

jQuery

Getting Started with jQuery



Why use jQuery?

How do you locate elements
with a specific class?

How do you apply styles to multiple
elements?



How many hours have you spent dealing
with cross-browser issues?

How do you handle events in a
cross-browser manner?

jQuery Makes Client-Side Dev Fun!



Getting Started with jQuery

- **To use jQuery**

- Download the jQuery script from <http://jquery.com>
- Need to support IE 6 – 8? Go with jQuery 1.x
- Don't need to support IE 6 – 8? Go with jQuery 2.x
- Reference the script in your web page:

```
<head>
```

```
  <script type="text/javascript" src="jquery.js"></script>
```

```
</head>
```

Getting jQuery Code Assistance

- **jQuery/JavaScript Editors:**

- Visual Studio
 - Eclipse (with plugins)
 - Aptana Studio
 - Sublime Text
 - WebStorm
 - Many more...
-

Agenda

- Why use jQuery?
 - Getting Started with jQuery
 - Using a Content Delivery Network (CDN)
 - Using the jQuery Ready Function
 - jQuery Documentation
-

Using a Content Delivery Network (CDN)

- Alternatively, you can use the Microsoft or Google CDN:

```
<head>  
  <script type="text/javascript"  
    src="http://ajax.microsoft.com/ajax/jquery/jquery-[version].js"></script>  
</head>
```

```
<head>  
  <script type="text/javascript"  
    src="http://ajax.googleapis.com/ajax/libs/jquery/[version]/jquery.min.js">  
  </script>  
</head>
```



What if the CDN is Down?

- If there was a problem with the CDN you can load the script locally:

```
<script type="text/javascript"
  src="http://ajax.googleapis.com/ajax/libs/jquery/[version]/jquery.min.js">
</script>
```

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




Detecting When a Page has Loaded

- Use **`$(document).ready()`** to detect when a page has loaded and is ready to use
- Called once DOM hierarchy is loaded (but before all images have loaded)

```
<script type="text/javascript">  
    $(document).ready(function(){  
        //Perform action here  
    });  
</script>
```

jQuery Documentation



write less, do more.

[Download](#) [API Documentation](#) [Blog](#) [Plugins](#) [Browser Support](#)

- Ajax
 - Global Ajax Event Handlers
 - Helper Functions
 - Low-Level Interface
 - Shorthand Methods
- Attributes
- Callbacks Object
- Core
- CSS
- Data
- Deferred Object
- Deprecated
 - Deprecated 1.3
 - Deprecated 1.7
 - Deprecated 1.8
- Dimensions
- Effects

jQuery API

.add() Traversing > Miscellaneous Traversing

Add elements to the set of matched elements.

.addBack() Traversing > Miscellaneous Traversing

Add the previous set of elements on the stack to the current set, optionally filtered by a selector.

.addClass() Attributes | Manipulation > Class Attribute | CSS

Adds the specified class(es) to each of the set of matched elements.

.after() Manipulation > DOM Insertion, Outside

Insert content, specified by the parameter, after each element in the set of matched elements.

.ajaxComplete() Ajax > Global Ajax Event Handlers

Register a handler to be called when Ajax requests complete. This is an AjaxEvent.

jQuery Selectors

Introduction to Selectors

- Selectors allow page elements to be selected
- Single or multiple elements are supported
- A selector identifies an HTML element/tag that you will manipulate with jQuery Code

```
<div id="CustomersDiv" class="Bright">  
  <span class="Text">Welcome John</span>  
</div>
```

Selector Syntax

`$(selectorExpression)`

`jQuery(selectorExpression)`

Selecting by Tag Name

`$('p')` selects all `<p>` elements

`$('a')` selects all `<a>` elements

Selecting Multiple Tags

- To reference multiple tags, use the , character to separate the elements:

`$('p, a, span')`

selects all paragraphs, anchors, and span elements

Selecting Descendants

- `$('ancestor descendant')` selects all descendants of the ancestor:

`$('table tr')`

Selects all `tr` elements that are descendants of the `table` element

- Descendants are children, grandchildren, etc of the designated ancestor element
-

Selecting by Element ID

- Use the # character to select elements by ID:

`$('#myID')`

selects `<p id="myID">` element

Selecting Elements by Class Name

- Use the `.` character to select elements by class name:

`$('.myClass')`

selects `<p class="myClass">` element

Selecting Multiple Class Names

- To reference multiple tags, use the , character to separate the class names:

```
$('.BlueDiv, .RedDiv')
```

selects all elements containing the class `BlueDiv` and `RedDiv`

Selecting by Tag Name and Class Name

- You can combine this with element names as well:

`$('a.myClass')`

selects only `<a>` tags with `class="myClass"`

Selecting By Attribute Value

- Use brackets *[attribute]* to select based on attribute name and/or attribute value:

`$('a[title]')`

selects all <a> elements that have a title attribute

`$('a[title="Programming Info"]')`

selects all anchor elements that have a "Programming Info" title attribute value

Selecting Input Elements

- **`$(':input')` selects all input elements including: input, select, textarea, button, image, radio and more**

`$(':input[type="radio"]')`

targets all radio buttons on the page...but is it the most efficient selector?

Using Contains in Selectors

- **:contains()** will select elements that match the contents within the contains exception:

`$('div:contains("`

`")')`

selects **div's** that contain the text
match is case-sensitive)

(note that the

`<div>Expert`

`Training</div>`

Selecting Even or Odd Rows in a Table

- `$('tr:odd')` and `$('tr:even')` is the jQuery syntax for selecting odd or even rows respectively
- Remember the index is 0 based - the first row in the table is 0:
 - Odd would return 1, 3, 5, 7, 9, etc
 - Even would return 0, 2, 4, 6, 8, etc

Selecting the First Child

- **`$('element:first-child')`** selects the first child of every element group:

`$('span:first-child')`

```
<div>
```

```
  <span>First Child, first group</span>
```

```
  <span>Second Child, first group</span>
```

```
</div>
```

```
<div>
```

```
  <span>First Child, second group</span>
```

```
  <span>Second Child, second group Child</span>
```

```
</div>
```

Using Starts With in Selectors

- **`[attribute^="value"]`** will select all elements with an attribute that begins with stated value:

`$('input[value^="Events"]')`

selects any input element whose value attribute begins with "Events":

```
<input type="button" value="Events – World"/>
```

```
<input type="button" value="Events – National"/>
```

```
<input type="button" value="Events – Local"/>
```

Using Ends With in Selectors

- `[attribute$="value"]` will select all elements with an attribute that ends with stated value:

`$(input[value$="Events"])`

selects any input element whose value attribute ends with "Events":

```
<input type="button" value="World Events"/>
```

```
<input type="button" value = "National Events"/>
```

```
<input type="button" value = "Local Events"/>
```

Find Attributes Containing a Value

- `[attribute*="value"]` will select all elements with an attribute that contain the stated value:

`$('input[value*="Events"]')`

selects any input element whose value attribute contains "Events":

```
<input type="button" value="World Events 2011"/>
```

```
<input type="button" value = "National Events 2011"/>
```

```
<input type="button" value = "Local Events 2011"/>
```

Summary

- **Selectors allow any type of HTML element to be located in an HTML page**
 - **Key jQuery selector characters include:**
 - # for ID selections
 - . for class selection
 - ^ for attributes starting with a value
 - \$ for attributes ending with a value
 - * for attributes containing a value
-

Interacting with the DOM

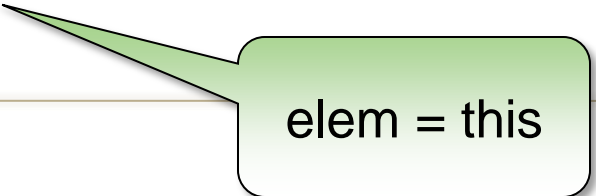
Iterating Through Nodes

- `.each(function(index, Element))` is used to iterate through jQuery objects:

```
$('div').each(function(index) {  
    alert(index + ' = ' + $(this).text());  
});
```

Iterates through each div element and returns its index number and text

```
$('div').each(function(index,elem) {  
    alert(index + ' = ' + $(elem).text());  
});
```



elem = this

Modifying Object Properties

- The ***this.propertyName*** statement can be used to modify an object's properties directly:

```
$('#div').each(function(i) {  
    this.title = "My Index = " + i;  
});
```

Iterates through each div and modifies the title. If the property does not exist, it will be added

Accessing Attributes

- Object attributes can be accessed using **attr()**:

```
var val = $('#CustomerDiv').attr('title');
```

Retrieves the value of the title attribute

Modifying Attributes

- `.attr(attributeName, value)` is the method used to access an object's attributes and modify the value:

```
$('img').attr('title', 'My Image Title');
```

Changes the title attribute to a value of *My Image Title*

Modifying Multiple Attributes

- To modify multiple attributes, pass a JSON object containing name/value pairs:

```
$('img').attr({  
  title: 'My Image Title',  
  style: 'border:2px solid black';  
});
```

JSON object passed and used to change title and border

Wait...What's JSON?

- JSON delimits objects using { and }
- The : character separates properties and values

```
{  
  FirstName: 'John',  
  LastName  
  Address: {  
    Street: '1234 Anywhere St.',  
    City: 'Phoenix',  
    State: 'AZ',  
    ZipCode: 85249  
  }  
}
```

Agenda

- Iterating Through Nodes
- Modifying DOM Object Properties
- Modifying Attributes
- **Adding and Removing Nodes**
- Modifying Styles
- Modifying Classes

Adding and Removing Nodes

- Four key methods handle inserting nodes into elements:

`.append()`

`.appendTo()`

`.prepend()`

`.prependTo()`

- To remove nodes from an element use `.remove()`

Appending to Nodes

- Appending adds children at the end of the matching element:

```
$('<span> (office)</span>').appendTo('.officePhone');
```

OR

```
$('.officePhone').append('<span> (office)</span>');
```

Would result in (office) being added into each .officePhone class element

Prepending to Nodes

- Prepending adds children at the beginning of the matching element:

```
$('<span>Phone: </span>').prependTo('.phone');
```

OR

```
$('.phone').prepend('<span>Phone: </span>');
```

Would result in Phone: being added into each .phone class element

Wrapping Elements

- The following HTML and `.wrap()` function:

```
<div class="state">Arizona</div>
```

```
$('.state').wrap('<div class="US_State" />');
```

Results in:

```
<div class="US_State">  
  <div class="state">Arizona</div>  
</div>
```

Removing Nodes

- `.remove()` will remove matched elements from the DOM:

```
$('.phone, .location').remove();
```

Will result in objects with `.phone` or `.location` classes being removed from the DOM

Agenda

- Iterating Through Nodes
- Modifying DOM Object Properties
- Modifying Attributes
- Adding and Removing Nodes
- **Modifying Styles**
- Modifying Classes

Modifying Styles

- The `.css()` function can be used to modify an object's style:

```
$("#div").css("color", "red");
```

Modifying Multiple Styles

- Multiple styles can be modified by passing a JSON object:

```
$('#div').css( {  
    'color' : '#ccc',  
    'font-weight' : 'bold'  
});
```

Agenda

- Iterating Through Nodes
- Modifying DOM Object Properties
- Modifying Attributes
- Adding and Removing Nodes
- Modifying Styles
- **Modifying Classes**

Modifying CSS Classes

- The four methods for working with CSS Class attributes are:

`.addClass()`

`.hasClass()`

`.removeClass()`

`.toggleClass()`

Adding a CSS Classes

- **.addClass()** adds one or more class names to the class attribute of each matched element:

```
$('p').addClass('classOne');
```

- More than one class:

```
$('p').addClass('classOne classTwo');
```

Matching CSS Classes

- **.hasClass()** returns true if the selected element has a matching class that is specified:

```
if($('p').hasClass('styleSpecific')) {  
    //Perform work  
}
```

Removing CSS Classes

- **.removeClass()** can remove one or more classes:

```
$('p').removeClass('classOne classTwo');
```

- Remove all class attributes for the matching selector:

```
$('p').removeClass();
```

Toggling CSS Classes

- **.toggleClass()** alternates adding or removing a class based on the current presence or absence of the class:

`$('#PhoneDetails').toggleClass('highlight');`

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

Agenda

- **jQuery Event Model Benefits**
- **Handling Events**
- **Binding to Events**
- **live(), delegate() and on()**
- **Handling Hover Events**

Handling Events using JavaScript

Question:

What type of JavaScript code do you write to handle a button click event?

Answer:

It depends on the browser!



Event Attachment Techniques

Most Browsers:

```
myButton.addEventListener('click', function() { },false);
```

Internet Explorer (IE8 and earlier):

```
myButton.attachEvent('onclick', function() { });
```



jQuery Event Model Benefits

- **Events notify a program that a user performed some type of action**
- **jQuery provides a cross-browser event model that works in IE, Chrome, Opera, FireFox, Safari and more**
- **jQuery event model is simple to use and provides a compact syntax**

jQuery Event Shortcut Functions

- **jQuery event shortcuts:**

- `click()`
- `blur()`
- `focus()`
- `dblclick()`
- `mousedown()`
- `mouseup()`
- `mouseover()`
- `keydown()`,
- `keypress()`
- See more at <http://api.jquery.com/category/events>

Handling Click Events

- `.click(handler(eventObject))` is used to listen for a click event or trigger a click event on an element

```
$('#myID').click(function() {  
    alert('The element myID was clicked');  
});
```

Handling Click Events

- Raising a click event from within another function:

```
$( '#otherID' ).click(function() {  
    $( '#myID' ).click();  
});
```

- This would fire when the element otherID was clicked and raise the click event for myID

Using on()

- **.on(eventType, handler(eventObject))** attaches a handler to an event for the selected element(s)

```
$('#MyDiv').on('click', function() {  
    //Handle click event  
});
```

Using off()

- **.off(event)** is used to remove a handler previously bound to an element:

`$('#test').click(handler);` can be unbound using
`$("#test").off();`

- Specific events can also be targeted using off():

`$('#test').off('click');`

Binding Multiple Events with on()

- on() allows multiple events to be bound to one or more elements
- Event names to bind are separated with a space:

```
$('#MyDiv').on('mouseenter mouseleave',  
    function() {  
        $(this).toggleClass('entered');  
    }  
);
```

live(), delegate(), and on() Functions

- **live(), delegate(), and on()** allow new DOM elements to automatically be "attached" to an event handler
- Allow children to be added to a container without explicitly attaching an event handler to each child

Using live()

- Event handlers can be set using live()
- The document object handles events by default
- Works even when new objects are added into the DOM:

```
$('.someClass').live('click',  
    someFunction);
```

live() removed in jQuery 1.9

- Stop live event handling using die():

```
$('.someClass').die('click', someFunction);
```


How live() Works

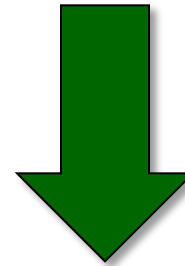
Document Object

Click Event

Event Handler



Click Event



``

`<p class="someClass" />`

`<div class="someClass" />`



`$('.someClass').live('click',
function() {});`

Using delegate()

- Newer version of live() added in jQuery 1.4
- A context object (#Divs in the sample below) handles events by default rather than the document object
- Works even when new objects are added into the DOM:

```
$('#Divs').delegate('div','click',someFunction);
```

- Stop delegate event handling using undelegate()

How delegate() Works

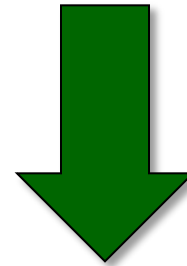
Context Object: #Divs

Click Event

Event Handler



Click Event



```
<span class="someClass" />  
<p class="someClass" />  
<div class="someClass" />
```

```
$('#Divs').delegate('.someClass',  
  'click',func() { });
```

The on() Function

- The on() function is a new replacement for the following:
 - bind()
 - delegate()
 - live()

```
$('div').on('click', function(){  
    alert('Clicked button!');  
});
```

Using on() with Child Objects

- The on() function can be used in place of live() and delegate()
- Works when new objects are added into the DOM:

```
$("#MyTable tbody").on("click", "tr",  
    function(event){  
        alert('Row was clicked and bubbled up');  
    });
```

Using on() with a Map

- Multiple events and handlers can be defined in on() using a "map":

```
$("#MyTable tr").on({  
  mouseenter: function(){  
    $(this).addClass("over");  
  },  
  mouseleave: function(){  
    $(this).removeClass("out");  
  }  
});
```

Handling Hover Events

- Hover events can be handled using **hover()**:

`$(selector).hover(handlerIn, handlerOut)`

- **handlerIn** is equivalent to **mouseenter** and **handlerOut** is equivalent to **mouseleave**

Using hover()

- This example highlights #target on mouseenter and sets it back to white on mouseleave

```
$('#target').hover(  
    function(){  
        $(this).css('background-color', '#00FF99');  
    },  
    function(){  
        $(this).css('background-color', '#FFFFFF');  
    }  
);
```


Alternate Hover Example

- Another option is `$(selector).hover(handlerInOut)`
- Fires the same handler for mouseenter and mouseleave events
- Used with jQuery's toggle methods:

```
$('#p').hover(function() {  
    $(this).toggleClass('over');  
});
```

This code will toggle the class applied to a paragraph element

jQuery Ajax Features

- **jQuery allows Ajax requests to be made from a page:**
 - Allows parts of a page to be updated
 - Cross-Browser Support
 - Simple API
 - GET and POST supported
 - Load JSON, XML, HTML or even scripts

jQuery Ajax Functions

- **jQuery provides several functions that can be used to send and receive data:**
 - `$(selector).load()`: Loads HTML data from the server
 - `$.get()` and `$.post()`: Get raw data from the server
 - `$.getJSON()`: Get/Post and return JSON
 - `$.ajax()`: Provides core functionality
- **jQuery Ajax functions work with REST APIs, Web Services and more**

Agenda

- jQuery Ajax Functions
- Loading HTML Content from the Server
- Making GET Requests
- Making POST Requests
- Introduction to the ajax() Function

Using load()

- `$(selector).load(url,data,callback)` allows HTML content to be loaded from a server and added into a DOM object:

```
$(document).ready(function(){  
    $('#HelpButton').click(function(){  
        $('#MyDiv').load('HelpDetails.html');  
    });  
});
```

Using load() With a Selector

- A selector can be added after the URL to filter the content that is returned from calling load():

```
$('#MyDiv').load('HelpDetails.html #MainTOC');
```

Passing Data using load()

- Data can be passed to the server using **load(url,data)**:

```
$( '#MyDiv' ).load( 'GetCustomers.aspx',  
    {PageSize:25});
```

Using a Callback Function with load()

- load() can be passed a callback function:

```
$('#OutputDiv').load('NotFound.html',  
    function (response, status, xhr) {  
        if (status == "error") {  
            alert(xhr.statusText);  
        }  
    });
```


Using get()

- **\$.get(url,data,callback,datatype)** can retrieve data from a server:

```
$.get('HelpDetails.html', function (data) {  
    $('#OutputDiv').html(data);  
});
```

Using getJSON()

- **\$.getJSON(url,data,callback)** can retrieve data from a server:

```
$.getJSON('CustomerJson.aspx',{id:1},  
    function (data) {  
        alert(data.FirstName + ' ' +  
            data.LastName);  
    });
```

Agenda

- jQuery Ajax Functions
- Loading HTML Content from the Server
- Making GET Requests
- **Making POST Requests**
- Introduction to the ajax() Function

Using post()

- **\$.post(url,data,callback,datatype)** can post data to a server and retrieve results:

```
$.post('GetCustomers.aspx', {PageSize:15},  
      function (data) {  
          $('#OutputDiv').html(data);  
      }  
);
```

Using post() to Call a WCF Service

- **post()** can also be used to interact with an Ajax-enabled WCF service:

```
$.post('CustomerService.svc/GetCustomers',  
    null, function (data) {  
        var cust = data.d[0];  
        alert(cust.FirstName + ' ' +  
            cust.LastName);  
    }, 'json');
```

Agenda

- jQuery Ajax Functions
- Loading HTML Content from the Server
- Making GET Requests
- Making POST Requests
- **Introduction to the ajax() Function**

The ajax() Function

- The ajax() function provides extra control over making Ajax calls to a server
- Configure using JSON properties:
 - contentType
 - data
 - dataType
 - error
 - success
 - type (GET or POST)

Using the ajax() Function

The ajax() function is configured by assigning values to JSON properties:

```
$.ajax({  
    url: '../CustomerService.svc/InsertCustomer',  
    data: customer,  
    dataType: 'json',  
    success: function (data, status, xhr) {  
        alert("Insert status: " + data.d.Status + '\n' +  
            data.d.Message);  
    },  
    error: function (xhr, status, error) {  
        alert('Error occurred: ' + status);  
    }  
});
```