The No Bullshit Bible : Creating Web 2.0 Startups & Programming

Written by James Gillmore
Edited by Holly Welch

FaceySpacey

www.faceyspacey.com

Javascript

TABLE OF CONTENTS

6-1 JAVASCRIPT | WHAT IS JAVASCRIPT?

So if I was to ask you what the language of the browser is what would you say? You may be tempted to say HTML, right? Well, the real answer is Javascript. HTML is a markup language, but it's hardly a language that any self-respecting developer would call a "language." I won't get into the exact definition of Javascript, which includes terms like "prototype-based" and "object oriented," etc, because you can read it here: http://en.wikipedia.org/wiki/JavaScript . But the main idea is that you can implement more complex programming structures than markup, e.g. loops, if/else logic, variables, and a lot more. You can even generate HTML on to the page without having to code traditional HTML.

So in short, Javascript is the "client side scripting" language of the browser. The reason it's called "client side" is because your browser is a client of your server that serves the website. PHP is on the server side and Javascript is on the client side.

The main things you will use Javascript for are AJAX requests to the server, dynamically updating the page without a page refresh (sometimes in combination with AJAX, and sometimes not), animation and generally fancy interactions that make your web app feel more responsive like a desktop application. To make a Web 2.0 app's interface and user experience sing it will all be thanks to Javascript.

Some technical notes to know are that Javascript's syntax is influenced by C like PHP is, which is why I think PHP is a great first language to learn. Basically you'll be learning two similar languages and have a lot of preparation for learning C, which is basically the grand father of modern programming and still very relevant. Cooler server side scripting languages of the day like Ruby and Python are a lot less like C. Therefore learning PHP and Javascript is a great place to start. They'll make it way easier to Learn C and C++ when the time comes, as well as Java. And by that time Ruby and Python will be no-brainers.

Another technical thing to note is that Javascript basically knows everything going on in a web page. It has access to all the HTML on the page, all the information the browser provides such as its width and height, the URL being accessed, etc. Javascript has hooks into all the information about the current web page.

6-2 JAVASCRIPT JQUERY | INTRODUCTION TO JQUERY

As is the style of these tutorials, we're going to jump you to productivity immediately. So rather than learn all the ins and outs of Javascript--and basically waste time--we're going to look at a framework built on top of it that makes it a lot easier: jQuery. jQuery is the most popular, most maintained, and basically most advanced Javascript framework.

Since Javascript was very much like PHP in its basic syntax, I didn't have to study Javascript as much to get productive. So the way I learned Javascript and jQuery was by jumping straight to installing ready-made jQuery plugins, modifying them a bit, and then writing my own jQuery code. After that I went back and learned a little bit more about Javascript to really understand what was going on. Javascript does have some unique and powerful aspects that won't be clear at the beginning as it's quite different from a classical language like PHP (i.e. it's object oriented style of programming is based on "class" templates"), but jQuery makes it so you don't have to know them.

Ok, so the first thing to note is that jQuery shares a lot in common with CSS, specifically selectors. Understanding selectors, which you should already by now if you read the CSS tutorials, is the crux of mastering jQuery. jQuery has its own format for writing selectors, but they look very similar to CSS selectors. And more importantly, it lets you do more with these selectors than just style elements (though you can do that too): you can trigger events when your selected elements are clicked for example, and you can move a selected element across the page. We'll start with these two examples.

Firstly, for jQuery to work, you have to embed it on your page. You can simply add the following line within your *<head>* element:

```
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.5.1/jquery.
min.js" type="text/javascript"></script>
```

That will load the jQuery framework code into all browsers viewing your website. It will load the code from google's servers, and they'll pay for its hosting.

After that's installed you can write jQuery code within *<script>* tags anywhere on your page.

Let's cover some jQuery selectors:

1. ***$("#header")*** *= get the element with id="header"*
2. ***$("h3")*** *= get all <h3> elements*
3. ***$(".photo")*** *= get all element with class="photo"*
4. ***$("div#content .photo")*** *= get all element with class="photo" nested in the <div id="content">*
5. ***$("ul li")*** *= get all <li> element nested in all <ul>*
6. ***$("ul li:first")*** *= get only the first <li> element of the <ul>*

Capiche. Therefore, here's how you would do what you saw can be done in CSS with jQuery:

```
$(".photo").css('color', 'blue');
```

That’s the equivalent of:

```
.photo {color: blue;}
```

Capiche. Not so difficult. Now you’re probably asking why do I need jQuery to do this when CSS can? Well obviously because you can do a lot more. Here we’re just building on similarties between the new material I’m teaching you and the CSS stuff you already know.

Let’s make it so when you click any elements with class “photo” they trigger a simple alert on the page that says “you clicked me!”:

```
$(“.photo”).click(function() {
   alert(“you clicked me!”);

});
```

So here we obviously “selected” all elements with class photo, and then called the *click()* function provided by jQuery, and then told it to call the function we passed into the *click()* function when the actual click is performed. There are 2 things to note here:

1) that jQuery selectors are actually objects, and have methods, which are separated with a period (rather than “->” like in PHP). Once you have one of these objects--usually built with a selector like this--you can call methods like *click()* to trigger all the event-driven stuff that CSS selectors can’t do.

2) the *click()* function is passed a parameter in the form of a function that will be called at a later point in time.

In this case that function parameter simply triggers the built-in browser function, *alert()*. The parameter we passed to the *click()* function is actually called an "anonymous function," which means it does not have a name. In PHP you don't use them much, but in Javascript and jQuery, you use them non-stop. The idea is that the code doesn't really need a name because you're not going to use it any other places. If you were, you could write this:

```
$(".photo").click(alertUser);

function alertUser() {
   alert("you clicked me!");

}
```

So does that make it clear how *click()* is really a function passed another function as a parameter. That relatively large block of code called an "anonymous" function is really just a big parameter passed to the*click()* function. What I'm trying to point out is simply that alertUser as shown in the first line above is a parameter, and that it's the same as the anonymous function from the first example. This is a key concept to get.

6-3 JAVASCRIPT JQUERY | ANIMATION WITH JQUERY

Animation is often one of the most intriguing things about client side coders that new Javascript/jQuery developers always want to jump to. So lets do it.

CSS:

```
.myDiv {position: relative; left: 10px; width:100px; font-size: 12px;}
```

HTML:

```
<div class="myDiv" />Some text...</div>
```

JQUERY:

```
$(".myDiv").animate({left: 500, fontSize: 36}, 5000, function() {
   alert("animation is complete!")

});
```

So basically what that does is move "*.myDiv*" by 500 pixels from the left. "*left*" is a standard CSS positioning property. Notice that *myDiv* is positioned 10 pixels from the left in the initial CSS rules. And the font size grows from *12px* to *36px*.

So obviously the *animate()* method is doing all the work here. It's applied to the elements selected by the *$(".myDiv")* selector, which in this case is just one element, but it could move multiple elements at the same time. It receives several parameters which configures exactly how it will animate. The first parameter is a list of CSS properties to "tween" style animate from their current settings to what's written in here.

The "*5000*" in the second parameter saying how long the animation should take to move those *490* pixels (i.e. from *10* pixels from the left to *500* pixels from the left).

The 3rd parameter is a function that is called at the end of the animation, which in this case will simply alert you that the “*animation is complete!*”

Now that’s the end of the tutorial, but lets look at how all the components truly look on a web page. I left out some required syntax that is really easy to remember--because it’s required and you’ll use it a lot--but wasn’t conducive to learning the main topics. Often when learning programming, they’ll flood you with tons of syntax and then get to the main topics, and it’s hard to know what you should be focusing on...So now that you got the main concepts, let’s look at the required syntax this is missing:

```
<html>
<head>

<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.5.1/jquery.min.js">
</script>

<style type="text/css">
.myDiv {position: relative; left: 10px; width:100px; height: 25px;
font-size: 12px;}
</style>

</head>

<body>

<div class="myDiv">Some text...</div>

<script type="text/javascript">
$(document).ready(function() {
   $(".myDiv").animate({
      left: 500,
      fontSize: 36
   }, 5000, function() {
      alert("animation is complete!")
   });
});
</script>

</body>

</html>
```

So notice the *<script>* tags that wrap the jQuery, and that also embed the jQuery framework at the top. Notice how the CSS is similarly wrapped, but in *<style>* elements. And notice how the page is wrapped in *<html>* tags, and how it's separated into 2 parts: *<head>* and *<body>*. The *<head>* is where you declare a bunch of meta data and generally where you enter helper tools like the jQuery framework etc.

6-4 JAVASCRIPT JQUERY | EMBEDDING JQUERY PLUGINS

So as I mentioned my first forays into the world of jQuery included embedding copy/paste jQuery plugins. A jQuery plugin is basically 3 parts:

*1) **PLUGIN LIBRARY CODE** - a mini sort of framework that you embed into your page similar to how you embed the entire jQuery library. You won't even need to look at this code--that's the whole point. You don't have to worry about its complexities.*

*2) **THE HTML** - corresponding HTML that the jQuery plugin controls. So if this was a slideshow plugin, here would be some HTML that lists photos on a page. The jQuery plugin will do something like hide all but one at a time, and transition to the other photos by unhiding and hiding them consecutively to create the slideshow effect.*

*3) **JQUERY CLIENT CONFIGURATION CODE** - a bit of simple jQuery you embed on the page with configurable properties that let you configure how the plugin works. For example, if the plugin is a slideshow, you can configure the time between transitions.*

For our example, we'll use a slideshow plugin we at FaceySpacey recently used on http://www.CrimeTV.com: http://nicolahibbert.com/demo/liteAccordion/

Go to that page really quickly and examine it. It's a quick tutorial on how to install and configure that tutorial with a demo of the result you will get at the top. That's how all these jQuery plugins come--i.e. they come with a demo and tutorial on a page or two. The idea is you can easily grasp it and implement it by copying how the demo uses it.

Let's examine their code and pinpoint the 3 aspects highlighted above.

1) PLUGIN LIBRARY CODE

```
<head>
   <link rel="stylesheet" href="liteAccordion.css">
</head>

<body>
   ... <!-- Before the closing body tag -->
   <script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.4/jquery.min.js"></
script>
   <script src="liteaccordion.jquery.js"></script>

</body>
```

2) THE HTML

```
<div class="myAccordion">
   <ol>
      <li>
         <h2><span>Slide One</span></h2>
         <div></div>
      </li>
      <li>
         <h2><span>Slide Two</span></h2>
         <div></div>
      </li>
      <li>
         <h2><span>Slide Three</span></h2>
         <div></div>
      </li>
   </ol>

</div>
```

3) JQUERY CLIENT CONFIGURATION CODE

```
<script>
```

```
$('.myAccordion').liteAccordion({
   slideSpeed : 800,
   containerWidth : 960,
   autoPlay : true
});

</script>
```

So let's start with #1 and #2 as they are the least verbose. #1 is just embedding the plugin library code:

`<script src="liteaccordion.jquery.js"></script>`

That points to a place on your server where the code that does all the heavy lifting that you don't need to worry about.

Then the configuration code in #3 is simply calling the *liteAccordion()* method that the library code provides on a standard jQuery object built from a selector that selects an element in the HTML from #2, i.e. the div with class "*myAccordion*".
The liteAccordion() method takes a block of configuration settings (an "object map") as its single parameter, and obviously you can see that they do things like adjust the "slideSpeed" and set it to 800, etc.

So now if we go back to #2, the HTML, and look at the markup there you will see that the main div containing everything has the class "*myAccordion*". Within that div you will see repeating similar HTML structures. These are the panels in the accordion. The overarching idea is that the plugin will know what to do if you structure your code precisely like this. You can put whatever content (i.e. text) in it that you want, but the elements need to be nested precisely as they are above. And that's it.

You don't need to know much more except to copy the markup and replace it with your own text (and images, e.g. when dealing with slideshow plugins), then call the plugin method on the parent element that contains the structured markup, and finally pass to that method some configurations of your own, such as the speed with which the accordion should transition. And bingo! You’re done.

Plugins are a great way for beginning web developers to breathe life into their pages. You’re going to want to google things like “jquery slideshow plugin” or “jquery accordion plugin” etc, and you’ll see that there are tons of plugins. More specifically you will see that tons of blogs have compiled lists comparing the top plugins for the given interface you’re trying to build. Here’s an example:

http://slodive.com/freebies/best-jquery-slideshow-gallery-plugins/ . Go there and find your favorite and try to make it work on a web page immediately. Just create a basic text file with the extension “.html” and copy/paste their demos properly and then open it in your browser to see. Don’t waste time before you get one of these to work and see how easy it is. Good luck!

Conclusion & Further Reading

We at FaceySpacey hope you've enjoyed our FaceySpacey Bible, and are coming away many times more ready to succeed at your next software startup. At the very least, you should have a birds-eye-view of what you need to do to get your startup, and have quelled a lot of insecurities you may have had regarding how you should execute it. That said, I will point to you to what you should do next.

As promised, the following is a list of the precise books I read to master web development using HTML/CSS, Javascript, PHP & MySQL. They are presented in the best order to most efficiently learn the subject at hand. It's similar to the order I read them in, but enhanced based on what I learned and the order I wish I read them in. Good luck:

HTML/CSS:

CSS Mastery: Advanced Web Standards Solutions

http://www.amazon.com/CSS-Mastery-Advanced-Standards-Solutions/dp/1430223979/

Before you start coding PHP, Javascript, etc, understand how HTML works. This is where you start. HTML is easy. Read this book in combination with studying the HTML & CSS tutorials on w3schools.

PHP & MySQL:

PHP & MySQL For Dummies, 4th Edition

http://www.amazon.com/PHP-MySQL-Dummies-Janet-Valade/dp/0470527587

This book--well an older edition--I read a year before I got serious about learning to code. I read it and didn't actually code anything i learned, but what it did was plant seeds in my head with regards to what programming is all about and what databases are all about, and how to connect the two. It assumes very little in what you may already know, and is an excellent start in your journey to becoming a master programmer.

PHP Object-Oriented Solutions

http://www.amazon.com/PHP-Object-Oriented-Solutions-David-Powers/dp/1430210117

This book is where I learned what OOP is. I didn't get the hang of it until reading the following book. Don't worry if you read this and have a hard time with it. This book and the next each have introductory chapters that go over how OOP works. It took me reading basically this book and the next book about the same stuff to get it. This book is a lot less complicated than the following and dives into practical examples & problems, whereas the next is a lot more theoretical.

PHP Objects, Patterns and Practice

http://www.amazon.com/Objects-Patterns-Practice-Experts-Source/dp/143022925X

After reading this book, I basically mastered OOP. It's a very hard book to get through if you're new to OOP, and goes into some very advanced stuff, specifically tons and tons of "design patterns." The design patterns are presented in as basic of a form as possible, but they weren't very practical like examples from the previous book in that you probably will never actually need any of the code used in the book. Either way, this is my favorite Programming of all time because it taught me how to think like a coder and how to solve complex problems with concise refactored solutions. It really showed me what is possible with OOP. You don't know PHP unless you've read this book.

Pro PHP: Patterns, Frameworks, Testing and More

http://www.amazon.com/Pro-PHP-Patterns-Frameworks-Testing/dp/1590598199

I read this book too just to solidify my experience with PHP, and cover all my bases. Check it out. It's optional.

PHP Functions Essential Reference
http://www.amazon.com/Functions-Essential-Reference-Torben-Wilson/dp/073570970X

At some point during my study of PHP I found this book and decided just to learn every PHP function available so that I could better understand the examples in the above books. Start reading this early on, and complete the whole thing. You'll quickly learn patterns in how PHP functions are named, and as a result be able to guess what a function does within the context of the examples in the above books--even if you don't remember precisely what it does.

Agile Web Application Development with Yii 1.1 and PHP5
http://www.amazon.com/Agile-Web-Application-Development-PHP5/dp/1847199585

I read this book in combination with reading the Blog Tutorial and Definitive Guide on YiiFramework.com. When you're done studying all these materials, you'll be amazed with how much power you have. This book isn't hard to read either. You'll love it if you reach this stage!

Javascript & jQuery:

Learning jQuery, Third Edition
http://www.amazon.com/Learning-jQuery-Third-Jonathan-Chaffer/dp/1849516545

jQuery is a framework built on top of the native browser language of Javascript. Usually one would recommend you learn the base language--Javascript--before learning an abstracted framework on top of it--jQuery. However because of the nature of jQuery and how comprehensive it is and because of how quirky Javascript is coming from PHP, I found it best to jump to jQuery and immediately start accomplishing the DOM manipulation tasks I needed. And ultimately because of
syntax similarities between PHP and Javascript I was able to get productive in Javascript without studying a single book just on Javascript. One thing I did different

when studying this book from the PHP books is I did every single tutorial as I read it. The reason is because when I learned PHP, I was learning my first real programming language--so it took me a lot of time to just digest things before I could code a single line, which is why I just read PHP book after PHP book before I got started until it all made sense. However, by the time I got to Javascript & jQuery, I understood how programming in general works and found it helpful for memorization purposes to immediately start doing the tutorials.

jQuery 1.3 with PHP

http://www.amazon.com/jQuery-1-3-PHP-Kae-Verens/dp/1847196985

With this book I didn't do all the tutorials like I did with *Learning jQuery*, but what reading this book did for me is taught me precisely how Ajax works and what it's all about. After reading it, coding features that required Ajax using Yii and PHP was obvious and a no-brainer.

JavaScript: The Good Parts

http://www.amazon.com/JavaScript-Good-Parts-Douglas-Crockford/dp/0596517742

This book gave me a deep understanding of the Javascript language and what it's truly all about. After reading it, many hours of debugging and head-scratching when coding Javascript & jQuery were removed from my schedule--because I finally learned the quirks of the Javascript language I needed to know.

Pro JavaScript Design Patterns

http://www.amazon.com/JavaScript-Design-Patterns-Recipes-Problem-Solution/dp/159059908X

Now this book took my Javascript game to the next level, gave me an idea of how jQuery was built, taught me how to do things similar to how you would in a "classical" OOP language like PHP, and completely ended any remaining head-scratching I was having with Javascript, particularly with how "scope" works in Javascript.

Linux:

The Official Ubuntu Server Book, 2nd Edition

http://www.amazon.com/Official-Ubuntu-Server-Book-2nd/dp/0137081332

Note: by the time I read this book I had already learned Linux through blogs on the internet. The best thing I can recommend you do is install Linux on your computer from the Ubuntu website, and start navigating around the command line, practicing Linux commands you learn off the web. Just google "linux tutorials" and you'll be off to a running start. That said, by the time I got proficient in Linux and after I read this book, I felt confident that I really knew what I was doing and had practical solutions for the most common problems you'll face at the command line.

Apache Cookbook: Solutions and Examples for Apache Administrators

http://www.amazon.com/Apache-Cookbook-Solutions-Examples-Administrators/dp/0596529945

This book I treat like a pocket reference and still refer to it often since it's impossible to remember all the different Apache configurations, given how comprehensive this web server application is. I did read it through when I first got it. I kinda skimmed it though--just to get an idea of what is possible. Getting an idea of what is possible without mastering a subject matter is so important in programming because you'll know where to look when you face a challenge that the subject matter can solve.

Pro Bash Programming

http://www.amazon.com/Bash-Programming-Experts-Voice-Linux/dp/1430219971

This is a little advanced for readers of the *FaceySpacey Bible*, but I'm putting it here because it really took my Linux skills to the next level.

NON-TECHNICAL BOOKS:

Smart and Gets Things Done: Joel Spolsky's Concise Guide to Finding the Best Technical Talent

http://www.amazon.com/Smart-Gets-Things-Done-Technical/dp/1590598385

If you plan to grow a small application into a large company, this book is a must. It's short. Read it.

SEO Book.com

http://www.seobook.com

This obviously isn't a book, but I read their entire site like a book, and its creator, Aaron Wall, expects you to read it like a book. When I was done reading it, I felt I was completely up to speed regarding what SEO is, how search engines work, and practical techniques to get better rankings in search engines.

For daily Startup Wisdom, checkout FaceySpacey.com/blog daily. And don't forget to download the entire No Bullshit FaceySpacey Bible or more individual chapters here: http://www.faceyspacey.com/resources?section=book .

*Thanks again from FaceySpacey and be sure to check out **FaceySpacey.com** often for further knowledge we kick to take your Startup to the Stars!*