AJAX

Malintha Adikari

1

What is AJAX

AJAX

- **AJAX** stands for "Asynchronous JavaScript And XMLHttpRequest"
- **AJAX** is a web development technique for creating interactive web applications
- **AJAX** is a set of Web development techniques using many Web technologies on the client side to create asynchronous Web applications.
- **AJAX** is also not a new technology, or another different language, just existing technologies used in new ways.

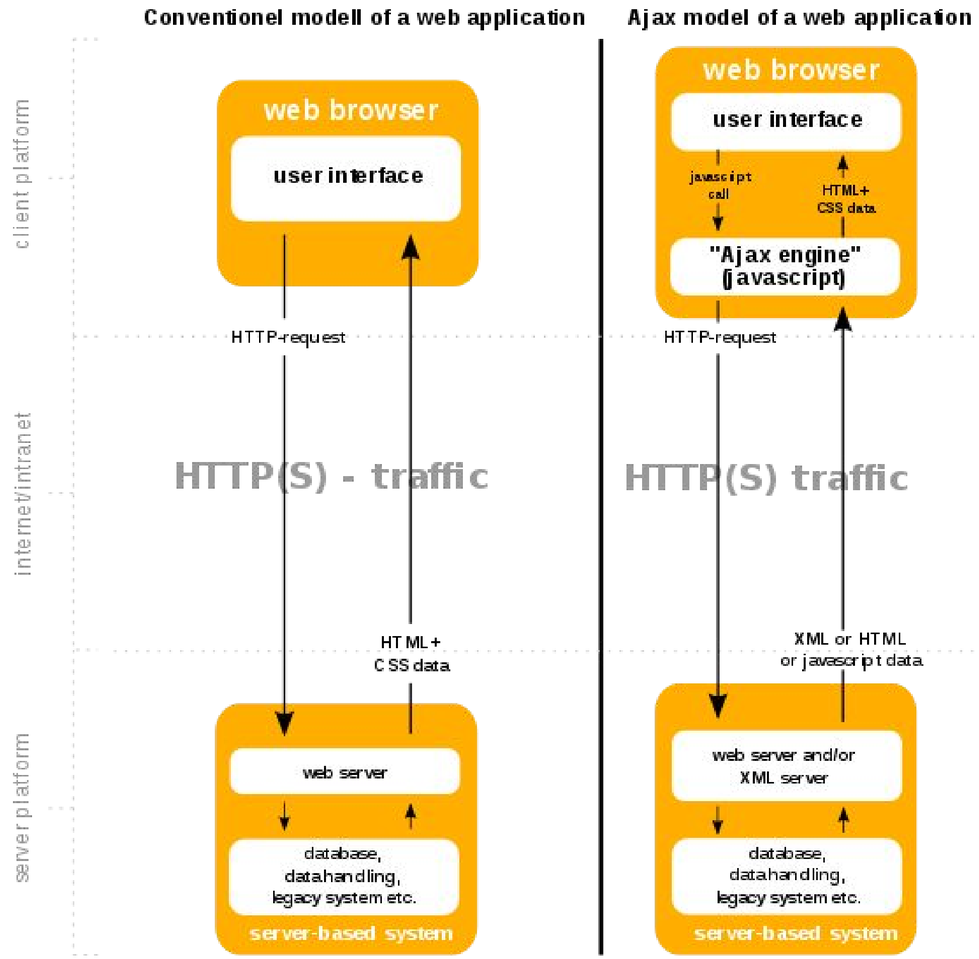
2

Why AJAX

Why AJAX

- Web applications can send and retrieve data from a server asynchronously (in the background) without interfering with the display and behavior of the existing page.
- AJAX decouples the data interchange layer from the presentation layer
- Allows for Web pages to change content dynamically without the need to reload the entire page.

AJAX vs Standard Form



3

AJAX Family

AJAX Family

AJAX cannot work independently. It is used in combination with other technologies to create interactive webpages.

JavaScript

- Loosely typed scripting language.
- JavaScript function is called when an event occurs in a page.
- Glue for the whole AJAX operation.

AJAX Family

AJAX cannot work independently. It is used in combination with other technologies to create interactive webpages.

DOM

- API for accessing and manipulating structured documents.
- Represents the structure of XML and HTML documents.

CSS

- Allows for a clear separation of the presentation style from the content and may be changed programmatically by JavaScript

AJAX Family

AJAX cannot work independently. It is used in combination with other technologies to create interactive webpages.

XMLHttpRequest

- JavaScript built-in object that performs asynchronous interaction with the server.

3

AJAX Steps

AJAX Steps

- 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.

AJAX Steps

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();">.

AJAX Steps

The XMLHttpRequest Object is Created

```
var ajaxRequest;

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) {
                  alert("Your browser broke!");
            return false;
         }
      }
   }
}
```

AJAX Steps

XMLHttpRequest Object is Configured

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);
}
```

AJAX Steps

Making Asynchronous 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;

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

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

AJAX Steps

Webserver Returns the Result Containing XML Document

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

```
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>");
   }
}
```

AJAX Steps

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. This function will receive the result from the server and will do the required processing.
- 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 = ...;
...
}
```

AJAX Steps

The HTML DOM is Updated

Final step - HTML page will be updated.

- 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
```

4

XMLHttpRequest Object

XMLHttpRequest Object

- XMLHttpRequest object is the key to AJAX.
- XMLHttpRequest (XHR) is an API that can be used by JavaScript, JScript, VBScript, and other web browser scripting languages to transfer and manipulate XML data to and from a webserver using HTTP
- Establishes an independent connection channel between a webpage Client-Side and Server-Side.
- Data returned from XMLHttpRequest calls will be provided by back-end databases.
- XMLHttpRequest can be used to fetch data in other formats (JSON or plain text.)

XMLHttpRequest Object

XMLHttpRequest Methods

- abort() - Cancels the current request.
- getAllResponseHeaders() - Returns the complete set of HTTP headers as a string.
- getResponseHeader(headerName) - Returns the value of the specified HTTP header.
- open(method, URL)
- open(method, URL, async)
- open(method, URL, async, userName)

XMLHttpRequest Object

XMLHttpRequest Methods

- open(method, URL, async, userName, password) -
  - Specifies the method, URL, and other optional attributes of a request.
  - method - can have a value of "GET", "POST", or "HEAD".
  - async - specifies whether the request should be handled asynchronously or not.
- send(content) - Sends the request.
- setRequestHeader(label, value) - Adds a label/value pair to the HTTP header to be sent.

XMLHttpRequest Object

XMLHttpRequest Properties

- **onreadystatechange** - An event handler for an event that fires at every state change.
- **readyState** - Defines the current state of the XMLHttpRequest object.

XMLHttpRequest Object

- **readyState**

  = 0 - After create the XMLHttpRequest object, but before call the open() method.

  = 1 - After call the open() method, but before call send().

  = 2 - After call send().

  = 3 - After the browser has established a communication with the server, but before the server has completed the response.

  = 4 - After the request has been completed, and the response data has been completely received from the server.

XMLHttpRequest Object

- **responseText** - Returns the response as a string.
- **responseXML** - Returns the response as XML. Returns an XML document object, which can be examined and parsed using the W3C DOM node tree methods and properties.
- **status** - Returns the status as a number (e.g., 404 for "Not Found" and 200 for "OK").
- **statusText** - Returns the status as a string (e.g., "Not Found" or "OK").

5

AJAX Drawbacks

AJAX Drawbacks

- **Complexity is increased**
  - Server-side developers will need to understand that presentation logic will be required in the HTML client pages as well as in the server-side logic.
  - Page developers must have JavaScript technology skills.

- **AJAX-based applications can be difficult to debug, test, and maintain**
  - JavaScript is hard to test - automatic testing is hard.
  - Weak modularity in JavaScript.

AJAX Drawbacks

- **No support of XMLHttpRequest in old browsers**
  - Iframe will help.

- **JavaScript technology dependency and incompatibility**
  - Must be enabled for applications to function.
  - Still some browser incompatibilities exist.

- **JavaScript code is visible to a hacker**

a. Poorly designed JavaScript code can invite security problems.

Thanks!

Any questions?