jQuery Ajax Support: Basics

For additional materials, please see http://www.coreservlets.com/. The JavaScript tutorial section contains complete source code for all examples in the entire tutorial series, plus exercises and exercise solutions for each topic.

For customized training related to JavaScript or Java, email hall@coreservlets.com. Marty is also available for consulting and development support.

Taught by lead author of Core Servlets & JSP, co-author of Core JSF (4th Ed), and this tutorial.

Available at public venues, or custom versions can be held on-site at your organization.

Courses developed and taught by Marty Hall
- JavaScript, HTML5, Ext JS, PrimeFaces, Java 8 programming, Spring Framework, Spring MVC, Android, GWT, custom mix of topics
- Courses available in any state or country.
- Maryland/DC companies can also choose afternoon/evening courses.

Courses developed and taught by coreservlets.com experts (edited by Marty)
- Hadoop, Hibernate/JPA, HTML5, RESTful Web Services

Contact hall@coreservlets.com for details.
Topics in This Section

- **Motivation**
  - Web apps in general
  - Ajax in particular

- **Data-centric Ajax: the $.ajax function**
  - Basics
  - Practicing interactively
  - Details and simple options
  - A complete example
  - Unobtrusive JavaScript
  - Inserting results into the page

- **Content-centric Ajax: the load function**
  - Retrieving and inserting in one fell swoop

For additional materials, please see http://www.coreservlets.com/. The JavaScript tutorial section contains complete source code for all examples in the entire tutorial series, plus exercises and exercise solutions for each topic.
## Ajax Overview

- **Acronym**
  - Asynchronous
  - JavaScript
  - And
  - XML
  - Jesse James Garrett first coined the term “AJAX”, but did not invent the technology

- **Now usually written as “Ajax”, not “AJAX”**
  - Meaning “the way JavaScript is used to talk to the server, get some result, and update the page”
    - Not always or even usually XML: data is also commonly sent as JSON, simple strings, or pieces of HTML. JSON or simple strings are often preferred in modern apps.
    - Not always asynchronous. Some apps queue up requests and process them one at a time. JSF does this by default.

## Why Web Apps?

- **Downsides to browser-based apps**
  - GUI is poor
    - HTML is OK for static documents, but lousy for programs
  - Communication is inefficient
    - HTTP is poor protocol for the way we now use Web apps
Why Web Apps? (Continued)

• So why does everyone want Web apps?
  – Universal access
    • Everyone already has a browser installed
    • Any computer on the network can access content
  – Automatic “updates”
    • Content comes from server, so is never out of date

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)

• Still browser based
  – Ajax is about “what is the best you can do with what is already in browsers?”

• “Real” browser-based active content
  – Failed: Java Applets
    • Not universally supported; can’t interact well with the HTML
  – Serious alternative: Flash/Flex
    • Not preinstalled on all PCs; not available for iPhone/iPad
  – Less widely installed
    • Microsoft Silverlight
    • JavaFX
Traditional Web Apps vs. Ajax Apps

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

- **Ajax Apps:**
  - Frequent Small Updates

**Web Page 1.**

**Web Page 2.**

**Google Home Page**
Ajax Jobs

Data for US jobs, averaged over all states. From indeed.com as of 11/2014

Average Salary of Jobs with Titles Matching Your Search

<table>
<thead>
<tr>
<th>Job</th>
<th>Average Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>java</td>
<td>$102,000</td>
</tr>
<tr>
<td>.net</td>
<td>$91,000</td>
</tr>
<tr>
<td>javascript</td>
<td>$99,000</td>
</tr>
<tr>
<td>php</td>
<td>$88,000</td>
</tr>
<tr>
<td>c#</td>
<td>$94,000</td>
</tr>
<tr>
<td>ajax</td>
<td>$94,000</td>
</tr>
</tbody>
</table>

In USD as of Nov 10, 2014
$.ajax: Basics

Three Styles of Ajax

- **Data-centric Ajax**
  - The server sends raw data to the client, often in JSON format
  - The client parses the data and builds HTML based on it
  - You generally use the $.ajax function when you need to examine server data or build custom HTML on the client

- **Content-centric Ajax**
  - The server sends the exact string or HTML to be displayed
  - The client simply inserts the content into the page
    - But style sheets can somewhat customize the look
  - You generally use the load function if you will always insert the server data unchanged

- **Script-centric Ajax**
  - The server sends JavaScript functions to the client
  - The client executes the functions
  - This approach is not discussed in this tutorial
    - Inflexible: Requires the server to know too much about client
$.ajax – Calling to Server with jQuery

$.ajax({url: "address", success: handlerFunction});

- **Interpretation**
  - Make network connection to server
  - Access the resource at the relative URL given by address
  - When the process returns successfully, call handlerFunction and pass it the text (contents) of the resource that was given by address

- **Notes**
  - You need a dot after the $  
    - It is $.ajax, not $ajax
  - You need to surround the options in curly braces  
    - It is $.ajax({url: ..., success: ...}), not $.ajax(url: ..., success: ...)

Reminder: Deploying Apps to Tomcat

- **$.ajax only works on apps running on a Web server**
  - It will not work if you simply load a local HTML file in a browser, since there is no relative URL to connect to  
    - Last section showed how to deploy to Apache Tomcat using Eclipse, but if you know how to deploy to any Web server already, you can use that instead

- **Start by deploying app containing jquery-test.html**
  - This was the HTML page from the earlier section on Getting Started with jQuery – it simply loads jQuery but does not perform any jQuery calls

- **Steps in Eclipse (review from previous section)**
  - Create new Dynamic Web Project named jquery-ajax-1
  - Put all code under the WebContent folder (jquery-test.html directly in WebContent and the CSS and JavaScript files in the css and scripts folders)
  - Deploy app by R-clicking Tomcat, Add and Remove, add jquery-ajax-1
  - Start server by R-clicking Tomcat, Start
    - Or http://localhost:8080/jquery-ajax-1/jquery-test.html if you did not change Tomcat port
server-result.txt is a file in the WebContent folder that contains the single line “This is the result from the server”. Test your relative URLs in a browser before calling them in an Ajax request.

For additional materials, please see http://www.coreservlets.com/. The JavaScript tutorial section contains complete source code for all examples in the entire tutorial series, plus exercises and exercise solutions for each topic.
$.ajax: More Details

- **$.ajax(optionsObject)**
  - Minimal form:
    
    ```javascript
    $.ajax({url: "address", success: handlerFunction});
    ```
  - The handler function gets the response text, not the response object.
    - jQuery guesses if it should be plain text, XML, or JSON from the response type. If you want to enforce that handler gets given type, use dataType option shown later.

- **Options for $.ajax({...})**
  - Almost-always used
    - url, success
  - Other common options
    - cache, data, dataType, error, type, username, password

- **Example**

  ```javascript
  $.ajax({url: "address", success: funct1, error: funct2});
  ```

Simple Form of Response Handlers

- **Function takes one argument (the text of the server resource)**

  ```javascript
  function someHandler(text) { ... }
  ```
  - Note that it gets the response text, not the response object
  - And, “text” can be XML or JSON, depending on dataType option

- **Quick example**

  ```javascript
  function popupResult(serverText) {
    alert("Data from server is " + serverText);
  }
  
  $.ajax({ url: "someAddress", success: popupResult });
  ```
More Complex Handlers

- **Full (optional) format: function takes three arguments**
  
  ```javascript
  function someHandler(text, status, request) { ... }
  ```
  
  - **text**
    - Response data from server
  
  - **status**
    - String describing the status: "success" or "notmodified". Rarely useful. In error handlers, the status string is more meaningful.
  
  - **request**
    - The XMLHttpRequest object (augmented jQuery version).

- **You can supply multiple success handlers**
  
  ```javascript
  success: [handler1, handler2, ...] (array of functions)
  ```

Basic Ajax with Raw JavaScript

- **(Handler gets response object)**
  
  ```javascript
  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() { someFunct(request); };
      request.open("GET", "some-url", true);
      request.send(null);
  }
  ```
Basic Ajax with jQuery

• (Handler gets response text)

$.ajax({url: "address", success: handlerFunct});
**Pieces Needed**

- **Some resource on the server**
  - We will use JSP here
- **The main jQuery file**
  - Defines $.ajax, etc.
- **Our own JavaScript file**
  - Defines function to be called when button is clicked.
    - This function calls $.ajax.
  - Defines function to be called when the Ajax request returns successfully.
    - This function is specified with “success” in the $.ajax call from the first function
- **An HTML file**
  - Loads jQuery and our own JavaScript file
  - Defines a button that, when clicked, calls the first function above
    - i.e., the function that calls $.ajax

**$.ajax Example Code: Server Resource (JSP)**

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

The example could easily be done with the simple text file (server-result.txt) used in the previous interactive example. The only advantage of using JSP to produce the server response here is that the output changes every second, so if you click the button multiple times, you can verify that you see a different result each time.
$.ajax Example Code: JavaScript

```javascript
function showTime1() {
    $.ajax({ url: "server-time.jsp",
             success: showAlert });
}

function showAlert(text) {
    alert(text);
}
```

$.ajax Example Code: HTML

```html
...<head><title>jQuery and Ajax</title>...
<script src="scripts/jquery.js"></script>
<script src="scripts/jquery-ajax-1.js"></script>
</head>
<body>
...
<fieldset>
    <legend>$.ajax: Basics (Using onclick handler in HTML)</legend>
    <input type="button" value="Show Time"
           onclick='showTime1()' />
</fieldset>
```
$.ajax: Results

function showTime1() {
    $.ajax(
        { url: "server-time.jsp",
          success: showAlert,
          cache: "false" });
}

function showAlert(text) {
    alert(text);
}

If the request uses GET, browsers are permitted (depending on end user browser settings) to cache the results locally. So, a later button click could, on some browsers, show the same result as an earlier click because the server is not contacted again. In many situations, caching is good: if you send the same data you get the same result, and caching would speed things up. But in this case, no data is sent, so for the exact same URL, the result is different each time. So, you should prevent caching.

An alternative would be to send the request with POST instead of GET, since browsers do not cache the results of POST requests. We will see how to do this in a later section.
Unobtrusive JavaScript

Registering Event Handlers in JavaScript

• Idea
  – Previous example set the onclick handler in the HTML. Although this is common with some Ajax libraries, jQuery advocates setting it in the JavaScript instead
    • Often referred to as “unobtrusive JavaScript”: no explicit JavaScript anywhere in the HTML page

• jQuery syntax
  $(function() {...});
    • Function runs after the DOM is loaded, but does not wait for images, as with window.onload in raw JavaScript
    • Use this approach to set up all event handlers

  $("#some-id").click(someHandler);
    • Assigns to onclick handler. Handler is passed an Event object with characteristics that are unified across browsers
<fieldset>
<legend>$.ajax: Basics (Using click function in JavaScript)</legend>
<input type="button" value="Show Time" id='time-button-1'/>
</fieldset>

$(function() {
    $('#time-button-1').click(showTime1);
});

function showTime1() {
    $.ajax({ url: "show-time.jsp",
        success: showAlert,
        cache: false });
}

function showAlert(text) {
    alert(text);
}
Redoing Time Alert: Results

Works exactly the same as previous example.

Inserting Ajax Results into Page

For additional materials, please see http://www.coreservlets.com/. The JavaScript tutorial section contains complete source code for all examples in the entire tutorial series, plus exercises and exercise solutions for each topic.
Quick Reminder: the html() Function

\$("some-css-selector")\html("some-string")

- Insert “some-string” as the body content of all HTML elements that match the pattern

- **Most common usage with Ajax**
  \$\('#id-of-result-area\')\html(textFromServer);
  \$\('#id-of-result-area\')\html(textBasedOnServerResult);

---

Big Idea

- **Popping up alert boxes not common**
  - Usually reserved for errors, and even then, you typically use styled dialog boxes as will be discussed in upcoming section on jQuery UI.
    - We did it just to illustrate the basics of $.ajax

- **Better to insert results into page**
  - Start with empty element that has an id
    - `<div id="target"></div>`, `<span id="target"></span>`, etc.
  - Get data from server
    - For now we will use simple string, but later we will parse JSON or other data. *We will see shortcut in next section (the load function) for inserting results unchanged.*
  - Insert it into page
    - \$\('#target\').html(dataFromServer)
Template Code

```javascript
function insertAjaxResult(address, resultRegion) {
    $.ajax({
        url: address,
        success: function(text) {
            insertText(text, resultRegion);
        }
    });
}

function insertText(text, resultRegion) {
    $(resultRegion).html(text);
}
```

In real life, you would do something to the text before inserting it, otherwise you would use the simpler "load" function. Later sections will cover the load function as well as parsing data and building results strings.

Redoing Time Alert: HTML

```
<fieldset>
    <legend>$.ajax: Inserting Result in Page</legend>
    <input type="button" value="Show Time"
         id="time-button-2"/>
    <h2 id="time-result"></h2>
</fieldset>
```
Redoing Time Alert: JavaScript

$(function() {
    ...
    "#time-button-2".click(showTime2);
});

function showTime2() {
    insertAjaxResult("show-time.jsp",
                    "#time-result");
}

Redoing Time Alert: Results
Content-Centric Ajax: The load Function

- **Big idea**
  - Very common to take server text and insert it into the page unchanged; so, there is a built-in jQuery function that does this process directly
  - Inserting server content unchanged is sometimes called “content-centric Ajax”, as opposed to “data-centric Ajax” where you get data from the server, parse it, build a string based on that server data, and then insert that string

- **Three forms**
  - Simplest form: retrieve contents of url from server and insert that text into result area
    
    $("#result-area-id").load("url");
    
    - Most common form in real life: pass data to the url, retrieve the contents, and insert that text into result area
    
    $("#result-area-id").load("url", data);
    
    - Pass data to the url, retrieve the contents, insert that text into result area, and then call an additional handler function to do more tasks. See next slide.
    
    $("#result-area-id").load("url", data, handlerFunction);
Why Handler Function for load?

- **The main handler function is built in**
  - Take text that is returned from server and insert it into the innerHTML property of the matching element(s)

- **So why allow a handler function as third argument?**
  - Answer: to do extra side effects
    - Highlight the result
    - Turn off a “waiting for data…” message
  - Alternative: often better to just use $.ajax instead of load with extra handler function

- **Reminder: call is asynchronous**
  - This fails for “waiting for data…” message
    \[
    \$(\"#message-div\").show();
    \$(\"#result-div\").load("some-url");
    \$(\"#message-div\").hide();
    \]

---

load Example: JavaScript

\[
\$(function() {
    ...
    \$(\"#time-button-3\").click(showTime3);
});
\]

function showTime3() {
    \$(\"#time-result-2\").load("server-time.jsp");
}
load Example: HTML

...<fieldset>
<legend>load: Inserting Result in Page</legend>
  <input type="button" value="Show Time"
        id="time-button-3"/>
  <h2 id="time-result-2"></h2>
</fieldset>

load Example: Results

It is now Tue Oct 27 15:07:09 EDT 2015
Wrap-Up

Summary

• General Ajax requests (data-centric Ajax)
  $.ajax({ url: "relative-address",
           success: handlerFunction });

• Assigning event handlers programmatically
  $(function() {
    $('#button-id').click(someFunction);
  });

• Inserting data into page explicitly
  $('#result-id').html("some string");

• Calling Ajax and inserting results, all in one fell swoop
  $('#result-id').load('relative-address');
Questions?

For additional materials, please see http://www.coreservlets.com/. The JavaScript tutorial section contains complete source code for all examples in the entire tutorial series, plus exercises and exercise solutions for each topic.