

Designers <3 jQuery



Welcome!

First time ever seeing me, start off with...

Who Am I?

Well, first of all, my name is

Buck.

I am originally from



I am employed by

Jive Software

UI Designer.



In 2001 I co-founded a company called

Inversion Designs



Web sites can behave like desktop applications.
Fancy for its time. Aurora Calendar.

Around 2004 / 2005 apps started getting really dynamic, so we pushed the envelope and started

The things we did back then were pretty fancy for its time, but would be considered tame by today's dynamic standards. Our main product was Aurora Calendar, an online app that allowed people to share calendars and delegate tasks to one another.

Realizing that we had to take risks if we were ever going to be super successful, we shifted gears in 2005 and started

Jotlet



Jotlet, using our 4 years of experience in dynamic web-calendaring to dive headfirst into the newly beginning AJAX era. In 2008, we were acquired by Jive and... well, here I am.

jQuery



Using since 2006. Got sick of programmers doing all the cool stuff. I wanted control.

I've been using jQuery since its inception in 2006. The other Jotlet co-founder, Adam Wulf, was the programmer and, to be frank, I got sick of him being able to do all the cool stuff. I'd routinely have to ask him "hey is this possible? hey is this possible? can you tweak this? it doesn't look quite right. can you move it over two pixels on hover, etc", so I took it upon myself to learn JavaScript and jQuery was the ticket for a designer like me.

So enough about

Me

You

What are you going to learn?

What are you going to (hopefully) get out of this discussion?

What are you going to learn?

- Why it's important for a designer to know at least a little JavaScript
- The brief history of jQuery
- How jQuery speaks like a designer

~~Presentation~~

Discussion



I don't have a PhD in jQuery, (although that would be pretty rad), I'm just speaking my experiences and how they can benefit you. If you have anything to add at any time, just blurt it out. Seriously.

So let's get started.

Why do I need to know JavaScript?



interaction Designer:
draw workflows, wireframes, prototypes, talk to customers, make sure the product is good, balance needs of customers with wants of PM with restrictions of devs, visual style and interactive design. Held responsible if users don't like the product, and I code the HTML / CSS.

Let's face it. I'm an interaction designer. I draw boxes and workflows and talk to the customers and make sure the right things are being surfaced in the right places. I balance the needs of the customers with the wants of Product Management with the restrictions of the developers, I'm responsible for the visual style of the product as well as the interactive bits. If the users don't like the product, I'm usually held responsible. On top of that, I even code the HTML and the CSS. DON'T I HAVE ENOUGH ON MY PLATE?

Well, yes. You do. But I promise you

Knowing JavaScript will make your life easier.

Learning jQuery will make knowing JavaScript easier.

How?



Well, I'll tell you.

But before I get into that, it's important to emphasize that

Static is out.



I'm sure all of you know this. Not only on a "web application" level, which is obvious, but even the most simple marketing pages can be made more interesting by making them interactive. Navigation animations like those seen on taptaptap and panic's websites can make the experience feel more accessible by maintaining a consistent context for the user to live in.

Plus making things dynamic is just plain fun. They engage the user, which is something the web medium does best.

- Form Validations
- Type-ahead / autocomplete search
- Drag & Drop
- Transition Animations
- Modals
- Hover / toggles
- AJAX
- ???

All of these are almost expected now-a-days. Plus, it's the interaction designers who are coming up with these things, or at least someone filling an interaction designer role. The future of what's possible is up to us.

One designer creates and implements a nifty UI trick and it doesn't take long before it's a paradigm that's seen across the web, from Facebook to CNN.

Okay, Buck. I know dynamic web pages are awesome, but seriously why do I need to be the one to make them that way?

Know it so you can understand it.



A pretty straightforward statement, really. Even if your role at your business doesn't require you to code JavaScript, understanding it will let you know what is and, more importantly, what isn't possible. The better you know JavaScript, the more you'll be able to stretch the boundaries of your interaction design and your knowledge will allow you to be as creative as you need to be.

How can you be expected to innovate when you don't even know the rules you're playing by? Sure you can just make things up willy nilly but you're wasting the time of your client, PM or developer if you're suggesting things that simply aren't possible.

So why else do you need to know JavaScript?

So you can be a bad-ass.



Raise your hands if you know HTML / CSS. Now, keep your hand raised if knowing HTML & CSS is a requirement for you to keep your job.

It won't be long before web designers will be pretty much required to know JavaScript. More and more businesses and clients think "hey JavaScript is on the client side- that's designer territory!"

Knowing JavaScript and implementing dynamic solutions will make you more marketable as a designer, and you'll bag that job over someone who just draws boxes for a living.

Richer Prototypes



I don't have to tell you the benefits of prototyping. There's a huge difference between showing someone and having them experience your design. Knowing JavaScript will allow you to quickly iterate prototypes without having to get another person or team involved.

Trust me, you'll be surprised how quickly you can get things working by knowing a bit of jQuery.

Knowing JavaScript will make your life easier.

Learning jQuery will
make knowing
JavaScript easier.

What makes jQuery so special?

HISTORY LESSON!

Getting the DOM
elements you wanted to
interact with used to suck.



So John Resig, at the age of 3 or something, wrote a CSS selector engine for JavaScript.

What does that mean, Buck?



I'm glad you asked.

Like I said, getting DOM elements to use in JavaScript used to suck. What I mean is in order to do anything fancy in JavaScript, you had to have a reference to the DOM element you wanted to work with. If you wanted fancy stuff to happen when you typed in a box, you had to first get that box in JavaScript. Most people did something like this...

```
<input type="text" onkeyup="alert('key up!');" />
```

I bet a lot of you have seen this. This is crappy because then you have JavaScript code right there with your HTML. If you wanted to use that code somewhere else, it wouldn't be very fun, especially if your code was particularly complex. Plus, events like these aren't always cross-browser friendly, and it's just not clean. It's like popping a style tag in the HTML. Ick.

So the better way was to get a reference to the DOM element you needed in the `<script>` tag up top. Stay with me, this is as technical as I'm going to get for the whole discussion.

Getting an Element with JavaScript

- `getElementsByTagName`
- `getElementById`
- `firstChild`
- `nextSibling`
- `parentNode`

So here's how people were getting an element with JavaScript. Don't bother writing this down or anything, in fact I'd be pretty happy if you just forgot it all once this slide was done.

However, it got even more complex, because when traversing the DOM, whitespace counts as a node, so to get the first child of a particular DOM element you'd have to use a function like this...

```
function next(elem) {  
  do {  
    elem = elem.nextSibling;  
  } while (elem && elem.nodeType !== 1);  
  return elem;  
}
```

```
function first(elem) {  
  elem = elem.firstChild;  
  return elem && elem.nodeType !== 1 ?  
    next (elem) : elem;  
}
```



You know what this looks like to me?

PROGRAMMING.

All of this to get a first child in a robust fashion. Thanks, but no thanks. Maybe I'll be a farmer instead.

Anyway, the point is it sucked and people knew it sucked, so they worked on what was called CSS selector engines for JavaScript, which would allow you to use the CSS all of us know and love to get DOM elements.

`$("div a:first")`

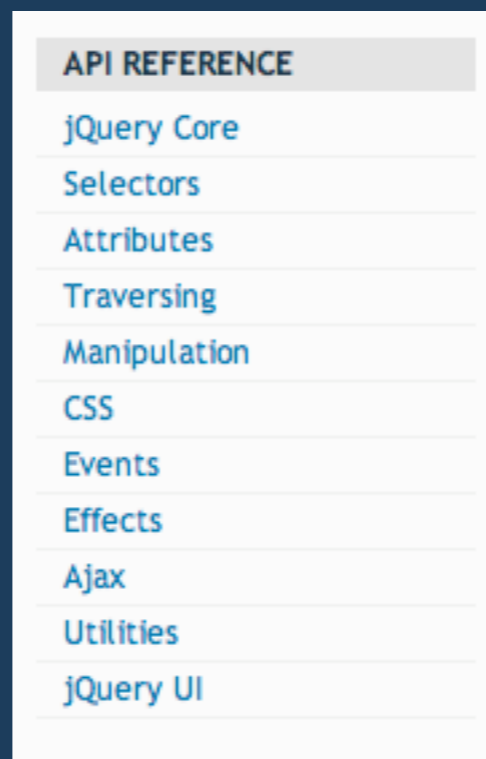
Like this. Get the first anchor tag in every div. Lovely.

So that's how jQuery got its start. It was properly announced announced at the beginning of 2006 and has been growing in scope and ease of use ever since.

What makes jQuery so special?

So back to the original question. “What makes jquery so special?” As in, why use it over any other of the multitude of libraries out there?

The documentation is phenomenal



docs.jquery.com



The documentation is phenomenal. It's easy to understand, it's broken up into logical chunks, and there are plenty of examples for each of the features that it provides. It assumes you're trying to do something, then it tells you how to do it. If you want to know about Selectors, Attributes, Traversing, Manipulation, CSS, etc... You know exactly where to go.

It's not a library that's obsessed with itself, so you don't have to dig through a lot of junk you don't understand to get to what you want to do. These categories are logical for us designers. Unlike...

API Docs
● Utility Methods
● Ajax
● Array
● Class
● Date
● Element
● Element.Methods
● Element.Methods.Simulated
● Enumerable
● Event
● Form
● Form.Element
● Function
● Hash
● Insertion
● Number
● Object
● ObjectRange
● PeriodicalExecuter
● Position
● Prototype
● String
● Template
● TimedObserver
● document
● document.viewport

This. This is one of the other primary JavaScript libraries, and you can see the two have a different audience altogether. If you want to know about `Element.Methods.simulated`, `Enumerable`, `Hash`, `ObjectRange`, or `PeriodicalExecuters`, then I guess jQuery isn't the library for you and you should probably just go home because I'm not going to do anything for you.

Frankly this stuff scares me because it is scary.

The number two reason to use jQuery...

Lots of people use it.



is that lots of people use it. All the kids are doin' it, and you want to be cool, don't you?

Actually, the benefits of this can't be overstated. If a lot of people use it, then you have...

Lots of people use it.

- Lots of plug-ins
- Lots of tutorials
- Lots of people to ask when you have a problem
- Lots of marketability



Lots of Plug-Ins : it's so easy to make plug-ins in jQuery, so lots of people do it. Lots of plug-ins means more stuff that you don't have to code.
Lots of tutorials : the best way to learn is by doing, and tutorials will show you step by step.
Lots of people to ask when you have a problem
Lots of marketability : as jQuery pulls ahead in the library race, knowing jQuery will make you a more valuable asset to potential companies or clients.

Another reason jQuery rocks my world is it's...

Highly Supported



Mozilla has at least one developer actively working on jQuery at a time, and jQuery releases alphas and betas of their releases well-before they update. Updates come often- jQuery has already had two releases this year, each one providing substantial performance gains and new features. (Prototype hasn't updated in over 200 days)

And the most important reason to use jQuery, and why you are all here, is...

Small Footprint



It's tiny, 19kb gzipped.
And the most important reason to use jQuery, and why you are all here, is...

jQuery Speaks Our Language



And by “our” I mean designers. We’re just not programmers.

There are certain things we know, and jQuery focuses on taking what we as designers know, and expanding upon that to create rich interactions. jQuery’s mantra is “do more with less”. Writing less code? That’s right up my alley.

If that’s not talking right to designers I don’t know what is.

If you already know HTML and CSS, here’s all the stuff you can do with JavaScript via jQuery...

- Get a DOM Element
- Get a *bunch* of DOM Elements
- Style those Elements
- Find out when a user is interacting with those elements
- Do stuff when that user interacts with those elements
- Animate any portion of that DOM element
- Make that element appear or disappear
- Create DOM elements
- Put those elements wherever you want



I'm serious. If you know HTML and CSS, you can already do all of this stuff with jQuery. It's just a matter of syntax and trust me, there isn't much. Dollar signs, dots and quotes, brackets, and whatnot.

Plus, if you think about it, if you know how to do this stuff, you can do pretty much anything you need to, which is a pretty good starting point if you ask me.

You don't believe me that you know all of this stuff?
Here I'll prove it.

```
<div class="container">  
  <a href="refreshportland.org">  
    Refresh yo-self before yo wreck yo-self.  
  </a>  
  <span class="spantastic">  
    Seriously. Go do it.  
  </span>  
</div>
```

You know what this is, right?

```
$(“ <div class=“container”>  
  <a href=“refreshportland.org”>  
    Refresh yo-self before yo wreck yo-self.  
  </a>  
  <span class=“spantastic”>  
    Seriously. Go do it.  
  </span>  
</div> ”)
```

Booya you just created three dom elements, gave them attributes, and put them in a DOM proper structure using jQuery. Good for you. Now you can inject this HTML wherever you want in the DOM.

```
var myDiv = document.createElement("DIV");
var myA = document.createElement("A");
var mySpan = document.createElement("SPAN");
var refreshYoself = document.createTextNode("Refresh yo-self
before yo wreck yo-self");
var seriously = document.createTextNode("Seriously. Go do
it.");
myDiv.setAttribute("class", "container");
myA.setAttribute("href", "refreshportland.org");
myA.appendChild(refreshYoself);
mySpan.appendChild(seriously);
myDiv.appendChild(myA);
myDiv.appendChild(mySpan);
```

This is how it would look without jQuery. Do you know what that is? I don't. JavaScript just doesn't speak my language, and I certainly don't speak whatever language this is.

#main a

Okay so you know what this is. It looks like CSS. Probably gets all of the anchor tags in the element with the ID “main”.

```
$("#main a")
```

Hey, good job, that's exactly what it does with jQuery, too. Now you have a handle on all of the anchors within the element ID "main".

```
#container ul.people li
```

```
border 1px solid blue
```

I'll keep going, since we're all having so much fun, but now we'll get fancy.

Again, this looks like CSS. There's a CSS selector and a CSS definition. Makes the border blue of the LIs that match this selection.

```
$(“ #container ul.people li ”)  
  .css(“border”,“1px solid blue”);
```



Yep. That’s right. Using jQuery you can now do this with JavaScript using information you already have in your brain.

“But Buck”, you ask. “Why would I want to change CSS in JavaScript at all?”

Let’s find out.

```
#container ul.people li    hover
                             border    1px solid blue
```

Okay so we know this first portion is a CSS selector, and this part here is a CSS definition. Well this looks like plain English, so without adding the jQuery you already know what this bad boy is going to do.

```
$(“ #container ul.people li ”). hover (function() {  
    $(this).css(“ border”, “1px solid blue”);  
});
```

WOAH. Holy crap, Buck, you just added a lot of junk up there.

It's okay. Don't worry too much about the syntax. Just understand that “this” means “that” and you're applying this CSS rule to the CSS selector when you hover.

“Hover” here is an event. When the user interacts with a web page, the browser is “listening” for these “events”. Each browser listens in its own cute little way, making things a nightmare for traditional JavaScript developers. Fortunately there's jQuery for us designers.

API REFERENCE
jQuery Core
Selectors
Attributes
Traversing
Manipulation
CSS
Events
Effects
Ajax
Utilities
jQuery UI

Okay so this is the jQuery documentation navigation. Can you guess where you'd go to learn about which events jQuery offers you?

...

It's the one called Events.

jQuery Events

- hover
- blur
- change
- click
- dblclick
- focus
- keydown
- keyup
- keypress
- mousedown
- mouseup
- mouseover
- mouseout
- scroll
- select
- submit



```
$("#main a").eventName();
```

Finding something to attach the event to with CSS and

...
including the event name. The code that goes here

...
Is what happens when that event is fired on that item.

Alright, that's all well and good but what if I want to do some seriously fancy stuff? Like what if I want to make things animate and fly around the page?

Built-In Animation

- `slideDown` / `slideUp`
- `fadeIn` / `fadeOut`
- `show` / `hide`

jQuery, of course, has you covered.

To show / hide an object you have the ability to ...

slide it down to show it, `slideUp` to hide it...

fade it in or fade it out...

or the plain old show / hide. You can edit the durations of any of these, and if you give show or hide a duration it does this fancy scale out zoom in thing.

“But Buck” you’re telling me, “this doesn’t satiate my need to make things fly around the page and change color and stuff.” It’s cool, jQuery supports

Custom Animation

For example, let's take our friend

```
$("#main a").animate({
    fontSize: "10em",
    opacity: ".5",
    left: "300px"},
    1000 );
```



main a. If, for some reason, we want to ...
animate some...
random attributes with it...
for a certain period of time...
we can.

This lovely piece of code will target all of the anchors inside element ID main and animate the fontsize to 10em, cut the opacity down by half, and the left property to 300px, in 1000 milliseconds, or one second.

Combine this with the events from a few slides ago and you're already on the way to some rad interactions. It doesn't look so scary, right?

That's because jQuery speaks our language.

So by now I'm sure you're thinking to yourself

Fun Examples






From real-life websites omg.

Panic Nav

The story of Coda.

So, we code web sites by hand. And one day, it hit us: **our web workflow was wonky**. We'd have our text editor open, with [Transmit](#) open to save files to the server. We'd be previewing in Safari, adjusting SQL in a Terminal, using a CSS editor and reading references on the web. **"This could be easier,"** we declared. **"And much cooler."**

What have we added to Coda lately?


-  **Plug-ins.** This is huge! Teach Coda new tricks, and extend its functionality, lickety-split. Download plug-ins that others have written, or write your own plug-ins using your favorite scripting language. This is just the beginning — learn more in the [Coda Developer Zone](#). We look forward to seeing what you make!
-  **Open quickly.** Hit **Control-Q** to instantly access the Open Quickly window. Type a few letters of the local file you want to edit, anywhere in your site. Ka-jang! Instantly open it. It's Spotlight for your website files!
-  **Smart Spelling.** Spell check your words — not your code. And in Mac OS X 10.5, do it as you type!
-  **Subversion.** Work with a team using the most popular source control system there is, baked right into the sidebar. Check out code, update, commit changes — yep, Coda just saved you even more time.
-  **Find across files.** You can now find and replace text across multiple local files — open files, files in a specific folder, or the files in your "site" — using the same sleek, inline search bar you know and love.

The story of Coda.


So, we code web sites by hand. And one day, it hit us: **our web workflow was wonky**. We'd have our text editor open, with [Transmit](#) open to save files to the server. We'd be previewing in Safari, adjusting SQL in a Terminal, using a CSS editor and reading references on the web. **"This could be easier,"** we declared. **"And much cooler."**


What have we added to Coda lately?

 **Plug-ins.** This is huge! Teach Coda new tricks, and extend its functionality, lickety-split. Download plug-ins that others have written, or write your own plug-ins using your favorite scripting language. This is just the beginning — learn more in the [Coda Developer Zone](#). We look forward to seeing what you make!

 **Open quickly.** Hit **Control-Q** to instantly access the Open Quickly window. Type a few letters of the local file you want to edit, anywhere in your site. Ka-jang! Instantly open it. It's Spotlight for your website files!

 **Smart Spelling.** Spell check your words — not your code. And in Mac OS X 10.5, do it as you type!

 **Subversion.** Work with a team using the most popular source control system there is, baked right into the sidebar. Check out code, update, commit changes — yep, Coda just saved you even more time.

 **Find across files.** You can now find and replace text across multiple local files — open files, files in a specific folder, or the files in your "site" — using the same sleek, inline search bar you know and love.



Double click, start work.

Your Coda experience starts with "Sites", über-favorites on cute little pieces of paper. When you're ready to start work, **just double click a site** — Coda will **instantly restore itself exactly to where you left it**, connecting to your server, restoring any splits and tabs, and allowing you to code, pronto. Plus, sites allow you to easily preview local or remote content, even if it's dynamically generated through a web server.

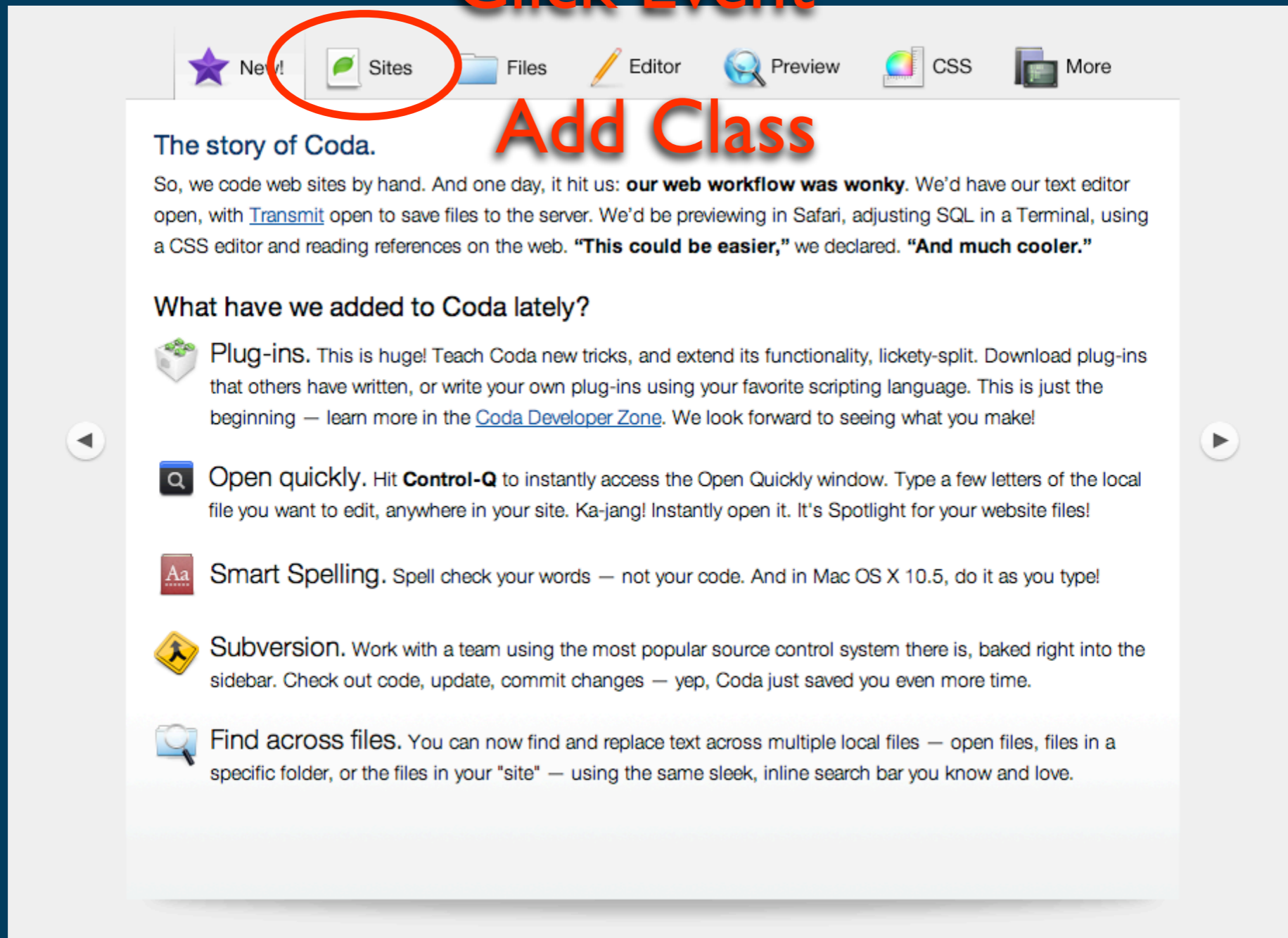
Plus, one-click publishing.

If you work on websites locally, then publish files remotely to your server, **Coda's sites will automatically track your local changes and do it all for you**. Just hit the big, shiny "Publish All" button. Ka-plam!

Publish All








Click Event



The screenshot shows the Coda application interface. At the top, there is a toolbar with several icons: a purple star labeled 'New!', a green leaf icon labeled 'Sites' (circled in red), a blue folder icon labeled 'Files', a yellow pencil icon labeled 'Editor', a magnifying glass icon labeled 'Preview', a color palette icon labeled 'CSS', and a stack of icons labeled 'More'. Below the toolbar, the main content area has a white background. At the top of this area, the text 'The story of Coda.' is followed by a large red 'Add Class' button. Below this, there is a paragraph of text: 'So, we code web sites by hand. And one day, it hit us: **our web workflow was wonky**. We'd have our text editor open, with [Transmit](#) open to save files to the server. We'd be previewing in Safari, adjusting SQL in a Terminal, using a CSS editor and reading references on the web. **"This could be easier,"** we declared. **"And much cooler."**

Below the paragraph is a section header: 'What have we added to Coda lately?'. This is followed by five list items, each with an icon and a short description:

-  **Plug-ins.** This is huge! Teach Coda new tricks, and extend its functionality, lickety-split. Download plug-ins that others have written, or write your own plug-ins using your favorite scripting language. This is just the beginning — learn more in the [Coda Developer Zone](#). We look forward to seeing what you make!
-  **Open quickly.** Hit **Control-Q** to instantly access the Open Quickly window. Type a few letters of the local file you want to edit, anywhere in your site. Ka-jang! Instantly open it. It's Spotlight for your website files!
-  **Smart Spelling.** Spell check your words — not your code. And in Mac OS X 10.5, do it as you type!
-  **Subversion.** Work with a team using the most popular source control system there is, baked right into the sidebar. Check out code, update, commit changes — yep, Coda just saved you even more time.
-  **Find across files.** You can now find and replace text across multiple local files — open files, files in a specific folder, or the files in your "site" — using the same sleek, inline search bar you know and love.

Animate offset

Refresh

NEXT REFRESH

Designers <3 jQuery

presented by **Buck Wilson**

jQuery: it's so hot right now, and its ease of use is making it the JavaScript library of choice for designers everywhere.

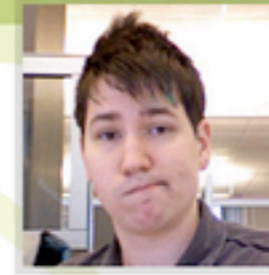
Learning jQuery can give you a greater depth of knowledge in creating rich user interactions on the web, allowing you to design with these interactions in mind.

Buck Wilson will take you for a ride through the history of JavaScript on the web, why jQuery has seen so much success, and how jQuery speaks the language of designers.

All aboard the jQuery train! Choo choo!

April 23rd, 2009, 6:30 PM at **Jive Software**

[Let us know you're coming](#)



Buck Wilson is a designer and developer at **Jive Software**. Prior to working at Jive, he co-founded **Jotlet**. Buck has written a number of tutorials on the subject of jQuery, including his latest: "jQuery for People Who Can't Code Good".

← PREV

NEXT ►

Everyday Git

presented by Carlos Rodriguez

Inside the web in Japan

presented by Hajime Kobayashi

Culture Shift - A place at the Long Table

presented by David Lowe-Rogstad

Designers <3 jQuery

presented by Buck Wilson

To be announced



NEXT REFRESH

Designers <3 jQuery

presented by **Buck Wilson**

jQuery: it's so hot right now, and its ease of use is making it the JavaScript library of choice for designers everywhere.

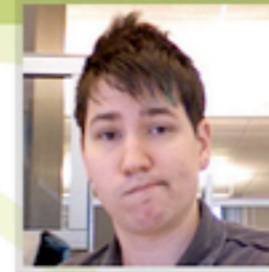
Learning jQuery can give you a greater depth of knowledge in creating rich user interactions on the web, allowing you to design with these interactions in mind.

Buck Wilson will take you for a ride through the history of JavaScript on the web, why jQuery has seen so much success, and how jQuery speaks the language of designers.

All aboard the jQuery train! Choo choo!

April 23rd, 2009, 6:30 PM at **Jive Software**

[Let us know you're coming](#)



Buck Wilson is a designer and developer at **Jive Software**. Prior to working at Jive, he co-founded **Jotlet**. Buck has written a number of tutorials on the subject of jQuery, including his latest: "jQuery for People Who Can't Code Good".

← PREV

NEXT ►

Everyday Git

presented by Carlos Rodriguez

Inside the web in Japan

presented by Hajime Kobayashi

Culture Shift - A place at the Long Table

presented by David Lowe-Rogstad

Designers <3 jQuery

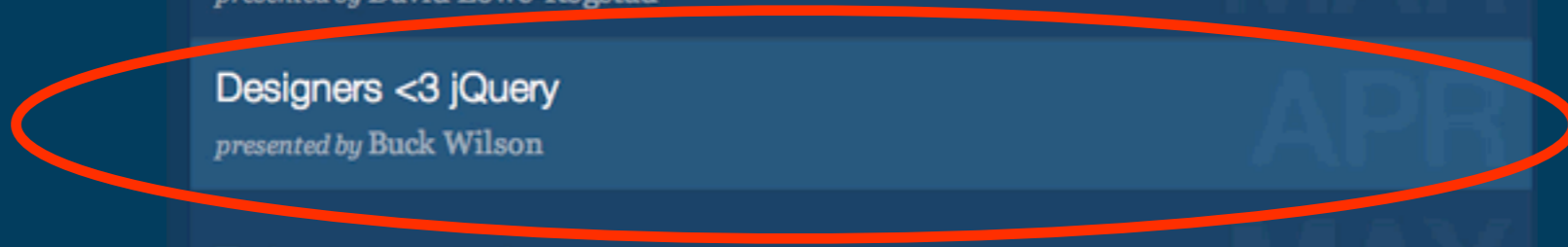
presented by Buck Wilson

To be announced

Opacity : 0

Display : None

Click Event



NEXT REFRESH

Designers <3 jQuery

presented by [Buck Wilson](#)

jQuery: it's so hot right now, and its ease of use is making it a popular choice for designers everywhere.

Learning jQuery can give you a greater depth of knowledge and control over interactions on the web, allowing you to design with these in mind.

Buck Wilson will take you for a ride through the history of JavaScript, how jQuery has seen so much success, and how jQuery speaks to the heart of the web.

All aboard the jQuery train! Choo choo!

April 23rd, 2009, 6:30 PM at [Jive Software](#)

[Let us know you're coming](#)

◀ PREV

NEXT ▶

Everyday Git

presented by [Carlos Rodriguez](#)

Inside the web in Japan

presented by [Hajime Kobayashi](#)

Culture Shift - A place at the Long Table

presented by [David Lowe-Rogstad](#)

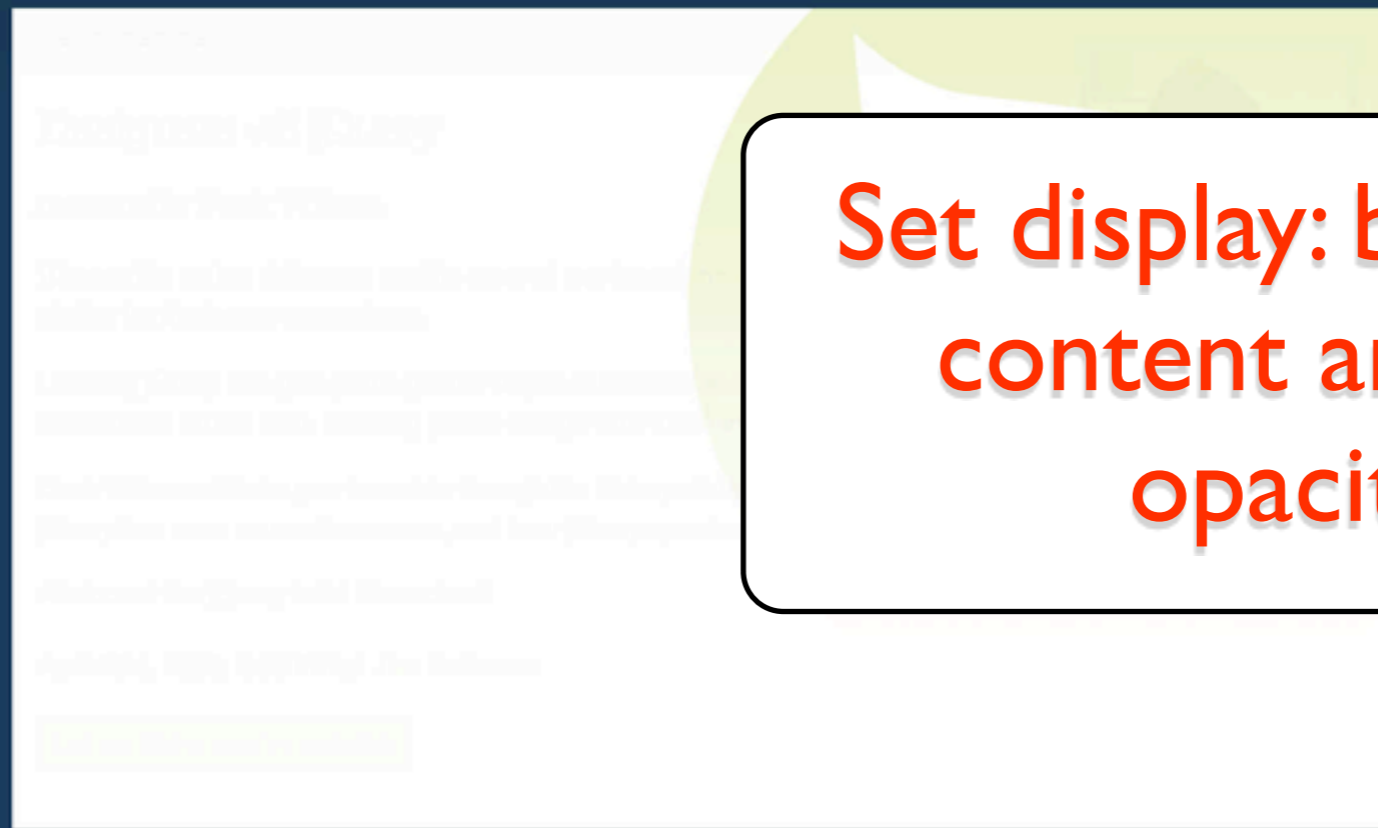
Designers <3 jQuery

presented by [Buck Wilson](#)

To be announced

Animate Height: height of
the new content div
\$("#target").outerHeight()

JAN
FEB
MAR
APR
MAY



Set display: block on new content and animate opacity to 1

◀ PREV NEXT ▶

Everyday Git
presented by Carlos Rodriguez

Inside the web in Japan
presented by Hajime Kobayashi

Culture Shift - A place at the Long Table
presented by David Lowe-Rogstad

Designers <3 jQuery
presented by Buck Wilson

JAN
FEB
MAR
APR

I CAN'T GET ENOUGH OF JQUERY



And I don't blame you. jQuery gives us designers a lot of power to make some pretty amazing things without having to spend our free time reading O'Reilly books.

So how do you learn more jQuery? I'm glad you asked!

I CAN'T GET ENOUGH OF JQUERY

- jQueryForDesigners.com
- learningjQuery.com
- cantProgramGood.com



First off I recommend jQueryForDesigners.com, which is run by Remy Sharp. His tutorials are in screencast format and he teaches you how to mimic popular UI effects using jQuery. He not only teaches you the hows but the whys.

Learning jQuery will give you some good tutorials as well, as well as some good dos and don'ts for clean code.

If you find yourself a little lost with these two, I'm starting a new series called "jQuery for People Who Can't Program Good and Want to Learn to Do Awesome Stuff" at cantprogramgood.com. These will be 8-10 minute episodes and I will start you off from the very very basics, so you get a good solid jQuery foundation. Look for that to launch sometime next month.