jQuery AJAX Methods

AJAX is the art of exchanging data with a server, and update parts of a web page - without reloading the whole page.

The following table lists all the jQuery AJAX methods:

|  |  |
| --- | --- |
| **Method** | **Description** |
| [$.ajax()](https://www.w3schools.com/jquery/ajax_ajax.asp) | Performs an async AJAX request |
| $.ajaxPrefilter() | Handle custom Ajax options or modify existing options before each  request is sent and before they are processed by $.ajax() |
| [$.ajaxSetup()](https://www.w3schools.com/jquery/ajax_ajaxsetup.asp) | Sets the default values for future AJAX requests |
| $.ajaxTransport() | Creates an object that handles the actual transmission of Ajax data |
| [$.get()](https://www.w3schools.com/jquery/ajax_get.asp) | Loads data from a server using an AJAX HTTP GET request |
| [$.getJSON()](https://www.w3schools.com/jquery/ajax_getjson.asp) | Loads JSON-encoded data from a server using a HTTP GET request |
| $.parseJSON() | Deprecated in version 3.0, use [JSON.parse()](https://www.w3schools.com/js/js_json_parse.asp" \t "_blank) instead. Takes a well-formed JSON string and returns the resulting JavaScript value |
| [$.getScript()](https://www.w3schools.com/jquery/ajax_getscript.asp) | Loads (and executes) a JavaScript from a server using an AJAX HTTP GET request |
| [$.param()](https://www.w3schools.com/jquery/ajax_param.asp) | Creates a serialized representation of an array or object (can be used as URL  query string for AJAX requests) |
| [$.post()](https://www.w3schools.com/jquery/ajax_post.asp) | Loads data from a server using an AJAX HTTP POST request |
| [ajaxComplete()](https://www.w3schools.com/jquery/ajax_ajaxcomplete.asp) | Specifies a function to run when the AJAX request completes |
| [ajaxError()](https://www.w3schools.com/jquery/ajax_ajaxerror.asp) | Specifies a function to run when the AJAX request completes with an error |
| [ajaxSend()](https://www.w3schools.com/jquery/ajax_ajaxsend.asp) | Specifies a function to run before the AJAX request is sent |
| [ajaxStart()](https://www.w3schools.com/jquery/ajax_ajaxstart.asp) | Specifies a function to run when the first AJAX request begins |
| [ajaxStop()](https://www.w3schools.com/jquery/ajax_ajaxstop.asp) | Specifies a function to run when all AJAX requests have completed |
| [ajaxSuccess()](https://www.w3schools.com/jquery/ajax_ajaxsuccess.asp) | Specifies a function to run when an AJAX request completes successfully |
| [load()](https://www.w3schools.com/jquery/ajax_load.asp) | Loads data from a server and puts the returned data into the selected element |
| [serialize()](https://www.w3schools.com/jquery/ajax_serialize.asp) | Encodes a set of form elements as a string for submission |
| [serializeArray()](https://www.w3schools.com/jquery/ajax_serializearray.asp) | Encodes a set of form elements as an array of names and values |

Definition and Usage

The ajax() method is used to perform an AJAX (asynchronous HTTP) request.

All jQuery AJAX methods use the ajax() method. This method is mostly used for requests where the other methods cannot be used.

Syntax

$.ajax(*{name:value, name:value, ... }*)

The parameters specifies one or more name/value pairs for the AJAX request.

Possible names/values in the table below:

|  |  |
| --- | --- |
| **Name** | **Value/Description** |
| Async | A Boolean value indicating whether the request should be  handled asynchronous or not. Default is true |
| beforeSend(*xhr*) | A function to run before the request is sent |
| Cache | A Boolean value indicating whether the browser should cache the requested pages. Default is true |
| complete(*xhr,status*) | A function to run when the request is finished (after success and error functions) |
| contentType | The content type used when sending data to the server. Default is: "application/x-www-form-urlencoded" |
| Context | Specifies the "this" value for all AJAX related callback functions |
| Data | Specifies data to be sent to the server |
| dataFilter(*data*,*type*) | A function used to handle the raw response data of the XMLHttpRequest |
| Datatype | The data type expected of the server response. |
| error(*xhr,status,error*) | A function to run if the request fails. |
| Global | A Boolean value specifying whether or not to trigger global AJAX event handles for the request. Default is true |
| ifModified | A Boolean value specifying whether a request is only successful if the response has changed since the last request. Default is: false. |
| Jsonp | A string overriding the callback function in a jsonp request |
| jsonpCallback | Specifies a name for the callback function in a jsonp request |
| Password | Specifies a password to be used in an HTTP access authentication request. |
| processData | A Boolean value specifying whether or not data sent with the request should be transformed into a query string. Default is true |
| scriptCharset | Specifies the charset for the request |
| success(*result,status,xhr*) | A function to be run when the request succeeds |
| Timeout | The local timeout (in milliseconds) for the request |
| traditional | A Boolean value specifying whether or not to use the traditional style of param serialization |
| Type | Specifies the type of request. (GET or POST) |
| url | Specifies the URL to send the request to. Default is the current page |
| username | Specifies a username to be used in an HTTP access authentication request |
| Xhr | A function used for creating the XMLHttpRequest object |

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("button").click(function(){

$.ajax({url: "demo\_test.txt", success: function(result){

$("#div1").html(result);

}});

});

});

</script>

</head>

<body>

<div id="div1"><h2>Let jQuery AJAX Change This Text</h2></div>

<button>Get External Content</button>

</body>

</html>

# **jQuery ajaxSetup() Method**

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("button").click(function(){

$.ajaxSetup({url: "demo\_ajax\_load.txt", success: function(result){

$("div").html(result);}});

$.ajax();

});

});

</script>

</head>

<body>

<div><h2>Let AJAX change this text</h2></div>

<button>Change Content</button>

</body>

</html>

Definition and Usage

The ajaxSetup() method sets default values for future AJAX requests.

Syntax

$.ajaxSetup(*{name:value, name:value, ... }*)

The parameters specifies the settings for AJAX requests with one or more name/value pairs.

Possible names/values in the table below:

|  |  |
| --- | --- |
| **Name** | **Value/Description** |
| Async | A Boolean value indicating whether the request should be handled asynchronous or not. Default is true |
| beforeSend(*xhr*) | A function to run before the request is sent |
| Cache | A Boolean value indicating whether the browser should cache the requested pages. Default is true |
| complete(*xhr,status*) | A function to run when the request is finished (after success and error functions) |
| contentType | The content type used when sending data to the server. Default is: "application/x-www-form-urlencoded" |
| Context | Specifies the "this" value for all AJAX related callback functions |
| Data | Specifies data to be sent to the server |
| dataFilter(*data*,*type*) | A function used to handle the raw response data of the XMLHttpRequest |
| datatype | The data type expected of the server response. |
| error(*xhr,status,error*) | A function to run if the request fails. |
| Global | A Boolean value specifying whether or not to trigger global AJAX event handles for the request. Default is true |
| ifModified | A Boolean value specifying whether a request is only successful if the response has changed since the last request. Default is: false. |
| Jsonp | A string overriding the callback function in a jsonp request |
| jsonpCallback | Specifies a name for the callback function in a jsonp request |
| Password | Specifies a password to be used in an HTTP access authentication request. |
| processData | A Boolean value specifying whether or not data sent with the request should be transformed into a query string. Default is true |
| scriptCharset | Specifies the charset for the request |
| success(*result,status,xhr*) | A function to be run when the request succeeds |
| Timeout | The local timeout (in milliseconds) for the request |
| traditional | A Boolean value specifying whether or not to use the traditional style of param serialization |
| Type | Specifies the type of request. (GET or POST) |
| url | Specifies the URL to send the request to. Default is the current page |
| username | Specifies a username to be used in an HTTP access authentication request |
| Xhr | A function used for creating the XMLHttpRequest object |

# **jQuery get() Method**

## Definition and Usage

The $.get() method loads data from the server using a HTTP GET request.

## Examples

Request "test.php", but ignore return results:

$.get("test.php");

Request "test.php" and send some additional data along with the request (ignore return results):

$.get("test.php", { name:"Donald", town:"Ducktown" });

Request "test.php" and pass arrays of data to the server (ignore return results):

$.get("test.php", { 'colors[]' : ["Red","Green","Blue"] });

Request "test.php" and alert the result of the request:

$.get("test.php", function(data){  
  alert("Data: " + data);  
});

## Syntax

$.get(*URL,data,function(data,status,xhr),dataType)*

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| URL | Required. Specifies the URL you wish to request |
| Data | Optional. Specifies data to send to the server along with the request |
| function(data,status,xhr) | Optional. Specifies a function to run if the request succeeds Additional parameters:   * data - contains the resulting data from the request * status - contains the status of the request ("success", "notmodified", "error", "timeout", or "parsererror") * xhr - contains the XMLHttpRequest object |
| datatype | Optional. Specifies the data type expected of the server response. By default jQuery performs an automatic guess. Possible types:   * "xml" - An XML document * "html" - HTML as plain text * "text" - A plain text string * "script" - Runs the response as JavaScript, and returns it as plain text * "json" - Runs the response as JSON, and returns a JavaScript object * "jsonp" - Loads in a JSON block using JSONP. Will add an "?callback=?" to the URL to specify the callback |

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("button").click(function(){

$.get("demo\_test.asp", function(data, status){

alert("Data: " + data + "\nStatus: " + status);

});

});

});

</script>

</head>

<body>

<button>Send an HTTP GET request to a page and get the result back</button>

</body>

</html>

Definition and Usage

The getJSON() method is used to get JSON data using an AJAX HTTP GET request.

Syntax

$(*selector*).getJSON(*url,data,success(data,status,xhr))*

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| *url* | Required. Specifies the url to send the request to |
| *Data* | Optional. Specifies data to be sent to the server |
| *success(data,status,xhr)* | Optional. Specifies the function to run if the request succeeds Additional parameters:   * *data* - contains the data returned from the server. * *status* - contains a string containing request status ("success", "notmodified", "error", "timeout", or "parsererror"). * *xhr* - contains the XMLHttpRequest object |

# **jQuery post() Method**

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("button").click(function(){

$.post("demo\_test\_post.asp",

{

name: "Donald Duck",

city: "Duckburg"

},

function(data,status){

alert("Data: " + data + "\nStatus: " + status);

});

});

});

</script>

</head>

<body>

<button>Send an HTTP POST request to a page and get the result back</button>

</body>

</html>

<!DOCTYPE html>

<html>

<head>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script>

$(document).ready(function(){

$("input").keyup(function(){

var txt = $("input").val();

$.post("demo\_ajax\_gethint.asp", {suggest: txt}, function(result){

$("span").html(result);

});

});

});

</script>

</head>

<body>

<p>Start typing a name in the input field below:</p>

First name:

<input type="text">

<p>Suggestions: <span></span></p>

<p>The file used in this example (<a href="demo\_ajax\_gethint.txt" target="\_blank">demo\_ajax\_gethint</a>) is explained in our Ajax tutorial</p>

</body>

</html>

Definition and Usage

The $.post() method loads data from the server using a HTTP POST request.

Syntax

$(*selector*).post(*URL,data,function(data,status,xhr),dataType)*

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| *URL* | Required. Specifies the url to send the request to |
| *Data* | Optional. Specifies data to send to the server along with the request |
| *function(data,status,xhr)* | Optional. Specifies a function to run if the request succeeds Additional parameters:   * *data* - contains the resulting data from the request * *status* - contains the status of the request ("success", "notmodified", "error", "timeout", or "parsererror") * *xhr* - contains the XMLHttpRequest object |
| *Datatype* | Optional. Specifies the data type expected of the server response. By default jQuery performs an automatic guess. Possible types:   * "xml" - An XML document * "html" - HTML as plain text * "text" - A plain text string * "script" - Runs the response as JavaScript, and returns it as plain text * "json" - Runs the response as JSON, and returns a JavaScript object * "jsonp" - Loads in a JSON block using SONP. Will add an "?callback=?" to the URL to specify the callback |

# **AJAX XML Example**

AJAX can be used for interactive communication with an XML file.

## AJAX XML Example

The following example will demonstrate how a web page can fetch information from an XML file with AJAX:

<!DOCTYPE html>

<html>

<style>

table,th,td {

border : 1px solid black;

border-collapse: collapse;

}

th,td {

padding: 5px;

}

</style>

<body>

<h2>The XMLHttpRequest Object</h2>

<button type="button" onclick="loadDoc()">Get my CD collection</button>

<br><br>

<table id="demo"></table>

<script>

function loadDoc() {

var xhttp = new XMLHttpRequest();

xhttp.onreadystatechange = function() {

if (this.readyState == 4 && this.status == 200) {

myFunction(this);

}

};

xhttp.open("GET", "cd\_catalog.xml", true);

xhttp.send();

}

function myFunction(xml) {

var i;

var xmlDoc = xml.responseXML;

var table="<tr><th>Artist</th><th>Title</th></tr>";

var x = xmlDoc.getElementsByTagName("CD");

for (i = 0; i <x.length; i++) {

table += "<tr><td>" +

x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +

"</td><td>" +

x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +

"</td></tr>";

}

document.getElementById("demo").innerHTML = table;

}

</script>

</body>

</html>

## Example Explained

When a user clicks on the "Get CD info" button above, the loadDoc() function is executed.

The loadDoc() function creates an XMLHttpRequest object, adds the function to be executed when the server response is ready, and sends the request off to the server.

When the server response is ready, an HTML table is built, nodes (elements) are extracted from the XML file, and it finally updates the element "demo" with the HTML table filled with XML data:

### **LoadXMLDoc()**

function loadDoc() {  
  var xhttp = new XMLHttpRequest();  
  xhttp.onreadystatechange = function() {  
    if (this.readyState == 4 && this.status == 200) {  
    myFunction(this);  
    }  
  };  
  xhttp.open("GET", "cd\_catalog.xml", true);  
  xhttp.send();  
}  
function myFunction(xml) {  
  var i;  
  var xmlDoc = xml.responseXML;  
  var table="<tr><th>Artist</th><th>Title</th></tr>";  
  var x = xmlDoc.getElementsByTagName("CD");  
  for (i = 0; i <x.length; i++) {   
    table += "<tr><td>" +  
    x[i].getElementsByTagName("ARTIST")[0].childNodes[0].nodeValue +  
    "</td><td>" +  
    x[i].getElementsByTagName("TITLE")[0].childNodes[0].nodeValue +  
    "</td></tr>";  
  }  
  document.getElementById("demo").innerHTML = table;  
}

## The XML File

The XML file used in the example above looks like this: "[cd\_catalog.xml](https://www.w3schools.com/js/cd_catalog.xml)".

This XML file does not appear to have any style information associated with it. The document tree is shown below.

<CATALOG>

<CD>

<TITLE>Empire Burlesque</TITLE>

<ARTIST>Bob Dylan</ARTIST>

<COUNTRY>USA</COUNTRY>

<COMPANY>Columbia</COMPANY>

<PRICE>10.90</PRICE>

<YEAR>1985</YEAR>

</CD>

<CD>

<TITLE>Hide your heart</TITLE>

<ARTIST>Bonnie Tyler</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>CBS Records</COMPANY>

<PRICE>9.90</PRICE>

<YEAR>1988</YEAR>

</CD>

<CD>

<TITLE>Greatest Hits</TITLE>

<ARTIST>Dolly Parton</ARTIST>

<COUNTRY>USA</COUNTRY>

<COMPANY>RCA</COMPANY>

<PRICE>9.90</PRICE>

<YEAR>1982</YEAR>

</CD>

<CD>

<TITLE>Still got the blues</TITLE>

<ARTIST>Gary Moore</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>Virgin records</COMPANY>

<PRICE>10.20</PRICE>

<YEAR>1990</YEAR>

</CD>

<CD>

<TITLE>Eros</TITLE>

<ARTIST>Eros Ramazzotti</ARTIST>

<COUNTRY>EU</COUNTRY>

<COMPANY>BMG</COMPANY>

<PRICE>9.90</PRICE>

<YEAR>1997</YEAR>

</CD>

<CD>

<TITLE>One night only</TITLE>

<ARTIST>Bee Gees</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>Polydor</COMPANY>

<PRICE>10.90</PRICE>

<YEAR>1998</YEAR>

</CD>

<CD>

<TITLE>Sylvias Mother</TITLE>

<ARTIST>Dr.Hook</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>CBS</COMPANY>

<PRICE>8.10</PRICE>

<YEAR>1973</YEAR>

</CD>

<CD>

<TITLE>Maggie May</TITLE>

<ARTIST>Rod Stewart</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>Pickwick</COMPANY>

<PRICE>8.50</PRICE>

<YEAR>1990</YEAR>

</CD>

<CD>

<TITLE>Romanza</TITLE>

<ARTIST>Andrea Bocelli</ARTIST>

<COUNTRY>EU</COUNTRY>

<COMPANY>Polydor</COMPANY>

<PRICE>10.80</PRICE>

<YEAR>1996</YEAR>

</CD>

<CD>

<TITLE>When a man loves a woman</TITLE>

<ARTIST>Percy Sledge</ARTIST>

<COUNTRY>USA</COUNTRY>

<COMPANY>Atlantic</COMPANY>

<PRICE>8.70</PRICE>

<YEAR>1987</YEAR>

</CD>

<CD>

<TITLE>Black angel</TITLE>

<ARTIST>Savage Rose</ARTIST>

<COUNTRY>EU</COUNTRY>

<COMPANY>Mega</COMPANY>

<PRICE>10.90</PRICE>

<YEAR>1995</YEAR>

</CD>

<CD>

<TITLE>1999 Grammy Nominees</TITLE>

<ARTIST>Many</ARTIST>

<COUNTRY>USA</COUNTRY>

<COMPANY>Grammy</COMPANY>

<PRICE>10.20</PRICE>

<YEAR>1999</YEAR>

</CD>

<CD>

<TITLE>For the good times</TITLE>

<ARTIST>Kenny Rogers</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>Mucik Master</COMPANY>

<PRICE>8.70</PRICE>

<YEAR>1995</YEAR>

</CD>

<CD>

<TITLE>Big Willie style</TITLE>

<ARTIST>Will Smith</ARTIST>

<COUNTRY>USA</COUNTRY>

<COMPANY>Columbia</COMPANY>

<PRICE>9.90</PRICE>

<YEAR>1997</YEAR>

</CD>

<CD>

<TITLE>Tupelo Honey</TITLE>

<ARTIST>Van Morrison</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>Polydor</COMPANY>

<PRICE>8.20</PRICE>

<YEAR>1971</YEAR>

</CD>

<CD>

<TITLE>Soulsville</TITLE>

<ARTIST>Jorn Hoel</ARTIST>

<COUNTRY>Norway</COUNTRY>

<COMPANY>WEA</COMPANY>

<PRICE>7.90</PRICE>

<YEAR>1996</YEAR>

</CD>

<CD>

<TITLE>The very best of</TITLE>

<ARTIST>Cat Stevens</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>Island</COMPANY>

<PRICE>8.90</PRICE>

<YEAR>1990</YEAR>

</CD>

<CD>

<TITLE>Stop</TITLE>

<ARTIST>Sam Brown</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>A and M</COMPANY>

<PRICE>8.90</PRICE>

<YEAR>1988</YEAR>

</CD>

<CD>

<TITLE>Bridge of Spies</TITLE>

<ARTIST>T'Pau</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>Siren</COMPANY>

<PRICE>7.90</PRICE>

<YEAR>1987</YEAR>

</CD>

<CD>

<TITLE>Private Dancer</TITLE>

<ARTIST>Tina Turner</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>Capitol</COMPANY>

<PRICE>8.90</PRICE>

<YEAR>1983</YEAR>

</CD>

<CD>

<TITLE>Midt om natten</TITLE>

<ARTIST>Kim Larsen</ARTIST>

<COUNTRY>EU</COUNTRY>

<COMPANY>Medley</COMPANY>

<PRICE>7.80</PRICE>

<YEAR>1983</YEAR>

</CD>

<CD>

<TITLE>Pavarotti Gala Concert</TITLE>

<ARTIST>Luciano Pavarotti</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>DECCA</COMPANY>

<PRICE>9.90</PRICE>

<YEAR>1991</YEAR>

</CD>

<CD>

<TITLE>The dock of the bay</TITLE>

<ARTIST>Otis Redding</ARTIST>

<COUNTRY>USA</COUNTRY>

<COMPANY>Stax Records</COMPANY>

<PRICE>7.90</PRICE>

<YEAR>1968</YEAR>

</CD>

<CD>

<TITLE>Picture book</TITLE>

<ARTIST>Simply Red</ARTIST>

<COUNTRY>EU</COUNTRY>

<COMPANY>Elektra</COMPANY>

<PRICE>7.20</PRICE>

<YEAR>1985</YEAR>

</CD>

<CD>

<TITLE>Red</TITLE>

<ARTIST>The Communards</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>London</COMPANY>

<PRICE>7.80</PRICE>

<YEAR>1987</YEAR>

</CD>

<CD>

<TITLE>Unchain my heart</TITLE>

<ARTIST>Joe Cocker</ARTIST>

<COUNTRY>USA</COUNTRY>

<COMPANY>EMI</COMPANY>

<PRICE>8.20</PRICE>

<YEAR>1987</YEAR>

</CD>

</CATALOG>

# **JSON vs XML**

Both JSON and XML can be used to receive data from a web server.

The following JSON and XML examples both define an employees object, with an array of 3 employees:

### **JSON Example**

{"employees":[  
  { "firstName":"John", "lastName":"Doe" },  
  { "firstName":"Anna", "lastName":"Smith" },  
  { "firstName":"Peter", "lastName":"Jones" }  
]}

### **XML Example**

<employees>  
  <employee>  
    <firstName>John</firstName> <lastName>Doe</lastName>  
  </employee>  
  <employee>  
    <firstName>Anna</firstName> <lastName>Smith</lastName>  
  </employee>  
  <employee>  
    <firstName>Peter</firstName> <lastName>Jones</lastName>  
  </employee>  
</employees>

## JSON is Like XML Because

* Both JSON and XML are "self describing" (human readable)
* Both JSON and XML are hierarchical (values within values)
* Both JSON and XML can be parsed and used by lots of programming languages
* Both JSON and XML can be fetched with an XMLHttpRequest

## JSON is Unlike XML Because

* JSON doesn't use end tag
* JSON is shorter
* JSON is quicker to read and write
* JSON can use arrays

The biggest difference is:

 XML has to be parsed with an XML parser. JSON can be parsed by a standard JavaScript function.

## Why JSON is Better Than XML

XML is much more difficult to parse than JSON.  
JSON is parsed into a ready-to-use JavaScript object.

For AJAX applications, JSON is faster and easier than XML:

Using XML

* Fetch an XML document
* Use the XML DOM to loop through the document
* Extract values and store in variables

Using JSON

* Fetch a JSON string
* JSON.Parse the JSON string

**AJAX with JSON file**

$.ajax({

dataType: "json",

url: url,

data: data,

success: success

});

$.getJSON( "ajax/test.json", **function**( data ) {

**var** items = [];

$.each( data, **function**( key, val ) {

items.push( "<li id='" + key + "'>" + val + "</li>" );

});

$( "<ul/>", {

"class": "my-new-list",

html: items.join( "" )

}).appendTo( "body" );

});