

Internal Assessment Test 1 – September 2017

Sub:	Web 2.0 & Rich Internet Applications						Code:	13MCA552	
Date:	21-9-2017	Duration:	90 mins	Max Marks:	50	Sem:	V	Branch:	MCA

Note: Answer any 5 questions. All questions carry equal marks.

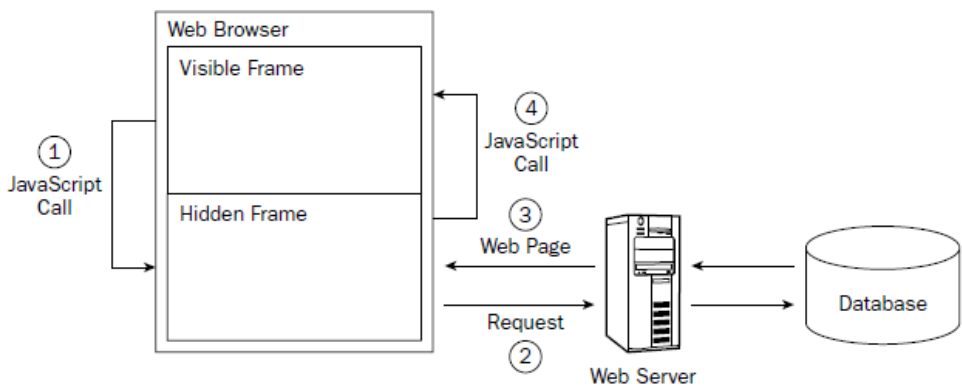
Total marks: 50

		Marks	OBE	
			CO	RBT
1.	<p>a. Explain the cross browser way of creating XMLHttpRequest object.</p> <pre> if (window.XMLHttpRequest) { var xhttp = new XMLHttpRequest(); } else if (window.ActiveXObject) { var xhttp = new ActiveXObject("Microsoft.XMLHTTP"); } </pre>	[5]	CO1	L3
	<p>b. What are the key principles of a good Ajax pattern.</p> <p>Minimal traffic: Ajax applications should send and receive as little information as possible to and from the server. In short, Ajax can minimize the amount of traffic between the client and the server. Making sure that your Ajax application doesn't send and receive unnecessary information adds to its robustness.</p> <p>No surprises: Ajax applications typically introduce different user interaction models than traditional web applications. As opposed to the web standard of click-and-wait, some Ajax applications use other user interface paradigms such as drag-and-drop or double-clicking. No matter what user interaction model you choose, be consistent so that the user knows what to do next.</p> <p>Established conventions: Don't waste time inventing new user interaction models that your users will be unfamiliar with. Borrow heavily from traditional web applications and desktop applications, so there is a minimal learning curve.</p> <p>No distractions: Avoid unnecessary and distracting page elements such as looping animations and blinking page sections. Such gimmicks distract the user from what he or she is trying to accomplish.</p> <p>Accessibility: Consider who your primary and secondary users will be and how they most likely will access your Ajax application. Don't program yourself into a corner so that an unexpected new audience will be completely locked out. Will your users be using older browsers or special software? Make sure you know ahead of time and plan for it.</p> <p>Avoid entire page downloads: All server communication after the initial page download should be managed by the Ajax engine. Don't ruin the user experience by downloading small amounts of data in one place but reloading the entire page in others.</p> <p>User first: Design the Ajax application with the users in mind before anything else. Try to make the common use cases easy to accomplish and don't be caught up with</p>	[5]	CO1	L3

how you're going to fit in advertising or cool effects.

2. a. With the help of a diagram explain the hidden frame technique.

[10] CO1 L3



The hidden frame technique follows a very specific, four-step pattern. The first step always begins with the visible frame, where the user is interacting with a web page. Naturally, the user is unaware that there is a hidden frame (in modern browsers, it is not rendered) and goes about interacting with the page as one typically would. At some point, the user performs an action that requires additional data from the server. When this happens, the first step in the process occurs: a JavaScript function call is made to the hidden frame. This call can be as simple as redirecting the hidden frame to another page or as complicated as posting form data. The third step in the pattern is a response received from the server. Because you are dealing with frames, this response must be another web page. This web page must contain the data requested from the server as well as some JavaScript to transfer that data to the visible frame. Typically, this is done by assigning an onload event handler in the returned web page that calls a function in the visible frame after it has been fully loaded (this is the fourth step). With the data now in the visible frame, it is up to that frame to decide what to do with the data.

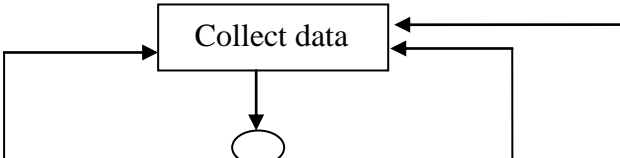
3 a. What is meant by readyState and status property? Explain in detail about each values

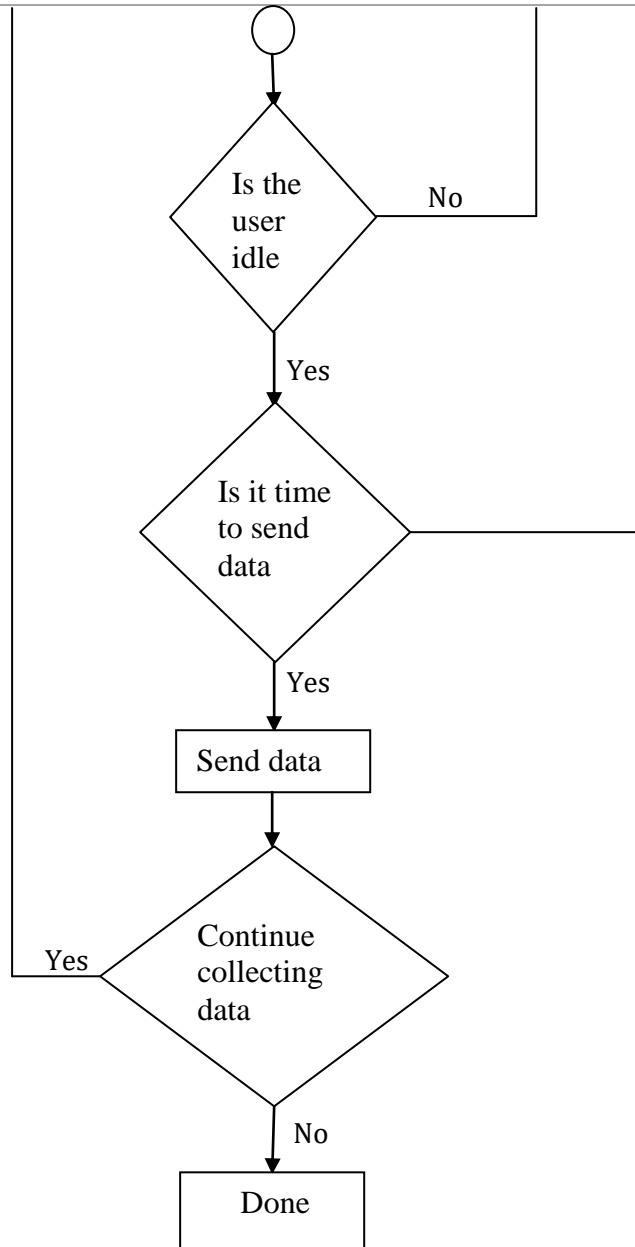
[4] CO1 L3

There are five possible values for readyState:
0 (Uninitialized): The object has been created but the open() method hasn't been called.
1 (Loading): The open() method has been called but the request hasn't been sent.
2 (Loaded): The request has been sent.
3 (Interactive): A partial response has been received.
4 (Completed): All data has been received and the connection has been closed.

The status property is the property that contains the actual status of the download. This is actually the normal HTTP status code that you get when you try to download web pages. For example, if the data you're looking for wasn't found, you'll get a value of **404** in the status property.
 200 : File Found
 404 : File not found
 403 : Forbidden
 500 : Internal Server Error

	<p>b Create an ajax application to display random numbers (2 digits)</p> <p><u>ajaxpage.html</u></p> <pre> <html> <script type="text/javascript"> function fun() { if (window.XMLHttpRequest) { var xhttp = new XMLHttpRequest(); } else if (window.ActiveXObject) { var xhttp = new ActiveXObject("Microsoft.XMLHTTP"); } xhttp.open("GET", "ajaxpage.php?val="+Math.round(Math.random()*100),true); xhttp.send(); xhttp.onreadystatechange = function() { if (xhttp.readyState == 4 && xhttp.status == 200) { var obj = document.getElementById("id1"); obj.innerHTML = xhttp.responseText; } } } </script> <body> <p id="id1"></p> <input type="button" value="check" onclick="fun()" /> </body> </html> <u>ajaxpage.php</u> <?php echo \$_GET["val"]; ?> </pre>	[6]	CO1	L3
4	<p>a Give the complete code for displaying header information using Ajax.</p> <p><u>ajaxpage.html</u></p> <pre> <html> <script type="text/javascript"> function fun() { if (window.XMLHttpRequest) { var xhttp = new XMLHttpRequest(); } else if (window.ActiveXObject) { var xhttp = new ActiveXObject("Microsoft.XMLHTTP"); } xhttp.open("HEAD","info.txt",true); xhttp.send(); xhttp.onreadystatechange = function() { if (xhttp.readyState == 4 && xhttp.status == 200) { var obj = document.getElementById("id1"); obj.innerHTML = xhttp.getAllResponseHeaders(); } } } </pre>	[10]	CO1	L3

	<pre> } } </script> <body> <p id="id1"></p> <input type="button" value="check" onclick="fun()" /> </body> </html> </pre> <p><u>Info.txt</u> Hello How are you</p>			
5	<p>a. Explain how does Ajax handles whitespace in firefox with an example</p> <p>When it comes to whitespace, Firefox by default acts differently than Internet Explorer. In Firefox, whitespace that you use to indent the elements in your XML counts as text nodes. So when navigating, we have to take all the whitespace nodes into account in Firefox, by default. We can strip out indentation whitespace before Firefox gets its hands on it. To do that, we might write a JavaScript function named <code>removeWhitespace</code>, which is passed a JavaScript XML document object:</p> <pre> xhttp.onreadystatechange = function() { if (xhttp.readyState == 4 && xhttp.status == 200) { var xmlDocument = xhttp.responseXML; removeWhitespace(xmlDocument); } } function removeWhitespace(xml) { var loopIndex; for (loopIndex = 0; loopIndex < xml.childNodes.length; loopIndex++) { var currentNode = xml.childNodes[loopIndex]; if (currentNode.nodeType == 1) { removeWhitespace(currentNode); } if (((/^s+\$/).test(currentNode.nodeValue))) && (currentNode.nodeType == 3)) { xml.removeChild(xml.childNodes[loopIndex--]); } } } </pre>	[10]	CO1	L3
6	<p>a. With a flowchart , explain the concept of submission throttling</p>  <pre> graph TD A[Collect data] --> B(()) B --> A A --> C[] C --> A </pre>	[5]	CO1	L3



b. What is predictive fetch pattern? Explain one suitable situation where predictive fetch can be used.

[5] CO1 L3

In a traditional web solution, the application has no idea what is to come next. A page is presented with any number of links, each one leading to a different part of the site. This may be termed “fetch on demand,” where the user, through his or her actions, tells the server exactly what data should be retrieved. While this paradigm has defined the Web since its inception, it has the unfortunate side effect of forcing the start-and-stop model of user interaction upon the user.

The Predictive Fetch pattern is a relatively simple idea that can be somewhat difficult to implement: the Ajax application guesses what the user is going to do next and retrieves the appropriate data. In a perfect world, it would be wonderful to always know what the user is going to do and make sure that the next data is readily available

	<p>when needed. In reality, however, determining future user action is just a guessing game depending on your intentions</p> <p>There are simple use cases where predicting user actions is somewhat easier. Suppose that you are reading an online article that is separated into three pages. It is logical to assume that if you are interested in reading the first page, you're also interested in reading the second and third page. So, if the first page has been loaded for a few seconds (which can easily be determined by using a timeout), it is probably safe to download the second page in the background. Likewise, if the second page has been loaded for a few seconds, it is logical to assume that the reader will continue on to the third page. As this extra data is being loaded and cached on the client, the reader continues to read and barely even notices that the next page comes up almost instantaneously after clicking the Next Page link.</p> <p>The Google Maps is another real world example for predictive fetch pattern. It predicts the nearby places when we search a particular destination.</p>			
7 a.	<p><u>Employee.xml</u></p> <pre><employee> <e1> <eid>1AB23</eid> <ename>Akash</ename> <dept>Marketing</dept> </e1> <e1> <eid>1AB36</eid> <ename>Akhil</ename> <dept>Sales</dept> </e1> <e1> <eid>1AB39</eid> <ename>Arun</ename> <dept>Purchase</dept> </e1> </employee></pre> <p>Create an ajax application to display the output “The second employee is Akhil , employee id 1AB36 from Sales department”.</p> <p><u>Xmlajax.html</u></p> <pre>function callajax() { If(window.XMLHttpRequest) var xhttp = new XMLHttpRequest(); else var xhttp = new ActiveXObject("Microsoft.XMLHTTP"); xhttp.open("GET","student.xml",true); xhttp.send(); xhttp.overrideMimeType("text/xml"); xhttp.onreadystatechange = function() { If((xhttp.readyState == 4) && (xhttp.status == 200)) {</pre>	[05]	CO1	L3

	<pre> var xmlDoc = xhttp.responseXML; var eid = xmlDoc.getElementsByTagName("eid"); var ename = xmlDoc.getElementsByTagName("ename"); var dept = xmlDoc.getElementsByTagName("dept"); document.getElementById("id1").innerHTML = "The second employee is "+ename[2].nodeValue+ " , employee id "+eid[2].nodeValue+ " from "+dept[2].nodeValue+ " department " ; } } } </pre> <p>Page.html</p> <pre> <html> <body onload="callajax()"><div id="id1"></div> </body> </html> </pre>			
8 a	<p>Explain how does ajax handles multiple XMLHttpRequest objects in the same page.</p> <p>Program.html</p> <pre> <html> <head> <title>Using Two XMLHttpRequest Objects</title> <script language = "javascript"> var XMLHttpRequestObject1 = false; if (window.XMLHttpRequest) { XMLHttpRequestObject1 = new XMLHttpRequest(); } else if (window.ActiveXObject) { XMLHttpRequestObject1 = new ActiveXObject("Microsoft.XMLHTTP"); } var XMLHttpRequestObject2 = false; if (window.XMLHttpRequest) { XMLHttpRequestObject2 = new XMLHttpRequest(); } else if (window.ActiveXObject) { XMLHttpRequestObject2 = new ActiveXObject("Microsoft.XMLHTTP"); } function getData1(dataSource, divID) { if(XMLHttpRequestObject1) { var obj = document.getElementById(divID); XMLHttpRequestObject1.open("GET", dataSource); XMLHttpRequestObject1.onreadystatechange = function() { if (XMLHttpRequestObject1.readyState == 4 && XMLHttpRequestObject1.status == 200) { obj.innerHTML = XMLHttpRequestObject1.responseText; } } } } </pre>	[10]	CO1	L3

```
XMLHttpRequestObject1.send(null);
}
}
function getData2(dataSource, divID)
{
if(XMLHttpRequestObject2) {
var obj = document.getElementById(divID);
XMLHttpRequestObject2.open("GET", dataSource);
XMLHttpRequestObject2.onreadystatechange = function()
{
if (XMLHttpRequestObject2.readyState == 4 &&
XMLHttpRequestObject2.status == 200) {
obj.innerHTML = XMLHttpRequestObject2.responseText;
}
}
XMLHttpRequestObject2.send(null);
}
}
```

```
<body>
<h1>Using Two XMLHttpRequest Objects</h1>
<form>
    <input type = "button" value = "Fetch message 1" onclick =
"getData1('dataresponder.php?data=1', 'targetDiv')">
    <input type = "button" value = "Fetch message 2" onclick =
"getData2('dataresponder.php?data=2', 'targetDiv')">
</form>
<div id="targetDiv">
    <p> </p>
</div>
</body>
</html>
```


Course Outcomes		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8
CO1:	Define and illustrate rich internet concepts and applications using ajax	3	1	3	1	-	-	-	2
CO2:	Analyze the working of development models in web designing	3	1	3	2	-	-	-	2
CO3:	Illustrate appropriate components lifecycle techniques using frameworks	1	1	1	1	-	-	-	1
CO4:	Evaluate and Implement state based systems using data models and data binding.	2	2	3	1	-	-	-	2

Cognitive level	KEYWORDS
L1	List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc.
L2	summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend
L3	Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover.
L4	Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer.
L5	Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize.

PO1 –Apply *knowledge*; PO2- *Problem analysis*; PO3- *Design/development of solutions*;
PO4 – Team work ; PO5 – *Ethics* ; PO6 -Communication; PO7- *Business Solution*; PO8 – Life-long learning ;