Session 1 : JQuery Fundamentals

February 2012

Contents

JQuery Fundamentals

- At its core, JQuery focuses on retrieving elements from our HTML pages and performing operations upon them.
- With JQuery, we will be able to leverage our knowledge and that degree of power to vastly simplify our JavaScript.
- JQuery places a high priority on ensuring our code will work in a consistent manner across all major browsers, many of the more difficult JavaScript problems, such as waiting until the page is loaded before performing page operations, have been silently solved for us.

Contents

JQuery Wrapper (1/3)

- When CSS was introduced to web technologies in order to separate design from content, a way was needed to refer to groups of page elements from external style sheets.
- The method developed was through the use of selectors, which concisely represent elements based upon their attributes or position within the HTML document.
- JQuery uses similar selectors to group elements.

Examples	Explanations
p a	• Refers to the group of all links (<a> elements) that are nested inside a <p> element. • JQuery makes use of the same selectors, supporting not only the common selectors currently used in CSS, but also the more powerful ones.
$(selector)/ JQuery(selector)	• To collect a group of elements, we use these simple syntax.
$("p a")	• It helps to retrieve the group of links nested inside a <p> element. • The $() function (an alias for the JQuery() function) returns a special JavaScript object containing an array of the DOM elements that match the selector.

JQuery Wrapper (2/3)

Examples	Explanations
$("div.notLongForThisWorld").fadeOut();	• It fades out all elements with the CSS class not-long for this world.
$("div.notLongForThisWorld").fadeOut().addClass("removed");	• A special feature of a large number of these methods, which we often refer to as JQuery commands, is that when they're done with their action (like a fading-out operation), they return the same group of elements, ready for another action. • For example, say that we want to add a new CSS class, removed, to each of the elements in addition to fading them out.
$("#someElement").html("I have added some text to an element"); or $("#someElement")[0].innerHTML = "……"	• Two statements produce identical results: • "I have added some text to an element"

JQuery Wrapper (3/3)

Examples	Explanations

$("div.fillMeIn")
.html("I have added some text to a group of nodes");
Or
var elements = $("div.fillMeIn");
for(i=0;i<elements.length;i++)
elements[i].innerHTML
	• If we want to achieve the same results with a selector that resulted in multiple matched elements, the following two fragments would produce identical results:
	
	• "I have added some text to a group of nodes"

$("p:even");	• This selector selects all even <p> elements.
$("tr:nth-child(1)");	• This selector selects the first row of each table.
$("body > div");	• This selector selects direct children of <body>.
$("a[href$=pdf]");	• This selector selects links to PDF files.
$("body > div:has(a)")	• This selector selects direct children of <body>-containing links.

Contents

Document Ready Handler (1/2)

- Document ready handler helps to write functions which are called after the DOM tree is created
- The function waits only until the document structure is fully parsed and the browser has converted the HTML into its DOM tree form before executing the script to apply the rich behaviors.
- Accomplishing this in across-browser manner is somewhat difficult, but JQuery provides a simple means to trigger the execution of code once the DOM tree, but not external image resources, has loaded.

Document Ready Handler (2/2)

Examples	Explanations
$(document).ready(function() { $("table tr:nth-child(even)").addClass("even"); });	• We wrap the document instance with the JQuery() function, and then we apply the ready() method, passing a function to be executed when the document is ready to be manipulated.
$(function() { $("table tr:nth-child(even)").addClass("even"); });	• By passing a function to $(), we instruct the browser to wait until the DOM has fully loaded (but only the DOM) before executing the code. • These can be executed multiple times within the same HTML document, and the browser will execute all of the functions we specify in the order that they are declared within the page. • In contrast, the window's onload technique allows for only a single function.

Contents

Making DOM Elements

- We can create DOM elements on the fly by passing the $() function a string that contains the HTML markup for those elements.
- For example, we can create a new paragraph element as follows:

```
$("<p>Hi there!</p>")
```

```
<html>
   <head>
     <title>Follow me!</title>
     <script type="text/javascript" src="../scripts/jquery-1.2.js">
     </script>
     <script type="text/javascript">
       $(function(){
         $("<p>Hi there!</p>").insertAfter("#followMe");
       });
     </script>
   </head>

  <body>
     <p id="followMe">Follow me!</p>
  </body>
</html>
```

Contents

Extending JQuery

- By extending JQuery, we can add our own functions which can use the powerful features it provides, particularly in the area of element selection.
- Let's look at a particular example:
  - JQuery doesn't come with a predefined function to disable a group of form elements.
  - And if we're using forms throughout our application, we might find it convenient to be able to use the following syntax:

```
$("form#myForm input.special").disable();
```

- Fortunately, and by design, JQuery makes it easy to extend its set of functions by extending the wrapper returned when we call $().

```
$.fn.disable = function() {
    return this.each(function() {
      if (typeof this.disabled != "undefined") this.disabled = true;
   });
}
```

Contents

Manipulating Elements (1/13)

Using basic CSS selectors

- For applying styles to page elements, web developers have become familiar with a small, but powerful and useful, group of selection methods that work across all browsers.
- Those methods include selection by an element's ID, CSS class name, tag name, and the DOM hierarchy of the page elements.
- Here are some examples.

Selector Types	Explanation
a	• This selector matches all link (<a>) elements.
#specialID	• This selector matches elements that have an id of specialID.
.specialClass	• This selector matches elements that have the class of specialClass.
a#specialID.specialClass	• This selector matches links with an id of specialID and a class of specialClass.
p a.specialClass	• This selector matches links with a class of specialClass declared within <p> elements.

Manipulating Elements (2/13)

Using child, container, and attribute selectors

- Example shows how we can select list elements directly under some list, but not list elements belonging to a sublist.

```
<ul class="myList">
    <li><a href="http://jquery.com">JQuery supports</a>
     <ul>
          <li><a href="css1">CSS1</a></li>
          <li><a href="css2">CSS2</a></li>
          <li><a href="css3">CSS3</a></li>
          <li>Basic XPath</li>
       </ul>
     </li>
     <li>jQuery also supports
       <ul>
       <li>Custom selectors</li>
       <li>Form selectors</li>
     </ul>
    </li>
</ul>
```

- Suppose we want to select the link to the remote JQuery site, but not the links to various local pages describing the different CSS specifications

Manipulating Elements (3/13)

- Consider a selector such as

Examples	Explanation
ul.myList > li > a	• This selector selects only links that are direct children of list elements, which are in turn direct children of elements that have the class myList.
a[href^=http://]	• Attribute selectors are also extremely powerful. • These example helps to select links that points to locations outside our sites.

Manipulating Elements (4/13)

- There are other ways to use attribute selectors.
- To match an element that possesses a specific attribute, regardless of its value, we can use

Examples	Explanation
form[method]	• This matches any <form> element that has an explicit method attribute.
input[type=text]	• This selector matches all input elements with a type of text.
div[title^=my]	• This selects all elements with title attributes whose value begins with my.
a[href$=.pdf]	• This is a useful selector for locating all links that reference PDF files.
a[href*=jquery.com]	• As we would expect, this selector matches all <a> elements that reference the JQuery site.

Manipulating Elements (5/13)

- To select an element only if it contains some other element.
- JQuery supports this kind of selection with the container selector:

Examples	Explanation
li:has(a)	• This selector matches all elements that contain an <a> element.
foo:not(bar:has(baz))	• Only a single level of nesting is supported. • Although it's possible to nest *one* level
foo:not(bar:has(baz:eq(2)))	• Additional levels of nesting, aren't supported.

Manipulating Elements (6/13)

Selector	Description
*	Matches any element.
E	Matches all element with tag name E.
E F	Matches all elements with tag name F that are descendents of E.
E>F	Matches all elements with tag name F that are direct children of E.
E+F	Matches all elements F immediately preceded by sibling E.
E~F	Matches all elements F preceded by any sibling E.
E:has(F)	Matches all elements with tag name E that have at least one descendent with tag name F.
E.C	Matches all elements E with class name C. Omitting E is the same as *.C.
E#I	Matches element E with id of I. Omitting E is the same as *#I.
E[A]	Matches all elements E with attribute A of any value.
E[A=V]	Matches all elements E with attribute A whose value is exactly V.
E[A^=V]	Matches all elements E with attribute A whose value begins with V.
E[A$=V]	Matches all elements E with attribute A whose value ends with V.
E[A*=V]	Matches all elements E with attribute A whose value contains V.

Manipulating Elements (7/13)

Selecting by position

- If we will need to select elements by their position on the page or in relation to other elements.
- For example - we might want to select the first link on the page, or every other paragraph, or the last list item of each list.
- JQuery supports mechanisms for achieving these specific selections.

Examples	Explanation
a:first	• This format of selector matches the first <a> element on the page.
p:odd	• This selector matches every odd paragraph element.
p:even	• We can also specify that evenly ordered elements be selected
li:last-child	• Chooses the last child of parent elements. • In this example, the last child of each element is matched.

Manipulating Elements (8/13)

Selecting by position

The more advanced positional selectors supported by jQuery: selecting elements based on their position in the DOM

Selector	Description
`:first`	The first match of the page. `li a:first` returns the first link also under a list item.
`:last`	The last match of the page. `li a:last` returns the last link also under a list item.
`:first-child`	The first child element. `li:first-child` returns the first item of each list.
`:last-child`	The last child element. `li:last-child` returns the last item of each list.
`:only-child`	Returns all elements that have no siblings.
`:nth-child(n)`	The *n*th child element. `li:nth-child(2)` returns the second list item of each list.
`:nth-child(even|odd)`	Even or odd children. `li:nth-child(even)` returns the even children of each list.
`:nth-child(Xn+Y)`	The *n*th child element computed by the supplied formula. If `Y` is `0`, it may be omitted. `li:nth-child(3n)` returns every third item, whereas `li:nth-child(5n+1)` returns the item after every fifth element.
`:even and :odd`	Even and odd matching elements page-wide. `li:even` returns every even list item.
`:eq(n)`	The *n*th matching element.
`:gt(n)`	Matching elements after (and excluding) the *n*th matching element.
`:lt(n)`	Matching elements before (and excluding) the *n*th matching element.

Manipulating Elements (9/13)

Selecting by position

- The nth-child selector starts counting from 1, whereas the other selectors start counting from 0.
- For CSS compatibility, nth-child starts with 1, but the JQuery custom selectors follow the more common programming convention of starting at 0.
- Consider the following table, containing a list of some programming languages and some basic information regarding them:

Examples	Explanation
table#languages tbody td:first-child	• We want to get all of the table cells that contained the names of programming languages. Because they are all the first cells in their row
table#languages tbody td:nth-child(1)	• We want to get first child of the table cell that contained the names of programming languages.

Manipulating Elements (10/13)

Using custom JQuery selectors

- The CSS selectors give us a great deal of power and flexibility to match the desired DOM elements, but sometimes we will want to select elements based on a characteristic that the CSS specification did not anticipate.
- For example, we might want to select all check boxes that have been checked by the user.
- Because trying to match by attribute will only check the initial state of the control as specified in the HTML markup, JQuery offers a custom selector, :checked, that filters the set of matched elements to those that are in checked state.
- For example, whereas the input selector selects all <input> elements, the input:checked narrows the search to only <input> elements that are checked.
- Combining these custom selectors can be powerful; consider :radio:checked and :checkbox:checked.

Manipulating Elements (11/13)

Selector	Description
`:animated`	Selects elements that are currently under animated control. Chapter 5 will cover animations and effects.
`:button`	Selects any button (`input[type=submit]`, `input[type=reset]`, `input[type=button]`, or `button`).
`:checkbox`	Selects only check box elements (`input[type=checkbox]`).
`:checked`	Selects only check boxes or radio buttons that are checked (supported by CSS).
`:contains(foo)`	Selects only elements containing the text *foo*.
`:disabled`	Selects only form elements that are disabled in the interface (supported by CSS).
`:enabled`	Selects only form elements that are enabled in the interface (supported by CSS).
`:file`	Selects all file elements (`input[type=file]`).
`:header`	Selects only elements that are headers; for example: `<h1>` through `<h6>` elements.
`:hidden`	Selects only elements that are hidden.
`:image`	Selects form images (`input[type=image]`).
`:input`	Selects only form elements (`input`, `select`, `textarea`, `button`).
`:not(filter)`	Negates the specified filter.
`:parent`	Selects only elements that have children (including text), but not empty elements.
`:password`	Selects only password elements (`input[type=password]`).
`:radio`	Selects only radio elements (`input[type=radio]`).
`:reset`	Selects reset buttons (`input[type=reset]` or `button[type=reset]`).
`:selected`	Selects option elements that are selected.
`:submit`	Selects submit buttons (`button[type=submit]` or `input[type=submit]`).
`:text`	Selects only text elements (`input[type=text]`).
`:visible`	Selects only elements that are visible.

Manipulating Elements (12/13)

Using custom JQuery selectors

Examples	Explanation
:checkbox:checked:enabled	• We can combine selector filters too. These example helps to select only enabled and checked check boxes.
input:not(:checkbox)	• :not filter is used to negate a match,which is supported for CSS filters and works with custom JQuery selector filters too. Example is to select non-check box <input> elements.

- It's important to recognize the distinction between filter and find selectors.
- Filter selectors selects a matching set of elements by applying a further selection criteria to them.
- Find selectors, such as the descendent selector (space character), the child selector (>), and the sibling selector (+), find other elements that bear some relationship to the ones already selected, rather than limiting the scope of the match with criteria applied to the matched elements.

Manipulating Elements (13/13)

Using custom JQuery selectors

- We can apply the :not filter to filter selectors, but not to find selectors.

```
div p:not(:hidden) – valid
div :not(p:hidden) – invalid
```

- In the first case, all <p> elements descending from a element that aren't hidden are selected.
- The second selector is illegal because it attempts to apply :not to a selector that isn't a filter (the p in p:hidden isn't a filter).
- To make things simpler, filter selectors are easily identified because they all begin with a colon character (:) or a square bracket character ([).
- Any other selector can't be used inside the :not() filter.

Contents

- JQuery Fundamentals
- JQuery Wrapper
- Document Ready Handler
- Making DOM Elements
- Extending JQuery
- Manipulating Elements
- Generating New HTML
- Wrapped Set
- Managing JQuery Chains

Generating New HTML

Examples	Explanation
$(" ")	• To create an empty element

$(" I have foo!
 I don't ")
.filter(".foo").click(function() {
alert("I'm foo!");
}).end().appendTo("#someParentDiv");
	• In this snippet, we first create two elements, one with class foo and one without.
	• We then narrow down the selection to only the with class foo and bind an event handler to it that will fire an alert dialog box when clicked.
	• Finally, we use the end() method to revert back to the full set of both elements and attach them to the DOM tree by appending them to the element with the id of someParentDiv.

Contents

Wrapped Set - Determining Size

```
$('#someDiv')
    .html('There are '+$('a').size()+' link(s) on this page.');
```

- The inner JQuery wrapper matches all elements of type <a> and returns the number of matched elements using the size() method.

Size Syntax

Command	Explanation
Size	• Returns the count of elements in the wrapped set
Parameter	• None
Returns	• The element count

Wrapped Set - Obtaining Elements (1/3)

Examples	Explanation
$('img[alt]')[0]	• It helps to obtain the first element in the set of all <img> elements with an alt attribute on the page.

Get Syntax

Command	Explanation
Get(index)	• Obtains one or all of the matched elements in the wrapped set. • If no parameter is specified, all elements in the wrapped set are returned in a JavaScript array. • If an index parameter is provided, the indexed element is returned.
Parameter	**Index** (Number) • The index of the single element to return. If omitted, the entire set is returned in an array.
Returns	• A DOM element or an array of DOM elements.

Wrapped Set - Obtaining Elements (2/3)

Examples	Explanation
$('img[alt]').get(0)	• It is equivalent to the previous example that used array indexing. • The get() method can also be used to obtain a plain JavaScript array of all the wrapped elements.
var allLabeledButtons =$('label+button').get();	• This statement wraps all the <button> elements on a page that are immediately preceded by <label> elements in a JQuery wrapper and then creates a JavaScript array of those elements to assign to the allLabeledButtons variable.

Wrapped Set - Obtaining Elements (3/3)

Command	Explanation
Index(element)	• Finds the passed element in the wrapped set and returns its ordinal index within the set. • If the element isn't resident in the set, the value -1 is returned.
Parameter	**element** (Element) • A reference to the element whose ordinal value is to be determined.
Returns	• The ordinal value of the passed element within the wrapped set or -1 if not found.

Example

```
var n = $('img').index($('img#findMe')[0]);

// We can use an inverse operation to find the index of a particular element in the wrapped set.
```

Wrapped Set - Slicing and Dicing

Examples	Explanation
$('img[alt],img[title]')	• It helps us to match all <img> elements that have either an alt or a title attribute.
$('img[alt]').add('img[title]')	• Using the add() method in this fashion allows us to chain a bunch of selectors together into an or relationship, creating the union of the elements that satisfy both of the selectors.

Add Syntax

Command	Explanation
add (expression)	• Adds elements, specified by the expression parameter, to the wrapped set. • The expression can be a selector, an HTML fragment, a DOM element, or an array of DOM elements.
Parameter	**expression** (String|Element|Array) • Specifies what is to be added to the matched set. • This parameter can be a JQuery selector, in which case any matched elements are added to the set.
Returns	• The wrapped set.

Wrapped Set - Creating & Honing the Set (1/6)

- Let's say that we want to apply a thick border to all <img> elements with alt attributes, and then apply a level of transparency to all <img> elements with either alt or title attributes.
- The comma operator (,) of CSS selectors won't help us with this one because we want to apply an operation to a wrapped set and then add more elements to it.
- We could easily accomplish this with multiple statements, but it would be more efficient and elegant to use the power of JQuery chaining to accomplish the task in a single statement, such as

Examples	Explanation
$('img[alt]').addClass('thickBorder').add('img[title]') .addClass('seeThrough')	• In this statement, we create a wrapped set of all <img> elements with alt attributes, apply a predefined class that applies a thick border, add the <img> elements that have title attributes, and finally apply a class that applies transparency to the newly augmented set.
$('p').add(' Hi there! ')	• This fragment creates a wrapped set of all <p> elements in the document, and then creates a new , and adds it to the wrapped set.

Wrapped Set - Creating & Honing the Set (2/6)

Examples	Explanation
$('img[title]').not('[title*=puppy]')	• We want to select all <img> elements in a page that sport a title attribute except for those that contain the text puppy in the title attribute value.

Not Syntax

Command	Explanation
Not(Expression)	• Removes elements from the matched set according to the value of the expression parameter. • If the parameter is a JQuery filter selector, any matching elements are removed. • If an element reference is passed, that element is removed from the set.
Parameter	**expression** (String|Element|Array) • A JQuery filter expression, element reference, or • array of element references defining what is to be removed from the wrapped set.
Returns	• The wrapped set.

Wrapped Set - Creating & Honing the Set (3/6)

- The filter() method, when passed a function, invokes that function for each wrapped element and removes any element whose function invocation returns the value false.
- Each invocation has access to the current wrapped element via the function context (this) in the body of the filtering function.

Examples	Explanation
$('td').filter(function() {return this.innerHTML.match(/^\d+$/)})	• This JQuery expression creates a wrapped set of all <td> elements and then invokes the function passed to the filter() method for each, with the current matched elements as the this value for the invocation. • The function uses a regular expression to determine if the element content matches the described pattern (a sequence of one or more digits), returning false if not. • Every element whose filter function invocation returns false is removed from the wrapped set.

Wrapped Set - Creating & Honing the Set (4/6)

Filter Syntax

Command	Explanation
Filter(Expression)	• Filters out elements from the wrapped set using a passed selector expression, or a filtering function.
Parameter	**Expression** (String|Function) • Specifies a JQuery selector used to remove all elements that do not match from the wrapped set, or a function that makes the filtering decision. • This function is invoked for each element in the set, with the current element set as the function context for that invocation. • Any element that returns an invocation of false is removed from the set.
Returns	• The wrapped set.

Wrapped Set - Creating & Honing the Set (5/6)

- Slice() - If we want to obtain a subset of the wrapped set, based on the position of elements within the set.
- This command creates and returns a new set from any contiguous portion, or a slice, of an original wrapped set.

Slice syntax

Examples	Explanation
slice (began, end)	• Creates and returns a new wrapped set containing a contiguous portion of the matched set.
Parameter	**begin -** (Number) • The zero-based position of the first element to be included in the returned slice. **end -** (Number) • The optional zero-based index of the first element not to be included in the returned slice, or one position beyond the last element to be included. • If omitted, the slice extends to the end of the set.
Returns	• The newly created wrapped set.

Wrapped Set - Creating & Honing the Set (6/6)

Examples on Slice

Examples	Explanation
$('*').slice(2,3);	• This statement selects all elements on the page and then generates a new set containing the third element in the matched set.
$('*').slice(0,4);	• Selects all elements on the page and then creates a set containing the first four elements.
$('*').slice(4);	• Matches all elements on the page and then returns a set containing all but only the first four elements.

Wrapped Set - Methods

Method	Description
`children()`	Returns a wrapped set consisting of all unique children of the wrapped elements.
`contents()`	Returns a wrapped set of the contents of the elements, which may include text nodes, in the wrapped set. (Frequently used to obtain the contents of `<iframe>` elements.)
`next()`	Returns a wrapped set consisting of all unique next siblings of the wrapped elements.
`nextAll()`	Returns a wrapped set containing all the following siblings of the wrapped elements.
`parent()`	Returns a wrapped set consisting of the unique direct parents of all wrapped elements.
`parents()`	Returns a wrapped set consisting of the unique ancestors of all wrapped elements. This includes the direct parents as well as the remaining ancestors all the way up to, but not including, the document root.
`prev()`	Returns a wrapped set consisting of all unique previous siblings of the wrapped elements.
`prevAll()`	Returns a wrapped set containing all the previous siblings of the wrapped elements.
`siblings()`	Returns a wrapped set consisting of all unique siblings of the wrapped elements.

Wrapped Set - More Ways (1/3)

Examples	Explanation
wrappedSet.find('p cite') OR $('p cite',wrappedSet)	• Find() method lets us search through an existing wrapped set and returns a new set that contains all elements that match a passed selector expression. • For example, given a wrapped set in variable wrappedSet, we can get another wrapped set of all citations (<cite> elements) within paragraphs

Size Syntax

Command	Explanation
find (selector)	• Returns a new wrapped set containing all elements of the original set that match the passed selector expression.
Parameter	**selector** (String) • A JQuery selector that elements must match to become part of the returned set.
Returns	• The newly created wrapped set.

Wrapped Set - More Ways (2/3)

- In addition to finding elements in a wrapped set that match a selector, JQuery also provides a method to find elements that contain a specified string.
- The contains() method will return a new wrapped set that consists of all elements that contain the passed string anywhere within its body content.

Contains Syntax

Command	Explanation
contains (text)	• Returns a new wrapped set composed of elements that contain the text string passed as the text parameter
Parameter	**text** (String) • The text that an element must contain in order to be added to the returned set
Returns	• The newly created wrapped set

Example

Examples	Explanation
$('p').contains('Lorem ipsum')	• This expression yields a wrapped set containing all paragraphs that contain the text Lorem ipsum.

Wrapped Set - More Ways (3/3)

- The is() method returns true if at least one element matches the selector, and false if not.

Is Syntax

Command	Explanation
is (selector)	• Determines if any element in the wrapped set matches the passed selector expression
Parameter	**selector** (String) • The selector expression to test against the elements of the wrapped set
Returns	• True if at least one element matches the passed selector; false if not

Example

Example	Explanation
var hasImage = $('*').is('img');	• This statement sets the value of the hasImage variable to true if the current page has an image element.

Contents

Managing JQuery Chains (1/2)

Examples	Explanation
$('img').clone().appendTo('#somewhere');	• Two wrapped sets are generated within this statement: • the original wrapped set of all the <img> elements on a page and a second wrapped set consisting of copies of those elements. • The clone() method returns this second set as its result, and it's that set that's operated on by the appendTo() command.
$('img').clone().appendTo('#somewhere').end().addClass('beenCloned');	• The appendTo() method returns the set of new clones, but by calling end() we back up to the previous wrapped set (the original images), which gets operated on by the addClass() command. • Without the intervening end() command, addClass() would have operated on the set of clones.

Managing JQuery Chains (2/2)

End Syntax

Command	Explanation
end ()	• Used within a chain of JQuery command to back up the wrapped set to a previously returned set
Returns	• The previous wrapped set.

andSelf Syntax

Command	Explanation
andSelf ()	• Merges the two previous wrapped sets in a command chain
Returns	• The merged wrapped set.

CurioCT - CitiusTech's Technology Q & A Forum

- In case of any questions please log on to http://interct/SitePages/CurioCTTechnology.aspx

THANK YOU