17. jQuery - AJAX Introduction

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

17.1. What is AJAX?

AJAX = Asynchronous JavaScript and XML.

In short; AJAX is about loading data in the background and display it on the webpage, without reloading the whole page.

Examples of applications using AJAX: Gmail, Google Maps, Youtube, and Facebook tabs.

17.2. What About jQuery and AJAX?

jQuery provides several methods for AJAX functionality.

With the jQuery AJAX methods, you can request text, HTML, XML, or JSON from a remote server using both HTTP Get and HTTP Post - And you can load the external data directly into the selected HTML elements of your web page!

Without jQuery, AJAX coding can be a bit tricky!

Writing regular AJAX code can be a bit tricky, because different browsers have different syntax for AJAX implementation. This means that you will have to write extra code to test for different browsers. However, the jQuery team has taken care of this for us, so that we can write AJAX functionality with only one single line of code.

In the next pages we will look at the most important jQuery AJAX methods.

17.3. 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>
```

The following example loads the content of the file "demo_test.txt" into a specific element:

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 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 succeed
- statusTxt - contains the status of the call
- xhr - contains the XMLHttpRequest object

The following example displays an alert box after the load() method completes. If the load() method has succeed, it displays "External content loaded successfully!", and if it fails it displays an error message:

Example

```
$("button").click(function(){
  $
("#div1").load("demo_test.txt",function(res
ponseTxt,statusTxt,xhr){
    if(statusTxt=="success")
      alert("External content loaded
```

```
    successfully!");
        if(statusTxt=="error")
          alert("Error: "+xhr.status+":
"+xhr.statusText);
      });
});
```

17.4. AJAX get() and post() Methods

The jQuery get() and post() methods are used to request data from the server with an HTTP GET or POST request.

a) 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

GET is basically used for just getting (retrieving) some data from the server. **Note:** The GET method may return cached data.

POST can also be used to get some data from the server. However, the POST method NEVER caches data, and is often used to send data along with the request.

b) 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.

The following example uses the $.get() method to retrieve data from a file on the server:

Example

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

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

The first parameter of $.get() is the URL we wish to request ("demo_test.asp").

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

Tip: Here is how the ASP file looks like ("demo_test.asp"):

```
<%
response.write("This is some text from an external ASP
file.")
%>
```

c) 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.

The following example uses the $.post() method to send some data along with the request:

Example

```
$("button").click(function(){
  $.post("demo_test_post.asp",
  {
    name:"Donald Duck",
    city:"Duckburg"
```

```
    },
    function(data,status){
      alert("Data: " + data + "\nStatus: " + status);
    });
});
```

The first parameter of $.post() is the URL we wish to request ("demo_test_post.asp").

Then we pass in some data to send along with the request (name and city).

The ASP script in "demo_test_post.asp" reads the parameters, process them, and return a result.

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

Tip: Here is how the ASP file looks like ("demo_test_post.asp"):

```
<%
dim fname,city
fname=Request.Form("name")
city=Request.Form("city")
Response.Write("Dear " & fname & ". ")
Response.Write("Hope you live well in " &
city & ".")
%>
```