Ajax: The Basics Part I

Originals of Slides and Source Code for Examples:
http://courses.coreservlets.com/Course-Materials/ajax.html

© 2008 Marty Hall

For live Ajax & GWT training, see training courses at http://courses.coreservlets.com/.

Taught by the author of *Core Servlets and JSP*, *More Servlets and JSP*, and this tutorial. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
  - Java 5, Java 6, intermediate/beginning servlets/JSP, advanced servlets/JSP, Struts, JSF, Ajax, GWT, custom mix of topics
- Courses developed and taught by coreservlets.com experts (edited by Marty)
  - Spring, Hibernate, EJB3, Ruby/Rails

Contact hall@coreservlets.com for details

Topics in This Section

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

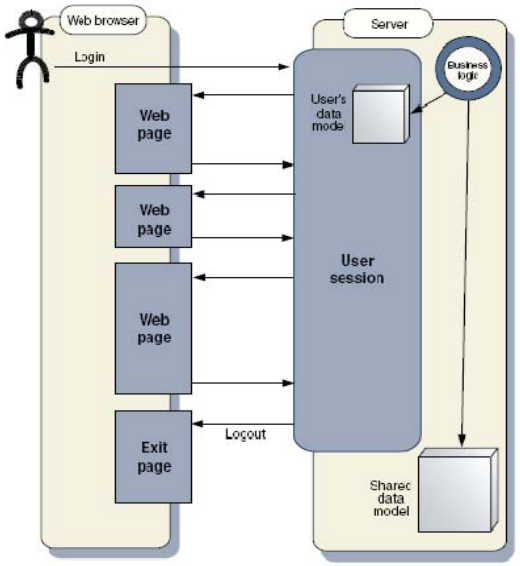
Motivation

Why Ajax?

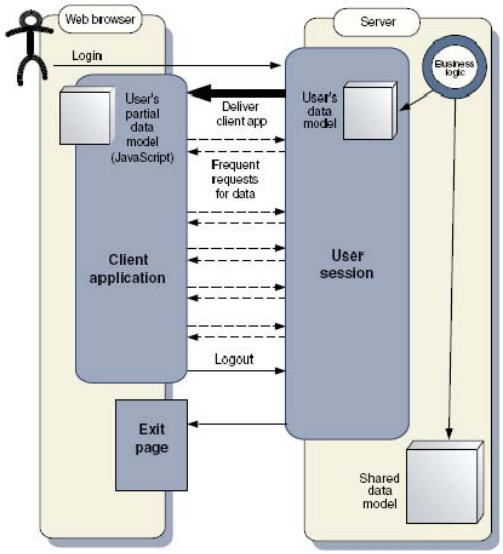
- **HTML and HTTP are weak**
  - Non-interactive
  - Coarse-grained updates
- **Everyone wants to use a browser**
  - Not an application that requires SW installation
- **"Real" browser-based active content**
  - Failed: Java Applets
    - Not universally supported; can't interact with the HTML
  - Serious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

Traditional Web Apps vs. Ajax Apps

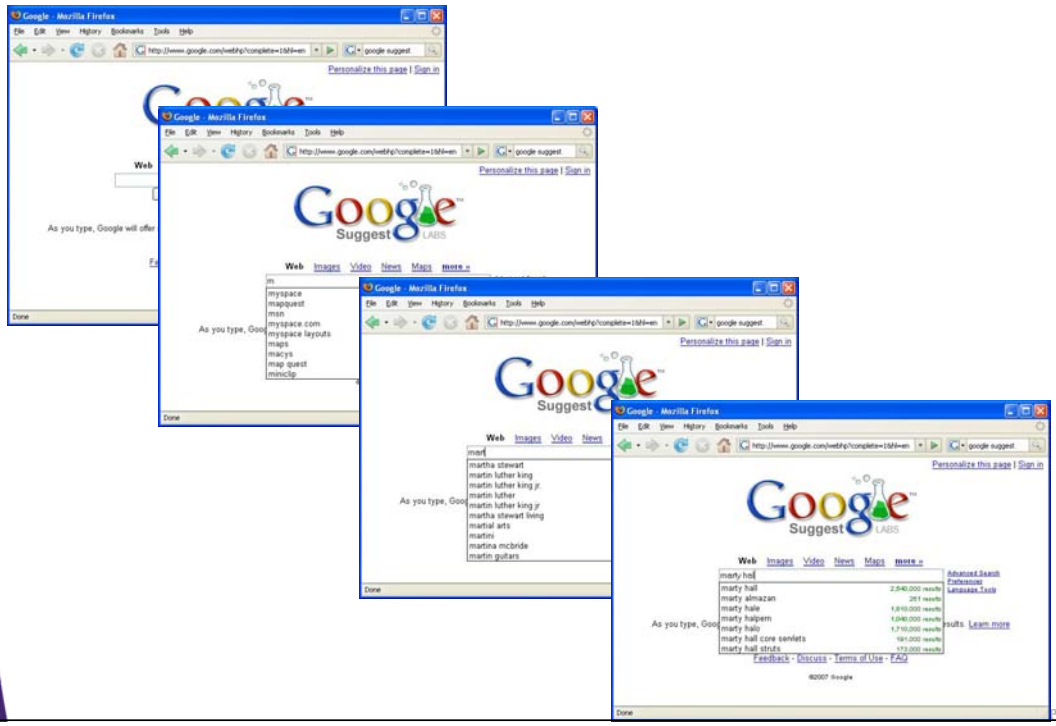
Infrequent large updates



Frequent small updates



Google Suggest (http://labs.google.com/suggest/)

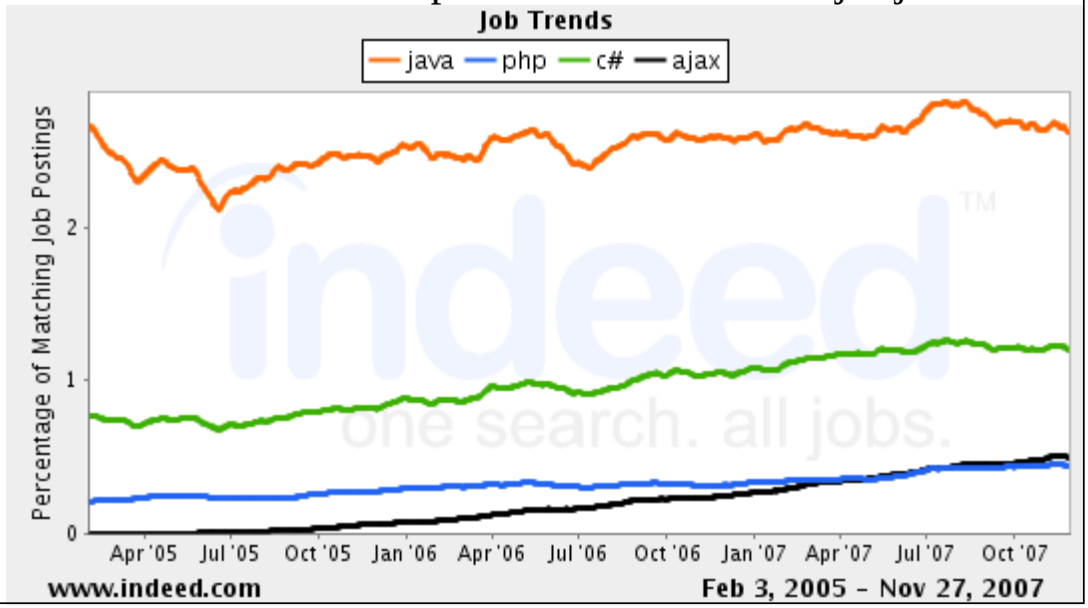


More Ajax Examples

- http://maps.google.com/
  - http://blog.grimpoteuthis.org/2005/02/mapping-google.html (analysis of Ajax approach)
- http://demo.nextapp.com/InteractiveTest/ia
- http://demo.backbase.com/explorer/
- http://java.samples.infragistics.com/NetAdvantage/JSF/2007.2/featurebrowser/fbhome.jsp
- http://www.laszlosystems.com/demos/
- http://www.smartclient.com/index.jsp#_Welcome
- http://www.simplica.com/ajax/example/ajax_example.htm?ap=ga3

Ajax Jobs (as of Nov 2007)

- **From indeed.com**
  - Indeed claims to compile data from most major job sites



The Basic Process

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

Define a Request Object

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

Version for Internet Explorer 5 and 6

Version for Firefox, Netscape 5+, Opera, Safari, Mozilla, and Internet Explorer 7

Fails on older and nonstandard browsers

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)

Handle Response

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

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

Text of server response

Pop up dialog box

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 not anonymous functions.
  - Anonymous functions (also called closures) are widely used in Lisp, Ruby, Scheme, C# (as of 2.0), ML, and other languages.

Common but Incorrect Approach (Global Request Variable)

```
var request;

function getRequestObject() { ... }

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

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

  - This is the approach shown in *Foundations of Ajax*, *Ajax in Practice*, *Ajax in Action*, and *JavaScript the Definitive Guide*

Problem with Common Approach: Race Conditions!

- **Scenario**
  - Two xhtml buttons, the first calling function1 and the second calling function2
  - function1 takes 5 seconds to get result from server
  - function2 takes 1 second to get result from server
- **Problem**
  - Suppose user presses button1, then one second later presses button2.
    - When function1 looks for request.responseText, it gets the response text of function 2!
    - The function you supply to onreadystatechange must take zero arguments, so you cannot use a normal (named) function.
- **Solution**
  - Use an anonymous function with a *local* copy of the request object embedded inside the code.

Corrected Approach (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) {
  ...
}
```

Complete JavaScript Code (show-message.js)

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

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"?> ← Omit this!
      <!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 .../> ← Don't do this.
    instead of <script...></script> ← Do this.

HTML Code (show-message.html)

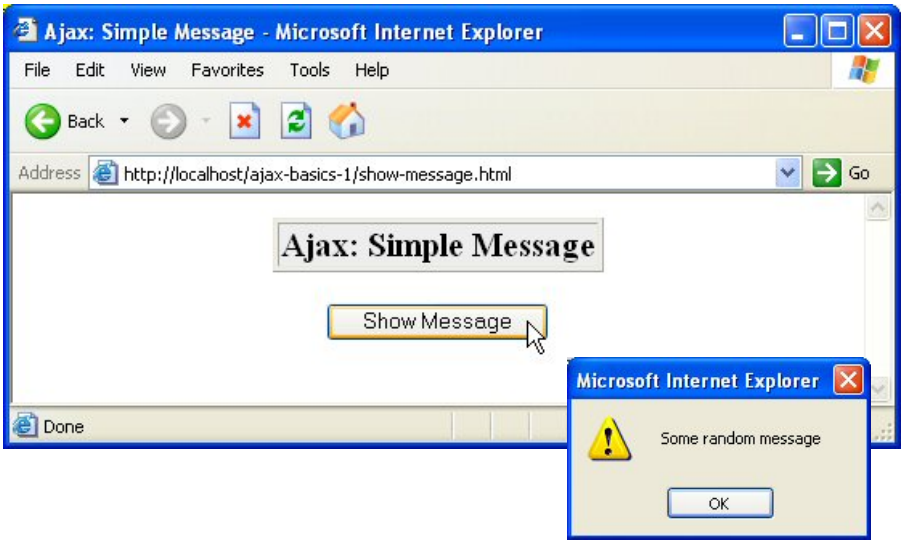
```
<!DOCTYPE html PUBLIC "..."
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head><title>Ajax: Simple Message</title>
<script src="show-message.js"
        type="text/javascript"></script>
</head>
<body>
<center>
<table border="1" bgcolor="gray">
  <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>
```

HTML Code (message-data.html)

```
Some random message
```

- **Note: executing this example**
  - Since main page uses relative URL and the HTML here has no dynamic content, you can run this example directly from the disk without using a server. But later examples require dynamic content, so all examples will be shown running on Tomcat.

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 an anonymous response 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**
  - 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.ActiveXObject) {
    return(new ActiveXObject("Microsoft.XMLHTTP"));
  } else if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } 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).

Complete JavaScript Code (Part of ajax-basics-1.js)

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

- **Load JavaScript from central location**

```
<script src="./scripts/ajax-basics.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

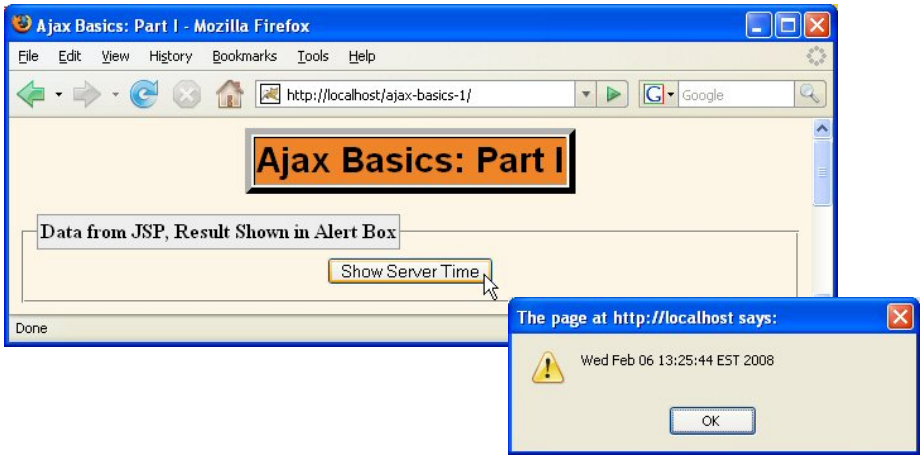
```
<!DOCTYPE html PUBLIC "...">
<html xmlns="http://www.w3.org/1999/xhtml">...
<link rel="stylesheet"
      href="./css/styles.css"
      type="text/css"/>
<script src="./scripts/ajax-basics.js"
        type="text/javascript"></script>...
<body>...
<fieldset>
  <legend>Data from JSP, Result Shown in 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() %>
```

- **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
    - Much more likely with IE than with other browsers
  - Solution: send Cache-Control and Pragma headers
- **Date was not formatted**
  - Just used the toString method of Date
  - Solution: use String.format (ala 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 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.ActiveXObject) {
    return(new ActiveXObject("Microsoft.XMLHTTP"));
  } else if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } 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-basics.js"
        type="text/javascript"></script>
...
<fieldset>
  <legend>
     Data from Servlet, Result Shown in Alert Box
  </legend>
  <form action="#">
    <input type="button" value="Show Server Time"
           onclick='ajaxAlert("show-time")'/>
  </form>
</fieldset>
...
```

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

Servlet Code

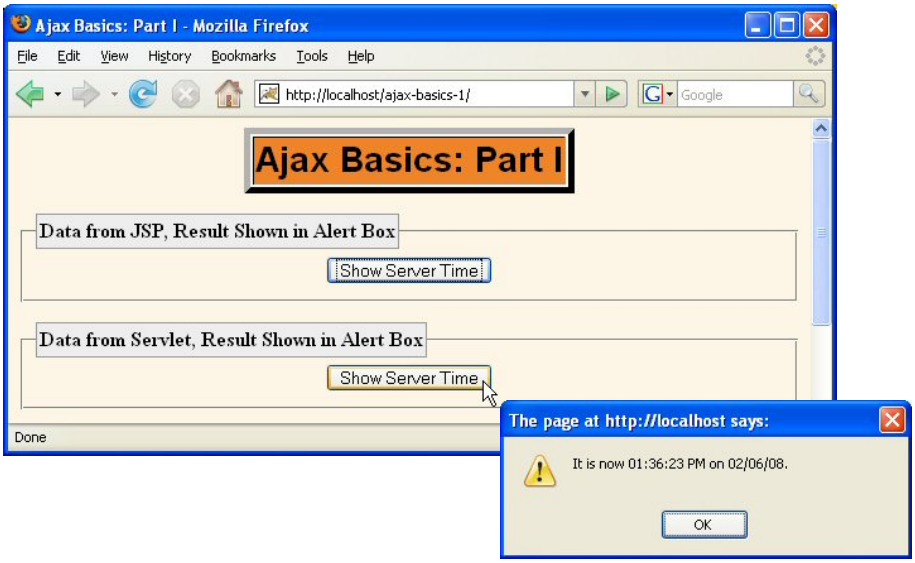
```
package coreservlets;
import ...

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");
    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



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
        - Note that Firebug has an outstanding DOM explorer. See next lecture.
    - 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

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 based on request.responseText
- **Result after running code**
  - <h2>Wow!</h2>
    - "View source" won't show this, but Firebug will.
- **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.ActiveXObject) {
    return(new ActiveXObject("Microsoft.XMLHTTP"));
  } else if (window.XMLHttpRequest) {
    return(new XMLHttpRequest());
  } else {
    return(null);
  }
}
```

No changes from previous example

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


HTML Code

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

Style Sheet Code (css/styles.css)

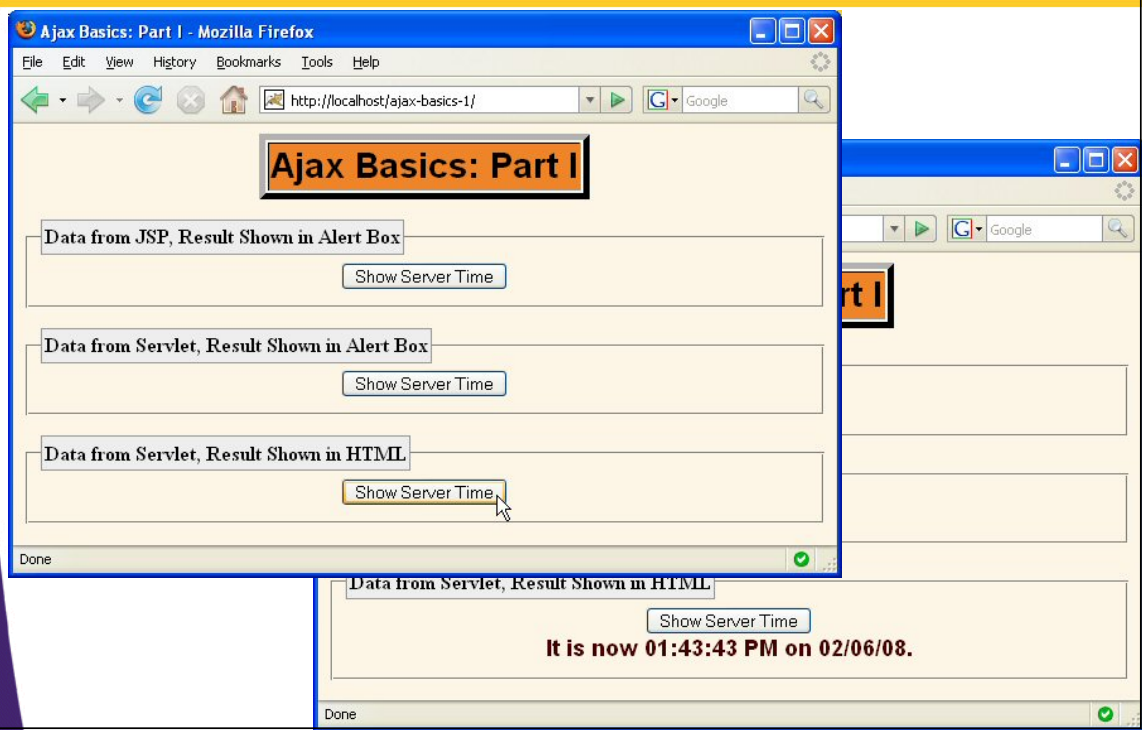
```
.ajaxResult { color: #440000;
              font-weight: bold;
              font-size: 18px;
              font-family: Arial, Helvetica, sans-serif;
}
```

- **Note**
  - Don't worry if you don't yet know much about style sheets. They will be covered in later lecture.

Servlet Code

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

Building HTML Output: Results



Wrapup

Preview of Next Sections

- **Ajax Basics: Part II**
  - Sending GET data
  - Reading data from textfields
  - Sending POST data
  - Ajax toolkits
- **Ajax Development and Testing Tools**
  - Tools for debugging Ajax
  - Tools for debugging JavaScript
  - Tools for building Ajax-based Web apps
  - Tools for developing xhtml
  - Tools for building and previewing style sheets
  - Tools for validating xhtml

Summary

- **JavaScript**
  - Define request object
    - Check for both Microsoft and non-MS objects. Identical in all apps.
  - Initiate request
    - Get request object
    - Designate an anonymous response handler function
    - Initiate a GET request
  - 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.
- **Java**
  - Use JSP, servlet, or combination (MVC) as appropriate.
  - Prevent browser caching.

Questions?