Ajax :

AJAX is an acronym for **Asynchronous JavaScript and XML**. It is a group of inter-related technologies like [JavaScript](https://www.javatpoint.com/javascript-tutorial), DOM, [XML](https://www.javatpoint.com/xml-tutorial), [HTML](https://www.javatpoint.com/html-tutorial)/[XHTML](https://www.javatpoint.com/xhtml-tutorial), [CSS](https://www.javatpoint.com/css-tutorial), [XMLHttpRequest](https://www.javatpoint.com/understanding-xmlhttprequest) 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?

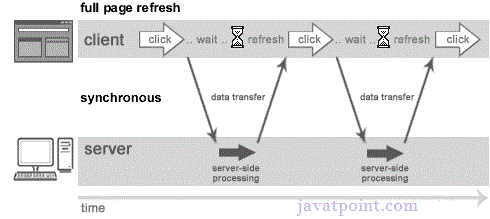
There are too many web applications running on the web that are using ajax technology like **gmail**, **facebook**,**twitter**,**google map**, **youtube** etc.

# 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.



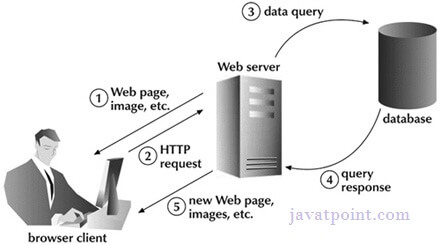
As you can see in the above image, full page is refreshed at request time and user is blocked until request completes.

Let's understand it another way.

19.2M

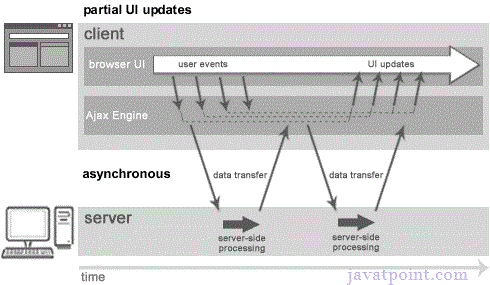
400

Java Try Catch



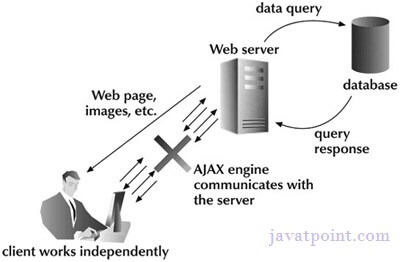
## Asynchronous (AJAX Web-Application Model)

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.



As you can see 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

As describe earlier, ajax is not a technology but group of inter-related technologies. [AJAX](https://www.javatpoint.com/ajax-tutorial) technologies includes:

* [HTML](https://www.javatpoint.com/html-tutorial)/[XHTML](https://www.javatpoint.com/xhtml-tutorial) and [CSS](https://www.javatpoint.com/css-tutorial)
* DOM
* [XML](https://www.javatpoint.com/xml-tutorial) or [JSON](https://www.javatpoint.com/json-tutorial)
* [XMLHttpRequest](https://www.javatpoint.com/understanding-xmlhttprequest)
* [JavaScript](https://www.javatpoint.com/javascript-tutorial)

## 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](https://www.javatpoint.com/understanding-synchronous-vs-asynchronous) between client and server. For more visit next page.

## JavaScript

It is used to bring above technologies together.

Independently, it is used mainly for client-side validation.

XMLHttpRequest

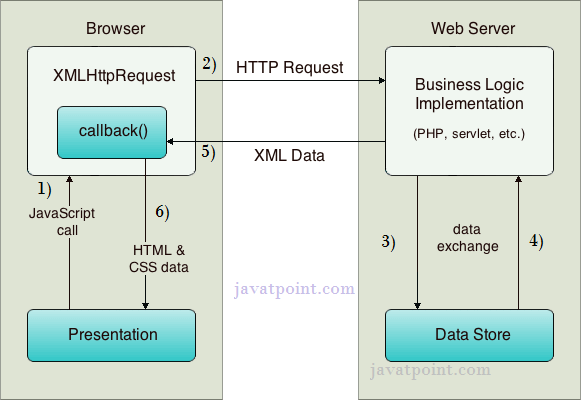
An object of XMLHttpRequest is used for asynchronous communication between client and server.

It performs following operations:

1. Sends data from the client in the background
2. Receives the data from the server
3. Updates the webpage without reloading it.

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.



As you can see in the above example, XMLHttpRequest object plays a important role.

1. User sends a request from the UI and a javascript call goes 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.

# Ajax Java Example

To create [ajax](https://www.javatpoint.com/ajax-tutorial) example, you need to use any server-side language e.g. [Servlet](https://www.javatpoint.com/servlet-tutorial), [JSP](https://www.javatpoint.com/jsp-tutorial), [PHP](https://www.javatpoint.com/php-tutorial), [ASP.Net](https://www.javatpoint.com/asp-net-tutorial) etc. Here we are using JSP for generating the server-side code.

In this example, we are simply printing the table of the given number.

#### Steps to create ajax example with jsp

You need to follow following steps:

1. load the org.json.jar file
2. create input page to receive any text or number
3. create server side page to process the request
4. provide entry in web.xml file

#### Load the org.json.jar file

download this example, we have included the org.json.jar file inside the WEB-INF/lib directory.

#### create input page to receive any text or number

In this page, we have created a form that gets input from the user. When user clicks on the showTable button, **sendInfo()** function is called. We have written all the ajax code inside this function.

We have called the **getInfo()** function whenever ready state changes. It writes the returned data in the web page dynamically by the help of **innerHTML** property.

**table1.html**

1. **<html>**
2. **<head>**
3. **<script>**
4. var request;
5. function sendInfo()
6. {
7. var v=document.vinform.t1.value;
8. var url="index.jsp?val="+v;
10. if(window.XMLHttpRequest){
11. request=new XMLHttpRequest();
12. }
13. else if(window.ActiveXObject){
14. request=new ActiveXObject("Microsoft.XMLHTTP");
15. }
17. try
18. {
19. request.onreadystatechange=getInfo;
20. request.open("GET",url,true);
21. request.send();
22. }
23. catch(e)
24. {
25. alert("Unable to connect to server");
26. }
27. }
29. function getInfo(){
30. if(request.readyState==4){
31. var val=request.responseText;
32. document.getElementById('amit').innerHTML=val;
33. }
34. }
36. **</script>**
37. **</head>**
38. **<body>**
39. **<marquee><h1>**This is an example of ajax**</h1></marquee>**
40. **<form** name="vinform"**>**
41. **<input** type="text" name="t1"**>**
42. **<input** type="button" value="ShowTable" onClick="sendInfo()"**>**
43. **</form>**
45. **<span** id="amit"**>** **</span>**
47. **</body>**
48. **</html>**

#### create server side page to process the request

In this jsp page, we printing the table of given number.

**index.jsp**

1. **<**%
2. int n=Integer.parseInt(request.getParameter("val"));
4. for(int i=1;i**<**=10;i++)
5. out.print(i\*n+"**<br>**");
7. %**>**

#### web.xml

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<web-app** version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5. http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"**>**
7. **<session-config>**
8. **<session-timeout>**
9. 30
10. **</session-timeout>**
11. **</session-config>**
12. **<welcome-file-list>**
13. **<welcome-file>**table1.html**</welcome-file>**
14. **</welcome-file-list>**
15. **</web-app>**

# Ajax Java Example with Database

In this example, we are interacting with the database. You don't have to make any extra effort. Only write the database logic in you server side page.

In this example, we have written the server side code inside the index.jsp file.

#### Steps to create ajax example with database through jsp

You need to follow following steps:

1. load the org.json.jar file
2. create input page to receive any text or number
3. create server side page to process the request

#### Load the org.json.jar file

download this example, we have included the org.json.jar file inside the WEB-INF/lib directory.

#### create input page to receive any text or number

In this page, we have created a form that gets input from the user. When user press any key **sendInfo()** function is called. We have written all the [ajax](https://www.javatpoint.com/ajax-tutorial) code inside this function.

We have called the **getInfo()** function whenever ready state changes. It writes the returned data in the web page dynamically by the help of **[innerHTML](https://www.javatpoint.com/javascript-innerHTML)** property.

**table1.html**

1. **<html>**
2. **<head>**
3. **<script>**
4. var request;
5. function sendInfo()
6. {
7. var v=document.vinform.t1.value;
8. var url="index.jsp?val="+v;
10. if(window.XMLHttpRequest){   //for Chrome, mozilla etc
11. request=new XMLHttpRequest();
12. }
13. else if(window.ActiveXObject){  //for IE only
14. request=new ActiveXObject("Microsoft.XMLHTTP");
15. }
17. try{
18. request.onreadystatechange=getInfo;
19. request.open("GET",url,true);
20. request.send();
21. }catch(e){alert("Unable to connect to server");}
22. }
24. function getInfo(){
25. if(request.readyState==4){
26. var val=request.responseText;
27. document.getElementById('amit').innerHTML=val;
28. }
29. }
31. **</script>**
32. **</head>**
33. **<body>**
34. **<marquee><h1>**This is an example of ajax**</h1></marquee>**
35. **<form** name="vinform"**>**
36. Enter id:**<input** type="text" name="t1" onkeyup="sendInfo()"**>**
37. **</form>**
39. **<span** id="amit"**>** **</span>**
41. **</body>**
42. **</html>**

#### create server side page to process the request

In this jsp page, we printing the id and name of the employee for the given id.

**index.jsp**

1. **<**%@ page import="java.sql.\*"%**>**
3. **<**%
4. String s=request.getParameter("val");
5. if(s==null || s.trim().equals("")){
6. out.print("Please enter id");
7. }else{
8. int id=Integer.parseInt(s);
9. out.print(id);
10. try{
11. Class.forName("com.mysql.jdbc.Driver");
12. Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/mdb","root","root");
13. PreparedStatement ps=con.prepareStatement("select \* from emp where id=?");
14. ps.setInt(1,id);
15. ResultSet rs=ps.executeQuery();
16. while(rs.next()){
17. out.print(rs.getInt(1)+" "+rs.getString(2));
18. }
19. con.close();
20. }catch(Exception e){e.printStackTrace();}
21. }
22. %**>**