xmlHTTP.html

```
<html>
<head>
<title>A clock based on the server's time</title>
<script language="javascript" type="text/javascript">
function getTime(){
  if ( (typeof XMLHttpRequest) != 'undefined') {
    xmlhttp = new XMLHttpRequest();
    xmlhttp.open("GET", "time.php", true);
    xmlhttp.onreadystatechange = function(){
      if (xmlhttp.readyState==4) {
            var c = document.getElementById("clock");
            c.innerHTML = xmlhttp.responseText;
      }
    };
    xmlhttp.send(null);
  }else{
       alert('This browser does not support the
XMLHttpRequest object');
  }
  setTimeout("getTime()", 10000);   // every ten seconds
}

</script>
</head>
<body onLoad="getTime()">
<span id="clock" name="clock">
--:--:--
</span>
</body>
</html>
<!-- Reference: http://jibbering.com/2002/4/
httprequest.html -->
```

time.php

```
<?php

 echo date("h:m:s");

?>
```

The script "time.php" is a server-side script, which means that it is interpreted by the PHP processor on the server before its output is sent to the client that uses HTTP's GET method to access it. In other words, instead of delivering the bytes as is, the way HTML is delivered directly from the hard drive, this script is run first and its output will be sent instead. The result is that we can use the script on the server side to provide timely content for our Javascript-enhanced web page (left). The web page uses the XMLHttpRequest object to send a "behind the scenes" GET request to the server, which then gives it some data it can display in the "clock" span.

The getTime() function is called the first time from the onLoad event (body tag), and subsequently it registers itself to be called again in 10 seconds (10000 milliseconds). The XMLHttpRequest object has an event called "onreadystatechange" that we use to respond when the request is finished. We implement the event by attaching a function definition to it. That function will then be called when the request's "ready state" changes. It may have failed due to a bad URL or something like that, so in the function's body we check to see that the xmlhttp.readyState variable is 4 (for reasons only the programmers of the XMLHttpRequest can explain!). At this point we can take advantage of the handy xmlhttp.responseText property, which now contains whatever was sent from the server side script. We then put that into our "clock" span.