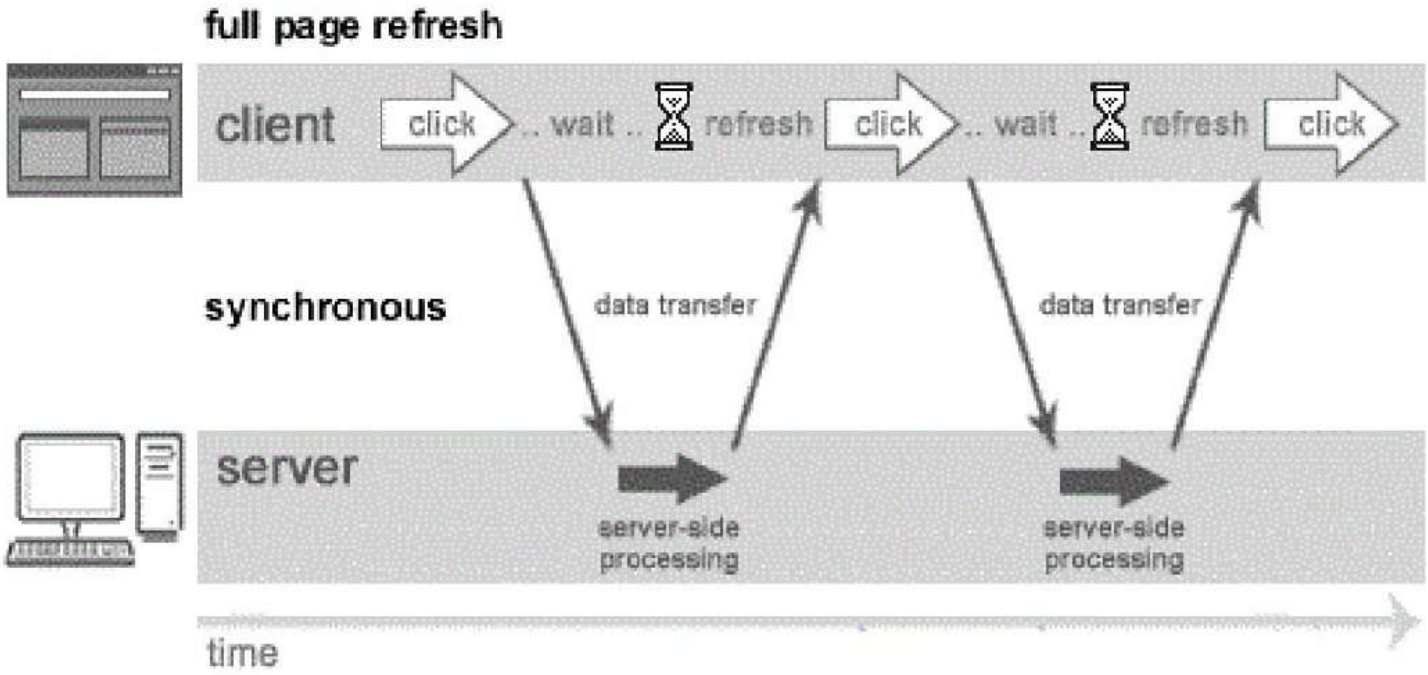
There are too many web applications running on the web that are using ajax technology like **gmail**, **acebook**,**twitter**, **google map**, **youtube** etc.

Synchronous vs Asynchronous

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

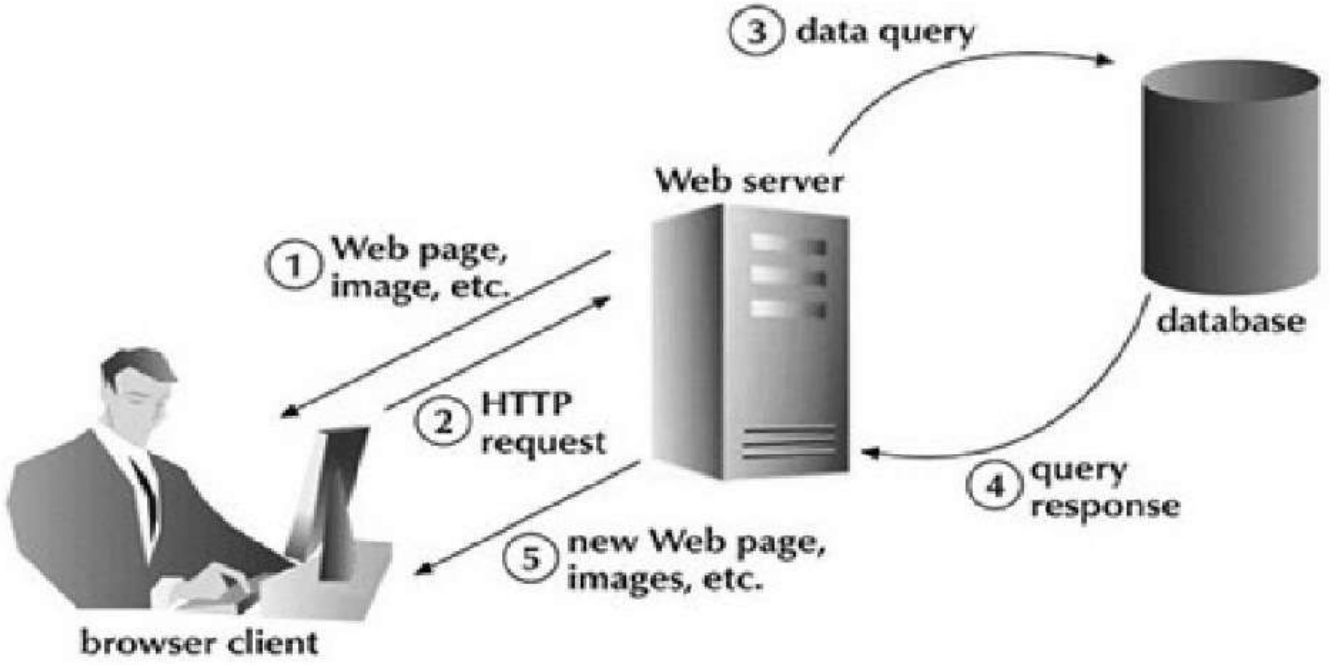
Synchronous (Classic Web-Application Model)

A synchronous request blocks the client until operation completes i.e. browser is not unresponsive. In such ase, javascript engine of the browser is blocked.



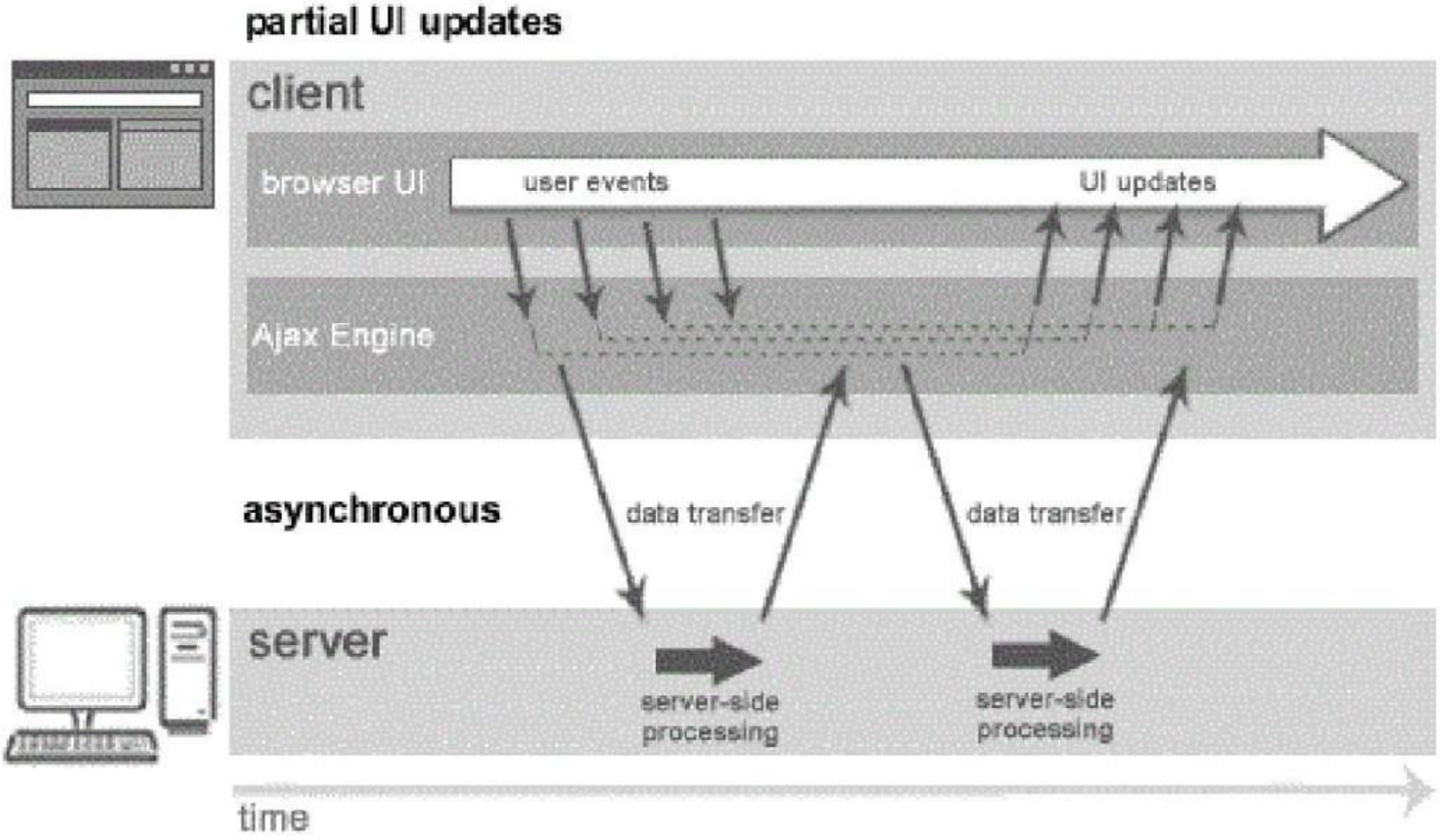
n the above image, full page is refreshed at request time and user is blocked until request completes.

Let's understand it another way.



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.



In the above image, full page is not refreshed at request time and user gets response from the ajax engine.

Let's try to understand asynchronous communication by the image given below.

Note: every blocking operation is not synchronous and every unblocking operation is not asynchronous.

AJAX Technologies

Ajax is not a technology but group of inter-related technologies. AJAX technologies includes:

- HTML/XHTML and CSS
- DOM
- XML or JSON
- XMLHttpRequest
- JavaScript

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.

Ajax Objects

1. XMLHttpRequest Object

All modern browsers support the XMLHttpRequest object.

The XMLHttpRequest object can be used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

Create an XMLHttpRequest Object

All modern browsers (Chrome, Firefox, IE7+, Edge, Safari Opera) have a built-in XMLHttpRequest object.

Syntax for creating an XMLHttpRequest object:

***var variablename* = new XMLHttpRequest();**

XMLHttpRequest Object Methods

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
open(*method,url,async,user,psw*)	Specifies the request *method*: the request type GET or POST *url*: the file location *async*: true (asynchronous) or false (synchronous) *user*: optional user name *psw*: optional password
send()	Sends the request to the server Used for GET requests
send(*string*)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

XMLHttpRequest Object Properties

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data

status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found"
statusText	Returns the status-text (e.g. "OK" or "Not Found")

How to Send a Request to Server

The XMLHttpRequest object is used to exchange data with a server.

Send a Request To a Server

To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:

```
xhttp.open("GET", "ajaxfirst.txt", true);

xhttp.send();
```

Server Response

The onreadystatechange Property

The **readyState** property holds the status of the XMLHttpRequest.

The **onreadystatechange** property defines a function to be executed when the readyState changes.

The **status** property and the **statusText** property holds the status of the XMLHttpRequest object.

The onreadystatechange function is called every time the readyState changes.

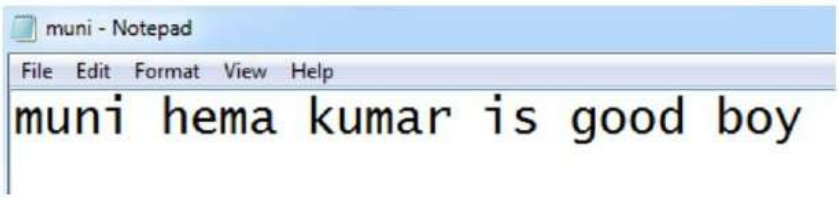
When readyState is 4 and status is 200, the response is ready:

```
xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
```

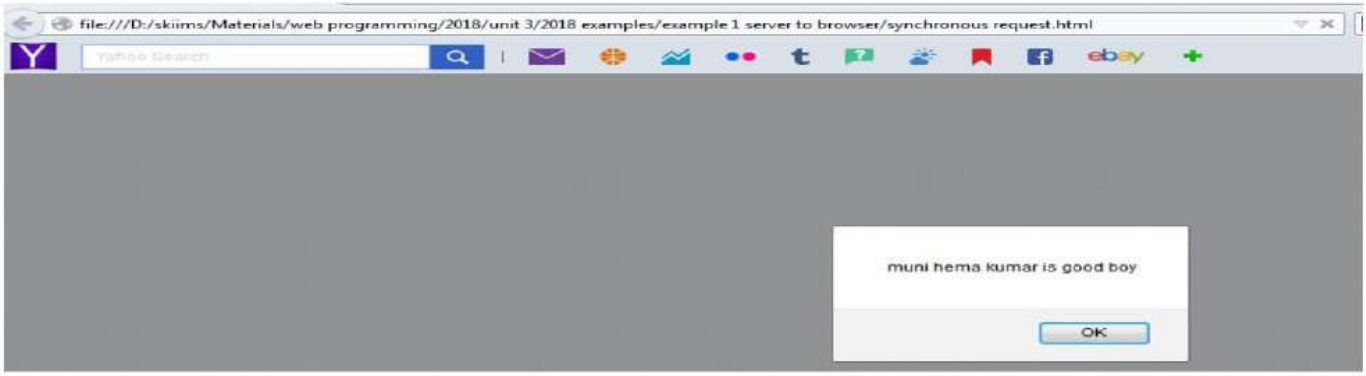
Synchronous Request(Mode):

This is not traditional mode we cannot get updated information without refreshing.

We must wait for response to do other java script code.
Step1: create xmlhttp request object
Step 2: call open method
Step 3: send request
Step 4: display responseText
synchronous.html

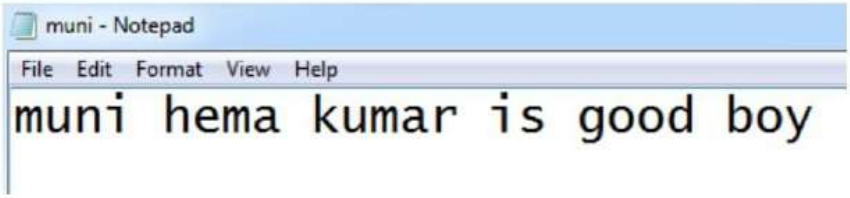


```
<script type="text/javascript">
var req=new XMLHttpRequest();
req.open("GET","muni.txt",false);
req.send();
alert(req.responseText);
</script>
```



Asynchronous Request(Mode):

This is traditional mode we can get updated information without refreshing.
In the same time we can do other java script codes.
Steps
Create XMLHttpRequest object
Call open method set the boolean value as true
onreadystatechange=function()
{
check ready state
if true display response text;
};
Call send method;



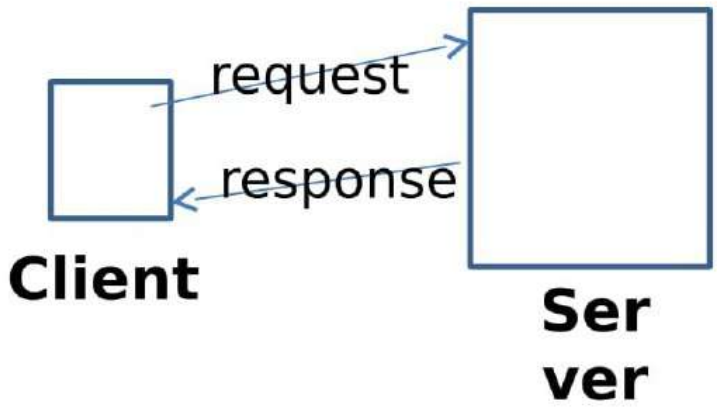
asynchronous.html

```
<script type="text/javascript">
function muni()
{
var req=new XMLHttpRequest();
req.open("GET","muni.txt",true);
req.onreadystatechange=function()
{
 if(req.readyState==4)
 {
  document.getElementById("ch").innerHTML=req.responseText;
 }
};
req.send();
```

```
}
</script>
<div id="ch">Hi students</div>
<button onclick="muni();">click to change</button>
```

2. JSON object:

- Java Script Object Notation
- It is a format to transfer data from server to client or client to server.
- The technology is used in Server the technology may be php, .net. Java it's not matter.
- What matter is From the server what you returned to client side would be data.
- You have design at client side but don't have data.
- The data coming from server.
- When client send request to server you will get response in some format from the server.
- I don't want html (it returns design page), I want data.



JSON Methods

1. JSON.parse():

A common use of JSON is to exchange data to/from a web server.

When receiving data from a web server, the data is always a string.

Parse the data with JSON.parse(), and the data becomes a JavaScript object.

To deserialize a json structure into java script object

Syntax:JSON.parse(xmlhttprequestobject.responseText);

Example:

parsejson.html

```
<script type="text/javascript">
function muni()
{
var req=new XMLHttpRequest();
req.open("GET","munijson.txt",true);
req.onreadystatechange=function()
{
 if(req.readyState==4)
 {
  var rss=JSON.parse(req.responseText);
  document.getElementById("ch").innerHTML=rss.name;
 }
};
req.send();
}
</script>
<div id="ch">Hi students</div>
<button onclick="muni();">click to change</button>
```

muni.txt

```
{ "name":"muni", "age":31, "city":"Tirupati" }
```

AJAX LIBRARIES

1. Dojo Toolkit 1.12

A JavaScript toolkit that saves you time and scales with your development process. Provides everything you need to build a Web app. Language utilities, UI components, and more, all in one place, designed to work together perfectly.

```
<script src="//ajax.googleapis.com/ajax/libs/dojo/1.12.1/dojo/dojo.js"></script>
```

someObect.someMethod();

Sending Data to the Server Using GET

```
<script type="dojo/method" data-dojo-event="onClick">
 dojo.xhrPost({
    url: 'HelloWorldResponsePOST.php',
    load: helloCallback,
    error: helloError,
    form: 'myForm'
 });
</script>
```

Reading Data from the Server

```
<script type="dojo/method" data-dojo-event="onClick">
 dojo.xhrGet({
    url: 'response.txt',
    load: helloCallback,
    error: helloError
 });
</script>
```

2. Prototype:

Prototype is a JavaScript library aims to ease the development of dynamic web applications. Prototype was developed by Sam Stephenson.

Prototype enables you to deal with Ajax calls in a very easy and fun way that is also safe (cross-browser).

```
<script type = "text/javascript" src = "/javascript/prototype.js"></script>
new Ajax.Request(url[, options]);
```

How to send data

```
function ajax_test()
{
        var url = "test_ajax_functions.php";
        var data = "something=nothing";
        var target = 'output';
        function ajax_response(resp)
        {
                alert(resp.responseText);
        }

        var myAjax = new Ajax.Updater(
                target,
                url,
```

```
            {method: 'post', parameters: data, onComplete: ajax_response}
        );
}
```

How to receive data

```
function SubmitRequest()
{
        new Ajax.Request('url', {
          method: 'get',
          onSuccess: successFunc,
          onFailure:  failureFunc
        });
      }
```

3. MooTools

MooTools is a collection of JavaScript utilities designed for the intermediate to advanced JavaScript developer. It allows you to write powerful and flexible code with its elegant, well documented, and coherent APIs.

MooTools code is extensively documented and easy to read, enabling you to extend the functionality to match your requirements.

```
var myRequest = new Request([options]);
```

Receiving data

```
 var myRequest = new Request({
            url: 'truncate.php',
            method: 'post',
            onRequest: function(){
            },
            onSuccess: function(responseText){
                alert("done!"+ responseText);
            },
            onFailure: function(){
                alert("failed");
            }
        });
```

Sending data

```
var myRequest = new Request({
      url: 'truncate.php',
      method: 'post',
      onRequest: function() {
      },
      onSuccess: function(responseText) {
        alert("done! " + responseText);
      },
      onFailure: function() {
        alert("failed");
      }
   });
```

4. Spry

The **Spry** is an open source Ajax library developed by Adobe Systems which is used in the construction of Rich Internet applications.[1] Unlike other pure JavaScript frameworks such as the Dojo Toolkit and Prototype, Spry is geared towards web designers, not web developers.

The Spry broadly consists of

- Spry Effects - animation effects like blind, fade, grow, highlight, shake, slide and squish.
- Spry Data - data binding to HTML markup using minimal code or proprietary markup. Spry uses Google's Xpath JavaScript library to convert XML into JavaScript objects. It can handle XML, HTML and JSON data.
- Spry Widgets - framework for development of widgets, and included widgets such as the accordion.

Usage

- Ajax development within an IDE such as Eclipse (software).
- Ajax generation from server code using ColdFusion. Ruby on Rails offers similar functionality.
- Ajax application generation from Adobe Flex code. OpenLaszlo will offer similar functionality with their "Legals" release (version 4).

5. YUI Library

The Yahoo! User Interface Library (**YUI**) is a discontinued open-source JavaScript library for building richly interactive web applications using techniques such as Ajax, DHTML, and DOM scripting. YUI includes several core CSS resources. Development on YUI began in 2005 and Yahoo! properties such as My Yahoo! and the Yahoo! front page began using YUI in the summer of that year. YUI was released for public use in February 2006.[1] It was actively developed by a core team of Yahoo! engineers.

Features

Core: The YUI Core is a light (31KB minified) set of tools for event management and DOM manipulation.
YUI Global Object: The YUI Global Object contains language utilities, a script loader, and other baseline infrastructure for YUI.

Dom Collection: Helps with common DOM scripting tasks, including element positioning and CSS style management.

Event Utility: Provides developers with easy and safe access to browser events (such as mouse clicks and key presses). It also provides the Custom Event object for publishing and subscribing to custom events.

Server:

- You want to print a file
- You will raise a request
- And for that request to some sort of software which can handle your request and print the document as per your demand.
- Piece software here print server.
- A server is piece of software that resides on a particular network and takes your request and do something according to that request.

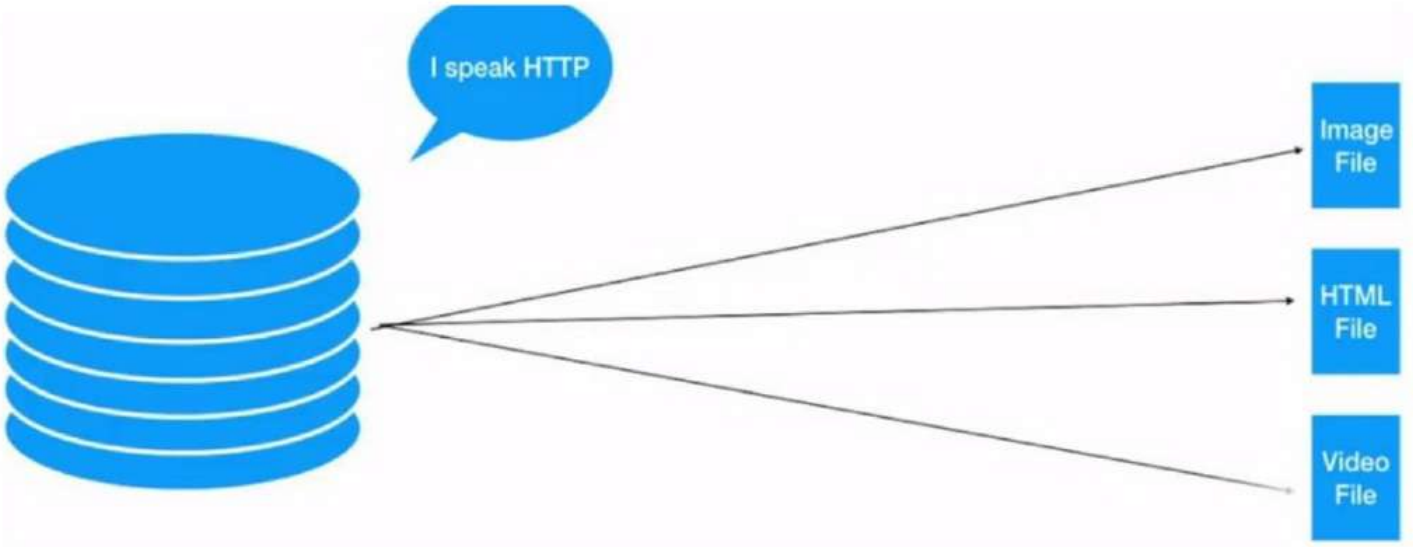
Web Server:

A **Web server** can be either a computer program or a computer running a program that is responsible for accepting HTTP requests from clients, serving back HTTP responses along with optional data contents, which usually are web pages such as HTML documents and linked objects on it.

Is all about dealing with the http content of content which can be handled over the http protocol.

A Web server is a system that delivers content or services to end users over the Internet. A Web server consists of a physical server, server operating system (OS) and software used to facilitate HTTP communication.

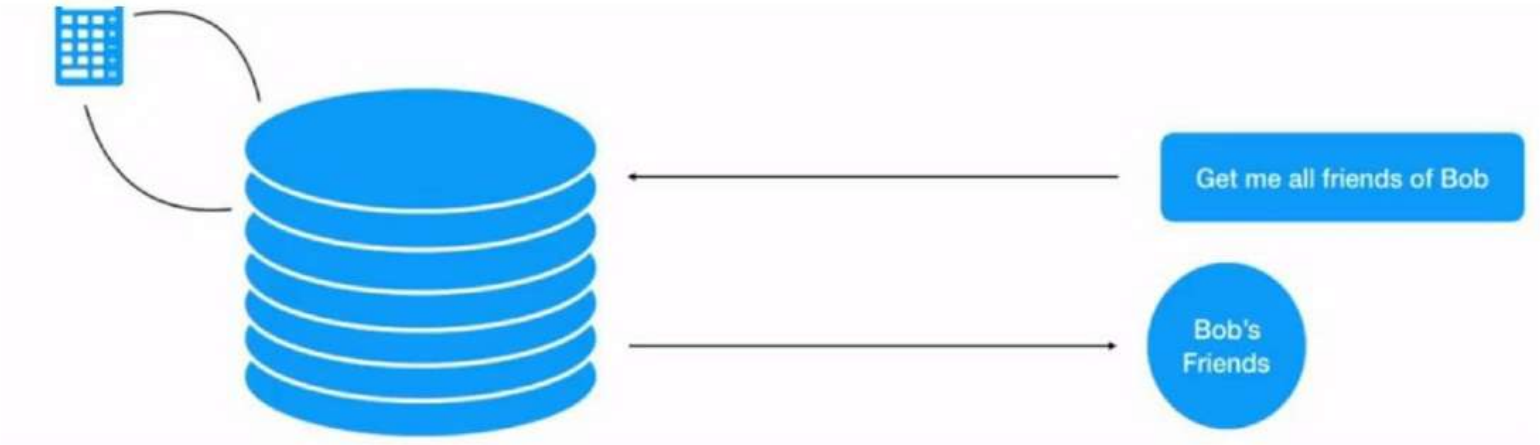
A Web server is a computer system that hosts websites. It runs Web server software, such as Apache or Microsoft IIS, which provides access to hosted webpages over the Internet.



Application Server:

An application server is the kind of software engine that will deliver various applications to another device. It is the kind of computer found in an office or university network that allows everyone in the network to run software off of the same machine.

Is all about executing the programming logic and produce the output according to application logic.

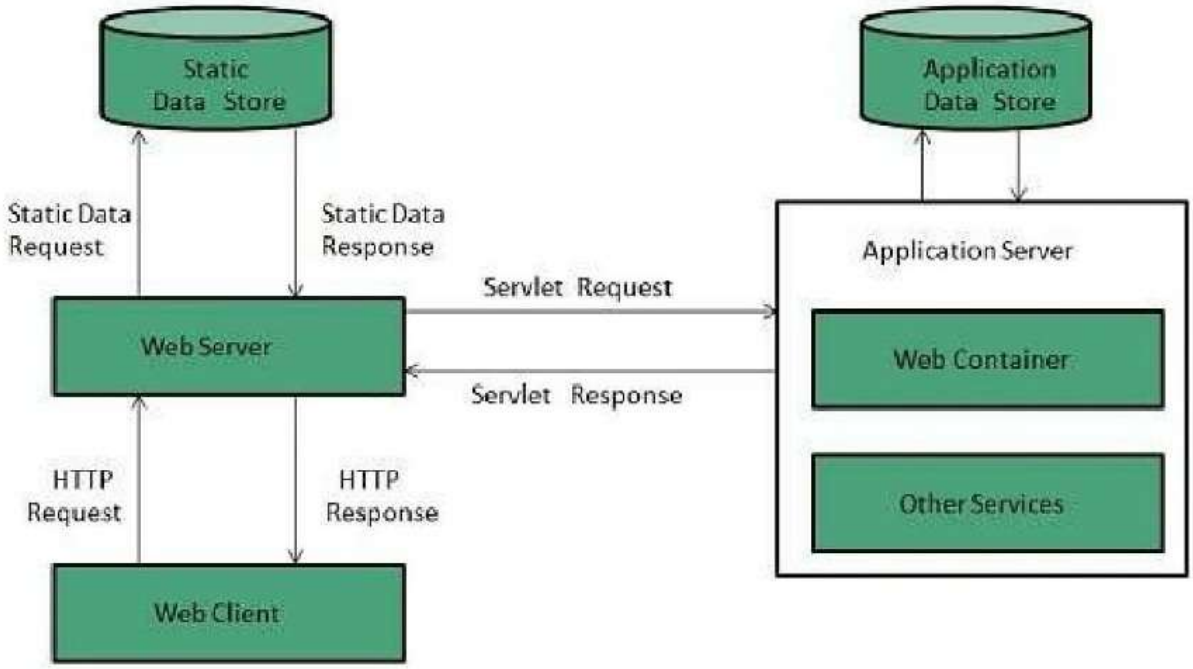


Web Server	Application Server
I have web application to run the web application we require web server.	**To run enterprise application we require application server.**
Web server provides environment to run web applications.	Application server provides environment to run enterprise applications.
Web server can provide support only for web related technologies	Application server can provide support any technologies

Web Server Working

Web server respond to the client request in either of the following two ways:

- Sending the file to the client associated with the requested URL.
- Generating response by invoking a script and communicating with database



- When client sends request for a web page, the web server search for the requested page if requested page is found then it will send it to client with an HTTP response.
- If the requested web page is not found, web server will the send an **HTTP response:Error 404 Not found.**
- If client has requested for some other resources then the web server will contact to the application server and data store to construct the HTTP response.

Server Architecture

Web Server Architecture follows the following two approaches:

1. Concurrent Approach
2. Single-Process-Event-Driven Approach.

Concurrent Approach

Concurrent approach allows the web server to handle multiple client requests at the same time. It can be achieved by following methods:

- Multi-process
- Multi-threaded
- Hybrid method.

Multi-processing

In this a single process (parent process) initiates several single-threaded child processes and distribute incoming requests to these child processes. Each of the child processes are responsible for handling single request.

It is the responsibility of parent process to monitor the load and decide if processes should be killed or forked.

Multi-threaded

Unlike Multi-process, it creates multiple single-threaded process.

Hybrid

It is combination of above two approaches. In this approach multiple process are created and each process initiates multiple threads. Each of the threads handles one connection. Using multiple threads in single process results in less load on system resources.

IIS(Internet Information Services/Server)

Stands for "Internet Information Services." IIS is a web server software package designed for Windows Server. It is used for hosting websites and other content on the Web.

Features:

- Microsoft's Internet Information Services provides a graphical user interface (GUI) for managing websites and the associated users.
- It provides a visual means of creating, configuring, and publishing sites on the web.
- The IIS Manager tool allows web administrators to modify website options, such as default pages, error pages, logging settings, security settings, and performance optimizations.
- IIS can serve both standard HTML webpages and dynamic webpages, such as ASP.NET applications and PHP pages.
- When a visitor accesses a page on a static website, IIS simply sends the HTML and associated images to the user's browser.
- When a page on a dynamic website is accessed, IIS runs any applications and processes any scripts contained in the page, then sends the resulting data to the user's browser.
- While IIS includes all the features necessary to host a website, it also supports extensions (or "modules") that add extra functionality to the server. For example, the WinCache Extension enables PHP scripts to run faster by caching PHP processes.
- The URL Rewrite module allows webmasters to publish pages with friendly URLs that are easier for visitors to type and remember.
- A streaming extension can be installed to provide streaming media to website visitors.
- IIS is a popular option for commercial websites, since it offers many advanced features and is supported by Microsoft.

- However, it also requires requires a commercial license and the pricing increases depending on the number of users. Therefore, Apache HTTP Server, which is open source and free for unlimited users, remains the most popular web server software.

Installing IIS

To install IIS:

1. In Windows, access the Control Panel and click **Add or Remove Programs**.
2. In the Add or Remove Programs window, click **Add/Remove Windows Components**.
3. Select the **Internet Information Services (IIS)** check box, click **Next**, then click **Finish**.
4. To learn how to use IIS, you can view the documentation at http://localhost/iishelp/iis/misc/default.asp.

Configuring IIS

To configure IIS:

1. Right-click the **My Computer** icon on your server computer desktop, and then click **Manage**.
2. In the **Computer Management** dialog box, open the **Services and Applications** node.
3. Click **Internet Information Services**, and then click **Web Sites**.
4. Right-click the **Default Web Site** node to start it, if it is not started already.
5. If a secure Internet connection is required, set up Secure Sockets Layer (SSL).

Apache

Apache is the most widely used web server software. Developed and maintained by Apache Software Foundation, Apache is an open source software available for free. It runs on 67% of all webservers in the world. It is fast, reliable, and secure. It can be highly customized to meet the needs of many different environments by using extensions and modules.

Apache is the most popular web server (after which comes Microsoft's IIS) available. The reasons behind its popularity, to name a few, are:

1. It is free to download and install.

2. It is open source: the source code is visible to anyone and everyone, which basically enables anyone (who can rise up to the challenge) to adjust the code, optimize it, and fix errors and security holes. People can add new features and write new modules.

3. It suits all needs: Apache can be used for small websites of one or two pages, or huge websites of hundreds and thousands of pages, serving millions of regular visitors each month. It can serve both static and dynamic content.

How Apache Works

Apache's main role is all about communication over networks, and it uses the TCP/IP protocol (Transmission Control Protocol/Internet Protocol which allows devices with IP addresses within the same network to communicate with one another).

The TCP/IP protocol is a set of rules that define how clients make requests and how servers respond, and determine how data is transmitted, delivered, received, and acknowledged.

The Apache server is set up to run through configuration files, in which directives are added to control its behavior. In its idle state, Apache listens to the IP addresses identified in its config file (HTTPd.conf). Whenever it receives a request, it analyzes the headers, applies the rules specified for it in the Config file, and takes action.

But one server can host many websites, not just one - though, to the outside world, they seem separate from one another. To achieve this, every one of those websites has to be assigned a different name, even if those all map eventually to the same machine. This is accomplished by using what is known as virtual hosts.

Since IP addresses are difficult to remember, we, as visitors to specific sites, usually type in their respective domain names into the URL address box on our browsers. The browser then connects to a DNS server, which translates the domain names to their IP addresses. The browser then takes the returned IP address and connects to it. The browser also sends a *Host* header with the request so that, if the server is hosting multiple sites, it will know which one to serve back.

For example, typing in www.google.com into your browser's address field might send the following request to the server at that IP address:

```
1    GET / HTTP/1.1

2    Host: www.google.com
```

The first line contains several pieces of information. First, there is the method (in this case it's a GET), the URI, which specifies which page to be retrieved or which program to be run (in this case it's the root directory denoted by the /), and finally there is the HTTP version (which in this case is HTTP 1.1).

HTTP is a request / response stateless protocol.

HTTP is a request / response stateless protocol. It's a set of rules that govern communication between a client and the server. The client (usually but not necessarily a web browser) makes a request, the server sends back a response, and communication stops. The server doesn't look forward for more communication as is the case with other protocols that stay at a waiting state after the request is over.

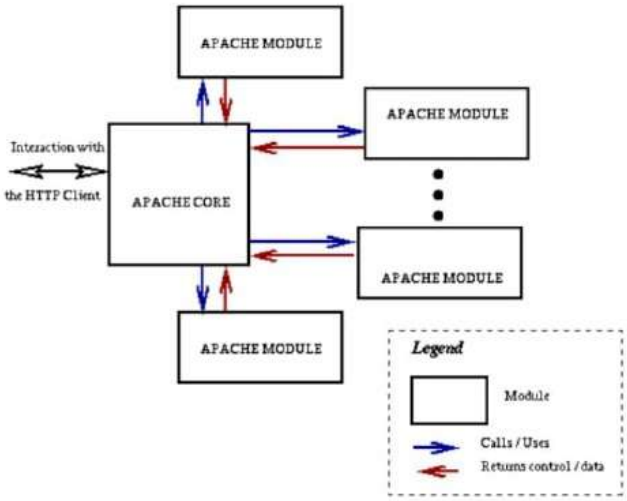
If the request is successful, the server returns a 200 status code (which means that the page is found), response headers, along with the requested data. The response header of an Apache server might look something like the following:

```
01  HTTP/1.1 200 OK
02  Date: Sun, 10 Jun 2012 19:19:21 GMT
03  Server: Apache
04  Expires: Wed, 11 Jan 1984 05:00:00 GMT
05  Cache-Control: no-cache, must-revalidate, max-age=0
06  Pragma: no-cache
07  Last-Modified: Sun, 10 Jun 2012 19:19:21 GMT
08  Vary: Accept-Encoding,User-Agent
09  Content-Type: text/html; charset=UTF-8
10  Content-Length: 7560
```

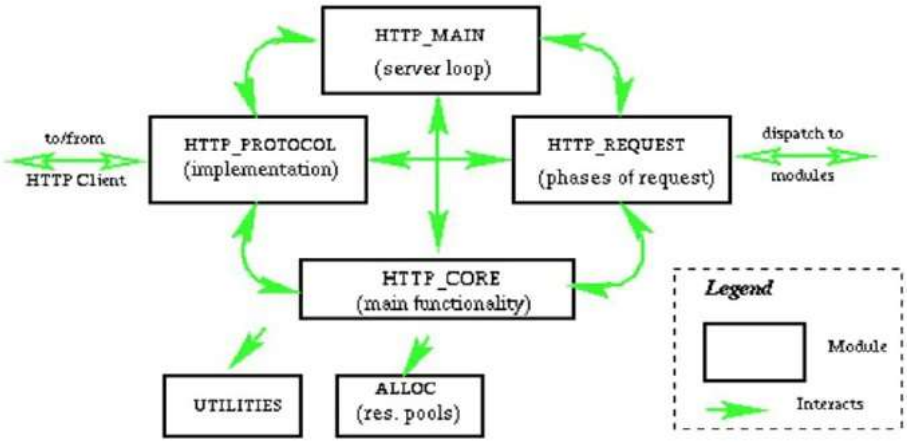
The first line in the response header is the status line. It contains the HTTP version and the status code. The date follows next, and then some information about the host server and the retrieved data. The Content-Type header lets the client know the type of data retrieved so it knows how to handle it. Content-Length lets the client know the size of the response body. If the request didn't go throw, the client would get an error code and message, such as the following response header in case of a page not found error:

```
1                    HTTP/1.1 404 Not Found
```

Apache Architecture



Apache Core: Which Handles the Basic functionality of the Server. Such as allocating requests and maintaining and pooling all the connections.



- The core components of make up the Apache core are as follows:
    - http_protocol.c: This is the component that handles all of the routines that communicate directly with the client by using the HTTP protocol. This is the component that knows how to also handle the socket connections through which the client connects to the server. All data transfer is done through this component.
    - http_main.c: this component is responsible for the startup of the server and contains the main server loop that waits for and accepts connections. It is also in charge of managing timeouts.
    - http_request.c: This component handles the flow of request processing, passing control to the modules as needed in the right order. It is also in charge of error handling.
    - http_core.c: the component implementing the most basic functionality, it just bairly serves documents.
    - alloc.c: the component that takes care of allocating resource pools, and keeping track of them.
    - http_config.c : this component provides functions for other utilities, including reading configuration files and managing the information gathered from those files (), as well as support for virtual hosts. An important function of http_config is that it forms the list of modules that will be called to service during different phases of the requests that are going on within the server.

How to deploy a webapplication in Tomcat Server(How to host a website)

Tomcat's manager application

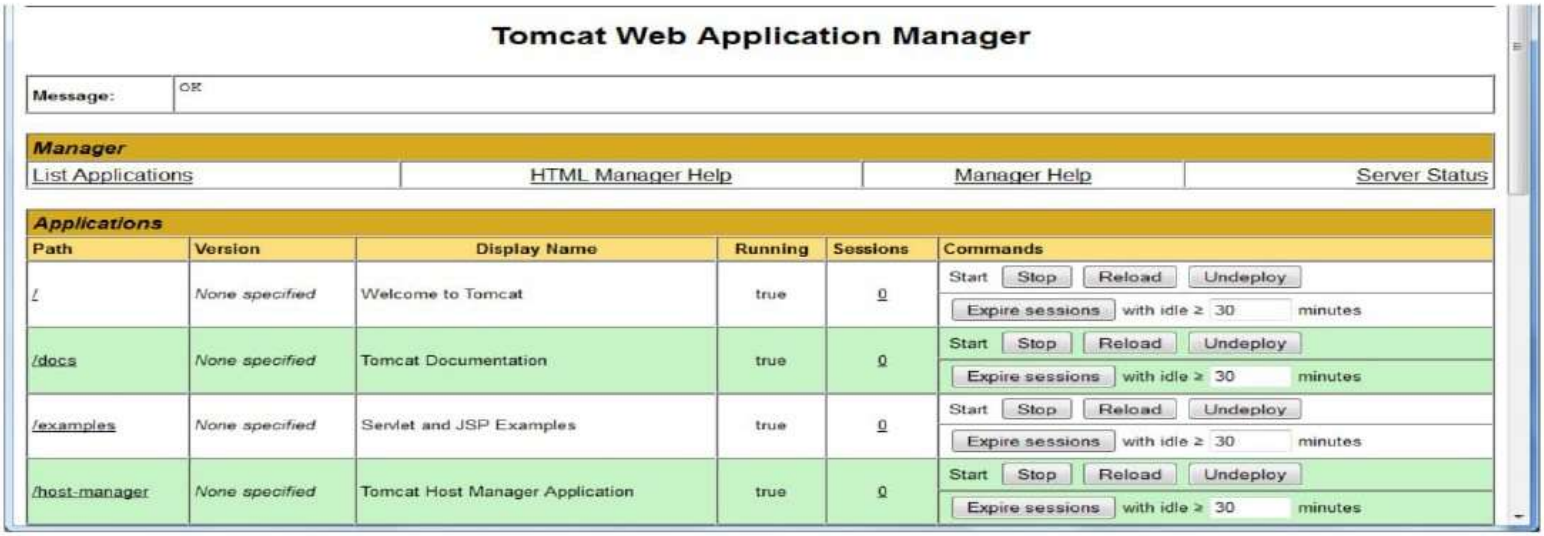
We can deploy the web application remotely via a web interface provided by Tomcat's manager application. You must have user name and password to access this application. The manager application is installed by default, but not always. So be sure that it is installed with your version of Tomcat.

Using the manager application, you can:

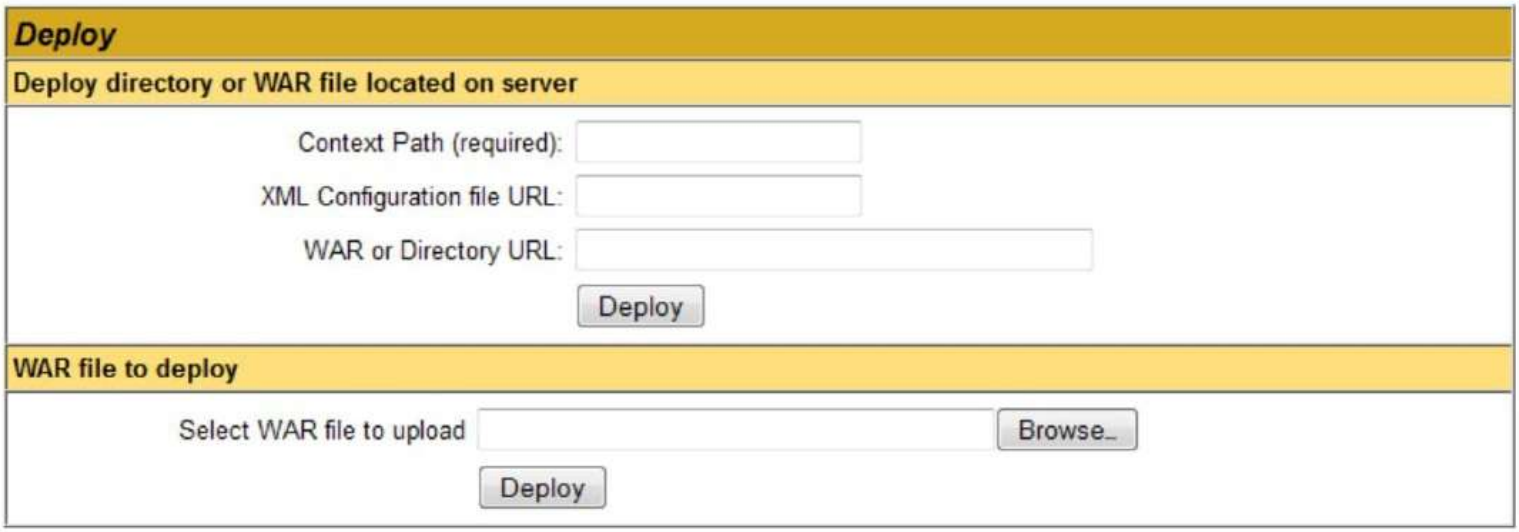
- View a list of applications deployed on the server and their status.
- Start, stop and restart an individual application.
- Deploy a new web application either by uploading a WAR file or supplying a directory on the server.
- Undeploy an individual application.

By default, the manager application is deployed under context */manager*, so to access it, type the following URL into your web browser's address bar (the port number may vary, depending on your server's configuration):
http://localhost:8080/manager

After supplying correct user name and password, you get into the following screen:



The list of deployed applications is shown at the top, scroll down a little bit to see the deployment section:



As we can see, there are two ways for deploying a web application using the manager:

- Deploy directory or WAR file located on server.
- WAR file to deploy.