



FREE eBook

LEARNING

ajax

Free unaffiliated eBook created from
Stack Overflow contributors.

#ajax

Table of Contents

About.....	1
Chapter 1: Getting started with ajax.....	2
Remarks.....	2
Examples.....	2
Installation or Setup.....	2
What is AJAX?.....	2
What is jQuery?.....	2
How to Add jQuery to your web site.....	2
Simple jQuery Example to Communicate with Server.....	3
Sync - Async ajax requests.....	4
Simple Ajax Request Sent Using the XMLHttpRequest Object.....	5
Using ajax in vanilla javascript with a simple callback.....	5
Performing an Asynchronous AJAX Call using TypeScript.....	6
Adding a Product to a Shopping Cart.....	6
Chapter 2: Callbacks.....	10
Examples.....	10
Interpreting errors with "error" callback.....	10
Credits.....	12

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: [ajax](#)

It is an unofficial and free ajax ebook created for educational purposes. All the content is extracted from [Stack Overflow Documentation](#), which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official ajax.

The content is released under Creative Commons BY-SA, and the list of contributors to each chapter are provided in the credits section at the end of this book. Images may be copyright of their respective owners unless otherwise specified. All trademarks and registered trademarks are the property of their respective company owners.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with ajax

Remarks

AJAX (asynchronous JavaScript and XML) allows you to request external data **without** blocking the execution of code. In many cases this is implemented in requesting pieces of a page or information from a server (via XMLHttpRequests) and then processing and displaying it using javascript.

The non-blocking nature of AJAX is what makes it such a widespread software pattern. Since javascript is blocking in the browser, a synchronous external call would make the browser unresponsive for the duration of the call until it either returned data or timed out. In effect making your application entirely dependent on external architecture and how well it will perform.

AJAX calls are usually abstracted out to provide additional functionality or readability, but implementations are (usually) built upon the [XMLHttpRequest Specification](#).

Examples

Installation or Setup

What is AJAX?

AJAX stands for Asynchronous JavaScript and XML. In a nutshell, it is the use of the XMLHttpRequest object to communicate with server-side scripts. It can send as well as receive information in a variety of formats, including JSON, XML, HTML, and even text files. -Mozilla Developer Network 2016

The easiest way of implementing AJAX, especially if you're planning on communicating with servers is by using jQuery.

What is jQuery?

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. -jquery.com

For those who haven't used much jQuery, think of it as functions that we can use to make our lives easier. This is perfect for using with AJAX as it cuts down on the amount of code we have to write to accomplish the same thing!

How to Add jQuery to your web site

If you need to Use Ajax, you need to add jQuery to your project. <http://jquery.com/download/> In This link you can see many ways to add jquery. You can use downloaded version of jQuery or you can use a CDN. <http://jquery.com/download/#jquery-39-s-cdn-provided-by-maxcdn>. But there is some security risk if you uses CDN. Because the project call out to use jquery, so a hacker could manipulate the call. So better if you could use downloaded version. Lets see how to add jquery to html project. It's easy. First example is to use Downloaded source. Use this link to <http://jquery.com/download/#jquery> download. If you are just want to use jquery, I suggest you to download **Download the compressed, production jQuery 3.1.1** When you download it add jquery-version.min.js to appropriate place (like javascript folder of your project) Then just add tag with src=jquery/location like below.

```
<head>

<script src="path/from/html/page/to/jquery.min.js"></script>

</head>
```

Lets see how to use an CDN. This link <http://jquery.com/download/#using-jquery-with-a-cdn> you can see various CDN (Content Delivery Network).

```
<head>

<script src="https://code.jquery.com/jquery-3.1.1.min.js" integrity="sha256-
hVVnYaiADRT02PzUGmuLJr8BLUSjGIZsDYGmIJLv2b8=" crossorigin="anonymous"></script>

</head>
```

As you can see in you just have to add the tags which the CDN provider supply to the . Now add some scripts to the html page to check it's working.

```
<script>
$(document).ready(function() {
    alert("jQuery Works")
});
</script>
```

If you see the **jQuery works** alert, That mean you added it correctly.

Simple jQuery Example to Communicate with Server

Taken from [jQuery.ajax API](#) web site:

```
$.ajax({
  method: "POST",
  url: "some.php",
  data: {
    name: "John",
    location: "Boston"
  },
  success: function(msg) {
    alert("Data Saved: " + msg);
  },
});
```

```

    error: function(e) {
        alert("Error: " + e);
    }
});

```

This chunk of code, due to jQuery, is easy to read and to understand what's going on.

- `$.ajax` - this bit calls jQuery's `ajax` functionality.
- `method: "POST"` - this line here declares that we're going to be using a POST method to communicate with the server. Read up on types of requests!
- `url` - this variable declares where the request is going to be **SENT** to. You're sending a request **TO** somewhere. That's the idea.
- `data` - pretty straight forward. This is the data you're sending with your request.
- `success` - this function here you write to decide what to do with the data you get back `msg!` as the example suggests, it's currently just creating an alert with the `msg` that gets returned.
- `error` - this function here you write to display error messages, or to provide actions to work when the `ajax` request went through errors.
- an alternative to `.done` is

```

success: function(result) {
    // do something
};

```

Sync - Async ajax requests

Asynchronous ajax call

With this type of ajax call, code does not wait for the call to complete.

```

$('form.ajaxSubmit').on('submit',function(){
    // initialization...
    var form = $(this);
    var formUrl = form.attr('action');
    var formType = form.attr('method');
    var formData = form.serialize();

    $.ajax({
        url: formUrl,
        type: formType,
        data: formData,
        async: true,
        success: function(data) {
            // .....
        }
    });
    /// code flows through without waiting for the call to complete
    return false;
});

```

Synchronous ajax call

With this type of ajax call, code waits for the call to complete.

```
$('#form.ajaxSubmit').on('submit',function(){
  // initialization...
  var form = $(this);
  var formUrl = form.attr('action');
  var formType = form.attr('method');
  var formData = form.serialize();
  var data = $.ajax({
    url: formUrl,
    type: formType,
    data: formData,
    async: false
  }).responseText;
  //// waits for call to complete
  return false;
});
```

Simple Ajax Request Sent Using the XMLHttpRequest Object

```
var httpRequest = new XMLHttpRequest();

httpRequest.onreadystatechange = getData;

httpRequest.open('GET', 'https://url/to/some.file', true);
httpRequest.send();

function getData(){
  if (httpRequest.readyState === XMLHttpRequest.DONE) {
    alert(httpRequest.responseText);
  }
}
```

`new XMLHttpRequest()` creates a new *XMLHttpRequest* object - this is what we will send our request with

The `onreadystatechange` bit tells our request to call `getData()` everytime it's status changes

`.open()` creates our request - this takes a *request method* ('GET', 'POST', etc.), a url of the page you're querying, and optionally, whether or not the request should be **asynchronous**

`.send()` sends our request - this optionally accepts data to send to the server like `.send(data)`

finally, the `getData()` is the function we've said should be called every time our request's **status changes**. if the `readyState` is equal to **DONE** then it alerts the `responseText` which is just the data recieved from the server.

More info can be found in the [getting started guide](#) on MDN.

Using ajax in vanilla javascript with a simple callback

Here is our function to create a simple ajax call written in vanilla javascript (not es2015):

```

function ajax(url, callback) {
    var xhr;

    if(typeof XMLHttpRequest !== 'undefined') xhr = new XMLHttpRequest();
    else {
        var versions = ["MSXML2.XmlHttp.5.0",
            "MSXML2.XmlHttp.4.0",
            "MSXML2.XmlHttp.3.0",
            "MSXML2.XmlHttp.2.0",
            "Microsoft.XmlHttp"]

        for(var i = 0, len = versions.length; i < len; i++) {
            try {
                xhr = new ActiveXObject(versions[i]);
                break;
            }
            catch(e){}
        } // end for
    }

    xhr.onreadystatechange = ensureReadiness;

    function ensureReadiness() {
        if(xhr.readyState < 4) {
            return;
        }

        if(xhr.status !== 200) {
            return;
        }

        // all is well
        if(xhr.readyState === 4) {
            callback(xhr);
        }
    }

    xhr.open('GET', url, true);
    xhr.send('');
}

```

and it could be used as:

```

ajax('myFile.html', function(response) {
    document.getElementById('container').innerHTML = response.responseText;
});

```

If you want to use EcmaScript 6 (also known as es2015) you can use the **fetch** method, which returns a promise:

```

fetch('myFile.json').then(function(res){
    return res.json();
});

```

For further reading about es2015 Promises follow the link bellow: [Promises](#)

Performing an Asynchronous AJAX Call using TypeScript

Adding a Product to a Shopping Cart

The following example demonstrates how to Add a product (or anything) to a Database Table asynchronously, using AJAX, and TypeScript.

```
declare var document;
declare var xhr: XMLHttpRequest;

window.onload = () =>
{
    Start();
};

function Start()
{
    // Setup XMLHttpRequest (xhr).
    if(XMLHttpRequest)
    {
        xhr = new XMLHttpRequest();
    }
    else
    {
        xhr = new ActiveXObject("Microsoft.XMLHTTP");
    }

    AttachEventListener(document.body, "click", HandleCheckBoxStateChange);
}

function AttachEventListener(element: any, e, f)
{
    // W3C Event Model.
    if(element.addEventListener)
    {
        element.addEventListener(e, f, false);
    }
    else if(element.attachEvent)
    {
        element.attachEvent("on" + e, (function(element, f)
        {
            return function()
            {
                f.call(element, window.event);
            };
        })
        (element, f));
    }

    element = null;
}

function HandleCheckBoxStateChange(e)
{
    var element = e.target || e.srcElement;

    if(element && element.type == "checkbox")
    {
        if(element.checked)
        {
```

```

        AddProductToCart(element);
    }
    else
    {
        // It is un-checked.
        // Remove item from cart.
    }
}
else
{
    break;
}
}

AddProductToCart(e)
{
    var element = <HTMLInputElement>document.getElementById(e.id);

    // Add the product to the Cart (Database table)
    xhr.onreadystatechange = function()
    {
        if(xhr.readyState == 4)
        {
            if(xhr.status == 200)
            {
                if(element != null)
                {
                    console.log("200: OK");
                }
                else
                {
                    console.log(":-(");
                }
            }
            else
            {
                // The server responded with a different response code; handle accordingly.
                // Probably not the most informative output.
                console.log(":-(");
            }
        }
    }

    var parameters = "ProductID=" + encodeURIComponent(e.id) + "&" + "Action=Add&Quantity=" +
    element.value;

    xhr.open("POST", "../Cart.cshtml");
    xhr.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    xhr.setRequestHeader("Content-length", parameters.length.toString());
    xhr.setRequestHeader("Connection", "close");

    xhr.send(parameters);

    return e.id;
}

```

In order to make this example complete, update your server-side code to actually insert this data into the database. The code above assumes you are using C# and have a `Cart.cshtml` file. However, simply replace `cshtml` with `php`, and write your own server-side logic using the language of your choice.

Read Getting started with ajax online: <https://riptutorial.com/ajax/topic/1082/getting-started-with-ajax>

Chapter 2: Callbacks

Examples

Interpreting errors with "error" callback

Errors, when managed properly by the server, will be returned to your client with a specific HTTP status code different from 2xx (see [RFC 2616 section 10](#)).

It's advised to catch globally your errors from your `$.ajaxSetup()` as demonstrated in the example below. Therefore all errors coming from your ajax calls will be automatically interpreted from the ajax setup.

```
$.ajaxSetup({
  error: function (jqXHR, exception, errorThrown) {
    var message;
    var statusErrorMap = {
      '400': "Server understood the request, but request content was invalid.",
      '401': "Unauthorized access.",
      '403': "Forbidden resource can't be accessed.",
      '500': "Internal server error.",
      '503': "Service unavailable."
    };
    if (jqXHR.status) {
      message = statusErrorMap[jqXHR.status];
      if (!message) {
        message = "Unknown Error.";
      }
    } else if (exception == 'parsererror') {
      message = "Error.\nParsing JSON Request failed.";
    } else if (exception == 'timeout') {
      message = "Request Time out.";
    } else if (exception == 'abort') {
      message = "Request was aborted by the server";
    } else {
      message = "Unknown Error.";
    }

    // How you will display your error message...
    console.log(message);
    console.log(errorThrown);
  }
});
```

You may also want to "overload" the `error` callback in a specific `$.ajax()` when you are waiting for a specific error message.

```
$.ajax({
  url: './api',
  data: { parametersObject },
  type: 'post',
  dataType: 'json',
  success: function(output) {
```

```
        // Interpret success
    },
    error: function(xhr, textStatus, ErrorThrown) {
        // Specific error will not be interpreted by $.ajaxSetup
    }
});
```

Read Callbacks online: <https://riptutorial.com/ajax/topic/7175/callbacks>

Credits

S. No	Chapters	Contributors
1	Getting started with ajax	acupajoe , adielhercules , Alessio Cantarella , Community , delete me , JhWebDevGuy , jignesh prajapati , MAZux , Menuka Ishan , Nicholas Qiao , TimTheEnchanter , Tolga Evcimen
2	Callbacks	maxime_039