



Hanselminutes

Hanselminutes is a weekly audio talk show with noted web developer and technologist Scott Hanselman and hosted by Carl Franklin. Scott discusses utilities and tools, gives practical how-to advice, and discusses ASP.NET or Windows issues and workarounds.

Text transcript of show #126

August 14, 2008

jQuery with John Resig

Scott chats with John Resig about how he developed jQuery, how it performs, and where he thinks it's headed.

(Transcription services provided by [PWOP Productions](#))



Our Sponsors

 **telerik**
deliver more than expected
<http://www.telerik.com>

 **nsoftware**

<http://www.nsoftware.com>

NET 
DEVELOPER'S JOURNAL

<http://dotnet.sys-con.com>





Lawrence Ryan: From hanselminutes.com, it's Hanselminutes, a weekly discussion with web developer and technologist, Scott Hanselman, hosted by Carl Franklin. This is Lawrence Ryan, announcing show #126, recorded live Friday, August 8, 2008. Support for Hanselminutes is provided by Telerik RadControls, the most comprehensive suite of components for Windows Forms and ASP.NET web applications, online at www.telerik.com. Support is also provided by .NET Developers Journal, the world's leading .NET developer magazine, online at www.sys-con.com. In this episode, Scott talks with jQuery creator, John Resig.

Scott Hanselman: Hi, this is Scott Hanselman and this is another episode of Hanselminutes and I'm sitting down today with John Resig, the creator of jQuery and JavaScript evangelist for Mozilla. Thanks a lot for taking the time to chat with me today.

John Resig: Well, thanks for having me.

Scott Hanselman: So, you brought jQuery to the world in January of 2006 at Bar Camp 2006, is that right?

John Resig: Yup, yup.

Scott Hanselman: That wasn't very long ago. It just seems like there's so much stuff on the Internet that's happening so quickly that it feels like jQuery has always been around, but I'm looking at my calendar and that was not very long ago.

John Resig: Yeah, a lot could happen in just a couple of years.

Scott Hanselman: Yeah. Was this something that came to you as a *Eureka!* moment or had you been working on this in the quiet for a long period of time before you brought it out?

John Resig: I guess in the quiet. I guess traditionally I'm pretty much a hacker at heart. I always have a bunch of these little tie projects that I like to work on in my spare time and a couple of these tie projects were just tools that I had built that sort of came together in the form of jQuery; so I was originally working on an animation toolkit that I was just playing around with and then I was off working on a selector library for doing toolset selectors from JavaScript and this huge tool came together and I introduced this little framework, but it was more of an amalgam rather than doing something from the outright.

Scott Hanselman: Hmm. A lot of people say that it fits together in such a natural and such a comfortable and maybe fluent is a cliché, but in a very fluent way. When did it come to you that you should have the jQuery object be returned from every method and then chain things together?

John Resig: One of the things I wanted was to have the shortest impact possible for doing things within the documents; for manipulating, for attaching events, for modifying attributes, all this, I wanted to have virtually a simple code. One hack that I had for doing this was chaining, returning the object itself. If I could do that, then it would allow the short set of syntax to be even shorter. I guess it was more incidental and it ended up becoming actually usable and interesting because it is more just that I wanted it to be really terse.

Scott Hanselman: Really? So, terseness was the goal, not fluency. You didn't want it to read like English.

John Resig: Not explicitly. I mean obviously I wanted it to still be understandable. Obviously, you can pretty easily sacrifice usability when you want to shoot for terseness. There are a lot of languages that are, you know, did an example of that. So, I didn't want that to occur. Instead I wanted to see how far you could get without having encrypted methods while you're still having things that made sense.

Scott Hanselman: Now, when you chain these kinds of things together and you say something like `$div.add` and then you start kind of popping back and forth between strings that have semantics and methods that have semantics, how did you find the balance between kind of the semantics that were tunneled into a string like saying `P.` or `div.test` versus breaking that out into methods because I suspect there's a lot of string parsing going on because you've got semantics for like CSS selector strings and things.

John Resig: Yeah. Essentially, there's certainly a lot of method overloading going on in jQuery, there's no doubt about that even down to argument level there, but I think this is something that sort of grew as the API developed, being able to define that -- if a selector would ever come in, then it should be a string and it should also be able to accept several types of argument. What happened is that internally in jQuery, there is a number of methods that's starting to be constructed that could accept any of the generic arguments such as a CSS selector string or a DOM element or an array of DOM elements and in this variety, the output would always be the same. The output would always be an array of DOM element. So, that way you could have these methods that could accept any of these inputs and it would still have the same effective behavior.

Scott Hanselman: Can you get in trouble with jQuery? I mean is the syntax so open and flexible that you can write something that looks like it ought to work but it just doesn't because the underlying engine just didn't see it coming?



John Resig: I think we've worked out most test cases pretty well. I mean we have a pretty strong test suite here where we do check for potential weirdness. So, in most cases now, I think we're quite solid.

Scott Hanselman: Now, what do you do your development in? I'm a Windows guy, although I've got a couple of Macs in the house, but I really find that I'm pretty much stuck in Notepad and Firebug is where I live when it comes to JavaScript development. It just doesn't feel like on the Windows side that there's a full-pledged world for developing this kind of stuff in IDE. Aptana is pretty good, the new version of Visual Studio is pretty good, but JavaScript for me still feels kind of loopy-goopy, which makes sense given the dynamicism of the language, but what do you use for your development of JavaScript applications?

John Resig: I'm very similar. I mean I use Firebug a lot and some other things with my development, but I don't use any IDEs in particular. I use a couple of different editors. I use Vim. I also use an editor for OSX called EasasEdit. I like that because it allows you to collaborate in real time with other people.

Scott Hanselman: Oh yeah, totally.

John Resig: It's a pretty simple editor, but the collaboration is quite nice and I've used that on a number of occasions. I'm not really a bells and whistles kind of guy. I was just using straightforward tools.

Scott Hanselman: Yeah. You do a lot of peer programming though that way?

John Resig: It depends on the situation. The situation where I find it to be most useful is in pure web development. If I'm working on a webpage and like there's one person editing the CSS, there's one person editing the JavaScript or two people editing a CSS, that way you can all pull these files in together in real time and be able to work very dynamically in that manner.

Scott Hanselman: I get the impression from reading your blog and from looking at the projects that you're very much kind of the *top-down* developer that, really, it's the result that matters and you visualize what you want the end result to look like and then you kind of push down make sure that the underlying library support that in an elegant way. That's just, again, gleaming this from reading your projects and you've got all these great examples where, you know, little stuff like the Spark Lines library, where the way that you implement this as simple as possible at the very top level. Is that how you work? You think about the example first? You think about the core first? Do you go from the inside in or from the inside out?

John Resig: I guess it depends on the project. I could think of two projects in particular where I worked from the outside in. One was where I recently ported a visualization programming language called processing.

Scott Hanselman: Yeah.

John Resig: It's implemented in Java typically and I ported that to JavaScript. In that case, I worked top down. I took a whole bunch of demos that works and I go through one by one and I made each demo work and in order to do that, I implemented a number of aspects, a parser for the programming language, various aspects of the API just going through piece by piece and getting more and more demos to work. I wouldn't say that that's the best way to develop at least in a holistic sense because there's the inevitability of the gaps in your development; but you can work around this though...

Scott Hanselman: Well, the reason that I asked it is that there's thoughtfulness to jQuery as an example. Even the code for the processing JS is so clean that it makes me wonder does that cleanliness come from a real clear understanding of where you want it to go from the inside out where you visualized the design and you knew that it was going to turn out that way or was it driven by requirements of the end result?

John Resig: I guess typically it's more in terms of by requirements.

Scott Hanselman: Really?

John Resig: I like to see things grow organically as I develop them and kind of mold it as it comes upon a result. I enjoy that style of development at least.

Scott Hanselman: Yeah. Did you do a lot of refactoring? Is it very iterative?

John Resig: Yeah, yeah, very much so.

Scott Hanselman: One of the neat things about jQuery and one of the things that I think made it take off was this notion of a plug-in, but because JavaScript is so loose, is so flexible, it's a pretty loose definition. A plug-in in jQuery is just a method or series of methods that also return the jQuery object. Is that right?

John Resig: Yup.

Scott Hanselman: I was just wondering why it's been successful because I was a prototype and script.aculo.us guy for the longest time and then just something about jQuery, I don't if it was the smallness of it or the richness of the plug-in library, I'm trying to



get an understanding of why is it that this particular library has taken off.

John Resig: A big thing for me that sort of really got me is simplicity of just rules in so many ways. It makes development simpler, you're able to leverage it more, it's easier to learn, it's easier to document, in so many ways, a more optimal experience. So, when you optimize for simplicity in your development and in your code, the result just ends up being better all around and this carries on into plug-in development. If you make adding in plug-in and developing plug-ins really simple, there's going to be a lot of plug-ins, which is the case with jQuery. There are hundreds and hundreds of plug-ins even to the point where there's a lot of duplication, but I don't particularly think that this is necessarily a bad thing because when it's that simple for a developer to get in and start extending it in that manner, there's a lot of healthy competition. You really see excellent code start to rise to the top, which I think is great.

Scott Hanselman: Hi, this is Scott Hanselman with a word from our sponsor. Do you know how to build web 2.0 AJAX applications with web 1.0 components? You really can't. Do you want to do the next generation web applications? You'll need next generation components just like the ones that our friends at Telerik have got, they are RadControls for ASP.NET AJAX. It is a huge pack of web controls built on top of ASP.NET AJAX. That will add previously impossible performance and interactivity to your next project. The new controls mirror the AJAX API from ASP.NET so development is really straightforward. The client scripts are shared, so loading time is not a problem. You just set a couple of properties and you will be able to automatically bind to web services for a really efficient operation. The new RadEditor from ASP.NET AJAX Telerik loads up to four times faster than before and the new RadGrid handles thousands of records in just milliseconds but as always it's best to try for yourself so you can visit telerik.com/aspnetajax and download a trial. Thanks a lot.

What are some of the plug-ins that you find yourself turning to that are ones that you didn't write? I mean do you use interface? What are the plug-ins that you turn to?

John Resig: There's a bunch. Definitely for user interface components, I'm almost always turning to plug-ins. I have very little desire to write another Accordion or to write another tabs component. Basically, a lot of those now have been integrated back into jQuery UI which is a project of ours where we maintain UI components. So, I frequently refer back to jQuery UI. There's also a number of other plug-ins. There's one called a Live Query which is quite cool and makes the CSS queries in jQuery work live. So, if you add new information to the document at any point like from an AJAX request or what have

you, it will rerun the query again and run live, which is quite powerful. Another plug-in that I use a lot is called the form plug-in. It's able to take any existing form and turn it into an AJAX form. I use that one all the time. It's just immensely useful. You can just drop it in and instantly you've converted in a way that it will degrade gracefully your form into something that's quite sexy.

Scott Hanselman: Yeah? Were there any of these plug-ins or where while you're developing jQuery or watching the jQuery community kind of grow up around it, were there any of these that caused you to say, "Wow. I didn't think that they would be able to pull that off and they did?"

John Resig: There are big projects. I mean just seeing large scale, you know, just people take on these large – there used to be this plug-in called Interface that is superseded by jQuery UI, but that whole string of development was quite fascinating to watch and I think it's pretty cool to see people develop these large, I guess, some projects of jQuery and really see them take a life of their own.

Scott Hanselman: At what point do you think that JavaScript as a language and the browser support got to the place where jQuery was possible? Because JavaScript was kind of a mess, not as a language, but the implementation of it across the different browsers was really pretty bad for a long time. Was it IE 7 and Firefox 2 that finally kind of pulled it together? At what point did this become possible?

John Resig: Yeah. At least what interested me about the development -- I've been doing web development for a long time now, but doing the front end development I really started to focus on back in about 2004 and that was about the time at which Firefox was released. I think that was at least for me a big turning point in my personal development, but I guess the big mountain was the release of Firefox the release of Firebug. Having Firebug in the toolset provided is actually like being able to have that console that you can dip in and play with JavaScript and being able to profile JavaScript. It is deceptively simple and just incredibly powerful. I mean those dramatically changed how I worked.

Scott Hanselman: What direction are you going to take jQuery and the team and take jQuery for 2.0? You've been kind of going forward with .01 type, 1.5.1, 1.5.2. Do you have some big master plan for 2.0? What would justify a major release like that?

John Resig: The biggest release that we're working on right now is jQuery 1.3 and this is as per usual we're focusing on performance significantly so. We're looking at a number of areas that are very critical and trying to improve them. The thing is that typically if you were to look at let's say a 2.0 release, which we really aren't focusing on at the moment, that



would tend to imply that there would be dramatic changes, so that would mean that there would be I would say, for example, dramatic API changes. There are places in jQuery where the API could probably use some polishing and maybe more descriptive names or what have you, but I don't think making those changes is that official enough to warrant breaking a significant number of existing codebases. I think it's far more important at this point to promote stability, performance and quality than to have any large overhauls. I think what's important here is that the plug-in architecture for jQuery really makes this possible because if there wasn't a way to easily snap in this new functionality somebody would be forced to be continually adding and just instantly growing and growing.

Scott Hanselman: I know that you guys really focus a lot on performance. I mean 1.2.6 was -- I misspoke earlier and I said 1.5, I was thinking about jQuery UI, but 1.2.6 has some major performance improvements. How are you figuring these improvements out and I'm curious how often you get into a situation where you have to write an entirely separate chunk of code to make it fast on one browser versus fast on another, a whole different branch to say, "Gosh, we can make it 100% on Firefox, but if we do it's going to suck on IE," or whatever?

John Resig: Yeah. Okay, so, the first part is figuring out where to improve. We have a bunch of tools that we use including Firebug, but I wrote a tool recently for profiling jQuery applications. You stick this little library into your webpage using jQuery and after the page finishes running, it dumps a full report out and it tells you exactly where all your time is being spent. This tool has been really eye opening and we've been able to use this in a number of places. We can insert it into major websites that use jQuery and get a feel for what is hurting them the most. For example, in 1.3 we're going to be making optimizations to our DOM modification code, so being able to append, insert before, insert after and we've made some dramatic changes because we found that to be one of the largest bottlenecks in application. So, that improvement is going to be quite, quite good. What was the other half of the question?

Scott Hanselman: Well, I was wondering if you ever got yourself in a situation where you found that you could make something 100% faster or some percent faster, but if you did, it would negatively impact some other browser and then you ended up basically with a switch statement and a different branch to say, you know, if IE, do it this way; else do it that way.

John Resig: There are cases where that happened and it is quite tricky. Generally speaking, we try to write our code in a very just agnostic way that does rely upon a particular browser in a certain

way. So, whenever we can, we try to determine ways in which we can make a smart guess about which will be faster. We'd prefer not to have one browser benefit at the sacrifice of another. I don't think we have ever had a case where it's been really dramatic. I mean we may have been able to get like a dramatic improvement in one browser to a slight degradation in another but I don't think we've ever have something quite extreme. It's hard because you have to take it really on a case by case basis. One thing that we are looking at here for 1.3 and that's going to be quite new and quite exciting is that we're working to remove all uses of browser user agent sniffing. This is something that virtually all modern JavaScript libraries use and saying if Internet Explorer do this or if Firefox do this and we worked out some code that is going to make it possible that we don't have to do that anymore in jQuery. It's a lot of code, but we think it will be really worth it and it's going to help to promote some good practices to web developers as a whole. So, we're really quite excited about that.

Scott Hanselman: Wow. So, this is a technique that other people are going to be able to use. It's going to raise the water level for everybody.

John Resig: Yeah, yeah. I mean an important part of this is what's called feature simulation and doing -- so, essentially, having a feature and seeing that it behaves like we expect it to; if it doesn't, then doing a fall back that it's something that works properly. What's interesting about this is that you effectively remove the need for sniffing at that point, for making a blind decision about a browser and generally the code just becomes much, much better.

Scott Hanselman: Wow. I really think it's great the way that you guys really do this development out in the open particularly with the test suites, I mean certainly open source projects that should be doing their work in the open, but you're so detailed in your differences during your test suites and showing here are the eight browsers we're testing on and here's the diff between this version and that version. Some of the improvements in speed are just obscene like between 1.2.3 and 1.2.6, some of the changes in JavaScript map like one of them was Opera 9 went from like two-and-a-half seconds to 95 milliseconds or something. That was just unreal amounts of improvement and if you go down the entire sheet, there isn't a single place where something didn't get at least slightly faster.

John Resig: Yeah, yeah. I guess that's an interesting part about having effectively a static codebase, I mean that's not to say that jQuery isn't going to change or add new methods, it's just that we're changing very, very little of the API and because of that, we can really just focus on iterating and improving what we have, fixing bugs, looking and digging and seeing where the performance pains are



and just consuming and improving and iterating and making it better, which I think is a really lucky situation that we're able to just iterate like this.

Scott Hanselman: One last question I wanted to ask you is how do you decide what plug-ins to bring in? Is a plug-in the best way for someone to kind of get noticed?

John Resig: It is. What's happened historically is people will write a plug-in and in a couple of cases, there has been plug-ins that have just been immensely useful and what we find are just people using them over and over again and they're coming to rely on them and other plug-ins are coming to rely on them. So, what we're starting to do now is to take these really successful plug-ins and start to integrate them back into core. This has been really useful for us and it's been great for the plug-in office as well to see their code effectively graduate and the users benefit. What's nice about this is that it serves as a testing ground because you don't actually have to introduce these new methods into the core. You can just say, "Oh, here. Use this plug-in. Let us know how it works and eventually we can integrate it back in."

Scott Hanselman: Yeah.

John Resig: I think that ended up being a really successful model development for us.

Scott Hanselman: I think that's a really successful model for development for everyone. I think that looking -- of course, hindsight is 20/20, so looking back it seems obvious, but the idea that if one makes it just incredibly easy to extend one's open source project rather than saying, "Let's all fight about who's going to get commit access and who's going to be able to extend the core just make it incredibly extensible and then pick the right ones to pull back in." It seems like it's such a more natural model for an open source project. It's just a more natural open source way of doing things, but that is all predicated on having an incredibly simple model for extensibility.

John Resig: Yeah, yeah.

Scott Hanselman: Cool. Well, John, I really appreciate you taking the time to sit down and talk to me today about jQuery and we'll have links to all the information about John and jQuery up on the show site. Thanks again.

John Resig: Oh yeah. Thanks for having me.

Scott Hanselman: And this has been another episode of Hanselminutes and we'll see you again next week.