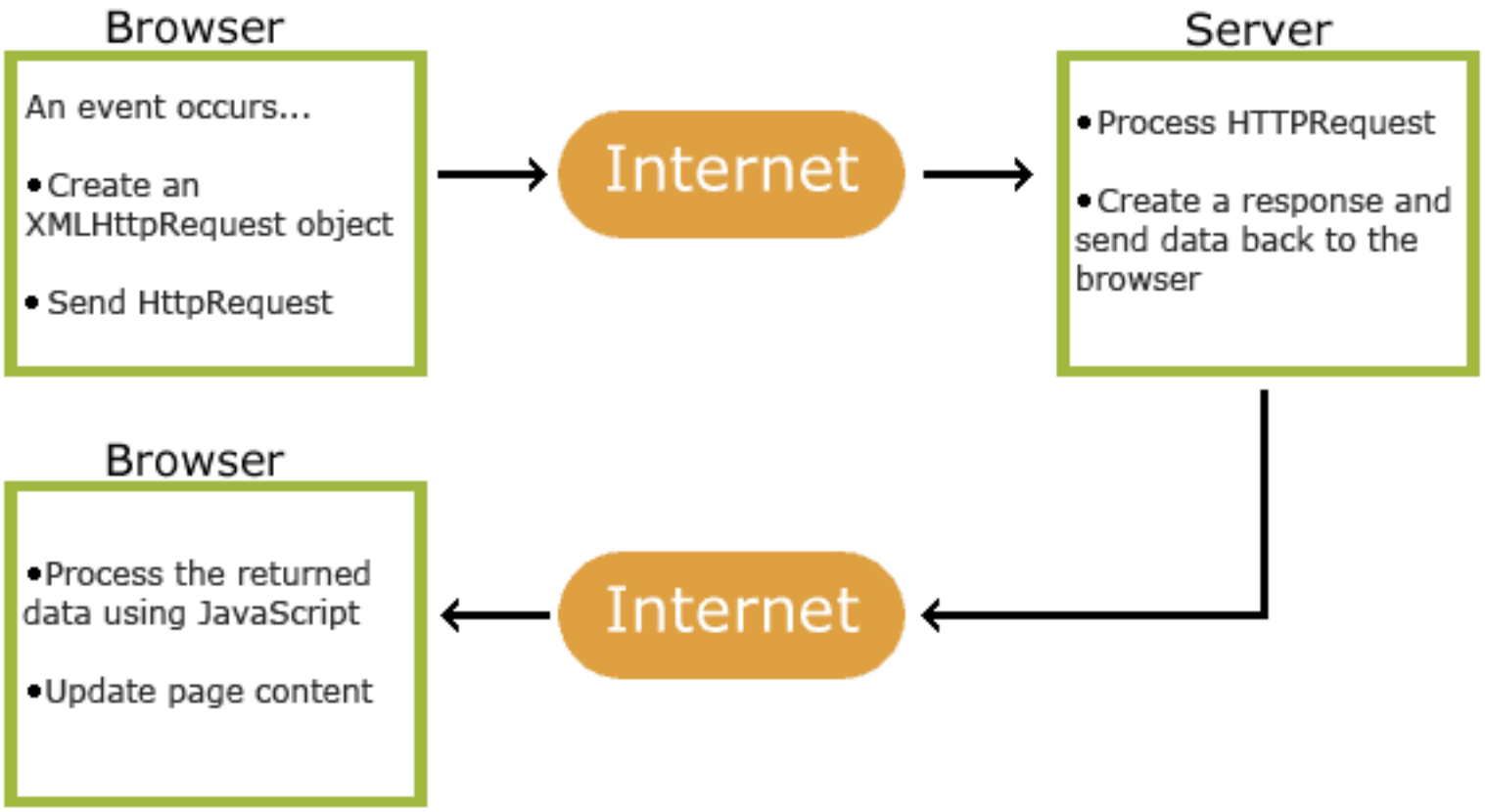
AJAX

What is AJAX

- Asynchronous Javascript And XML
  - allows the updating of a web page without doing a page reload
    - creates much nicer user experience
- AJAX is not really a technology by itself
  - combination of Javascript, XML and some server-side scripting to create the XML
    - server-side scripting could be done in PHP, .NET, Java Servlet or Java Server Page (JSP)

General Technique



Sending a request for a URL

- create xmlHttpRequest Object
  - All modern browsers (Chrome, IE7+, Firefox, Safari, and Opera) have a built-in XMLHttpRequest object:
    - objXMLHttp=new XMLHttpRequest()
  - Old versions of Internet Explorer (IE5 and IE6) use an ActiveX Object:
    - objXMLHttp=new ActiveXObject("Microsoft.XMLHTTP")
- create the URL
- tell the browser the name of the function to handle the response
- send the url to the server

example

```
var xhttp;
if (window.XMLHttpRequest) {
    xhttp = new XMLHttpRequest();
    } else {
    // code for IE5, IE6
    xhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
```

```
var url="servertime.php"
xmlHttp.onreadystatechange=stateChanged
xmlHttp.open("GET",url,true)
xmlHttp.send(null)
```

XMLHttpRequest Object

- Methods:
  - open(“method”,”URL”, async) sets a pending request
    - method: the type of request: GET or POST
    - url: the server (file) location
    - async: true (asynchronous) or false (synchronous)
  - send(content) – transmits the request to the server
    - empty content: use for GET
    - string content: use for POST

XMLHttpRequest Object

- GET or POST?
  - GET is simpler and faster than POST, and can be used in most cases.
  - However, always use POST requests when:
    - A cached file is not an option (update a file or database on the server).
    - Sending a large amount of data to the server (POST has no size limitations).
    - Sending user input (which can contain unknown characters), POST is more robust and secure than GET.

XMLHttpRequest Object

- Asynchronous - True or False?
  - With AJAX, the JavaScript does not have to wait for the server response, but can instead:
    - execute other scripts while waiting for server response
    - deal with the response when the response ready
  - async=true, specify a function to execute when the response is ready in the onreadystatechange event.
  - async=false is not recommended. JavaScript will NOT continue to execute, until the server response is ready. If the server is busy or slow, the application will hang or stop. Do NOT write an onreadystatechange function - just put the code after the send() statement.

(continued)

- Properties
  - onreadystatechange - event handler to use
  - readyState (0-uninitialized, 1-loading, 2-loaded, 3-interactive, 4- complete)
  - responseText – string version of the data returned
  - responseXML – DOM compatible document object returned by server
  - status – http response header code (200 – good)
  - statusText – string message of status code

The server-side script

- creates a “well formed XML document”
- sets the content type to text/xml
- can be written in any language
  - PHP
  - ASP
  - .NET
  - Java
  - JSP

sample PHP script

```
<?
   // a time document
   header("Content-type: text/xml");
   print("<time>");
   print("<currtime>".time()."</currtime>");
   print("</time>");
?>
```

stateChange

- when the document is received by the browser, control is transferred to where ever we told it to
  - xmlHttp.onreadystatechange=stateChanged
  - in this case the function named stateChanged

stateChanged

```
function stateChanged()
{
   if (xmlHttp.readyState==4 || xmlHttp.status==200)
     {
       //update the DOM with the data
       document.getElementById("time").innerHTML=xmlHttp.responseText
     }
}
```

XMLHttpRequest Object

- Additional methods:
  - abort() - stop the current request
  - getAllResponseHeaders - Returns complete set of headers (labels and values) as a string
  - getResponseHeader(“headerLabel”) – returns the string value of the requested header field
  - setRequestHeader(“label”,”value”) – sets label/value in the header

example

```
<!DOCTYPE html>
<html>
<body>

<p>The getResponseHeader() function is used to return specific header information from a resource, like length, server-type, content-type, last-modified, etc.</p>
<p>Last modified: <span id="demo"></span></p>

<button onclick="loadDoc('ajax_info.txt')">Get "Last-Modified" information</button>

<script>
function loadDoc(url) {
  var xhttp=new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200) {
      document.getElementById("demo").innerHTML = xhttp.getResponseHeader(“Last-Modified”);
    }
  };
  xhttp.open("GET", url, true);
  xhttp.send();
}
</script>

</body>
</html>
```