

Ajax Programming

Session 560556

Dr. William A. Hansen
Manta Technologies Inc.
whansen@mantatech.com



Copyright 2010. Manta Technologies Inc.

1

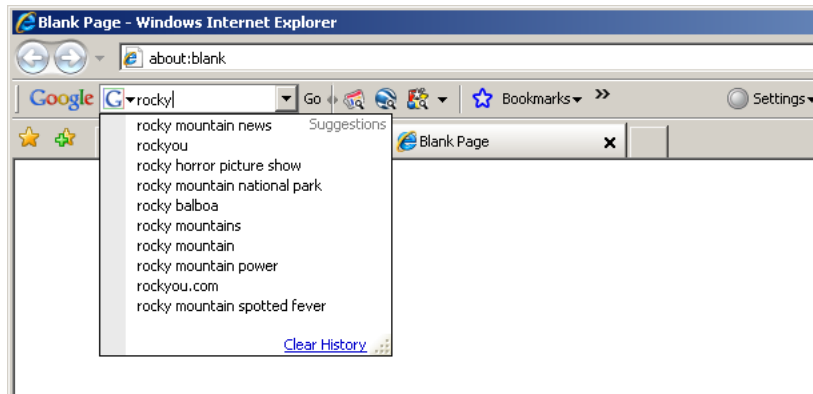
Ajax Concepts

- "Ajax" is an acronym for ***Asynchronous JavaScript And XML***.
- Not a new technology (neither new nor a stand-alone technology)
- Collection of techniques
 - Allows a web developer to dynamically update the page that is currently being displayed by the browser.
 - Uses data from the server.
 - Does not require the browser to display a fresh page.
- Surprisingly:
 - The request need not be asynchronous.
 - Most uses of Ajax do not involve XML.
 - You don't need to know JavaScript.



Copyright 2010. Manta Technologies Inc.

2



Copyright 2010. Manta Technologies Inc.

3

Auto Completion

- Made popular by the Google toolbar
- Produces the pop-up list of potential choices that appears when you type a search request
- Not a list of your previous requests, but a list of similar requests that have been made frequently by other Google users
- Complete your request by finishing your typing or by using your mouse to select one of the offered choices



Copyright 2010. Manta Technologies Inc.

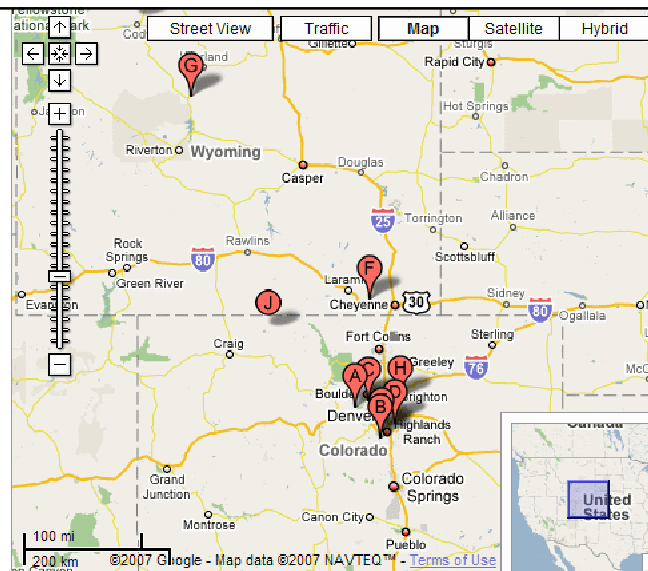
4

- The Google auto completion facility illustrates three features common to most Ajax applications:
 - Data is retrieved from the server without the user having to explicitly request a new page.
 - The user can perform other tasks (such as continuing to type) while the data is being retrieved.
 - Once the response from the server has been received, the new data appears on the screen without affecting the rest of the page.



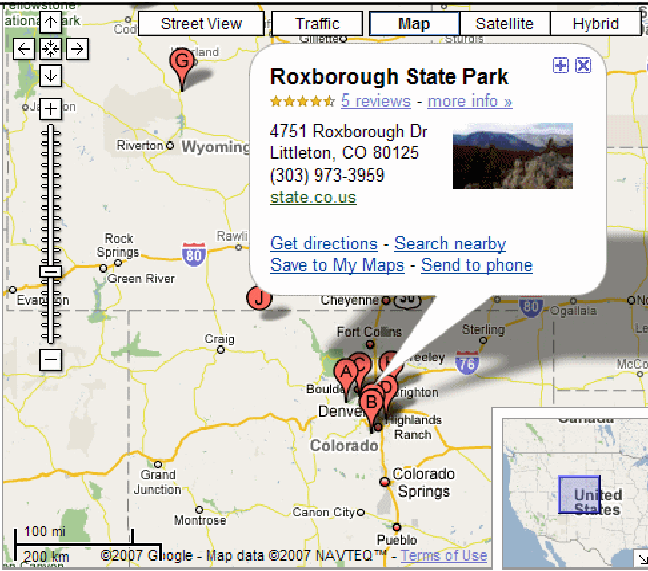
Copyright 2010. Manta Technologies Inc.

5



Copyright 2010. Manta Technologies Inc.

6



Roxborough State Park
 ★★★★★ 5 reviews - [more info »](#)
 4751 Roxborough Dr
 Littleton, CO 80125
 (303) 973-3959
state.co.us
[Get directions](#) - [Search nearby](#)
[Save to My Maps](#) - [Send to phone](#)

100 mi
 200 km
 ©2007 Google - Map data ©2007 NAVTEQ™ - [Terms of Use](#)

Manta Technologies Inc.

Copyright 2010. Manta Technologies Inc. 7

More Info Window

- Used by Google Maps
- When you click (or roll over) a point of interest, Ajax techniques are used to retrieve and display information about this point in a separate area that overlays the existing map.
- Information becomes available when it is needed, without the need to re-display the window.
- Data is retrieved only when it is “requested.”

Manta Technologies Inc.

Copyright 2010. Manta Technologies Inc. 8

MantaNow

- Manta's online delivery system, MantaNow, makes extensive use of Ajax techniques.
- A complete session is retrieved once, when the student enters a new session.
 - The original purpose was to support immediate Forward/Back over a dial-up connection.
- All figures are retrieved "under-the-covers" while the student reads the first couple screens.
- Other hidden browser/server interactions include:
 - Sign-on and authentication
 - Set/Get audio controls
 - Bookmarking
 - Logging and student tracking



Copyright 2010. Manta Technologies Inc.

9

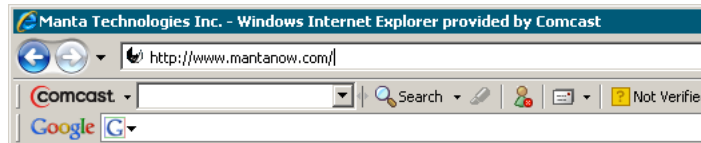
- These techniques have been used for years.
- The current rage started in 2005, when the collection of techniques were named "Ajax" by Jesse James Garrett.
- His essay, "Ajax: A New Approach to Web Applications," described ways to make web applications closer to "thick client" applications that are installed on the desktop.



Copyright 2010. Manta Technologies Inc.

10

Browser/Server Review



- One way to request a new page is to enter a URL in the browser's address line.



Copyright 2010. Manta Technologies Inc.

11

- The HTML for an existing page can include a link that, when clicked, replaces the current document with a new one.
Click `<a href="ebus0252.htm">here</a>` to go to the next page.

- The HTML can include a **<form>** element that causes a server program to be executed when the form's submit button is clicked:

```
<form name="customerData" method="post"
      action="cgi-bin/mycgipgm.pgm">
```

- Typically, the server program responds by creating and returning a new HTML document.



Copyright 2010. Manta Technologies Inc.

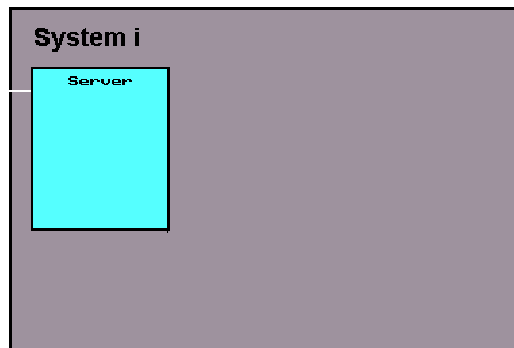
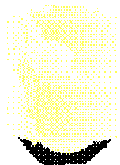
12

- JavaScript also provides several ways to accomplish this task:
 - You can dynamically change the source property of a window, which causes a new document to be loaded.
 - You can use the **window.open()** method to display a document in a new window.
 - You can use the **window.write()** method to replace the HTML document that is displayed in a named window.
- The browser uses the same process to satisfy the request.



Copyright 2010. Manta Technologies Inc.

13

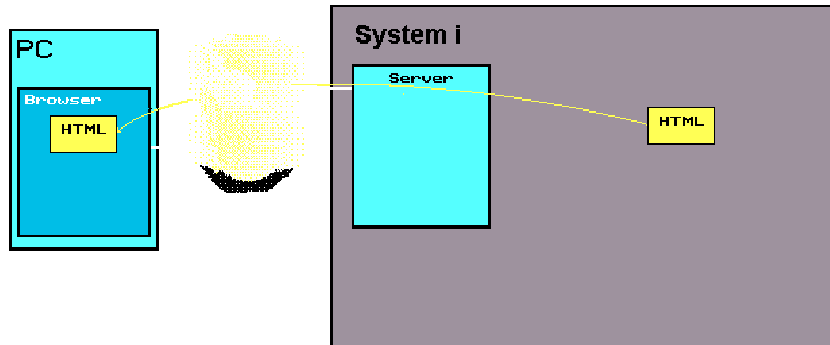


- The browser resolves the domain name to an IP address and connects to the HTTP server.



Copyright 2010. Manta Technologies Inc.

14

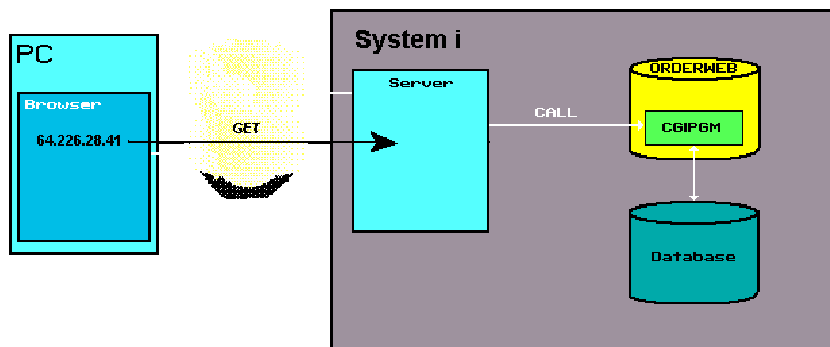


- An HTTP **GET** or **POST** method is used to identify the HTML document to be retrieved or CGI program to be invoked.



Copyright 2010. Manta Technologies Inc.

15

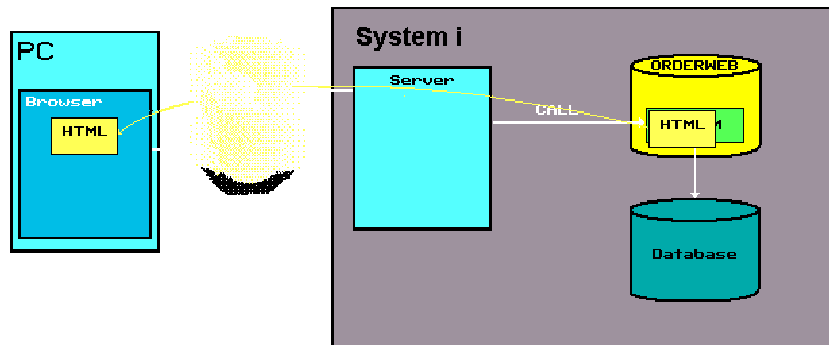


- In the case of a CGI program or servlet request, the appropriate program is invoked.
- This program might update a database or retrieve data for display.



Copyright 2010. Manta Technologies Inc.

16



- It is the job of the server program to create output, in the form of a complete HTML document, that is returned to the browser for display.



Copyright 2010. Manta Technologies Inc.

17

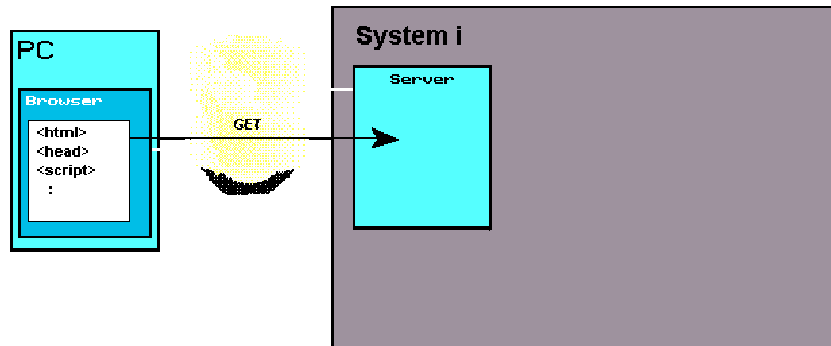
What's different about Ajax?

- Very little
- Modern browsers expose their internal methods used for communication with the server.
- Ajax involves using these methods to do the down and dirty work yourself.



Copyright 2010. Manta Technologies Inc.

18



- You make an HTTP request directly from a JavaScript program.



Copyright 2010. Manta Technologies Inc.

19

- Implemented via a browser-defined application programming interface (API) that lets you control the details of the resulting HTTP request.
- These details include:
 - The URL to be used
 - Whether the **GET** or **POST** method should be used for passing variables
 - The JavaScript function to get control when the request has been processed
 - How errors (such as "page not found") should be handled
 - Whether the HTTP request should be synchronous or asynchronous



Copyright 2010. Manta Technologies Inc.

20

- With a *synchronous* HTTP request, the browser makes the request to the server and then waits for the response.
 - During this time, the user **cannot** enter data, click buttons, or otherwise communicate with the browser.
- With an *asynchronous* HTTP request, the browser makes the request to the server and then continues, without waiting for a response.
 - During this time, the user has access to whatever other functionality that is supported by the application.
- When the HTTP request is eventually satisfied, a specified function within the JavaScript program gets control to perform the next task.
 - This typically involves using the returned data to modify the current document.



Copyright 2010. Manta Technologies Inc.

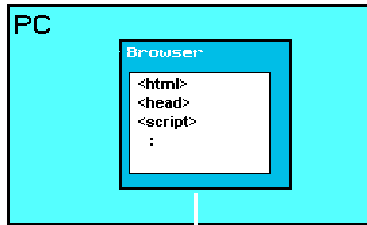
21

- It is the ability to perform asynchronous HTTP requests that make Ajax programming a valuable tool for improving the user's experience.
- In effect, you can hide the delay needed to retrieve new data while letting the user continue to perform useful work.
- Another difference with Ajax is the form of the data that is returned.



Copyright 2010. Manta Technologies Inc.

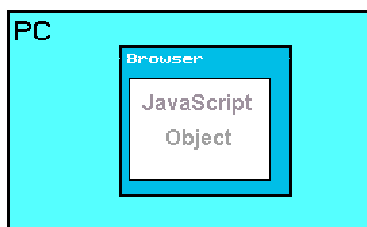
22



- With a traditional request, the browser receives the data that is returned by the server and displays it in the window specified.
- As a result, the data must be in the form of a complete HTML page.

Copyright 2010. Manta Technologies Inc.

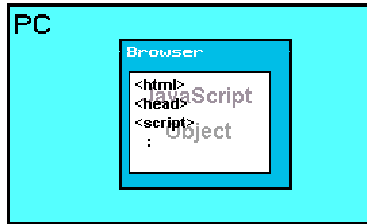
23



- With an Ajax request, the data returned by the server is stored in a browser object that is available to the JavaScript program.

Copyright 2010. Manta Technologies Inc.

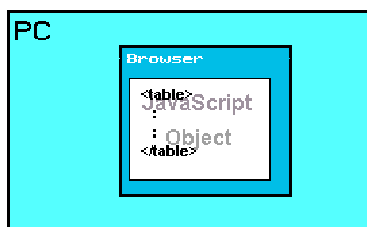
24



- As in the traditional, nonAjax situation, the server could return a complete HTML document that the JavaScript program uses to replace the complete contents of a frame or another window.
- Replacing the current document is problematic, since it overwrites the JavaScript program that is creating the new document.

Copyright 2010. Manta Technologies Inc.

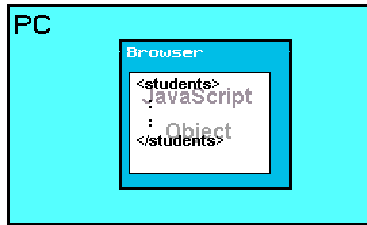
25



- Typically, an Ajax application is designed so that the server program returns only a single HTML object.
 - Pull-down list
 - Table
- This is called an HTML *fragment*.
- Advantages:
 - Performance is improved, because less data needs to be downloaded from the server for each request.
 - By not replacing the complete page when data is retrieved, the user interaction is smoother and less jarring visually.

Copyright 2010. Manta Technologies Inc.

26



- The object that is downloaded need not be an HTML document.
- It can also be plain text or an XML document.
- Returning XML data is beneficial when multiple JavaScript programs need the same data.
- Using XML removes the need for the server program to produce output in the exact form needed by a particular JavaScript application.
- Instead, the server program can produce output in a generic format that can be used by multiple applications that need the same data.
- JavaScript methods are available to easily transform an XML object into the HTML needed to display the data for the user.

Copyright 2010. Manta Technologies Inc.

27

- The advantages of Ajax are summarized by its name, *Asynchronous JavaScript And XML*.
- Using Ajax techniques, your JavaScript program can perform asynchronous HTTP requests while the user is free to perform other browser tasks.
- The data retrieved from the server can be in the form of plain text, HTML, or XML.

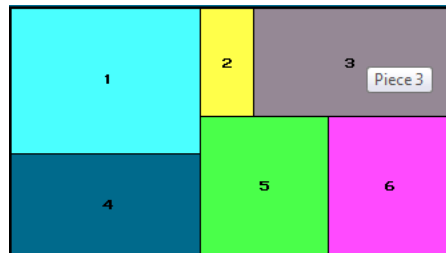


Copyright 2010. Manta Technologies Inc.

28

Making Server Requests

- Sample mapping application



Copyright 2010. Manta Technologies Inc.

29

Map Definition

```

<map name="mymap">
  <area shape="rect" coords="138,41,281,150" title="Piece 1" />
  <area shape="rect" coords="280,42,321,122" title="Piece 2" />
  <area shape="rect" coords="322,42,470,123" title="Piece 3" />
  <area shape="rect" coords="139,151,281,225" title="Piece 4" />
  <area shape="rect" coords="281,124,376,227" title="Piece 5" />
  <area shape="rect" coords="377,125,469,226" title="Piece 6" />
</map>
```



Copyright 2010. Manta Technologies Inc.

30

Triggering Events

```

<map name="mymap">
<area shape="rect" onMouseOver="displayText(0)" onMouseOut="hideBox()"
  coords="138,41,281,150" />
<area shape="rect" onMouseOver="displayText(1)" onMouseOut="hideBox()"
  coords="280,42,321,122" />
<area shape="rect" onMouseOver="displayText(2)" onMouseOut="hideBox()"
  coords="322,42,470,123" />
<area shape="rect" onMouseOver="displayText(3)" onMouseOut="hideBox()"
  coords="139,151,281,225" />
<area shape="rect" onMouseOver="displayText(4)" onMouseOut="hideBox()"
  coords="281,124,376,227" />
<area shape="rect" onMouseOver="displayText(5)" onMouseOut="hideBox()"
  coords="377,125,469,226" />
</map>
```



Copyright 2010. Manta Technologies Inc.

31

Overlay Definition

```
<div id='mybox' class='mybox'></div>
```

Used a style of:

```
div.mybox {
  position:absolute;
  display:none;
  background-color:yellow;
  padding:2;
  border-style:solid;
  border-width:2;
  border-color:black;
}
```



Copyright 2010. Manta Technologies Inc.

32

Quick Test

```
function displayText(n) {  
    var areaDescriptions = new Array();  
    areaDescriptions[0] = "Piece 1";  
    areaDescriptions[1] = "Piece 2";  
    areaDescriptions[2] = "Piece 3";  
    areaDescriptions[3] = "Piece 4";  
    areaDescriptions[4] = "Piece 5";  
    areaDescriptions[5] = "Piece 6";  
  
    displayBox(areaDescriptions[n]);  
}
```

- Our goal is to replace hardcoded values with data retrieved from the server.



Copyright 2010. Manta Technologies Inc.

33

```
function displayBox(x){  
    var topbox = document.getElementsByTagName('mybox');  
    topbox.style.top = window.event.offsetY;  
    topbox.style.left = window.event.offsetX;  
    topbox.style.display = "inline";  
    topbox.innerHTML = x;  
}  
  
function hideBox() {  
    var topbox = document.getElementsByTagName('mybox');  
    topbox.style.display = "none";  
}
```



Copyright 2010. Manta Technologies Inc.

34

The XMLHttpRequest Object

- A JavaScript program can make direct requests to a server using the **XMLHttpRequest** object.
 - Originally introduced in Internet Explorer 5
 - Now supported by all popular browsers
- Like an array or other JavaScript object, you must define the **XMLHttpRequest** object before you can use it.
- The steps to do so differ somewhat based on the browser being used.
 - Prior to version 7, Internet Explorer implemented **XMLHttpRequest** as an ActiveX object.
 - The other popular browsers (and Internet Explorer 7) implement **XMLHttpRequest** as a native JavaScript object.



Copyright 2010. Manta Technologies Inc.

35

- Rather than testing for the different browsers, I recommend the following steps:
 1. Test whether **XMLHttpRequest** is a valid object type in this browser. If so, use it.
 2. Otherwise, try using the ActiveX version.
 3. If this also fails, the user has an old or damaged browser.



Copyright 2010. Manta Technologies Inc.

36

- This logic is implemented by the following code:

```
if (typeof XMLHttpRequest != "undefined") {
    var xmlHttp = new XMLHttpRequest();
} else {
    if (window.ActiveXObject) {
        var xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
    } else {
        alert('XMLHTTP is not available. ' +
            'You have an obsolete or damaged browser.');
```



Copyright 2010. Manta Technologies Inc.

37

- To be extra safe, you can also check for various Microsoft implementations that were present in some of their older versions.
- For better readability, place the code into a separate function.

```
function createXMLHttp() {
    // This function instantiates a new HTTP Request object.
    if (typeof XMLHttpRequest != "undefined") {
        return new XMLHttpRequest();
    } else {
        if (window.ActiveXObject) {
            var aVersions = ["Msxml2.XMLHTTP.5.0", "Msxml2.XMLHTTP.4.0",
                "Msxml2.XMLHTTP.3.0", "Msxml2.XMLHTTP",
                "Microsoft.XMLHTTP"];
            for (var i = 0; i < aVersions.length; i++) {
                try {
                    var oXmlHttp = new ActiveXObject(aVersions[i]);
                    return oXmlHttp;
                } catch (e) {
                }
            }
        }
    }
    alert('XMLHTTP is not available. ' +
        'You have an obsolete or damaged browser.');
```



Copyright 2010. Manta Technologies Inc.

38

- The **FOR** loop tries each ActiveX object in order until one is successful.
- The **catch** traps and ignores an error for each object type that is not found, thus avoiding a JavaScript error.
- Checking for all of the older versions is a bit of overkill, because few people still use versions of Internet Explorer before Version 6.
- However, this is the type of function that you will write once and then copy into your other programs, so being extra safe does not hurt you in any way.



Copyright 2010. Manta Technologies Inc.

39

- This function can be called as follows:

```
var xmlHttp = createXMLHttp();
```

- This statement creates a new HTTP request object and names it **xmlHttp**.



Copyright 2010. Manta Technologies Inc.

40

Initiating a Request

- Once you have created an **XMLHttpRequest** object, you can use it to communicate with the server.
- In our example, we want to retrieve a simple message, given a message number.
- Here is the modified code for the **displayText** function.

```
displayText(n) {  
    var myUrl = 'http://www.mantanow.com/getMessage.cgi?msg=' + n;  
    xmlhttp.open('GET', myUrl);  
    xmlhttp.onreadystatechange = callback;  
    xmlhttp.send(null);  
}
```



Copyright 2010. Manta Technologies Inc.

41

```
var myUrl = 'http://www.mantanow.com/getMessage.cgi?msg=' + n;
```

- Defines the URL of the server program that will process this request.
- Similar to what you would code for a **<FORM>** element.
- The difference is that you must explicitly code the parameters that are passed to the server program.
- Because the **GET** method is used, we are concatenating a fixed character string with the data that was passed to the function.



Copyright 2010. Manta Technologies Inc.

42

```
xmlHttp.open('GET', myUrl);
```

- Invokes the **open()** method of the previously defined **XMLHttpRequest** object.
- You must open this object before you can use it to send a request.
- The first parameter should be "GET" or "POST," indicating the type of HTTP request to be made.
- The second parameter specifies the URL to which the request is to be sent.
- The third parameter specifies whether this is to be a synchronous or an asynchronous request.
 - The default value (**true**) specifies an asynchronous request.
 - Specify **false** to force a synchronous request.
 - Because one of the advantages of using Ajax is the ability to perform asynchronous requests, the **open** method is typically coded as shown, without the third parameter.
 - A synchronous request is needed only when you are performing a series of requests that must be completed in a particular order.



Copyright 2010. Manta Technologies Inc.

43

```
xmlHttp.onreadystatechange = callback;
```

- Specifies a function (**callback** in this case) that is to be called when there is a change to the *ready state* of this request.
- Possible ready states will be listed shortly.
- For now, just be aware that we need to define a function that will get control when the message is returned from the server.
- It is this function that will extract the data for display.



Copyright 2010. Manta Technologies Inc.

44

```
xmlHttp.send(null);
```

- The final line causes the request to be sent to the server.



Copyright 2010. Manta Technologies Inc.

45

Handling State Changes

- As soon as the server program has processed our request, it will return the desired message to the browser, changing the state of our HTTP request object.
- This state change causes the browser to call the function that was specified as the **onreadystatechange** property.
- This function is often called the *callback* function, which is why we decided to name it **callback**.



Copyright 2010. Manta Technologies Inc.

46

- You might think that our callback function could be as simple as the following:

```
function callback() {  
    (extract data into the variable named msg)  
    displayBox(msg);  
}
```

- Extracting the returned data is easy, at least in this simple example.
- The returned data is stored within the HTTP request object.
- You can extract it using the **responseText** property:

```
var msg = xmlhttp.responseText;
```



Copyright 2010. Manta Technologies Inc.

47

- In this example, the HTTP request object, **xmlHttp**, was defined as a global variable so that it could be referenced by multiple functions, including **callback()**.
- The server program returns a simple text string, which is extracted from **xmlHttp** using the **responseText** property.
- The server can also return data in the form of an XML object.
- In the latter case, you would use the **responseXML** method instead of **responseText**.



Copyright 2010. Manta Technologies Inc.

48

- A complication arises from the fact that the state of an HTTP request object changes not only when the final data is returned, but also at several earlier steps in the browser-server communication process.
- Moreover, the final result may not be the return of data, but a communication failure.
- Our function must handle all possibilities.



Copyright 2010. Manta Technologies Inc.

49

HTTP Request Objects States

State	Meaning
0	Uninitialized
1	Loading
2	Loaded
3	Interactive
4	Complete



Copyright 2010. Manta Technologies Inc.

50

- You can view these states using the **readyState** property of the object.
- In general, you are not concerned about the intermediate states.
- All you want to know is when the request is complete.

```
function callback() {
    if (xmlHttp.readyState == 4) {
        var msg = xmlHttp.responseText;
        displayBox(msg);
    }
}
```

- The **IF** statement causes the main block of code to be processed when the HTTP request is complete.
- The intermediate changes of state (which also call **callback()**) are ignored.



Copyright 2010. Manta Technologies Inc.

51

You can't always get what you want

- There may be a problem with your connection to the Internet.
- The URL you specified may be incorrect.
- The server may be down.
- The server may be busy.
- The server may be available, but the program you are calling has an error that prevents it from sending a valid response.



Copyright 2010. Manta Technologies Inc.

52

- For some errors, the browser will automatically retry the request.
- Eventually, after a preset timeout interval, the browser will decide that the problem is permanent and stop.
- At this point, the **readyState** of the HTTP request object changes to **4**, indicating that the request is complete.
- To determine whether the request was successful, you need to check the **status** of the HTTP request object.
 - Another property of the object
 - Can be thought of as a return code that tells you whether the request was successful or, if not, gives you a reason for the failure
- The status value for a successful request is **200**.
- You can also test for specific failure codes, such as a **404** ("not found") error.



Copyright 2010. Manta Technologies Inc.

53

```
function callback() {
    if (xmlHttp.readyState == 4) {
        if (xmlHttp.status == 200) {
            //Request was successful
            var msg = xmlHttp.responseText;
            displayBox(msg);
        } else {
            //Request was unsuccessful
            alert("The browser has experienced an error. " +
                "The error code is " + xmlHttp.status +
                " , which means " + xmlHttp.statusText + ".");
        }
    }
}
```

- The **statusText** property, which gives a short description of the status code.
- For example, status **404** translates to **Not Found**.



Copyright 2010. Manta Technologies Inc.

54

Summary

- You have seen the complete program.
- The HTML
 - Displays the graphic and defines the map
 - Invokes JavaScript functions for **onMouseOver** and **onMouseOut**
 - Includes a dummy **<div>** to be filled and positioned
- The JavaScript has code to
 - Create an HTTP request object
 - Use the object to get data from the server
 - Receive control at callback to format the returned data
 - Create the box overlaying the map
 - Remove the box



Copyright 2010. Manta Technologies Inc.

55

Creating an Ajax Toolkit

- The Ajax code in the example did the job, but was not reusable.
- Your goal should be a collection of functions that you can reuse in similar circumstances.
- This is the idea behind many Ajax/JavaScript libraries that are available online.
- There are other toolboxes (e.g., JSON), that let you avoid most JavaScript coding.



Copyright 2010. Manta Technologies Inc.

56

Creating the HTTP Request Object

```
function createXMLHttpRequest() {  
    // This function instantiates a new HTTP Request object.  
    if (typeof XMLHttpRequest != "undefined") {  
        return new XMLHttpRequest();  
    } else {  
        if (window.ActiveXObject) {  
            var aVersions = ["Msxml2.XMLHTTP.5.0", "Msxml2.XMLHTTP.4.0",  
                             "Msxml2.XMLHTTP.3.0", "Msxml2.XMLHTTP",  
                             "Microsoft.XMLHTTP"];  
            for (var i = 0; i < aVersions.length; i++) {  
                try {  
                    var oXmlHttp = new ActiveXObject(aVersions[i]);  
                    return oXmlHttp;  
                } catch (e) {  
                }  
            }  
        }  
    }  
    alert('XMLHTTP is not available. ' +  
          'You have an obsolete or damaged browser.');
```



Copyright 2010. Manta Technologies Inc.

57

Getting Data

```
function callAjax(url, objCallBack) {  
    var xmlHttp = createXMLHttpRequest();  
    xmlHttp.open('GET', url);  
    xmlHttp.onreadystatechange = function() {  
        if (xmlHttp.readyState == 4) {  
            if (xmlHttp.status == 200) {  
                objCallBack(xmlHttp);  
            } else {  
                alert("The browser has experienced an error. " +  
                    "The error code is " + xmlHttp.status +  
                    " , which means " + xmlHttp.statusText + ".");  
            }  
        }  
    }  
    xmlHttp.send(null);  
}
```



Copyright 2010. Manta Technologies Inc.

58

- Changes from the earlier example:
 - The interface code is in a function by itself
 - Input parameters are the URL and name of the callback function
 - Could modify this to use **GET** or **POST** and an async/nonasync flag
 - HTTP request object is created here
 - Generic callback code is included here, with callback function only called after a successful retrieval
 - HTTP request object is passed to callback program



Copyright 2010. Manta Technologies Inc.

59

- Recall that **displayText(n)** is the **onMouseOver** event handler that triggers the process.
- With the above changes, the JavaScript program becomes the following.

```
function displayText(n) {
    var myUrl = 'http://www.mantanow.com/getMessage.cgi?msg=' + n;
    callAjax(url, myCallback);
}

function myCallback(xmlHttp) {
    var msg = xmlHttp.responseText;
    displayBox(msg);
}
```



Copyright 2010. Manta Technologies Inc.

60

Other Methods

- In addition to **open** and **send**, there are a few other methods supported by HTTP request objects.
- Suppose we have created an HTTP request object named **xmlHttp**.
 - xmlHttp.abort()** — Terminates the current request
 - xmlHttp.getAllResponseHeaders()** — Returns the HTTP response header that precedes the message received from the server
 - xmlHttp.getResponseHeader("header")** — Retrieves only the specified component, such as "Context-Type".
 - xmlHttp.setRequestHeader("header", "value")** — Sets the HTTP request header. Example:

```
xmlHttp.setRequestHeader("Cache-Control", "nocache");
```



Copyright 2010. Manta Technologies Inc.

61

Using Post

- With the HTTP **GET** method,
 - Input data is appended to the URL
 - 4096-character URL limit
 - Query string is passed to the CGI program as an environment variable
- With the HTTP **POST** method,
 - Input data is sent inside the **POST** request
 - No size limit
 - Query string is passed to the CGI program via **STDIN**



Copyright 2010. Manta Technologies Inc.

62

GET/POST Examples

GET

```
var myUrl = 'http://www.mantanow.com/getMessage.cgi?msg=' + n;  
xmlHttp.open('GET', myUrl);  
xmlHttp.onreadystatechange = callback;  
xmlHttp.send(null);
```

POST

```
var myUrl = 'http://www.mantanow.com/getMessage.cgi';  
xmlHttp.open('POST', myUrl);  
xmlHttp.setRequestHeader('Content-Type',  
    "application/x-www-form-urlencoded");  
xmlHttp.onreadystatechange = callback;  
xmlHttp.send("msg=" + n);
```



Copyright 2010. Manta Technologies Inc.

63

- Four changes:
 - Change **GET** to **POST** in the **open** method.
 - Delete the query string from the URL.
 - Pass the query string as the only parameter of the **send** method.
 - Set the Context-Type to "application/x-www-form-urlencoded"



Copyright 2010. Manta Technologies Inc.

64

Encoding Parameters

- Problems can result if the parameters contain special characters (`/ \ . , & ?`) or a blank.
- Such data must be *encoded* to avoid ambiguity.
- Each space in the input string must be replaced with a plus sign (+).
- Other special characters must be converted to hex values preceded by a percent sign (%).
 - comma → **%2C**
 - single quote → **%27**
- For a traditional HTML form, such data is converted by the browser.
- In an Ajax application, it becomes your responsibility to encode the data.



Copyright 2010. Manta Technologies Inc.

65

- This job becomes trivial if you include the following function in your JavaScript program:

```
function encodeMe(value) {  
    //Encode parameters for placement on a URL.  
    if (encodeURIComponent) return encodeURIComponent(value);  
    if (escape) return escape(value);  
    return value;  
}
```

- This code tests for the presence of the **encodeURIComponent** and **escape** methods and uses whichever one it finds first.
- **encodeURIComponent** is preferred, if it is supported by the user's browser.
 - The older **escape** method does not encode the plus sign (+), which is interpreted as a space on the server.
 - **escape** does not correctly handle non-ASCII characters.
- In either case, the method accepts a non-encoded string and returns the corresponding encoded string.



Copyright 2010. Manta Technologies Inc.

66

Use of Encoding

```
var myUrl = "login.ezt";
myUrl += "?company="      + encodeMe(p1);
myUrl += "&student_ID="   + encodeMe(p2);
myUrl += "&student_PW="   + encodeMe(p3);
myUrl += "&new_PW="       + encodeMe(p4);
```

- A question mark (?) must separate the program name from the first parameter.
- An ampersand (&) is required before each subsequent parameter.



Copyright 2010. Manta Technologies Inc.

67

- I like putting such code into a separate function:

```
function makeURL(p1, p2, p3, p4) {
    var myUrl = "login.ezt";
    myUrl += "?company="      + encodeMe(p1);
    myUrl += "&student_ID="   + encodeMe(p2);
    myUrl += "&student_PW="   + encodeMe(p3);
    myUrl += "&new_PW="       + encodeMe(p4);
    return myUrl;
}
```

- This function can be invoked as follows:

```
xmlHttp.open('GET', makeURL(company, sID, sPW, newPW));
xmlHttp.onreadystatechange = callback;
xmlHttp.send(null);
```



Copyright 2010. Manta Technologies Inc.

68

- If you will be using **POST**, you could define the URL separately and use a function to build the query string.

```
function makeQuery(p1, p2, p3, p4) {
    var MyParms = "company=" + encodeMe(p1);
    MyParms += "&student_ID=" + encodeMe(p2);
    MyParms += "&student_PW=" + encodeMe(p3);
    MyParms += "&new_PW=" + encodeMe(p4);
    return MyParms;
}
```

- This function can be used in a **POST** request as follows:

```
var myUrl = 'http://www.mantanow.com/login.ezt';
xmlHttp.open('POST', myUrl);
xmlHttp.setRequestHeader('Context-Type',
    "application/x-www-form-urlencoded");
xmlHttp.onreadystatechange = callback;
xmlHttp.send(makeQuery(company, sID, sPW, newPW));
```



Copyright 2010. Manta Technologies Inc.

69

Passing Data as XML

- You have seen how XML data can be received using **responseXML** in place of **responseText**.
- You can send data as XML.
 - Use JavaScript to create the XML object (say, **myXMLObj**).
 - Send as a single **POST** parameter.

```
var myUrl = 'http://www.mantanow.com/deleteStudents.ezt';
xmlHttp.open('POST', myUrl);
xmlHttp.setRequestHeader('Context-Type',
    "application/x-www-form-urlencoded");
xmlHttp.onreadystatechange = deleteCallback;
xmlHttp.send(myXMLObj);
```



Copyright 2010. Manta Technologies Inc.

70

Questions

whansen@mantatech.com



Copyright 2010. Manta Technologies Inc.

71