Why JQuery?

Vishwak Lab's white paper

19th February 2009

G.Venkat (GV)
Vishwak Solutions Pvt. Ltd

Contents

Purpose of this document

Vishwak's Web Labs frequently evaluate and select latest tools and technologies to improve our internal efficiency and productivity. As part of this continuous process we recently evaluated few of the newer technologies in this case **JQuery**. This document scope is to share the observations.

The document provides a brief about the observations recorded during the evaluation the team had carried-out on the JQuery v1.3.1. The initiative has been to evaluate how efficiently we can use the JQuery with-in a boundary of certain parameters the analysis has been carried-out under.

The document should not be treated as a 'Help' in fixing the issues while using the JQuery. The general audience for the document is the technical teams and solution-architects to help them the best-way to use JQuery.

Disclaimer: *These evaluations are purely the opinion of the author(s) of this article based on their observations during the evaluation of the JQuery; Vishwak doesn't necessarily endorse any of these observations. This article is shared purely as a knowledge sharing initiative for the general audience.*

Why use JQuery

JQuery is a small library based on few very simple intuitive principles. Working with RAW JavaScript or DOM Programming does bring-in its challenges while handling the browser-specific quirks. JQuery provides quiet a few utilities to help working in browser agnostic environments. JQuery library strikes right balance between size, feature set and ease of use.

JQuery would help bring-in considerable productivity gains and it's easy to learn and work. This tool has the potential to bring-about quiet a change in client side development with complex UI with front end logic in JavaScript library.

You can load the JQuery library in any Web page by adding HTML SCRIPT element to “src” attribute which should utilize the path to where the file is located on your server.

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

It has small library with Document Object Model (DOM), Quick access to AJAX, without a lot of bloat and some basic animations. The basic JQuery is just 15K and JQuery holds a strong and flexible mechanism for adding methods and functionalities which bundled as plug-ins. It supports CSS 1-3 and basic X-Path; works with Firefox 1+, Internet explorer 5.5+, Safari 1.3+ and Opera 8.5+.

Before we get into samples, there is one basic and simple JQuery library which is ***.Ready().*** Everything in the JQuery utility is HTML and DOM. So to process this, the document should be loaded before we use JQuery features. This can be done using ready event. “Ready” is the first event fired once the document loaded.

```
$(document).ready(function() {

// Your code goes here

});
```

It elegantly ***finds*** and manipulates HTML elements using very nice “***selector***” API which allows any JQuerian to query for HTML elements and then “***apply***” commands to them. It can be chained very easily so that the result of one command can feed into another. For example, the sample one line code finds all ***<div>*** elements within the page which has a CSS class as ***“slide***” and then animate it to slowly disappear.

$(“div.slide”).slideup(‘slow’).addClass(“removed”);   that's how easy it is.

What you can get from JQuery

DOM element selector

You can select DOM elements using JQuery selectors which allow you to do things faster and efficient. JQuery uses CSS 3.0 syntax with some extensions to select either single or multiple elements in a HTML document. With CSS syntax you can select elements by ID, CSS class, attribute filter, relationship to another element or filter conditions which can be chained. In a normal JavaScript, it is very difficult to select all 2nd column LI elements in a HTML. In JQuery you can achieve this using a simple line of code like ***$("#liEntries li:nth-child(2)").***

All the selectors can be added as a JQuery object which is called ***wrapped set***. This will be stored in an array-like structure which we can iterate thru the wrapped set like array or select individual element by indexer. JQuery functions can also be assigned to all the selected elements.

Wrapped set operations

The real power of JQuery ***wrapped set*** comes forth when we apply JQuery operations to all DOM selected elements. ***JQuery.fn*** has more than a 100 functions that can operate on matched set and allows manipulating and retrieving selected DOM objects in a batch. If we want to apply styles to alternate rows or any specific rows it is very easy to do that dynamically. Class names can also be dynamically applied. Few methods allow us to go for "***get***" and "***set***" operations for ***.CSS().*** Attributes can also be retrieved and set using .***ATTR()*** or retrieve or set a value using ***.VAL(), .HTML()*** or ***.TEXT().***

An element can also be ***cloned*** from the HTML document and injected to another document either to pre-pend or append or be reversed. Wrapped set operations are also chainable using methods into single command which means it can be taken once and used many times. It can be expanded using ***.FIND(), .FILTER()*** or ***.ADD()*** methods.

Event Handling in JQuery

Events are handled very efficiently in JQuery. In JavaScript DOM manipulations for AJAX are done using ***Async*** calls which cause issues while working with different browsers because of DOM implementation. JQuery provides very easy mechanism for binding and unbinding event handlers and provides a normalized event model for all supported browsers. All events are called in the context of the element that caused the event and they receive a fixed up and browser normalized event object that is consistent.

To accomplish that we get selectors, a whole slew of operations that can be performed on the wrapped set, DOM normalization for most browsers, Ajax functionality, a host of utility functions for object/array manipulation and a number of basic effect functionality.

JQuery also supports JSON for any HTTP callbacks.

Easy plug-ins

JQuery is a language with DOM extension library with tight focus on core functionality. For everything else JQuery provides an easy plug-in API for almost all operations you can do with DOM operations.

JQuery API allows extending the core JQuery object operations simply by creating a function and passing parameter to JQuery wrapped set. Using this method allows you to use the plug-ins which will receive a wrapped set and can operate and participate in JQuery chaining. If no plug-ins won't satisfy your requirement, you can write your own plug-in which gives you more comfort. This simple and powerful plug-in model is easy to learn and work.

JQuery with ASP.NET

JQuery is also supported in Visual Studio .NET with fairly full featured ***intellisense*** support. You need to have VS.NET 2008 with SP1 and this will work only with VS.NET 2008. SP1 allows parsing of the script file, but you still need to mark up the JQuery script file in order to get useful Intellisense. To fix this Microsoft recently released a jQuery Intellisense file that provides detailed documentation for JQuery plus a hot fix specific to JavaScript Intellisense that can automatically utilize this Intellisense file as long as you follow a specific naming convention.

The way this works is that you can download the JQuery Intellisense file and name it with the same name as the script with a vsdoc.js postfix like JQuery-vsdoc.js. Place this file in the same folder as the JQuery.js script file. If you then reference JQuery.js via <script> tag, Script Manager or script reference tags (/// <reference path="~/scripts/jquery.js" />), Visual Studio looks at the –vsdoc.js file to get Intellisense information. With the file in place you get full Intellisense all the way down the jQuery call-chain as well as for custom plug-ins (although there it stops unless you mark the plug-ins with VS style help comments). The Intellisense script file is basically a marked up version of JQuery that is full of Visual Studio style help comments and the hot fix provides the mechanism for Intellisense to link this –vsdoc.js file without having to include and hide the Intellisense script file.

JQuery with other libraries

JQuery library and virtually all of this plug-ins are wrapped within the JQuery namespace. “Global” objects are stored inside JQuery namespace and there shouldn’t be any name conflict with Prototype or Moo Tools or YUI.

You can override the default “$” function using ***jQuery.noconflict()*** at any point JQuery and other library loaded like

```
<html>
<head>
    <script src="prototype.js"></script>
    <script src="jquery.js"></script>
    <script>
        jQuery.noConflict();
        // Use jQuery via jQuery(...)
        jQuery(document).ready(function()
        {
            jQuery("div").hide();
        });
// Use Prototype with $(...).
        $('HTMLid').hide();
    </script>
</head>
<body></body>
</html>
```

Or there is also another option to make sure it won’t conflict with another library, you have something like example here where “j” has been added along with “$”

```
<html>
<head>
    <script src="prototype.js"></script>
    <script src="jquery.js"></script>
    <script>
        var $j = jQuery.noConflict();
    // Use jQuery via $j(...)
        $j(document).ready(function()
    {
        $j("div").hide();
    });
    // Use Prototype with $(...),
        $('HTMLid').hide();
    </script>
</head>
<body></body>
</html>
```

If you include JQuery before libraries, you can use “JQuery” instead of “$” and you can use other library like prototype with “$” without any conflict. You can also assign “JQuery” to a variable, or to a function like (function ($) { /* your code */}) (JQuery) . The only downside of using function is you can’t use the prototype methods.

When compared to other libraries, the usage of JQuery libraries and its functions is efficient and it will not conflict with other libraries you are using. The option of using multiple libraries without any conflict is handled very efficiently in JQuery.

Plug-ins and authorizations

As mentioned earlier you can add methods and functionalities and bundle them as a plug-in. The way plug-ins is handled in JQuery is very efficient. Here are a few things we need to consider when writing a plug-in.

Always name the plug-in as “JQuery.” [with file name] like ***JQuery.animate.js.,*** this will give you more flexibility for maintenance. All new methods should be attached ***to JQuery.fn*** object. Etc..

When we are using multiple public static methods we should add them to ***JQuery object*** to avoid namespace cluttering.

As a good practice, you should avoid passing too many parameters to a function in a plug-in. As a flexible option you should design the plug-in to accept a URL, name as string, size as numbers and *Boolean* with optional parameters. It is good practice to have a default value for a optional variable. If needed we can over-write it always.

Conclusion

Though everything looks good in JQuery, there are some downsides like - number formatting, date formatting and window support along with other native JavaScript support. It is not a perfect tool to solve your entire problem with JavaScript and DOM problems you are facing. Butwe am sure; it will not give you any showstoppers. For non-DOM related functionalities you might need to write some helper functions to support. Another issue we see is versioning and possible interference with other libraries which uses “$” like “Prototype” which has many similar functionality like JQuery. But Prototype is not backed with CSS and XPath selectors and with the ability to chain methods to do interesting functionalities in concise manner. If you use the library as downloadable, then you will end up in renaming files and references frequently as and when the library is updated. If you use fixed one then you may end up versioning issues.

Our internet search result did not throw-up us any negative feedback and who ever uses it seem to be comfortable. But you can judge for yourself.

References

1. You can find some JQuery help controls in the below links:
   http://www.west-wind.com/tools/westwindajaxtoolkit/docs/?page=_2hh0s4zli.htm
2. Also you can find more documents and help in:
   http://www.JQuery.com
3. Nice example on all JQuery methods in XML format:
   http://jquery.com/api/data/jquery-docs-xml.xml
4. JQuery with .NET
   http://weblogs.asp.net/bleroy/archive/2008/09/28/jquery-now-officially-part-of-the-net-developer-s-toolbox.aspx
5. JQuery intellisense in VS.NET 2008:
   http://blogs.ipona.com/james/archive/2008/02/15/JQuery-IntelliSense-in-Visual-Studio-2008.aspx
6. Integrating Sharpoint and JQuery:
   http://weblogs.asp.net/jan/archive/2008/11/20/sharepoint-2007-and-jquery-1.aspx
7. You can download JQuery from:
   http://docs.jquery.com/Downloading_jQuery

About the Author

G.Venkat (GV) is a Program Head, Vishwak Web Standards team with 12 plus years of experience in web technologies. Worked on various web projects with Microsoft, he has a good knowledge on CMS and content publishing platform in web and mobile portals. Works closely with Microsoft, MSN Web and MSN Mobile groups. GV is also involved in designing and implementing custom web applications.

About Vishwak

Vishwak Solutions is a Global Innovator in application development and maintenance focusing on sophisticated web portals, mobile portal applications and enterprise solutions.

Microsoft®
GOLD CERTIFIED
Partner

Vishwak as a "Microsoft Gold Certified" partner a tier one portal solutions company adopts to high class web portals with latest technology like .NET for business logics and XHTML/CSS/JQuery/JS for front end and also pioneer in Mobile Web technologies.

With deep domain experience in developing and managing leading Media portals Vishwak brings sophisticated tools to ease your portal development project. Whether you are in Media, Entertainment or Publishing Vishwak brings in world class portal technology and process. In the digital economy this helps you to differentiate, extend your brand value and enhance revenue opportunities.

We are always closer to you with offices in Redmond (Washington, USA), London (UK) apart from our global delivery center in Chennai (India)