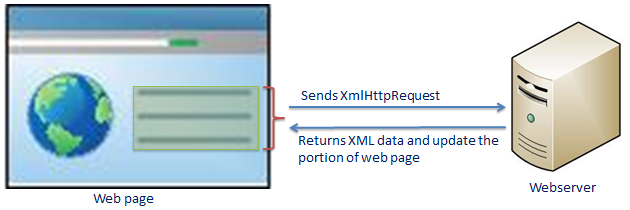
**jQuery Ajax Introduction:**

Ajax stands for "Asynchronous JavaScript and XML".

JavaScript includes features of sending asynchronous http request using XMLHttpRequest object. Ajax is about using this ability of JavaScript to send asynchronous http request and get the xml data as a response (also in other formats) and update the part of a web page (using JavaScript) without reloading or refreshing entire web page.

The following figure illustrates the Ajax functionality.

[](http://www.tutorialsteacher.com/Content/images/jquery/ajax-introduction.png)Ajax

The jQuery library includes various methods to send Ajax requests. These methods internally use XMLHttpRequest object of JavaScript. The following table lists all the Ajax methods of jQuery.

| **jQuery Ajax Methods** | **Description** |
| --- | --- |
| ajax() | Sends asynchronous http request to the server. |
| get() | Sends http GET request to load the data from the server. |
| Post() | Sends http POST request to submit or load the data to the server. |
| getJSON() | Sends http GET request to load JSON encoded data from the server. |
| getScript() | Sends http GET request to load the JavaScript file from the server and then executes it. |
| load() | Sends http request to load the html or text content from the server and add them to DOM element(s). |

The jQuery library also includes following events which will be fired based on the state of the Ajax request.

| **jQuery Ajax Events** | **Description** |
| --- | --- |
| ajaxComplete() | Register a handler function to be called when Ajax requests complete. |
| ajaxError() | Register a handler function to be called when Ajax requests complete with an error. |
| ajaxSend() | Register a handler function to be called before Ajax request is sent. |
| ajaxStart() | Register a handler function to be called when the first Ajax request begins. |
| ajaxStop() | Register a handler function to be called when all the Ajax requests have completed. |
| ajaxSuccess() | Register a handler function to be called when Ajax request completes successfully. |

**Advantages of jQuery Ajax:**

1. Cross-browser support
2. Simple methods to use
3. Ability to send GET and POST requests
4. Ability to Load JSON, XML, HTML or Scripts

Let's look at an overview of important jQuery Ajax methods in the next section.

**jQuery ajax() method:**

The jQuery ajax() method provides core functionality of Ajax in jQuery. It sends asynchronous HTTP requests to the server.

Syntax:

$.ajax(url);

$.ajax(url,[options]);

Parameter description:

* url: A string URL to which you want to submit or retrieve the data
* options: Configuration options for Ajax request. An options parameter can be specified using JSON format. This parameter is optional.

The following table list all the options available for configuring Ajax request.

| **Options** | **Description** |
| --- | --- |
| accepts | The content type sent in the request header that tells the server what kind of response it will accept in return. |
| async | By default, all requests are sent asynchronously. Set it false to make it synchronous. |
| beforeSend | A callback function to be executed before Ajax request is sent. |
| cache | A boolean indicating browser cache. Default is true. |
| complete | A callback function to be executed when request finishes. |
| contentType | A string containing a type of content when sending MIME content to the server.Default is "application/x-www-form-urlencoded; charset=UTF-8" |
| crossDomain | A boolean value indicating whether a request is a cross-domain. |
| data | A data to be sent to the server. It can be JSON object, string or array. |
| dataType | The type of data that you're expecting back from the server. |
| error | A callback function to be executed when the request fails. |
| global | A Boolean indicating whether to trigger a global Ajax request handler or not. Default is true. |
| headers | An object of additional header key/value pairs to send along with request. |
| ifModified | Allow the request to be successful only if the response has changed since the last request. This is done by checking the Last-Modified header. Default value is false. |
| isLocal | Allow the current environment to be recognized as local. |
| jsonp | Override the callback function name in a JSONP request. This value will be used instead of 'callback' in the 'callback=?' part of the query string in the url. |
| jsonpCallback | String containing the callback function name for a JSONP request. |
| mimeType | String containing a mime type to override the XMLHttpRequest mime type. |
| password | A password to be used with XMLHttpRequest in response to an HTTP access authentication request. |
| processData | A Boolean indicating whether data assigned to data option will be converted to a query string. Default is true. |
| statusCode | A JSON object containing numeric HTTP codes and functions to be called when the response has the corresponding code. |
| success | A callback function to be executed when Ajax request succeeds. |
| timeout | A number value in milliseconds for the request timeout. |
| type | A type of http request e.g. POST, PUT and GET. Default is GET. |
| url | A string containing the URL to which the request is sent. |
| username | A username to be used with XMLHttpRequest in response to an HTTP access authentication request. |
| xhr | A callback for creating the XMLHttpRequest object. |
| xhrFields | An object of fieldName-fieldValue pairs to set on the native XMLHttpRequest object. |

Let's see how to send http requests using $.ajax() (or jQuery.ajax()) method.

**Perform Ajax request:**

The ajax() methods performs asynchronous http request and gets the data from the server. The following example shows how to send a simple Ajax request.

Example: jQuery Ajax request

$.ajax('/jquery/getdata', // request url

{

success: function (data, status, xhr) {// success callback function

$('p').append(data);

}

});

<p></p>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-101)

In the above example, first parameter '/getData' of ajax() method is a url from which we want to retrieve the data.

http://www.tutorialsteacher.com/Content/images/tips.pngBy default ajax() method performs http GET request if option parameter does not include **method** option.

The second parameter is options parameter in JSON format where we have specified callback function that will be executed when request succeeds. You can configure other options as mentioned in the above table.

The following example shows how to get the JSON data using ajax() method.

Example: get JSON data

$.ajax('/jquery/getjsondata',

{

dataType: 'json', // type of response data

timeout: 500, // timeout milliseconds

success: function (data,status,xhr) { // success callback function

$('p').append(data.firstName + ' ' + data.middleName + ' ' + data.lastName);

},

error: function (jqXhr, textStatus, errorMessage) { // error callback

$('p').append('Error: ' + errorMessage);

}

});

<p></p>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-102)

In the above example, first parameter is a request url which will return JSON data. In the options parameter, we have specified dataType and timeout options. The dataType option specifies the type of response data, in this case it is JSON. The timeout parameter specifies request timeout in milliseconds. We have also specified callback functions for error and success.

The ajax() method returns an object of jQuery XMLHttpRequest. The following example shows how to use jQuery XMLHttpRequest object.

Example: ajax() method

var ajaxReq = $.ajax('GetJsonData', {

dataType: 'json',

timeout: 500

});

ajaxReq.success(function (data, status, jqXhr) {

$('p').append(data.firstName + ' ' + data.middleName + ' ' + data.lastName);

})

ajaxReq.error(function (jqXhr, textStatus, errorMessage) {

$('p').append('Error: ' + errorMessage);

})

<p></p>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-103)

**Perform Http POST request using ajax() method:**

The ajax() method can send all type of http requests. The following example sends http POST request to the server.

Example: Send POST request with ajax() method:

$.ajax('/jquery/submitData', {

type: 'POST', // http method

data: { myData: 'This is my data.' }, // data to submit

success: function (data, status, xhr) {

$('p').append('status: ' + status + ', data: ' + data);

},

error: function (jqXhr, textStatus, errorMessage) {

$('p').append('Error' + errorMessage);

}

});

<p></p>

[Try it](http://www.tutorialsteacher.com/codeeditor?cid=jquery-104)

In the above example, first parameter is a url which is used to submit the data. In the options parameter, we have specified a type option as a POST, so ajax() method will send http POST request. Also, we have specified data option as a JSON object containing data which will be submitted to the server.

So this way you can send GET, POST or PUT request using ajax() method.

**Points to Remember :**

1. $.ajax() method allows you to send asynchronous http requests to submit or retrieve data from the server without reloading the whole page.
2. $.ajax() can be used to send http GET, POST, PUT, DELETE etc. request. It can retrieve any type of response from the server.
3. Syntax: $.ajax(url,[options])
4. Use option parameter to customize ajax request as per your need.

**jQuery Selector Reference:**

| **Selector Pattern** | **Example** | **Description** |
| --- | --- | --- |
| Element name | $('div') | Selects all< div> elements |
| :first | $('div:first') | Selects first <div> element in a DOM hierarchy. |
| :last | $('div:last') | Selects last <div> element in a DOM hierarchy. |
| Multiple elements | $('p, div, code') | Selects <p>,<div> and <code> elements |
| parent descendant | $('div p') | Selects all <p> elements which is descendant of <div> |
| parent child | $('div > p') | Selects <p> element which is child of <div> |
| \* | $(\*) | Selects all elements |
| #Id | $("#myDiv") | Selects element whose id is *myDiv* |
| element#id | $("div#myDiv") | Selects <div> element whose Id is *myDiv* |
| #id, #id | $("#myDiv1, #myDiv2") | Selects multiple elements whose id is myDiv1 or myDiv2. |
| .class | $(".myCSSClass") | Selects all the elements with *class=myCSSClass*. |
| .class .class | $(".myCSSClass1, .myCSSClass2 ") | Finds all elements whose class attribute is set to *myCSSClass1* or *myCSSClass2* |
| element.class | $("div.myCSSClass") | Finds all <div> elements with *class=myCSSClass* |
| :first-child | $("p:first-child") | Selects all <p> elements which is the first child of its parent element. (parent element can be anything) |
| :last-child | $("p:last-child") | Selects all <p> elements which is the last child of its parent element. (parent element can be anything) |
| :nth-child(n) | $("p:nth-child(5)") | Selects all <p> elements which is the 5th child of its parent element. (parent element can be anything) |
| :nth-last-child(n) | $("p:nth-last-child(2)") | Selects all <p> elements which is the 2nd last child of its parent element. (parent element can be anything) |
| :only-child | $("p:only-child") | Selects all <p> elements which is the only child of its parent element. (parent element can be anything) |
| [attribute] | $('[class]') | Selects all the elements with the *class* attribute(whatever the value). |
| [element[attribute] | $("div[class]") | Selects all the <div> elements that have a *class* attribute(whatever the value). |
| element[attribute = value] | $("div[class='myCls']") | Selects all the <div> elements whose class attributes are equal to *myCls*. |
| element[attribute |= value] | $("div[class|= 'myCls']") | Selects all the <div> elements whose class attributes are either equal to *myCls* or starting with *myCls* string followed by a hyphen (-). |
| element[attribute \*= "value"] | $("div[class \*= 'myCls']") | Selects <div> elements whose class attributes contains *myCls*. |
| element[attribute ~= "value"] | $("div[class ~= 'myCls']") | Selects div elements whose class attributes contains *myCls*, delimited by spaces. |
| element[attribute $= "value"] | $("div[class $= 'myCls']") | Selects <div> elements whose class attribute value ends with *myCls*. The comparison is case sensitive. |
| element[attribute != "value"] | $("div[class != 'myCls']") | Selects <div> elements which do not have *class* attribute or value does not equal to myCls. |
| element[attribute ^= "value"] | $("div[class ^= 'myCls']") | Selects <div> elements whose *class* attribute value starts with myCls. |
| :contains("value") | $("div:contains('tutorialsteacher')" | Selects all <div> elements that contains the text *'tutorialsteacher'* |
| :input | $(":input") | Selects all input elements. |
| :button | $(":button") | Selects all input elements where type="button". |
| :radio | $(":radio") | Selects all input types where type="radio" |
| :text | $(":radio") | Selects all input elements where type="text" . |
| ":checkbox" | $(":checkbox") | Selects all checkbox elements. |
| :submit | $(":submit") | Selects all input elements where type="submit". |
| :password | $(":password") | Selects all input elements where type="password". |
| :reset | $(":reset") | Selects all input elements where type="reset". |
| :image | $(':image') | Selects all input elements where type="image". |
| :file | $(':file') | Selects all input elements where type="file". |
| :enabled | $(':enabled') | Selects all enabled input elements. |
| :disabled | $(':disabled') | Selects all disabled input elements. |
| :selected | $(':selected') | Selects all selected input elements. |
| :checked | $(':checked') | Selects all checked input elements. |
| :hidden | $(':hidden') | Selects all hidden elements. |
| :visible | $(':visible') | Selects all visible elements. |
| :odd | $('tr:odd') | Selects all odd rows. (1,3,5,7..) |
| :even | $('tr:even') | Selects all even rows.(0,2,4,6..) |