# The No Bullshit Bible : Creating Web 2.0 Startups & Programming

Written by James Gillmore
Edited by Holly Welch

FaceySpacey
www.faceyspacey.com

# HTML / CSS

TABLE OF CONTENTS

## 2-1 HTML/CSS | INTRODUCTION TO HTML ELEMENTS

In this tutorial I'm going to teach you how to learn HTML. You're going to learn it mainly from w3schools: http://www.w3schools.com/html/default.asp

You're also going to have to study their CSS tutorials: http://www.w3schools.com/css/default.asp

So what does CSS have to do with HTML exactly? CSS is all about styling the content of your page, i.e. it's colors, placement/position, fonts, borders, etc. HTML is all about the actual content on the page. HTML without CSS generally will appear very plain and stacked on top of each other. CSS allows you to build columns and blocks on the same line. It's biggest strength is therefore helping you to position your HTML in pretty structures.

So what's great about the w3schools HTML & CSS tutorials it they have a tool to test your HTML/CSS right there. You can make a change and click a button and see what effect it has immediately.

So on the w3schools site go through the links in the left column, top to bottom, studying each tutorial in order and whenever you are presented with an option to test something out, do it. You will arrive at an editor like this one: http://www.w3schools.com/html/tryit.asp?filename=tryhtml_intro

For example, in that window, you will see this:

```
<html>
   <body>
      <h1>My First Heading</h1>
      <p>My first paragraph.</p>
   </body>
</html>
```

Try adding the following line after the line with the *<h1>* "element":

```
<h2>My Sub-heading</h2>
```

Basically in html, u wrap your text content in "*elements*." An element itself is made of up an "*opening tag*", e.g: *<h1>* and a "*closing tag*", e.g: *</h1>* . The closing tag is the same as the opening tag except it has a forward slash in front of the name of the type of element. Often people confuse the definition of a "*tag*" and an "*element*." An element is the opening and closing tags and the content in between. A tag is just the beginning keyword or the ending keyword, and of course is wrapped in "< >". Tags themselves are just a short keyword wrapped in those arrows. There are many keywords you'll have to learn, but you'll find that to be the easiest part. Don't worry so much about the tag type/name. Really the way to think about it as if they all have the same name. The only difference is some provide built-in styling features that style the content in between the tags in different ways, usually in how it's placed. In CSS you can also refer to these tags to style only elements of a certain tag type.

Generally HTML is just a bunch of nodes/blocks written on a page like this. You can also put them inside each other, for example:

```
<div>
   <h1>Title</h1>
</div>
```

Here the *<h1>* element is "nested" within the * * element. Think of it like a tree, like an "outline" in school. When viewing various tools examine HTML (such as Firebug in Firefox and Inspector in Chrome and Safari) you can expand and collapse these nodes, e.g. hide all the sub nodes nested within a parent node. This allows you to quickly browse through HTML code. It's similar to collapsing/expanding sub-folders in a file/folder browser on your desktop computer.

The "*div*" tag is one of the most common tags you will use. It's just a generic tag for dividing and grouping blocks of content. I assume it stands for "divider." Anyway, just know that you'll end up with is a tree of tags nested within each other.

## 2-2 HTML/CSS | INTRODUCTION TO CSS SELECTORS

First off "CSS" stands for "Cascading Style Sheets." The main idea is you can write "sheets" of code just for styling that's separate from the HTML content. Imagine you have a bunch of content on a page. You have a * * tag that groups a block of content like this:

```
<div>
   <h1>Title</h1>
   <h2>Sub Heading</h2>
   <p>Paragraph in an article for example...</p>
</div>
```

To stylize this, you can specify neatly in code elsewhere the following:

```
div  {border-style: dotted;}
div p {color: blue;}
h1 {font-size: 18px;}
```

The beauty of CSS are these "selectors." The selectors are the parts before the opening curly brace. They let you select the content on the page and refer to it in order to apply styles to them. This is a great thing because it separates styling code from content code, and the result is 2 places where 2 different types of code is and they are therefore more concise and easier to read. The messier way to code this with HTML and CSS styling code combined is:

```
<div style="border-style: dotted;">
   <h1 style="font-size: 18px;">Title</h1>
   <h2>Sub Heading</h2>
   <p style="color: blue;">Paragraph in an article for example...</p>
</div>
```

See how the content-centric HTML code is now busier. It makes it harder to read the CSS styling code and the HTML content code. With the code in 2 places as in the previous example it's easier to read each set of code. The benefit is also that you can write CSS that will affect more than just one block of code. All * * elements and all *<p>* elements within * * elements and all *<h1>* elements on the page will have the same CSS styles applied to it, and yet you only had to write that CSS once! So that means even less code. If you did it the "inline" way above, you would have to re-write the same CSS rules (more technically called "properties" and "values" in multiple places. Note: something like "border-style" is called a "property" and "dotted" is that property's "value."

## 2-3 HTML/CSS | MORE ON CSS SELECTORS

There are several other very useful must-know tools in CSS. Specifically, these are more selectors, but of a slightly different flavor. You can specify rules like this:

```
.myText {color: blue; ont-size: 18px;}
#coolStyle {font-weight: bold; color: red;}

<div>
   <p class="myText">Paragraph in an article for example...</p>
   <p class="myText">Paragraph in an article for example...</p>
<p class="myText">Blurb 'n <span id="coolStyle">cool thing</span></p>
</div>
```

So the 4 words/characters to notice above are:

```
.
#
Class
Id
```

Basically the period and sharp symbol are CSS hooks that connect to respectively to their HTML counterparts, the "*class*" and "*id*" attributes. "Attributes" by the way are extra pieces of information which you put within opening HTML tags. The "style" attribute you previously saw to supply "inline styles" was another type of attribute. So the idea is that the above *<p>* elements have the styling provided by the *.myText* selector, and the ** element has the *#coolStyle* style supplied.

You use classes when you want styles to be applied to multiple elements when you can easily refer to them (i.e. select them) with simple element-named selectors, such as the ones you saw in the last CSS tutorial. For example:

```
.myText {color: blue; font-size: 18px;}
```

would apply to all * * elements on the page, and we only want to apply that style to some divs. That's when you use class selectors.

You use *ID* selectors when you want the style to only apply to one element on the page. Think of *ID* selectors as a way to give an element a unique identification name, i.e. "*ID*."

## 2-4 HTML/CSS | SUMMARY OF HTML & CSS TERMS

I'm going to finish the HTML/CSS section of our tutorials with a description of the main terms you learned:

*1) **CSS** - Cascading Style Sheets*

*2) **Selector** - a CSS directive that refers to an HTML element or multiple elements*

*3) **Element** - an opening and closing HTML tag and the content in between*

*4) **Tag** - the actual text between < > as either an opening or closing tag*

*5) **ID** - identifier for a unique HTML element whose selector is a sharp symbol*

*6) **Class** - a custom selector beginning with a period that allows you to select all HTML elements on the page that have a matching class attribute whose value has the same name as what follows the period*

*7) **Attribute** - a text string within an opening tag whose value follows an equal sign and is enclosed in quotations; generally speaking, it further defines the behavior of the element.*

Ultimately, the main concepts of HTML & CSS are very easy to grasp, and these tutorials sum them up. It becomes more challenging when you put it into practice and realize your page doesn't always behave the way you'd expect, especially once you start checking it in all browsers and realize different browsers render the page differently. There are a lot of nuances with CSS especially in terms of how the page behaves. There are also a lot of CSS properties to learn, with some like the "float" property doing some unique things that take a while to get the hang of. That said, the basic syntax is pretty much summarized in my HTML tutorials, and more than enough to give you a running start. If you can get this, you should just start coding some HTML

pages, and try out different CSS properties you learn about on w3schools: http://www.w3schools.com/css/ .

Try them all until you see what's possible. The most challenging part will be positioning, not things like colors and borders and fonts. 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!*