JQuery-AJAX

Presenter:
Agubata Odinaka

What is Ajax?

Definition

AJAX – Asynchronous Javascript and XML

AJAX is not a programming language, but a new way to use existing standards.

It is the act of exchanging data with a server, and updating parts of a web page – without reloading

What we can do with JQuery-AJAX?

JQuery is a great tool which provides a rich set of AJAX methods to develop next generation web application. We can use JQuery AJAX to do the following:

- Load External Pages
- Communicate Asynchronously with the Server

Load External Pages

It is very easy to load any static or dynamic data using JQuery AJAX. JQuery provides **load()** method to do the job –

Syntax

Here is the simple syntax for **load()** method –

```
[selector].load(URL, [data], [callback]);
```

Here is the description of all the parameters –

- **URL** – The URL of the server-side script/page to which the request is sent. It could be a PHP script which generates data dynamically or out of a database.

- **data** – This optional parameter represents an object whose properties are serialized into properly encoded parameters to be passed to the request. If specified, the request is made using the **POST** method. If omitted, the **GET** method is used.
- **callback** – A callback function invoked after the response data has been loaded into the elements of the matched set. The first parameter passed to this function is the response text received from the server and second parameter is the status code.

Example

Assuming we have a HTML page like this:

```
<!DOCTYPE html>
<html>
    <head>
            <script src="jquery.min.js“ type=“text/javascript”></script>
    </head>
    <body>
        <h1>Using Jquery Load Function</h1>
        <input id=“pageLoad" type="submit"/>
        <div id=“loadHere”></div>
    </body>
</html>
```

Assuming we have another HTML page like this saved as load.php:

```
<div >
    <h1>This is the loaded page</h1>
        <span>Linkschool Online gives you the freedom to learn what
you want, when you want. Learning is simple and more fun on
Linkschool E-learning Platform.
<a href="learnonline.php"> Learn more...</a></span>
 </div>
```

```
<script type=“text/javascript”>
    $(document).ready(function(){
        $(“#pageLoad").click(function(){
            $("#loadHere").load(“newPage.php", function(responseTxt,
        statusTxt, xhr){
                if(statusTxt == "success")
                    alert("External content loaded successfully!");
                if(statusTxt == "error")
                    alert("Error: " + xhr.status + ": " + xhr.statusText);
            });
        })
    });
</script>
```

Here **load()** initiates an Ajax request to the specified URL(newPage.php) file. After loading this file, all the content would be populated inside tagged with ID *loadHere*. 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

In the previous example, we displayed an alert box after the load() method was completed. If the load() method succeeds, it displays "External content loaded successfully!", and if it fails it displays an error message:

Communicate Asynchronously with the Server

JQuery and AJAX can be used together to communicate asynchronously with the server ice, it can be used to send data to, and also receive data from the server without the page reloading. There are two possible ways to achieve this:

- The JQuery get() method
- The JQuery post() method

The JQuery $.get() method

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

Syntax:

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

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

The optional data specifies data to send to the server along with the request

The optional callback function specifies a function to run if the request succeeds

The optional dataType specifies the data type expected of the server response.

Example

Assuming we have a HTML page like this:

```
<!DOCTYPE html>
<html>
    <head>
            <script src="jquery.min.js“ type=“text/javascript”></script>
    </head>
    <body>
        <h1>Using Jquery AJAX GET  Method</h1>
        <input placeholder=“chat” id=“chatId” type=“text”/>
        <input id=“sendChat" type="submit"/>
        <div id=“seeChat”></div>
    </body>
</html>
```

Assuming we have a server-side page saved as serverPage.php:

```
<?php
    extract($_GET,EXTR_OVERWRITE);
    echo $message.” it is cool”;
?>
```

```
<script type=“text/javascript”>
    $(document).ready(function(){
        $(“#sendChat").click(function(){
            var msg=$(“#chatId”).val();
            if(msg.length>0){
                $.get(“serverPage.php", {message: msg} , function(data, status){
                    if(status==“success”){
                        $("#seeChat").html(data);
                    }else{
                        alert(“Network Error, try again later”);
                    }
                })
            }
        })
    });
</script>
```

The JQuery $.post() method

POST requests are identical to GET requests in JQuery. So, generally which method you should use either $.get() or $.post() is basically depends on the requirements of your server-side code. If you have large amount of data to be transmitted (e.g. form data) you need to use POST, because GET has a stringent limit on the data transfer.

Syntax:

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

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

The optional data specifies data to send to the server along with the request

The optional callback function specifies a function to run if the request succeeds

The optional dataType specifies the data type expected of the server response.

Example

Assuming we have a HTML page like this:

```
<!DOCTYPE html>
<html>
    <head>
            <script src="jquery.min.js“ type=“text/javascript”></script>
    </head>
    <body>
        <h1>Using Jquery AJAX POST  Method</h1>
        <input placeholder=“chat” id=“chatId” type=“text”/>
        <input id=“sendChat" type="submit"/>
        <div id=“seeChat”></div>
    </body>
</html>
```

Assuming we have a server-side page saved as serverPage.php:

```
<?php
    extract($_POST,EXTR_OVERWRITE);
    echo $message.” it is cool”;
?>
```

```
<script type=“text/javascript”>
    $(document).ready(function(){
        $(“#sendChat").click(function(){
            var msg=$(“#chatId”).val();
            if(msg.length>0){
                $.post(“serverPage.php", {message: msg} , function(data, status){
                    if(status==“success”){
                        $("#seeChat").html(data);
                    }else{
                        alert(“Network Error, try again later”);
                    }
                })
            }
        })
    });
</script>
```

Another POST Example

Assuming we have a HTML page like this:

```
<!DOCTYPE html>
<html>
    <head>
        <script src="jquery.min.js“ type=“text/javascript”></script>
    </head>
    <body>
        <form action=“javascript:;”>
            <input placeholder=“chat” name=“message” id=“chatId” type=“text”/>
            <input name=“id” type=“text”/>
            <input value=“send” type="submit"/>
        </form>
    </body>
</html>
```

Assuming we have a server-side page saved as serverPage.php:

```
<?php
    extract($_POST,EXTR_OVERWRITE);
     $query="INSERT INTO MyTable(senderId, message) VALUES(?,?)";
     $statement=$connection->prepare($query);
     $statement->bind_param("ss",$id, $message);
     $statement->execute();
     $connection->close();
?>
```

```
<script type=“text/javascript”>
    $(document).ready(function(){
        $(“form").submit(function(event){
            var msg=$(“#chatId”).val();
            if(msg.length>0){
                $.post(“serverPage.php", $(this).serialize() , function(data, status){
                    if(status==“success”){
                        $(“<div>").insertBefore(“form”).html(msg);
                    }else{
                        alert(“Network Error, try again later”);
                    }
                })
            }
        })
    });
</script>
```

Questions