Session 10 – Ajax: The Basics

Agenda

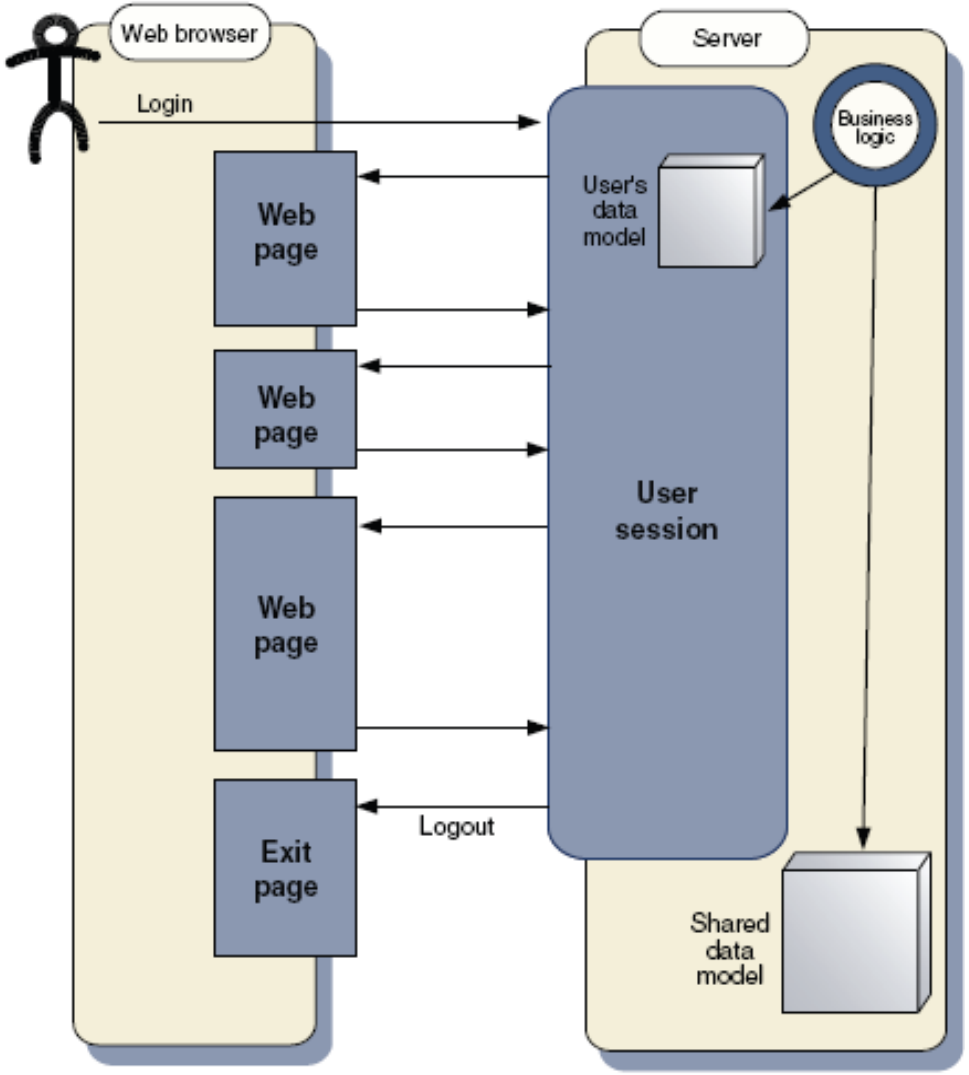
- Ajax motivation
- The basic Ajax process
- Using dynamic content and JSP
- Using dynamic content and servlets
- Sending GET data
- Sending POST data
- Displaying HTML results
- Parsing and displaying XML results
- Toolkits

Motivation

Why Ajax?

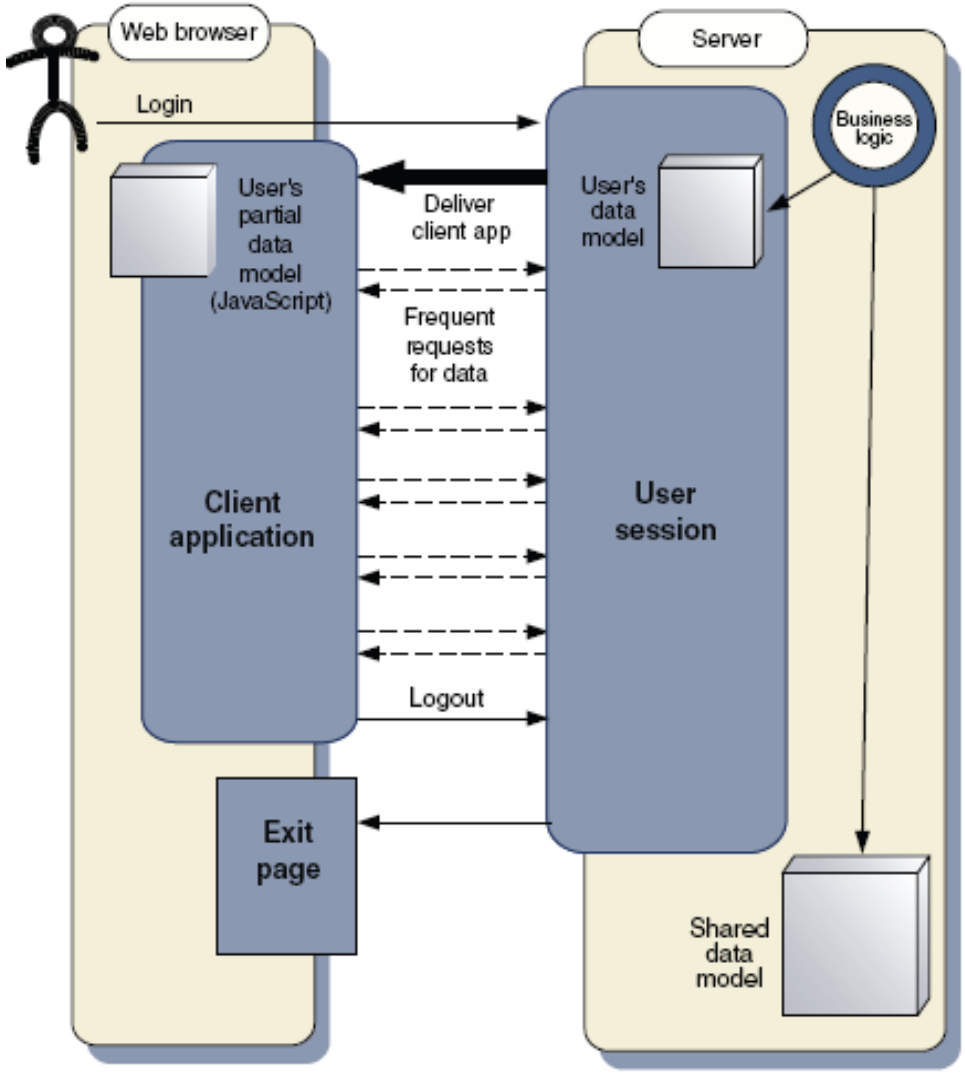
- **HTML and HTTP are weak**
  - Non-interactive
  - Coarse-grained updates
- **Everyone wants to use a browser**
  - Not a custom application
- **"Real" browser-based active content**
  - Failed: Java Applets
    - Not universally supported; can't interact with the HTML
  - Serious alternative: Flash (and Flex)
    - Not yet universally supported; limited power
  - New and unproven
    - Microsoft Silverlight
    - JavaFX
    - Adobe Apollo

Lifecycle of a classic web application



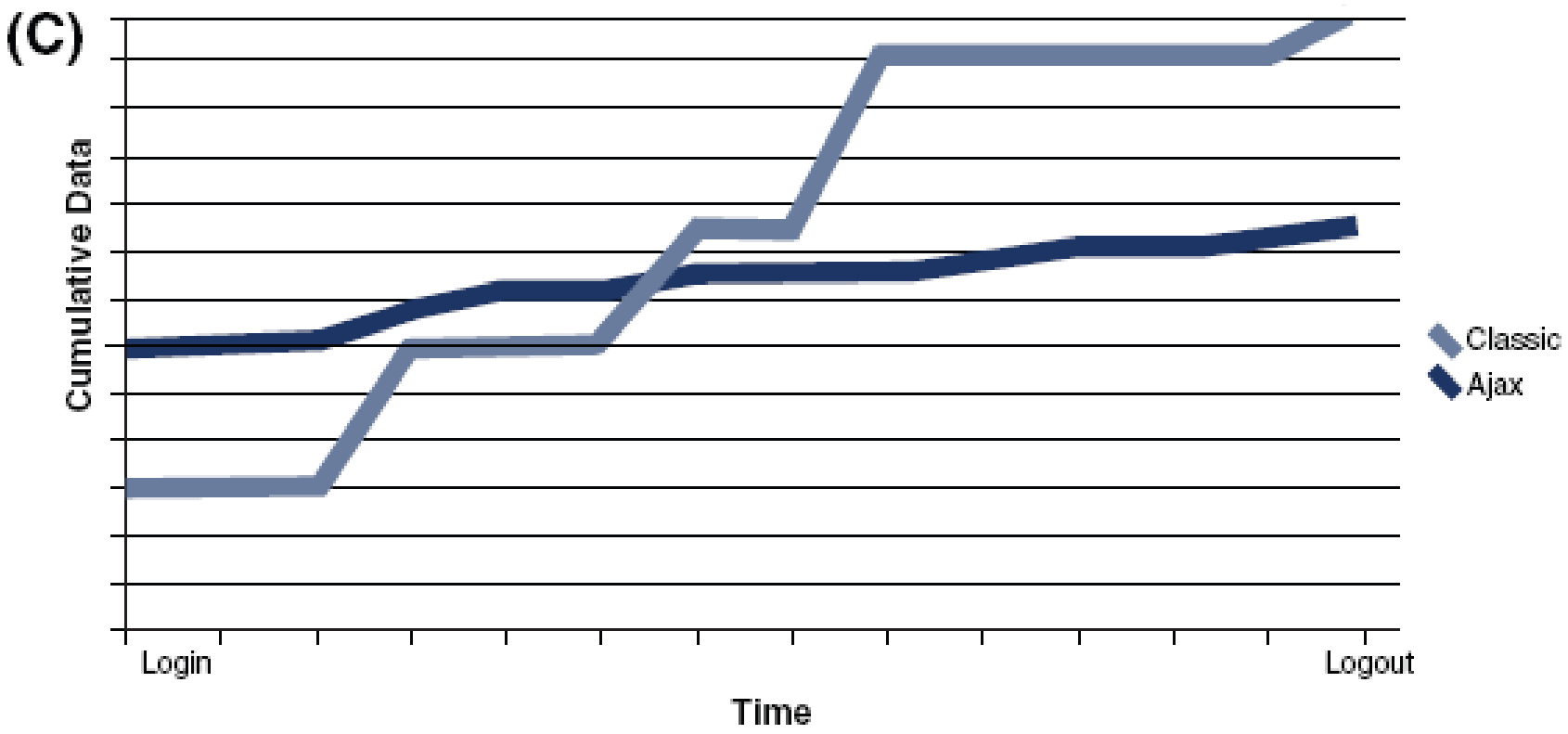
All the state of the user's "conversation" with the application is held on the web server.
The user sees a succession of pages, none of which can advance the broader conversation without going back to the server.

Lifecycle of an Ajax application



When the user logs in, a client application is delivered to the browser.
This application can field many user interactions independently, or else send requests to the server behind the scenes, without interrupting the user's workflow.

Advantage of Ajax on data sending



The Basic Process

The Basic Ajax Process

- **JavaScript**
  - Define an object for sending HTTP requests
  - Initiate request
    - Get request object
    - Designate a request 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**
  - Loads JavaScript
  - Designates control that initiates request
  - Gives ids to input elements that will be read by script

Define a Request Object

```
var request;

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

Version for Internet Explorer (IE 5 and later)

Version for Netscape, Firefox, and Opera (Netscape 5 and later)

Fails on older and nonstandard browsers

Initiate Request

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

Response handler function name

URL of server-side resource

Don't wait for response
(Send request asynchronously)

POST data
(always null for GET)

Handle Response

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

Response is returned from server
(handler gets invoked multiple times)

Text of server response

Pop up dialog box

Complete JavaScript Code (show-message.js)

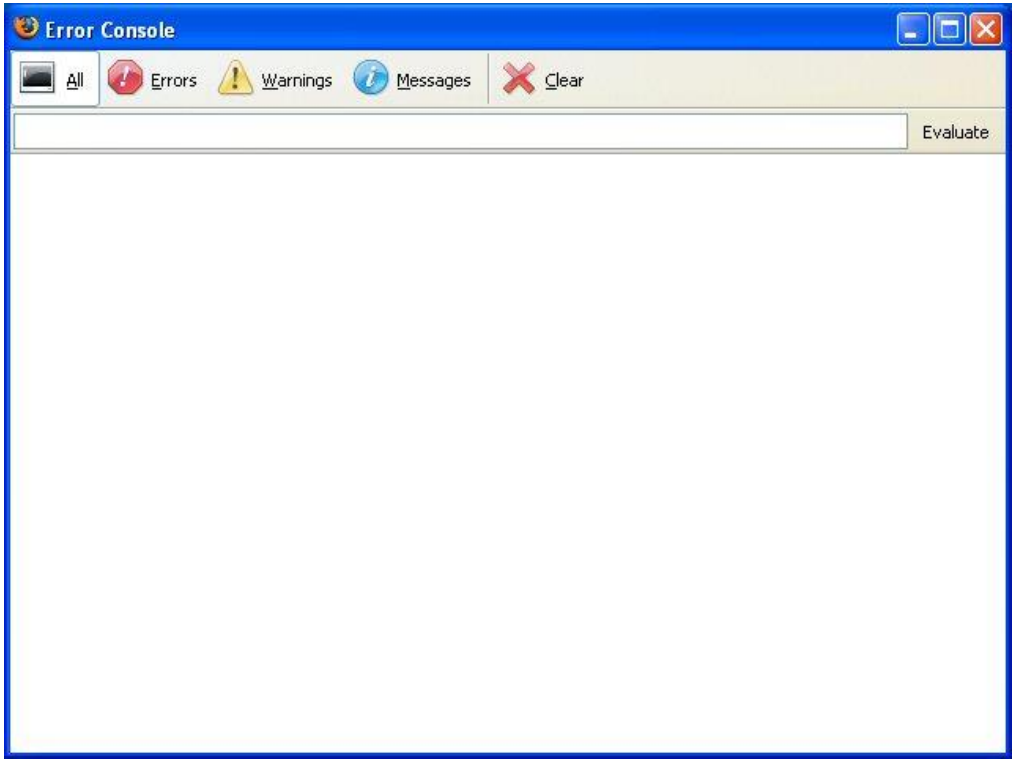
```
var request;

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

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

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

Exercise

- Given a file “Books.xml” to store data as you practice in lab session.
- Write pages named “Ajax.html”, “Data.java” and “javascript.js” to load data from “Books.xml” and display them in Ajax.html by table format.

The Firefox JavaScript Console

- **Invoke with Control-Shift-J**



- **Also see Venkman JavaScript debugger**
  - http://www.mozilla.org/projects/venkman/
  - https://addons.mozilla.org/firefox/216/

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()"/>
```

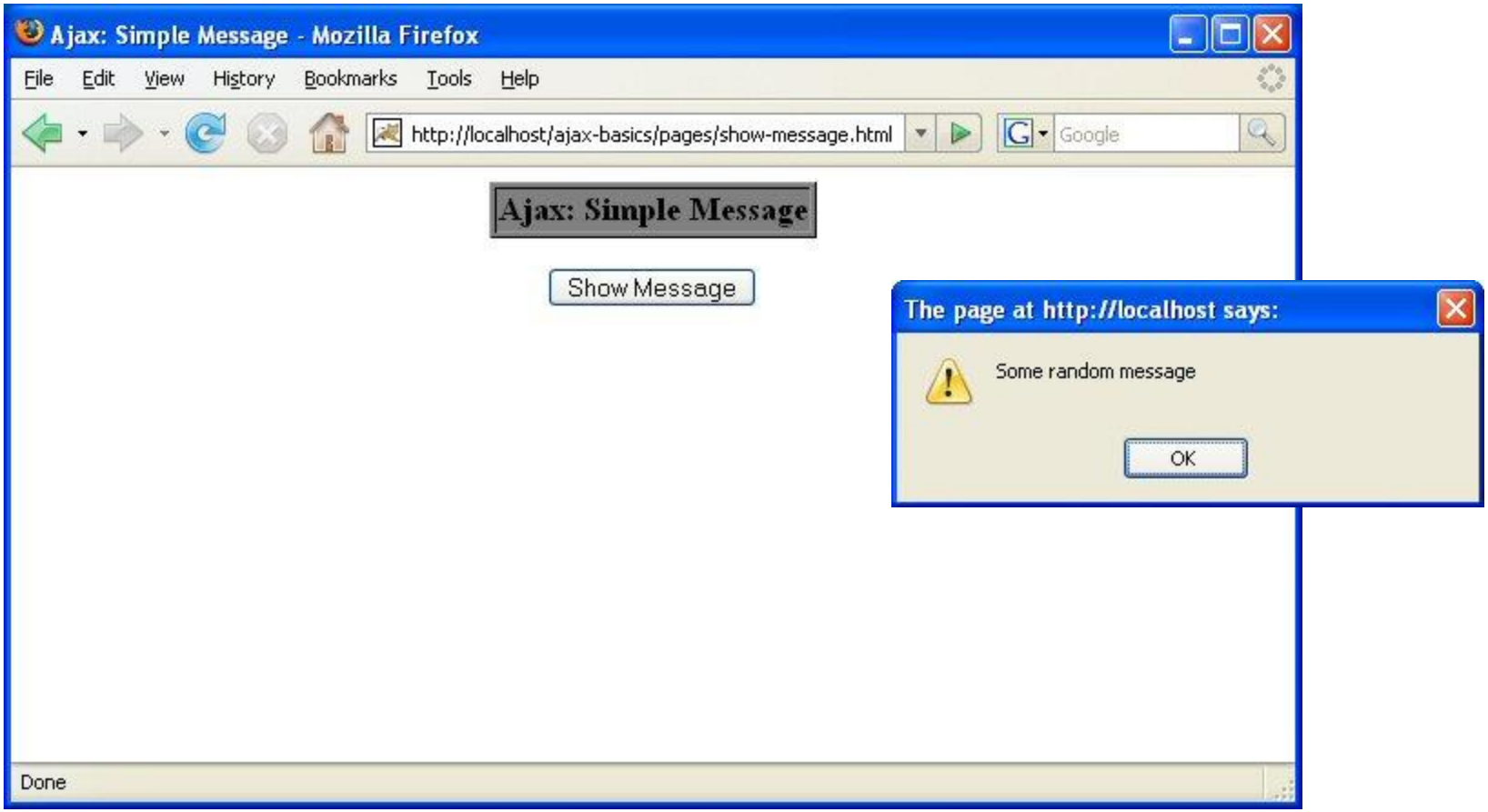
Internet Explorer XHTML Bugs

- **Can't handle XML header**
  - XML documents in general are supposed to start with XML header:
    - <?xml version="1.0" encoding="UTF-8"?>
      <!DOCTYPE html ...>
      <html xmlns="http://www.w3.org/1999/xhtml">...</html>
  - XHTML specification recommends using it
  - *But*... Internet Explorer will switch to quirks-mode (from standards-mode) if DOCTYPE is not first line.
    - Many recent style sheet formats will be ignored
    - So omit XML header
- **Needs separate end tags in some places**
  - Scripts will not load if you use <script .../> instead of <script...></script>

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/>
<form action="#">
 <input type="button" value="Show Message"
        onclick="sendRequest()"/>
</form>
</center></body></html>
```

The Basic Process: Results



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 a request handler function
      - Supply as onreadystatechange attribute of request
    - Initiate a GET or POST request to a JSP page
    - 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**
  - Loads JavaScript from centralized directory
  - Designates control that initiates request
  - Gives ids to input elements that will be read by script

Define a Request Object

```
var request;

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

No changes from previous example

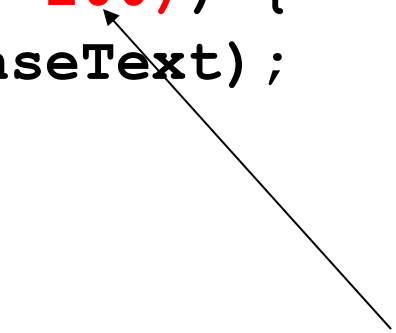
Initiate Request

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



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

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

```
var request;

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

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

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

HTML Code

- **Loads JavaScript from central location**

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

- **Passes JSP address to main function**

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

- **Uses style sheet**

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

Note single quotes (Because of double quotes inside parens.)

HTML Code (show-time-1.html)

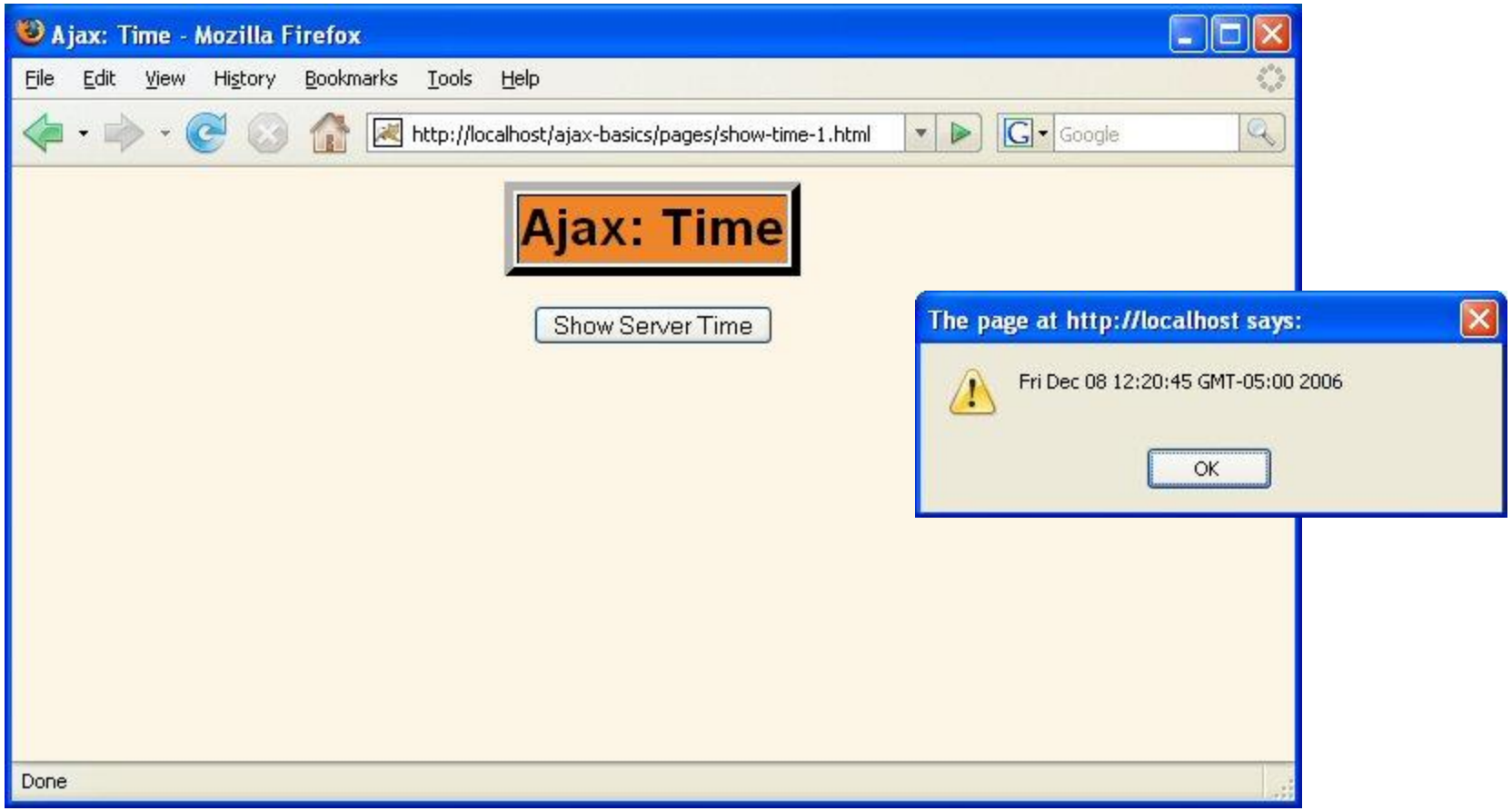
```
<!DOCTYPE html PUBLIC "..."
  "http://www.w3.org/..."><
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Ajax: Time</title>
<link rel="stylesheet"
      href="../css/styles.css"
      type="text/css"/>
<script src="../scripts/ajax-basics.js"
        type="text/javascript"></script>
</head>
<body>
...
<form action="#">
 <input type="button" value="Show Server Time"
        onclick='sendRequest("show-time.jsp")'/>
</form>
</center></body></html>
```

JSP Code (show-time.jsp)

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

- **Note: executing this example**
  - You must run from Tomcat.
    - Otherwise JSP cannot execute
    - Otherwise status code is -1, not 200

Message from JSP: Results



Dynamic Content from Servlet

JSP Example: Design Deficiencies

- **Caching problems**
  - The URL stays the same but the output changes
  - So if browser caches page, you get the wrong time
  - Solution: send Cache-Control and Pragma headers
- **Date was not formatted**
  - Just used the toString method of Date
  - Solution: use String.format (sprintf) and %t controls
- **JSP is wrong technology**
  - JSP is best for lots of HTML and little or no logic/Java
  - But now we have logic but no HTML
  - Solution: use a servlet

Steps

- **JavaScript**
  - Define an object for sending HTTP requests
  - Initiate request
    - Get request object
    - Designate a request 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**
  - Loads JavaScript from centralized directory
  - Designates control that initiates request
  - Gives ids to input elements that will be read by script

Define a Request Object

```
var request;

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

No changes from previous example

Initiate Request

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

No changes from previous example

Handle Response

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

No changes from previous example

HTML Code (show-time-2.html)

```
<!DOCTYPE html PUBLIC "..."
  "http://www.w3.org/..."><
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Ajax: Time</title>
<link rel="stylesheet"
      href="../css/styles.css"
      type="text/css"/>
<script src="../scripts/ajax-basics.js"
        type="text/javascript"></script>
</head>
<body>
...
<form action="#">
 <input type="button" value="Show Server Time"
        onclick='sendRequest("../show-time")'/>
</form>
</center></body></html>
```

Address of servlet. (From url-pattern of servlet-mapping.)

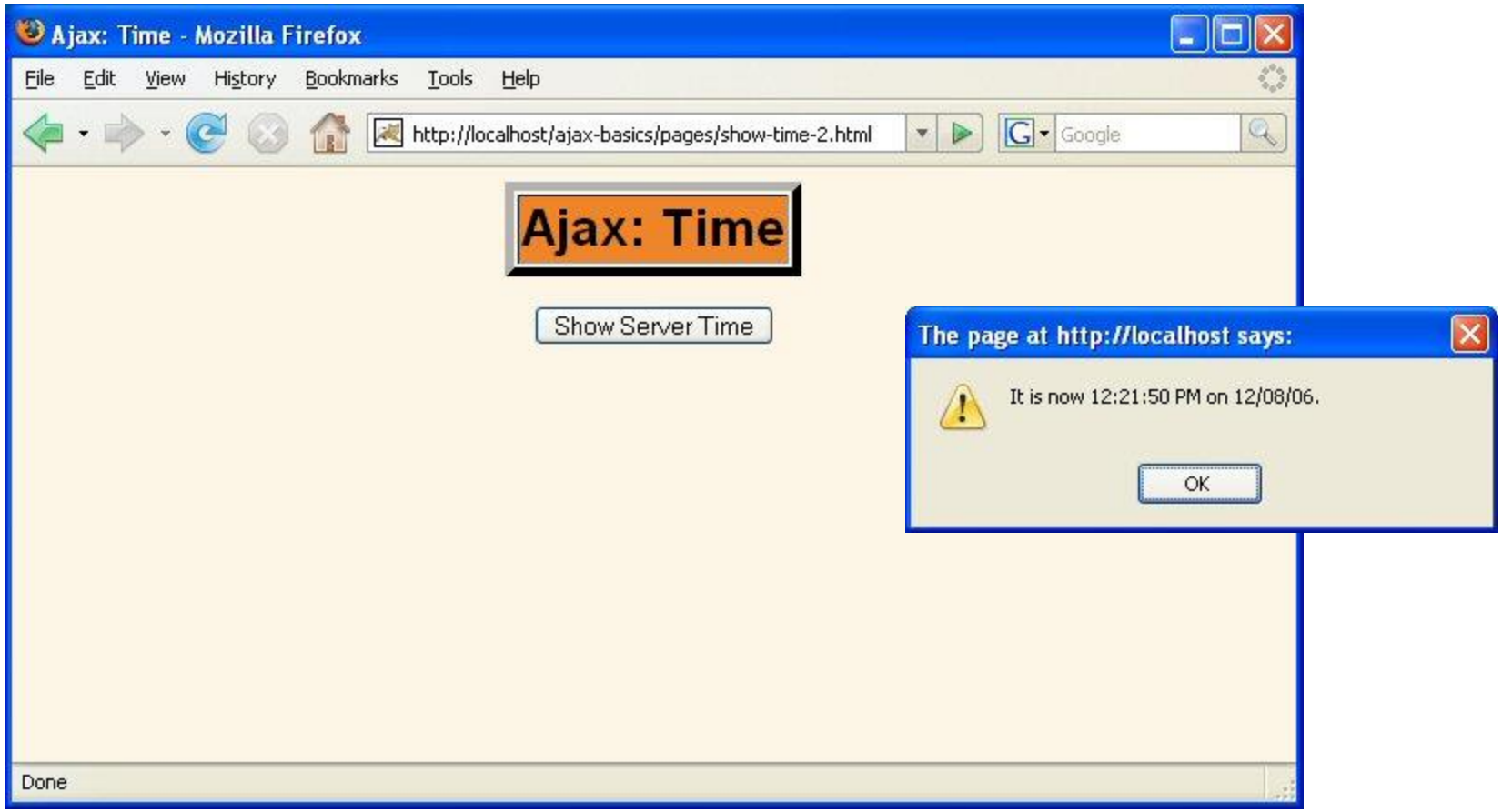
Servlet Code

```
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");
    response.setContentType("text/html");
    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>
...
```

Message from Servlet: Results



Sending GET Data

Servlet Example: Design Deficiencies

- **No data sent from HTML page to servlet**
  - Solution: attach data to end of the URL (GET data)
    - Use normal GET format:
      - *mainaddress?var1=val1&var2=val2*

Steps

- **JavaScript**
  - Define an object for sending HTTP requests
  - Initiate request
    - Get request object
    - Designate a request handler function
      - Supply as onreadystatechange attribute of request
    - Initiate a GET request to a servlet
      - URL has GET data attached at the end
    - 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**
  - Loads JavaScript from centralized directory
  - Designates control that initiates request
  - Gives ids to input elements that will be read by script

JavaScript Code

- **No changes from previous example**

HTML Code (show-time-3.html)

```
<!DOCTYPE html PUBLIC "..."
  "http://www.w3.org/...">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Ajax: Time</title>
<link rel="stylesheet"
      href="../css/styles.css"
      type="text/css"/>
<script src="../scripts/ajax-basics.js"
        type="text/javascript"></script>
</head>
<body>
...
<form action="#">
 <input type="button" value="Show Time in Chicago"
        onclick=
         'sendRequest("../show-time-in-city?city=Chicago")'/>
</form>
</center></body></html>
```

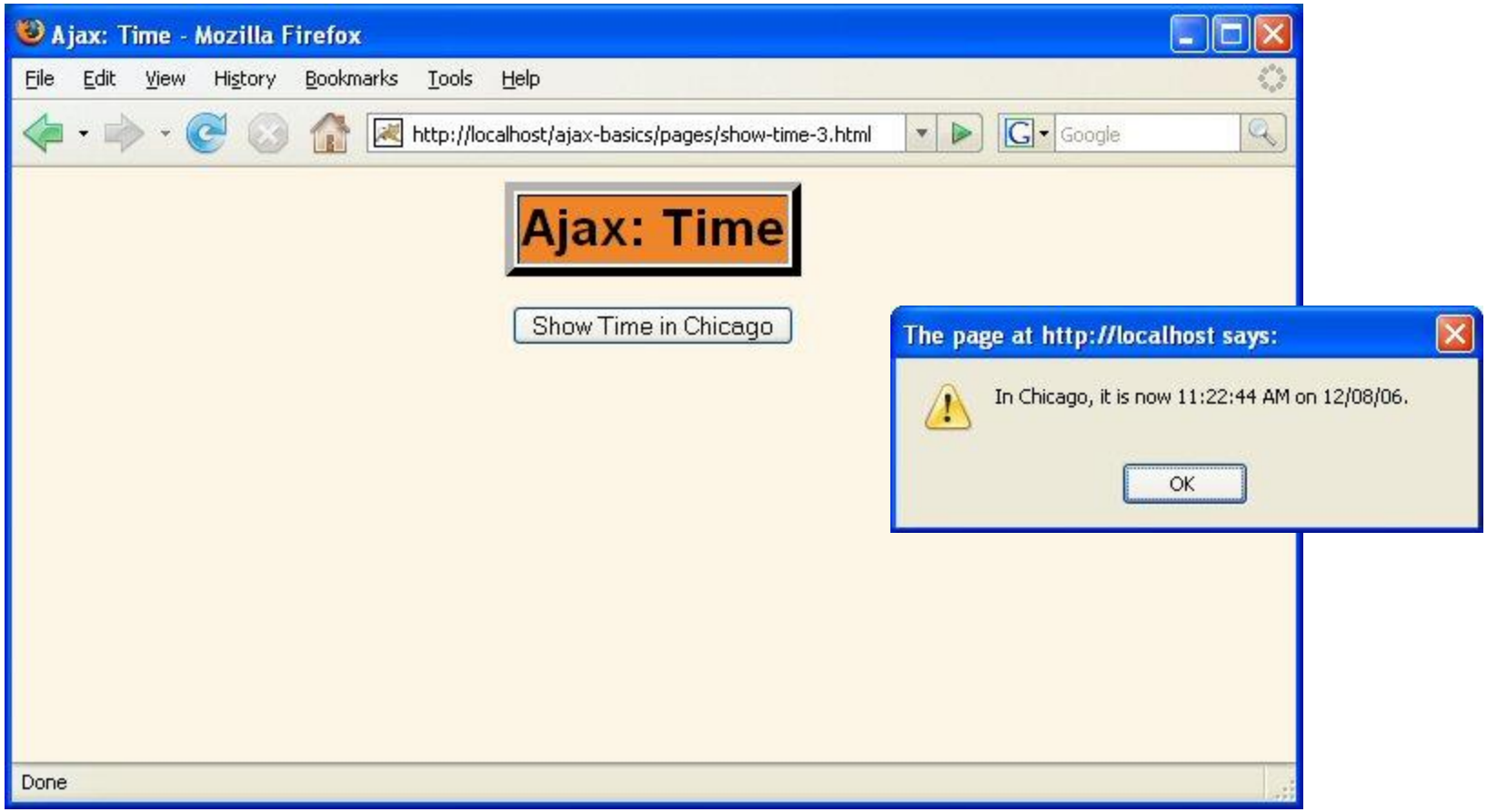
Servlet Code

```
public class ShowTimeInCity extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Pragma", "no-cache");
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String city = request.getParameter("city");
    ...
    String message = TimeZone.getTimeString(city);
    ...
    out.print(message);
  }

  ...
}
```

TimeZone Class

- **Maintains a list of cities and associated time zones**
  - Given the name of a city, it finds the difference in hours between that city's time and server time (east coast US)
- **Computes server time**
  - Using standard GregorianCalendar class
- **Converts to time in that city**
  - By calling the "add" method with the timezone offset
- **Formats the time and day**
  - Using String.format with %tr and %tD

Sending GET Data: Results



Sending POST Data

GET Example: Design Deficiencies

- **City name was always Chicago**
  - Solution: read data from textfield
- **Data sent by GET**
  - Sometimes POST is preferred
  - Solution: use POST instead of GET
- **GET vs. POST**
  - In normal Web pages, there are compelling reasons for choosing POST or GET
    - POST: simpler URL, data hidden from people looking over your shoulder, larger amounts of data can be sent
    - GET: can bookmark results page
  - With Ajax, end users don't see URL, so choice is relatively arbitrary
    - Unless there is a very large amount of data

Steps

- **JavaScript**
  - Define an object for sending HTTP requests
  - Initiate request
    - Get request object
    - Designate a request handler function
      - Supply as onreadystatechange attribute of request
    - Initiate a POST request to a servlet
      - Put data to the "send" function
    - 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**
  - Loads JavaScript from centralized directory
  - Designates control that initiates request
  - Gives ids to input elements that will be read by script

Sending POST Data in JavaScript

- **Collect data from form**
  - Give ids to input elements
    <input type="text" id="*some-id*"/>
  - Read data
    var value1 = document.getElementById("*some-id*").value;
  - URL-encode data and form into query string
    var data = "var1=" + escape(value1);
- **Specify POST instead of GET in "open"**
    request.open("POST", address, true);
- **Specify form encoding type**
    request.setRequestHeader("Content-Type",
    "application/x-www-form-urlencoded");
- **Supply data in "send"**
    request.send(data);

Define a Request Object

```
var request;

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

No changes from previous example

Initiate Request

```
function sendRequestWithData(address, data,
                             responseHandler) {
  request = getRequestObject();
  request.onreadystatechange = responseHandler;
  request.open("POST", address, true);
  request.setRequestHeader("Content-Type",
          "application/x-www-form-urlencoded");
  request.send(data);
}

function showTimeInCity() {
  var address = "../show-time-in-city";
  var city = document.getElementById("city").value;
  var data = "city=" + escape(city);
  sendRequestWithData(address, data, showResponseAlert);
}
```

No changes from previous example

Handle Response

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

No changes from previous example

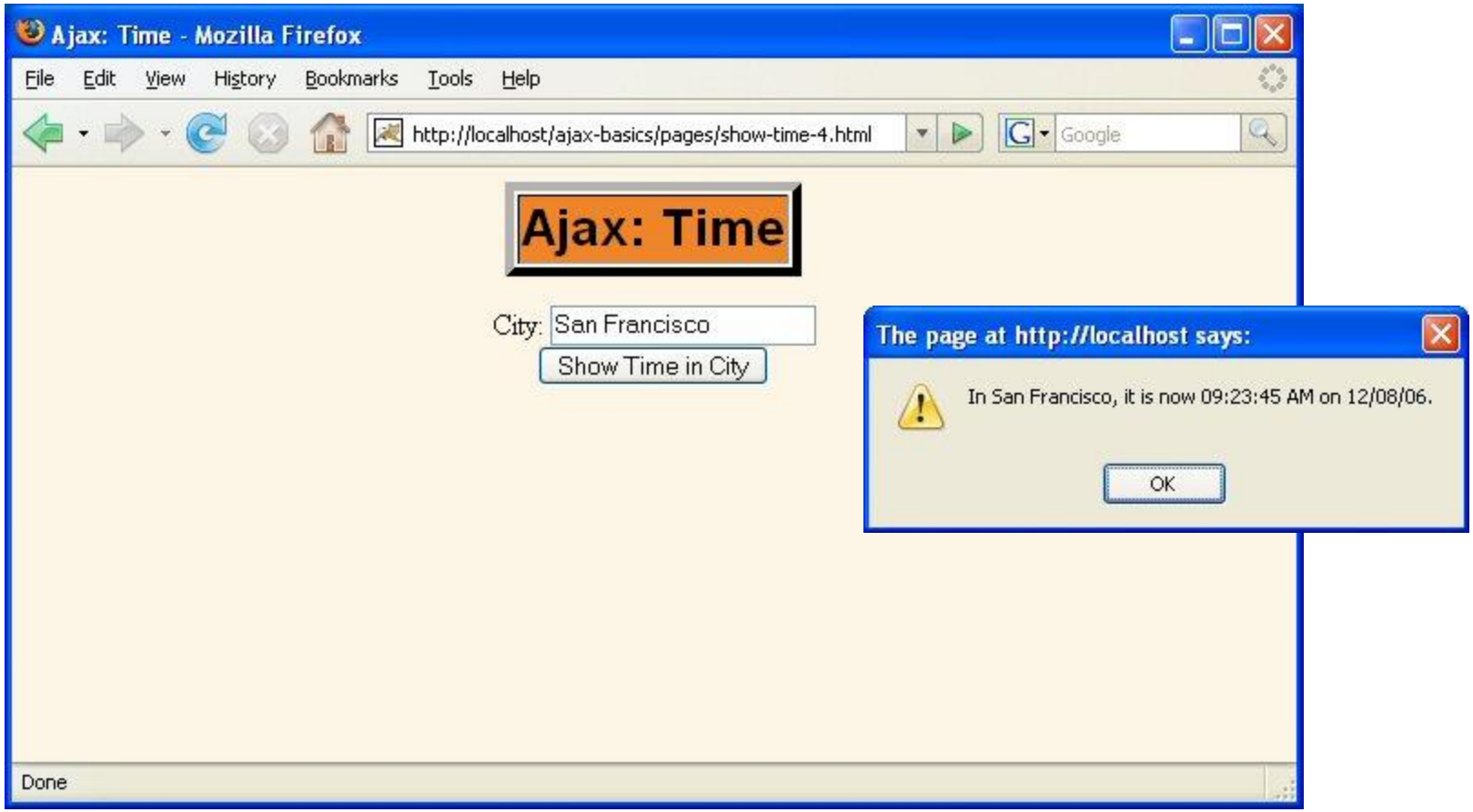
HTML Code

```
<!DOCTYPE html PUBLIC "..."
  "http://www.w3.org/..."><
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Ajax: Time</title>
<link rel="stylesheet"
      href="../css/styles.css"
      type="text/css"/>
<script src="../scripts/ajax-basics.js"
        type="text/javascript"></script>
</head>
<body>
...
<form action="#">
 City: <input type="text" id="city"/><br/>
 <input type="button" value="Show Time in City"
        onclick="showTimeInCity()"/>
</form>
</center></body></html>
```

Servlet Code

```
public class ShowTimeInCity extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Pragma", "no-cache");
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String city = request.getParameter("city");
    ...
    String message = TimeZone.getTimeString(city);
    ...
    out.print(message);
  }

  public void doPost(HttpServletRequest request,
                     HttpServletResponse response)
      throws ServletException, IOException {
    doGet(request, response);
  }
}
```

Sending POST Data: Results



Displaying HTML Output

POST 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 DOM to update page with result text

Steps

- **JavaScript**
  - Define an object for sending HTTP requests
  - Initiate request
    - Get request object
    - Designate a request handler function
    - Initiate a 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
      - Use innerHTML to insert result into "div" element
- **HTML**
  - Loads JavaScript from centralized directory
  - Designates control that initiates request
  - Gives ids to input elements that will be read by script
  - Defines a blank "div" element with a known id

Updating HTML Page Asynchronously

- **HTML**
  - Defines initially blank div element

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

- **JavaScript**
  - Finds element (getElementById) and inserts text into innerHTML property

```
document.getElementById("resultText").innerHTML =
    request.responseText;
```

Define a Request Object

```
var request;

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

No changes from previous example

Initiate Request

```
function sendRequestWithData(address, data,
                             responseHandler) {
  request = getRequestObject();
  request.onreadystatechange = responseHandler;
  request.open("POST", address, true);
  request.setRequestHeader("Content-Type",
          "application/x-www-form-urlencoded");
  request.send(data);
}


function displayTimeInCity() {
  var address = "../show-time-in-city";
  var city = document.getElementById("city").value;
  var data = "city=" + escape(city) + "&useHTML=true";
  sendRequestWithData(address, data, showResponseText);
}
```

No changes from previous example

Handle Response

```
function showResponseText() {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    document.getElementById("resultText").innerHTML =
      request.responseText;
  }
}
```

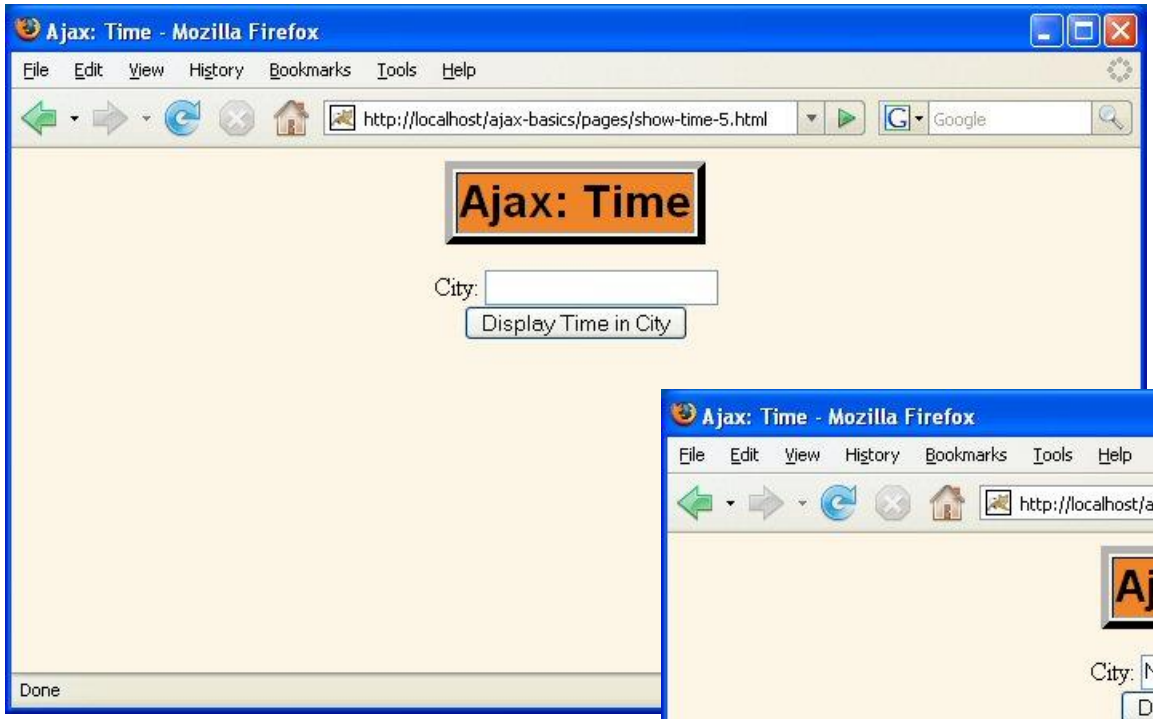
HTML Code (show-time-5.html)

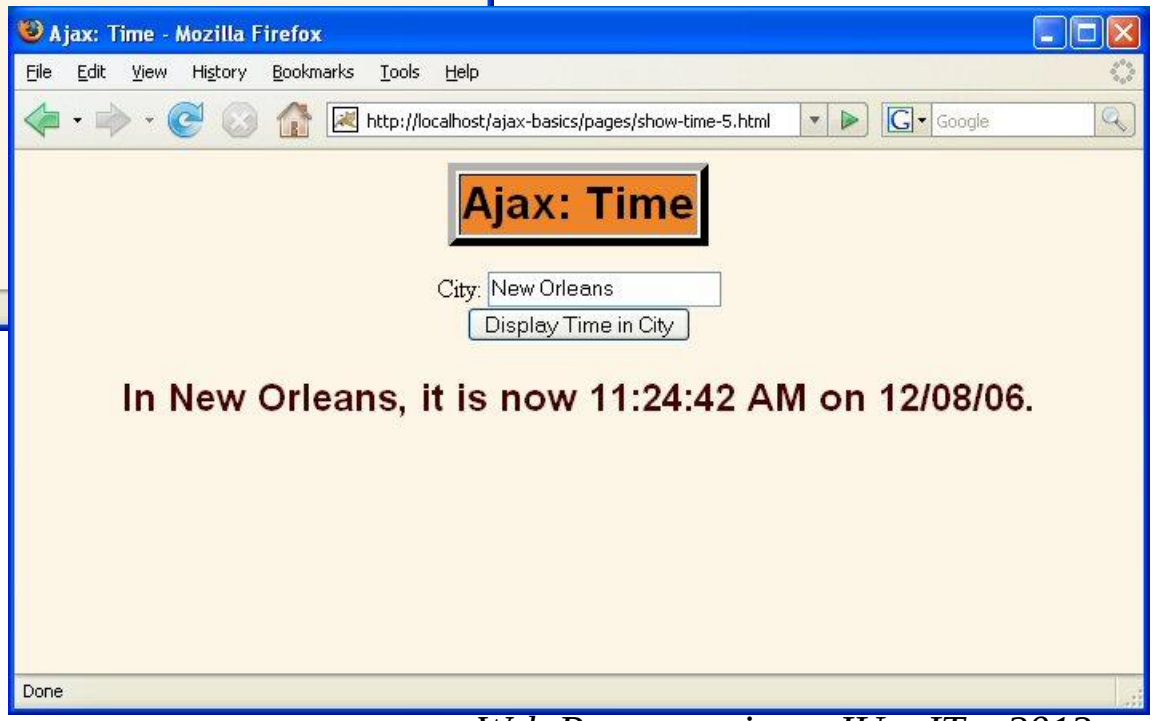
```
<!DOCTYPE html PUBLIC "..." "http://www.w3.org/...">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Ajax: Time</title>
<link rel="stylesheet".../>
<script src="../scripts/ajax-basics.js"
        type="text/javascript"></script>
</head>
<body>
...
<form action="#">
 City: <input type="text" id="city"/><br/>
 <input type="button" value="Display Time in City"
        onclick="displayTimeInCity()"/>
</form>
<div id="resultText"></div>
</center></body></html>
```

Servlet Code

```
public class ShowTimeInCity extends HttpServlet {
  public void doGet(HttpServletRequest request,
                    HttpServletResponse response)
      throws ServletException, IOException {
    response.setHeader("Cache-Control", "no-cache");
    response.setHeader("Pragma", "no-cache");
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String city = request.getParameter("city");
    boolean useHTML = false;
    if (request.getParameter("useHTML") != null) {
      useHTML = true;
    }
    String message = TimeZone.getTimeString(city);
    if (useHTML) {
      message = String.format("<h2>%s</h2>", message);
    }
    out.print(message);
  }
  public void doPost(...) ... { doGet(request, response); }
```

Displaying HTML Output: Results





Parsing and Displaying XML Output

HTML Example: Design Deficiencies

- **Java code generated HTML**
  - Page author has no control over format
  - Cannot use the same data for different tasks
  - Having server-side resource (servlet) generate HTML is often easier and better. But not always.
- **Solution**
  - Have servlet return XML content
  - JavaScript parses XML and decides what to do with it
- **Secondary problem**
  - Generating XML from a servlet is inconvenient
- **Secondary solution**
  - Use MVC architecture on server
    - Servlet creates dynamic data
    - JSP formats the data
    - See detailed lecture on using MVC in Java: http://courses.coreservlets.com/Course-Materials/csajsp2.html

Steps

- **JavaScript**
  - Define an object for sending HTTP requests
  - Initiate request
    - Get request object
    - Designate a request handler function
    - Initiate a POST request to a servlet (that uses MVC)
    - 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
      - Parse data. Use innerHTML to insert result into "div" element
- **HTML**
  - Loads JavaScript from centralized directory
  - Designates control that initiates request
  - Gives ids to input elements that will be read by script
  - Defines a blank "div" element with a known id

Parsing XML in JavaScript

- **Getting the main XML document**
  - Use responseXML instead of responseText

```
var xmlDocument = request.responseXML;
```

  - Get array of elements with getElementsByTagName

```
var names = xmlDocument.getElementsByTagName("name");
```

  - Get body text by getting value of first child node

```
for(var i=0; i<names.length; i++) {
  var name = names[i].childNodes[0].nodeValue;
  doSomethingWith(name);
}
```

- **See detailed lecture on parsing XML with DOM in Java**
  - http://courses.coreservlets.com/Course-Materials/java5.html
  - Java API and JavaScript API are very similar

Define a Request Object

```
var request;

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

No changes from previous example

Initiate Request

```
function sendRequest() {
  request = getRequestObject();
  request.onreadystatechange = showCityTable;
  request.open("POST", "../show-times-in-cities", true);
  request.setRequestHeader("Content-Type",
          "application/x-www-form-urlencoded");
  var timezone = document.getElementById("timezone").value;
  request.send("timezone=" + timezone);
}
```

Handle Response

```
function showCityTable() {
  if ((request.readyState == 4) &&
      (request.status == 200)) {
    var xmlDocument = request.responseXML;
    var names = xmlDocument.getElementsByTagName("name");
    var times = xmlDocument.getElementsByTagName("time");
    var days = xmlDocument.getElementsByTagName("day");
    var tableData = getTableStart();
    for(var i=0; i<names.length; i++) {
      var name = names[i].childNodes[0].nodeValue;
      var time = times[i].childNodes[0].nodeValue;
      var day = days[i].childNodes[0].nodeValue;
      tableData = tableData + getRowData(name, time, day);
    }
    tableData = tableData + getTableEnd();
    document.getElementById("resultText").innerHTML =
      tableData;
  }
}
```

Auxiliary Functions

```
function getTableStart() {
  return("<table border='1'>\n" +
    "  <tr><th>City</th><th>Time</th><th>Day</th></tr>\n");
}

function getRowData(name, time, day) {
  return("  <tr><td>" + name +
         "</td><td>" + time +
         "</td><td>" + day +
         "</td></tr>\n");
}

function getTableEnd() {
  return("</table>\n");
}
```

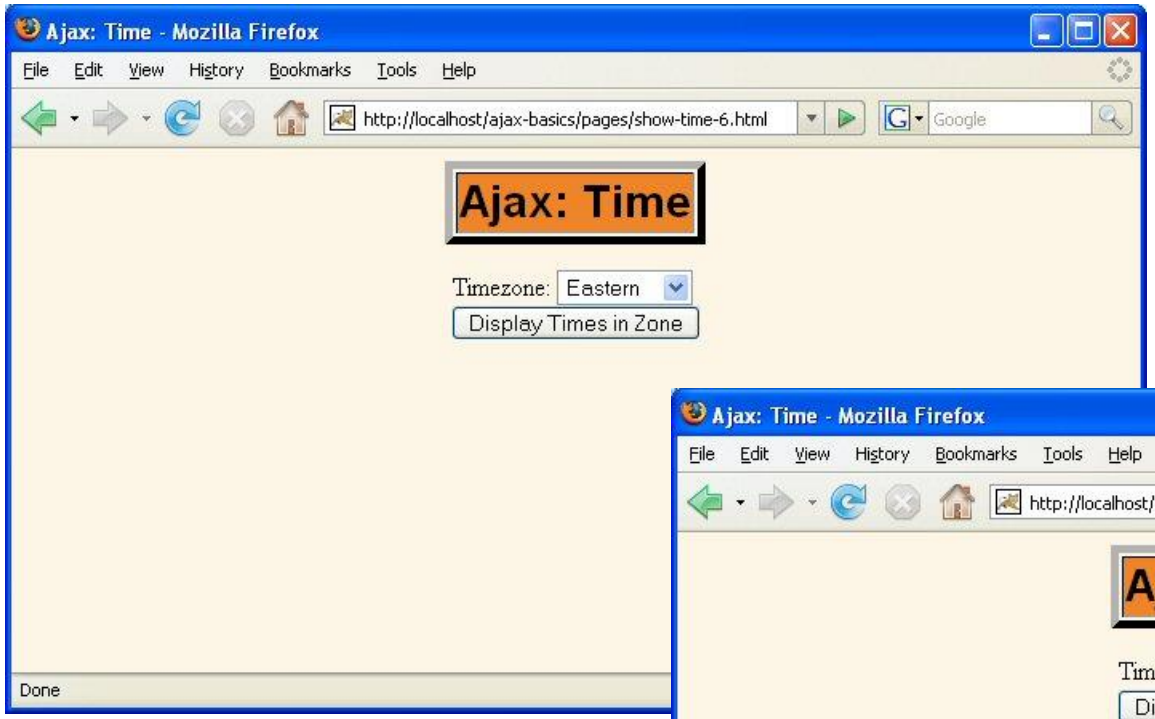
Servlet Code

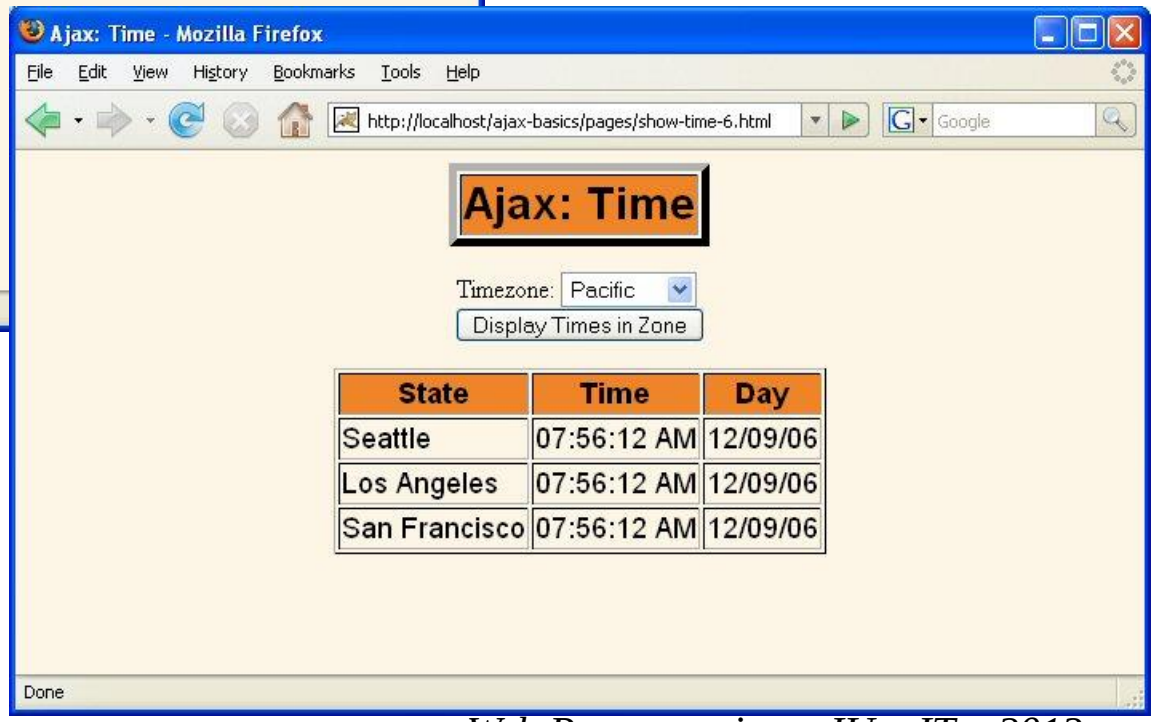
```
response.setHeader("Cache-Control", "no-cache");
response.setHeader("Pragma", "no-cache");
response.setContentType("text/xml");
String timezone = request.getParameter("timezone");
List legalZones =
  Arrays.asList("eastern", "central", "mountain", "pacific");
if ((timezone == null) || (!legalZones.contains(timezone))) {
  timezone = "eastern";
}
timezone = timezone.toLowerCase();
String outputPage =
  String.format("/WEB-INF/results/%s.jsp", timezone);
FormattedTimeAndDay timeAndDay =
  new FormattedTimeAndDay(timezone);
request.setAttribute("timeAndDay", timeAndDay);
RequestDispatcher dispatcher =
  request.getRequestDispatcher(outputPage);
dispatcher.include(request, response);
```

JSP Code (eastern.jsp)

```
<?xml version="1.0" encoding="UTF-8"?>
<cities>
  <city>
    <name>New York</name>
    <time>${timeAndDay.time}</time>
    <day>${timeAndDay.day}</day>
  </city>
  <city>
    <name>Philadelphia</name>
    <time>${timeAndDay.time}</time>
    <day>${timeAndDay.day}</day>
  </city>
  <city>
    <name>Boston</name>
    <time>${timeAndDay.time}</time>
    <day>${timeAndDay.day}</day>
  </city>
</cities>
```

Parsing and Displaying XML Output: Results





Ajax Tools

Tools and Toolkits

- **Client-Side Tools**
  - Dojo.
    - Open source JavaScript toolkit with Ajax support
    - http://www.dojotoolkit.org/
  - Google Web Toolkit
    - Write code in Java, translate it to JavaScript
    - http://code.google.com/webtoolkit/
  - script.aculo.us
    - Free JavaScript toolkit with Ajax support
    - http://script.aculo.us/
  - Yahoo User Interface Library (YUI)
    - Free JavaScript toolkit with some Ajax support
    - http://developer.yahoo.com/yui/

Tools and Toolkits (Continued)

- **Server-Side Tools**
  - Direct Web Remoting
    - Lets you call Java methods semi-directly from JavaScript
    - http://getahead.ltd.uk/dwr/
  - JSON/JSON-RPC
    - For sending data to/from JavaScript with less parsing
    - http://www.json.org/
    - http://json-rpc.org/
  - JSP custom tag libraries
    - Create tags that generate into HTML and JavaScript
    - http://courses.coreservlets.com/Course-Materials/msajsp.html

Tools and Toolkits (Continued)

- **Hybrid Client/Server Tools**
  - AjaxTags (built on top of script.aculo.us)
    - JSP custom tags that generate Ajax functionality
      - Supports many powerful Ajax capabilities with very simple syntax
      - http://ajaxtags.sourceforge.net
  - JavaServer Faces (JSF) component libraries
    - Trinidad (formerly Oracle ADF)
      - http://www.oracle.com/technology/products/jdev/htdocs/partners/addins/exchange/jsf/ (also myfaces.apache.org)
    - Tomahawk
      - http://myfaces.apache.org/tomahawk/
    - Ajax4JSF
      - http://labs.jboss.com/jbossajax4jsf/
    - IceFaces
      - http://www.icefaces.org/
    - Build your own
      - http://courses.coreservlets.com/Course-Materials/jsf.html

Books

- **Foundations of Ajax**
  - Asleson and Schutta. APress.
  - Geared around Java on the server-side.
- **Ajax in Action**
  - Crane, Pascarello, and James. Manning.
  - Geared around Java on the server-side.
- **Pro JSF and Ajax**
  - Jacobi and Fallows. APress.
  - Geared around JavaServer Faces integration.
- **Professional Ajax**
  - Zakas, et al. Wrox. *Wait for 2nd edition.*
  - Geared around Java on the server-side.

Summary

- **JavaScript**
  - Define request object
    - Check for both Microsoft and non-MS objects. Identical in all apps.
  - Initiate request
    - Get request object
    - Designate a request handler function
    - Initiate a GET or POST request
    - Send data (null for GET)
  - Handle response
    - Wait for readyState of 4 and HTTP status of 200
    - Extract return text with responseText or responseXML
    - Do something with result
      - Use innerHTML to insert result into "div" element
- **HTML**
  - Give ids to input elements and to div. Initiate process.
- **Java**
  - Use JSP, servlet, or combination (MVC) as appropriate.