

AJAX with ASP.NET MVC

`$.ajax({...})`

K. Scott Allen



Overview

AJAX
Helpers

jQuery
UI

Client
Validation

JSON
Actions

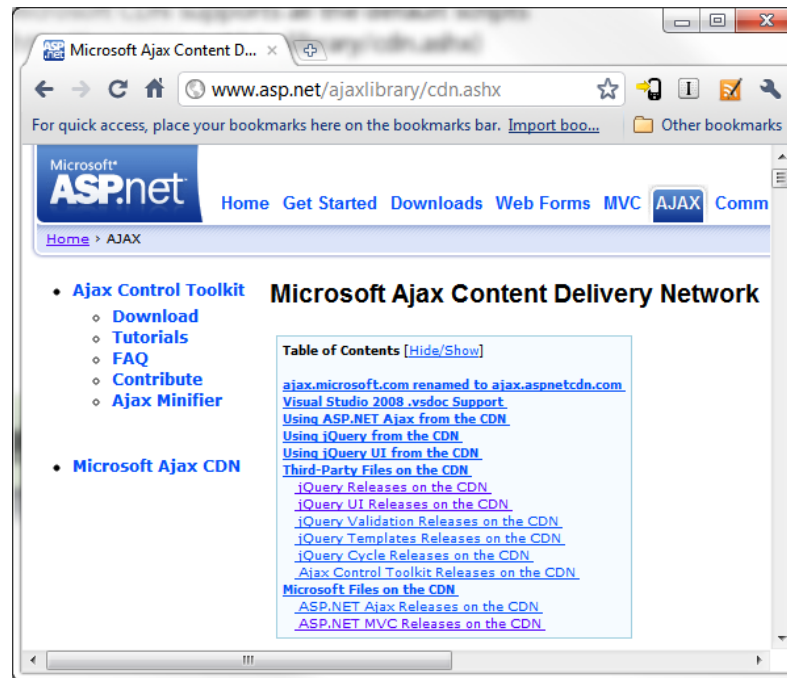
jQuery
Templates

AJAX Infrastructure

- **jQuery**
 - Functional orientation
 - Plug-in oriented
 - CSS Selectors
 - DOM Manipulation

Managing Scripts

- Where do I place script tags?
 - Inside the <head> or at the bottom of a page?
- Scripts and content delivery networks
 - Use them for faster load times
 - Microsoft CDN supports all the default scripts (<http://www.asp.net/ajaxlibrary/cdn.ashx>)



Ajax Helpers

- Create links and forms that send async requests

```
<div id="timeDisplay">
    <%= Ajax.ActionLink("Click here to set the server time",
        "ServerTime",
        new AjaxOptions
        {
            UpdateTargetId="timeDisplay",
            HttpMethod="GET", // default
            InsertionMode= InsertionMode.Replace, // default
        })
    %>
</div>
```

Unobtrusive JavaScript

- **No script injected into page**
 - Only data- attributes with necessary AJAX settings
- **Requires unobtrusive extensions script**
 - jquery.unobtrusive-ajax.js (AJAX helpers)
 - jquery.validate.unobtrusive.js (Client validation)

```
<div id="latestReview">  
  <a data-ajax="true" data-ajax-loading="#progress" data-ajax-method="GET"  
    data-ajax-mode="replace" data-ajax-update="#latestReview"  
    href="/Home/LatestReview">Click here to see the latest review</a>  
</div>
```

AjaxOptions

- Includes events for
 - OnBegin
 - OnComplete
 - OnFailure
 - OnSuccess
- Confirmation prompt

```
<%= Ajax.ActionLink("Click here to set the server time",  
    "ServerTime",  
    new AjaxOptions  
    {  
        LoadingElementId="loadingDisplay",  
        Confirm="Are you sure?",  
        UpdateTargetId="timeDisplay",  
    })
```

```
%>
```

```
<div id="loadingDisplay" style="display:none">  
      
</div>
```

Partial Page Rendering

- Use PartialViewResults in controller actions

```
if (Request.IsAjaxRequest())  
{  
    return PartialView("MovieTable", movies);  
}  
  
return View("Index", movies);
```

```
<% using (Ajax.BeginForm("Index", "Movie",  
    new AjaxOptions { HttpMethod="GET",  
        UpdateTargetId="movieTable" }))  
    { %>  
        <input type="text" name="q" />  
        <input type="submit" value="Search" />  
    <% } %>
```


Ajax Helpers and Errors

- **Default behavior is to fail silently**
- **Override default by specifying OnFailure option**

```
function searchFailed(ajaxContext) {
    var response = ajaxContext.get_response();
    var element = ajaxContext.get_updateTarget();
    element.innerHTML = "Error: server returned a " +
        response.get_statusCode();
}
```

Client Validation

- Based on model annotations ([Required])
- Requires jQuery validation plug-in (provided)
- Enabled by default in web.config
- Completely unobtrusive!
 - HTML "data-" attributes

```
<%
```

```
    Html.EnableClientValidation(false);
```

```
%>
```

Custom Client Validations

- 1. Implement `IClientValidatable`
 - In attributes or in self-validating models
- 2. Write a jQuery validation adapter
 - Maps metadata to validation rules

```
public IEnumerable<ModelClientValidationRule>
    GetClientValidationRules(
        ModelMetadata metadata,
        ControllerContext context)
{
    var rule = new ModelClientValidationRule(
        rule.ErrorMessage = FormatErrorMessage(me
        rule.ValidationParameters["maxwords"] = M
        rule.ValidationType = "wordcount";
    yield return rule;
}
```

```
jQuery.validator.unobtrusive.adapters.addSingleVal(
    "wordcount", "maxwords");

jQuery.validator.addMethod("wordcount",
    function (value, element, maxWords) {
        if (value) {
            var wordCount = value.split(' ').length;
            if (wordCount <= maxWords) {
                return true;
            }
            return false;
        }
        return true;
    });
```

Remote Validation

- Because some validations are impossible on the client-side
 - Require a database lookup, for example

```
[Remote("UsernameCheck", ErrorMessage="Invalid username")]  
public string Username { get; set; }
```

Beyond the Built-in Helpers

- **Ajax Helpers cover simple scenarios**
 - Replacing HTML content
 - Partial page rendering
- **Other scenarios require some JavaScript coding**
 - Auto-complete textboxes
 - Client-side validation
 - Invoking JSON services and actions
 - Animations

Auto-Complete

- jQuery auto-complete plug-in
 - Included in jQuery UI script

```
$(function() {  
    $("#searchBox").autocomplete("/Movie/SearchCandidates",  
                                { minChars: 3 });  
});
```

```
public string SearchCandidates(string q, int limit)  
{  
    var ctx = new MoviesContext();  
    var movies = ctx.MovieSet  
        .Where(m => m.Title.StartsWith(q))  
        .OrderByDescending(m => m.ReleaseDate)  
        .Take(limit)  
        .Select(m => m.Title);  
  
    return String.Join(Environment.NewLine, movies.ToArray());  
}
```

Date Pickers and Other Widgets

- Many lightweight jQuery widgets available from jQuery UI
 - <http://jqueryui.com/>
 - Calendar
 - Accordion
 - Slider
 - Dialog
 - Tab
 - Progress Bar

```
$(".create #ReleaseDate").datepicker();
```

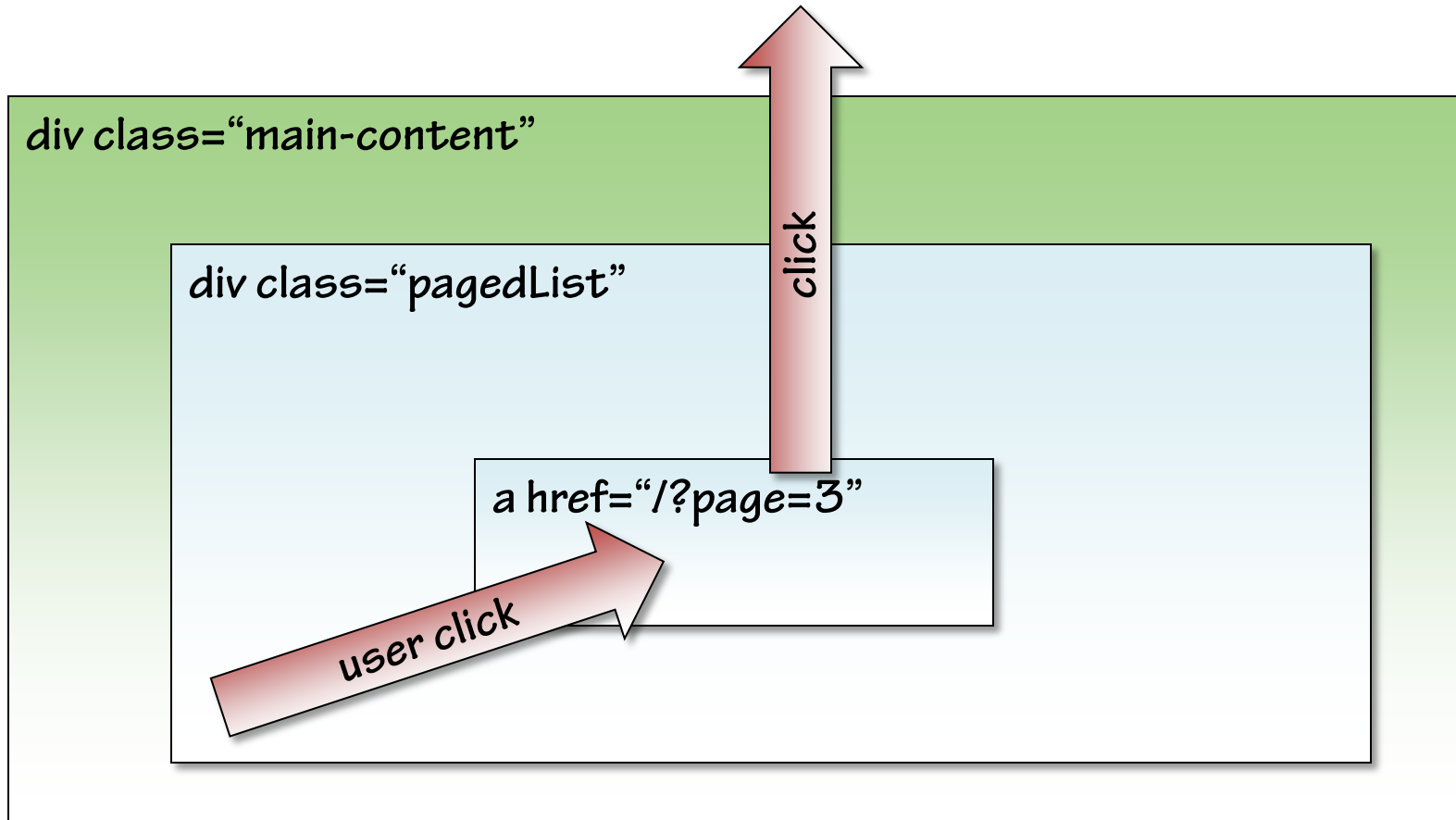
JSON & MVC

- JsonResult will serialize objects to JSON
- jQuery and ASP.NET AJAX can work with JSON

```
$.getJSON("/Instructor/InstructorNames", "", function(data) {  
    $(data).each(function() {  
        ...  
    });  
});
```

```
public JsonResult InstructorNames()  
{  
    var repository = new InstructorRepository();  
    var names =  
        from i in repository.FindAll()  
        select new  
        {  
            ID = i.ID,  
            Name = i.Name  
        };  
    return Json(names);  
}
```


jQuery .on relies on events bubbling up the DOM through parent elements



Summary



AJAX
Helpers

jQuery
UI

jQuery

JSON
Actions

Searching
Paging