# JavaScript Polls

## Web developers?

## used (or tried using) JavaScript?

Why?

## experienced cross browser issues?

## resorted to requiring a "standard" web browser for intranet sites?

I have.

## shied away from JavaScript, even relenting to round-tripping?

Why?

## Use without fully understanding (copy & paste)?

I did.

# jQuery Polls

## Use jQuery?

## Familiar with CSS selectors?

Learning jQuery will be extremely comfortable for you folk.

## Use AJAX?

jQuery makes this a breeze.

# Reasons for Using JavaScript?

## Client-Side Event Handling

Normally, we'd use event-handling declarations via JavaScript (onclick, onload, etc.).

## Avoid Pop-Up Windows with Floating DIVs

One of the first "advanced" uses I had for JavaScript was the ability to avoid pop-up blockers when trying to legitimately improve the user experience with a dialog box.

## Dynamically Hide & Unhide Items

Of course, using JavaScript to hide and unhide items has been a very common need.

## Highlight Fields (Errors, etc.)

Being able to do client-side validation was an early requirement for making responsive Web-based apps that didn't necessarily require a round-trip. Highlighting fields in error was part of this requirement.

## AJAX Calls

Early on, making direct AJAX calls via JavaScript was a necessary evil (before third-party JavaScript libraries eased the pain, albeit inconsistently).

## What Else?

Did I miss anything? It's been a while since I used JavaScript exclusively, so I may have missed another common use case.

# Why jQuery?

## Easily Create Interactive Sites

Normally, we'd use event-handling declarations via JavaScript.

## No Need to be a JavaScript Expert (Gateway)

## Great to Learn From

You can learn a lot about JavaScript (and best practices) from looking at the (non-minified) code.

## Cross-Browser Abstraction (HUGE!!!)

Although some edge cases aren't always handled correctly, it works 99.5% of the time.

## Fluid Syntax

As you’ll see, because each jQuery API call returns the wrapped set of selected objects, it’s easy to chain multiple API calls.

## Promotes Unobtrusive Code & Separation of Concerns

Extending the benefits of CSS for separating styling from markup, jQuery also makes it easy to separate client-side code from your markup.

## Use for Any Web App

Since this is client-side technology, and AJAX use does not care how it's supported on the server side, it does not matter what technology is used for building a website. jQuery can be put to full use on the client side.

## Large Market Acceptance & Microsoft Support

Microsoft now officially supports jQuery (thanks to their use of it in ASP.NET MVC). In fact, and it replaces the MS AJAX Toolkit in many respects, and Microsoft is even contributing to it now (ex: templates).

## Best Community Support

## Tremendous Plugin Ecosystem

Thousands, including jQueryUI.

## Great Documentation

Best documentation I've seen for an open source project, bar none. Reflects passion for the product.

# Making Use of jQuery

In only a few steps, you'll be able to put jQuery to full use. It's amazing how much functionality they've stuffed into such a small footprint.

## Single .js File

Only a single JavaScript (.js) file is required to use jQuery. There are other plugins that we may often use, but for the basic functionality we're learning today, everything is self-contained in this single file.

## Minified and Full Versions

The minified version is what should be used in production. White space and comments are removed, and variable names shortened. The full version is good for debugging and great for learning.

## Content Delivery Networks (CDNs)

We can also protect against a CDN being down.

## <script> Tag

On our page, we'll need a <script> tag wrapping any calls we make to the jQuery API (and all other JavaScript we'll be using). It's recommended that we offload this into its own .js file, and include it in the same way we include the jQuery.js file (\*after\* the jQuery reference). There are exceptions that we'll discuss.

## Intellisense for Visual Studio

Microsoft provides a file we should reference in each file (web page or .js file) where you'll be writing jQuery code. This will allow Visual Studio provide the kind of Intellisense we're used to.

# Selectors

## By Tag (<p> tag)

In this example, only <p> tags are selected. This item contains the only <p> tag on the page.

## By Id (id="byId")

In this example, only an object with an id of “byId” is selected. This is the most efficient selector, because it directly targets a single object.

## By Class 1 (class="byClass")

In this example, only objects marked with a class of “byClass” are selected. As you can see, two objects on this page have this class. It's extremely common to assign one or more classes to an object for the purpose of manipulating it through jQuery in several different ways.

## By Class 2 (class="byClass")

In this example, only objects marked with a class of “byClass” are selected. As you can see, two objects on this page have this class. It's extremely common to assign one or more classes to an object for the purpose of manipulating it through jQuery in several different ways.

## Multiple (.byClass, #byId, #byMultiple)

In this example, several selectors are used to wrap up a set of objects for jQuery to manipulate at once. If multiple operations are to be performed on a wrapped set of objects, it's efficient to first assign the set to a variable, so that the selection only needs to be performed once.

## By Descendant (<span> within <li>)

In this example, any <span> objects that fall within (children of) <li> tags are selected.

## By Direct Child (<span> within <li>)

In this example, only <span> objects that are direct children of <li> tags are selected. The difference between this selector and the previous one is that the first object is not selected in this case, because there is a child <div> tag for the <li> tag, and <span> is actually a grandchild.

## More Information on Selectors

# Events

## $(document).ready

The $(document).ready event is where all the magic happens. It’s triggered once the DOM is completely loaded and ready to be referenced, but before extra resources, such as images are retrieved. A shortcut you can use is simply $(function).

## Shortcuts (.dblClick)

There are built-in shortcuts for many common events in jQuery. But if you want more control, you can manually bind, as in the next example.

## .Bind / .Unbind

Directly binding to events allows for more control. You can “bind” and “unbind” as needed, as shown in this example. This works well for objects that exist on the page upon creation, but not for objects added dynamically on the client side.

## .Live / .Die

Unfortunately, when we bind to events (or use a shortcut for the same purpose), any objects added dynamically that would fall under the same selector wrapper would not be bound. Via the “live” API, we can bind future objects, because the event handling is delegated to the full document.

## .Delegate / .Undelegate

In a recent release of jQuery, late binding of events was made more efficient via the “delegate” API. With this API, we can specify the delegate instead of falling back to the main parent, the document. In this example, we’re specifying a parent object as the delegate, so the event only needs to bubble up one level to be caught and handled.

## .On / .Off

In the next version of jQuery (1.7 – currently in beta), the preceding sets of binding options have been superseded by the “on” and “off” overloaded APIs. Based on the parameters passed to these, we get the same effect as with “bind”, “live”, or “delegate”.

## Suppressing Default Behavior

Although you can suppress default behavior on objects like submission buttons by returning false from the event handler, this may not have the desired effect. It would actually prevent events from bubbling up. Using the jQuery "preventDefault" method will suppress the default behavior for the current event. The "stopPropagation" method will also prevent bubbling up as well.

## .hover

## More Information

# DOM Manipulation

## Adding and Removing Classes (.addClass, .hasClass, .removeClass)

Classes are not only used for adding and removing styles to HTML objects. jQuery makes it easy to use classes for storing state as well. In this example, we're adding a new class, “modified”, to track that we've changed the value of a button's text.

## Adding and Removing Web Elements (.append, .remove)

Dynamically adding and removing HTML elements is a breeze with jQuery. But don't forget that the wiring of events don't come without some forethought.

## Querying and Manipulating Styles (.css)

In addition to adding and removing classes, jQuery makes it easy to manipulate CSS.

## Querying and Manipulating Attributes (.attr)

jQuery also makes it easy to query and update HTML attributes.

## Manipulating the Text and HTML of Elements (.text, .val, .html)

## Intro to AJAX

jQuery makes AJAX easy, but its functionality is a topic unto itself. We'll just introduce it here.

## More Information

# What's Next? / Summary / Resources

## What's Next?

Just start using it! Experiment with Cody Lindley's Selector page. Play around with its AJAX capabilities. Google (Bing) the tons of resources out there. And research the resources curated by Elijah Manor.

## Summary

jQuery is the most fun I've had with an API in over 20 years (since FUNCky.lib for Clipper). You can feel like you have total control over your web pages, and can get so much done in such a short period of time, just by touching the surface. I don't even know half of the API, and can still be productive.

## jQuery Documentation

## jQuery Selector Documentation

## jQuery Event Documentation

## jQuery Manipulation Documentation

## jQuery Tutorials

## Cody Lindley's jQuery Selector Page

## Elijah Manor’s Links

## Mark Freedman’s Links