AJAX - Action

This chapter gives you a clear picture of the exact steps of AJAX operation.

Steps of AJAX Operation

- A client event occurs.
- An XMLHttpRequest object is created.
- The XMLHttpRequest object is configured.
- The XMLHttpRequest object makes an asynchronous request to the Webserver.
- The Webserver returns the result containing XML document.
- The XMLHttpRequest object calls the callback() function and processes the result.
- The HTML DOM is updated.

Let us take these steps one by one.

A Client Event Occurs

- A JavaScript function is called as the result of an event.
- Example – validateUserId() JavaScript function is mapped as an event handler to an onkeyup event on input form field whose id is set to "userid"
- <input type = "text" size = "20" id = "userid" name = "id" onkeyup = "validateUserId();">.

The XMLHttpRequest Object is Created

```
var ajaxRequest;  // The variable that makes Ajax possible!
function ajaxFunction() {
   try {
      // Opera 8.0+, Firefox, Safari
      ajaxRequest = new XMLHttpRequest();
```

```
   } catch (e) {
      // Internet Explorer Browsers
      try {
         ajaxRequest = new ActiveXObject("Msxml2.XMLHTTP");
      } catch (e) {
         try {
            ajaxRequest = new ActiveXObject("Microsoft.XMLHTTP");
         } catch (e) {
            // Something went wrong
            alert("Your browser broke!");
            return false;
         }
      }
   }
}
```

The XMLHttpRequest Object is Configured

In this step, we will write a function that will be triggered by the client event and a callback function processRequest() will be registered.

```
function validateUserId() {
   ajaxFunction();

   // Here processRequest() is the callback function.
   ajaxRequest.onreadystatechange = processRequest;

   if (!target) target = document.getElementById("userid");
   var url = "validate?id=" + escape(target.value);

   ajaxRequest.open("GET", url, true);
   ajaxRequest.send(null);
}
```

Making Asynchronous Request to the Webserver

Source code is available in the above piece of code. Code written in bold typeface is responsible to make a request to the webserver. This is all being done using the XMLHttpRequest object ajaxRequest.

```
function validateUserId() {
   ajaxFunction();

   // Here processRequest() is the callback function.
   ajaxRequest.onreadystatechange = processRequest;

   <b>if (!target) target = document.getElementById("userid");
   var url = "validate?id = " + escape(target.value);

   ajaxRequest.open("GET", url, true);
   ajaxRequest.send(null);</b>
}
```

Assume you enter Zara in the userid box, then in the above request, the URL is set to "validate?id = Zara".

Webserver Returns the Result Containing XML Document

You can implement your server-side script in any language, however its logic should be as follows.

- Get a request from the client.
- Parse the input from the client.
- Do required processing.
- Send the output to the client.

If we assume that you are going to write a servlet, then here is the piece of code.

```
public void doGet(HttpServletRequest request,
   HttpServletResponse response) throws IOException, ServletException {
   String targetId = request.getParameter("id");

   if ((targetId != null) && !accounts.containsKey(targetId.trim())) {
      response.setContentType("text/xml");
      response.setHeader("Cache-Control", "no-cache");
      response.getWriter().write("<valid>true</valid>");
   } else {
      response.setContentType("text/xml");
      response.setHeader("Cache-Control", "no-cache");
      response.getWriter().write("<valid>false</valid>");
```

```
   }
}
```

Callback Function processRequest() is Called

The XMLHttpRequest object was configured to call the processRequest() function when there is a state change to the readyState of the XMLHttpRequest object. Now this function will receive the result from the server and will do the required processing. As in the following example, it sets a variable message on true or false based on the returned value from the Webserver.

```
function processRequest() {
   if (req.readyState == 4) {
      if (req.status == 200) {
         var message = ...;
...
}
```

The HTML DOM is Updated

This is the final step and in this step, your HTML page will be updated. It happens in the following way −

- JavaScript gets a reference to any element in a page using DOM API.
- The recommended way to gain a reference to an element is to call.

```
document.getElementById("userIdMessage"),
// where "userIdMessage" is the ID attribute
// of an element appearing in the HTML document
```

- JavaScript may now be used to modify the element's attributes; modify the element's style properties; or add, remove, or modify the child elements. Here is an example −

```
<script type = "text/javascript">
   <!--
   function setMessageUsingDOM(message) {
      var userMessageElement = document.getElementById("userIdMessage");
```

```
      if (message == "false") {
         userMessageElement.style.color = "red";
         messageText = "Invalid User Id";
      } else {
         userMessageElement.style.color = "green";
         messageText = "Valid User Id";
      }

      var messageBody = document.createTextNode(messageText);

      // if the messageBody element has been created simple
      // replace it otherwise append the new element
      if (userMessageElement.childNodes[0]) {
         userMessageElement.replaceChild(messageBody, userMessageElement.childNode
      } else {
         userMessageElement.appendChild(messageBody);
      }
   }
   -->
</script>
<body>
   <div id = "userIdMessage"><div>
</body>
```

If you have understood the above-mentioned seven steps, then you are almost done with AJAX. In the next chapter, we will see XMLHttpRequest object in more detail.