AJAX

AJAX

- AJAX stands for **A**synchronous **J**avaScript **A**nd **X**ML.
- AJAX is a type of programming made popular in 2005 by Google (with Google Suggest).
- AJAX is not a new programming language, but a new way to use existing standards.
- With AJAX you can create better, faster, and more user-friendly web applications.
- AJAX is based on JavaScript and HTTP requests.

AJAX is Based on Web Standards

- AJAX is based on the following web standards:
  - **JavaScript**
  - **XML**
  - **HTML**
  - **CSS**

AJAX is About Better Internet Applications

- Web applications have many benefits over desktop applications
  - they can reach a larger audience
  - are easier to install and support
  - easier to develop.
- Internet applications are not always as "rich" and user-friendly as traditional desktop applications.
- With AJAX, Internet applications can be made richer and more user-friendly.

Java Script Based Web applications

- To request a database information from a server we do the following
  - Make an html form
  - Use GET or POST method
  - Click Submit button and wait for the response from server
  - Next a new page is loaded with the results
    - Full page refresh
  - The same cycle continues resulting in a slower application
  - Makes less user friendly

AJAX Uses HTTP Requests

- With AJAX, JavaScript communicates directly with the server, through the JavaScript **XMLHttpRequest** object
- With an HTTP request, a web page can make a request to, and get a response from a web server - without reloading the page.
- The user will stay on the same page, and he or she will not notice that scripts request pages, or send data to a server in the background

AJAX - Browser Support

- The keystone of AJAX is the **XMLHttpRequest** object
- Different browsers use different methods to create the XMLHttpRequest object.
  - Internet Explorer uses an **ActiveXObject**,
  - other browsers uses the built-in JavaScript object called **XMLHttpRequest**.

```
<html>

<body>

<script type="text/javascript">
    function ajaxFunction() {
    var xmlHttp; try {
    // Firefox, Opera 8.0+, Safari
    xmlHttp=new XMLHttpRequest();
    } catch (e) {
    // Internet Explorer
    try {
    xmlHttp=new ActiveXObject("Msxml2.XMLHTTP");
    } catch (e) {
    try {
    xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
    } catch (e) {
    alert("Your browser does not support AJAX!");
    return false;
    }
    }
    }
    }
    </script>
```

three important properties of the XMLHttpRequest object

The **`onreadystatechange`** Property

- After a request to the server, we need a function that can receive the data that is returned by the server.
- The onreadystatechange property stores the function that will process the response from a server.
- The following code defines an empty function and sets the onreadystatechange property at the same time:

```
xmlHttp.onreadystatechange=function() {
 // The code goes here!
}
```

The readyState Property

- The `readyState` property holds the status of the server's response.

- Each time the readyState changes, the *onreadystatechange* function will be executed

- The possible values for the readyState property

State	Description
0	The request is not initialized
1	The request has been set up
2	The request has been sent
3	The request is in process
4	The request is complete

The readyState Property

```
xmlHttp.onreadystatechange=function() {
        if(xmlHttp.readyState==4) {

        // Get the data from the server's response

        }

}
```

The responseText Property

- The data sent back from the server can be retrieved with the ***responseText*** property.

```
xmlHttp.onreadystatechange=function() {
    if(xmlHttp.readyState==4) {

     document.myForm.time.value=xmlHttp.responseText;

    }

}
```

- Alternatively the XML data sent by the Server can be received using the ***responseXML*** property

AJAX - Request a Server

- To send off a request to the server, we use the **open()** method and the **send()** method.
- The **open()** method takes three arguments.
  - The first argument defines which method to use when sending the request (GET or POST).
  - The second argument specifies the URL of the server-side script.
  - The third argument specifies that the request should be handled asynchronously.
  - The send() method sends the request off to the server.

```
xmlHttp.open("GET",“urlp-attern",true);

xmlHttp.send(null);
```