AJAX

- Ajax motivation
- The basic Ajax process
- The need for anonymous functions
- Using dynamic content and JSP
- Using dynamic content and servlets
- Displaying HTML results

What is AJAX?

- What is in the name ?
  - **AJAX** stands for :

    Asynchronous JavaScript and XML

What is Ajax?

- In essence, Ajax is an efficient way for a web application to handle user interactions with a web page - ***a way that reduces the need to do a page refresh or full page reload for every user interaction***.

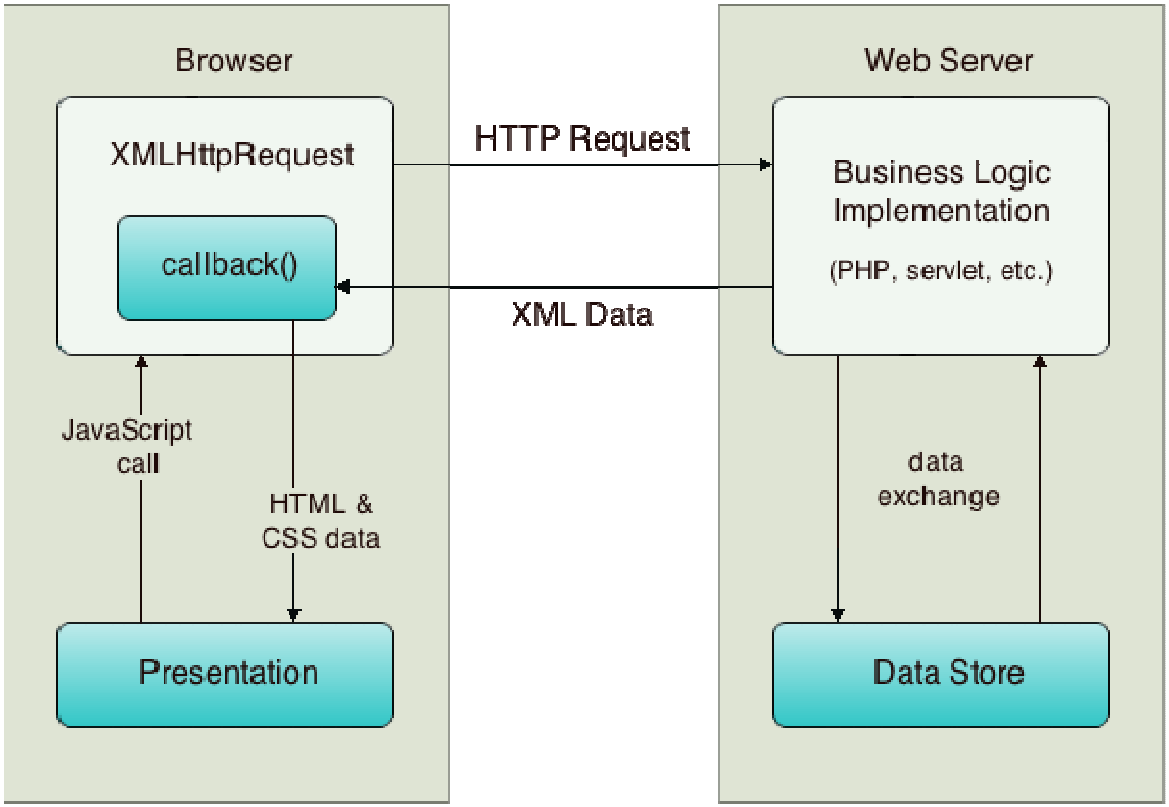
- Ajax interactions are handled asynchronously in the background. As this happens, a user can continue working with the page.

- Ajax interactions are initiated by JavaScript code. When the Ajax interaction is complete, JavaScript updates the HTML source of the page. **The changes are made immediately without requiring a page refresh**.

What is Ajax?

- Ajax interactions can be used to do things such as :
  1. Validate form entries (while the user is entering them) using server-side logic,
  2. Retrieve detailed data from the server,
  3. Dynamically update data on a page, and
  4. Submit partial forms from the page.

What is Ajax?


Browser
XMLHttpRequest
callback()
JavaScript call
HTML & CSS data
Presentation
HTTP Request
XML Data
Web Server
Business Logic Implementation
(PHP, servlet, etc.)
data exchange
Data Store

What is Ajax?

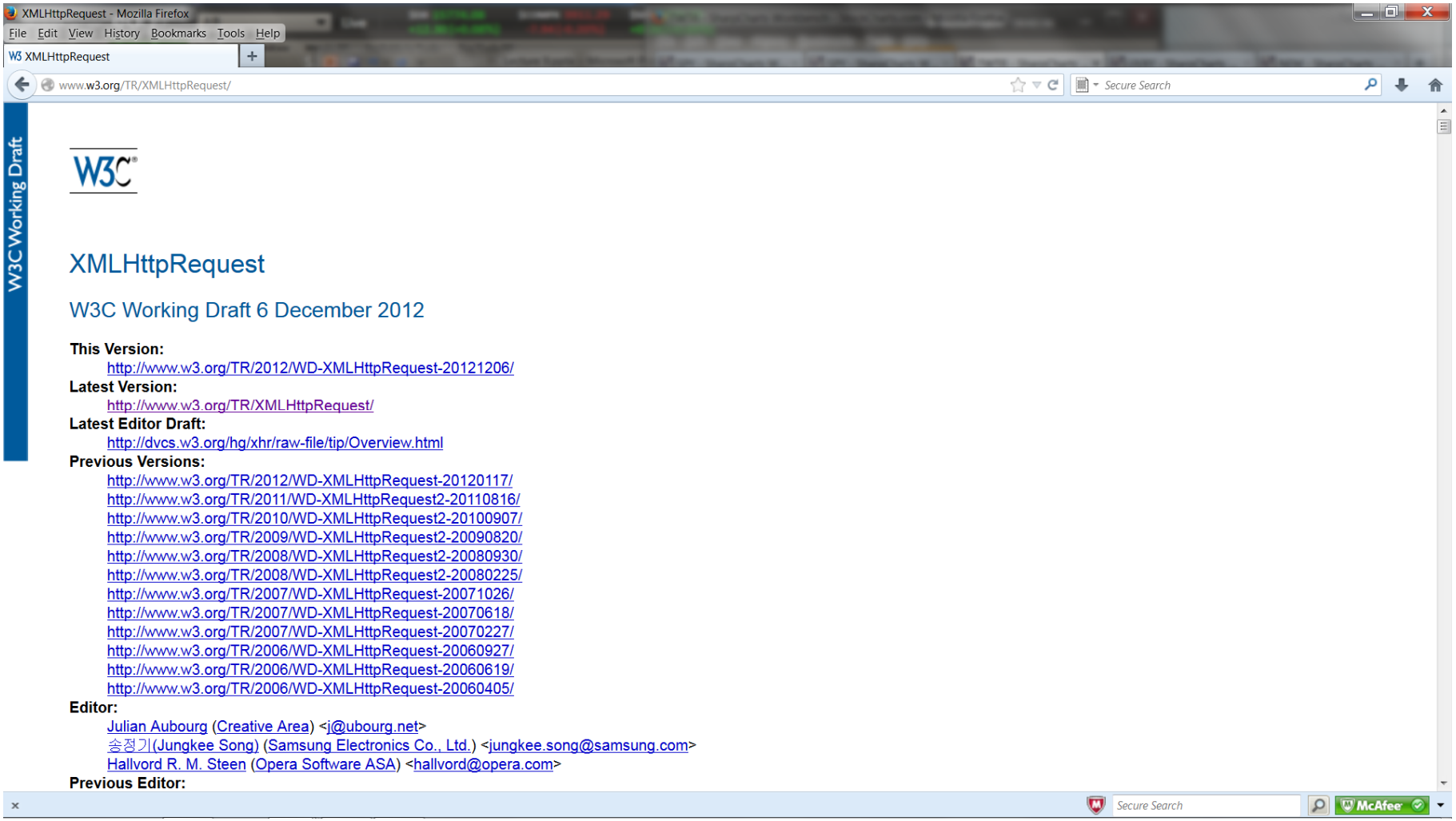
- The process flow of the diagram can be described by the following steps:

1. The user triggers an event, for example by releasing a key when typing in a name. This results in a JavaScript call to a function that initializes an XMLHttpRequest object.
2. The XMLHttpRequest object is configured with a request parameter that includes the ID of the component that triggered the event, and any value that the user entered. The XMLHttpRequest object then makes an asynchronous request to the web server.
3. On the web server, an object such as a servlet or listener handles the request. Data is retrieved from the data store, and a response is prepared containing the data in the form of an XML document.
4. Finally, the XMLHttpRequest object receives the XML data using a callback function, processes it, and updates the HTML DOM (Document Object Model) to display the page containing the new data.

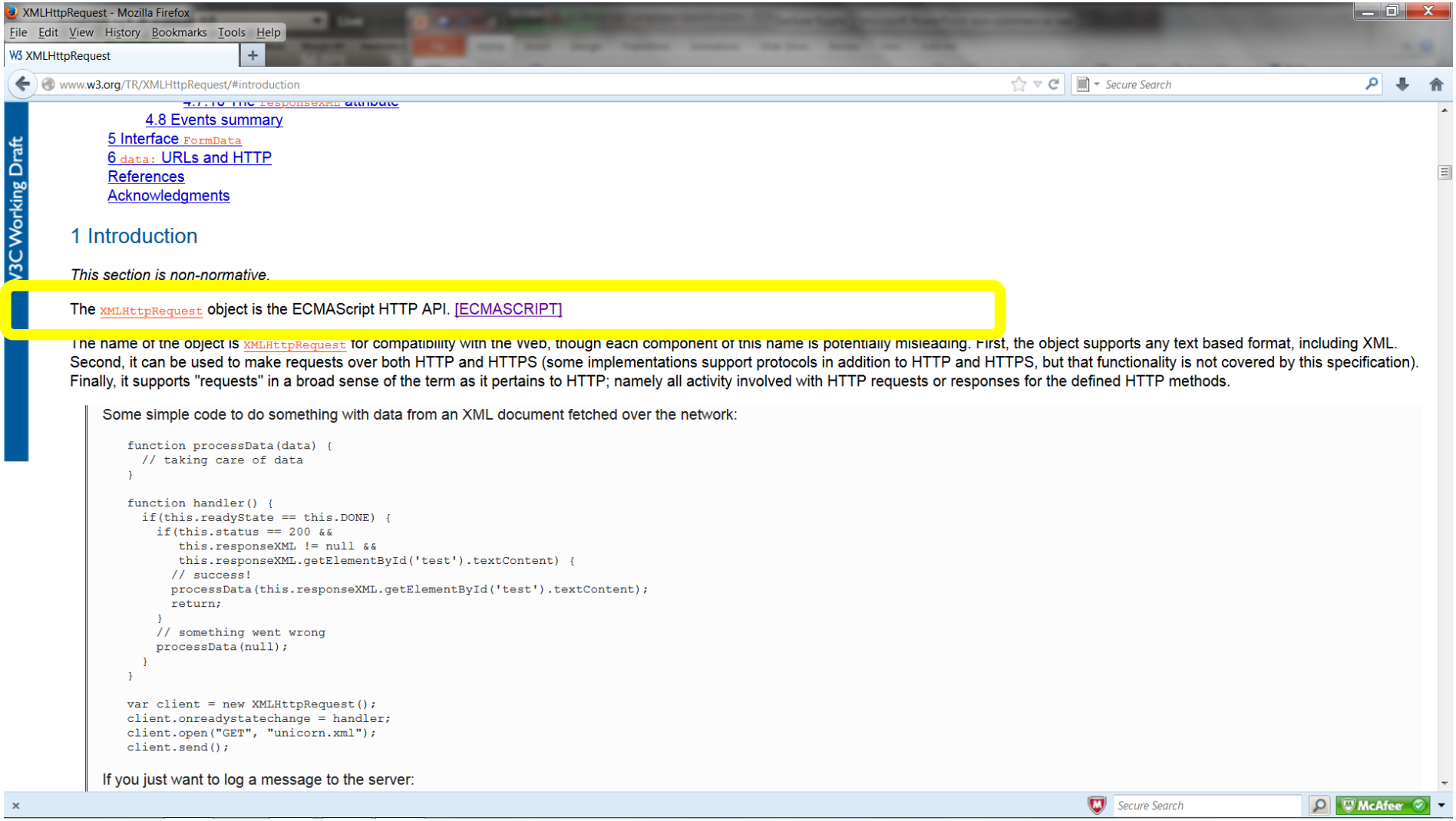
Why Ajax?

- Solve three key problems of Web apps
  - Coarse-grained updates
  - Synchronous: you are frozen while waiting for result
  - Extremely limited options for widgets (GUI elements)

What is XMLHttpRequest?

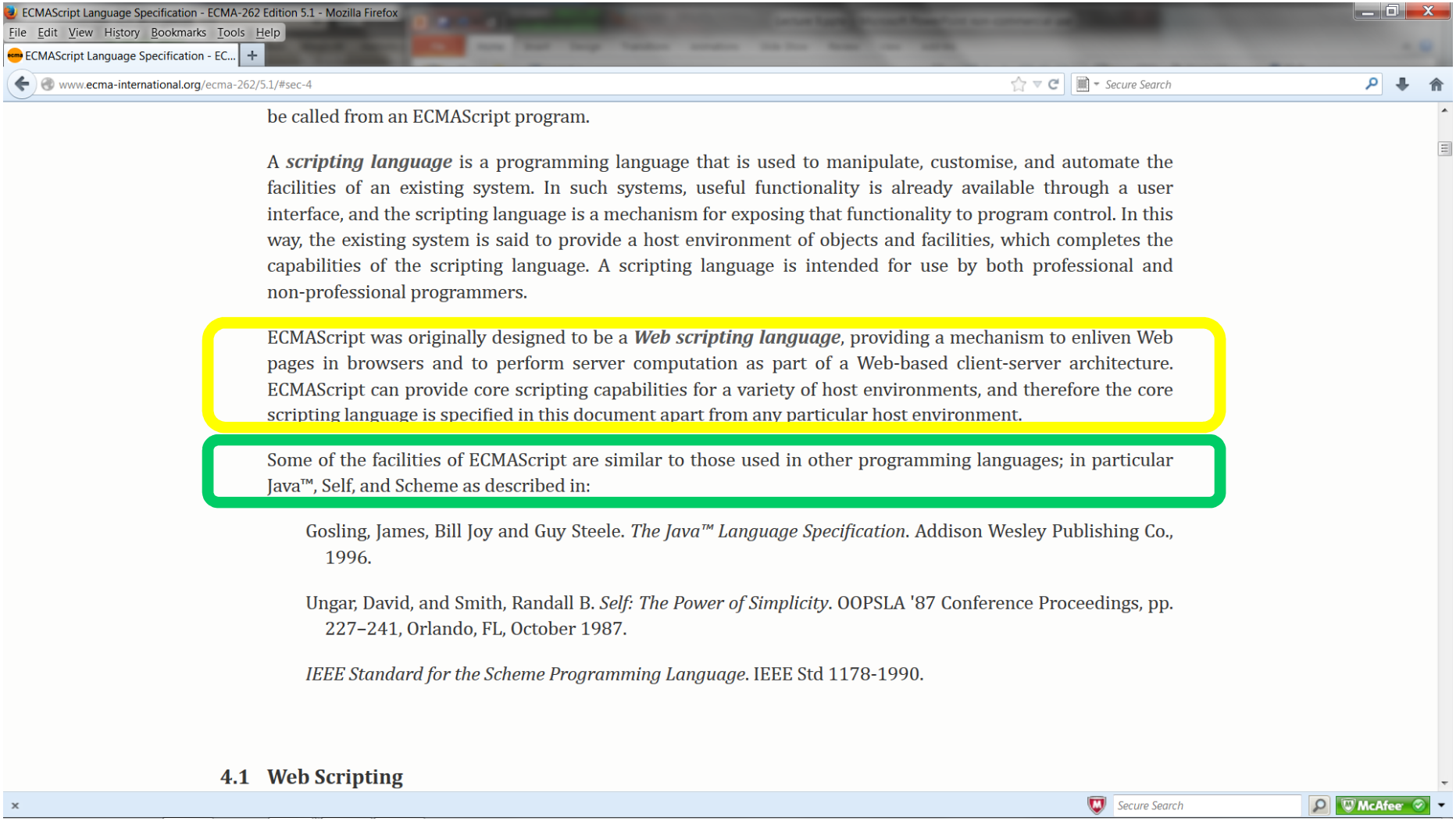


Origin of XMLHttpRequest?

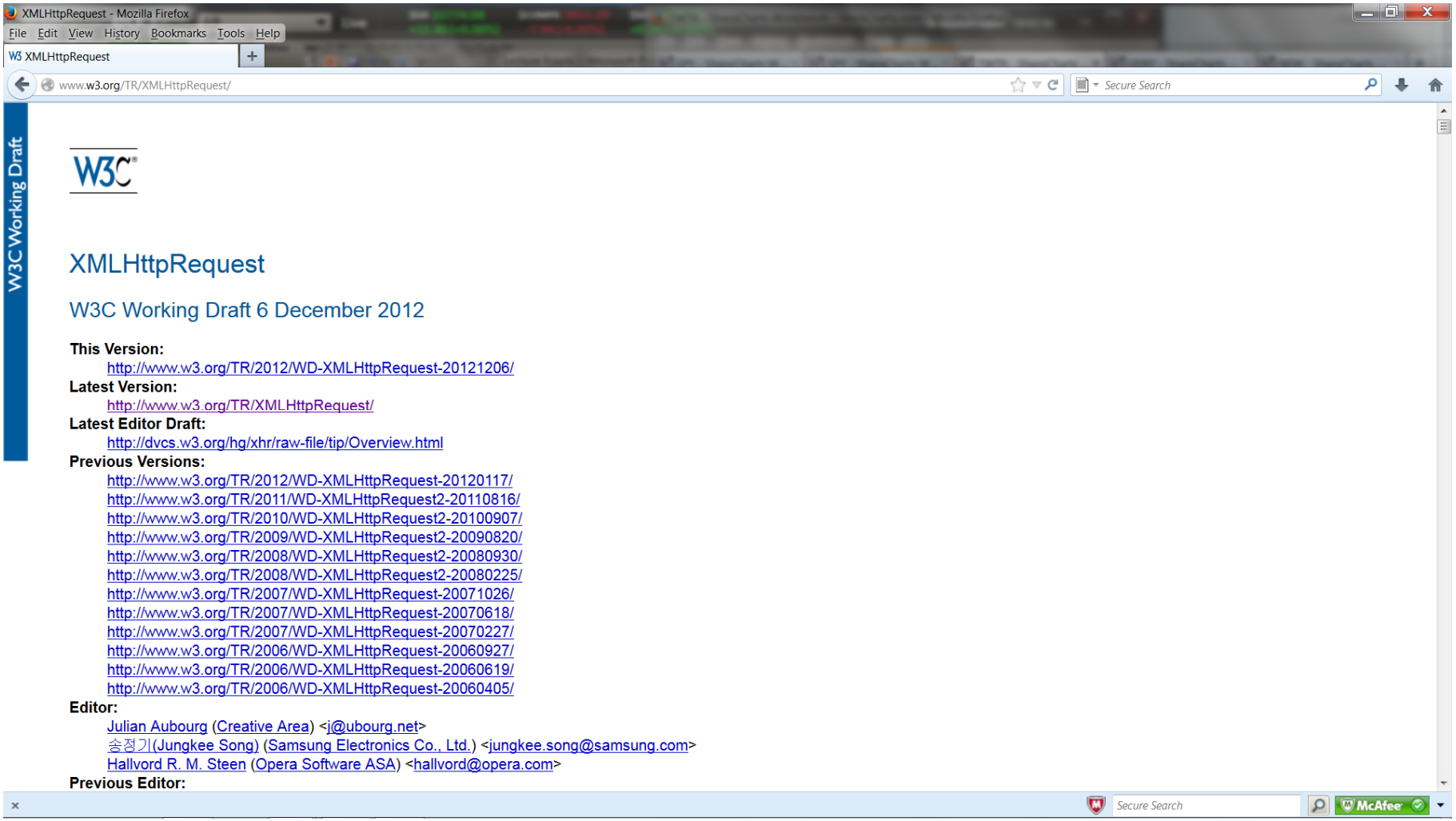


Origin of XMLHttpRequest?

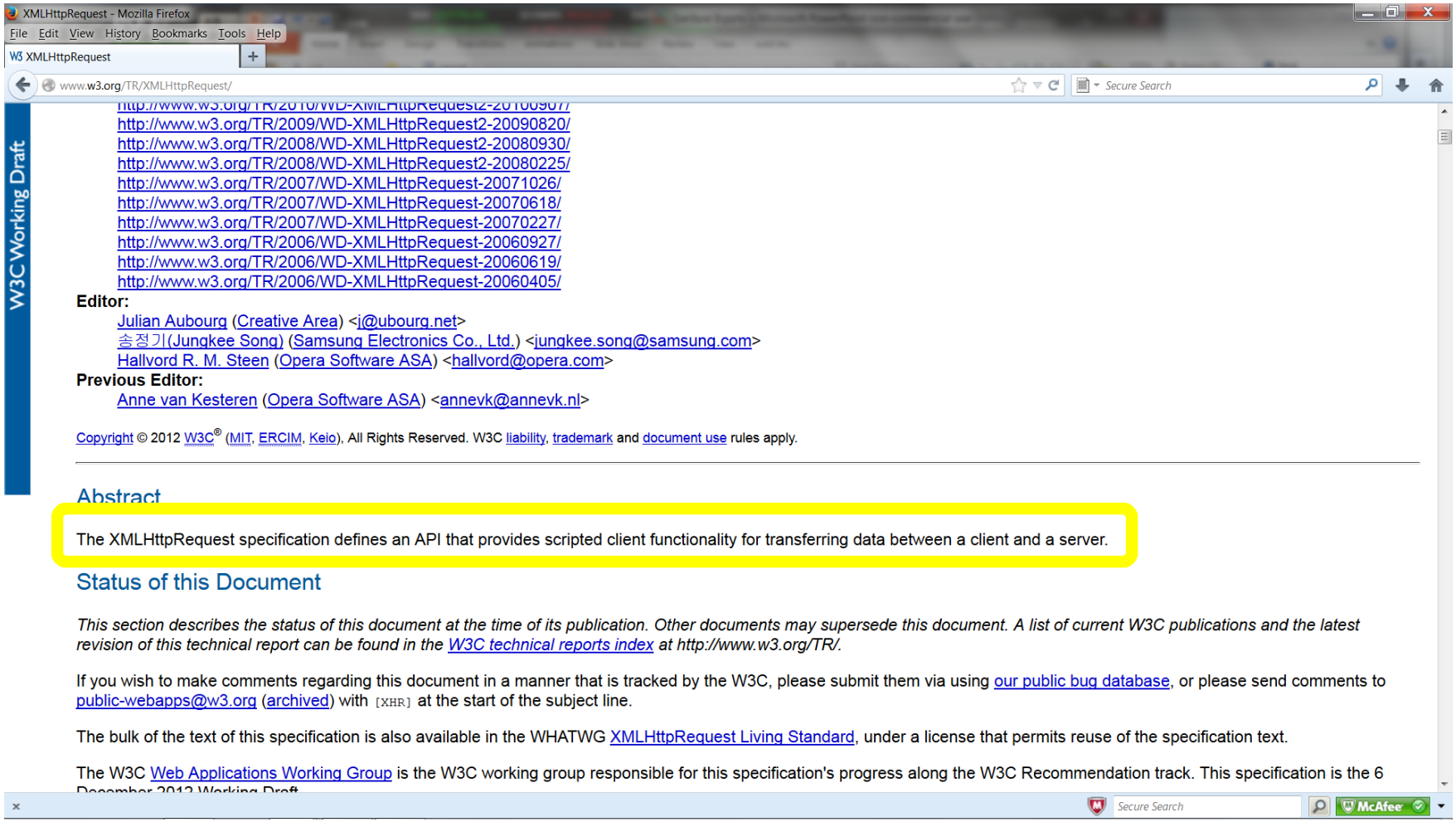
Origin of XMLHttpRequest?



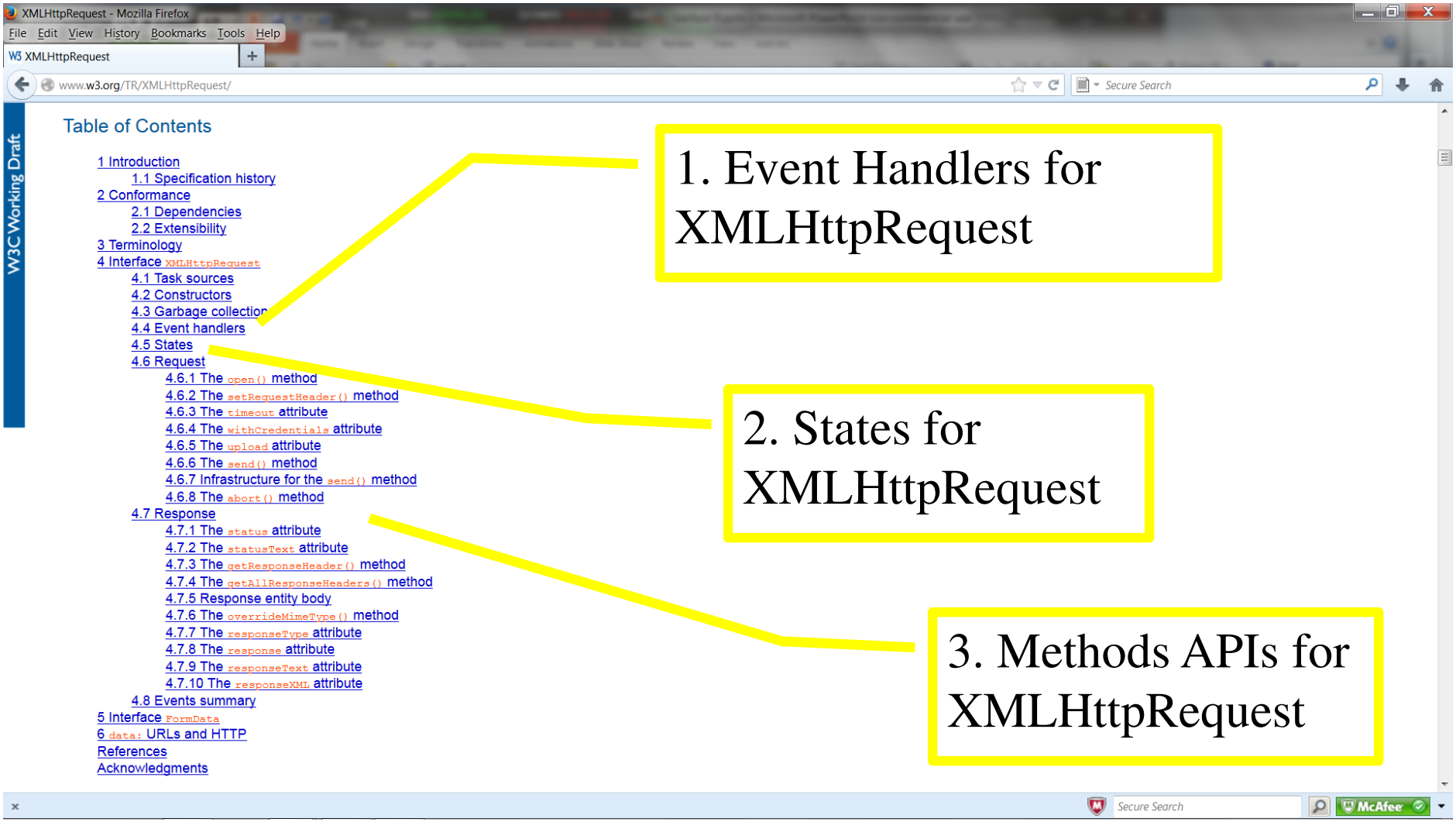
What is XMLHttpRequest?



What is XMLHttpRequest?

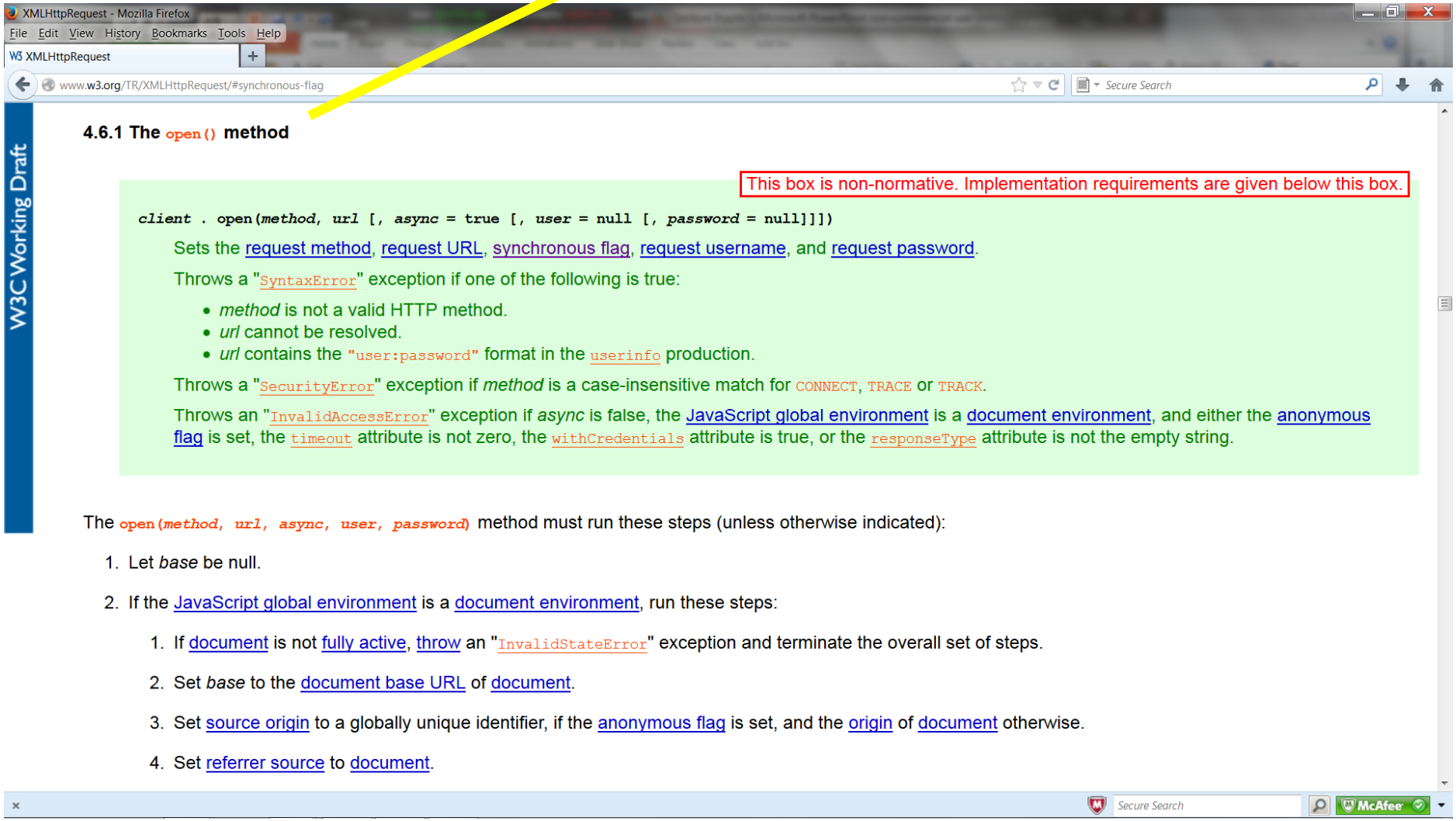


What is XMLHttpRequest?



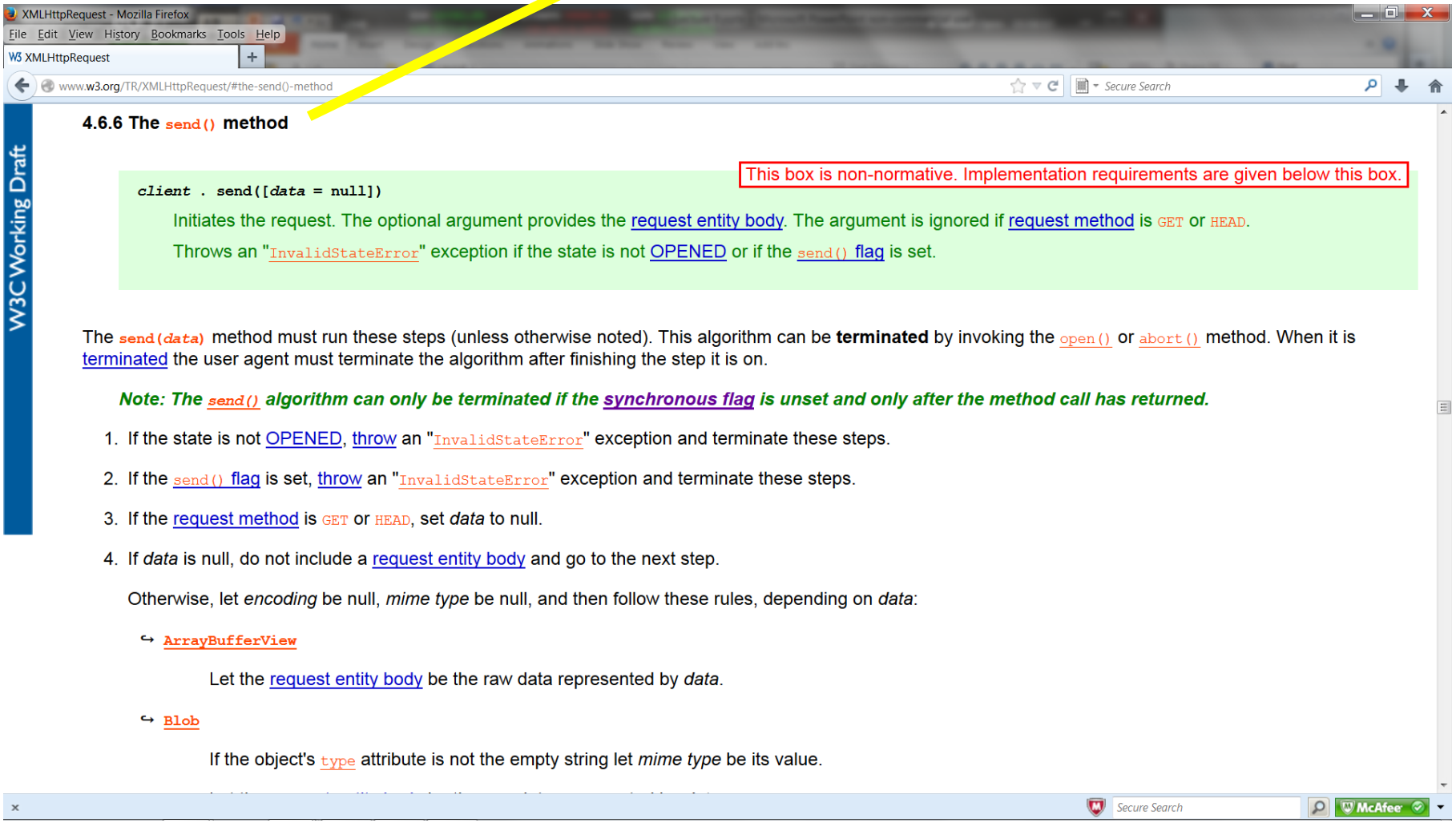
What is XMLHttpRequest?

1. **Open** method



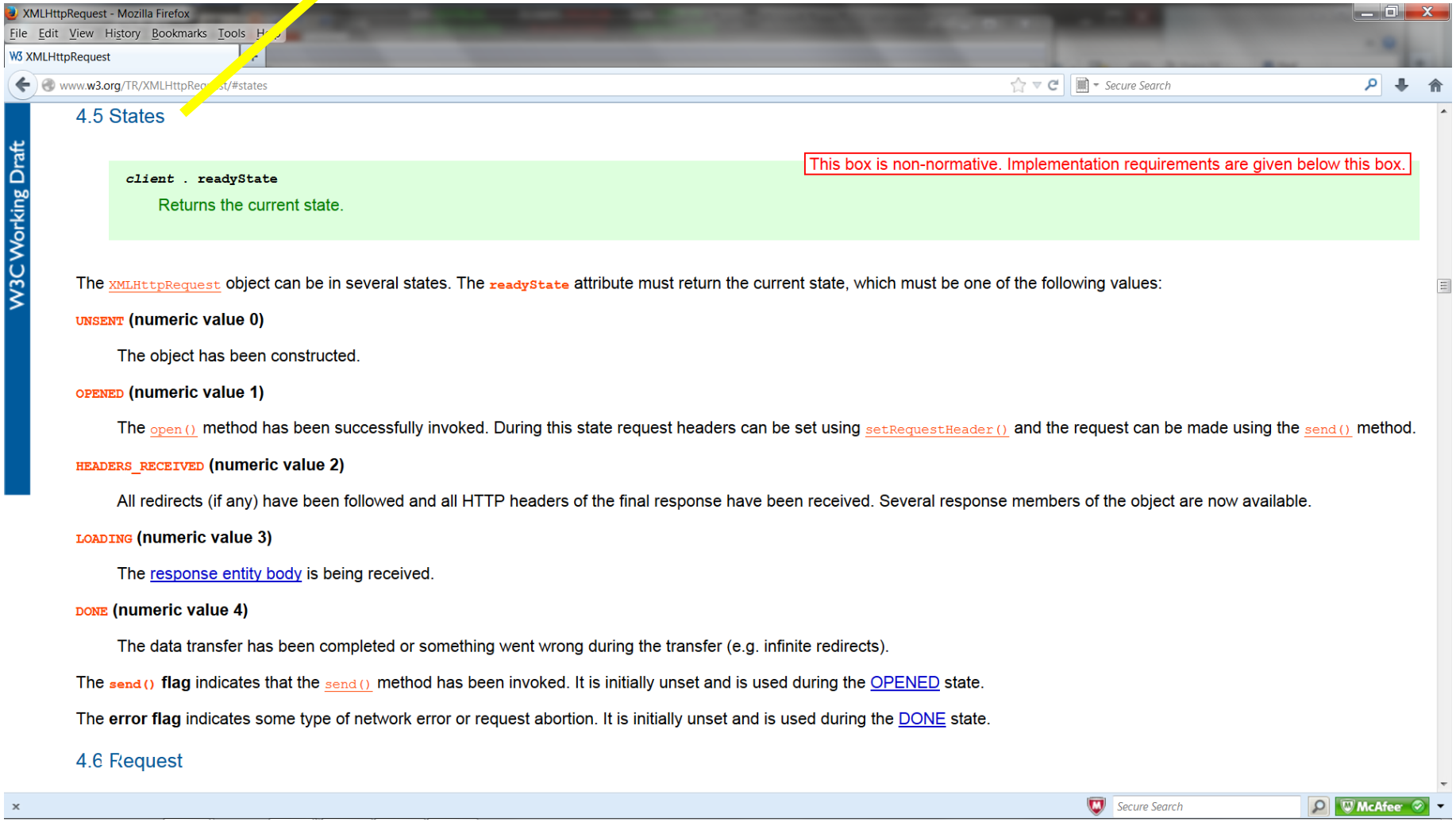
What is XMLHttpRequest?

2. **Send** method

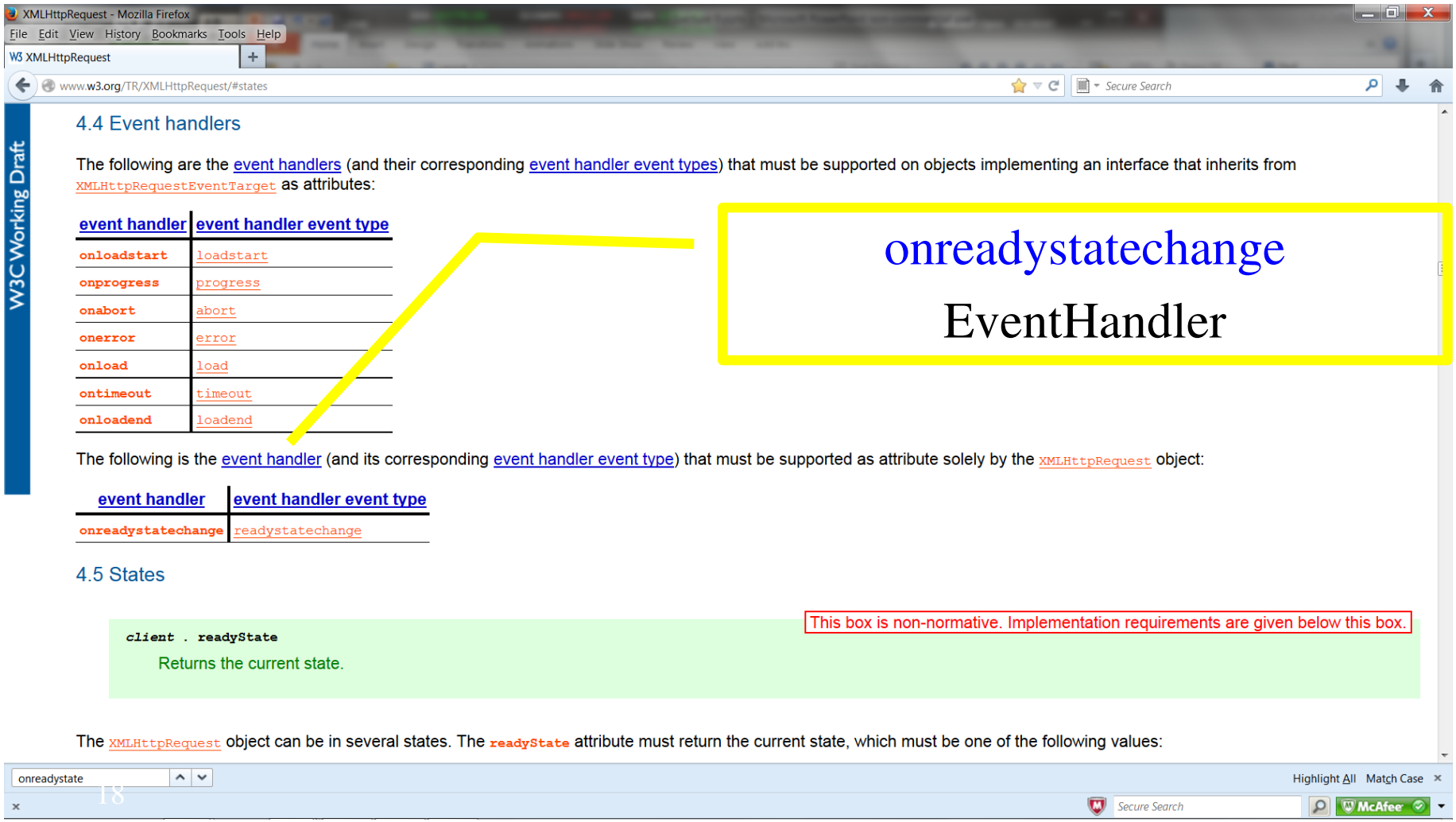


What is XMLHttpRequest?

readyState of XMLHttpRequest Object



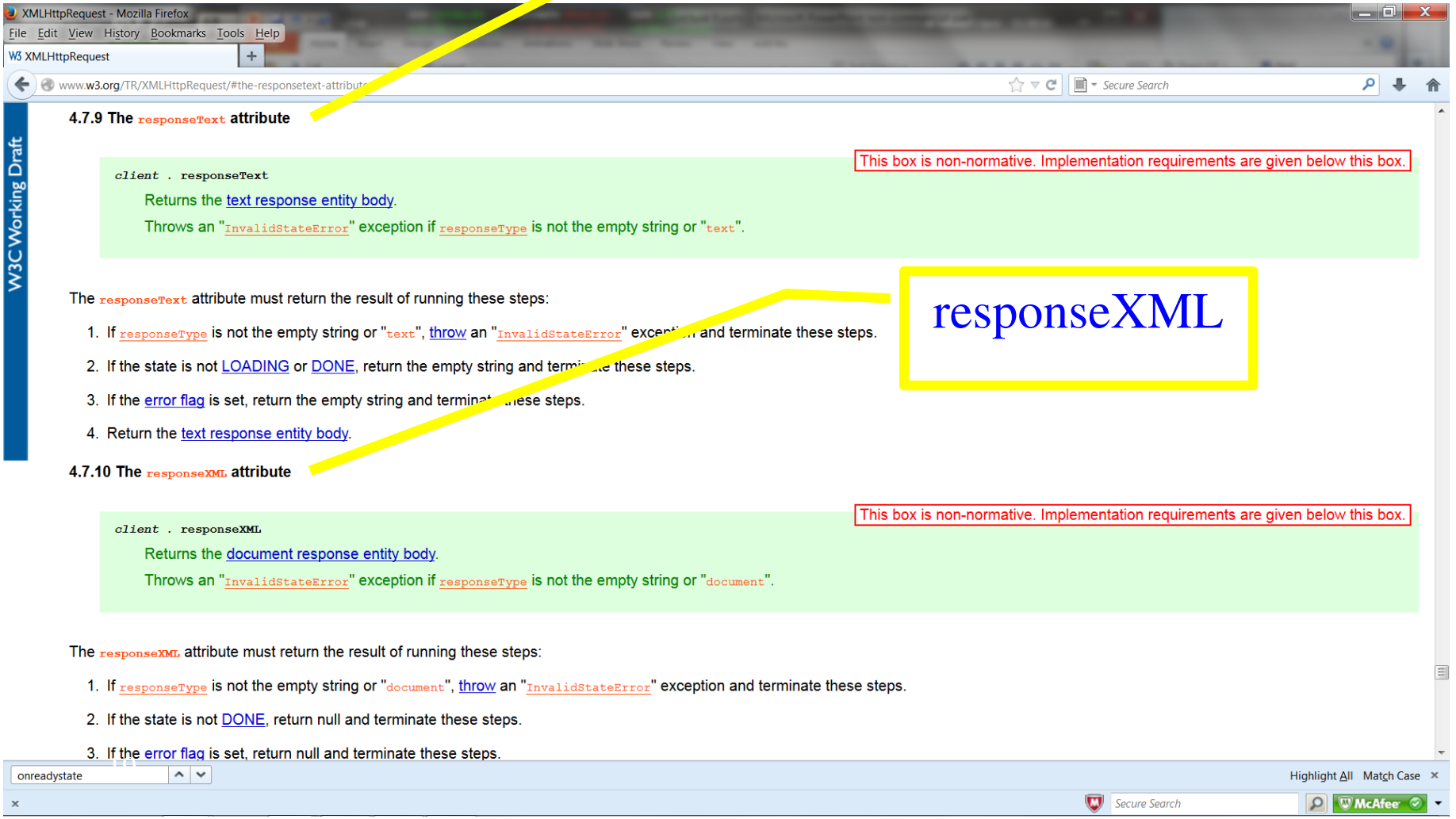
What is XMLHttpRequest?



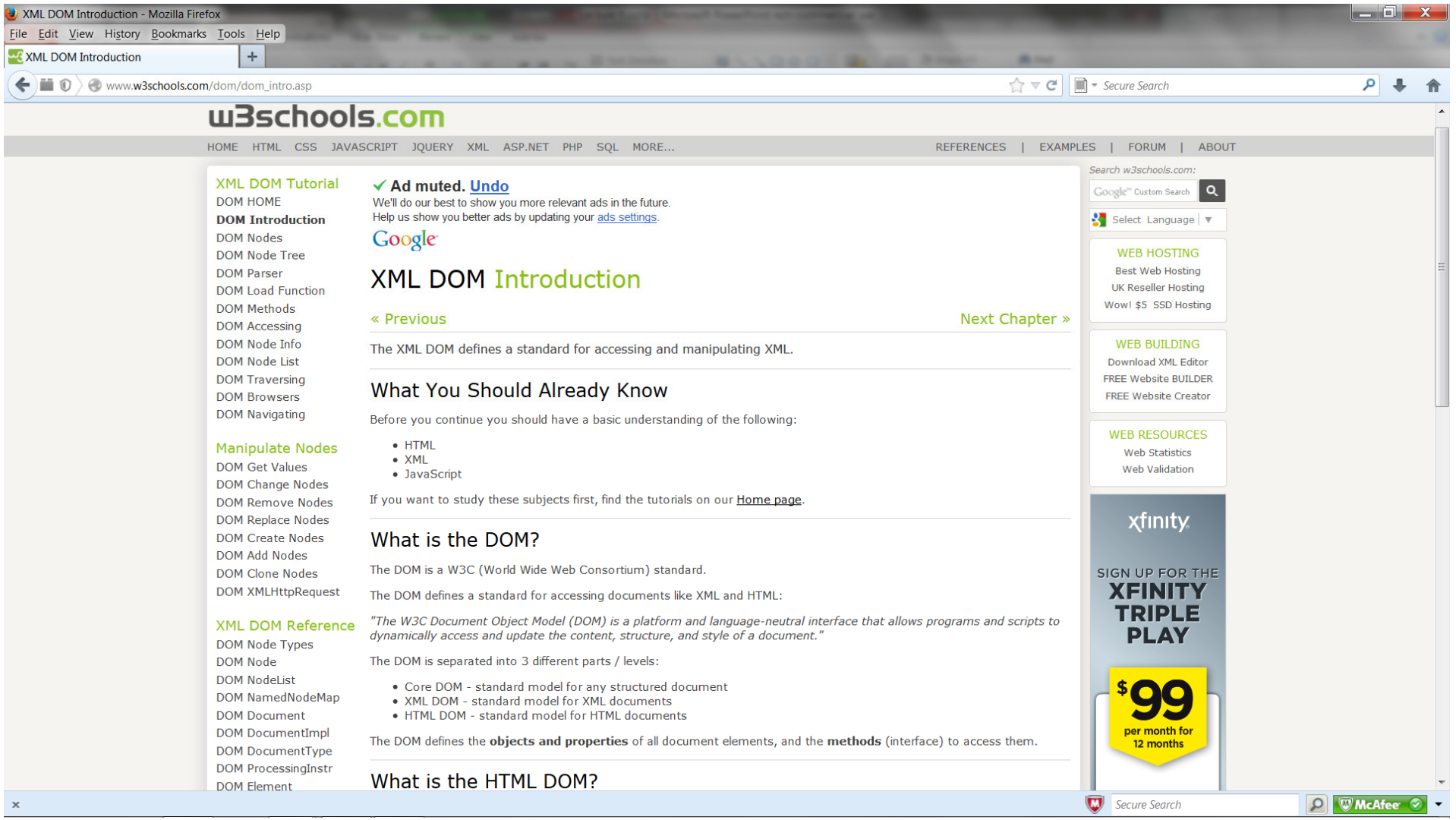
What is XMLHttpRequest?

responseText

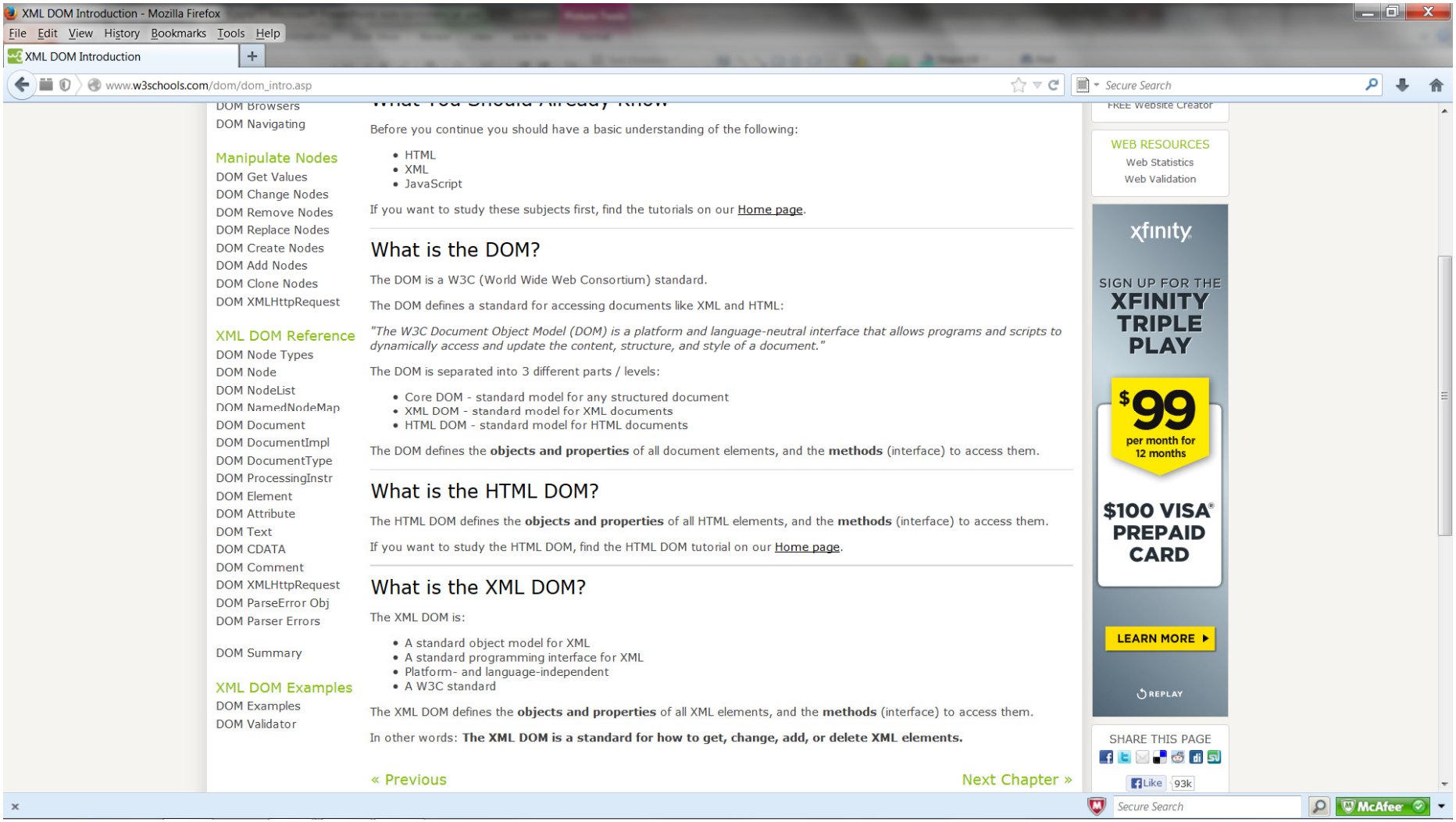
responseXML



What is DOM?

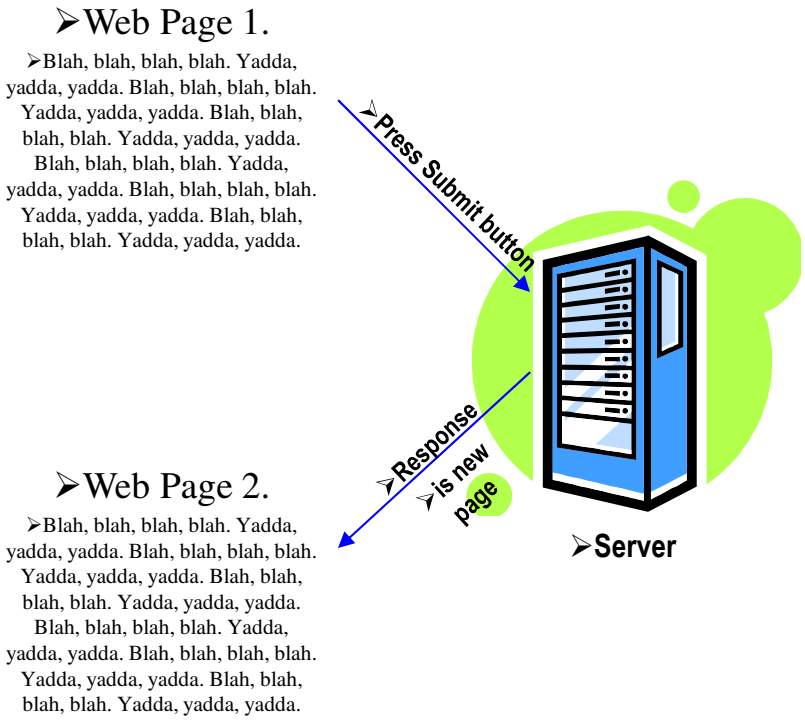

XML DOM Introduction - Mozilla Firefox
File Edit View History Bookmarks Tools Help
XML DOM Introduction
www.w3schools.com/dom/dom_intro.asp
Secure Search
w3schools.com
HOME HTML CSS JAVASCRIPT JQUERY XML ASP.NET PHP SQL MORE...
REFERENCES | EXAMPLES | FORUM | ABOUT
XML DOM Tutorial
DOM HOME
DOM Introduction
DOM Nodes
DOM Node Tree
DOM Parser
DOM Load Function
DOM Methods
DOM Accessing
DOM Node Info
DOM Node List
DOM Traversing
DOM Browsers
DOM Navigating
Manipulate Nodes
DOM Get Values
DOM Change Nodes
DOM Remove Nodes
DOM Replace Nodes
DOM Create Nodes
DOM Add Nodes
DOM Clone Nodes
DOM XMLHttpRequest
XML DOM Reference
DOM Node Types
DOM Node
DOM NodeList
DOM NamedNodeMap
DOM Document
DOM DocumentImpl
DOM DocumentType
DOM ProcessingInstr
DOM Element
Ad muted. Undo
We'll do our best to show you more relevant ads in the future.
Help us show you better ads by updating your ads settings.
Google
XML DOM Introduction
« Previous
Next Chapter »
The XML DOM defines a standard for accessing and manipulating XML.
What You Should Already Know
Before you continue you should have a basic understanding of the following:
HTML
XML
JavaScript
If you want to study these subjects first, find the tutorials on our Home page.
What is the DOM?
The DOM is a W3C (World Wide Web Consortium) standard.
The DOM defines a standard for accessing documents like XML and HTML:
"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."
The DOM is separated into 3 different parts / levels:
Core DOM - standard model for any structured document
XML DOM - standard model for XML documents
HTML DOM - standard model for HTML documents
The DOM defines the objects and properties of all document elements, and the methods (interface) to access them.
What is the HTML DOM?
Search w3schools.com:
Google Custom Search
Select Language
WEB HOSTING
Best Web Hosting
UK Reseller Hosting
Wow! $5 SSD Hosting
WEB BUILDING
Download XML Editor
FREE Website BUILDER
FREE Website Creator
WEB RESOURCES
Web Statistics
Web Validation
xfinity
SIGN UP FOR THE
XFINITY
TRIPLE
PLAY
$99
per month for
12 months
Secure Search
McAfee

What is DOM?

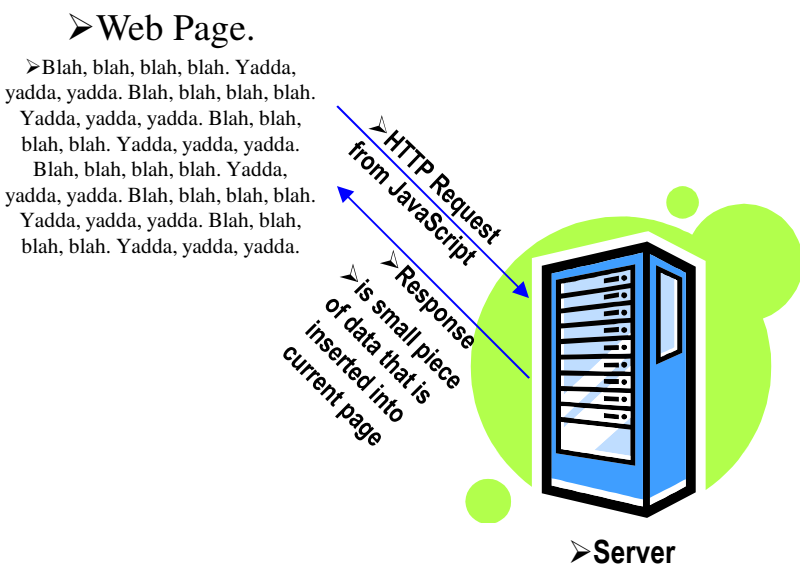

XML DOM Introduction - Mozilla Firefox
File Edit View History Bookmarks Tools Help
XML DOM Introduction
www.w3schools.com/dom/dom_intro.asp
Secure Search
DOM Browsers
DOM Navigating
Manipulate Nodes
DOM Get Values
DOM Change Nodes
DOM Remove Nodes
DOM Replace Nodes
DOM Create Nodes
DOM Add Nodes
DOM Clone Nodes
DOM XMLHttpRequest
XML DOM Reference
DOM Node Types
DOM Node
DOM NodeList
DOM NamedNodeMap
DOM Document
DOM DocumentImpl
DOM DocumentType
DOM ProcessingInstr
DOM Element
DOM Attribute
DOM Text
DOM CDATA
DOM Comment
DOM XMLHttpRequest
DOM ParseError Obj
DOM Parser Errors
DOM Summary
XML DOM Examples
DOM Examples
DOM Validator
Before you continue you should have a basic understanding of the following:
HTML
XML
JavaScript
If you want to study these subjects first, find the tutorials on our Home page.
What is the DOM?
The DOM is a W3C (World Wide Web Consortium) standard.
The DOM defines a standard for accessing documents like XML and HTML:
"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."
The DOM is separated into 3 different parts / levels:
Core DOM - standard model for any structured document
XML DOM - standard model for XML documents
HTML DOM - standard model for HTML documents
The DOM defines the objects and properties of all document elements, and the methods (interface) to access them.
What is the HTML DOM?
The HTML DOM defines the objects and properties of all HTML elements, and the methods (interface) to access them.
If you want to study the HTML DOM, find the HTML DOM tutorial on our Home page.
What is the XML DOM?
The XML DOM is:
A standard object model for XML
A standard programming interface for XML
Platform- and language-independent
A W3C standard
The XML DOM defines the objects and properties of all XML elements, and the methods (interface) to access them.
In other words: The XML DOM is a standard for how to get, change, add, or delete XML elements.
« Previous
Next Chapter »
FREE Website Creator
WEB RESOURCES
Web Statistics
Web Validation
xfinity
SIGN UP FOR THE XFINITY TRIPLE PLAY
$99 per month for 12 months
$100 VISA® PREPAID CARD
LEARN MORE
REPLAY
SHARE THIS PAGE
Like 93k
Secure Search
McAfee

Traditional Web Apps vs. Ajax Apps

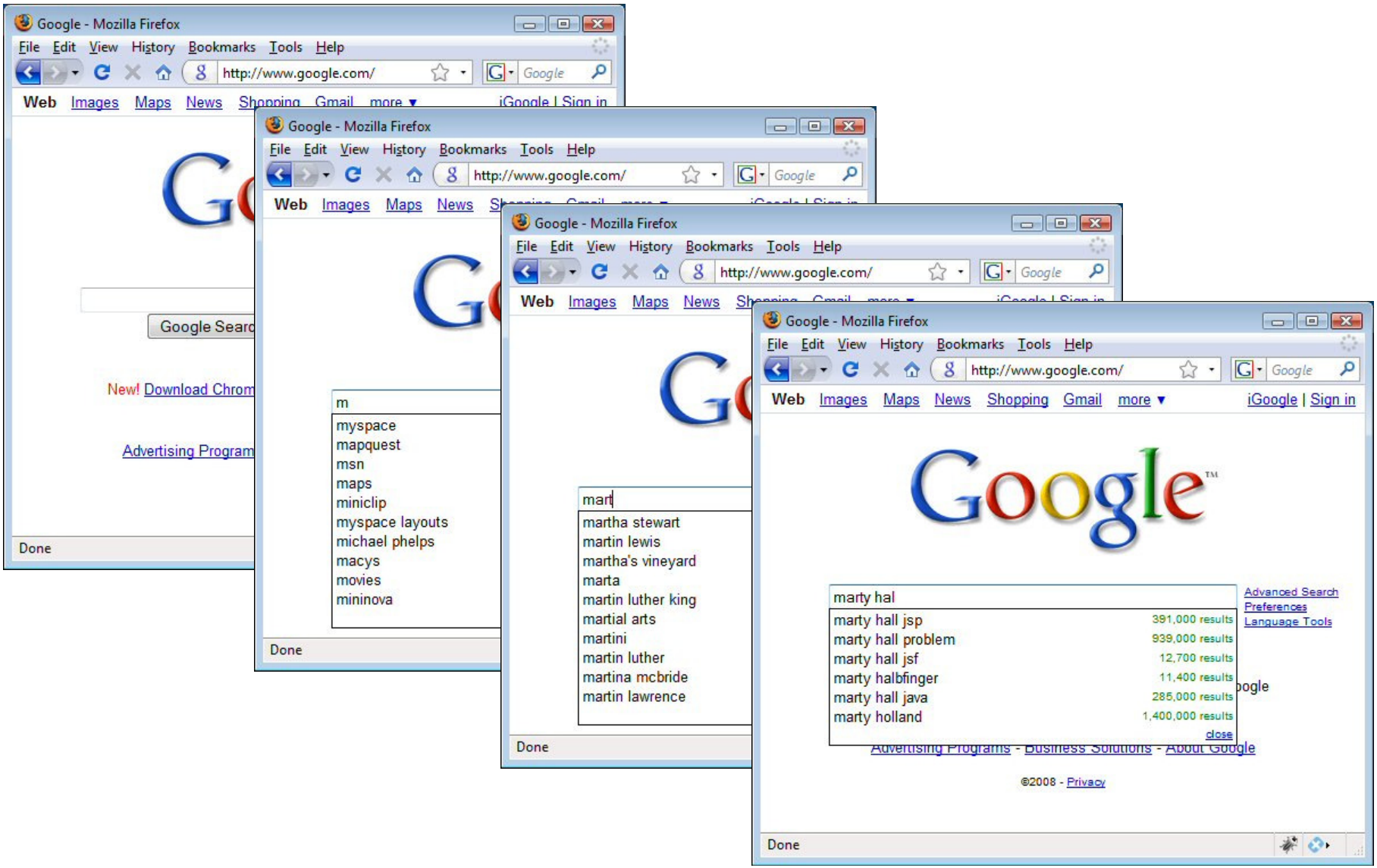
- **Traditional Web Apps: Infrequent Large Updates**



- **Ajax Apps: Frequent Small Updates**

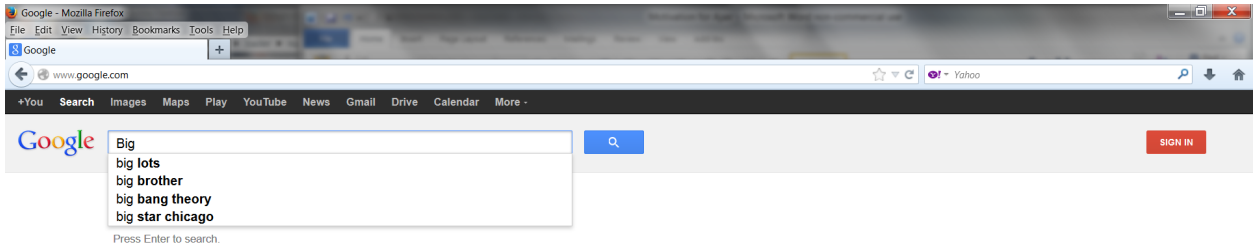


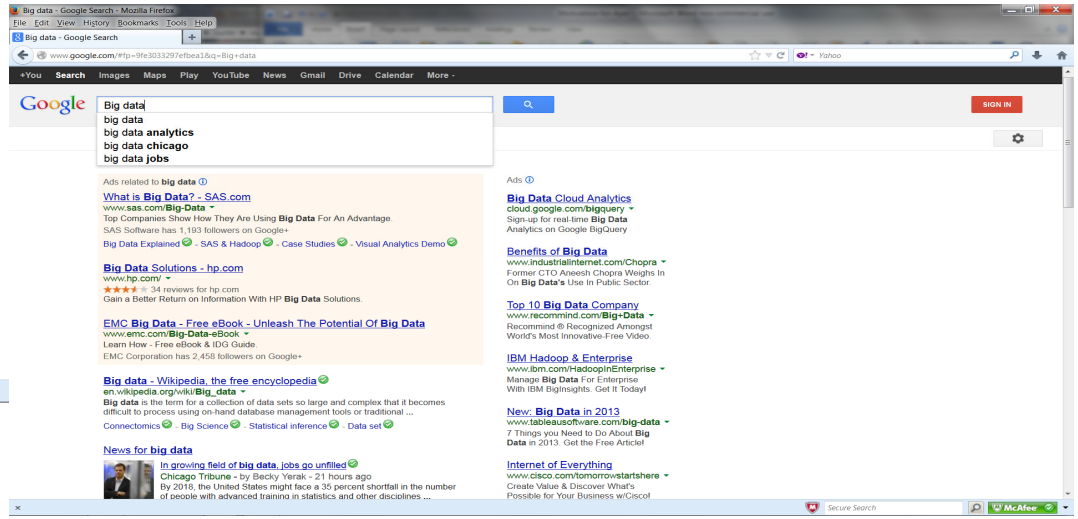
Google Home Page
(formerly Google Suggest)



Why Ajax?

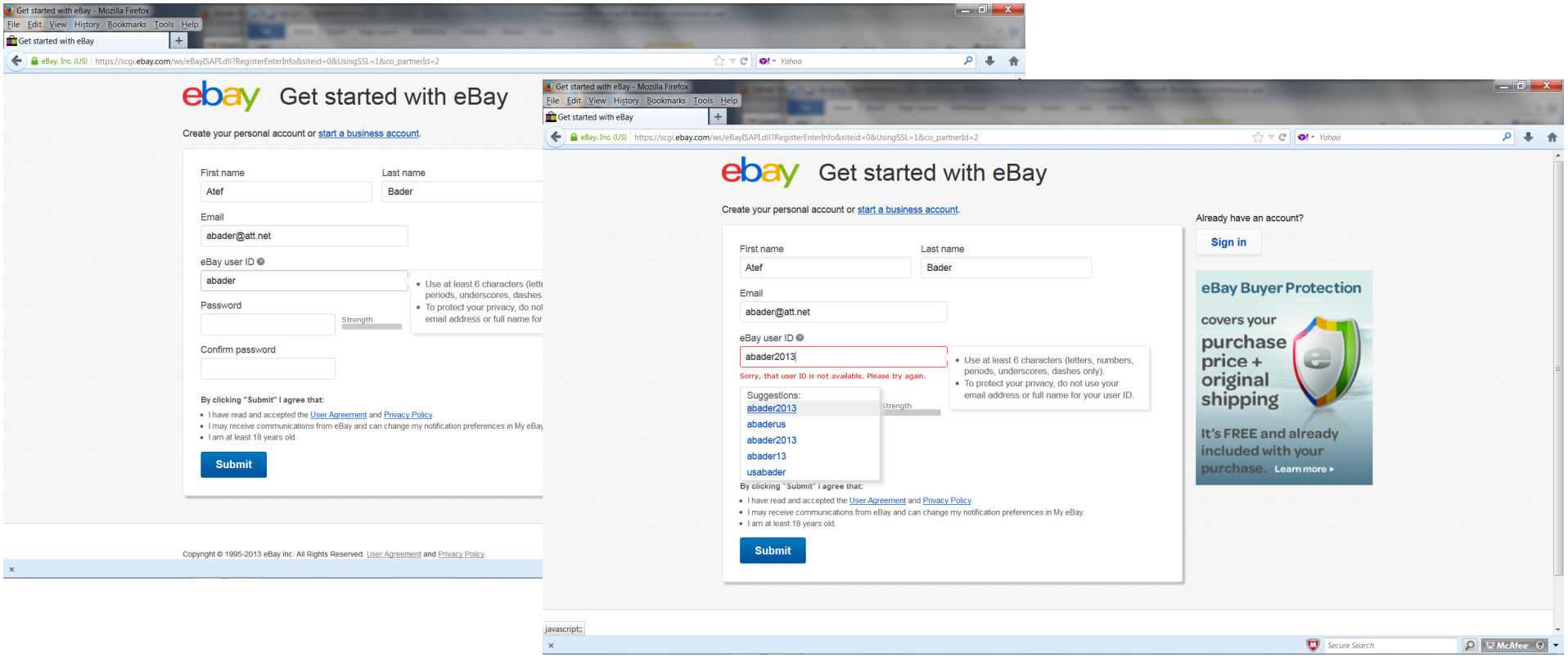
- **Scenario 1**
  - **You did NOT press the enter key yet.; all what you are doing is only typing your search keyword, and AJAX doing the work in the background (asynchronously AJAX/Google trying to guess the next keyword you are trying to type and giving you suggestions and partial results)**



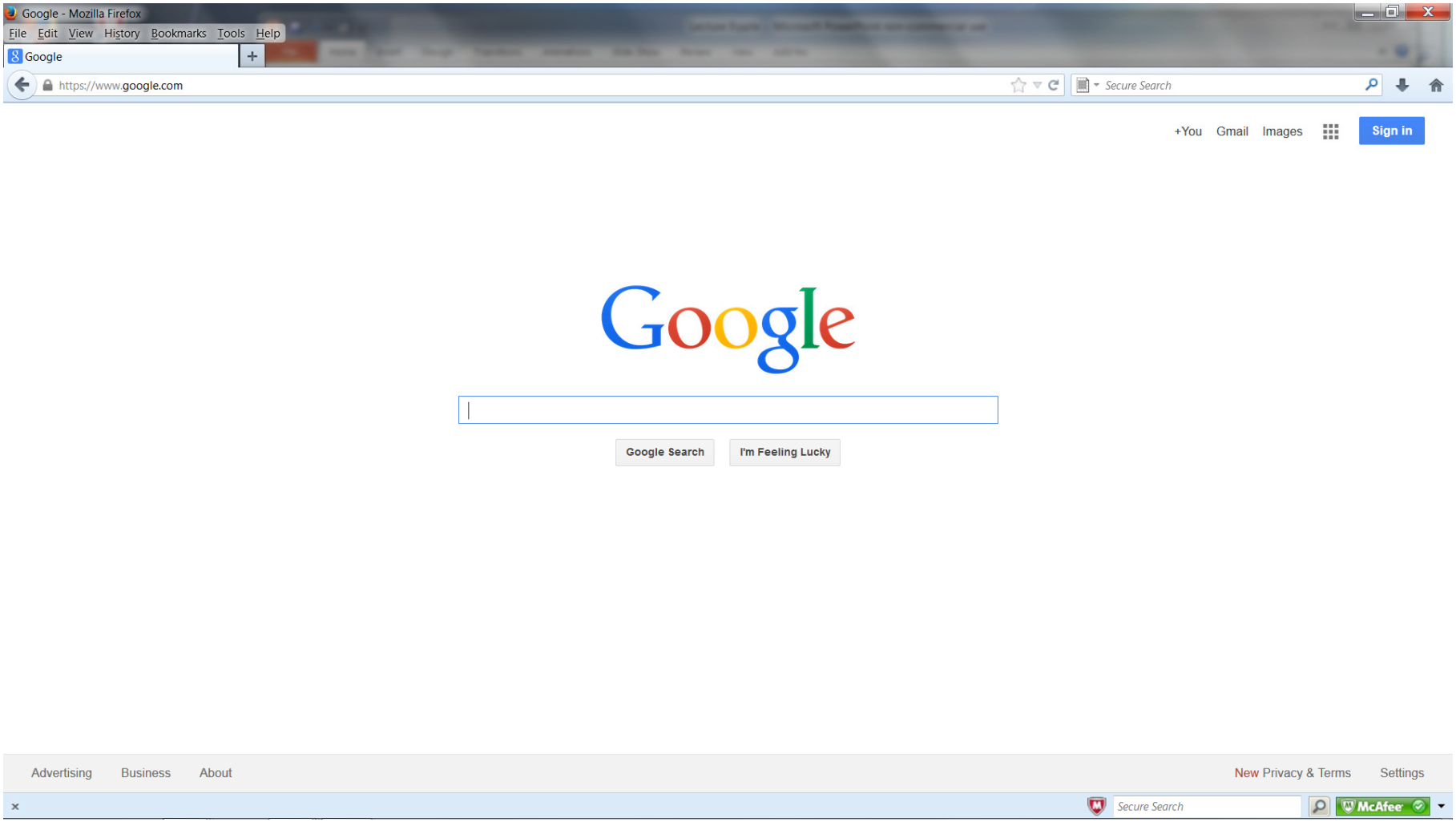


Why Ajax?

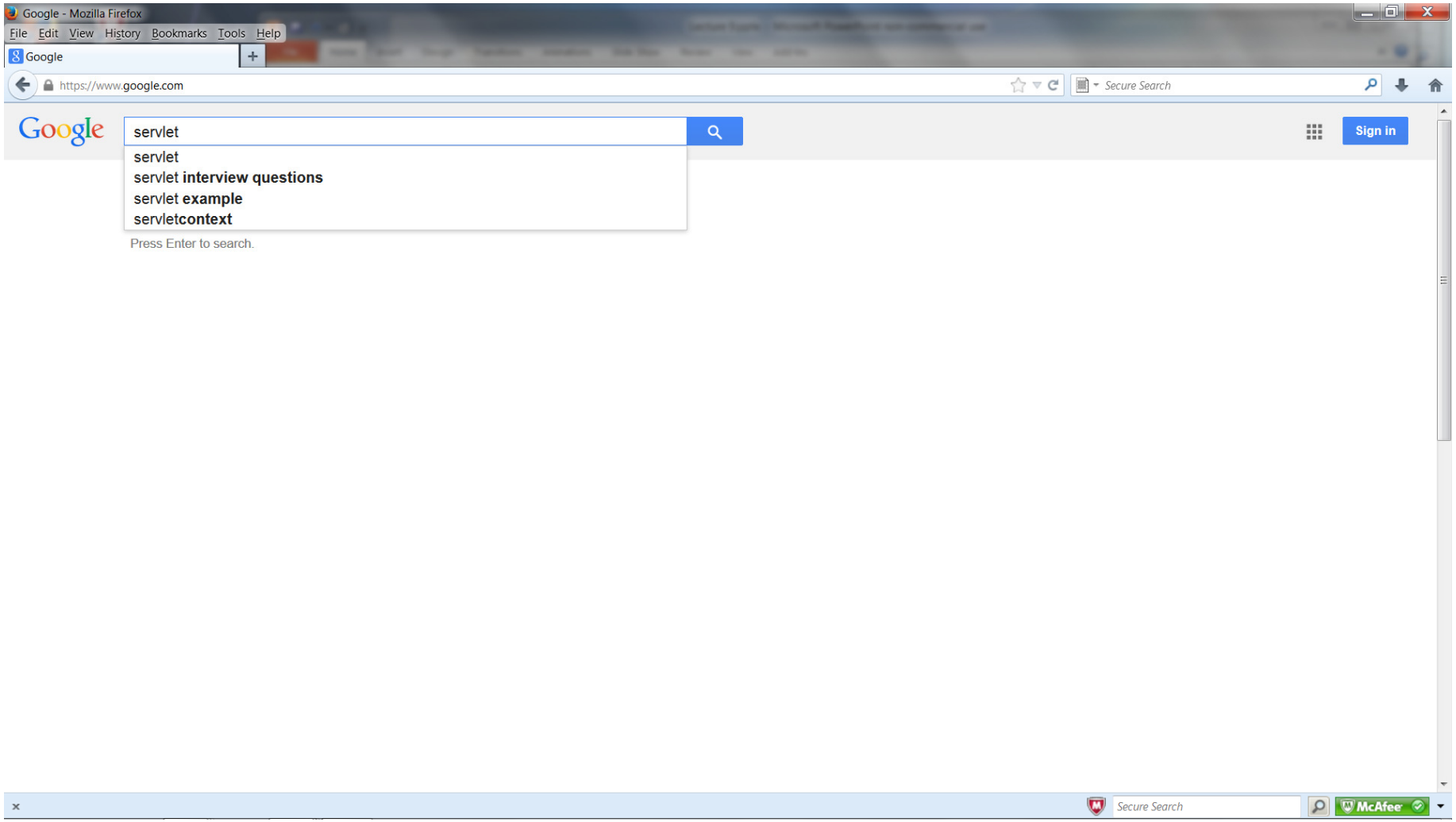
- **Scenario 2**
  - **You are trying to create a new login, and entered the Login ID you want but before going to the next field to enter the password you want, AJAX in the background asynchronously checks with the server to see if the requested Login ID is used by someone else and provides you with a list of potential/candidate Login IDs that you might choose from.**



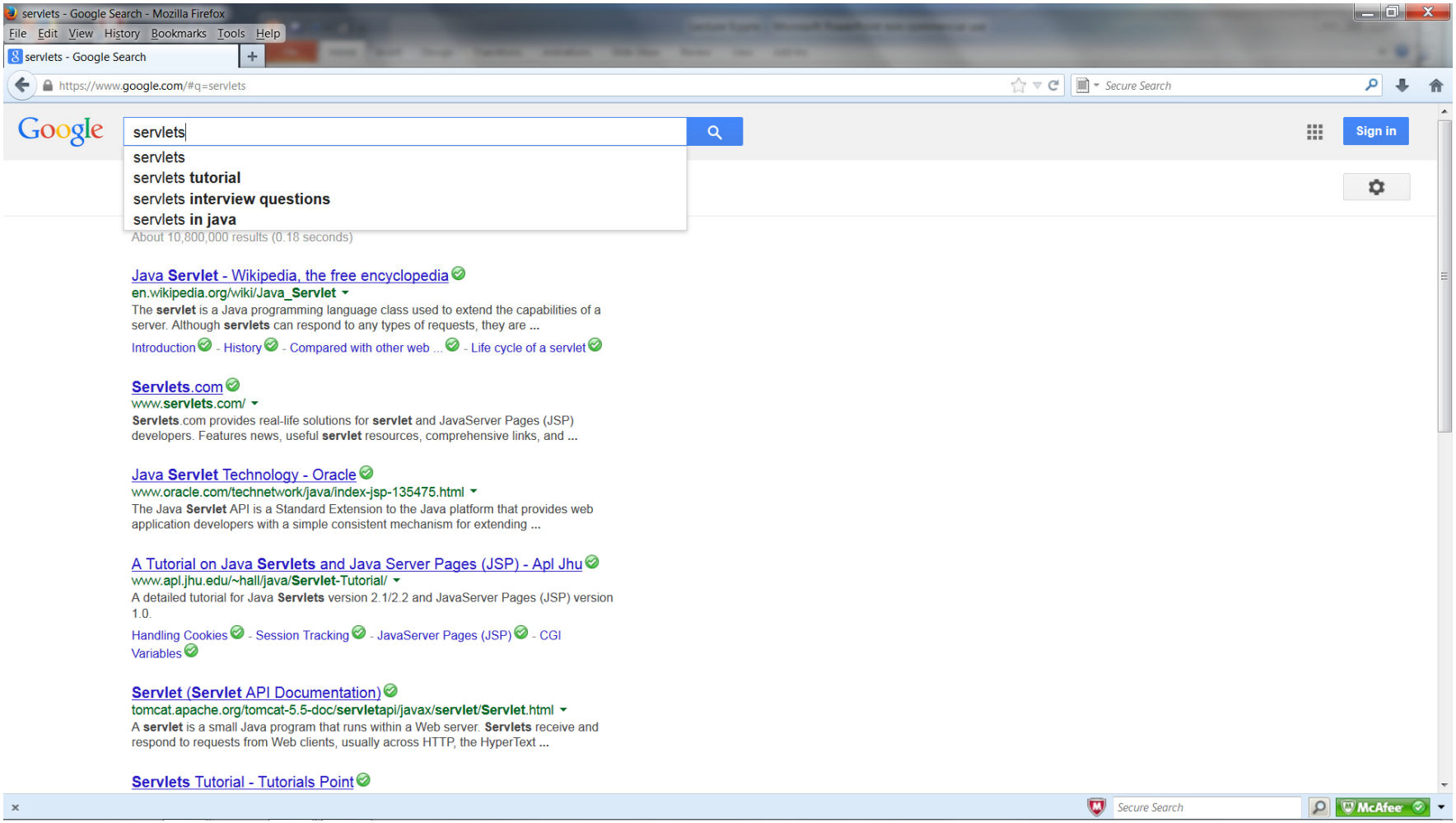
Google Home Page



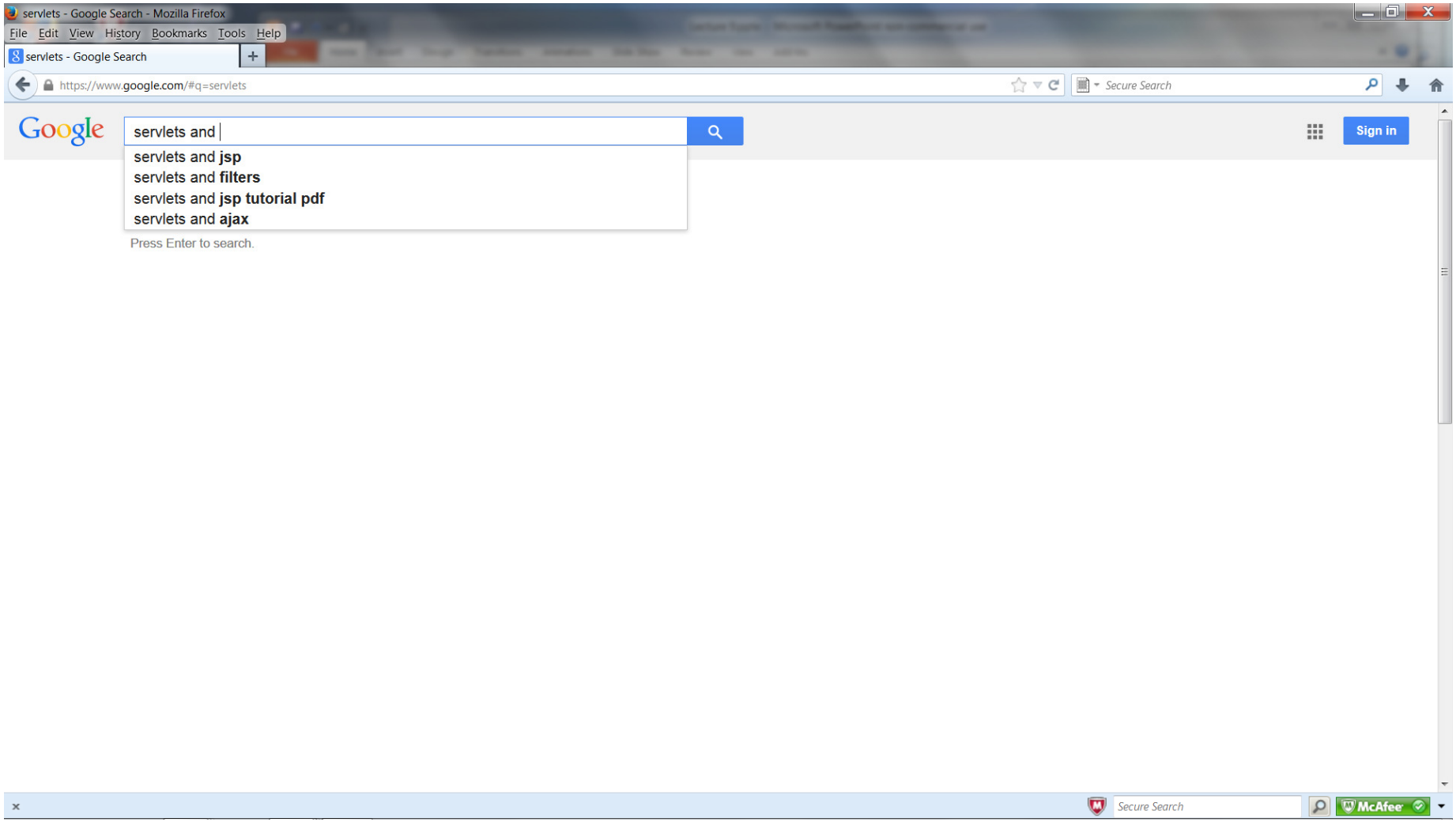
Google Home Page



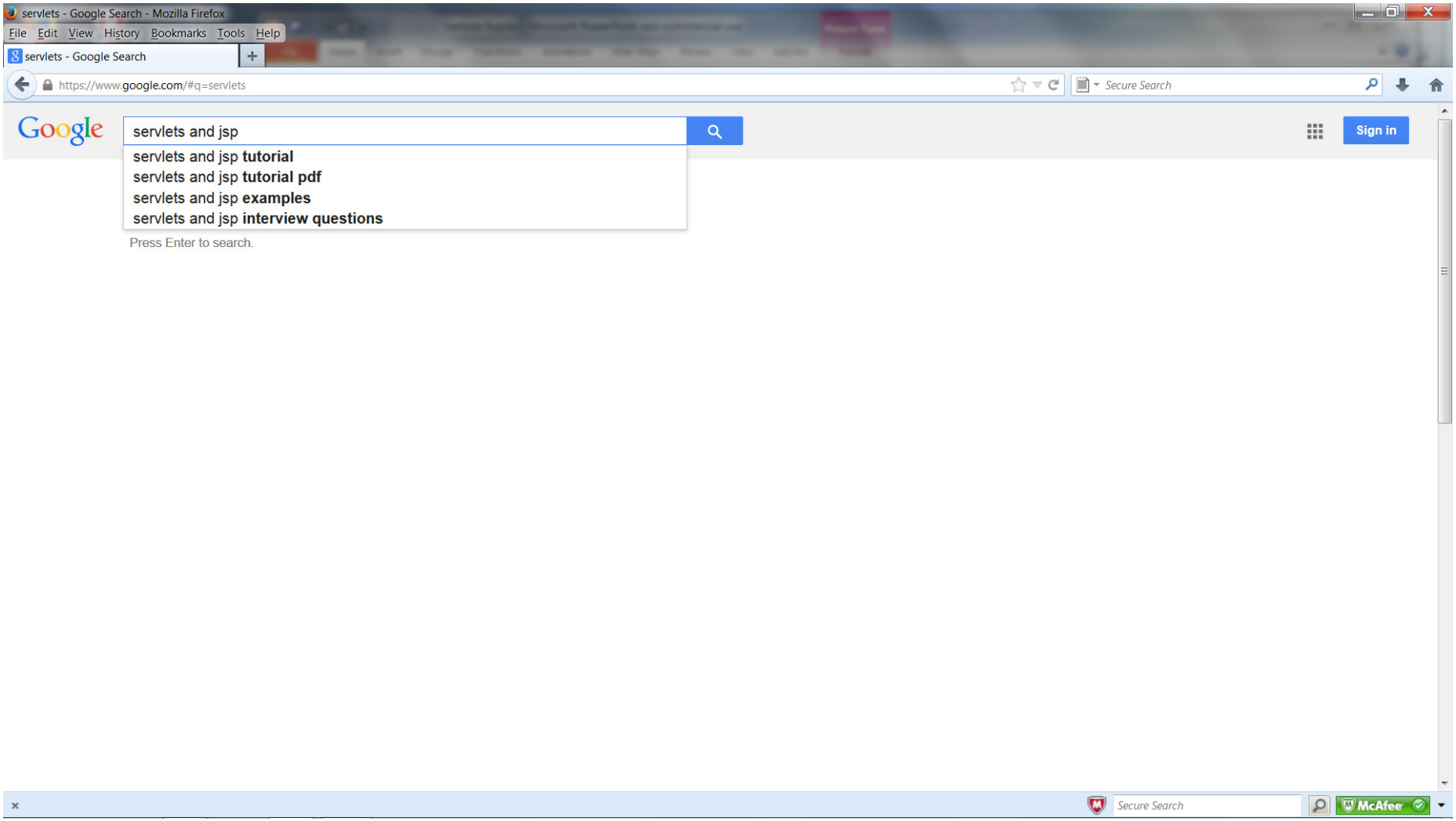
Google Home Page



Google Home Page



Google Home Page



The Basic Ajax Process

- JavaScript
  - Define an object for sending HTTP requests
  - Initiate request
    - Get request object
    - Designate an anonymous response handler function
      - Supply as onreadystatechange attribute of request
    - Initiate a GET or POST request
    - Send data
  - Handle response
    - Wait for readyState of 4 and HTTP status of 200
    - Extract return text with responseText or responseXML
    - Do something with result
- HTML
  - Load JavaScript
  - Designate control that initiates request
  - Give ids to input elements and to output placeholder region

The Basic Ajax Process

- **For JavaScripts, 3 Simple Steps:**
  1. getRequestObject
  2. sendRequest
  3. handleResponse

Define a Request Object

```
function getRequestObject() {
  if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else if (window.ActiveXObject) {
    return(new ActiveXObject("Microsoft.XMLHTTP"));
  } else {
    return(null);
  }
}
```

- Version for Firefox, Netscape 5+, Opera, Safari, Mozilla, Chrome, Internet Explorer 7, and IE 8.
- Version for Internet Explorer 5.5 and 6
- Fails on older and nonstandard browsers. You don't want to do "throw new Error(…)" here because this is for very old browsers, and Error came only in JavaScript 1.5.

Initiate Request

```
function sendRequest() {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { handleResponse(request) };
  request.open("GET", "message-data.html", true);
  request.send(null);
}
```

- Code to call when server responds
- URL of server-side resource. Must be on same
- server that page was loaded from (*).
- POST data
- (always null for GET requests)
- Don't wait for response (Send request asynchronously)

- (*) The Same Source (or Same Origin) rule is relaxed somewhat in Firefox 3.5 and later. But, you can only make cross-site requests if the URL you connect to puts in a header specifically allowing it. For details, see http://www.petefreitag.com/item/703.cfm and https://developer.mozilla.org/En/HTTP_access_control

Handle Response

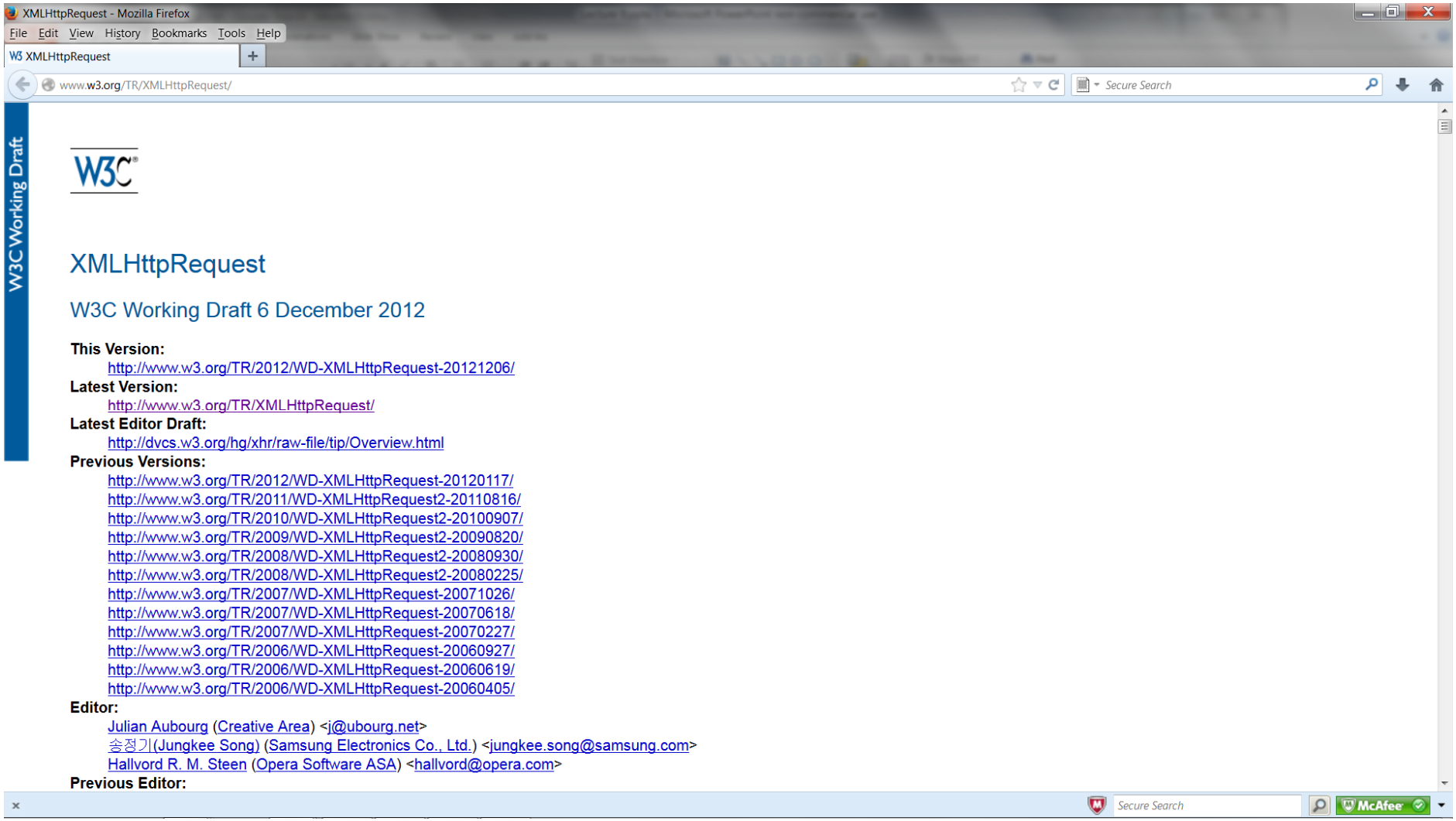
```
function handleResponse(request) {
  if (request.readyState == 4) {
    alert(request.responseText);
  }
}
```

- 4 means response from server is complete
- (handler gets invoked multiple times –
- ignore the first ones)

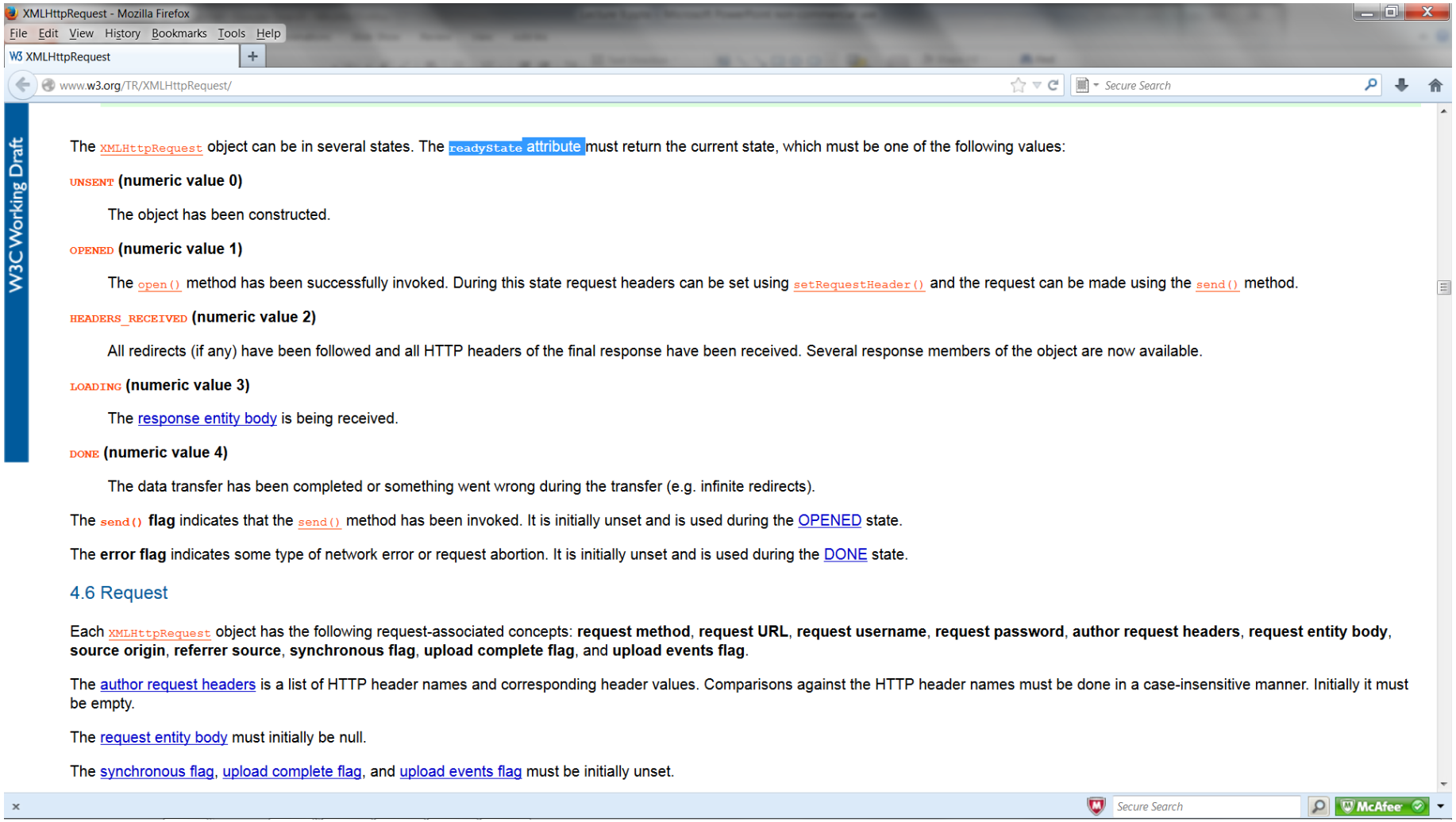
- Text of server response

- Pop up dialog box

XMLHttpRequest API and Docs



XMLHttpRequest : *readyState* attribute



First-class Functions in JavaScript

- JavaScript lets you pass functions around

```
function doSomethingWithResponse() { code }
request.onreadystatechange = doSomethingWithResponse;
```

  - This is somewhat similar to function pointers in C/C++
    - Java does not permit this

- JavaScript allows anonymous functions

```
var request = getRequestObject();
request.onreadystatechange =
  function() { code-that-uses-request-variable };
```

  - Java has anonymous classes, but no anonymous functions
  - C and C++ have nothing like anonymous functions.
  - Anonymous functions (also called closures) are widely used in Lisp, Ruby, Scheme, C# (as of 2.0), Python, Visual Basic, ML, PHP (as of 5.3), Clojure, Go, & others.

First-Class Functions: Examples

```
function square(x) { return(x * x); }
function triple(x) { return(x * 3); }
function doOperation(f, x) { return(f(x)); }

doOperation(square, 5); → 25

doOperation(triple, 10); → 30

var functions = [square, triple];

functions[0](10);  → 100

functions[1](20);  → 60
```

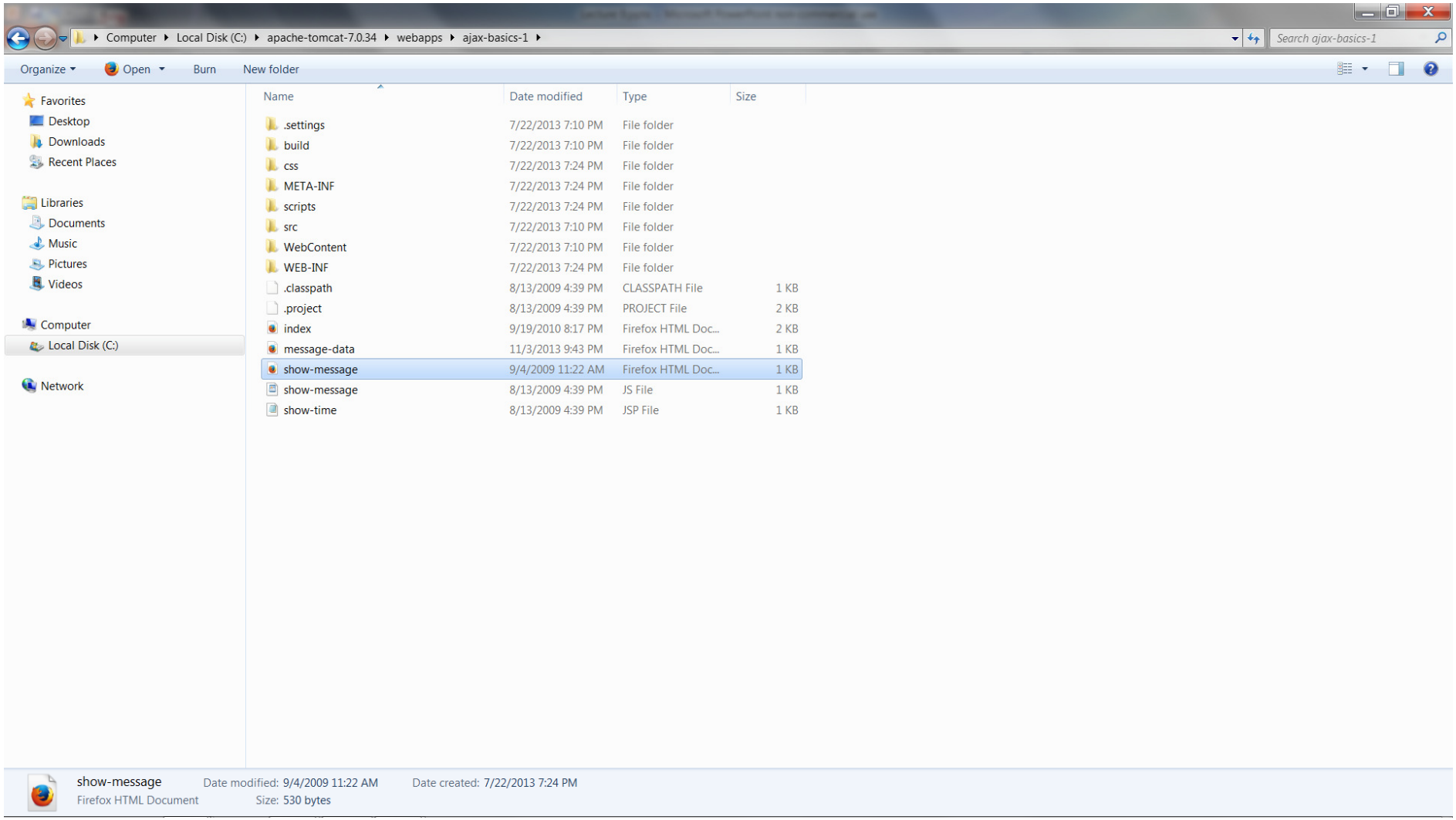
Anonymous Functions: Examples

```
function square(x) { return(x * x); }

square(10);  → 100
(function(x) { return(x * x); })(10);  → 100

function makeMultiplier(n) {
  return(function(x) { return(x * n); });
}

var factor = 5;
var f = makeMultiplier(factor);

f(3);  → 15
factor = 500;
f(3);  → 15
```

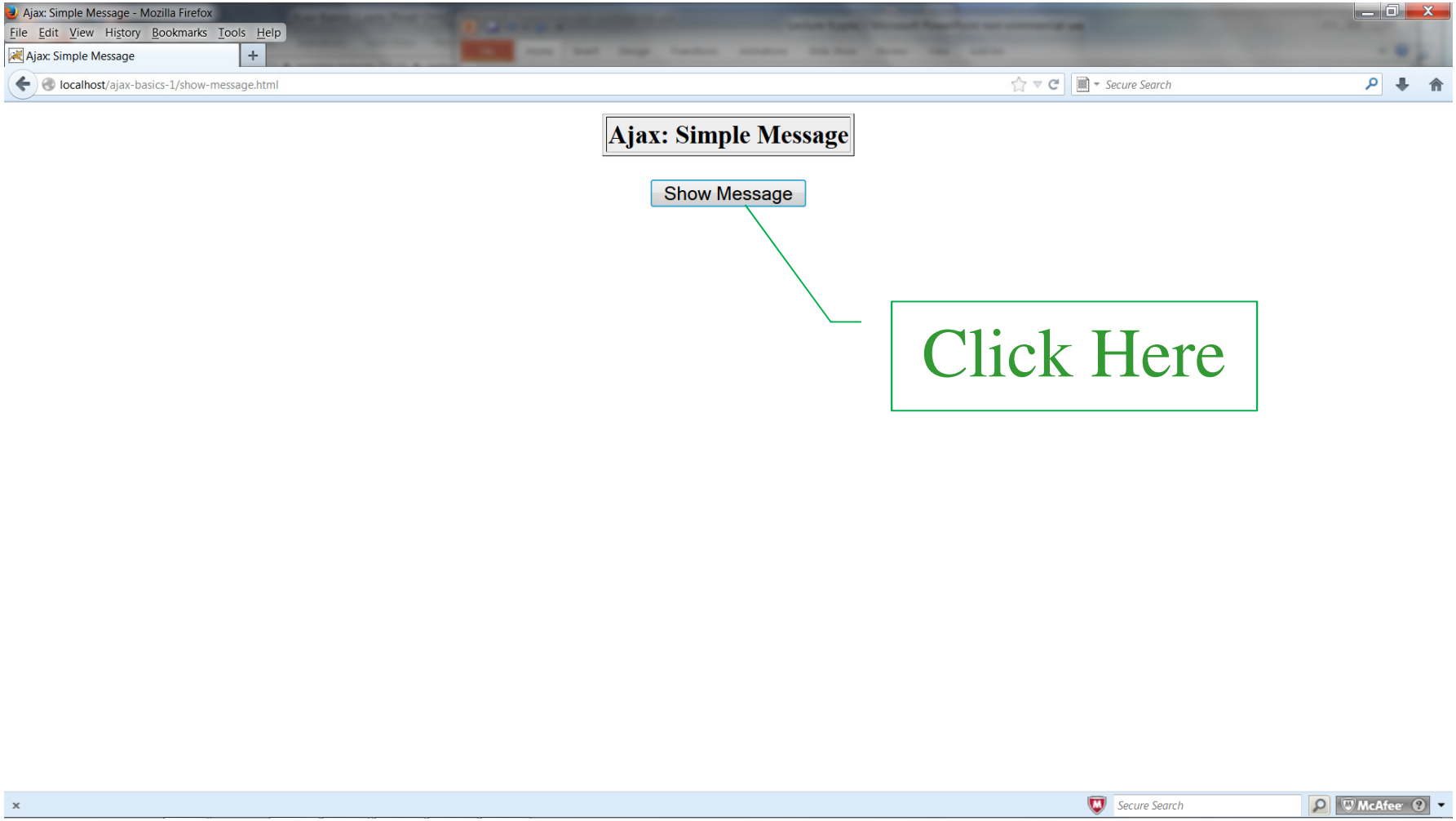
Local Request Variable

```
function getRequestObject() { ... }

function sendRequest() {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { handleResponse(request); };
  request.open("GET", "...", true);
  request.send(null);
}

function handleResponse(request) {
  ...
}
```

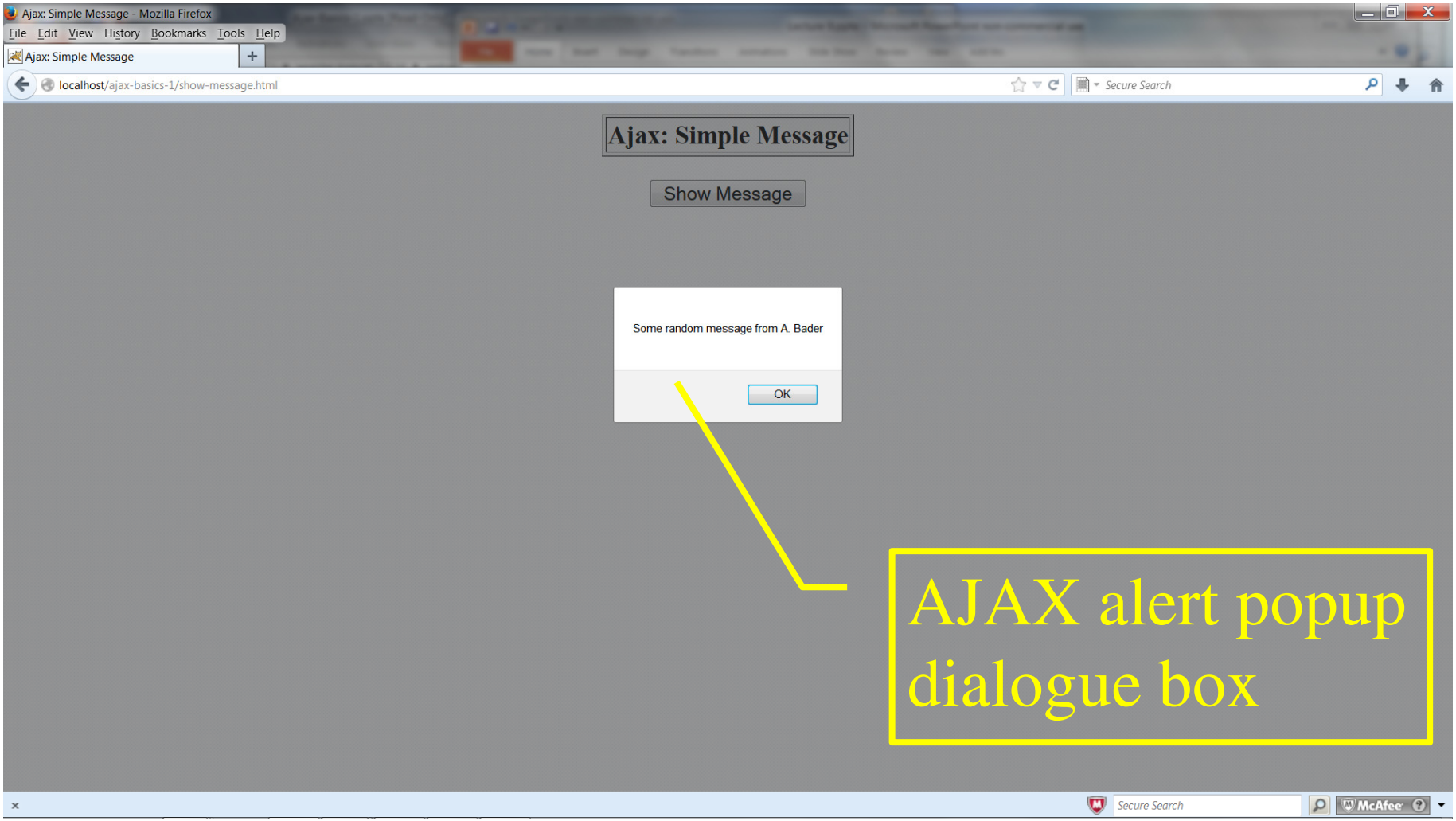
Our First Simple AJAX WebApp



Our First Simple AJAX WebApp

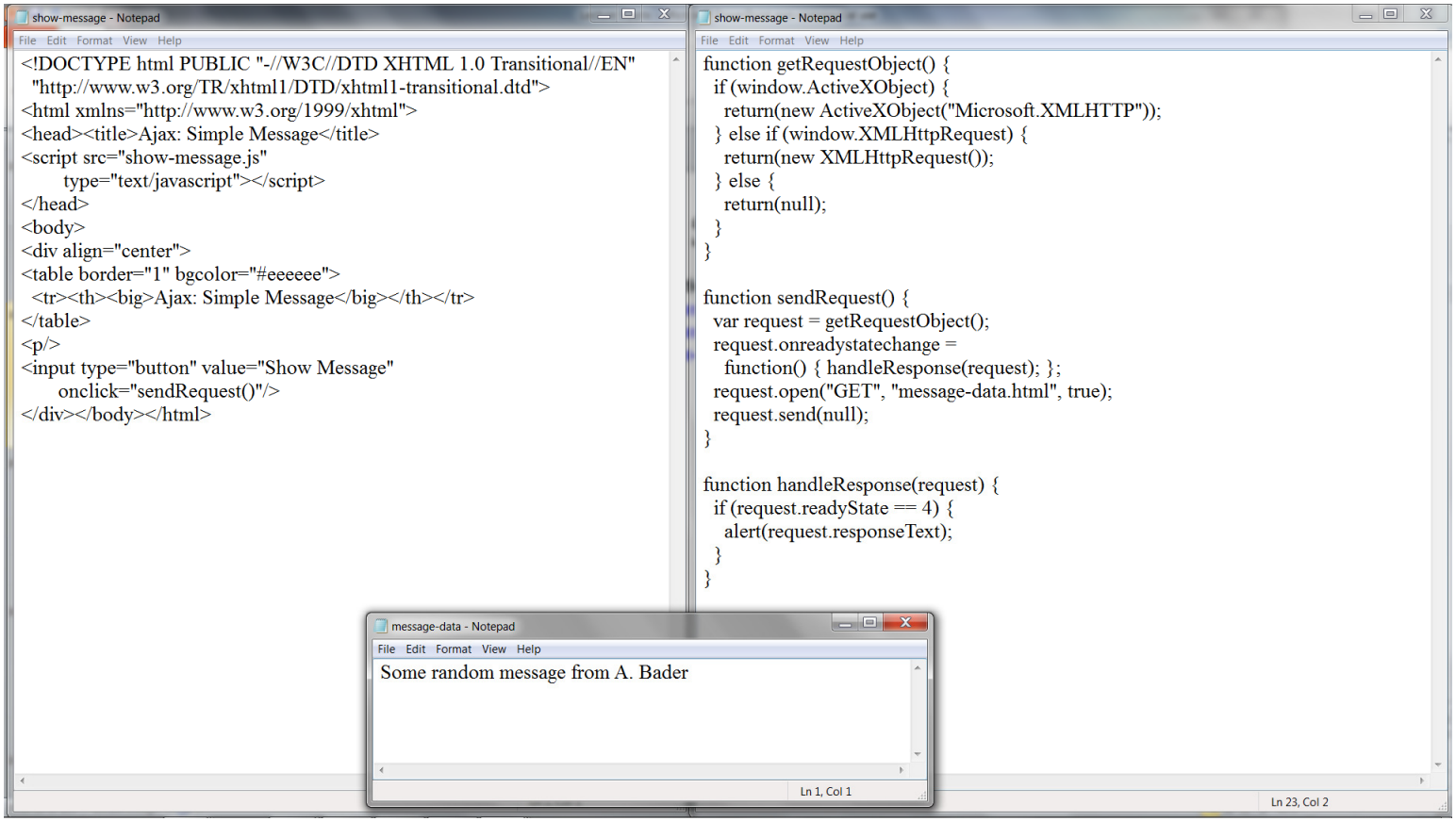


Our First Simple AJAX WebApp



Our First Simple AJAX WebApp

- **For this application, 3 files:**
  1. **show-message.html**
  2. **show-message.js**
  3. **message-data.html**



Complete JavaScript Code (show-message.js)

```
function getRequestObject() {
  if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else if (window.ActiveXObject) {
    return(new ActiveXObject("Microsoft.XMLHTTP"));
  } else {
    return(null);
  }
}

function sendRequest() {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { handleResponse(request); };
  request.open("GET", "message-data.html", true);
  request.send(null);
}

function handleResponse(request) {
  if (request.readyState == 4) {
    alert(request.responseText);
  }
}
```

HTML Code

- Use xhtml, not HTML 4
  - In order to manipulate it with DOM

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">...</html>
```

    - Due to IE bug, do not use XML header before the DOCTYPE
- Load the JavaScript file

```
<script src="relative-url-of-JavaScript-file"
        type="text/javascript"></script>
```

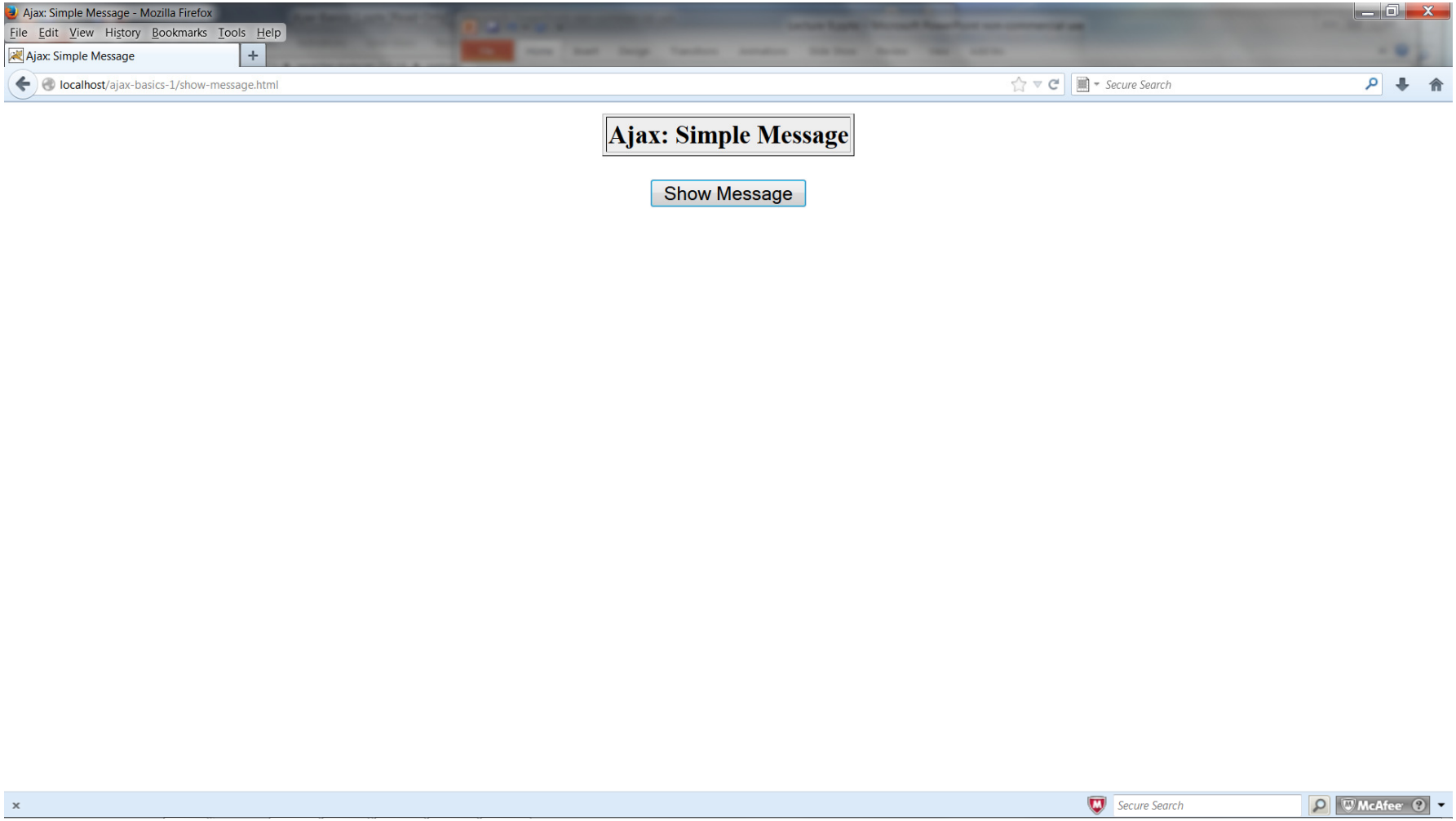
    - Use separate </script> end tag
- Designate control to initiate request

```
<input type="button" value="button label"
       onclick="mainFunction()"/>
```

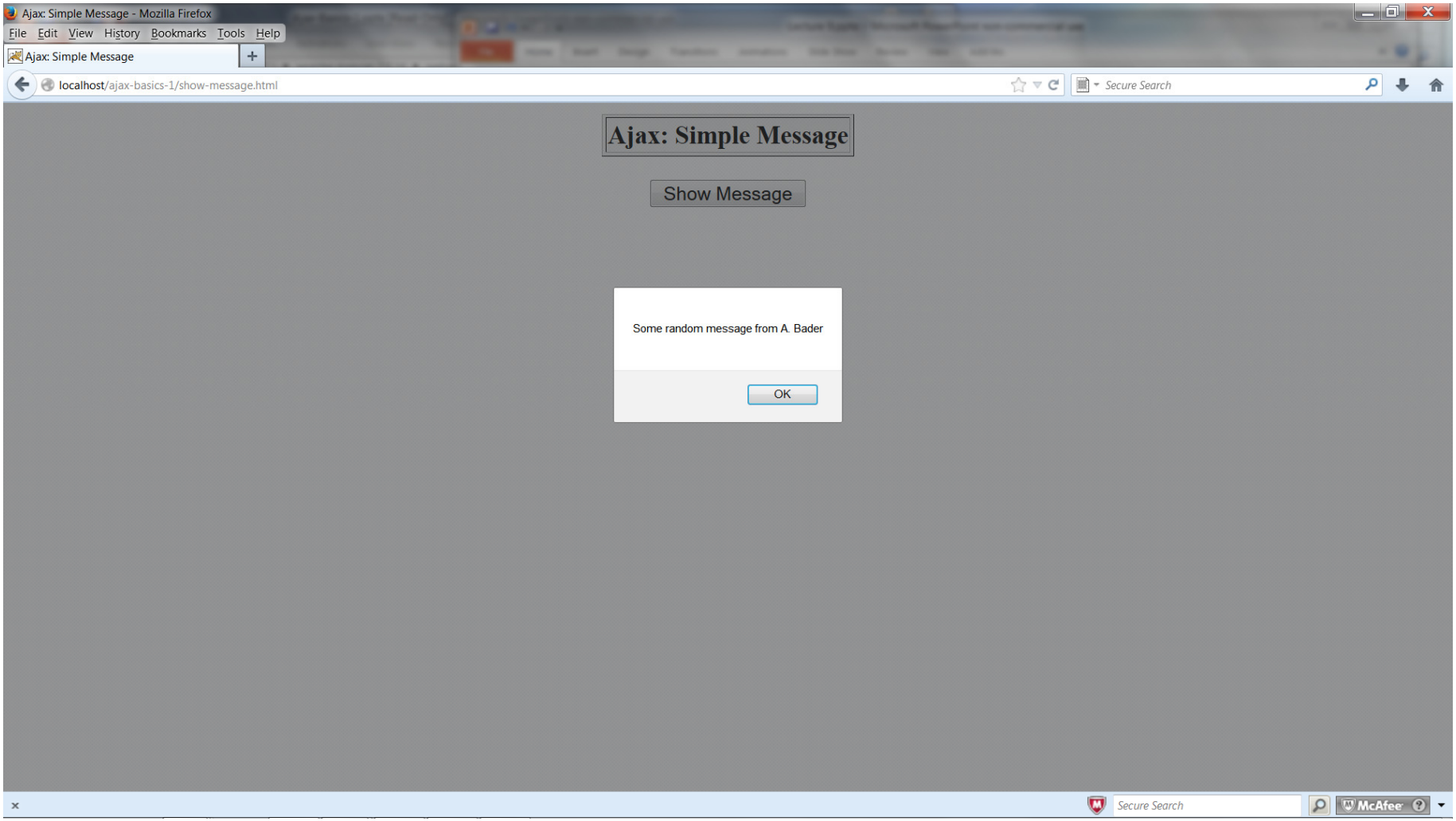
HTML Code (show-message.html)

```
<!DOCTYPE html PUBLIC "..."
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Ajax: Simple Message</title>
<script src="show-message.js"
        type="text/javascript"></script>
</head>
<body>
<center>
<table border="1" bgcolor="gray">
  <tr><th><big>Ajax: Simple Message</big></th></tr>
</table>
<p/>
<input type="button" value="Show Message"
       onclick="sendRequest()"/>
</center></body></html>
```

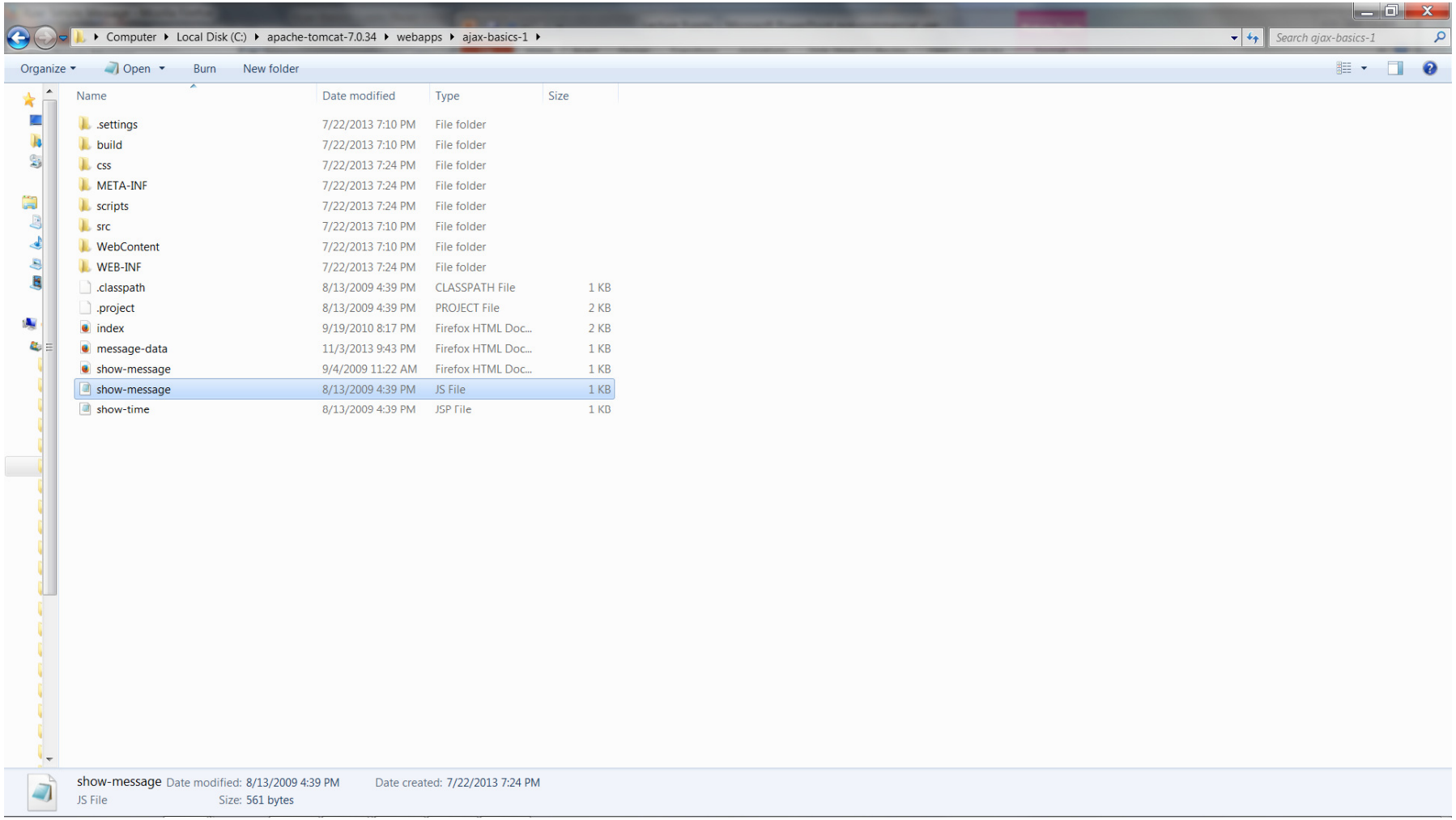
The Basic Process: Results



The Basic Process: Results



WebApp C:\apache-tomcat-7.0.34\webapps\ajax-basics-1



Dynamic Content from JSP

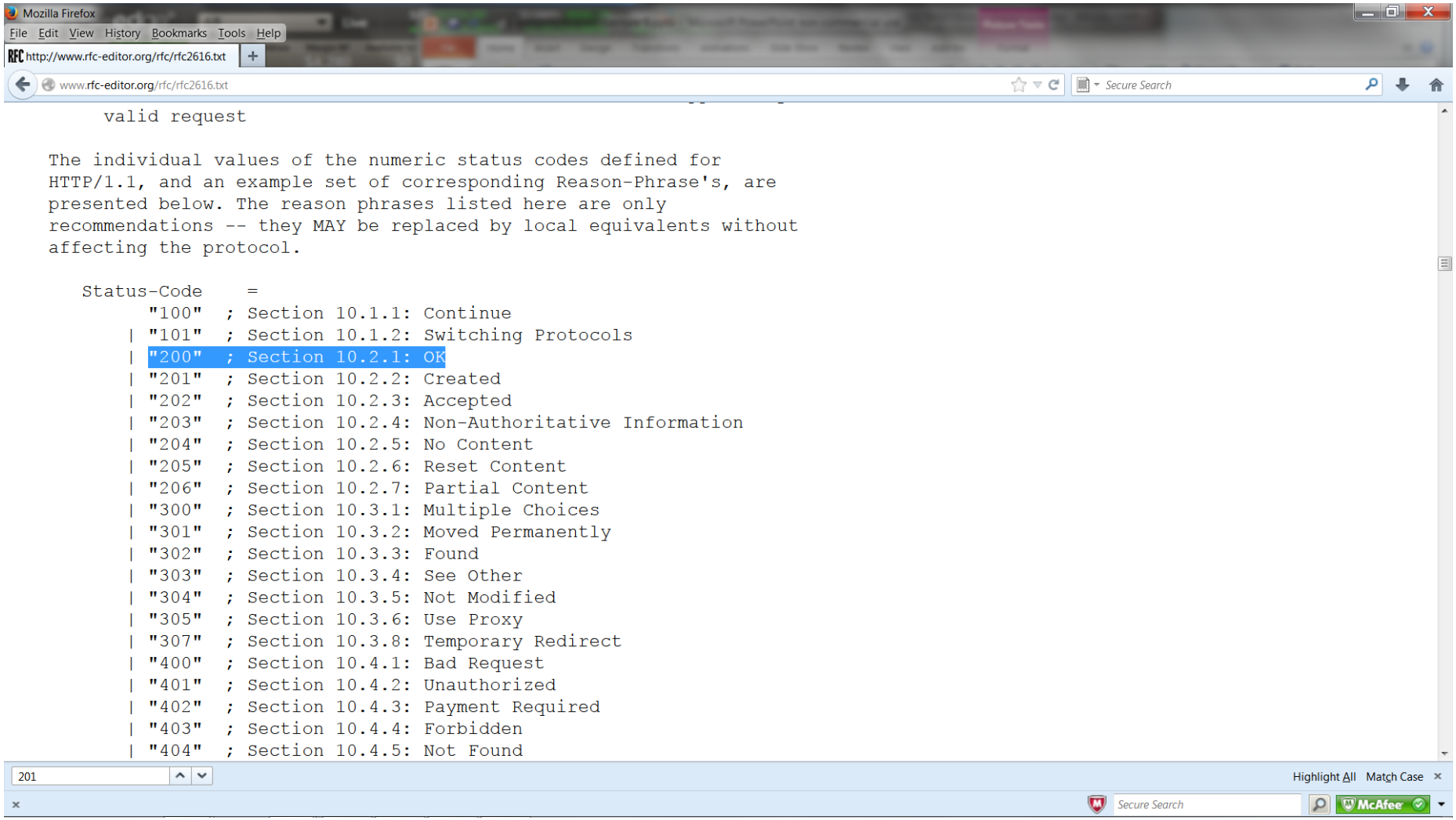
First Example: Design Deficiencies

- Content was the same on each request
  - Could have just hardcoded the alert value in JavaScript
  - Instead, invoke a JSP page on the server
- Resource address hardcoded in JavaScript
  - Prevents functions from applying to multiple situations
  - Instead, make generic function and pass address to it
- JavaScript file was in same folder as HTML
  - Makes it hard to reuse the JavaScript in different pages
  - Instead, make a special directory for JavaScript
- No style sheet was used
  - Less for JavaScript to work with when manipulating page
  - Use CSS for normal reasons as well as for JavaScript

Steps

- JavaScript
  - Define an object for sending HTTP requests
  - Initiate request
    - Get request object
    - Designate an anonymous response handler function
      - Supply as onreadystatechange attribute of request
    - Initiate a GET or POST request to a JSP page
      - Get the address from a variable instead of hardcoding it
    - Send data
  - Handle response
    - Wait for readyState of 4 and HTTP status of 200
    - Extract return text with responseText or responseXML
    - Do something with result
- HTML
  - Load JavaScript from centralized directory. Use style sheet.
  - Designate control that initiates request
  - Give id to output placeholder region

HTTP/1.1 Status-Code



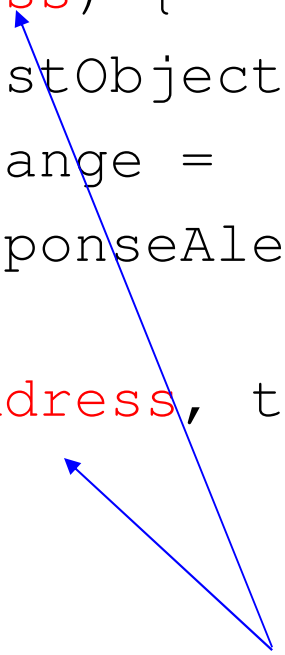
Define a Request Object

```
function getRequestObject() {
  if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else if (window.ActiveXObject) {
    return(new
  ActiveXObject("Microsoft.XMLHTTP"));
  } else {
    return(null);
  }
}
```

- No changes from previous example.
- This code stays the same for entire section.

Initiate Request

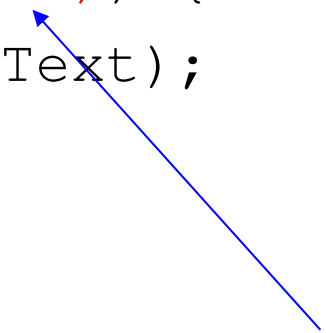
```
function ajaxAlert(address) {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { showResponseAlert(request);
  };
  request.open("GET", address, true);
  request.send(null);
}
```



- Relative URL of server-side resource.
(In this example, we will pass in the address of a JSP page.)

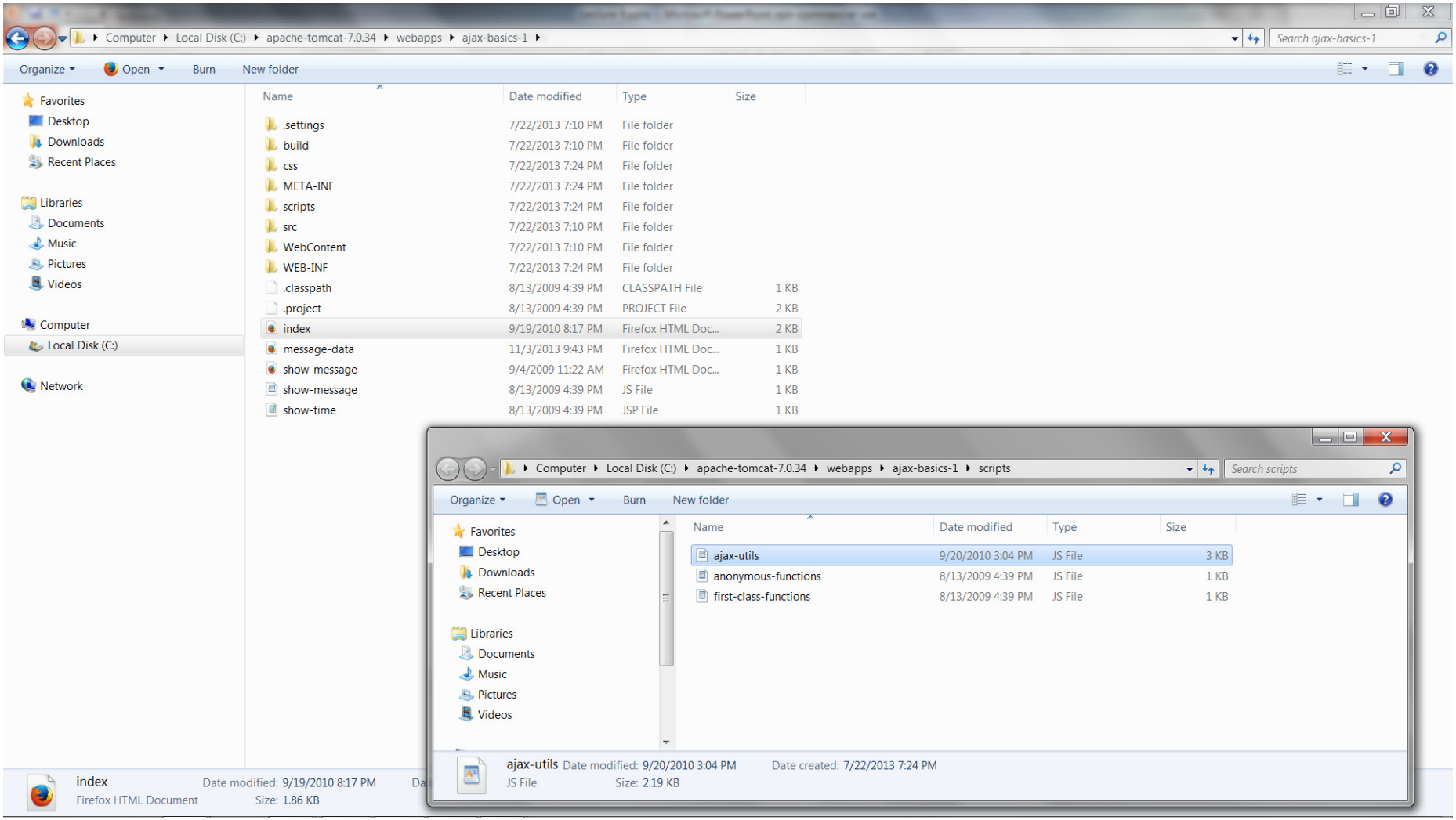
Handle Response

```
function showResponseAlert(request) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    alert(request.responseText);
  }
}
```



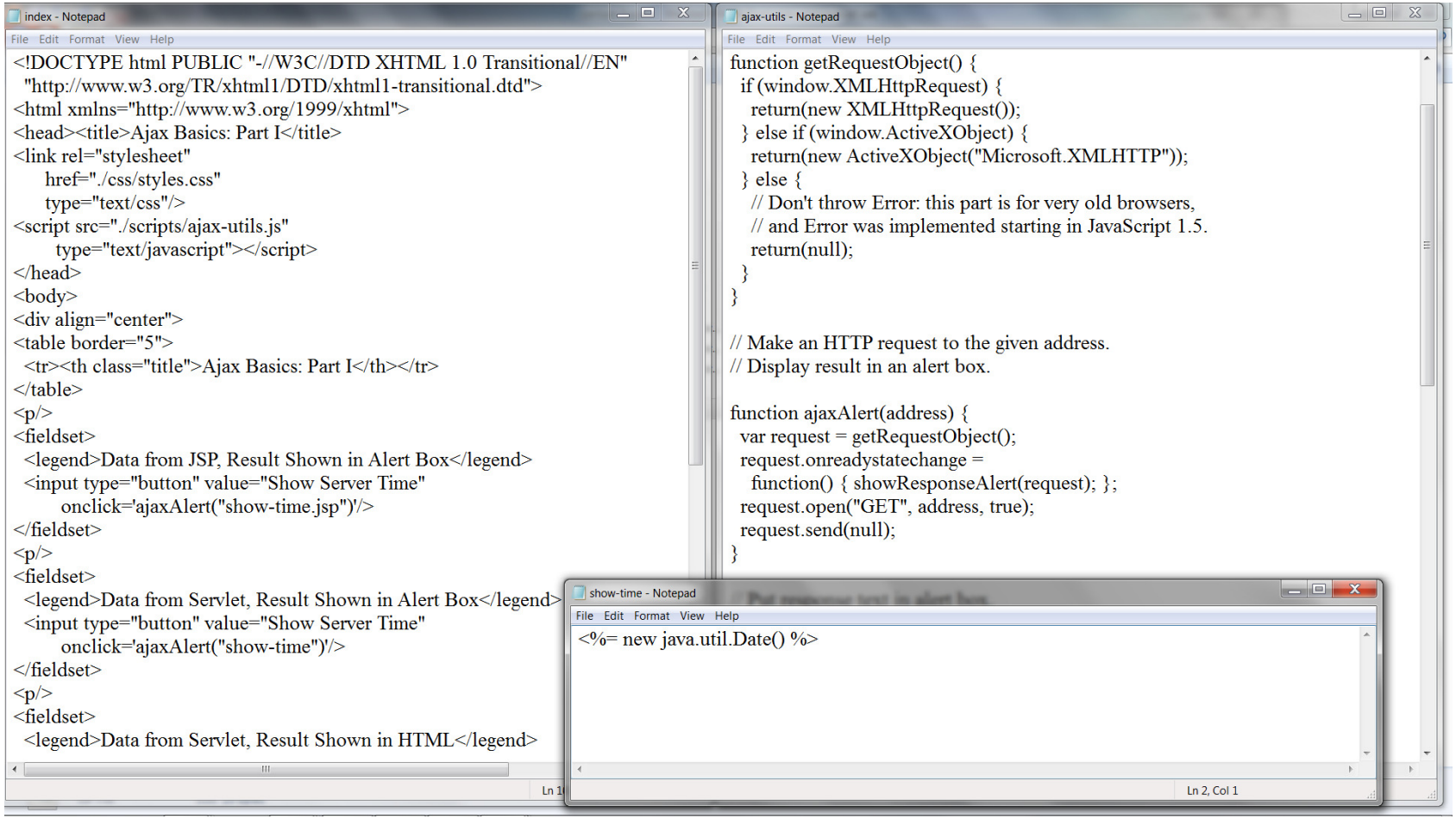
- Server response came back with no errors (HTTP status code 200).

Source Code Directories



Source Code Directories

- **For this application, 3 files:**
  1. **index.html**
  2. **ajax-utils.js**
  3. **show-time.jsp**



index - Notepad

File Edit Format View Help

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Ajax Basics: Part I</title>
<link rel="stylesheet"
      href="./css/styles.css"
      type="text/css"/>
<script src="./scripts/ajax-utils.js"
        type="text/javascript"></script>
</head>
<body>
<div align="center">
<table border="5">
  <tr><th class="title">Ajax Basics: Part I</th></tr>
</table>
<p/>
<fieldset>
  <legend>Data from JSP, Result Shown in Alert Box</legend>
  <input type="button" value="Show Server Time"
         onclick='ajaxAlert("show-time.jsp")'/>
</fieldset>
<p/>
<fieldset>
  <legend>Data from Servlet, Result Shown in Alert Box</legend>
  <input type="button" value="Show Server Time"
         onclick='ajaxAlert("show-time")'/>
</fieldset>
<p/>
<fieldset>
  <legend>Data from Servlet, Result Shown in HTML</legend>
```

ajax-utils - Notepad

File Edit Format View Help

```
function getRequestObject() {
  if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else if (window.ActiveXObject) {
    return(new ActiveXObject("Microsoft.XMLHTTP"));
  } else {
    // Don't throw Error: this part is for very old browsers,
    // and Error was implemented starting in JavaScript 1.5.
    return(null);
  }
}

// Make an HTTP request to the given address.
// Display result in an alert box.

function ajaxAlert(address) {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { showResponseAlert(request); };
  request.open("GET", address, true);
  request.send(null);
}
```

show-time - Notepad

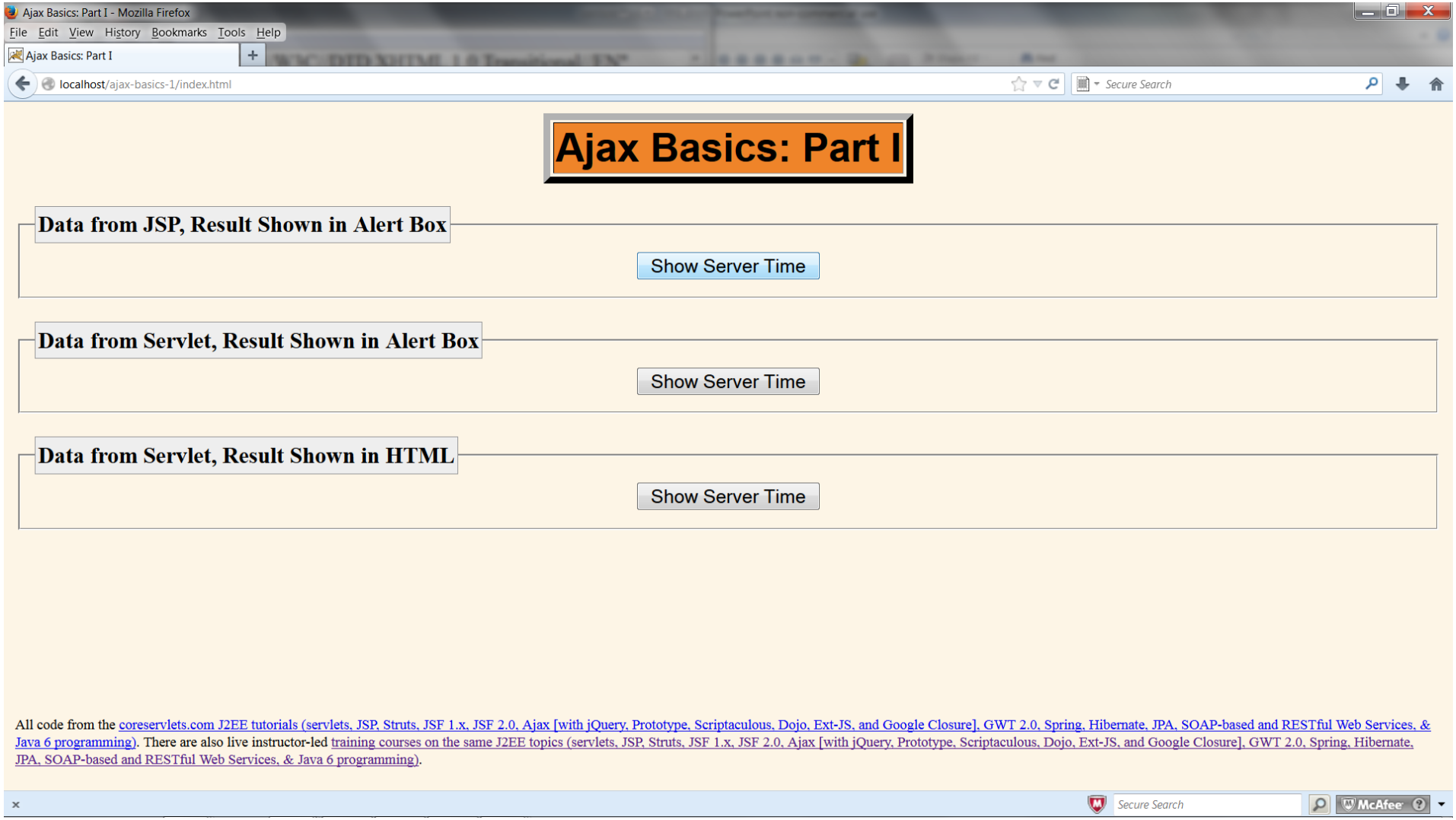
File Edit Format View Help

```
<%= new java.util.Date() %>
```

Ln 2, Col 1

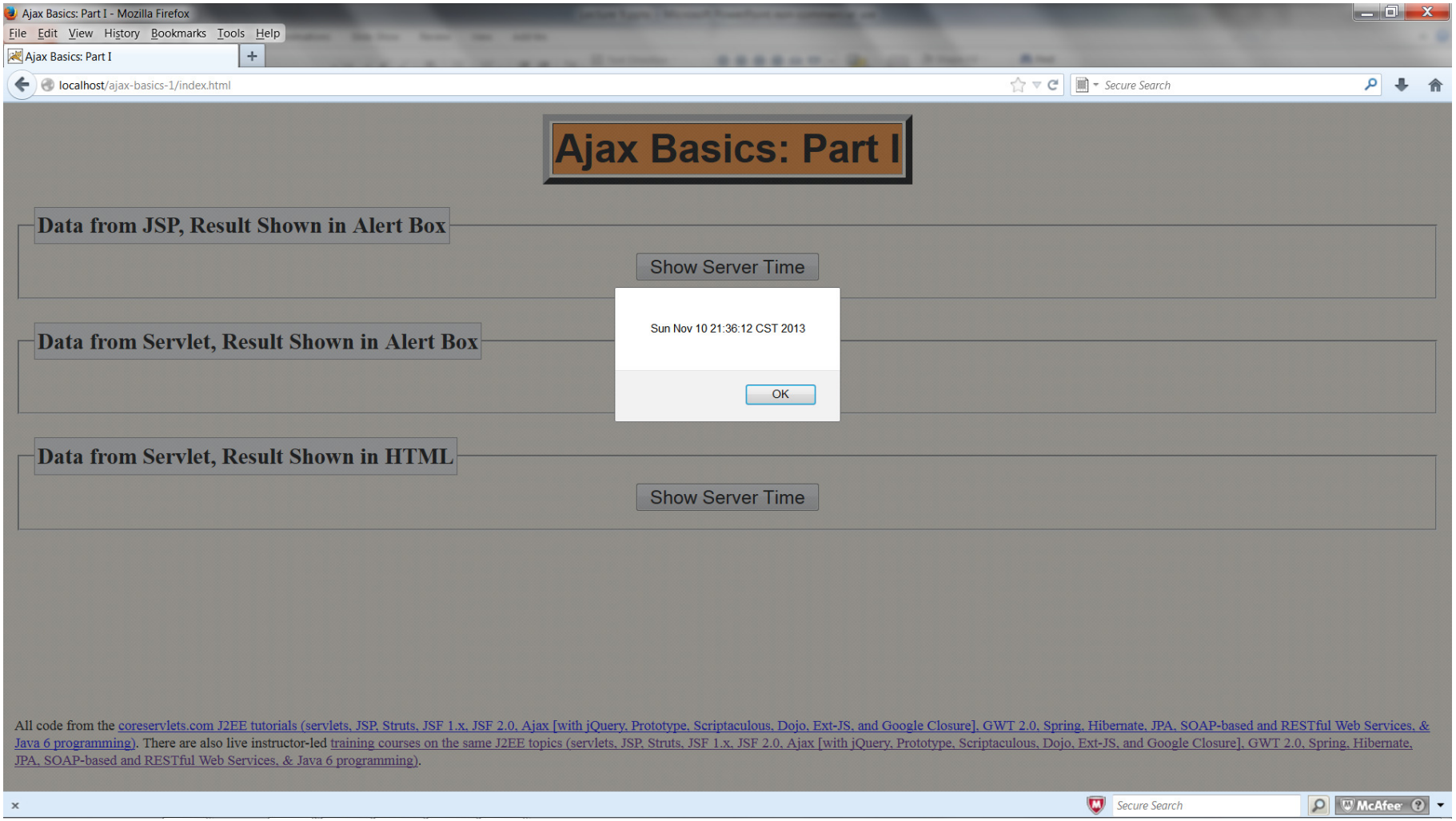
Source Code Directories

- **Sample Output**



Source Code Directories

- **Sample Output**



Complete JavaScript Code (Part of ajax-utils.js)

```
function getRequestObject() {
  if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else if (window.ActiveXObject) {
    return(new ActiveXObject("Microsoft.XMLHTTP"));
  } else {
    return(null);
  }
}

function ajaxAlert(address) {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { showResponseAlert(request); }
  request.open("GET", address, true);
  request.send(null);
}

function showResponseAlert(request) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    alert(request.responseText);
  }
}
```

HTML Code index.html

- Load JavaScript from central location

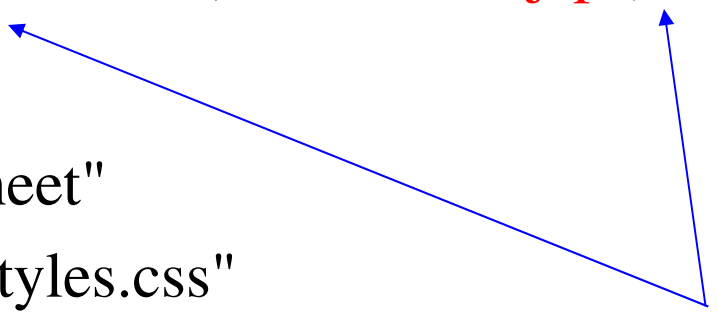
```
<script src="./scripts/ajax-utils.js"
        type="text/javascript"></script>
```

- Pass JSP address to main function

```
<input type="button" value="Show Server Time"
       onclick='ajaxAlert("show-time.jsp")'/>
```

- Use style sheet

```
<link rel="stylesheet"
      href="./css/styles.css"
      type="text/css"/>
```



➢Note single quotes (because of double quotes inside parens).

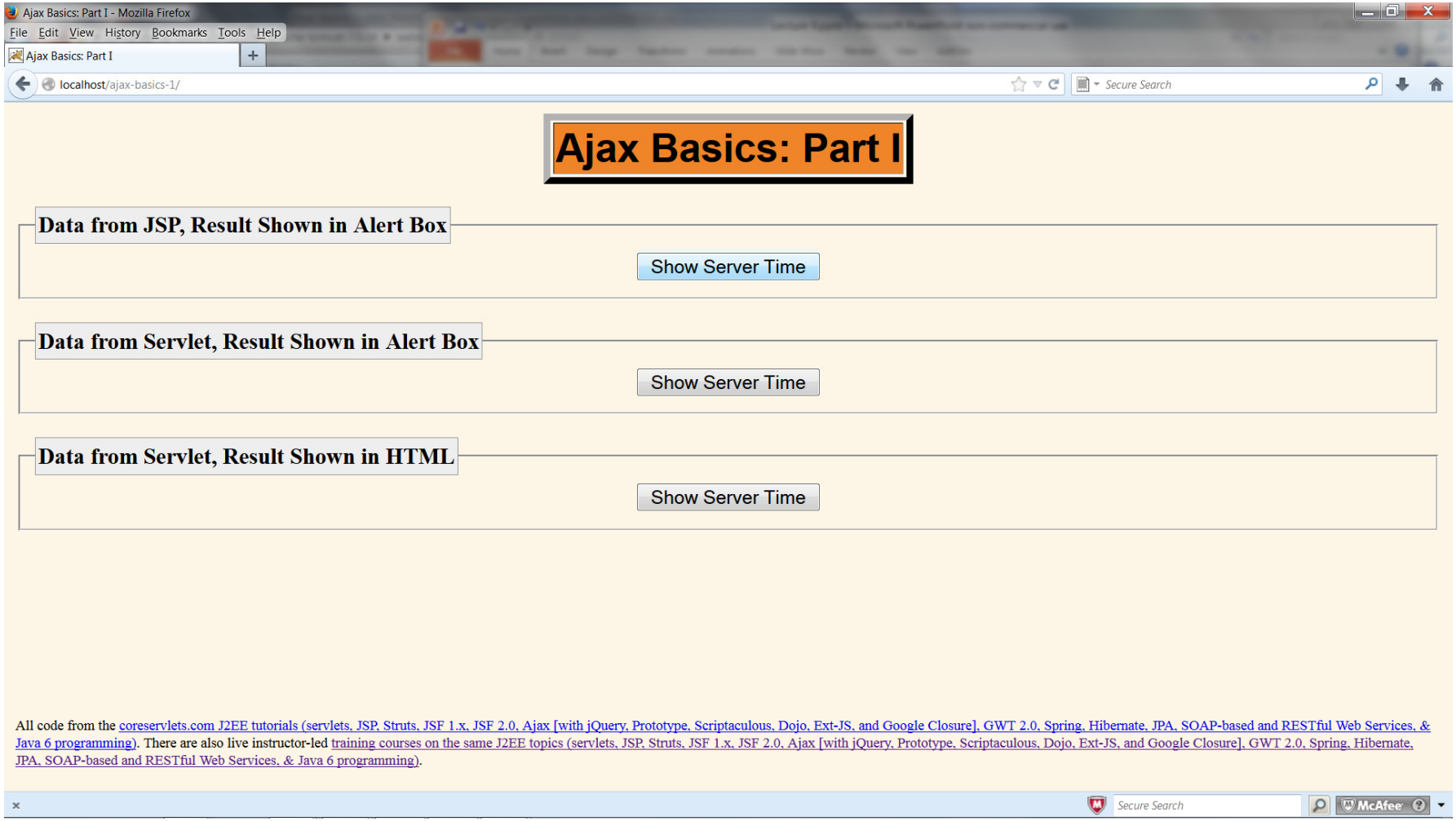
HTML Code

```
<!DOCTYPE html PUBLIC "...">
<html xmlns="http://www.w3.org/1999/xhtml">...
<link rel="stylesheet"
      href="./css/styles.css"
      type="text/css"/>
<script src="./scripts/ajax-utils.js"
        type="text/javascript"></script>...
<body>...
<fieldset>
  <legend>Data from JSP, Result Shown in Alert Box </legend>
  <form action="#">
    <input type="button" value="Show Server Time"
           onclick='ajaxAlert("show-time.jsp")'/>
  </form>
</fieldset>
...
```

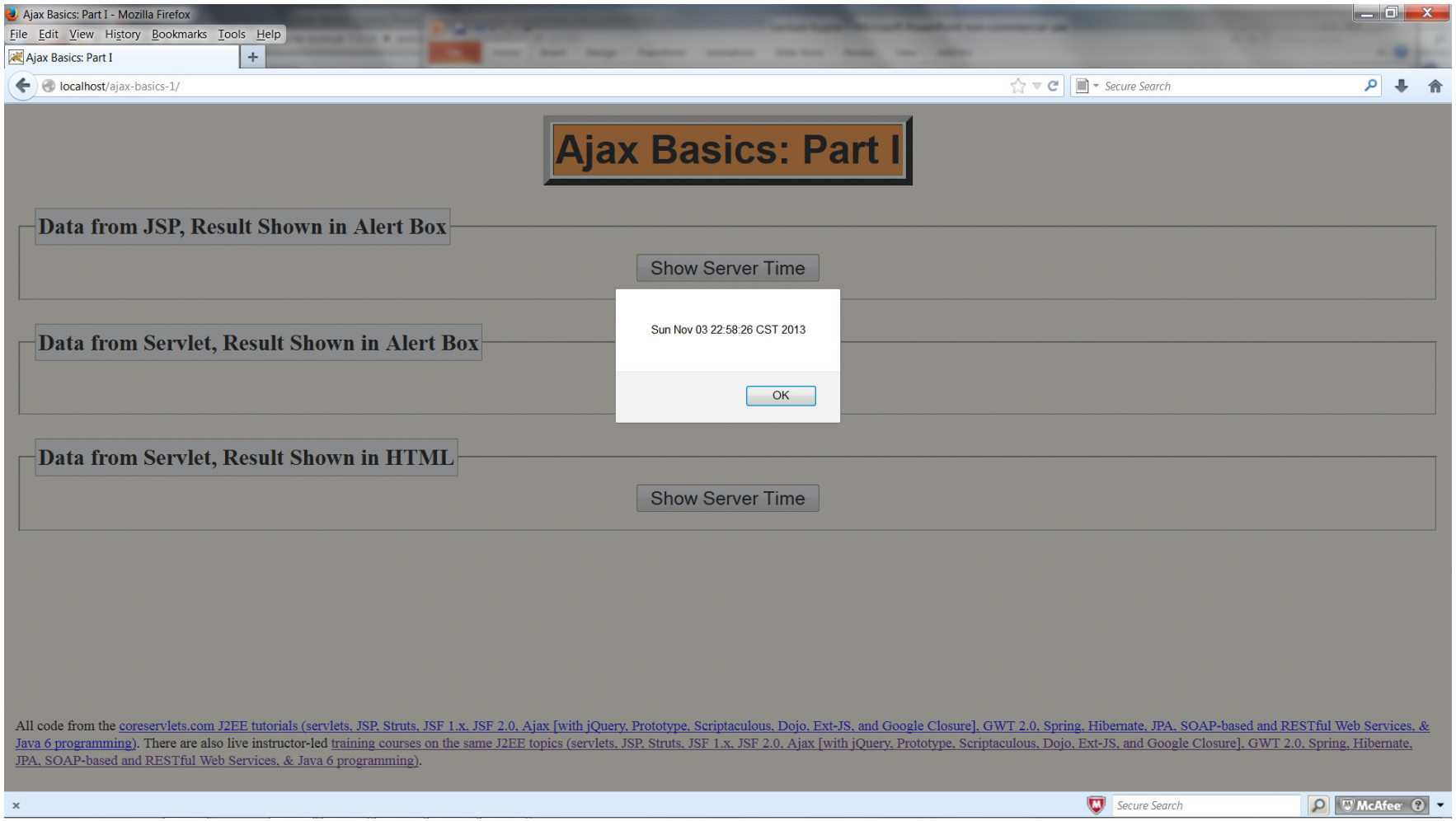
JSP Code (show-time.jsp)

```
<%= new java.util.Date() %>
```

Message from JSP: Results



Message from JSP: Results



Dynamic Content from Servlet

Steps

- JavaScript
  - Define an object for sending HTTP requests
  - Initiate request
    - Get request object
    - Designate an anonymous response handler function
      - Supply as onreadystatechange attribute of request
    - Initiate a GET or POST request to a servlet
    - Send data
  - Handle response
    - Wait for readyState of 4 and HTTP status of 200
    - Extract return text with responseText or responseXML
    - Do something with result
- HTML
  - Load JavaScript from centralized directory. Use style sheet.
  - Designate control that initiates request
  - Give id to output placeholder region

Define a Request Object, Initiate Request, Handle Response

```
function getRequestObject() {
  if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else if (window.ActiveXObject) {
    return(new ActiveXObject("Microsoft.XMLHTTP"));
  } else {
    return(null);
  }
}

function ajaxAlert(address) {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { showResponseAlert(request); }
  request.open("GET", address, true);
  request.send(null);
}

function showResponseAlert(request) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    alert(request.responseText);
  }
}
```

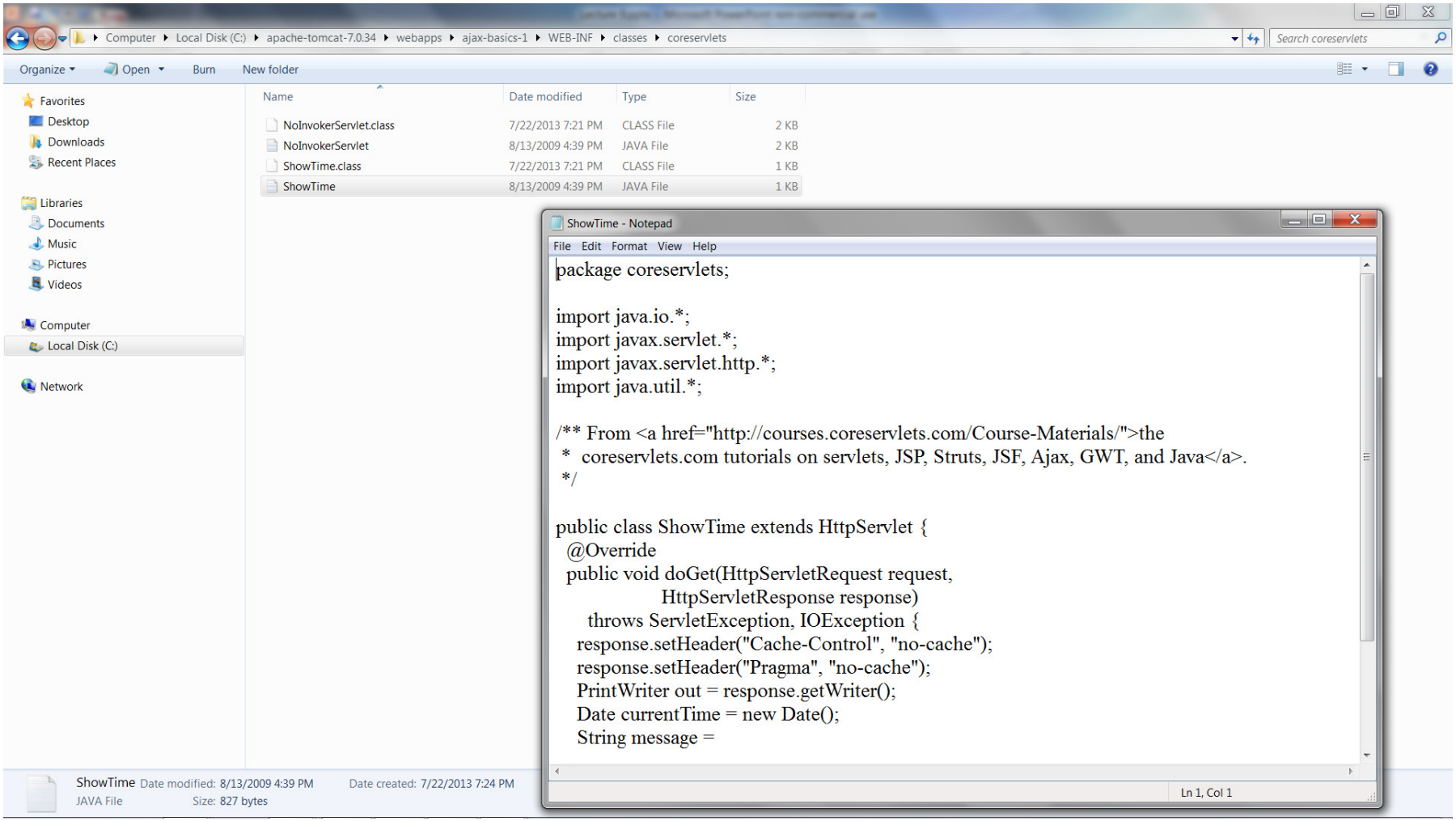
- No changes from previous example. Only address changes, and address comes from the HTML page.

HTML Code

```
...
<link rel="stylesheet"
      href="./css/styles.css"
      type="text/css"/>
<script src="./scripts/ajax-utils.js"
        type="text/javascript"></script>
...
<fieldset>
  <legend>
     Data from Servlet, Result Shown in Alert Box
  </legend>
  <input type="button" value="Show Server Time"
         onclick='ajaxAlert("show-time")'/>
</fieldset>
...
```

- Address of servlet from @WebServlet (servlets 3.0) or from url-pattern of servlet-mapping in web.xml (servlets 2.5 and earlier).

Location of the Servlet Code



Servlet Code

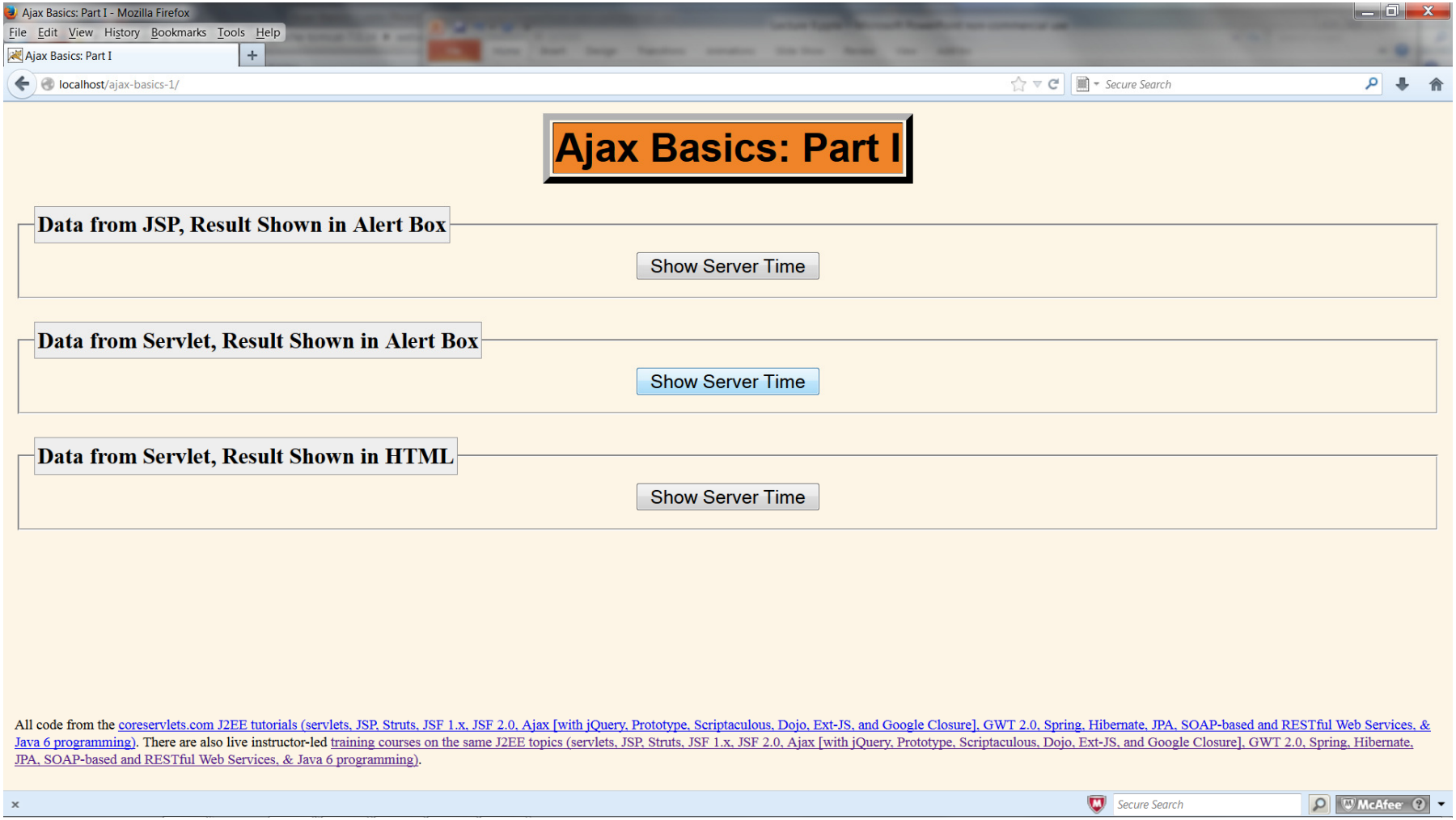
```
package coreservlets;
import ...

public class ShowTime extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Pragma", "no-cache");
    PrintWriter out = response.getWriter();
    Date currentTime = new Date();
    String message =
      String.format("It is now %tr on %tD.",
                    currentTime, currentTime);
    out.print(message);
  }
}
```

web.xml

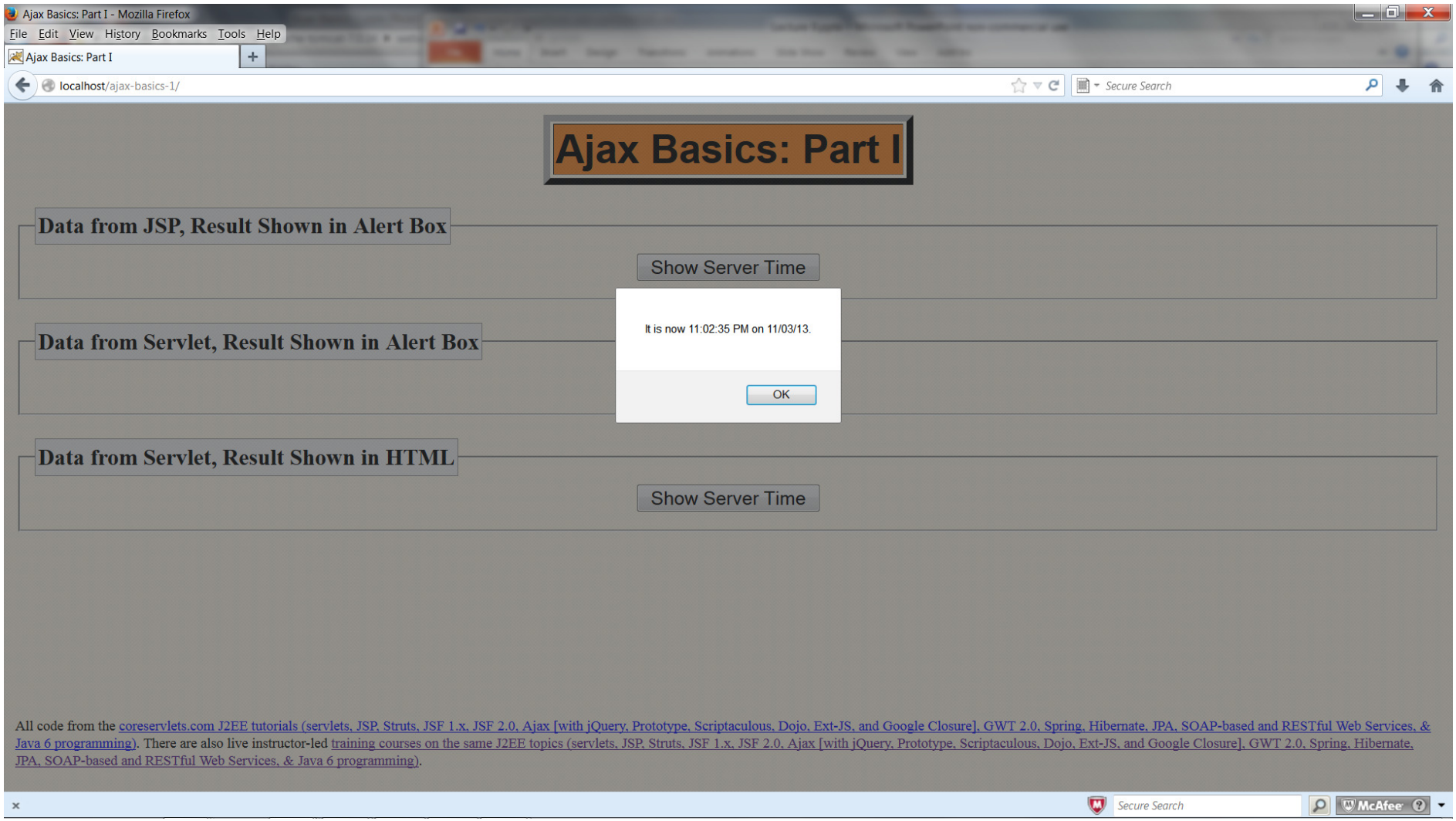
```
...
  <servlet>
    <servlet-name>ShowTime</servlet-name>
    <servlet-class>coreservlets.ShowTime</servlet-
   class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ShowTime</servlet-name>
    <url-pattern>/show-time</url-pattern>
  </servlet-mapping>
...
```

- In Tomcat 7 or other servers that support servlets 3.0, you can use @WebServlet("/show-time") above the servlet class definition, and omit web.xml entirely.

Message from Servlet: Results



Message from Servlet: Results

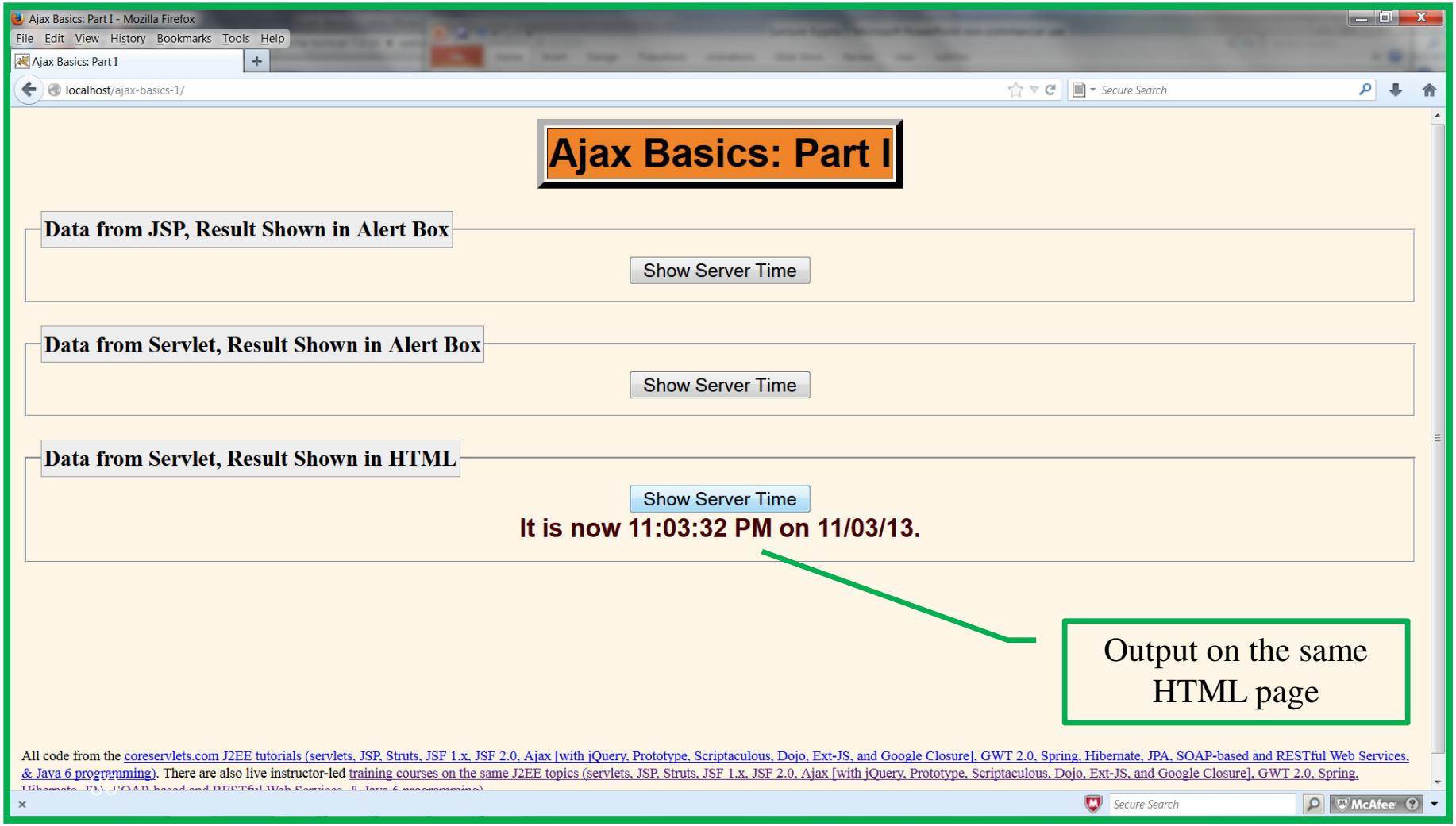


Building HTML Output

ShowTime Servlet Example: Design Deficiencies

- Results always shown in dialog (alert) box
  - Alerts usually reserved for errors or warnings
  - Users prefer normal results inside page
  - Solution: use Dynamic HTML to update page with result
    - HTML plus CSS styles represented in the DOM
      - DOM stands for "Document Object Model", an XML view of page
    - JavaScript can insert elements into the DOM
      - Find an element with given id
        - » someElement = document.getElementById(id);
      - Insert HTML inside
        - » someElement.innerHTML = "<h1>blah</h1>";
    - JavaScript can also read the DOM
      - E.g., look up textfield values (see upcoming example)
        - » document.getElementById(id).value

We want HTML Output on the same page not in a popup dialog box



Sample Code Snippets for This Approach

index - Notepad

File Edit Format View Help

```
<fieldset>
  <legend>Data from JSP, Result Shown in Alert Box</legend>
  <input type="button" value="Show Server Time"
         onclick='ajaxAlert("show-time.jsp")'/>
</fieldset>
<p/>
<fieldset>
  <legend>Data from Servlet, Result Shown in Alert Box</legend>
  <input type="button" value="Show Server Time"
         onclick='ajaxAlert("show-time")'/>
</fieldset>
<p/>
<fieldset>
  <legend>Data from Servlet, Result Shown in HTML</legend>
  <input type="button" value="Show Server Time"
         onclick='ajaxResult("show-time", "timeResult1")'/>
  <div id="timeResult1" class="ajaxResult"></div>
</fieldset>
<p/>
</div>



<br/><br/><br/><br/><br/><br/>
<font size="-3">All code from the
<a href="http://courses.coreservlets.com/Course-Materials/">
coreservlets.com J2EE tutorials (servlets, JSP, Struts, JSF 1.x, JSF 2.0, Ajax [
Scriptaculous, Dojo, Ext-JS, and Google Closure], GWT 2.0, Spring, Hiberna
SOAP-based and RESTful Web Services, &amp;amp; Java 6 programming)</a>. T
<a href="http://courses.coreservlets.com/">training courses on
the same J2EE topics (servlets, JSP, Struts, JSF 1.x, JSF 2.0, Ajax [with jQue
```

Ln 31, Col 1

ajax-utils - Notepad

File Edit Format View Help

```
// Make an HTTP request to the given address.
// Display result in the HTML element that has given ID.

function ajaxResult(address, resultRegion) {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { showResponseText(request,
                                  resultRegion); };
  request.open("GET", address, true);
  request.send(null);
}

// Put response text in the HTML element that has given ID.

function showResponseText(request, resultRegion) {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    htmlInsert(resultRegion, request.responseText);
  }
}

//Insert the html data into the element that has the specified id.

function htmlInsert(id, htmlData) {
  document.getElementById(id).innerHTML = htmlData;
}

// Trick so that the Firebug console.log function will
```

Ln 40, Col 1

Dynamically Inserting Text

- HTML
  -
- JavaScript
  - resultRegion = document.getElementById("results-placeholder");
  - resultRegion.innerHTML = "<h2>Wow!</h2>";
    - For the innerHTML text, you usually use request.responseText or some string derived from request.responseText
- Result after running code
  - <h2>Wow!</h2>
    - "View source" won't show this.
- Warning
  - Make sure what you insert results in legal xhtml
    - You can't insert block-level elements into inline elements
    - Use correct case for the inserted text

Summary of New Features

- HTML
  - Define initially blank div element

```
<div id="resultText"></div>
```

- JavaScript response handler
  - Supply an id (resultRegion), find element with that id, and insert response text into innerHTML property

```
document.getElementById(resultRegion).innerHTML =
    request.responseText;
```

Steps

- JavaScript
  - Define an object for sending HTTP requests
  - Initiate request
    - Get request object
    - Designate an anonymous response handler function
      - Supply as onreadystatechange attribute of request
    - Initiate a GET or POST request to a servlet
    - Send data
  - Handle response
    - Wait for readyState of 4 and HTTP status of 200
    - Extract return text with responseText or responseXML
    - Use innerHTML to insert result into designated element
- HTML
  - Load JavaScript from centralized directory. Use style sheet.
  - Designate control that initiates request
  - Give id to output placeholder region

Define a Request Object

```
function getRequestObject() {
  if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else if (window.ActiveXObject) {
    return(new
  ActiveXObject("Microsoft.XMLHTTP"));
  } else {
    return(null);
  }
}
```

- No changes from previous examples

Initiate Request

```
function ajaxResult(address, resultRegion) {
  var request = getRequestObject();
  request.onreadystatechange =
    function() { showResponseText(request,
                                  resultRegion); };
  request.open("GET", address, true);
  request.send(null);
}
```

Handle Response

```
function showResponseText(request, resultRegion)
   {
   if ((request.readyState == 4) &&
       (request.status == 200)) {
     htmlInsert(resultRegion,
   request.responseText);
   }
}

function htmlInsert(id, htmlData) {
   document.getElementById(id).innerHTML =
   htmlData;
}
```

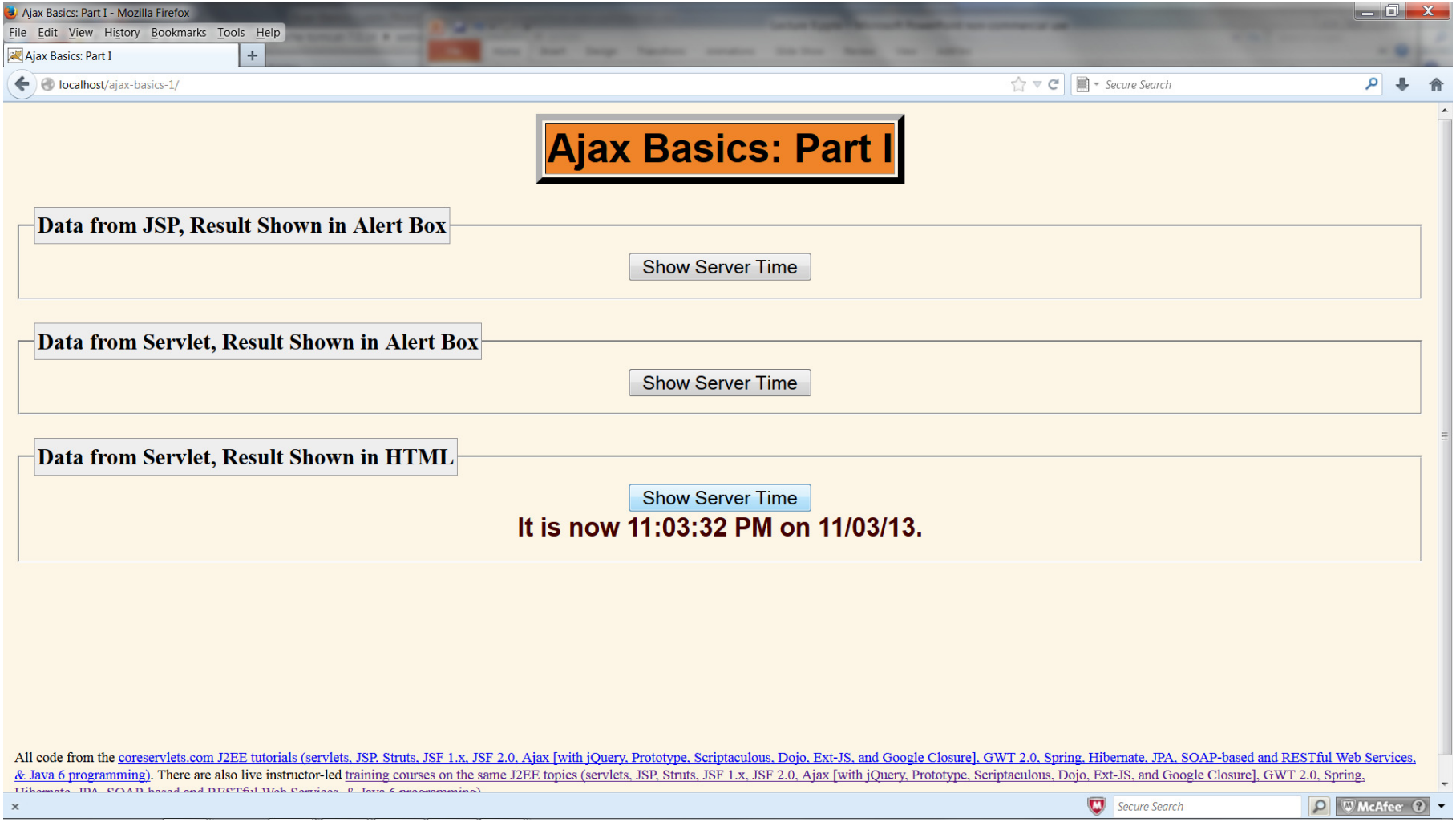
HTML Code

```
...
<link rel="stylesheet"
      href="./css/styles.css"
      type="text/css"/>
<script src="./scripts/ajax-utils.js"
        type="text/javascript"></script>
...
<fieldset>
  <legend>Data from Servlet, Result Shown in HTML</legend>
  <input type="button" value="Show Server Time"
         onclick='ajaxResult("show-time","timeResult1")'/>
  <div id="timeResult1" class="ajaxResult"></div>
</fieldset>
...
```

Servlet Code

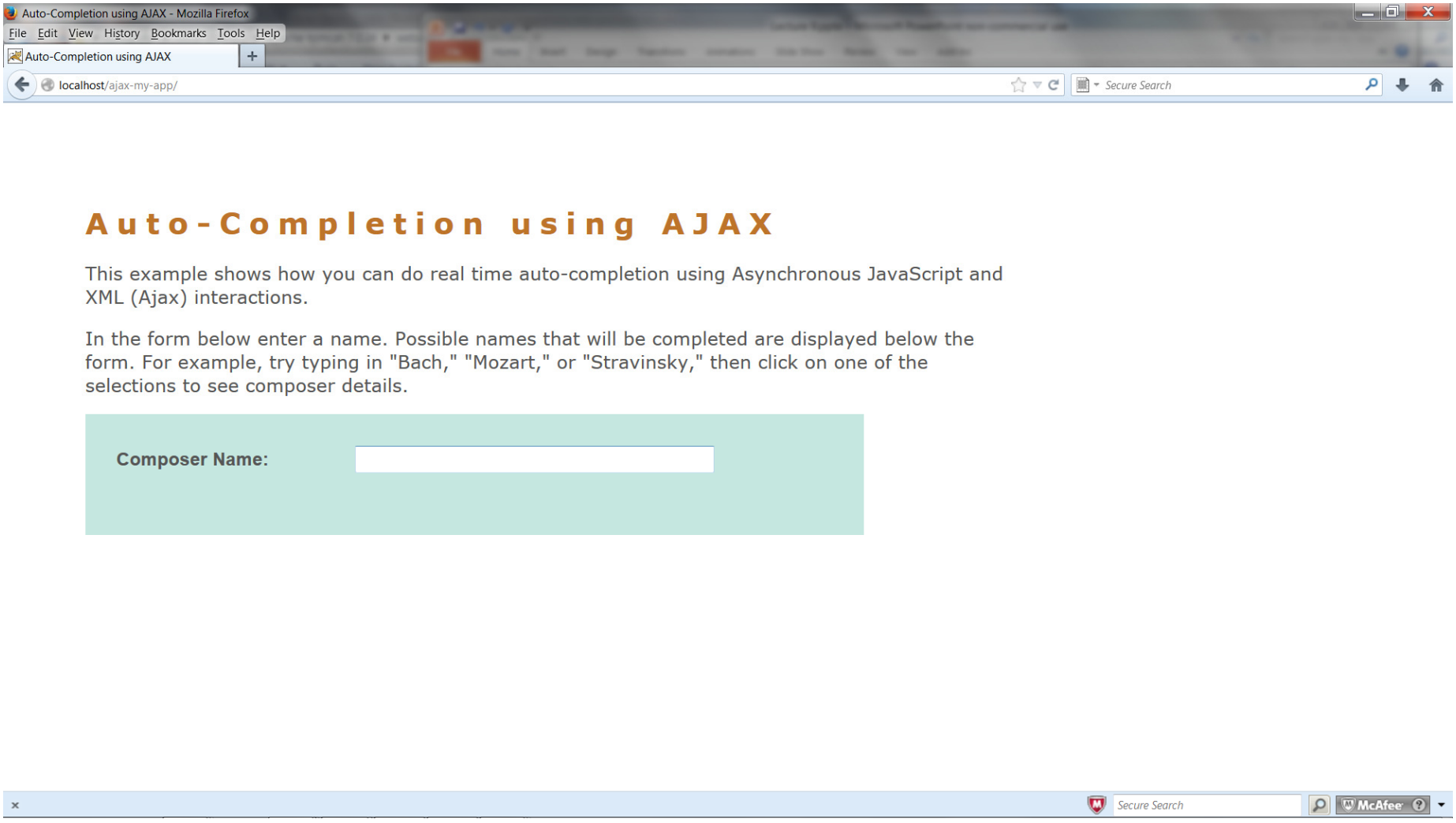
- No changes from previous example
  - Returns a formatted time string

Building HTML Output: Results



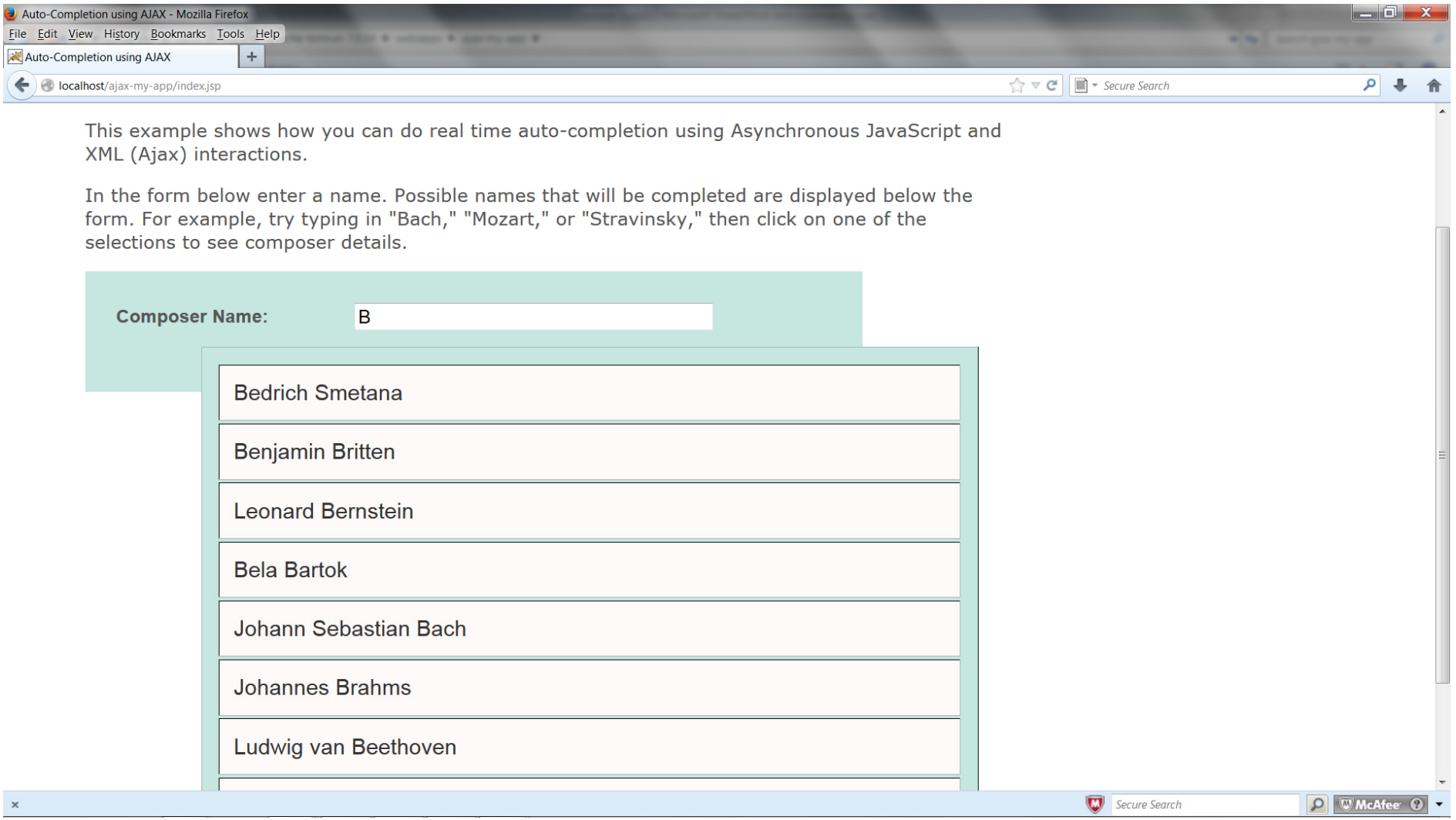
Auto-Completion using AJAX

- First you get this …. Default file name is index.jsp



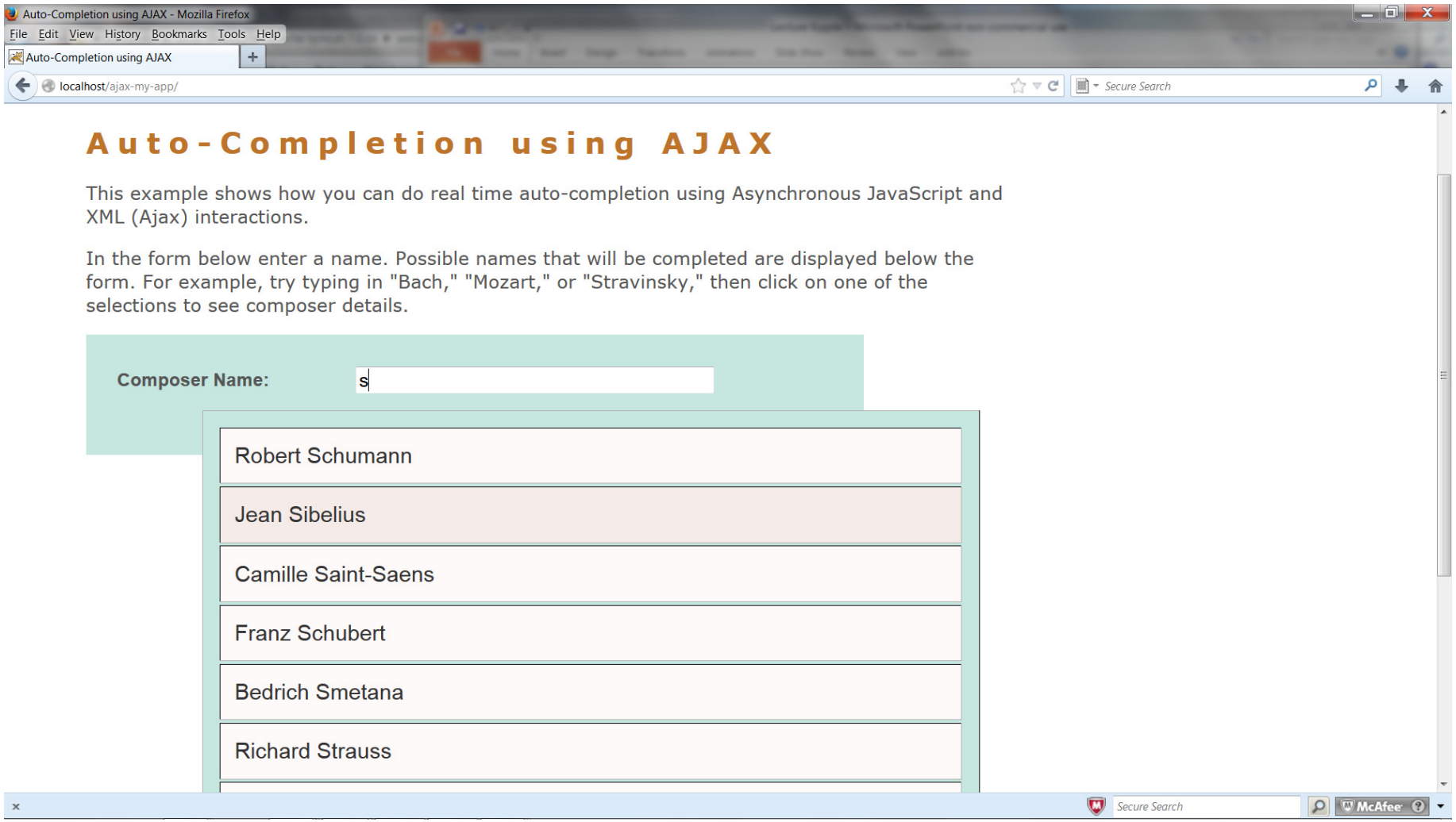
Auto-Completion using AJAX

- And as you start typing you get this ….

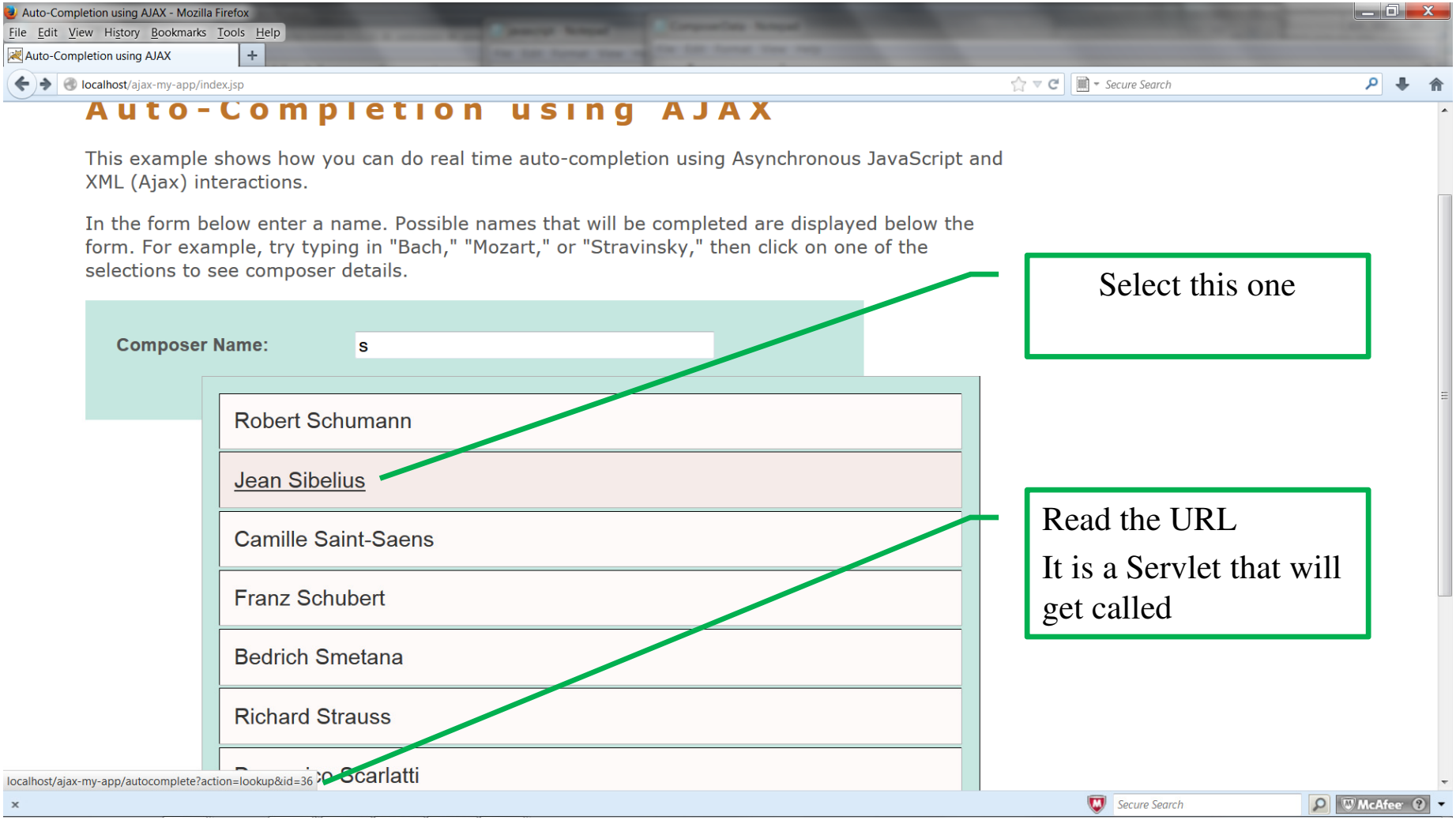


Auto-Completion using AJAX

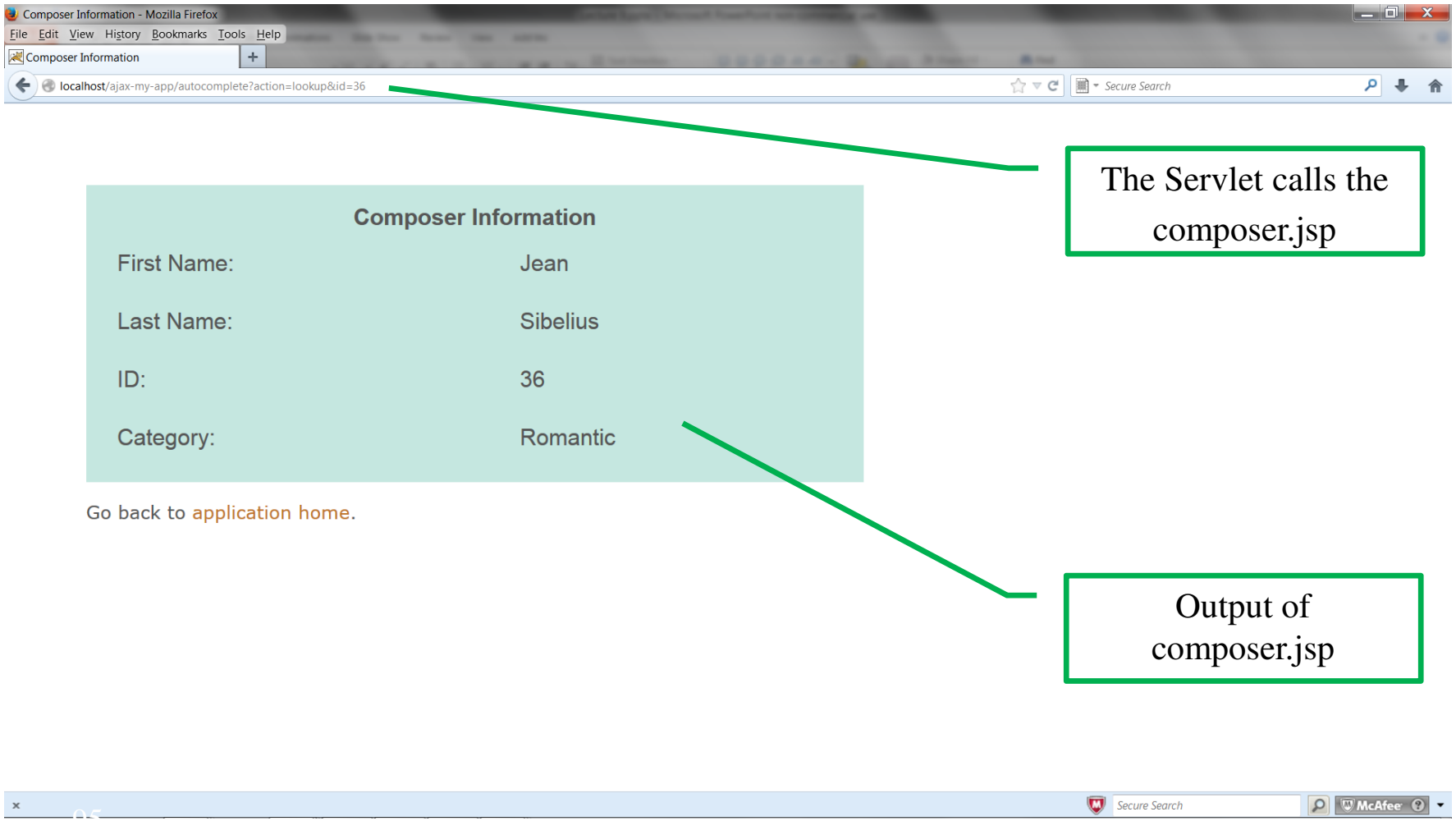
- Or this …



Auto-Completion using AJAX

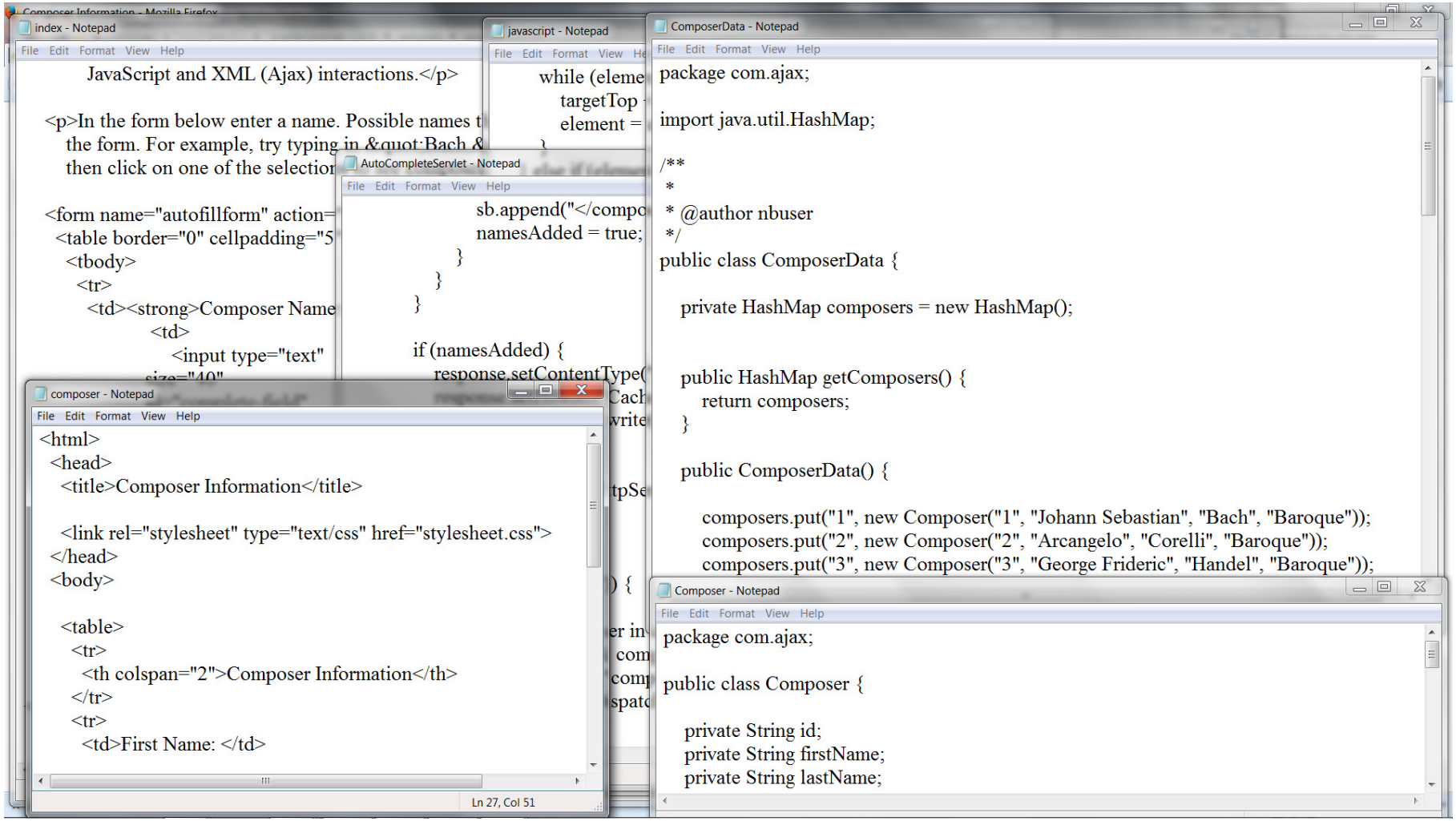


Auto-Completion using AJAX



Auto-Completion using AJAX

- Six Files For Auto-Completion Example



Auto-Completion using AJAX

- index.jsp file

index - Notepad

File Edit Format View Help

```
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Auto-Completion using AJAX</title>

     <script type="text/javascript" src="javascript.js"></script>
     <link rel="stylesheet" type="text/css" href="stylesheet.css">
</head>
<body onload="init()">
  <h1>Auto-Completion using AJAX</h1>

     <p>This example shows how you can do real time auto-completion using Asynchronous
        JavaScript and XML (Ajax) interactions.</p>

  <p>In the form below enter a name. Possible names that will be completed are displayed below
     the form. For example, try typing in &amp;quot;Bach,&amp;quot &amp;quot;Mozart,&amp;quot; or &amp;quot;Stravinsky,&amp;quot;
     then click on one of the selections to see composer details.</p>

  <form name="autofillform" action="autocomplete">
    <table border="0" cellpadding="5">
      <tbody>
        <tr>
          <td><strong>Composer Name:</strong></td>
                <td>
                   <input type="text"
                size="40"
                id="complete-field"
                        onkeyup="doCompletion()">
                </td>
        </tr>
        <tr>
           <td id="auto-row" colspan="2">
             <table id="complete-table" class="popupBox" />
```

Ln 36, Col 1

Auto-Completion using AJAX

- javascript.js - Complete Action

javascript - Notepad

File Edit Format View Help

```
var req;
var isIE;
var completeField;
var completeTable;
var autoRow;

function init() {
    completeField = document.getElementById("complete-field");
    completeTable = document.getElementById("complete-table");
    autoRow = document.getElementById("auto-row");
    completeTable.style.top = getElementY(autoRow) + "px";
}

function doCompletion() {

    var url = "autocomplete?action=complete&id=" + escape(completeField.value);
    req = initRequest();
    req.open("GET", url, true);
    req.onreadystatechange = callback;
    req.send(null);
}

function initRequest() {
    if (window.XMLHttpRequest) {
        if (navigator.userAgent.indexOf('MSIE') != -1) {
            isIE = true;
        }
        return new XMLHttpRequest();
    } else if (window.ActiveXObject) {
        isIE = true;
        return new ActiveXObject("Microsoft.XMLHTTP");
```

Ln 14, Col 1

Auto-Completion using AJAX

- javascript.js – Lookup Action

javascript - Notepad

File Edit Format View Help

```
        }
    }
}

function appendComposer(firstName,lastName,composerId) {

    var row;
    var cell;
    var linkElement;

    if (isIE) {
        completeTable.style.display = 'block';
        row = completeTable.insertRow(completeTable.rows.length);
        cell = row.insertCell(0);
    } else {
        completeTable.style.display = 'table';
        row = document.createElement("tr");
        cell = document.createElement("td");
        row.appendChild(cell);
        completeTable.appendChild(row);
    }

    cell.className = "popupCell";

    linkElement = document.createElement("a");
    linkElement.className = "popupItem";
    linkElement.setAttribute("href", "autocomplete?action=lookup&id=" + composerId);
    linkElement.appendChild(document.createTextNode(firstName + " " + lastName));
    cell.appendChild(linkElement);
}
```

Ln 67, Col 4

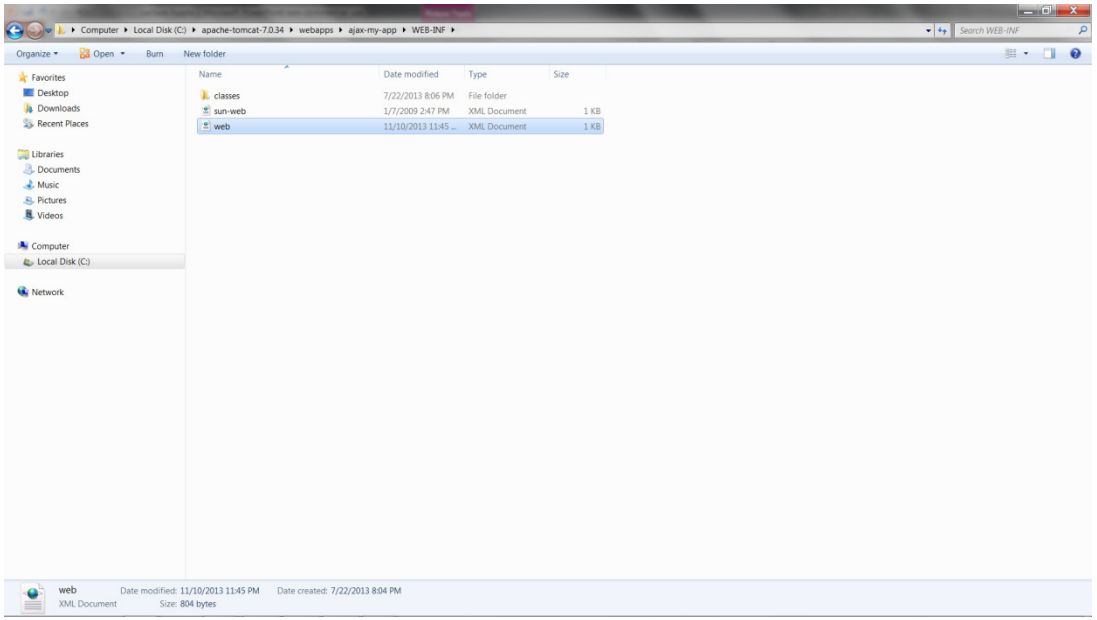
Auto-Completion using AJAX

- Web.xml for Servlet URL mapping

web - Notepad

File Edit Format View Help

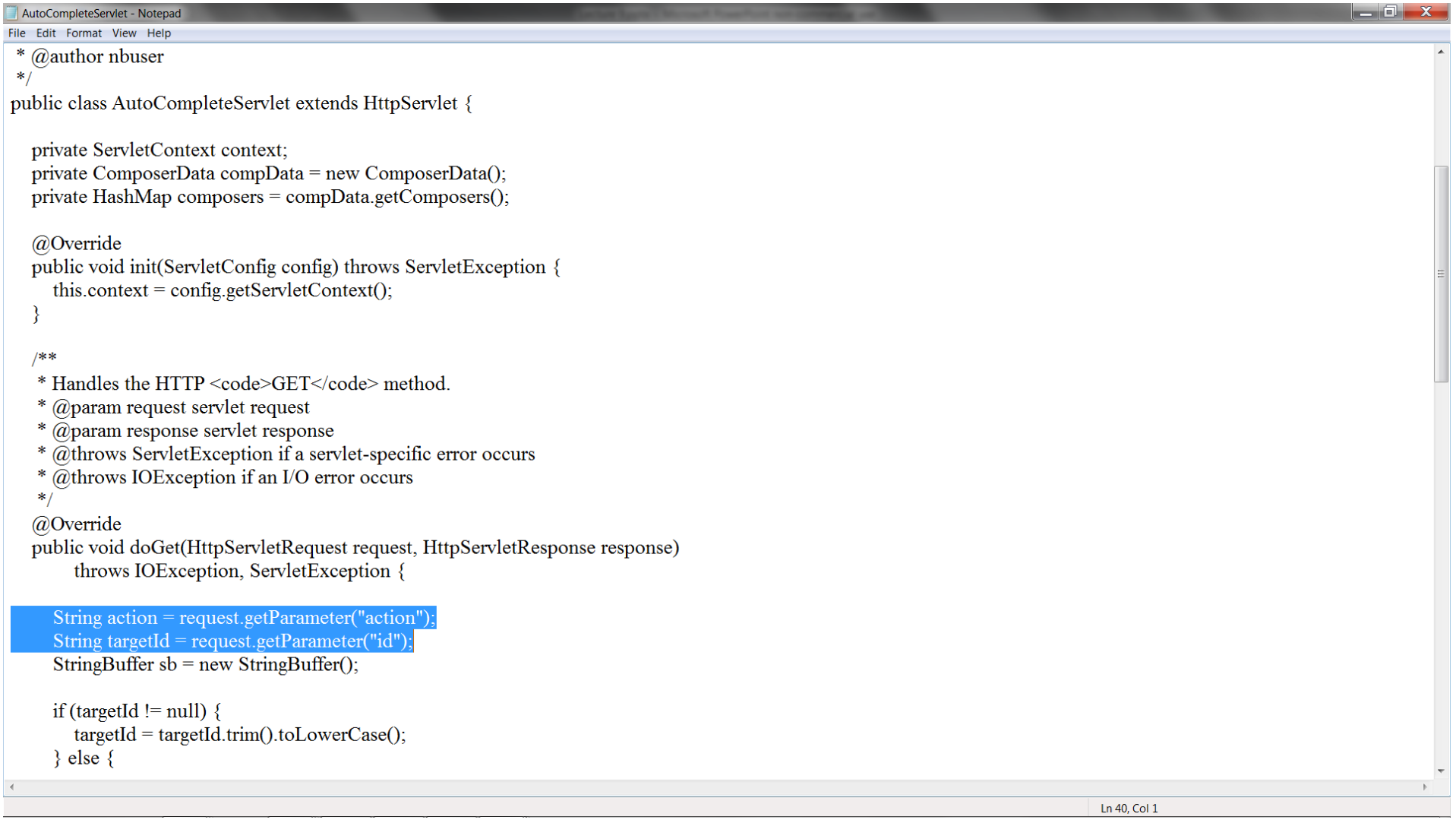
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://java.sun.com/xn
    <servlet>
        <servlet-name>AutoCompleteServlet</servlet-name>
        <servlet-class>com.ajax.AutoCompleteServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>AutoCompleteServlet</servlet-name>
        <url-pattern>/autocomplete</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>
</web-app>
```



Ln 7, Col 1

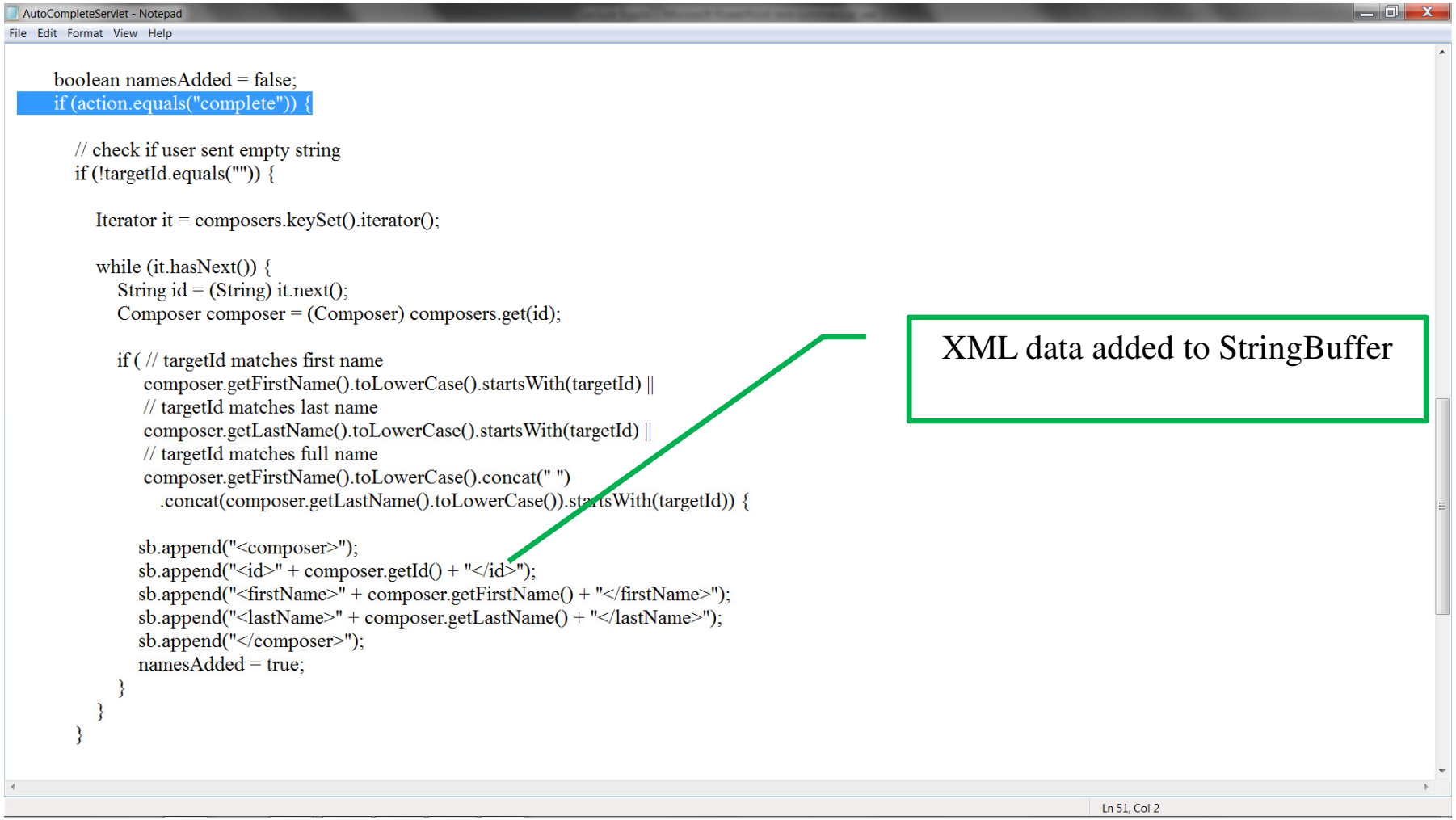
Auto-Completion using AJAX

- AutoCompleteServlet



Auto-Completion using AJAX

- AutoCompleteServlet



Auto-Completion using AJAX

- AutoCompleteServlet

AutoCompleteServlet - Notepad

File Edit Format View Help

```
                // targetId matches full name
                composer.getFirstName().toLowerCase().concat(" ")
                  .concat(composer.getLastName().toLowerCase()).startsWith(targetId)) {

                sb.append("<composer>");
                sb.append("<id>" + composer.getId() + "</id>");
                sb.append("<firstName>" + composer.getFirstName() + "</firstName>");
                sb.append("<lastName>" + composer.getLastName() + "</lastName>");
                sb.append("</composer>");
                namesAdded = true;
            }
        }
    }

    if (namesAdded) {
        response.setContentType("text/xml");
        response.setHeader("Cache-Control", "no-cache");
        response.getWriter().write("<composers>" + sb.toString() + "</composers>");
    } else {
        //nothing to show
        response.setStatus(HttpServletResponse.SC_NO_CONTENT);
    }
}

if (action.equals("lookup")) {

    // put the target composer in the request scope to display
    if ((targetId != null) && composers.containsKey(targetId.trim())) {
        request.setAttribute("composer", composers.get(targetId));
        context.getRequestDispatcher("/composer.jsp").forward(request, response);
    }
```

Write XML data to Response object

Ln 80, Col 1

Auto-Completion using AJAX

- AutoCompleteServlet

AutoCompleteServlet - Notepad

File Edit Format View Help

```
                sb.append("<composer>");
                sb.append("<id>" + composer.getId() + "</id>");
                sb.append("<firstName>" + composer.getFirstName() + "</firstName>");
                sb.append("<lastName>" + composer.getLastName() + "</lastName>");
                sb.append("</composer>");
                namesAdded = true;
            }
        }
    }

    if (namesAdded) {
        response.setContentType("text/xml");
        response.setHeader("Cache-Control", "no-cache");
        response.getWriter().write("<composers>" + sb.toString() + "</composers>");
    } else {
        //nothing to show
        response.setStatus(HttpServletResponse.SC_NO_CONTENT);
    }
  }

  if (action.equals("lookup")) {

    // put the target composer in the request scope to display
    if ((targetId != null) && composers.containsKey(targetId.trim())) {
      request.setAttribute("composer", composers.get(targetId));
      context.getRequestDispatcher("/composer.jsp").forward(request, response);
    }
  }
 }
}
```

If the user selects an item from the table-list of items “lookup”, trigger the composer.jsp

Ln 90, Col 2

Auto-Completion using AJAX

- ComposerData Class

ComposerData - Notepad

File Edit Format View Help

```
package com.ajax;

import java.util.HashMap;

/**
 *
 * @author nbuser
 */
public class ComposerData {

    private HashMap composers = new HashMap();


    public HashMap getComposers() {
        return composers;
    }

    public ComposerData() {

        composers.put("1", new Composer("1", "Johann Sebastian", "Bach", "Baroque"));
        composers.put("2", new Composer("2", "Arcangelo", "Corelli", "Baroque"));
        composers.put("3", new Composer("3", "George Frideric", "Handel", "Baroque"));
        composers.put("4", new Composer("4", "Henry", "Purcell", "Baroque"));
        composers.put("5", new Composer("5", "Jean-Philippe", "Rameau", "Baroque"));
        composers.put("6", new Composer("6", "Domenico", "Scarlatti", "Baroque"));
        composers.put("7", new Composer("7", "Antonio", "Vivaldi", "Baroque"));

        composers.put("8", new Composer("8", "Ludwig van", "Beethoven", "Classical"));
        composers.put("9", new Composer("9", "Johannes", "Brahms", "Classical"));
        composers.put("10", new Composer("10", "Francesco", "Cavalli", "Classical"));
        composers.put("11", new Composer("11", "Fryderyk Franciszek", "Chopin", "Classical"));
```

Ln 15, Col 26

Auto-Completion using AJAX

- Composer Class

Composer - Notepad

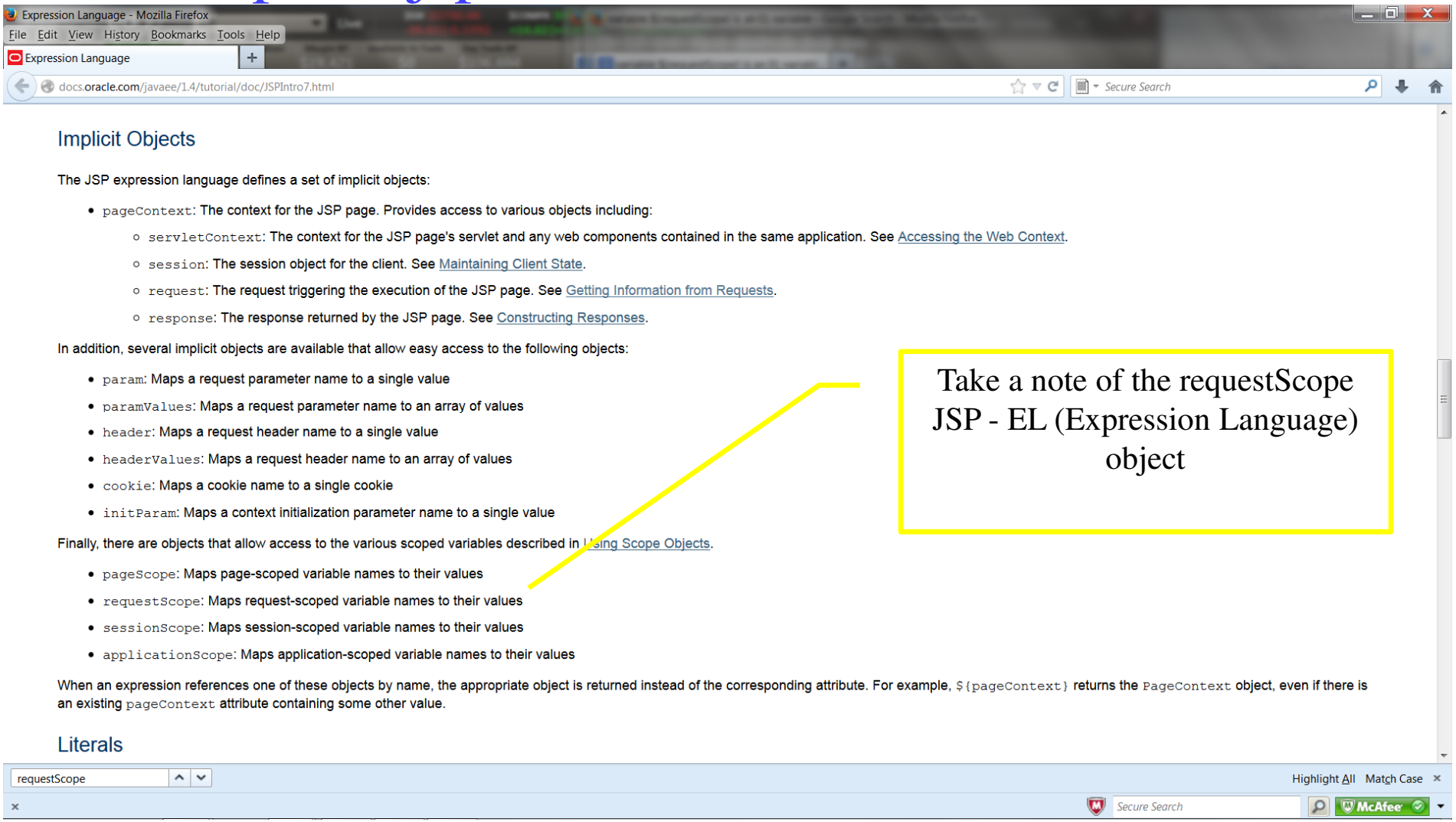
File Edit Format View Help

```
package com.ajax;

public class Composer {

    private String id;
    private String firstName;
    private String lastName;
    private String category;


    public Composer (String id, String firstName, String lastName, String category) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.category = category;
    }

    public String getCategory() {
        return category;
    }

    public String getId() {
        return id;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
```

Ln 25, Col 5

Auto-Completion using AJAX

- Composer.jsp



Take a note of the requestScope JSP - EL (Expression Language) object

Auto-Completion using AJAX

- Composer.jsp

composer - Notepad

File Edit Format View Help

```
<html>
 <head>
  <title>Composer Information</title>

  <link rel="stylesheet" type="text/css" href="stylesheet.css">
 </head>
 <body>

  <table>
   <tr>
    <th colspan="2">Composer Information</th>
   </tr>
   <tr>
    <td>First Name: </td>
    <td>${requestScope.composer.firstName}</td>
   </tr>
   <tr>
    <td>Last Name: </td>
    <td>${requestScope.composer.lastName}</td>
   </tr>
   <tr>
    <td>ID: </td>
    <td>${requestScope.composer.id}</td>
   </tr>
   <tr>
    <td>Category: </td>
    <td>${requestScope.composer.category}</td>
   </tr>
  </table>

  <p>Go back to <a href="index.jsp" class="link">application home</a>.</p>
```

Ln 27, Col 51

Summary

- JavaScript
  - Define request object
    - Check for both Microsoft and non-MS objects. Identical code in all apps.
  - Initiate request
    - Get request object
    - Designate an anonymous response handler function
    - Initiate a GET request with relative URL
  - Handle response
    - Wait for readyState of 4 and HTTP status of 200
    - Extract return text with responseText
    - Do something with result
      - Use innerHTML to insert result into designated element
- HTML
  - Give id to placeholder (often a div). Initiate process on user action.
- Java
  - Use JSP, servlet, or combination (MVC) as appropriate.
  - Prevent browser caching.