ANMOL GOYAL

Ajax Tutorial : A Beginner's Guide

A Step By Step Process

TABLE OF CONTENT

AJAX Tutorial

AJAX, is a web development technique for creating interactive web applications.

If you know JavaScript, HTML, CSS, and XML, then you need to spend just one hour to start with AJAX.

Audience

This tutorial will be useful for web developers who want learn how to create interactive webpages as well as improve their speed and usability using AJAX.

Prerequisites

It is highly recommended that you are familiar with HTML and JavaScript before attempting this tutorial.

What is AJAX?

- AJAX stands for **A**synchronous **Ja**vaScript and **X**ML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.
- Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.
- Conventional web applications transmit information to and from the server using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.
- With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.
- XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
- AJAX is a web browser technology independent of web server software.
- A user can continue to use the application while the client program requests information from the server in the background.
- Intuitive and natural user interaction. Clicking is not required, mouse movement is a sufficient event trigger.
- Data-driven as opposed to page-driven.

Rich Internet Application Technology

AJAX is the most viable Rich Internet Application (RIA) technology so far. It is getting tremendous industry momentum and several tool kit and frameworks are emerging. But at the same time, AJAX has browser incompatibility and it is supported by JavaScript, which is hard to maintain and debug.

AJAX is Based on Open Standards

AJAX is based on the following open standards:

- Browser-based presentation using HTML and Cascading Style Sheets (CSS).
- Data is stored in XML format and fetched from the server.
- Behind-the-scenes data fetches using XMLHttpRequest objects in the browser.
- JavaScript to make everything happen.

AJAX - Technologies

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.

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.

XMLHttpRequest

- JavaScript object that performs asynchronous interaction with the server.

AJAX - Examples

Here is a list of some famous web applications that make use of AJAX.

Google Maps

A user can drag an entire map by using the mouse, rather than clicking on a button.

- http://maps.google.com/

Google Suggest

As you type, Google will offer suggestions. Use the arrow keys to navigate the results.

- http://www.google.com/webhp?complete=1&hl=en

Gmail

Gmail is a webmail, built on the idea that email can be more intuitive, efficient and useful.

- http://gmail.com/

Yahoo Maps (new)

Now it's even easier and more fun to get where you're going!

- http://maps.yahoo.com/

Difference in AJAX and Conventional CGI Program

Try these two examples one by one and you will feel the difference. While trying AJAX example, there is not discontinuity and you get the response very quickly, but when you try the standard GCI example, you would have to wait for the response and your page also gets refreshed.

AJAX Example:

* =

Standard Example:

* =

NOTE: We have given a more complex example in AJAX Database

AJAX - Browser Support

All the available browsers cannot support AJAX. Here is a list of major browsers, that support AJAX.

- Mozilla Firefox 1.0 and above.
- Netscape version 7.1 and above.
- Apple Safari 1.2 and above.
- Microsoft Internet Explorer 5 and above.
- Konqueror.
- Opera 7.6 and above.

When you write your next application, do consider the browsers that do not support AJAX.

NOTE: When we say that a browser does not support AJAX, it simply means that the browser does not support creation of Javascript object XMLHttpRequest object.

Writing Browser Specific Code

The Simplest way to make your source code compatible with a browser is to use *try...catch* blocks in your JavaScript.

```
<html>
<body>
  <script language="javascript" type="text/javascript"> <!--
  //Browser Support Code
  function ajaxFunction(){
    var ajaxRequest;  // The variable that makes Ajax possible!

    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;

                                            }

                                            }

                                            }

                                            }

  //-->
  </script>

  <form name='myForm'>
    Name: <input type='text' name='username' > <br > Time: <input type='text' name='time' /> </form>

</body>
</html>
```

In the above JavaScript code, we try three times to make our XMLHttpRequest object. Our first attempt:

- *ajaxRequest = new XMLHttpRequest();*

It is for Opera 8.0+, Firefox, and Safari browsers. If it fails, we try two more times to make the correct object for an Internet Explorer browser with:

- *ajaxRequest = new ActiveXObject("Msxml2.XMLHTTP");*
- *ajaxRequest = new ActiveXObject("Microsoft.XMLHTTP");*

If it doesn't work, then we can use a very outdated browser that doesn't support *XMLHttpRequest*, which also means it doesn't support Ajax.

Most likely though, our variable *ajaxRequest* will now be set to whatever *XMLHttpRequest* standard the browser uses and we can start sending data to the server. The step-wise AJAX workflow is explained in the next chapter.

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; if (!target) target =
document.getElementById("userid"); var url = "validate?id=" + escape(target.value);
ajaxRequest.open("GET", url, true); ajaxRequest.send(null); }
```

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("true"); }
  else
                                        {
    response.setContentType("text/xml"); response.setHeader("Cache-Control", "no-cache");
response.getWriter().write("false"); }
                                        }
```

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"); var messageText;

  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.childNodes[0]); }
  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.

AJAX - XMLHttpRequest

The XMLHttpRequest object is the key to AJAX. It has been available ever since Internet Explorer 5.5 was released in July 2000, but was not fully discovered until AJAX and Web 2.0 in 2005 became popular.

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, establishing an independent connection channel between a webpage's Client-Side and Server-Side.

The data returned from XMLHttpRequest calls will often be provided by back-end databases. Besides XML, XMLHttpRequest can be used to fetch data in other formats, *e.g.* JSON or even plain text.

You already have seen a couple of examples on how to create an XMLHttpRequest object.

Listed below is listed are some of the methods and properties that you have to get familiar with.

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) open(method, URL, async, userName, password) Specifies the method, URL, and other optional attributes of a request.

  The method parameter can have a value of "GET", "POST", or "HEAD". Other HTTP methods, such as "PUT" and "DELETE" (primarily used in REST applications) may be possible.

  The "async" parameter specifies whether the request should be handled asynchronously or not. "true" means that the script processing carries on after the send() method without waiting for a response, and "false" means that the script waits for a response before continuing script processing.
- **send(content)**

  Sends the request.
- **setRequestHeader(label, value)**

  Adds a label/value pair to the HTTP header to be sent.

XMLHttpRequest Properties

- **onreadystatechange**

  An event handler for an event that fires at every state change.

- **readyState**

  The readyState property defines the current state of the XMLHttpRequest object.

  The following table provides a list of 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 completed.

  readyState = 0 After you have created the XMLHttpRequest object, but before you have called the open() method.

  readyState = 1 After you have called the open() method, but before you have called send().

  readyState = 2 After you have called send().

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

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

- **responseText**

  Returns the response as a string.

- **responseXML**

  Returns the response as XML. This property 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").

AJAX - Database Operations

To clearly illustrate how easy it is to access information from a database using AJAX, we are going to build MySQL queries on the fly and display the results on "ajax.html". But before we proceed, let us do the ground work. Create a table using the following command.

NOTE: We are assuming you have sufficient privilege to perform the following MySQL operations

```
CREATE TABLE 'ajax_example' (
   'name' varchar(50) NOT NULL,
   'age' int(11) NOT NULL,
   'sex' varchar(1) NOT NULL,
   'wpm' int(11) NOT NULL,
   PRIMARY KEY  ('name')
)
```

Now dump the following data into this table using the following SQL statements:

```
INSERT INTO 'ajax_example' VALUES ('Jerry', 120, 'm', 20); INSERT INTO 'ajax_example' VALUES ('Regis', 75, 'm', 44); INSERT INTO 'ajax_example' VALUES ('Frank', 45, 'm', 87); INSERT INTO 'ajax_example' VALUES ('Jill', 22, 'f', 72); INSERT INTO 'ajax_example' VALUES ('Tracy', 27, 'f', 0); INSERT INTO 'ajax_example' VALUES ('Julie', 35, 'f', 90);
```

Client Side HTML File

Now let us have our client side HTML file, which is ajax.html, and it will have the following code: <html>

```
<body>
<script language="javascript" type="text/javascript"> <!--
//Browser Support Code
function ajaxFunction(){
  var ajaxRequest;  // The variable that makes Ajax possible!
  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;

                                                          }

                                                          }

                                                          }


  // Create a function that will receive data // sent from the server and will update // div section in the same
page.
  ajaxRequest.onreadystatechange = function(){

    if(ajaxRequest.readyState == 4){
      var ajaxDisplay = document.getElementById('ajaxDiv'); ajaxDisplay.innerHTML =
ajaxRequest.responseText; }

                                                          }


  // Now get the value from user and pass it to // server script.
  var age = document.getElementById('age').value; var wpm = document.getElementById('wpm').value;
var sex = document.getElementById('sex').value; var queryString = "?age=" + age ; queryString +=
"&wpm=" + wpm + "&sex=" + sex; ajaxRequest.open("GET", "ajax-example.php" + queryString, true);
ajaxRequest.send(null);

                                                  }

//-->
</script>

<form name='myForm'>

  Max Age: <input type='text' id='age' > <br > Max WPM: <input type='text' id='wpm' > <br > Sex:
  <select id='sex'>
    <option value="m">m</option> <option value="f">f</option> </select>
  <input type='button' onclick='ajaxFunction()' value='Query MySQL'/> </form>
<div id='ajaxDiv'>Your result will display here</div> </body>
</html>
```

NOTE: The way of passing variables in the Query is according to HTTP standard and have formA.

URL?variable1=value1;&variable2=value2; The above code will give you a screen as given below: **NOTE**: This is dummy screen and would not work Max Age:

Max WPM:

Sex:

Your result will display here in this section after you have made your entry.

NOTE: This is a dummy screen.

Server Side PHP File

Your client-side script is ready. Now, we have to write our server-side script, which will fetch age, wpm, and sex from the database and will send it back to the client. Put the following code into the file "ajax-example.php".

```
<?php
$dbhost = "localhost";
$dbuser = "dbusername";
$dbpass = "dbpassword";
$dbname = "dbname";

//Connect to MySQL Server
mysql_connect($dbhost, $dbuser, $dbpass);
//Select Database
mysql_select_db($dbname) or die(mysql_error());
// Retrieve data from Query String
$age = $_GET['age'];
$sex = $_GET['sex'];
$wpm = $_GET['wpm'];



// Escape User Input to help prevent SQL Injection $age = mysql_real_escape_string($age);
$sex = mysql_real_escape_string($sex);
$wpm = mysql_real_escape_string($wpm);

//build query
$query = "SELECT * FROM ajax_example WHERE sex = '$sex'"; if(is_numeric($age))
   $query .= " AND age <= $age"; if(is_numeric($wpm))
   $query .= " AND wpm <= $wpm";
//Execute query
$qry_result = mysql_query($query) or die(mysql_error()); //Build Result String
$display_string = "<table>"; $display_string .= "<tr>"; $display_string .= "<th>Name</th>";
$display_string .= "<th>Age</th>"; $display_string .= "<th>Sex</th>"; $display_string .= "
<th>WPM</th>"; $display_string .= "</tr>"; // Insert a new row in the table for each person returned
while($row = mysql_fetch_array($qry_result)){
   $display_string .= "<tr>"; $display_string .= "<td>$row[name]</td>"; $display_string .= "<td>$row[age]
</td>"; $display_string .= "<td>$row[sex]</td>"; $display_string .= "<td>$row[wpm]</td>";
$display_string .= "</tr>"; }

echo "Query: " . $query . "<br />"; $display_string .= "</table>"; echo $display_string;
?>
```

Now try by entering a valid value (e.g., 120) in *Max Age* or any other box and then click Query MySQL button.

Max Age:

Max WPM:

Sex:

Your result will display here in this section after you have made your entry.

If you have successfully completed this lesson, then you know how to use MySQL, PHP, HTML, and Javascript in tandem to write AJAX applications.

AJAX - Security

AJAX Security: Server Side

- AJAX-based Web applications use the same server-side security schemes of regular Web applications.
- You specify authentication, authorization, and data protection requirements in your web.xml file (declarative) or in your program (programmatic).
- AJAX-based Web applications are subject to the same security threats as regular Web applications.

AJAX Security: Client Side

- JavaScript code is visible to a user/hacker. Hacker can use JavaScript code for inferring server-side weaknesses.
- JavaScript code is downloaded from the server and executed ("eval") at the client and can compromise the client by mal-intended code.
- Downloaded JavaScript code is constrained by the sand-box security model and can be relaxed for signed JavaScript.

AJAX - Issues

AJAX is growing very fast and that is the reason that it contains many issues with it. We hope with the passes of time, they will be resolved and AJAX will become ideal for web applications. We are listing down a few issues that AJAX currently suffers from.

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.
- Lack of design patterns or best practice guidelines yet.

Toolkits/Frameworks are not mature yet

- Most of them are in beta phase.

No standardization of the XMLHttpRequest yet

- Future version of IE will address this.

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

- Poorly designed JavaScript code can invite security problems.

AJAX - Quick Guide

What is AJAX?

- AJAX stands for **A**synchronous **Ja**vaScript and **X**ML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.
- Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.
- Conventional web applications transmit information to and from the sever using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.
- With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.
- XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
- AJAX is a web browser technology independent of web server software.
- A user can continue to use the application while the client program requests information from the server in the background.
- Intuitive and natural user interaction. Clicking is not required, mouse movement is a sufficient event trigger.
- Data-driven as opposed to page-driven.

Rich Internet Application Technology

AJAX is the most viable Rich Internet Application (RIA) technology so far. It is getting tremendous industry momentum and several tool kit and frameworks are emerging. But at the same time, AJAX has browser incompatibility and it is supported by JavaScript, which is hard to maintain and debug.

AJAX is Based on Open Standards

AJAX is based on the following open standards:

- Browser-based presentation using HTML and Cascading Style Sheets (CSS).
- Data is stored in XML format and fetched from the server.
- Behind-the-scenes data fetches using XMLHttpRequest objects in the browser.
- JavaScript to make everything happen.

AJAX - Technologies

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.

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.

XMLHttpRequest

- JavaScript object that performs asynchronous interaction with the server.

AJAX - Examples

Here is a list of some famous web applications that make use of AJAX.

Google Maps

A user can drag an entire map by using the mouse, rather than clicking on a button.

- http://maps.google.com/

Google Suggest

As you type, Google will offer suggestions. Use the arrow keys to navigate the results.

- http://www.google.com/webhp?complete=1&hl=en

Gmail

Gmail is a webmail, built on the idea that email can be more intuitive, efficient and useful.

- http://gmail.com/

Yahoo Maps (new)

Now it's even easier and more fun to get where you're going!

- http://maps.yahoo.com/

Difference in AJAX and Conventional CGI Program

Try these two examples one by one and you will feel the difference. While trying AJAX example, there is not discontinuity and you get the response very quickly, but when you try the standard GCI example, you would have to wait for the response and your page also gets refreshed.

AJAX Example:

* =

Standard Example:

0 * 0 = 0

NOTE: We have given a more complex example in AJAX Database.

AJAX - Browser Support

All the available browsers cannot support AJAX. Here is a list of major browsers, that support AJAX.

- Mozilla Firefox 1.0 and above.
- Netscape version 7.1 and above.
- Apple Safari 1.2 and above.
- Microsoft Internet Explorer 5 and above.
- Konqueror.
- Opera 7.6 and above.

When you write your next application, do consider the browsers that do not support AJAX.

NOTE: When we say that a browser does not support AJAX, it simply means that the browser does not support creation of Javascript object XMLHttpRequest object.

Writing Browser Specific Code

The Simplest way to make your source code compatible with a browser is to use *try...catch* blocks in your JavaScript.

```
<html>
<body>
<script language="javascript" type="text/javascript"> <!--
//Browser Support Code
function ajaxFunction(){
  var ajaxRequest;  // The variable that makes Ajax possible!

  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;

                                                        }

                                                        }

                                                        }

                                                        }

//-->
</script>
<form name='myForm'>
  Name: <input type='text' name='username' > <br > Time: <input type='text' name='time' /> </form>
</body>
</html>
```

In the above JavaScript code, we try three times to make our XMLHttpRequest object. Our first attempt:

- *ajaxRequest = new XMLHttpRequest();*

It is for Opera 8.0+, Firefox, and Safari browsers. If it fails, we try two more times to make the correct object for an Internet Explorer browser with:

- *ajaxRequest = new ActiveXObject("Msxml2.XMLHTTP");*
- *ajaxRequest = new ActiveXObject("Microsoft.XMLHTTP");*

If it doesn't work, then we can use a very outdated browser that doesn't support *XMLHttpRequest,* which also means it doesn't support Ajax.

Most likely though, our variable *ajaxRequest* will now be set to whatever *XMLHttpRequest* standard the browser uses and we can start sending data to the server. The step-wise AJAX workflow is explained in the next chapter.

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; if (!target) target =
document.getElementById("userid"); var url = "validate?id=" + escape(target.value);
ajaxRequest.open("GET", url, true); ajaxRequest.send(null); }
```

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("true"); }
  else

{

    response.setContentType("text/xml"); response.setHeader("Cache-Control", "no-cache");
response.getWriter().write("false"); }

}
```

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"); var messageText;

  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.childNodes[0]); }
  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.

AJAX - XMLHttpRequest

The XMLHttpRequest object is the key to AJAX. It has been available ever since Internet Explorer 5.5 was released in July 2000, but was not fully discovered until AJAX and Web 2.0 in 2005 became popular.

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, establishing an independent connection channel between a webpage's Client-Side and Server-Side.

The data returned from XMLHttpRequest calls will often be provided by back-end databases. Besides XML, XMLHttpRequest can be used to fetch data in other formats, *e.g.* JSON or even plain text.

You already have seen a couple of examples on how to create an XMLHttpRequest object.

Listed below is listed are some of the methods and properties that you have to get familiar with.

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) open(method, URL, async, userName, password) Specifies the method, URL, and other optional attributes of a request.

  The method parameter can have a value of "GET", "POST", or "HEAD". Other HTTP methods, such as "PUT" and "DELETE" (primarily used in REST applications) may be possible.

  The "async" parameter specifies whether the request should be handled asynchronously or not. "true" means that the script processing carries on after the send() method without waiting for a response, and "false" means that the script waits for a response before continuing script processing.
- **send(content)**

  Sends the request.
- **setRequestHeader(label, value)**

  Adds a label/value pair to the HTTP header to be sent.

XMLHttpRequest Properties

- **onreadystatechange**

  An event handler for an event that fires at every state change.

- **readyState**

  The readyState property defines the current state of the XMLHttpRequest object.

  The following table provides a list of 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 completed.

  readyState = 0 After you have created the XMLHttpRequest object, but before you have called the open() method.

  readyState = 1 After you have called the open() method, but before you have called send().

  readyState = 2 After you have called send().

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

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

- **responseText**

  Returns the response as a string.

- **responseXML**

  Returns the response as XML. This property 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").

AJAX - Database Operations

To clearly illustrate how easy it is to access information from a database using AJAX, we are going to build MySQL queries on the fly and display the results on "ajax.html". But before we proceed, let us do the ground work. Create a table using the following command.

NOTE: We are assuming you have sufficient privilege to perform the following MySQL operations

```
CREATE TABLE 'ajax_example' (
   'name' varchar(50) NOT NULL,
   'age' int(11) NOT NULL,
   'sex' varchar(1) NOT NULL,
   'wpm' int(11) NOT NULL,
   PRIMARY KEY  ('name')
)
```

Now dump the following data into this table using the following SQL statements:

```
INSERT INTO 'ajax_example' VALUES ('Jerry', 120, 'm', 20); INSERT INTO 'ajax_example' VALUES ('Regis', 75, 'm', 44); INSERT INTO 'ajax_example' VALUES ('Frank', 45, 'm', 87); INSERT INTO 'ajax_example' VALUES ('Jill', 22, 'f', 72); INSERT INTO 'ajax_example' VALUES ('Tracy', 27, 'f', 0); INSERT INTO 'ajax_example' VALUES ('Julie', 35, 'f', 90);
```

Client Side HTML File

Now let us have our client side HTML file, which is ajax.html, and it will have the following code: <html>

```
<body>
<script language="javascript" type="text/javascript"> <!--
//Browser Support Code
function ajaxFunction(){
  var ajaxRequest;  // The variable that makes Ajax possible!
  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;

                                                       }

                                                       }

                                                       }


  // Create a function that will receive data // sent from the server and will update // div section in the same
page.
  ajaxRequest.onreadystatechange = function(){

    if(ajaxRequest.readyState == 4){
      var ajaxDisplay = document.getElementById('ajaxDiv'); ajaxDisplay.innerHTML =
ajaxRequest.responseText; }

                                                       }


  // Now get the value from user and pass it to // server script.
  var age = document.getElementById('age').value; var wpm = document.getElementById('wpm').value;
var sex = document.getElementById('sex').value; var queryString = "?age=" + age ; queryString +=
"&wpm=" + wpm + "&sex=" + sex; ajaxRequest.open("GET", "ajax-example.php" + queryString, true);
ajaxRequest.send(null);

}

//-->
</script>
<form name='myForm'>
  Max Age: <input type='text' id='age' > <br > Max WPM: <input type='text' id='wpm' > <br > Sex:
<select id='sex'>
  <option value="m">m</option> <option value="f">f</option> </select>
  <input type='button' onclick='ajaxFunction()' value='Query MySQL'/> </form>
<div id='ajaxDiv'>Your result will display here</div> </body>
</html>
```

NOTE: The way of passing variables in the Query is according to HTTP standard and have formA.

URL?variable1=value1;&variable2=value2; The above code will give you a screen as given below: **NOTE**: This is dummy screen and would not work Max Age:

Max WPM:

Sex:

Your result will display here in this section after you have made your entry.

NOTE: This is a dummy screen.

Server Side PHP File

Your client-side script is ready. Now, we have to write our server-side script, which will fetch age, wpm, and sex from the database and will send it back to the client. Put the following code into the file "ajax-example.php".

```
<?php
$dbhost = "localhost";
$dbuser = "dbusername";
$dbpass = "dbpassword";
$dbname = "dbname";

  //Connect to MySQL Server
mysql_connect($dbhost, $dbuser, $dbpass);
  //Select Database
mysql_select_db($dbname) or die(mysql_error());
  // Retrieve data from Query String
$age = $_GET['age'];
$sex = $_GET['sex'];
$wpm = $_GET['wpm'];



  // Escape User Input to help prevent SQL Injection $age = mysql_real_escape_string($age);
$sex = mysql_real_escape_string($sex);
$wpm = mysql_real_escape_string($wpm);

  //build query
$query = "SELECT * FROM ajax_example WHERE sex = '$sex'"; if(is_numeric($age))
$query .= " AND age <= $age"; if(is_numeric($wpm))
$query .= " AND wpm <= $wpm";
  //Execute query
$qry_result = mysql_query($query) or die(mysql_error()); //Build Result String
$display_string = "<table>"; $display_string .= "<tr>"; $display_string .= "<th>Name</th>";
$display_string .= "<th>Age</th>"; $display_string .= "<th>Sex</th>"; $display_string .= "
<th>WPM</th>"; $display_string .= "</tr>"; // Insert a new row in the table for each person returned
while($row = mysql_fetch_array($qry_result)){
  $display_string .= "<tr>"; $display_string .= "<td>$row[name]</td>"; $display_string .= "<td>$row[age]
</td>"; $display_string .= "<td>$row[sex]</td>"; $display_string .= "<td>$row[wpm]</td>";
$display_string .= "</tr>"; }
echo "Query: " . $query . "<br />"; $display_string .= "</table>"; echo $display_string;
?>
```

Now try by entering a valid value (e.g., 120) in *Max Age* or any other box and then click Query MySQL button.

Max Age:

Max WPM:

Sex:

Your result will display here in this section after you have made your entry.

If you have successfully completed this lesson, then you know how to use MySQL, PHP, HTML, and Javascript in tandem to write AJAX applications.

AJAX - Security

AJAX Security: Server Side

- AJAX-based Web applications use the same server-side security schemes of regular Web applications.
- You specify authentication, authorization, and data protection requirements in your web.xml file (declarative) or in your program (programmatic).
- AJAX-based Web applications are subject to the same security threats as regular Web applications.

AJAX Security: Client Side

- JavaScript code is visible to a user/hacker. Hacker can use JavaScript code for inferring server-side weaknesses.
- JavaScript code is downloaded from the server and executed ("eval") at the client and can compromise the client by mal-intended code.
- Downloaded JavaScript code is constrained by the sand-box security model and can be relaxed for signed JavaScript.

AJAX - Issues

AJAX is growing very fast and that is the reason that it contains many issues with it. We hope with the passes of time, they will be resolved and AJAX will become ideal for web applications. We are listing down a few issues that AJAX currently suffers from.

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.
- Lack of design patterns or best practice guidelines yet.

Toolkits/Frameworks are not mature yet

- Most of them are in beta phase.

No standardization of the XMLHttpRequest yet

- Future version of IE will address this.

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

- Poorly designed JavaScript code can invite security problems.

Developer's Best Practices

What is Practice?

When I'm saying "Practice", what does it mean? I would say:

- Practice is **a habit.**
- Practice is **a routine.**
- Practice **does not need to remember.**
- Practice **comes by practicing.**
- Practice **needs dedication and commitment.**

There are thousands of examples which you think about practice. I can list few for your understanding.

Shooting, Driving, Writing

Any of the above listed skills comes from practice. When initially you start driving, you need to remember each step and you think twice before taking any action, but once you "have good practice" of driving, then you do not need to remember any step. It becomes your habit and routine, for example, your feet goes automatically at brake if you see a red light but definitely it comes from practising a lot and needs a lot of dedication and commitment.

One of the most important attributes of practice is that it **forces you not to divert** from what you used to do.

There could be a driver but would you assume him an efficient driver if he is driving at a speed of 20 miles per hours and meeting with accidents so frequently and bringing lots of scratches in the car on a daily basis?

Software development is also not different than other skills like shooting, writing or driving. To become a **successful** software developer you need lot of practice, dedication and commitment.

Through this small article, I'm going to tell you few major best software developer's practices, which you may find useful. So let's start...

Code Reading & Reading

Best Practice 1-Keep Reading Existing Software Source Code

Let me ask you few basic questions before we start with one of the most important best practices required for a software developer.

- Do you read movie magazines?
- Do you read newspapers?
- Do you read roadside advertisements?
- Do you read junk written here and there?
- Do you just read....?

Definitely your answer will be positive but if I ask you one more question in the series:

Do you read Software Source Code?

Only few software developers will have positive answer because reading and understanding an existing software source code is the most boring task. If you are one of them who feels reading software source code is a boring task, then you are missing one of the most important best practices, which a software developer should have in his/her life.

If you want to become a novelist, can you just start writing novels? I would say 100% no!!, you definitely need to read hundreds of novels before you start writing **GOOD** novels. If you want to become a movie script writer, can you start writing good movie scripts until you have gone through various good movie scripts?, again my answer would be no!!

So, if you want to write a good software code, then how it will be possible for you to write a good source code without reading tons of source codes? Even if you will write something, then how would you and know which the best is?

Reading source code written by others gives you opportunity to criticize the mistakes done in writing that code. You will be able to identify the mistakes other software developers have done in their source code which you should not repeat.

There are many attributes of software codes (indentation, comments, history header, function structure, etc.), which you will learn by reading existing code, specially, a code written by well-experienced software developers. Spend some time in reading others' source code and I'm sure you would be able to write **BEAUTIFUL** source code in few days or few weeks and you will be able to fix the mistakes, which you were doing so far in writing the source code.

One thing to experiment, just go in the past and check the code you had written few years ago, you will definitely laugh....because you are always improving by doing practice.

Documentation is the Key

Best Practice 2 - Complete your documents before next step

I was so passionate to write source code even without completely understanding and documenting the requirements. Design document and test cases documentation were nowhere in the software development life cycle ....there was direct jump to the coding.

At later stages I found myself in big trouble and soon I realized **Documentation is the Key** to become successful software developer, tester or architect.

Before you start developing small or big software, you should have answer for the following questions:

- Where is the Requirements Specification?
- Where is the Impact Analysis Document?
- Where is the Design Document?
- Have you documented all the assumptions, limitations properly?
- Have you done review of all the documents?
- Did you get sign off on all the documents from all the stakeholders?

Once you have positive answers for all the above questions, you are safe and ready to proceed for the coding. Many organizations would have strict rules to be followed, but others would not have. Best practice is to complete all the required documentation and take appropriate approvals before proceeding for the software coding.

What you learn today, prepares you for tomorrow!

So, again it is one of the best practices to have documentation as much as possible. Few important documents, which will prepare you for future are:

- Design Approaches
- Tips and Tricks
- Special functions, commands and instructions
- Lessons learnt
- Peculiar situations
- Debugging methods
- Best Practices
- Anything which can help you in future

Keeping documents electronically does not cost you. So let's start maintaining required documentation.

Follow the Standards

Best Practice 3 - Follow the defined standards, don't create it

Most of the standard software organizations maintain their coding standards. These standards would have been set up by well-experienced software developers after spending years with software development. This is equivalent to following footsteps of great people left behind them.

If your organization does not have any standard, then I would suggest to search on internet for coding standards off different programming languages and you will find many. A coding standard would fix the rules about various important attributes of the code, few are listed below:

- File Naming convention
- Function & Module Naming convention
- Variable Naming convention
- History, Indentation, Comments
- Readability guidelines
- List of do's and don'ts

But once defined, start following the defined standard instead of creating or changing them every day. I would definitely say:

Source code is your BABY!

So keep it clean, consistent and beautiful. When I say beautiful, it really means beautiful. If your code looks beautiful, then it would be easy for others to read and understand it. If you will keep changing coding rules everyday, then after few days you, yourself would not be able to read and understand the code written by you.

Write to be Reviewed

Best Practice 4 - Code should be written to be reviewed

While writing your software code, keep in mind that someone is going to review your code and you will have to face criticism about one or more of the following points but not limited to:

- Bad coding
- Not following standard
- Not keeping performance in mind
- History, Indentation, Comments are not appropriate.
- Readability is poor
- Open files are not closed
- Allocated memory has not been released
- Too many global variables.
- Too much hard coding.
- Poor error handling.
- No modularity.
- Repeated code.

Keep all the above-mentioned points in your mind while coding and stop them before they jump in your source code. Once you are done with your coding, go for a self-review atleast once. I'm sure, a self-review would help you in removing 90% problems yourself.

Once you are completely done with your coding and self review, request your peer for a code review. I would strongly recommend to accept review comments happily and should be thankful to your code reviewers about the comments. Same time, it is never good to criticize any source code written by someone else. If you never did it, try it once and check the coder's expression.

Accept criticism but don't criticize

Poorly written source code teaches you to write good source code provided you take it positively and learn a lesson from it.

Your target should be to stop the bugs at first place and create a BUG-FREE code. Think like a tester, so that you should have a challenge for the testers.

Testing is the Religion

Best Practice 5 - Testing to be followed like a religion

Testing is mandatory after every small or big change no matter how tight schedule you have or you just changed a small comment inside the code, you have testing due for the changed code.

There is nothing like trust while developing software, no matter how expert or how senior you are in writing source code, you would have to perform testing for each and every change you did in the code.

- Tight schedule, no compromise.
- Changed just a comment, still you have to test it.
- Changed just a variable name, testing has to be done.
- If you feel lazy...it's too dangerous.

If you don't want to follow it? You will be in trouble!

Celebrate every bug you find

Yes, you should not feel unhappy if you or another tester finds a bug in your software source code. Following are the enough reasons to celebrate this important discovery:

- Bugs are your enemies, so you have killed one.
- Now your software is having one bug less.
- Mistakes are good as long as they are not repeating.
- What you learn today, prepares you for tomorrow

Same time, do not criticize any developer in case any bug arises in his/her code because so far at least I do not know any programmer, who can write bug-free source code in the world, second this is one of the reasons we have a separate phase in SDLC (Software Development Life Cycle) which we call post production support (or support & maintenance).

Keep the Assets Safely

Best Practice 6 - Keep your Code and Documents Safe

A smart developer keeps habit of taking daily backup of the produced artifacts, otherwise machine crash can crash you as well. You should keep your artifacts at your local machine as well as another secure machine, so that in case of machine crash, you can continue with the saved copy of the source code or documents.

If you have the habit of taking daily backup then in worst scenario you may lose at most one-day effort, but if you take weekly or monthly backup, then there is a risk of losing whole-week or whole-month effort, and you will face biggest disappointment you ever had.

Multiple copies create confusion

This is true that having backup is one of the most important best practices, but it should be maintained in well managed way as you can use tags like name, date and time of the backup, version, *etc*. If you have multiple copies of the same source code or document, then it will create confusion and it would be difficult to identify latest code or document.

It is strongly recommended to use proper source code version control system. There are many source code version control software applications available for free (like SCCS, CVS, Subversion etc.) which you can use to store different versions of the software. But while using a source code control system, follow the rules below:

- Always take source code from the version control system.
- Always assign a new version to every change.
- Always put source code back into control system.

Password sharing is strictly prohibited

- Love, affection, friendship and relationship are on top of everything, but never embrace anybody asking for password.
- If you are sticking to first point, then why you would share your password with anyone if you are not asking from anybody.
- Keep changing it on a frequent basis and it's good if you have some logic to drive your passwords, otherwise during your long vacation, you will forget them.

Handy Tools & Techniques

Best Practice 7 - Keep your Tools & Techniques Handy

I remember an instance when I wanted to find out **debug** keyword in all the C++ files available in various directories and sub-directories, it took me 30 minutes to find the command, but finally, I kept a note of the command, and whenever I'm in need, I use it without wasting a second.

```
$find . -name \*.cpp -exec grep -q "debug" '{}' \; -print
```

So, I made it one of the best practices to keep such commands and tools handy so that they can be used anytime without doing any R&D and to save valuable time. Better to maintain a text file having all such frequently used commands and create its link at desktop.

Few Essential Tools

It depends on what type of programming, coding you are doing but following are few of the essential tools, which should be readily available with a software developer:

- A good text editor to write and edit the program.
- A nice debugger to debug the program.
- A memory detector in case you are using dynamic memory allocation.
- Putty to connect to a remote machine.
- WinSCP or FileZilla to ftp files on a remote machine.
- IDE (Integrated Development Environment) for rapid development.

Always keep adding new tools & techniques in your box

Make sure you keep applying latest patches of your tools and utilities and same time I will suggest to clean unwanted software from your computer as they unnecessarily make your computer slow and you never know if any one of them is having a security hole, which can expose your computer to the outside world.

Eager to Learn

Best Practice 8 -Leave the ego behind, Be eager to learn

We always learn from books and nowadays from internet. But IT is such a field, where we learn a lot from our colleagues. They are our best references, but there are software developers, who either feel shy in asking their doubts or are not thankful to others, so ultimately when they ask next time, they get zero answer.

IT is vast and nobody can have complete knowledge on any subject. Everyday, we come across different problems. So Ask...Don't feel shy if you don't know X.

I'm not suggesting you to bother someone unreasonably and asking for spoon feeding to learn anything. NO, be polite, thankful, directly come to the point, understand and support others.

New technologies are coming everyday

If you want to sustain in the market, then you would have to keep yourself updated with latest IT tools, and technologies. Following are the few sources:

- Technical Forums over the internet.
- Technical magazines on various IT subjects.
- Technical Bulletin Boards
- Conferences, Trainings and Workshops
- Latest versions of old tools and packages, languages, etc

Stress Management

As you grow at your position, your responsibilities increase in multiples of your salary increment which definitely brings lots of stress in your personal and professional life. As such, there is no formula to get rid of your stress and you will find fat books and training programs to teach you how to manage stress, but I believe an open communication is the biggest weapon, which can help you up to some extent to relieve yourself from big stress.

Let's identify root cause of the stress

You are a software professional, you should know how to debug a problem. Similar way, stress is a problem for you and you have to debug it, You should find out why it's coming to you and what root causes are. Let's take few examples, which may be the cause of stress in your day-2-day life:

- Workload is too much and you are not able to handle it properly.
- You had been assigned to a module, which is not ready though deadline has arrived.
- So far you really do not know what exactly you have to do and how exactly you have to do?
- You had developed a code, which got deleted by mistake or not working at final moment.
- You are leader of the team, but team is not doing so great and ultimately delivery is getting delayed.
- Though you have enough time to deliver, but meanwhile, you planned for a travel as well which may cause delay in your delivery.

Communication, Communication.....& Just Communication

For you none of them should be a problem if you take them in professional way. Let's pick up any of the above-mentioned points, for example, first point where you feel overloaded and not able to finish your task within office hours.

Simply set up a small meeting with your manager and put the facts in front of him, mentioning your current assignments, bottlenecks and reasons why you feel you are overloaded. You can request him to share one more resource with you or to give you more time. I'm sure your manager will listen on this and will help you if he needs a good delivery from you. You need to plan how you are going to convince your manager about it and make him realize that what you are saying is correct.

Similar way any of the mentioned issues can be resolved with proper communication with your manager and if it's not working with manager, then many organizations give you chance to talk to your higher management and take your issues to them. So in case your problem does not get resolved, you take it to higher level, but you need to be careful because it can be a little sensitive as none of the managers would like you to bypass him and talk to his boss directly. But yes it could be your last try if nothing is working.

One more important issue is poor prioritization of the work. If you can discuss priority of the work with your manager then you can handle scheduling all the tasks one after another. You can give some extra time after office hours or during weekend to relieve yourself.

Personal vs Professional

Try to identify whether you are not able to work properly because you have some personal issues and they are impacting your professional life. In such case, your family is the best one, who can help you in resolving your personal issues. You can share your personal issues with your close friends or family, spouse, *etc.* and get them fixed as soon as possible. If it is growing very serious, then it's better to talk to your manager and explain him the situation and try to get few days off and then fix your personal issues and come back to catch up with your work.

Stress could be momentarily

Aha, it's part of everybody's life and you should not get stressed due to little overload, little delayed delivery or some minor issues happening around you. Let's make them part of your day-2-day life and keep moving on. So, let's do a little more extra overtime to finish your delivery, take little help from your friends, be ready to listen few comments from your manager.

Make sure you are not repeating problems, and problems are also not repeating with you, and if this is the case, then it’s time to take action and find out its solution.

Few more quick remedies

Try to use any of the following if they relieve you from stress:

- Some exercise
- Little or more yoga, meditation
- Morning walk
- Evening movie
- Pass some time with your friends, family, spouse, kids.
- Avoid sitting for long time and have a coffee break at work, read magazine, newpapers, internet browsing, using stress-removal toys.

Bottom line is that you should not keep quiet and keep creating a volcano, which will erupt someday later and produce lots of damages. Be communicative, be transparent and be honest. Keep in mind, if you are under stress, then your productivity will reduce unexpectedly, so try to keep yourself healthy, happy and active.....

Managing Managers

As a software developer i.e., programmer, one of the most challenging issues you face is related to managing your manager and his/her expectations. You may come across various complicated and confusing situations, which are unexpected and difficult to resolve and ultimately you become a victim of unnecessary stress we discussed in last chapter. Following examples may be few of them:

- Your manager does not give you due respect and value.
- One of your peers does not deliver still he is always in news and getting appreciation notes.
- There has been some misunderstanding between you and your manager.
- A cold war is running between you and your manager.
- From last few years, your manager did not think about your promotion or salary revision.
- You think your manager is not capable enough and it is difficult to convince him/her.
- It does not matter what you deliver, still you have to get negative feedback.
- Your manager does not like you because of XYZ reasons.

Just think what is going on between you and your manager, I am sure you will be able to add your issue in the above list. That's the first and most important task to identify why there is an issue. It could be X... Y... or Z....

Managers are always correct....

Yes, if you are disagreeing with me then its obvious why you are in trouble. Try to recall when you were a kid, and your parents always stopped you from doing X...Y...or ...Z activities and they used to emphasize on certain things, which you never liked in your childhood. But now will definitely say, Alas! it would have been so good for us if we would have done the way parents instructed. Now if you are inline with me then it means you found out half of the solution of your problems.

So, crux of the discussion is the given attention what your manager is asking for and do the way he suggested. Your ultimate goal should be to make your manager happy and few of the points can help you in achieving this:

- Try to give fast deliveries, it does not matter if you put your effort during weekend.
- Reduce your complaints about things around you.
- Reduce your demands in terms of salary revisions or promotions.
- Do not lose a chance to present your work to your manager, does not matter its small or big but your manager should be aware of what you do.
- Be neutral as much as possible, do not criticize any other peer in front of the manager.
- Take things positive done or presented by your manager, as I said they are always right.
- You will have to observe why your manager likes any particular resource and try to inline with that resource.
- Never try to think your manager is inferior to you, that may be the case but it's not allowed to think like that, otherwise by doing so you can create problem for yourself.

Managers always need great resources

Great, so you have adopted all the points mentioned above, now you will say I will give fast and clean deliveries by putting my honest efforts during weekends and holidays, still I should not demand for salary hikes or promotion, why????

My answer is yes, you do it and things will come automatically, just have patience. You will hardly need to demand for anything once you make your manager realize that you are one of the brightest resources and you are most important for the project. Once you achieve this, your manager would never like to lose you, and now it's your time to enjoy your work and working environment.

If still you find things are not moving as per expectation, then you have to initiate a healthy discussion with your manager and ask for the reasons why you are not getting hikes, and promotions. It could be some other HR-related issues or project budget, *etc.* You can ask for improvement areas if needed and set expectations accordingly, but again, your cycle will start from the above-mentioned activities.

If you have some misunderstandings with your manager then call for a meeting with the manager and accept the mistakes you have done if any and clarify the things which went wrong and give an assurance to take care of such incidents in future.

Many things depend on situation and you need to be smart enough to understand the situation and act accordingly. All the very best.

Career Planning

Today's professional life is very dynamic and to move along with it we need a proper career planning. When you start your career as a software developer, you really do not know how exactly you will perform in the industry, though you have confidence that whatever you do, will be done in the best way. So take some time to investigate yourself, what are your major strengths and weaknesses and based on at least 3-4 years of experience you can come up with different options:

- Do you want to continue as software developer forever, which could be a very good option and there are many people, who love coding forever.
- If you are very good in designing software components and your past designs have been appreciated a lot, then you can think to go in technical side and become software architect.
- If you are very good in managing things, have good command over people and have great convincing abilities, then you can think of going towards management role, which will start with leading a small team.
- If you are very good in managing things and at the same time you have great architectural sense, then you can think of becoming techno-manager, where you will keep contributing in designing components and will manage team and projects.

Whatever it is, you must be aware of where do you want to reach. Once you are sure about this, you should start working in the same direction starting from your project preference till your trainings and certifications. Your current organization may not be giving you appropriate opportunity to reach your desired destination then you can wait for right time and make a move to other good organization but it should not be very frequent. I have seen guys doing monkey jump every six months from one organization to another one just because of little hike and it’s being done without a proper thinking and proper planning but these fellows do not know what they are losing in long run.

You can discuss about your career path with your manager/line manager and most of the organizations have standard career path defined for their employees, so you can check if it suits your interests and work accordingly.

When to make a move?

This is very interesting question that when I should move to another organization, but I can not answer it in simple words. You know your career path and if your current organization is enough to put you at your final destination, then why do you want to leave it. Leaving an organization just because of few bucks is never a good reason, even leaving organizations too frequently is not a good idea though you are getting great position and big hikes, this is simply because you are losing your credibility and none of the good companies will rely on you because you are always behind money and position, so who knows when you will leave them.

If you have some internal HR or Management issues within your organization, then try to resolve them because you never know your next organization may have even bigger issues than your current organization. You can discuss your issues with your manager, director or with HR and resolve them gracefully.

If you see no further growth and good career options in the current organization and same time your learning curve got a saturation, then its time to make a shift to another organization. There may be a situation when you are not getting a fat salary and having great position in your current organization but you are learning a lot, which will add a lot of value in your resume and your career, then better to stick with the current organization until your learning is over.

Summary

To summarize, it is easy to do just coding but to become a good programmer i.e., software developer needs some hard work and dedication in doing lot of practice. There could be a list of thousands of best practices, which can be listed down by veteran software developers but let us eat the quantity, which we can digest easily.

Just keep your list small but follow them strictly throughout your developer's life.

Tomorrow your kids are going to use it...

I'm sure, tomorrow same Book will be used by your kids if luckily they are software developers i.e., programmers or engineers, so let us improve it all together. If you like this Book, then share it with others and write me back about the improvement.