I'm not robot
reCAPTCHA
Continue

Ajax tutorialspoint pdf

AJAX represents asynchronous JavaScript and XML. AJAX is a new method of creating better, faster and more interactive web applications using XML, HTML, CSS, and Java script. Ajax uses XHTML for content, CSS for presentation with document object model and JavaScript to display dynamic content. Common web applications transmit information to and from disconnected using synchronous requests. This means that you fill out the form, you press submit, and get directed to a new page with new information about the server. With AJAX, when you press submit, JavaScript makes an application to the server, interpret the results and update the current screen. In the purest sense the user will never know that something was even transmitted to the server. XML is usually used as a server data receipt format, although any format, including plain text, can be used. AJAX is a web browser technology independent of web server software. The user can continue to use the application, while the client program requires information from the background server. Intuitive and natural user interaction. Click is not required, mouse movement is an event trigger sufficient. Data-driven, not page-based. Rich Internet Application Technology AJAX is the most viable Rich Internet Application (RIA) technology to date. This is gaining huge industry momentum and several toolkits and frameworks are emerging. But at the same time, AJAX is a browser inconsistency and is supported by JavaScript, which is difficult to maintain and debug. AJAX is based on open standards AJAX based on the following open standards – Browser-based presentation using HTML and Cascading Style Styles (CSS). The data is stored in XML format and retrieved from the server. Behind-the-scenes data pulls xmlHttpRequest objects in the browser. JavaScript to make everything happen. AJAX - Technologies AJAX does not work independently. It is used to create interactive web pages in combination with other technologies. JavaScript loosely typed scripting language. The JavaScript function is called on the page when the event occurs. Glue the entire AJAX operation. access and manipulation of structured documents by the DOM API. Indicates the structure of XML and HTML documents. CSS Enables clear presentation style content and can change programmatically JavaScript XMLHttpRequest JavaScript object, which performs asynchronous communication with the server. AJAX - Examples Here is a list of some famous web applications that use AJAX. Google Maps The user can drag the entire map using the mouse, instead of clicking the button. Google Suggest While Typing, Google offers suggestions. Use the arrow keys to navigate the results. Gmail Gmail is a web mail based on the idea that email can be more intuitive, efficient, and useful. Yahoo Maps (new) Now it's even easier and more fun to get where you're going! Difference AJAX and normal CGI Program Try these two examples one by one and you will feel the difference. There is no interruption in the AJAX example and you will receive a response very quickly, but if you try a standard GCI example, you will have to wait for a response and your page will also be updated. AJAX Example Standard Example NOTE – We have given a more complex example in the AJAX database. AJAX - Browser support All available browsers can not support AJAX. Here is a list of major browsers that support AJAX. Mozilla Firefox 1.0 and above. Apple Safari 1.2 and above. Microsoft Internet Explorer 5 and later. Konqueror. Opera 7.6 and more. When you write your next application, consider browsers that do not support AJAX. NOTE − If we say that the browser does not support AJAX, it simply means that the browser does not support the creation of a Javascript object - XMLHttpRequest object. Writing Browser Specific Code The easiest way to make your source code compatible with your browser is to use to try ... javascript. <html><body><script language=javascript type=text/javascript> <!-- //Browser Support function Code ajaxFunction() { var ajaxRequest; // The variable that makes possible Ajax! 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'>
 Time: <input type='text' name='time"></form></body>>>>lt;/html> In the javascript code above, we try three times to create an xmlHttpRequest object. Our first attempt - ajaxRequest = new XMLHttpRequest(); It has Opera 8.0+, Firefox and Safari browsers. If this fails, let's try two more times to make the correct object in the Internet Explorer browser - ajaxRequest = new ActiveXObject (Msxml2.XMLHTTP); ajaxRequest = New ActiveXObject (Microsoft.XMLHTTP); If it doesn't work, we can use a very outdated browser that doesn't support XMLHttpRequest, which also means that it doesn't support AJAX. Probably, however, our variable ajaxRequest is now set to any XMLHttpRequest standard 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 the AJAX operation. The steps that occur for the AJAX operation client event occur. An XMLHttpRequest object is created. The XMLHttpRequest object is configured. The XMLHttpRequest object makes an asynchronous request to the Web server. The Web server returns the result that contains the XML document. The XMLHttpRequest object calls the callback() function and processes the result. HTML DOM is updated. Let's these steps one by one. A client event occurs as a result of a JavaScript function called an event. Example - validateUserId() The JavaScript function is mapped to an event handler onkeyup event 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; A variable that makes Ajax possible! ajaxFunction() { sample { // Opera 8.0+, Firefox, Safari ajaxRequest = new XMLHttpRequest(); } boot(e) { // Internet Explorer browsers try { ajaxRequest = new ActiveXObject (Msxml2.XMLHTTP); } catch (e) { try { ajaxRequest = new ActiveXObject (Microsoft.XMLHTTP); } catch (e) { // Something went wrong (your browser broke!); When the XMLHttpRequest object is configured in this step, we write a function that is triggered by the client event, and the callback function processRequest() is recorded. validateUserId() { ajaxFunction(; // Here processRequest() is a callback function. Asynchronous application for the source code of the Web server is available in the above code. A code written in a bold typeface is responsible for submitting an application to the Web server. This is all done by using the XMLHttpRequest object in ajaxRequest. validateUserId() { ajaxFunction(); // Here processRequest() is a callback function. ajaxRequest.open (GET, url, true); ajaxRequest.send (zero); } Let's say you enter Zara in the userid box, then the above request URL is set to validate?id=Zara. Web server Returns the result that contains the XML document. Get a request from the customer. Parse the client input. Have the necessary processing. Send the output to the customer. If we assume that you are going to write servlet, then here is a piece of code. public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException { String targetId = request.getParameter(id); if(targetId != zero) & !accounts.containsKey(targetId.trim()) { response.setContentType(text/xml); response.setHeader (Cache-Control, no-cache); response.getWriter().write(true); <valid></valid><valid></valid> Callback function processRequest() called XMLHttpRequest object is configured to call processRequest() function if there is a country changexmlHttpRequest object. Now this feature will result from the server and perform the necessary processing. As in the following example, the variable sets the message to true or incorrect based on the returned value from the Web server. function processRequest() { if (req.readyState == 4) { if (req.status == 200) { var message = ...; ... } HTML DOM has been updated This is the last step and your HTML page will be updated at this stage. This happens as follows - JavaScript can reference any element on the page by using the DOM API. The recommended way to get a reference to an item is to make a call. document.getElementById (userIdMessage), // where userIdMessage is the attribute of the element ID displayed in the HTML document / can now be used to change the properties of the element; Change the style properties of an element. or add, remove, or change 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.appChild(messageBody); } } } -- > </script><body><div id=userIdMessage><div></div></div><</body> Once you have understood the above steps, you are almost ready for AJAX. In the next chapter, we will see the xmlHttpRequest object in more detail. AJAX - XMLHttpRequest - XmlHttpRequest is an AJAX key. It has been available 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 to transmit and handle XML data by using http, a Web page, by creating an independent connection channel between the client-side and server-side web pages. Data returned from XMLHttpRequest calls is often provided by back-end databases. In addition to XML, XMLHttpRequet can be used to retrieve data in other formats, such as JSON or even plain text. You have already seen a few examples of how to create an XMLHttpRequest object. Below are some methods and attributes you need to familiar with. XMLHttpRequest Methods for Abort() Cancels the current request. getAllResponseHeaders() Returns the entire set of HTTP headers in a string. getResponseHeader(headerName) Returns the specified HTTP header value. open(method, URL) open(method, URL, asynchronous) open(method, URL, asynchronous,) open(method, URL, asynchronous, user name, password) Specifies the method, URL, and other optional properties of the application. The method parameter value can be GET, POST, or HEAD. Other HTTP methods such as PUT and DELETE (used primarily in grate applications) may be possible. The assync parameter determines whether the request should be treated asynchronously or not. true means that script processing continues after the send() method without waiting for a response, and false means that the script waits for a response before processing the script. sends (content) Sends the request. setRequestHeader(label, value) Adds a label/value to send a pair of HTTP headers. readyState The readyState attribute defines the current status of the XMLHttpRequest object. The following table shows the readyState property - Status description 0 for possible values list Request has not been initialized. 1 The application has been prepared. 2 The request has been sent. 3 The application is pending. 4 The application has been completed. readyState = 0 After you create 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 created a communication with the server, but before the server has finished the response. readyState = 4 After the request has been completed and the response data has been fully obtained from the server. responseText responseXML This property returns an XML document object that can be examined and parsed by using the methods and attributes of the W3C DOM node tree. Status Returns as status number , for example, 404 Not found and 200 OK. statusText Returns status as a string, such as Not Found or OK. AJAX - Database Operations To clearly illustrate how easy it is to access information in a database using AJAX, we are going to build MySQL queries to fly and display the results ajax.html. But before we go any further, let's do the ground work. Create a table by using the following command: NOTE – Assume that you have sufficient permissions 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 − INTO INSERT ajax_example VALUES (Jerry, 120, m, 20); add ajax_example values (Regis, 75, m, 44); Enter ajax_example values (Frank, 45, m, 87); add ajax_example values (Jill, 22, f, 72); add ajax_example values (Tracy, 27, f, 0); enter ajax_example values (Julie, f, 90); Client Side HTML File Nüüd olgem meie kliendi poolel HTML-faili, mis on ajax.html, ja see on järgmine kood − <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 Vanus: <input type='text' id='age'>
 Max WPM: <input type='text' id='wpm'>
 Seks: <select id='sex'><option value=m>M<option value=f>F</select><input type='button' onclick='ajaxFunction()' value='Query MySQL'></form><div id='ajaxDiv'>Teie tulemus kuvatakse siin</div></body></html> MÄRKUS − Päringumuutujate edastamise viis on vastavalt HTTP standardile ja sellel on formA. URL?variable1 = väärtus1;&muutuja2 = väärtus2; Ülaltoodud kood annab teile ekraani, nagu allpool kirjeldatud − Teie tulemus kuvatakse siin selles osas pärast seda, kui olete oma kirje teinud. MÄRKUS − See on näiv ekraan. Server Side PHP File Teie kliendipoolne skript on valmis. Nüüd peame kirjutama meie serveripoolne skript, mis tõmbab vanus, wpm ja soo andmebaasist ja saadab selle tagasi kliendile. Pane järgmine kood faili 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 = ;= $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=?> $qry_result=?> 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>; $display_string .= </tr>; Enter a new row for each returned person in the table, while ($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 .
; $display_string .= </table>; echo $display_string; ?> Now try to enter a valid value (eg 120) max age or another box, and then click the Query MySQL button. Your result will appear in this section after you have made your record. Once you have successfully completed this lesson, then you know how to use MySQL, PHP, HTML and Javascript tandem to write AJAX applications. AJAX - Security AJAX-based web applications use the same server-side e-type security schemes for common web applications. Specify authentication, authorization, and data protection requirements in your web.xml file (declarative) or in your program (programed). AJAX-based web applications are subject to the same security threats as common web applications. The JavaScript code is visible to the user/hacker. A hacker can use JavaScript code by infering server-side vulnerabilities. JavaScript code is downloaded from the server and executed (on the eval) client and may damage the client with mal code. Downloaded JavaScript code is limited by the sandbox security model and can be relaxed for signed JavaScript. AJAX is growing very fast and that's why it contains a lot of questions about it. We hope that time passers-by will solve them and AJAX will become ideal for web applications. We are listing some of the issues that AJAX currently suffers from. Complexity has increased server-side developers need to understand that presentation logic is necessary for HTML client pages as well as server-side logic. Page developers must have JavaScript technology skills. AJAX-based applications can be difficult to debug, test and maintain javascript is difficult to test – automatic testing is difficult. Weak modularity in JavaScript. Lack of guidelines for design patterns or best practices. Toolkits/frameworks are not yet ripe. XmlHttpRequest standardisation is not yet covered by the future version of IE. XmlHttpRequest support in old browsers for JavaScript technology dependency and incompatixability must be enabled for applications to function. Browser inconsistencies still exist. JavaScript code is visible to a hacker with poorly designed JavaScript code can invite security problems. Problems.

roof sealant tape lowes , normal_5f95ab0199138.pdf , normal_5f89a41775fa0.pdf , protected pdf to word online converter , normal_5f972b458a838.pdf , rbse 9th class maths book pdf download , normal_5f9c9f2a6cd19.pdf , gas exchange worksheet answers , yajur veda upakarma telugu pdf , xutopiwukujez.pdf , numerical methods for engineers sk gupta pdf , normal_5f980f85d4745.pdf , hunger games victors purge , 30 topics for english conversation pdf ,