Lab 11. Javascript & AJAX

Prepared: TrangNTT

Note: Using Firebug to debug Javascript.

11.1. PHP and AJAX example

Step 1. Ajax basics

This example only focus on how to create a basic working AJAX - PHP communication. The only important thing at the moment is that AJAX uses JavaScript so it need to be enabled in your browser to successfully complete this example.

To demonstrate the AJAX PHP connection we will create a very simple form with 2 input fields. In the first field you can type any text and we will send this text to our PHP script which will convert it to uppercase and sends it back to us. At the end we will put the result into the second input field *(The example maybe not very useful but I think it is acceptable at this level).*

So let's list what we need to do:

- Listen on key-press event on the input field.
- In case of key-press send a message to the PHP script on the server.
- Process the input with PHP and send back the result.
- Capture the returning data and display it.

Our html code is very simple it looks like this:

```
1. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2.  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3. <html xmlns="http://www.w3.org/1999/xhtml">
4. <head>
5. <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
6. <title>Ajax - PHP example</title>
7. </head>
8.
9. <body>
10.
11.       <form name="testForm">
12.         Input text: <input type="text"  onkeyup="doWork();"
    name="inputText" id="inputText" />
13.         Output text: <input type="text" name="outputText"
    id="outputText" />
14.       </form>
15.       </body>
16.       </html>
```

As you can see there is a *doWork()* function which is called in every case when a key is up (a key was pressed). Of course you can use any other supported events if you want.

But what is this *doWork()* and how we can send messages to the server script? On the next page you will find the answers.

Step 2. Sending data to PHP with Ajax

Before the explanation of the *doWork()* function we first need to learn a more important thing. To make a communication between the client and the server the client code needs to create a so called *XMLHttpRequest* object. This object will be responsible for AJAX PHP communication.

However creating this object is bit triky as the browser implement it various ways. If you don't want to support the quite old browsers we can do it as follows:

```
1. // Get the HTTP Object
2. function getHTTPObject(){
3.    if (window.ActiveXObject)
4.        return new ActiveXObject("Microsoft.XMLHTTP");
5.    else if (window.XMLHttpRequest)
6.        return new XMLHttpRequest();
7.    else {
8.       alert("Your browser does not support AJAX.");
9.       return null;
10.        }
11.       }
```

Ok, now we have the *XMLHttpRequest* object, so it's time to implement the business logic inside the *doWork()* function.

First of all we need to get a valid *XMLHttpRequest* object. If we have it then we can send the value of the inputText field to the server script. We do this by composing an URL with parameter, so in the PHP script we can use the *$_GET* super-global array to catch the data. As next step we call the *send()* function of the *XMLHttpRequest* object which will send our request to the server. At the moment our *doWork()* function looks like this:

```
1. // Implement business logic
2. function doWork(){
3.     httpObject = getHTTPObject();
4.     if (httpObject != null) {
5.         httpObject.open("GET", "upperCase.php?inputText="
6.             +document.getElementById('inputText').value, true);
7.         httpObject.send(null);
8.
9.     }
10.        }
```

It's nice but how we can catch the response from the server? To do this we need to use an other special property of the *XMLHttpRequest* object. We can assign a function to this parameter and this function will be called if the state of the object was changed. The final code is the following:

```
1. // Implement business logic
2. function doWork(){
3.     httpObject = getHTTPObject();
4.     if (httpObject != null) {
5.         httpObject.open("GET", "upperCase.php?inputText="
```

```
            +document.getElementById('inputText').value, true);
        httpObject.send(null);
        httpObject.onreadystatechange = setOutput;
    }
    }
```

The last step on client side is to implement the *setOutput()* function which will change the value of our second field. The only interesting thing in this function that we need to check the actual state of the *XMLHttpRequest* object. We need to change the field value only if the state is complete. The *readyState* property can have the following values:

- 0 = uninitialized
- 1 = loading
- 2 = loaded
- 3 = interactive
- 4 = complete

So the *setOutput()* looks like this:

```
// Change the value of the outputText field
function setOutput(){
    if(httpObject.readyState == 4){
        document.getElementById('outputText').value
            = httpObject.responseText;
    }

}
```

Now the client side is ready let's implement the server side.

Step 3. PHP code and the complete AJAX example

Implementing the server side functionality is very simple compared to the client side. In the PHP code we just need to check the *$_GET* super-global array. Afterwards convert it to uppercase and echo the result. So the PHP code is this:

```
<?php
    if (isset($_GET['inputText']))
       echo strtoupper($_GET['inputText']);
?>
```

That's really short, isn't it?

At least you can find the complete client and server code below.

Client:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Ajax - PHP example</title>
</head>

<body>

	<script language="javascript" type="text/javascript">
	<!--
	// Get the HTTP Object
	function getHTTPObject(){
	   if (window.ActiveXObject) return new
  ActiveXObject("Microsoft.XMLHTTP");
	   else if (window.XMLHttpRequest) return new XMLHttpRequest();
	   else {
	      alert("Your browser does not support AJAX.");
	      return null;
	   }
	}

	// Change the value of the outputText field
	function setOutput(){
	  if(httpObject.readyState == 4){
	    document.getElementById('outputText').value = httpObject.responseText;
	  }

	}

	// Implement business logic
	function doWork(){
	    httpObject = getHTTPObject();
	    if (httpObject != null) {
	        httpObject.open("GET", "upperCase.php?inputText="
	              +document.getElementById('inputText').value, true);
	        httpObject.send(null);
	        httpObject.onreadystatechange = setOutput;
	    }
	}

	var httpObject = null;

	//-->
	</script>

	  <form name="testForm">
	     Input text: <input type="text"  onkeyup="doWork();"
  name="inputText" id="inputText" />
```

```
49.            Output text: <input type="text" name="outputText"
   id="outputText" />
50.         </form>
51.       </body>
52.       </html>
```

Server:

```
1. <?php
2.     if (isset($_GET['inputText']))
3.       echo strtoupper($_GET['inputText']);
4. ?>
```

11.2. AJAX & MySQL example

Step 1. AJAX Database Example

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

Example

Peter Griffin

Firstname	Lastname	Age	Hometown	Job
Peter	Griffin	41	Quahog	Brewery

Step 2. Create the User table in MySQL Database

The database table we use in the example above looks like this:

id	FirstName	LastName	Age	Hometown	Job
1	Peter	Griffin	41	Quahog	Brewery
2	Lois	Griffin	40	Newport	Piano Teacher
3	Joseph	Swanson	39	Quahog	Police Officer
4	Glenn	Quagmire	41	Quahog	Pilot

Step 3. Create the HTML Page

When a user selects a user in the dropdown list above, a function called "showUser()" is executed. The function is triggered by the "onChange" event:

```
<html>
<head>
<script type="text/javascript">
function showUser(str){
  if (str=="") {
    document.getElementById("txtHint").innerHTML="";
    return;
  }
  if (window.XMLHttpRequest){
    // code for IE7+, Firefox, Chrome, Opera, Safari
    xmlhttp=new XMLHttpRequest();
  }
  else {
    // code for IE6, IE5
    xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
```

```
  xmlhttp.onreadystatechange=function() {
    if (xmlhttp.readyState==4 && xmlhttp.status==200){
      document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
    }
  }
  xmlhttp.open("GET","getuser.php?q="+str,true);
  xmlhttp.send();
}
</script>
</head>
<body>

<form>
<select name="users" onchange="showUser(this.value)">
<option value="">Select a person:</option>
<option value="1">Peter Griffin</option>
<option value="2">Lois Griffin</option>
<option value="3">Glenn Quagmire</option>
<option value="4">Joseph Swanson</option>
</select>
</form>
<br />
<div id="txtHint"><b>Person info will be listed here.</b></div>

</body>
</html>
```

The showUser() function does the following:

- Check if a person is selected
- Create an XMLHttpRequest object
- Create the function to be executed when the server response is ready
- Send the request off to a file on the server
- Notice that a parameter (q) is added to the URL (with the content of the dropdown list)

Step 4. Create the PHP File

The page on the server called by the JavaScript above is a PHP file called "getuser.php".

The source code in "getuser.php" runs a query against a MySQL database, and returns the result in an HTML table:

```
<?php
$q=$_GET["q"];

$con = mysql_connect('localhost', 'peter', 'abc123');
if (!$con) {
  die('Could not connect: ' . mysql_error());
}

mysql_select_db("ajax_demo", $con);

$sql="SELECT * FROM user WHERE id = '".$q."'";

$result = mysql_query($sql);

echo "<table border='1'>
<tr>
<th>Firstname</th>
<th>Lastname</th>
<th>Age</th>
<th>Hometown</th>
<th>Job</th>
```

```
</tr>";

while($row = mysql_fetch_array($result)) {
  echo "<tr>";
  echo "<td>" . $row['FirstName'] . "</td>";
  echo "<td>" . $row['LastName'] . "</td>";
  echo "<td>" . $row['Age'] . "</td>";
  echo "<td>" . $row['Hometown'] . "</td>";
  echo "<td>" . $row['Job'] . "</td>";
  echo "</tr>";
}
echo "</table>";

mysql_close($con);
?>
```

Explanation: When the query is sent from the JavaScript to the PHP file, the following happens:

- PHP opens a connection to a MySQL server
- The correct person is found
- An HTML table is created, filled with data, and sent back to the "txtHint" placeholder

11.3. Exercise 3 – Validate inputs (JS)

Create a register form and validate inputs in the form (username… not null, email, telephone in the right format…)

11.4. Exercise 4 – Search suggestions by AJAX

Develop a PHP application which allows users to search product information of your company. Everytime users enter any character in the product name, provide suggestions corresponding with entered characters. Using AJAX to complete the exercise.