"You will wander the underworld blind, deaf, and dumb, and all the dead will know: This is Hector, the fool who thought he killed Achilles."

-Achilles,
The Illiad

The Other AJAX

~

R. Amrita
P.M. Krishnan

AJAX

Ajax is a methodology of designing web applications , so that they have the look and feel of desktop applications .

What is Ajax – *Exactly?*

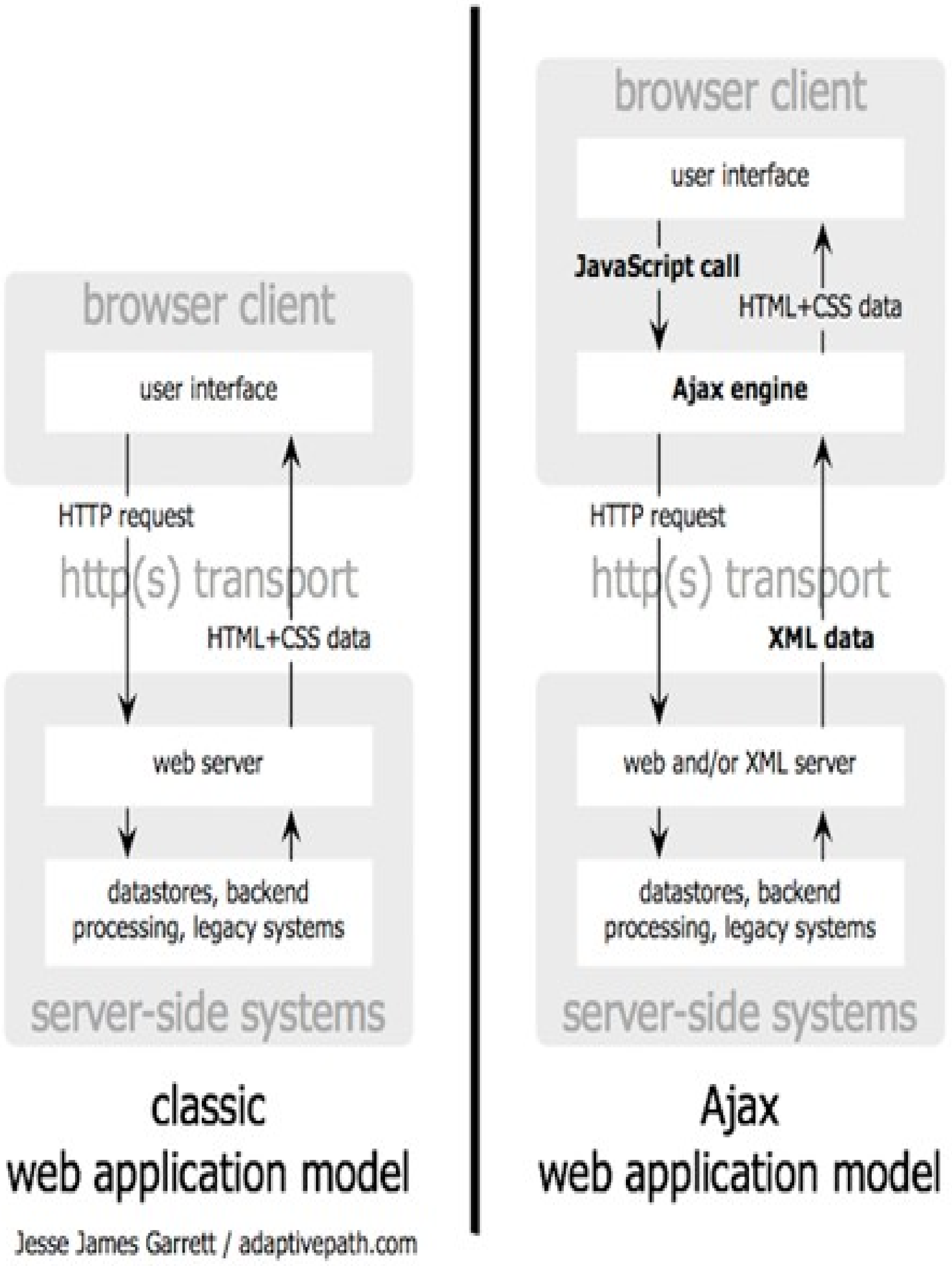
AJAX stands for **A**synchronous **Ja**vaScript and **X**ML.

AJAX is **not** a programming nor a scripting language, nor is it a new technology but is really a collection of technologies.

It is, in essence, the convergence of :

- Standards based presentation using XHTML and CSS
- dynamic display and interaction using DOM
- data interchange and manipulation using XML and XSLT
- asynchronous data retrieval using XMLHttpRequest
- and JavaScript to bind them all

The “asynchronous” part means that the browser isn't going to wait for data to be returned from the server, but can handle that data as it is sent back, **when** it is sent back.



You don't have to put your application on hold until the data arrives. In Architecture terms it similar to saying: *There is no stall in the Pipeline.*

How Ajax is Different

An Ajax application eliminates the start-stop-start-stop nature of interaction on the Web by introducing an intermediary — an Ajax engine — between the user and the server.

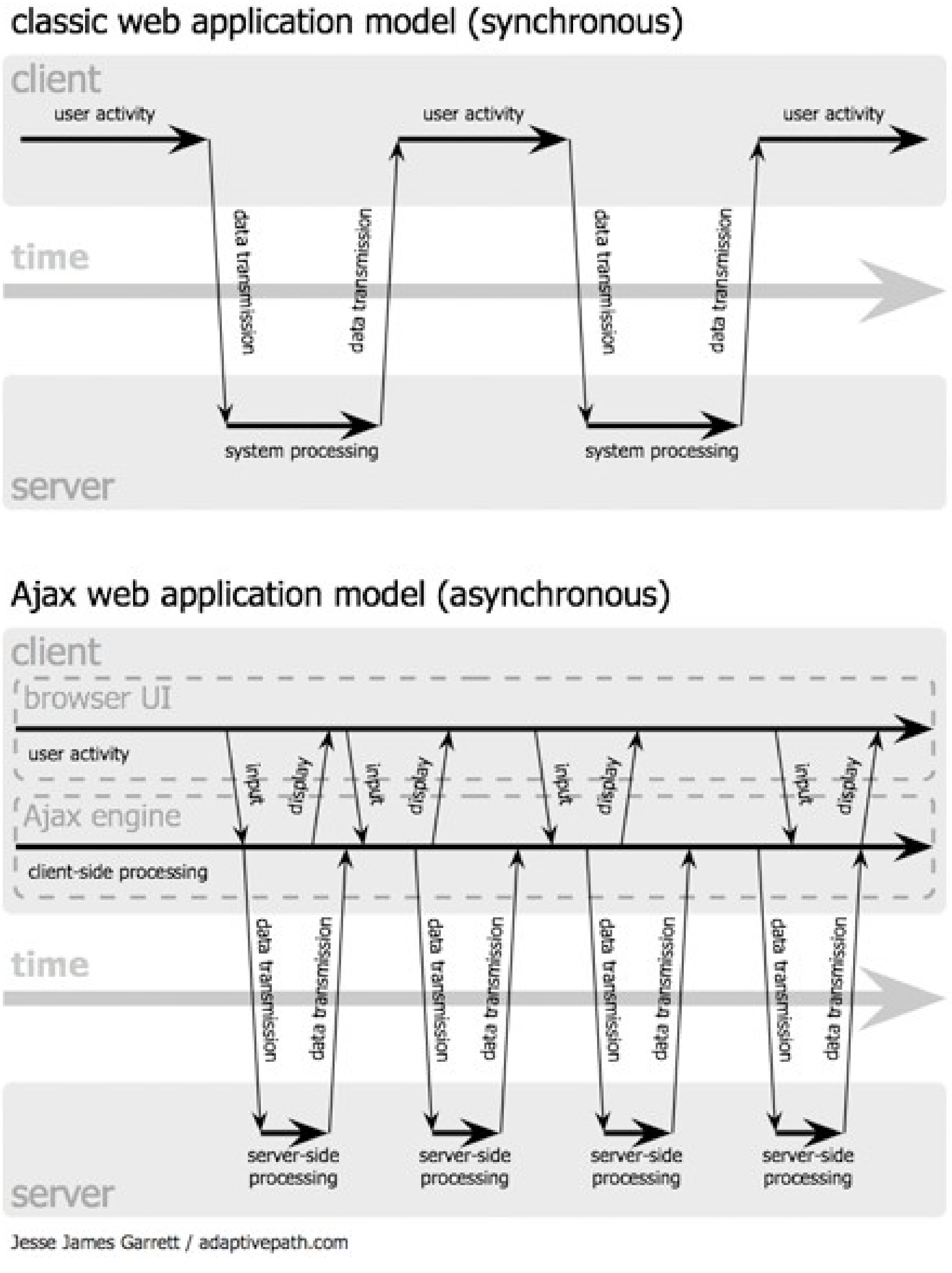
Instead of loading a webpage, at the start of the session, the browser loads an Ajax engine — written in JavaScript and usually tucked away in a hidden frame.

This engine is responsible for both rendering the interface the user sees and communicating with the server on the user's behalf.

The Ajax engine allows the user's interaction with the application to happen asynchronously — independent of communication with the server.

So the user is never staring at a blank browser window and an hourglass icon, waiting around for the server to do something.

The Synchronous and Asynchronous interaction pattern of Web Applications



The JavaScript part of the term Ajax is also very important because that's what makes Ajax happen in the browser.

Ajax relies on JavaScript in the browser to connect to the server and to handle the data that the server sends back.

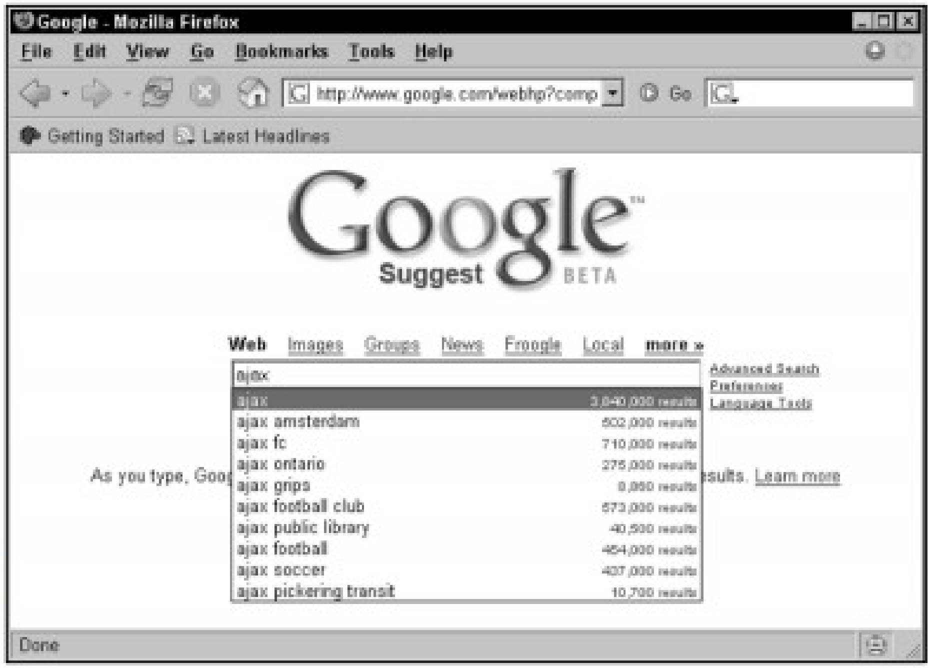
About XML, as our class linguist would say : It is the *lingua franca* of the Web, providing a text-based way to send data back and forth across the Internet.

For that reason, Ajax applications are often written to handle data sent back from the server using XML.

What is the being done with it???

Searching in real time with live searches

One of the signature use is that you can do with Ajax is live searching, where you get search results instantly, as you enter the term you're searching for. For example, http://www.google.com/webhp?complete=1&hl=en, the page which appears as shown. As you enter a term to searched, Ajax contacts Google behind the scenes, and you see a drop-down menu that displays common search terms from Google that might match what you're typing. If you want to select one of those terms, just click it in the menu.


Google - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.google.com/webhp?comp
Go
Getting Started Latest Headlines
Google Suggest BETA
Web Images Groups News Froogle Local more »
Advanced Search
Preferences
Language Tools
As you type, Goo
sults. Learn more
Done

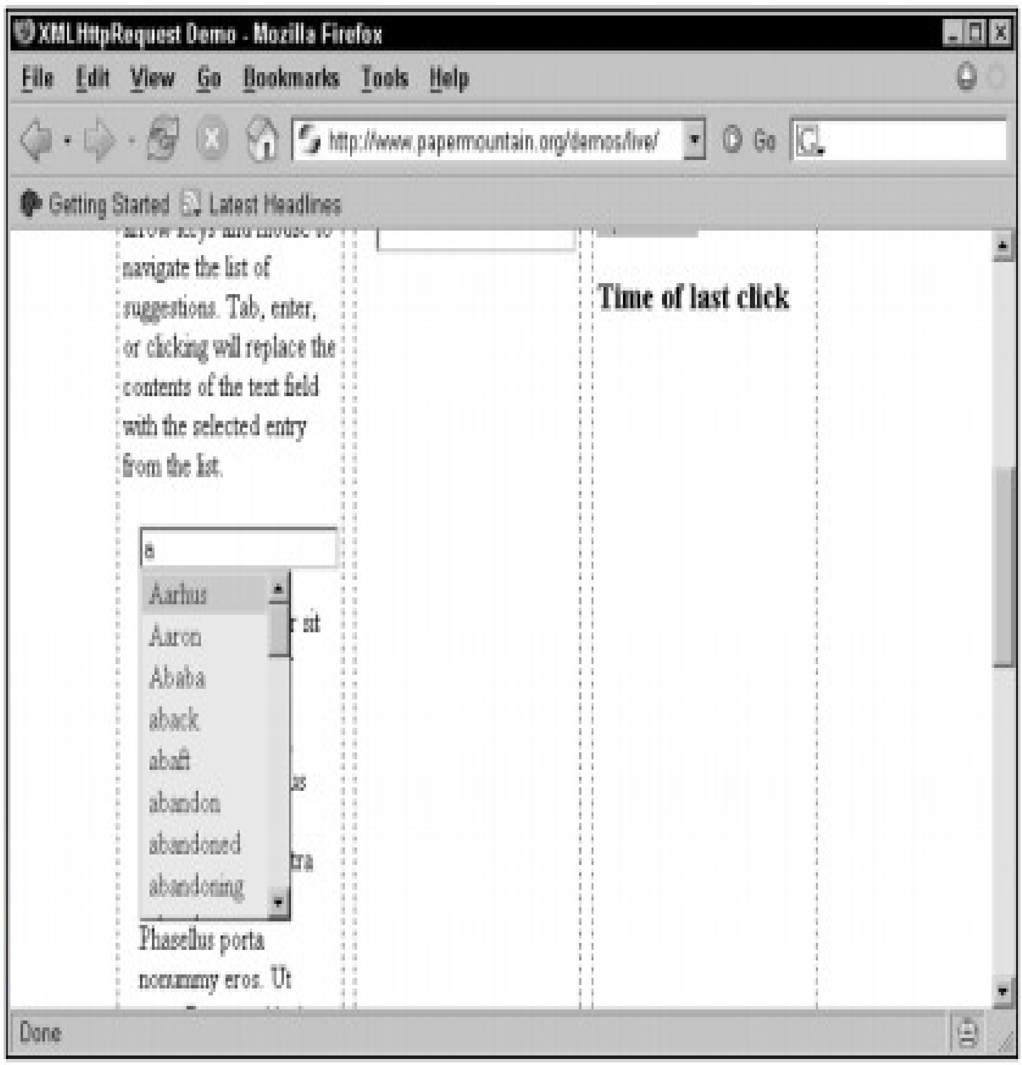
ajax

ajax	3,040,000 results
ajax amsterdam	502,000 results
ajax fc	710,000 results
ajax ontario	275,000 results
ajax grips	9,860 results
ajax football club	573,000 results
ajax public library	40,500 results
ajax football	454,000 results
ajax soccer	407,000 results
ajax pickering transit	10,700 results

Figure 1-2: A Google live search.

Getting the answer with auto complete

Closely allied to live search applications are auto complete applications, which try to guess the word you're entering by getting a list of similar words from the server and displaying them. You can see an example at www.paper mountain.org/demos/live, which appears in the figure. As you enter a word, this example looks up words that might match in a dictionary on the server and displays them, as you see in the figure. If you see the right one, just click it to enter it in the text field, saving you some typing.


XMLHttpRequest Demo - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.papermountain.org/demos/live/
Go
Getting Started
Latest Headlines
Time of last click
a
Aarhus
Aaron
Ababa
aback
abaft
abandon
abandoned
abandoning
Done

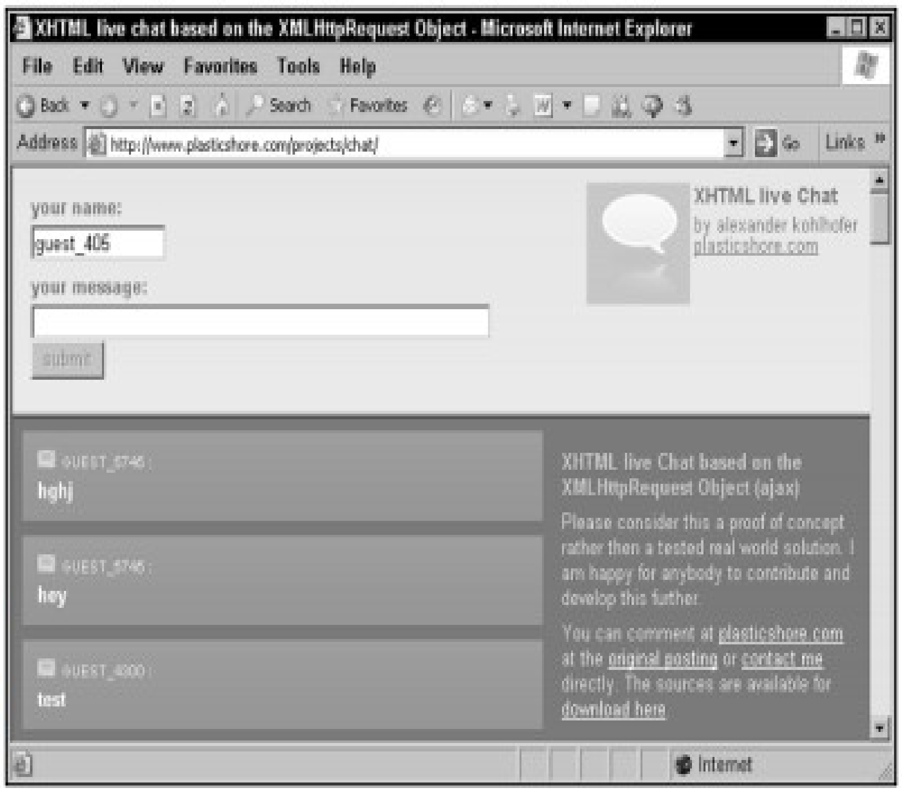
navigate the list of suggestions. Tab, enter, or clicking will replace the contents of the text field with the selected entry from the list.

Phasellus porta nonummy eros. Ut

Figure 1-3: An autocomplete example.

<u>Chattings</u>

Because Ajax excels at updating Web pages without refreshing the displayed page, it's a great choice for Web-based chat programs, where many users can chat together at the same time. Take a look at www.plasticshore.com/ projects/chat, for example, which you can see in the figure. Here, you just enter your text and click the Submit button to send that text to the server. All the while, you can see everyone else currently chatting — no page refresh needed.


XHTML live chat based on the XMLHttpRequest Object - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.plasticshore.com/projects/chat/
Go Links
your name:
guest_406
your message:
submit
XHTML live Chat
by alexander kohlhofer
plasticshore.com
Internet

GUEST_5746 :
hghj

GUEST_5746 :
hey

GUEST_4000 :
test

XHTML live Chat based on the XMLHttpRequest Object (ajax)

Please consider this a proof of concept rather then a tested real world solution. I am happy for anybody to contribute and develop this futher.

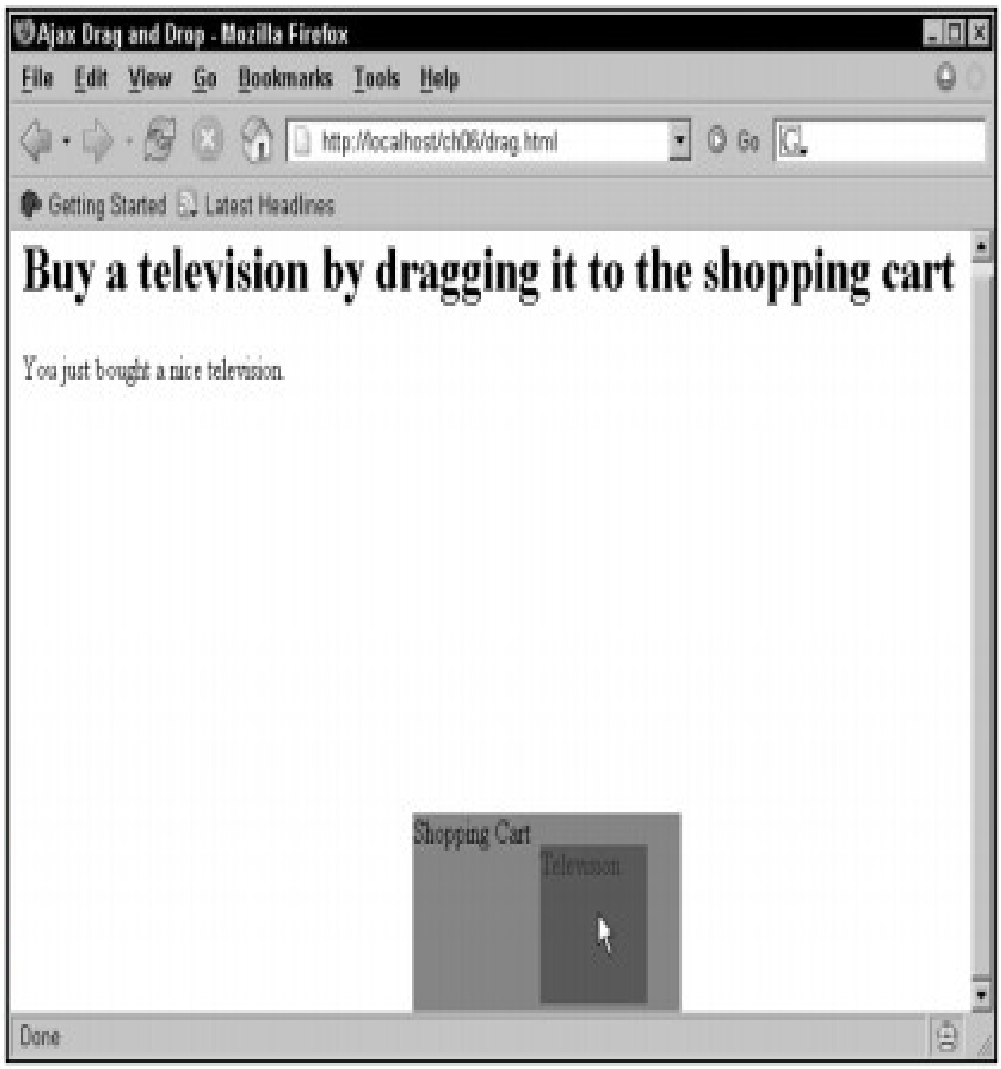
You can comment at plasticshore.com at the original posting or contact me directly. The sources are available for download here

Figure 1-4: An Ajax-based chat application.

There are plenty of Ajax-based chat rooms around. Take a look at `http://treehouse.ofb.net/chat/?lang=en` for another example.

Dragging and dropping with Ajax

At the beginning of this chapter, I mention a drag-and-drop shopping cart example. As shown in the figure, when the user drags the television to the shopping cart in the lower-right, the server is notified that the user bought a television. Then the server sends back the text that appears in the upper left, "You just bought a nice television."


Ajax Drag and Drop - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/ch06/drag.html
Go
Getting Started
Latest Headlines
Shopping Cart
Television
Done

Buy a television by dragging it to the shopping cart

You just bought a nice television.

Figure 1-5: Drag-and-drop shopping.

Gaming with Ajax

Here's a different one — a magic diary that answers you back using Ajax techniques, as shown in the figure. You can find it at http://pandorabots.com/pandora/talk?botid=c96f911b3e35f9e1. When you type something , such as "Hello," the server is notified and sends back an appropriate response that then appears in the diary, such as "Hi there!"

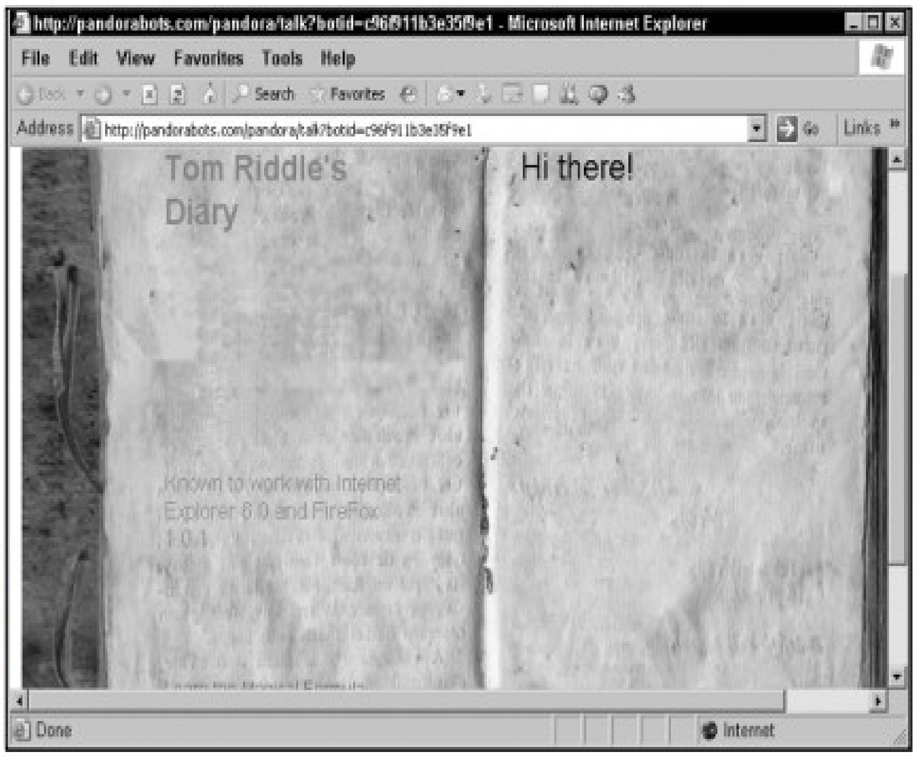
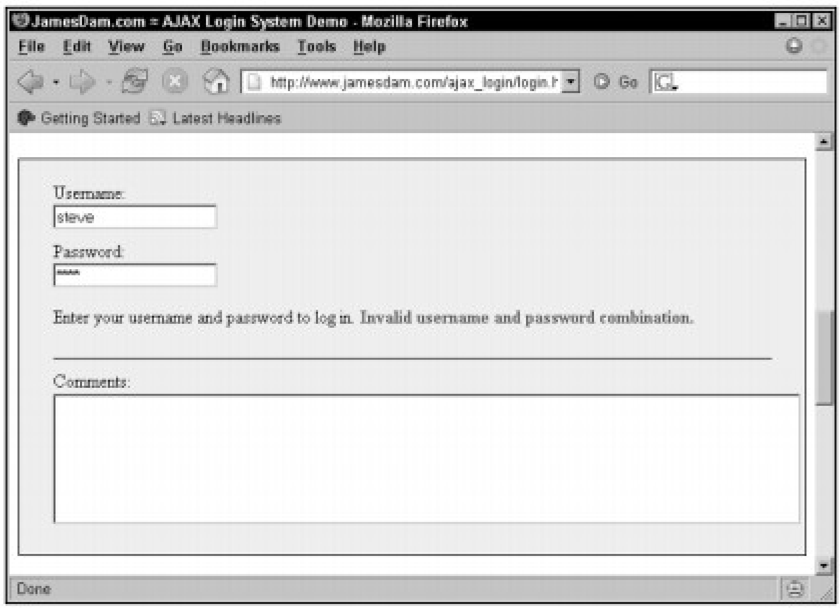

http://pandorabots.com/pandora/talk?botid=c96f911b3e35f9e1 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://pandorabots.com/pandora/talk?botid=c96f911b3e35f9e1
Go Links
Tom Riddle's Diary
Hi there!
Known to work with Internet Explorer 6.0 and FireFox 1.0.1
Done
Internet

Figure 1-6: An interactive Ajax-enabled diary.

Getting instant login feedback

Another Internet task that can involve many annoying page refreshes is logging in to a site. If you type the wrong login name, for example, you get a new page explaining the problem, have to log in on another page, and so on. How about getting instant feedback on your login attempt, courtesy of Ajax? That's possible, too. Take a look at www.jamesdam.com/ajax_login/ login.html, which appears in the figure. I've entered an incorrect username and password, and the application says so immediately


JamesDam.com = AJAX Login System Demo - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.jamesdam.com/ajax_login/login.h
Go
Getting Started
Latest Headlines
Done

Username:
steve

Password:
*****

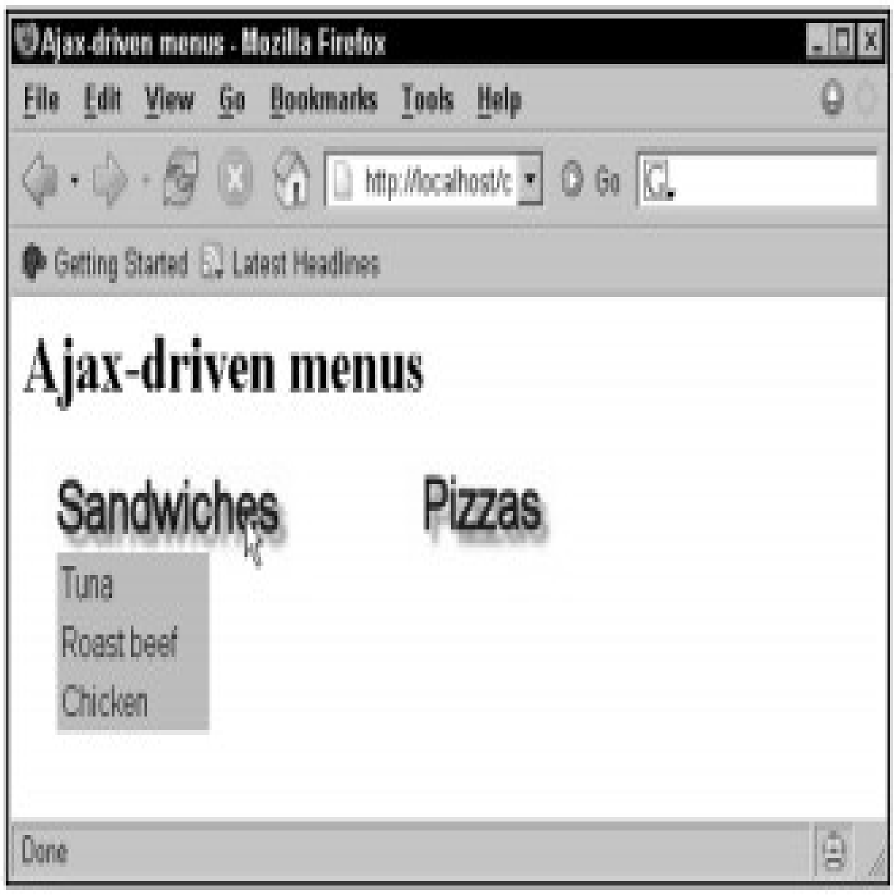
Enter your username and password to log in. Invalid username and password combination.

Comments:

Figure 1-7: Ajax makes correcting login mistakes easier.

Ajax-enabled pop-up menus

You can grab data from the server as soon as the user needs it using Ajax. For example, take a look at the application in Figure 1-8. The pop-up menus appear when you move the mouse and display text retrieved from the server using Ajax techniques. By accessing the server, Ajax allows you to set up an interactive menu system that responds to the menu choices the user has already made.


Ajax-driven menus - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/c
Go
Getting Started
Latest Headlines
Done

Ajax-driven menus

Sandwiches

Pizzas

Tuna

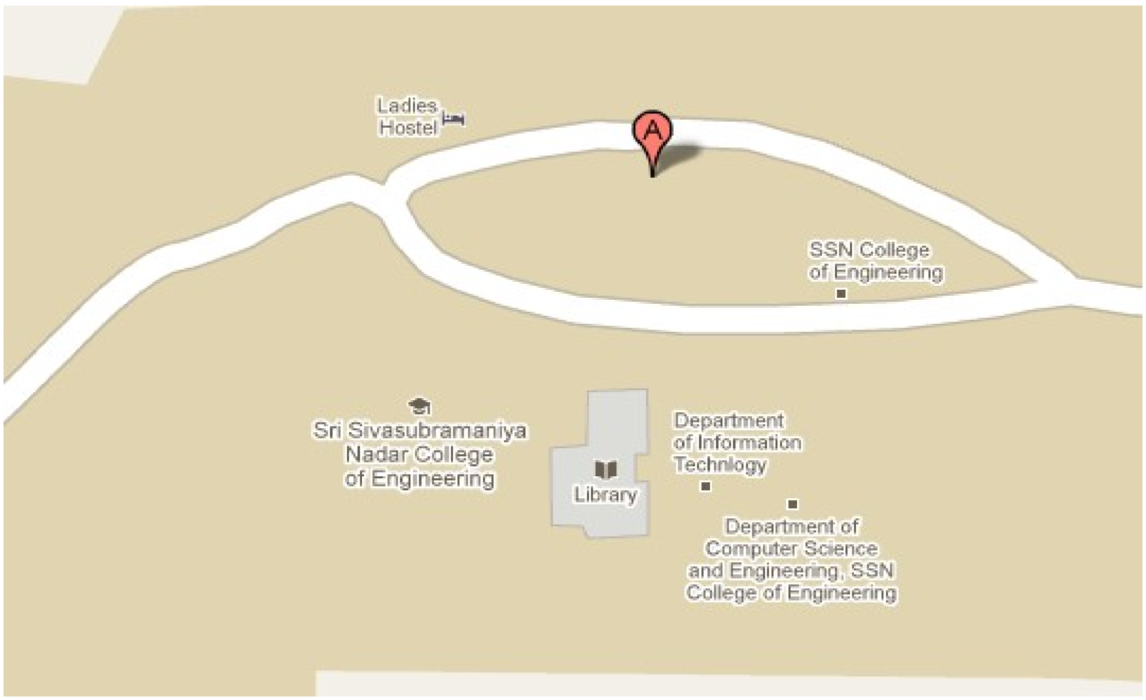
Roast beef

Chicken

Figure 1-8: Ajax-enabled pop-up menus.

Google Maps and Ajax

One of the most famous Ajax application is Google Maps, at http://maps.google.com, which you can see at work in the figure, zooming in on our college. Whenever you modify the view, the whole map does not get refreshed but only the section of interest.


Ladies Hostel
A
SSN College of Engineering
Sri Sivasubramaniya Nadar College of Engineering
Library
Department of Information Technlogy
Department of Computer Science and Engineering, SSN College of Engineering

Step-by-step AJAX Application development

Assumptions

All the following codes for displaying AJAX methodology is based on the assumption that it works on the Apache Server with Tomcat Container, along with the required versions of XHTML, CSS and XML.

It is assumed that Mozilla Firefox is the browser used since the use of ActiveXObject is redundant, along with the use of IE itself.

The Build-up

Here we try to build, step-by-step, a web application that generates a XHTML document within which a visit counter appears to update itself automatically, **without reloading the document.**

The General Atomic steps in building an AJAX powered Web app

- Setting up the XHTML document with relevant CSS and servlet.
- Writing the Javascript
- Creating an XMLHttpRequest object.
- Configuring the XMLHttpRequest object.
- Handling data downloads and downloading data

Step 1 :

Create the XHTML file which uses the AJAX methodology.

In our case, it is the counter.html which displays the number of times it is visited. When it is loaded, it calls the init() function in JavaScript.

```
<html xmlns='http://www.w3.org/1999/xhtml'>
<head>
<title> VisitCountUpdate.java
</title>
<script type='text/javascript' src='vs.js'>
</script>
<meta http-equiv='Content-Script-Type'
content='text/javascript' />
</head>
<body onload='init();'>
<p>
Hello World!
</p>
<p>
 This page has been viewed <span
id='visits'>12</span> times since the most
recent server restart.
</p>
</body></html>
```

The servlet : VisitCountUpdate

The doPost() method of visitcountupdate will be accessed by XMLHttpRequest instances to obtain the current visit count.

It returns the count in XML formatting.

```
   public class  visitcountupdate extends
HttpServlet
     {
      public void doPost (HttpServletRequest
request, HttpServletResponse response ) throws
ServletException, IOException
      {

  response.setContentType(“application/xml”);
   PrintWriter servletOut=response.getWriter();
 servletOut.println (
   “<?xml version='1.0' encoding='UTF-8'?>\n”
+
   “<count>”+ visits +  “</count>”
  );
 servletOut.close();
    }}
```

<u>Step 2:</u>

Create a JavaScript file that contains the init() method, which in turn calls the getVisits() method which retrieves the visits to the website whenever there is an update in the count. It does this by calling the updateVisits() function which uses DOM to dynamically set the value of the visits node.

So initially the Javascript file looks like

```
function init() {
window.setInterval("getVisits()",7000);
}

function getVisits() {
// Some XMLRequestObject code
updateVisits()
}
function updateVisits() {
// Some DOM scripting to dynamically set the
visits count in HTML page.
}
```

Now the javascript is put through the next three steps.

Step 3 :

All the AJAX magic is done using the XMLHttpRequest object.

What is the XMLRequestObject?
This is the object that forms the basis of AJAX. The XMLHttpRequest Object is built into modern browsers and an instance of XMLHttpRequest Object allows a JavaScript program to send an HTTP request to a server and receive back a response containing an XML document.

In this step, a variable named 'connection' to store the XMLHttpRequest Object is created and is initally set to false.

- if (window.XMLHttpRequest) is used to check whether the browser supports XMLRequestObject.
- Next is to create a XMLHttpRequest Object using 'connection'

```
function getVisits() {
var connection=false;
if(window.XMLHttpRequest) {
connection = new XMLHttpRequest();
}
// more code to come
}
```

Methods of the XMLHttpRequest Object

Method	Description
open(*method,url,async*)	Specifies the type of request, the URL, and if the request should be handled asynchronously or not. *method*: the type of request: GET or POST *url*: the location of the file on the server *async*: true (asynchronous) or false (synchronous)
send(*string*)	Sends the request off to the server. *string*: Only used for POST requests
setRequestHeader(*header,value*)	Adds HTTP headers to the request. *header*: specifies the header name *value*: specifies the header value

Three important properties of the XMLHttpRequest object:

Property	Description
onreadystatechange	Stores a function (or the name of a function) to be called automatically each time the readyState property changes
readyState	Holds the status of the XMLHttpRequest. Changes from 0 to 4: 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 404: Page not found

The onreadystatechange event is triggered every time the readyState changes.

The readyState property holds the status of the XMLHttpRequest.

When readyState is 4 and status is 200, the response is ready.

Step 4: Configuring the XMLHttpRequest Object.

The step in configuring the XMLHttpRequest Object is to *open* it.

The open("method","URL",true).

The true flag indicates the asynchronous nature of the call.
There are also optional username and password parameters.

```
function getVisits() {
var connection=false;
if(window.XMLHttpRequest) {
connection = new XMLHttpRequest();
}
connection.open("POST","http://localhost:8081/mya
jax/visitcountupdate",true);
// more code to come
}
```

Step 5: Handling data downloads

A callback function (update()) is to be set up that will be called when data has been downloaded. XMLHttpRequest Object will call that callback function when there is to be a download.

The function is called each time the value of the *readyState* property of the XMLHttpRequest Object instance changes.

The send() method is used to send the acutal Http request to the server.

```
function getVisits() {
var connection=false;
if(window.XMLHttpRequest) {
connection = new XMLHttpRequest();
}
connection.open(“POST”,”http://localhost:8081/myajax/visitcountupdate”,true);
connection.onreadystatechange = function update()
{
updateVisits(connection);
};

connection.setRequestHeader(“Content-Type”,”application/x-www-form-urlencoded”);
connection.send(“”);
}
```

Step 5: Part Two:

The value 4 indicates that the entire response has been received and the status '200' indicates the HttpResponse status of 200.

The updation of the visits is done only when the following condition is satisfied.

```
if(connection.readyState==4 && connection.status == 200)
```

The updateVisits function has two Document nodes: document displayed in the client area of browser and the connection.responseXML , the root of the tree representing the XML response from server.

The DOM processing replaces the text within visits element of the browser's tree with the character data in the response XML.The *responseXML* property of the XMLHttpRequest instance represents a node of type Document.

```
function updateVisits(connection) {
if(connection.readyState==4 && connection.status
== 200) {
var visits=document.getElementById("visits");
var
count=connection.responseXML.documentElement;
visits.childNodes[0].data=count.childNodes[0].data;
}
return;}
```

The completed overall JavaScript File:

```
function init() {
window.setInterval("getVisits()",3000);
}

function getVisits() {
var connection=false;
if(window.XMLHttpRequest) {
connection = new XMLHttpRequest();
}

if (connection)
{
connection.open("POST","http://localhost:8081/myajax/visitcountupdate",true);
connection.onreadystatechange = function update()
{
updateVisits(connection);
};

connection.setRequestHeader("Content-Type","application/x-www-form-urlencoded");
connection.send("");
}
return;
}
```

```
function updateVisits(connection) {
if(connection.readyState==4 && connection.status
== 200) {
var visits=document.getElementById(“visits”);
var
count=connection.responseXML.documentElement;
visits.childNodes[0].data=count.childNodes[0].data;
}
return;
}
```

THANK YOU!!!

“But if doom denies this to me, then I will have naught: neither life diminished, nor *love* halved, nor honour abated.”

~

Lord Denethor,
The Lord of the Rings

THE END