AJAX TUTORIAL



AJAX tutorial covers concepts and examples of AJAX technology for beginners and professionals.

AJAX is an acronym for **Asynchronous JavaScript and XML**. It is a group of inter-related technologies like 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 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.

Next topics of AJAX Technology Tutorial

Our AJAX tutorial covers all the topics of AJAX.

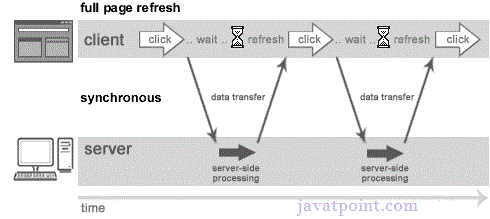
# Understanding Synchronous vs Asynchronous

1. [Synchronous Vs Asynchronous](http://www.javatpoint.com/understanding-synchronous-vs-asynchronous)

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

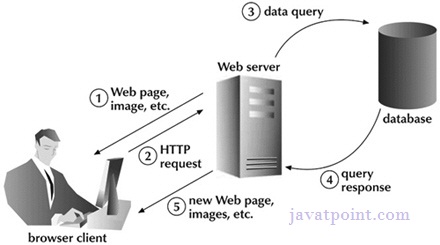
## Synchronous (Classic Web-Application Model)

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



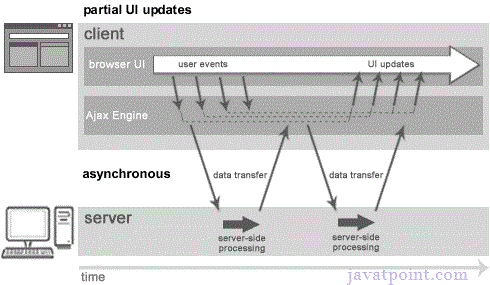
As you can see in the above image, full page is refreshed at request time and user is blocked until request completes.

Let's understand it another way.



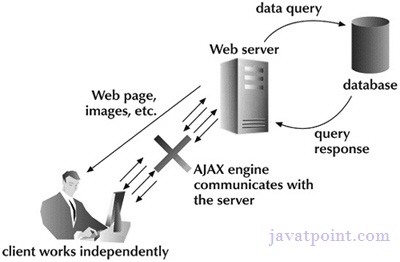
## Asynchronous (AJAX Web-Application Model)

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 technologies include:

* 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. For more visit next page.

JavaScript

It is used to bring above technologies together.

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

**Next Topic**[Understanding Xmlhttprequest](http://www.javatpoint.com/understanding-xmlhttprequest)

# Understanding 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.

Properties of XMLHttpRequest object

The common properties of XMLHttpRequest object are as follows:

|  |  |
| --- | --- |
| **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

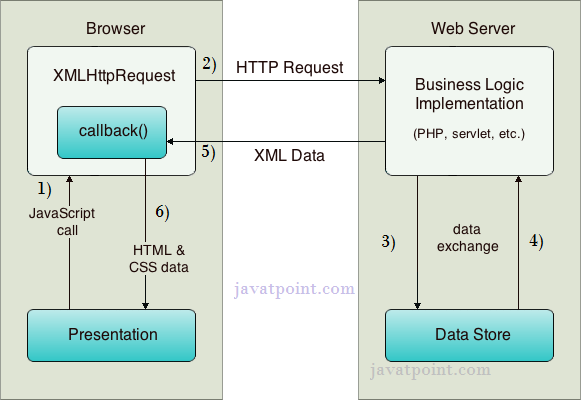
The important methods of XMLHttpRequest object are as follows:

|  |  |
| --- | --- |
| **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. |

**Next Topic**[How Ajax Works](http://www.javatpoint.com/how-ajax-works)

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.

**Next Topic**[Ajax Example](http://www.javatpoint.com/ajax-example)

AJAX EXAMPLE

To create ajax example, you need to use any server-side language e.g. servlet, jsp, php, asp.net 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**

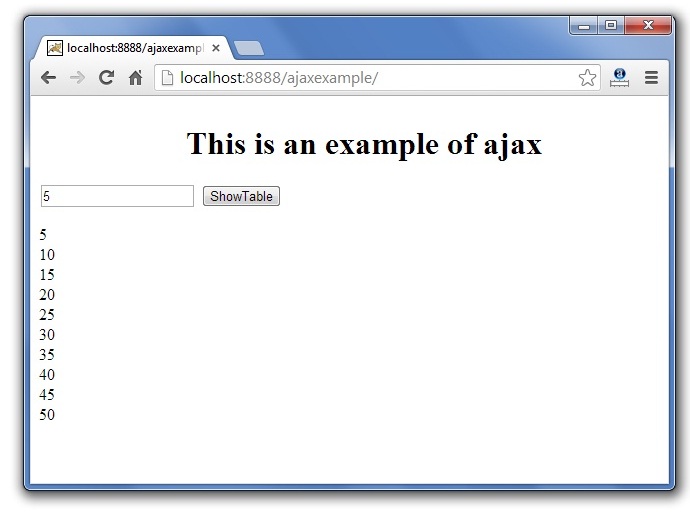
1. **<**%
2. int n=Integer.parseInt(request.getParameter("val"));
4. for(int i=1;i**<**=10;i++)
5. out.print(i\*n+"**<br>**");
7. %**>**

**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>**

[download this ajax example](http://www.javatpoint.com/src/ajax/ajaxexample.zip)

**Output**



**Next Topic**[Ajax Example With Database](http://www.javatpoint.com/ajax-example-with-database)

AJAX 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
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 press any key **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. 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**

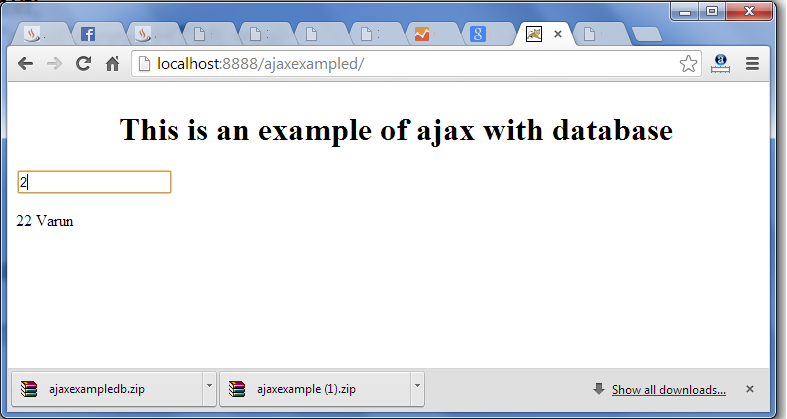
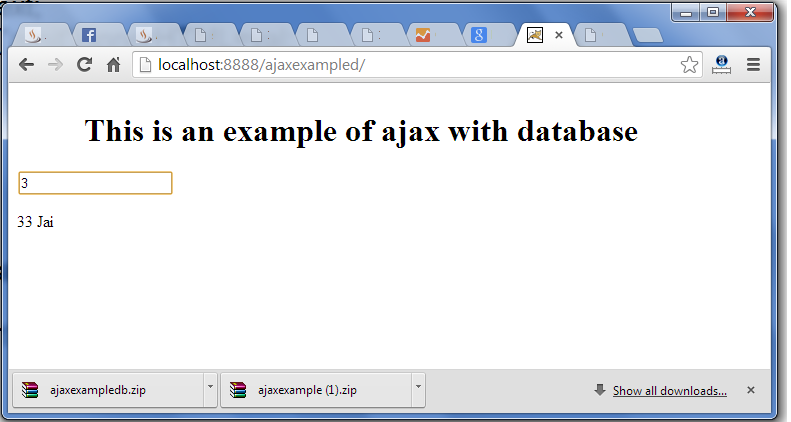
1. **<**%@ page import="java.sql.\*"%**>**
3. **<**%
4. String s=request.getParameter("val");
5. if(s==null || s.trim().equals("")){
6. out.print("Please enter id");
7. }else{
8. int id=Integer.parseInt(s);
9. out.print(id);
10. try{
11. Class.forName("com.mysql.jdbc.Driver");
12. Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/mdb","root","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. %**>**

**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>**

[download this ajax example](http://www.javatpoint.com/src/ajax/ajaxexampled.zip)

**Output**

# AJAX JSON Example

We can get JSON data by AJAX code. AJAX provides facility to get response asynchronously. It doesn't reload the page and saves bandwidth.

### AJAX JSON Example

Let's see a simple example of getting JSON data using AJAX code.

1. <html>
2. <head>
3. <meta content="text/html; charset=utf-8">
4. <title>AJAX JSON by Javatpoint</title>
5. <script type="application/javascript">
6. function load()
7. {
8. var url = "http://date.jsontest.com/";//use any url that have json data
9. var request;
11. **if**(window.XMLHttpRequest){
12. request=**new** XMLHttpRequest();//for Chrome, mozilla etc
13. }
14. **else** **if**(window.ActiveXObject){
15. request=**new** ActiveXObject("Microsoft.XMLHTTP");//for IE only
16. }
17. request.onreadystatechange  = function(){
18. **if** (request.readyState == 4  )
19. {
20. var jsonObj = JSON.parse(request.responseText);//JSON.parse() returns JSON object
21. document.getElementById("date").innerHTML =  jsonObj.date;
22. document.getElementById("time").innerHTML = jsonObj.time;
23. }
24. }
25. request.open("GET", url, **true**);
26. request.send();
27. }
28. </script>
29. </head>
30. <body>
32. Date: <span id="date"></span><br/>
33. Time: <span id="time"></span><br/>
35. <button type="button" onclick="load()">Load Information</button>
36. </body>
37. </html>

Output:

Date:

Time:

Load Information

**Next Topic**[JSON Object](http://www.javatpoint.com/json-object)

JSON Object

JSON object holds key/value pair. Each key is represented as a string in JSON and value can be of any type. The keys and values are separated by colon. Each key/value pair is separated by comma.

The curly brace **{** represents JSON object.

Let's see an example of JSON object.

1. {
2. "employee": {
3. "name":       "sonoo",
4. "salary":      56000,
5. "married":    **true**
6. }
7. }

In the above example, employee is an object in which "name", "salary" and "married" are the key. In this example, there are string, number and boolean value for the keys.

JSON Object with Strings

The string value must be enclosed within double quote.

1. {
2. "name":       "sonoo",
3. "email":      "sonoojaiswal1987@gmail.com"
4. }

JSON Object with Numbers

JSON supports numbers in double precision floating-point format. The number can be digits (0-9), fractions (.33, .532 etc) and exponents (e, e+, e-,E, E+, E-).

1. {
2. "integer": 34,
3. "fraction": .2145,
4. "exponent": 6.61789e+0
5. }

JSON Object with Booleans

JSON also supports boolean values *true* or *false*.

1. {
2. "first": **true**,
3. "second": **false**
4. }

JSON Nested Object Example

A JSON object can have another object also. Let's see a simple example of JSON object having another object.

1. {
2. "firstName": "Sonoo",
3. "lastName": "Jaiswal",
4. "age": 27,
5. "address" : {
6. "streetAddress": "Plot-6, Mohan Nagar",
7. "city": "Ghaziabad",
8. "state": "UP",
9. "postalCode": "201007"
10. }
11. }

**Next Topic**[JSON Array](http://www.javatpoint.com/json-array)

JSON Array

JSON array represents ordered list of values. JSON array can store multiple values. It can store string, number, boolean or object in JSON array.

In JSON array, values must be separated by comma.

The **[** (square bracket) represents JSON array.

JSON Array of Strings

Let's see an example of JSON arrays storing string values.

1. ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"]

JSON Array of Numbers

Let's see an example of JSON arrays storing number values.

1. [12, 34, 56, 43, 95]

JSON Array of Booleans

Let's see an example of JSON arrays storing boolean values.

1. [**true**, **true**, **false**, **false**, **true**]

JSON Array of Objects

Let's see a simple JSON array example having 4 objects.

1. {"employees":[
2. {"name":"Ram", "email":"ram@gmail.com", "age":23},
3. {"name":"Shyam", "email":"shyam23@gmail.com", "age":28},
4. {"name":"John", "email":"john@gmail.com", "age":33},
5. {"name":"Bob", "email":"bob32@gmail.com", "age":41}
6. ]}

JSON Multidimensional Array

We can store array inside JSON array, it is known as array of arrays or multidimensional array.

1. [
2. [ "a", "b", "c" ],
3. [ "m", "n", "o" ],
4. [ "x", "y", "z" ]
5. ]

**Next Topic**[JSON Comments](http://www.javatpoint.com/json-comments)

JSON Comments

JSON doesn't support comments. It is not a standard.

But you can do some tricks such as adding extra attribute for comment in JSON object, for example:

1. {
2. "employee": {
3. "name":       "Bob",
4. "salary":      56000,
5. "comments":    "He is a nice man"
6. }
7. }

**Next Topic**[PHP JSON Example](http://www.javatpoint.com/php-json-example)

# PHP JSON

PHP allows us to encode and decode JSON by the help of json\_encode() and json\_decode functions.

## 1) PHP json\_encode

The json\_encode() function returns the JSON representation of a value. In other words, it converts PHP variable (containing array) into JSON.

Syntax:

1. string json\_encode ( mixed $value [, **int** $options = 0 [, **int** $depth = 512 ]] )

### PHP json\_encode example 1

Let's see the example to encode JSON.

1. <?php
2. $arr = array('a' => 1, 'b' => 2, 'c' => 3, 'd' => 4, 'e' => 5);
3. echo json\_encode($arr);
4. ?>

Output

{"a":1,"b":2,"c":3,"d":4,"e":5}

### PHP json\_encode example 2

Let's see the example to encode JSON.

1. <?php
2. $arr2 = array('firstName' => 'Rahul', 'lastName' => 'Kumar', 'email' => 'rahul@gmail.com');
3. echo json\_encode($arr2);
4. ?>

Output

{"firstName":"Rahul","lastName":"Kumar","email":"rahul@gmail.com"}

## 2) PHP json\_decode

The json\_decode() function decodes the JSON string. In other words, it converts JSON string into a PHP variable.

Syntax:

1. mixed json\_decode ( string $json [, bool $assoc = **false** [, **int** $depth = 512 [, **int** $options = 0 ]]] )

### PHP json\_decode example 1

Let's see the example to decode JSON string.

1. <?php
2. $json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';
3. var\_dump(json\_decode($json, **true**));//true means returned object will be converted into associative array
4. ?>

Output

array(5) {

["a"] => int(1)

["b"] => int(2)

["c"] => int(3)

["d"] => int(4)

["e"] => int(5)

}

### PHP json\_decode example 2

Let's see the example to decode JSON string.

1. <?php
2. $json2 = '{"firstName" : "Rahul", "lastName" : "Kumar", "email" : "rahul@gmail.com"}';
3. var\_dump(json\_decode($json2, **true**));
4. ?>

Output

array(3) {

["firstName"]=> string(5) "Rahul"

["lastName"]=> string(5) "Kumar"

["email"]=> string(15) "rahul@gmail.com"

}

**Next Topic**[Java JSON Example](http://www.javatpoint.com/java-json-example)

Java JSON

The **json.simple** library allows us to read and write JSON data in Java. In other words, we can encode and decode JSON object in java using json.simple library.

The org.json.simple package contains important classes for JSON API.

* JSONValue
* JSONObject
* JSONArray
* JsonString
* JsonNumber

Install json.simple

To install json.simple, you need to set classpath of json-simple.jar or add the Maven dependency.

1) [Download json-simple.jar](http://www.javatpoint.com/jsonpages/json-simple-1.1.1.jar), Or

2) To add maven dependency, write the following code in pom.xml file.

1. <dependency>
2. <groupId>com.googlecode.json-simple</groupId>
3. <artifactId>json-simple</artifactId>
4. <version>1.1</version>
5. </dependency>

1) Java JSON Encode

Let's see a simple example to encode JSON object in java.

1. **import** org.json.simple.JSONObject;
2. **public** **class** JsonExample1{
3. **public** **static** **void** main(String args[]){
4. JSONObject obj=**new** JSONObject();
5. obj.put("name","sonoo");
6. obj.put("age",**new** Integer(27));
7. obj.put("salary",**new** Double(600000));
8. System.out.print(obj);
9. }}

Output:

{"name":"sonoo","salary":600000.0,"age":27}

Java JSON Encode using Map

Let's see a simple example to encode JSON object using map in java.

1. **import** java.util.HashMap;
2. **import** java.util.Map;
3. **import** org.json.simple.JSONValue;
4. **public** **class** JsonExample2{
5. **public** **static** **void** main(String args[]){
6. Map obj=**new** HashMap();
7. obj.put("name","sonoo");
8. obj.put("age",**new** Integer(27));
9. obj.put("salary",**new** Double(600000));
10. String jsonText = JSONValue.toJSONString(obj);
11. System.out.print(jsonText);
12. }}

Output:

{"name":"sonoo","salary":600000.0,"age":27}

Java JSON Array Encode

Let's see a simple example to encode JSON array in java.

1. **import** org.json.simple.JSONArray;
2. **public** **class** JsonExample1{
3. **public** **static** **void** main(String args[]){
4. JSONArray arr = **new** JSONArray();
5. arr.add("sonoo");
6. arr.add(**new** Integer(27));
7. arr.add(**new** Double(600000));
8. System.out.print(arr);
9. }}

Output:

["sonoo",27,600000.0]

Java JSON Array Encode using List

Let's see a simple example to encode JSON array using List in java.

1. **import** java.util.ArrayList;
2. **import** java.util.List;
3. **import** org.json.simple.JSONValue;
4. **public** **class** JsonExample1{
5. **public** **static** **void** main(String args[]){
6. List arr = **new** ArrayList();
7. arr.add("sonoo");
8. arr.add(**new** Integer(27));
9. arr.add(**new** Double(600000));
10. String jsonText = JSONValue.toJSONString(arr);
11. System.out.print(jsonText);
12. }}

Output:

["sonoo",27,600000.0]

2) Java JSON Decode

Let's see a simple example to decode JSON string in java.

1. **import** org.json.simple.JSONObject;
2. **import** org.json.simple.JSONValue;
3. **public** **class** JsonDecodeExample1 {
4. **public** **static** **void** main(String[] args) {
5. String s="{\"name\":\"sonoo\",\"salary\":600000.0,\"age\":27}";
6. Object obj=JSONValue.parse(s);
7. JSONObject jsonObject = (JSONObject) obj;
9. String name = (String) jsonObject.get("name");
10. **double** salary = (Double) jsonObject.get("salary");
11. **long** age = (Long) jsonObject.get("age");
12. System.out.println(name+" "+salary+" "+age);
13. }
14. }

Output:

sonoo 600000.0 27

**Next Topic**[AJAX JSON Example](http://www.javatpoint.com/ajax-json-example)