AJAX Introduction

AJAX is a developer's dream, because you can:

- Read data from a web server - after the page has loaded
- Update a web page without reloading the page
- Send data to a web server - in the background

What is AJAX?

AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.

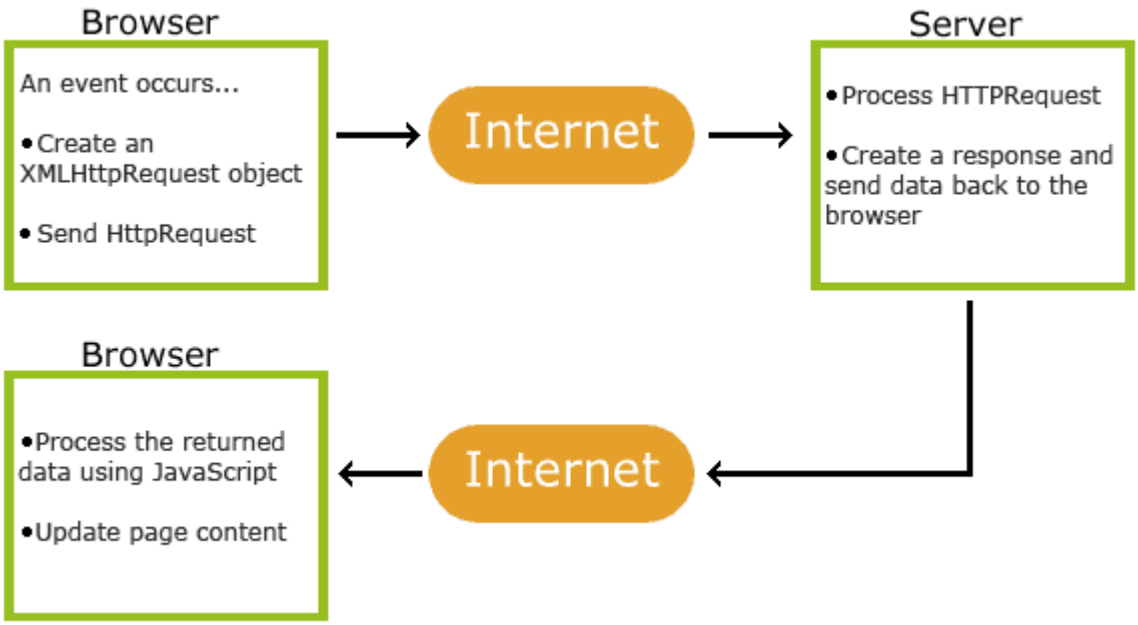
AJAX is not a programming language.

AJAX just uses a combination of:

- A browser built-in `XMLHttpRequest` object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data)

AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.

How AJAX Works



- 1. An event occurs in a web page (the page is loaded, a button is clicked)
- 2. An XMLHttpRequest object is created by JavaScript
- 3. The XMLHttpRequest object sends a request to a web server
- 4. The server processes the request

- 5. The server sends a response back to the web page
- 6. The response is read by JavaScript
- 7. Proper action (like page update) is performed by JavaScript

AJAX - The XMLHttpRequest Object

Create an XMLHttpRequest Object

Syntax for creating an `XMLHttpRequest` object:

```
variable = new XMLHttpRequest();
```

Example

```
var xhttp = new XMLHttpRequest();
```

Access Across Domains

For security reasons, modern browsers do not allow access across domains.

This means that both the web page and the XML file it tries to load, must be located on the same server.

Older Browsers (IE5 and IE6)

Old versions of Internet Explorer (5/6) use an ActiveX object instead of the `XMLHttpRequest` object:

```
variable = new ActiveXObject("Microsoft.XMLHTTP");
```

Example

```
if (window.XMLHttpRequest) {
  // code for modern browsers
  xmlhttp = new XMLHttpRequest();
 } else {
  // code for old IE browsers
  xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
```

XMLHttpRequest Object Methods

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information

getResponseHeader()	Returns specific header information
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" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

The url - A File On a Server

The url parameter of the `open()` method, is an address to a file on a server:

The file can be any kind of file, like .txt and .xml, or server scripting files like .asp and .php (which can perform actions on the server before sending the response back).

Asynchronous

By sending asynchronously, the JavaScript does not have to wait for the server response, but can instead:

- execute other scripts while waiting for server response
- deal with the response after the response is ready

Synchronous

Sometimes async = false are used for quick testing.

Since the code will wait for server completion, there is no need for an `onreadystatechange` function:

Example

```
xhttp.open("GET", "ajax_info.txt", false);
xhttp.send();
document.getElementById("demo").innerHTML = xhttp.responseText;
```

Synchronous XMLHttpRequest (async = false) is not recommended because the JavaScript will stop executing until the server response is ready. If the server is busy or slow, the application will hang or stop.

jQuery - AJAX Introduction

What About jQuery and AJAX?

jQuery provides several methods for AJAX functionality.

jQuery load() Method

The jQuery `load()` method is a simple, but powerful AJAX method.

The `load()` method loads data from a server and puts the returned data into the selected element.

Syntax:

$(*selector*).load(*URL,data,callback*);

- The required URL parameter specifies the URL you wish to load.
- The optional data parameter specifies a set of querystring key/value pairs to send along with the request.
- The optional callback parameter is the name of a function to be executed after the `load()` method is completed.

Here is the content of our example file: "demo_test.txt":

```
<h2>jQuery and AJAX is FUN!!!</h2>
<p id="p1">This is some text in a paragraph.</p>
```

Example

```
$("#div1").load("demo_test.txt");
```

It is also possible to add a jQuery selector to the URL parameter.

The following example loads the content of the element with id="p1", inside the file "demo_test.txt", into a specific `<div>` element:

Example

```
$("#div1").load("demo_test.txt #p1");
```

The optional callback parameter specifies a callback function to run when the `load()` method is completed. The callback function can have different parameters:

- `responseTxt` - contains the resulting content if the call succeeds
- `statusTxt` - contains the status of the call
- `xhr` - contains the XMLHttpRequest object

Example

```
$("button").click(function(){
  $("#div1").load("demo_test.txt", function(responseTxt, statusTxt, xhr){
    if(statusTxt == "success")
      alert("External content loaded successfully!");
    if(statusTxt == "error")
      alert("Error: " + xhr.status + ": " + xhr.statusText);
  });
});
```

jQuery - AJAX get() and post() Methods

HTTP Request: GET vs. POST

Two commonly used methods for a request-response between a client and server are: GET and POST.

- **GET** - Requests data from a specified resource
- **POST** - Submits data to be processed to a specified resource

jQuery $.get() Method

The `$.get()` method requests data from the server with an HTTP GET request.

Syntax:

```
$.get(URL,callback);
```

The required URL parameter specifies the URL you wish to request.

The optional callback parameter is the name of a function to be executed if the request succeeds.

Example

```
$("button").click(function(){
  $.get("demo_test.asp", function(data, status){
    alert("Data: " + data + "\nStatus: " + status);
  });
});
```

The first callback parameter holds the content of the page requested, and the second callback parameter holds the status of the request.

jQuery $.post() Method

The `$.post()` method requests data from the server using an HTTP POST request.

Syntax:

```
$.post(URL,data,callback);
```

The required URL parameter specifies the URL you wish to request.

The optional data parameter specifies some data to send along with the request.

The optional callback parameter is the name of a function to be executed if the request succeeds.

Example

```
$("button").click(function(){
  $.post("demo_test_post.asp",
  {
    name: "Donald Duck",
    city: "Duckburg"
  },
  function(data, status){
    alert("Data: " + data + "\nStatus: " + status);
  });
});
```

JSON - Introduction

Sending Data

If you have data stored in a JavaScript object, you can convert the object into JSON, and send it to a server:

Example

```
var myObj = {name: "John", age: 31, city: "New York"};
var myJSON = JSON.stringify(myObj);
window.location = "demo_json.php?x=" + myJSON;
```

Receiving Data

If you receive data in JSON format, you can convert it into a JavaScript object:

Example

```
var myJSON = '{"name":"John", "age":31, "city":"New York"}';
var myObj = JSON.parse(myJSON);
document.getElementById("demo").innerHTML = myObj.name;
```

Storing Data

When storing data, the data has to be a certain format, and regardless of where you choose to store it, *text* is always one of the legal formats.

JSON makes it possible to store JavaScript objects as text.

Example

Storing data in local storage

```
// Storing data:
myObj = {name: "John", age: 31, city: "New York"};
myJSON = JSON.stringify(myObj);
localStorage.setItem("testJSON", myJSON);

// Retrieving data:
```

```
text = localStorage.getItem("testJSON");
obj = JSON.parse(text);
document.getElementById("demo").innerHTML = obj.name;
```


jQuery Introduction

What is jQuery?

jQuery is a lightweight, "write less, do more", JavaScript library.

jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.

The jQuery library contains the following features:

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

Will jQuery work in all browsers?

The jQuery team knows all about cross-browser issues, and they have written this knowledge into the jQuery library. jQuery will run exactly the same in all major browsers.

Adding jQuery to Your Web Pages

There are several ways to start using jQuery on your web site. You can:

- Download the jQuery library from jQuery.com
- Include jQuery from a CDN, like Google

Downloading jQuery

There are two versions of jQuery available for downloading:

- Production version - this is for your live website because it has been minified and compressed
- Development version - this is for testing and development (uncompressed and readable code)

```
<head>
<script src="jquery-3.5.1.min.js"></script>
</head>
```

Tip: Place the downloaded file in the same directory as the pages where you wish to use it.

jQuery CDN

If you don't want to download and host jQuery yourself, you can include it from a CDN (Content Delivery Network).

Google CDN:

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
</head>
```

jQuery Syntax

The jQuery syntax is tailor-made for **selecting** HTML elements and performing some **action** on the element(s).

Basic syntax is: **$(*selector*).*action*()**

- A $ sign to define/access jQuery
- A (*selector*) to "query (or find)" HTML elements
- A jQuery *action*() to be performed on the element(s)

Examples:

`$(this).hide()` - hides the current element.

The Document Ready Event

```
$(document).ready(function(){

  //jQuery methods go here...

});
```

This is to prevent any jQuery code from running before the document is finished loading (is ready).

It is good practice to wait for the document to be fully loaded and ready before working with it. This also allows you to have your JavaScript code before the body of your document, in the head section.

jQuery Selectors

jQuery selectors allow you to select and manipulate HTML element(s).

All selectors in jQuery start with the dollar sign and parentheses: $( ).

The element Selector

The jQuery element selector selects elements based on the element name.

```
$("p")
```

The #id Selector

To find an element with a specific id, write a hash character, followed by the id of the HTML element:

```
$("#test")
```

The .class Selector

To find elements with a specific class, write a period character, followed by the name of the class:

```
$(".test")
```

More Examples of jQuery Selectors

Syntax	Description	Example
$("*")	Selects all elements	
$(this)	Selects the current HTML element	
$("p.intro")	Selects all <p> elements with class="intro"	
$("p:first")	Selects the first <p> element	
$("ul li:first")	Selects the first element of the first	
$("ul li:first-child")	Selects the first element of every	
$("[href]")	Selects all elements with an href attribute	
$("a[target='_blank']")	Selects all <a> elements with a target attribute value equal to "_blank"	
$("a[target!='_blank']")	Selects all <a> elements with a target attribute value NOT equal to "_blank"	
$(":button")	Selects all <button> elements and <input> elements of type="button"	
$("tr:even")	Selects all even <tr> elements	
$("tr:odd")	Selects all odd <tr> elements	

jQuery Event Methods

jQuery Syntax For Event Methods

In jQuery, most DOM events have an equivalent jQuery method.

Commonly Used jQuery Event Methods

$(document).ready()

The `$(document).ready()` method allows us to execute a function when the document is fully loaded.

click()

The function is executed when the user clicks on the HTML element.

Example

```
$("p").click(function(){
  $(this).hide();
});
```

The on() Method

The `on()` method attaches one or more event handlers for the selected elements.

Attach a click event to a `<p>` element:

Example

```
$("p").on("click", function(){
  $(this).hide();
});
```

Attach multiple event handlers to a `<p>` element:

Example

```
$("p").on({
  mouseenter: function(){
    $(this).css("background-color", "lightgray");
  },
  mouseleave: function(){
    $(this).css("background-color", "lightblue");
  },
  click: function(){
    $(this).css("background-color", "yellow");
  }
});
```


jQuery Effects

jQuery hide() and show()

With jQuery, you can hide and show HTML elements with the `hide()` and `show()` methods:

Syntax:

```
$(selector).hide(speed,callback);
```

```
$(selector).show(speed,callback);
```

The optional speed parameter specifies the speed of the hiding/showing, and can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the `hide()` or `show()` method completes

jQuery toggle()

You can also toggle between hiding and showing an element with the `toggle()` method.

Shown elements are hidden and hidden elements are shown:

Syntax:

```
$(selector).toggle(speed,callback);
```

jQuery Fading Methods

With jQuery you can fade an element in and out of visibility.

jQuery has the following fade methods:

- `fadeIn()`
- `fadeOut()`
- `fadeToggle()`
- `fadeTo()`

jQuery slideDown() Method

The jQuery `slideDown()` method is used to slide down an element.

Syntax:

```
$(selector).slideDown(speed,callback);
```

jQuery slideUp() Method

The jQuery `slideUp()` method is used to slide up an element.

Syntax:

```
$(selector).slideUp(speed,callback);
```

jQuery slideToggle() Method

The jQuery `slideToggle()` method toggles between the `slideDown()` and `slideUp()` methods.

```
$(selector).slideToggle(speed,callback);
```

jQuery slideDown() Method

The jQuery `slideDown()` method is used to slide down an element.

Syntax:

```
$(selector).slideDown(speed,callback);
```

jQuery slideUp() Method

The jQuery `slideUp()` method is used to slide up an element.

Syntax:

```
$(selector).slideUp(speed,callback);
```

jQuery slideToggle() Method

The jQuery `slideToggle()` method toggles between the `slideDown()` and `slideUp()` methods.

```
$(selector).slideToggle(speed,callback);
```

jQuery - Get Content and Attributes

jQuery DOM Manipulation

DOM = Document Object Model

The DOM defines a standard for accessing HTML and XML documents:

Get Content - text(), html(), and val()

Three simple, but useful, jQuery methods for DOM manipulation are:

- `text()` - Sets or returns the text content of selected elements
- `html()` - Sets or returns the content of selected elements (including HTML markup)
- `val()` - Sets or returns the value of form fields

Example

```
$("#btn1").click(function(){
  alert("Text: " + $("#test").text());
});
$("#btn2").click(function(){
  alert("HTML: " + $("#test").html());
});
```

```
$("#btn1").click(function(){
  alert("Value: " + $("#test").val());
});
```