Javascript Now Works

An Introduction to jQuery

Central PA Open Source Conference
2008

Presented By:
Nathan Powell

How'd I get here?

- Social worker
- Interested in Perl/Linux
- Went to work on a Systems Portal
- Web Development
- Linux Systems Administration

What is jQuery?

- jQuery is a Javascript library
- Lightweight
- Unobtrusive
- Created by John Resig
  - Javascript Evangelist with Mozilla.
  - Ported the Processing Language to JS

Some Background

- Originally developed at Netscape
- Deployed in Netscape's browser in late 1995
- Has NOTHING to do with Java
- Submitted and excepted by ECMA for standardization
  - ECMAScript (the standardized version name)
    - two major dialects
      - JavaScript
      - Jscript

Background Continued

- Javascript can appear simple
- Interpreted scripting
- Dynamic typing
- Objects as associative arrays
- Prototype based
- Functional aspects
  - First class functions
  - Closures

Aren't there already Javascript libraries to choose from?

- Prototype
- Moo Tools
- ExtJS
- YUI
- Dojo
- Sproutcore

Why not jQuery?

- What you want to do is already done by another library.
- You hate Open Source
- Your language toolkit/framework prefers another library (Rails/Prototype)
- Widget toolkit focused (ExtJS/Dojo/Sproutcore)

Why jQuery?

- Lightweight
- Pluggable
- Takes cross browser pain away
- Unobtrusive
- Hot selector syntax
- Darling child
- MIT/GPL (You pick)

What is unobtrusive Javascript?

- The last step in getting the crud out of our view
  - Separation of business logic from view logic
  - MVC
  - CSS
  - Unobtrusive Javascript
    - Create included files that contain JS we need
    - Attach behaviors/event handlers on load
    - Graceful degradation

How it looks now:

```
<a href="#" onClick="awesomeFunction(this)">Click Me!</a>
```

How we want it to look:

```
<a href="http://google.com">Click Me!</a>
```

How do I do that?

- Attach event handlers and execute Javascript on document ready
- Use selector syntax
- Manipulate elements from outside the document
- Create applications as though Javascript wasn't available

Show me

```
<html>
<head>
  <script type="text/javascript" src="jquery.js"></script>
  <script type="text/javascript">
    $(function(){
        $('a').click(function(){
          alert("Oh Hai!");
          return false;
        });
    });
  </script>
</head>

</body>
  <a href="http://google.com">Click Me!</a>
</body>
</html>
```

Um, awesome, but what the heck is with the $()

- Alias to the main jQuery object
  - jQuery()
  - $()
  - Used for the initial document ready function
    - $(document).ready(function(){})
    - $(function(){})
  - Used as the wrapper
    - Used for the creation of the wrapped set
      - $('some selector')

Honestly.

- $ is a valid character for JS variables (seriously, look it up)
- If it really bothers you, you can use jQuery() instead (but other kids will make fun of you)
- After a short time, you'll love the brevity.

Introducing Selectors

- Selectors are string identifiers for elements in the page we wish to operate on.
  - All Anchors
    - $(“a”)
  - All Paragraphs
    - $(“p”)
  - All divs
    - $(“div”)
- All return a *wrapped set*.

Selectors Continued

- The selectors are CSS selectors
  - #someId
  - .someClass
  - p.someClass
  - div a.someClass
- Example, adding a notice to the page when validation fails
  - $('div input.someClass').addClass(“error”)

Child Selectors

- We can explicitly select elements that are child elements
  - $(“#someList > li”).size
    - Would give us the number of items in the list someList
- We can go as deep as we need to
  - $(“ol#someList > li > a”)
    - Just give me the links in of the list items of the list

Selectors and Attributes

- We can specify elements with specific attributes
  - $(“input[type=text]”).addClass(“error”)
    - Gives us all input elements of type text
  - $(“img[alt=vacation]”)
    - Give me all the images with the alt attribute of vacation

Selectors, Attributes and Regular Expressions

- It is possible to use basic regular expression type syntax in your selectors.
  - $("a[href^=http://google]")
    - Return to me all the anchors with an href attribute that begins with "http://google"
  - $("a[href$=.wmv]")
    - Return all the links that end in .wmv
  - $("img[src*=nathanpowell.org]")
    - Return all the images that are being linked from my domain

Container Selectors

- We can match elements that contain other elements
  - $(“li:has(a)”)
    - Would match all list items that contain links
  - This is not the same as
    - $(“li > a”)
  - Which would return the links, not the list items.

Positional Selectors

- We can select items based on where they are in the dom
  - $(“a:odd”)
    - Returns every other link, starting with the first one
  - $(“li:first-child”)
    - Would return the first item in each list
  - $(“li:only-child”)
    - Would return lists items that have no siblings
  - $(“tr:nth-child(even)”)
    - Would return the even rows of a table

Positional Selectors Supported

- :first
- :last
- :first-child
- :last-child
- :only-child
- :nth-child(*n*)
- :nth-child(*odd*|*even*)

Selectors Final

- jQuery provides us with a rich selector syntax.
- This syntax is what makes jQuery so powerful.
- WAY more robust than what I have shown you here.
- Custom selectors
- If it exists you can select it.

Manipulating the wrapped set

- Once we have our wrapped set, we of course want to manipulate it.
- jQuery supports many expected methods for operating on the wrapped set.
- Since all methods of this nature return a wrapped set, we can chain methods together.
- Write your own!

Common methods you'll want to use

- .size()
  - $(“a”).size()
    - Returns the count of elements in the wrapped set
- .get(index)
  - $(“ol.someClass li”).get(2)
    - Return the second list item from the ordered list with the class someClass

Common Methods Continued

- .add(*expression*)
  - $(“img[alt=vacation”).add(“img[alt=work”)
    - Add elements matching *expression* to the wrapped set
- .not(*expression*)
  - $(“img[alt=vacation]”).not(“img[href$=.org]”)
    - Remove elements matching *expression*
- .slice(*n*,*n*)
  - $(“p”).slice(0, 3)
    - Return a wrapped set containing the first 3 paragraph elements

Common Methods Continued

- .contains(*string*)
  - $(“p”).contains(“Nathan Powell”)
    - Return wrapped set of paragraphs that contain the *string* “Nathan Powell”.
- .each(*iterator*)
  - $(“p”).each(function(n){alert(n)})
    - Would iterate over the wrapped set and invoke the function for each element.

Still More Methods!

- .attr(*attributes*)
  - $('input#someId').attr('disabled', 'disabled');
    - Disable a form element with an id of someId.
- .css(*name, value*)
  - $('div.error').css('color', 'red')
    - Change the font color to red on any div with the class error.
- .html(*text*)
  - $(div.error).html(“Password already taken”)
    - Replace error div content with *text*

Remember

- I didn't go over all the built-in methods for wrapped sets.
- You can chain methods together since most methods return, either a new wrapped set, or the original wrapped set.
- If there is a method you need, that jQuery doesn't have, you can write your own.

Events

- Unobtrusive
- Cross event model
  - Internet Explorer
  - DOM Level *n* Event Models
- Use with wrapped sets
  - Add or remove event handlers on all elements.
- Intuitive

Event Handlers

- .bind(*event, data, listener*)
  - $('a').bind('click', function(event){alert('Oh Hai!')})
- Events that can be bound are intuitive
  - blur
  - change
  - submit
  - mouseover
  - keypress
  - etc

Toggle

```
$('img').toggle(
  function(event){
    $(event.target).css('opacity', 0.4)
  },
  function(event){
    $(event.target).css('opacity', 1.0)
  }
);
```

Remember

- Lots more to event handling than we covered here.
- Cross browser event handling FTW.!
- Unobtrusive (sorry but it's important to understand)
- Wouldn't work if Javascript's functions were not first class.

Utility Functions

- Utility functions are utilitarian functions supplied to us by the jQuery library.
- Sometimes called commands
- Invoked with the dollar dot notation.
  - $.utilityFunctionName

Flags

- $.browser
  - Browser detection is flawed, avoid at **most** costs.
  - When you **have** to have it, use $.browser.
- $boxModel
  - Size of content with padding and margins
  - Boolean: true for W3C, else false.
- $.styleFloat
  - Accounts for cssFloat/styleFloat in element style property (IE uses styleFloat)

Other useful utilities

- $.noConflict
  - Stops crushing the $ name space when other libraries are using it (See Prototype)
- $.trim(*string*);
  - Trim leading and trailing whitespace from string
- $.each(*array, function*)
  - Iterates over *array* invoking *function* for each
  - Do not confuse with wrapper method

$.grep()

- $.grep(*array, function, invert*)
  - $.grep([1, 2, 3], function(n){return n < 3}, true);
    - Iterates over the passed array, invoking the function for each iteration, returning collected values. Inverts call back value.
    - This would return 3.
    - Unix users beware, doesn't require a regex (though you can certainly put on in the *function*).

Others

- $.map(*array, function*)
  - Return *array* after iterating over it and passing each element to *function*
- $.inArray(*value, array*)
  - Search for *value* in *array*
- $.unique(*array*)
  - Return only unique elements of an array
- $.getScript(*url, function*)
  - Dynamically load a script file

Ajax

- One wrapped set method
  - $('wrapped_set').load(
    '/resource.rb',
    {paramater: list},
    function(){}
    );

Ajax Continued

- Or as you might expect
  - $.get(resource, parameters, function(){})
  - $.post(resource, parameters, function(){})

Plugins

- Pluggable
  - http://plugins.jquery.com/
- Plugins
  - The PNG fix you have implemented 900 times
    - http://plugins.jquery.com/project/pngFix
  - Interface with CouchDB
    - http://plugins.jquery.com/project/jqcouch
  - Add Google Charts
    - http://plugins.jquery.com/project/gchart

Resources

- Web
  - The main site
    - http://jquery.com/
  - The tutorial section
    - http://docs.jquery.com/Tutorials
- Books
  - jQuery in Action
  - Javascript the Definitive Guide