jQuery Tips and Tricks

Dan Wahlin
http://weblogs.asp.net/dwahlin
@DanWahlin

Elijah Manor
http://elijahmanor.com
@ElijahManor

pluralsight
hardcore developer training

jQuery Fundamentals

This course will guide you through the features of the jQuery "write less, do more" library

Authored by: Dan Wahlin
Duration: 5h 32m
Level: Beginner

Fixing Common jQuery Bugs

In this course we will examine common bugs that are accidentally introduced when developing with jQuery.

Authored by: Elijah Manor
Duration: 2h 7m
Level: Intermediate
Released: 6/28/2013

jQuery Advanced Topics

A deep look at several advanced concepts in jQuery from performance to plugins to promises

Authored by: Joe Eames
Duration: 3h 3m
Level: Advanced

DOM Tips and Tricks

Event Tips and Tricks

Ajax and Data Tips and Tricks

Utility Tips and Tricks

DOM Tips and Tricks

Using a CDN with a Fallback

Working with Selectors

Limit DOM Interactions

Checking if an Element Exists

Using the end() Function with Chaining

filter() vs find()

Using Objects with Setter Methods

Use Classes over Styles

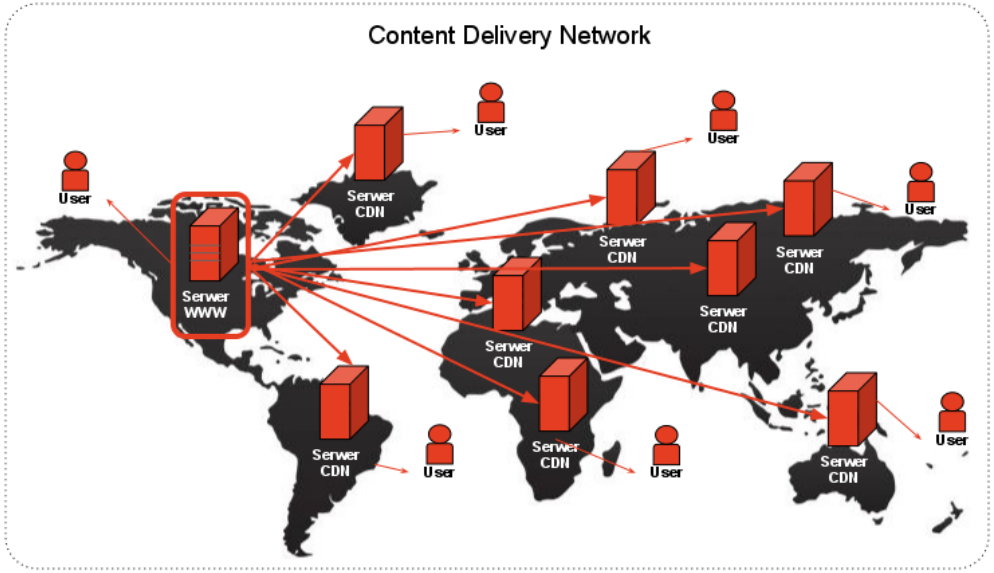
DOM

Using a CDN with a Fallback

pluralsight
hardcore developer training

Content Delivery Network (CDN) Benefits

- **Content Delivery Networks (CDNs) provide several benefits:**
  - Caching of scripts
  - Support for http and https
  - Regional servers – decreased latency
  - Allows for more concurrent calls (parallelism)



CDN Providers

- Google, Microsoft, jQuery and others provide CDNs that host scripts such as jQuery:

```
<script
  src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js">
</script>
```

No protocol defined

```
<script
  src="//ajax.aspnetcdn.com/ajax/jQuery/jquery-1.9.1.min.js">
</script>
```

Providing a CDN Fallback

- If there was a problem loading a script from a CDN you can provide a local fallback:

```
<script src="//CDNLocation.com/libs/jquery.min.js">
</script>

<script>window.jQuery || document.write(
  '<script src="js/jquery-version.min.js"><\/script>')
</script>
```

Fallbacks with AMD

```
require.config({
    enforceDefine: true,
    paths: {
        jquery: [
          "//CDN.com/libs/jquery/[version]/jquery.min.js",
           //Fallback to local script
          "lib/jquery"
        ]
    }
});

//Later
require(["jquery"], function ($) { ... });
```

DOM

Working with Selectors

pluralsight
hardcore developer training

Selector Caching

- Cache selectors to avoid re-querying the DOM:

Without Caching or Chaining

```
function processData() {
   $(".display").addClass("processed");
   $(".display").fadeIn(500);
}

function removeTitle() {
   $(".display").removeAttr("title");
}
```

With Caching & Chaining

```
var display = $(".display");

function processData() {
    display.addClass("processed")
           .fadeIn(500);
}

function removeTitle() {
    display.removeAttr("title");
}
```

Use $(this) to Access the Target Element

- Use $(this) in callback functions to access the target element:

Bad

```
$("#btn").click(function () {
    $("#btn").addClass("highlight");
});
```

Good

```
$("#btn").click(function () {
    $(this).addClass("highlight");
});
```

`this` is the raw DOM element

`$(this)` is the jQuery wrapped DOM element

Defining a Selector Context #1

```
<div id="emailContainer">
    <div class="panel">
        …
    </div>
    <div class="panel">
        …
    </div>
</div>
<div id="ordersContainer">
    <div class="panel">
        …
    </div>
    <div class="panel">
        …
    </div>
</div>
```

```
var emailDiv = $("#emailContainer");
```

How do you select nodes contained in emailDiv?

```
getPanels(emailDiv);

function getPanels(containerDiv) {
    var panels =
      $(".panel", containerDiv);
    ...
}
```

Context of Selector

Defining a Selector Context #2

```
<div id="emailContainer">
    <div class="panel">
        …
    </div>
    <div class="panel">
        …
    </div>
</div>
<div id="ordersContainer">
    <div class="panel">
        …
    </div>
    <div class="panel">
        …
    </div>
</div>
```

```
var emailDiv = $("#emailContainer");

getPanels(emailDiv);

function getPanels(containerDiv) {
    var panels =
      containerDiv.find(".panel");
    ...
}
```

Context of Selector

Simplifying Code with Custom Selectors

- What if you need to find all elements that use the Arial font?

```
var arialDivs = [];
$("div").each(function () {
    var div = $(this);
    if (div.css("font-family") === "Arial") {
        arialDivs.push(div);
    }
});
```

- Create a re-useable function or a custom jQuery selector if this code is called multiple times

Creating a Custom Psuedo-Class Selector

- jQuery supports the creation of custom selectors using $.expr:

Create Custom Selector

```
$.extend($.expr[":"], {
    hasArialFont: function (element) {
        return $(element).css("font-family") === "Arial";
    }
});
```

Use Custom Selector

```
$("div:hasArialFont").click(function () {
    alert("Element has Arial font");
});
```

DOM

Limit DOM Interactions

Limit DOM Interactions

- Avoid manipulating the DOM from within a loop:

```
var parentDiv = $("#emailList");
for (var i = 0; i < total; i++) {
    parentDiv.append("<div>" + i + "</div>");
}
```

Updates the DOM every time through the loop

Limit DOM Interaction #1

- Instead of manipulating the DOM in a loop use string concatenation:

Used to append HTML fragments as strings

```
var divs = "";
for (var i = 0; i < total; i++) {
    divs += "<div>" + i + "</div>";
}
$("#emailList").html(divs);
```

DOM only updated once

Limit DOM Interaction #2

- Instead of manipulating the DOM in a loop use an array:

Used to hold HTML fragments

```
var divs = [];
for (var i = 0; i < total; i++) {
    divs.push("<div>" + i + "</div>");
}
$("#emailList").html(divs.join(""));
```

Convert array into an HTML string

DOM

Checking if an Element Exists

pluralsight
hardcore developer training

Check if an Element Exists

- How do you check if an element already exists in the DOM?
- If a selector's length is greater than 0 then one or more elements exist:

```
var link = $("#mainLink");
if (link.length) { //element exists
    link.attr("title", "Pluralsight Courses");
}
else {
    alert("No elements found");
}
```

DOM

find() vs. filter()

pluralsight
hardcore developer training

find() vs. filter()

- When do you use find() instead of filter()?

```
<ul id="people">
    <li class="publisher">Elijah</li>
    <li class="publisher">Dan</li>
    <li class="owner">Fritz</li>
</ul>

var $names = $('#people li');

var $pubs = $names.find('.publisher');

var $pubs2 = $names.filter('.publisher');
```

Looks for descendants of `li`. No matches.

Filters existing `li` elements. 2 matches.

Using the end() Function with Chaining

Using the end() Function

- Chaining methods can cause the context to change
- Use the end() function to change the context

```
var cust = getCustomer(custID);

$('<div class="cust"><span /></div>')
    .find("span") //get to span
        .attr("title", cust.name)
        .data(cust)
        .html(cust.name)
        .click(showCust)
    .end() //get back to span's parent div
    .appendTo("#divContainer");
```

Context switches to `span` element

end() switches context to parent `div`

DOM

Using an Object with Setter Methods

Using an Object with Setter Methods

- When calling a setter method multiple times use an object:

Less Maintainable

```
$("a.main").attr("href", "http://pluralsight.com")
           .attr("title", "Pluralsight Courses");
```

More Maintainable

```
$("a.main").attr({
    "href": "http://pluralsight.com",
    "title": "Pluralsight Courses"
});
```

DOM

Use Classes over Styles

pluralsight
hardcore developer training

Setting Multiple Styles with css()

- Multiple styles can be set using the jQuery css() function:

```
$("div").css({
    "background": "#efefef",
    "color": "#000",
    "border": "1px solid black"
});
```

- Is this the best way to set styles?

Use Classes Over Styles

- Favor adding classes over setting styles:

```
<style type="text/css">
    .panel {
            background: #efefef;
            color: #000;
            border: 1px solid black;
    }
</style>

...

$("div").addClass("panel");
```

Faster and easier to maintain by using classes

Toggling CSS Classes

- Toggle CSS classes using toggleClass():

```
<style type="text/css">
     .highlight { background:yellow; }
</style>

$("div").on("click", function() {
     $(this).toggleClass("highlight");
});

$("div").on("mouseenter mouseleave", function (e) {
    $(this).toggleClass("highlight", e.type ===
        "mouseenter");
});
```

Summary

- **Several jQuery techniques can be applied when working with the DOM:**
  - Use a CDN with a fallback
  - Cache selectors where possible
  - Limit DOM interactions
  - Use the end() function as appropriate when chaining
  - filter() and find() provide different ways to select nodes
  - Using objects literals with setter methods
  - Prefer classes over styles