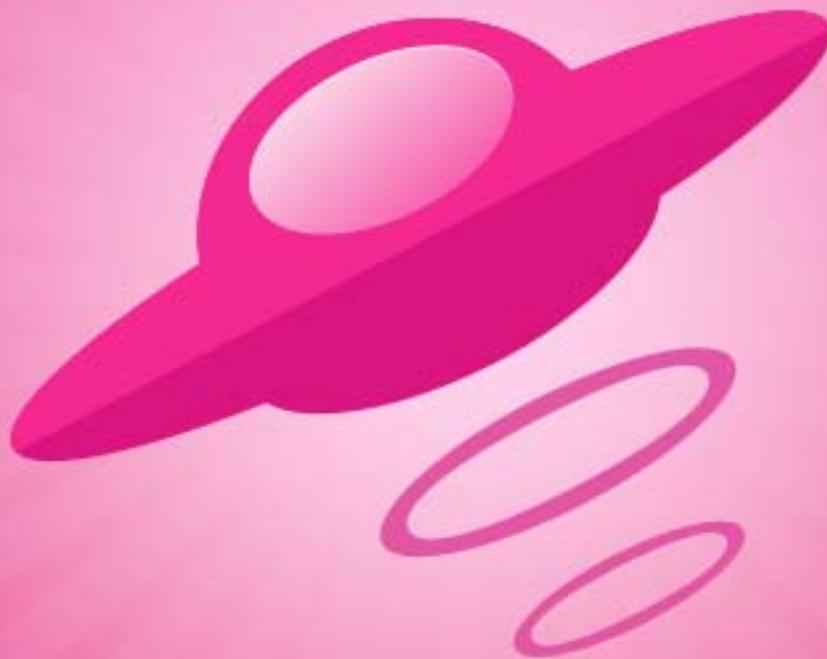# The No Bullshit Bible :
Creating Web 2.0 Startups
& Programming

Written by James Gillmore
Edited by Holly Welch

FaceySpacey

www.faceyspacey.com

FaceySpacey Bible - The No Bullshit Bible: Creating Web 2.0 Startups & Programming

12-1 SEO | BASIC THEORY BEHIND SEO

SEO until you know it is often perceived as something quite mystical. The truth is it's not really. Let's discuss the history of SEO a bit.

Up until 2000/2001, Yahoo ruled SEO. They weighted pages based on the onsite usage of keywords. Pages that used a keyword phrase the most came up at the top of search engines.

When Google came around, they did something quite magical which made it so keyword-stuffed spam pages didn't ruel the search engine results pages ("SERPs") and ultimately made the cream rise to the top. What they did was index all the links across the web from one page to another. This task was actually extremely complex. Imagine your webpage links to my webpage. Imagine your webpage has 5 links pointing to it (from other sites). Now imagine another webpage links to my webpage, and this webpage has 500 links pointing to it (again from other sites). Which link do you think would be better for me to get to point to my site? Your link or the one from the other guy that has 500 links pointing at it? Obviously the latter. Well, this was a big undertaking for google because the "web" of links from one site to another is ultimately endless. It's not just a chain of 2 or 3 sites. It's very many linked one after another. So Google wrote an algorithm to go for many degrees of separation and track the weightings of these links, each step of the way tracking how many links each page has and the corresponding weight those links have, and so on.
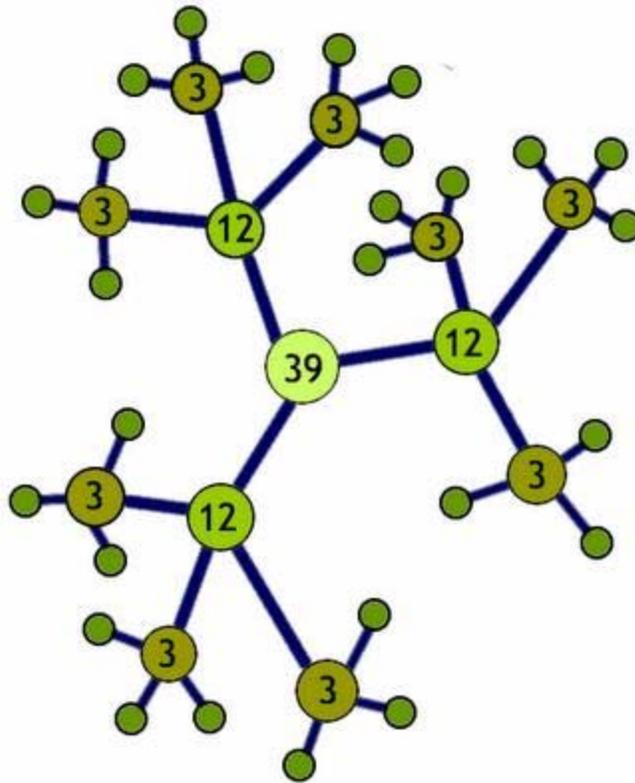
The next thing they did was track which keywords (and keyword phrases more specifically) were attached to these links. A link looks like this: Click Here. In that example, "Click Here" is called the "anchor text," and by the way that HTML as a whole an "Anchor" element. So what google did is factor in the "anchor text" associated with each link to determine what keywords your site should rank for. They combined this information with the information previously described about link weighting to determine what pages should rank the highest for an endless combination of keyword phrases. And modern SEO was born, and your rankings are still very much based on these algorithms today.

12-2 SEO | INTRODUCTION TO LINK ACQUISITION

So if you read the last tutorial, it's probably crystal clear that all that matters to rank for your keywords in "Google" (collectively used to refer to all search engines since Google has like a 75% market share) is how many links you have pointing to your pages with your all important keywords, and the quality of these links. But how do you get them?

There is basically 4 ways:

1) analyze the links your competitors have pointing to them, and get links from the same websites and webpages.

2) buy powerful paid links through the various paid link marketplaces

3) hire a cheap Indian firm to spam every user generated site on the web with links to your webpages.

4) make your content very linkable. This means basically be a great "product" (i.e. information resource) that people will be very likely to link to.

In this and the following tutorial we're going to cover **#1** only. **#2** is easy if you want to do it. The risk is that Google really really doesn't like this and if you're caught in a network of sites that are all busted for doing this (which marketplaces that sell these paid links bring together), you may get penalized. **#3**actually works very well and has for some time, as ultimately you're putting links in natural places (i.e. user generated content sites where anyone can submit), but just automating how quickly you can do it. And we also won't cover **#4** either because the short of it is you need to be exceptional. For example, be the first to pioneer blog tutorials in some new or obscure niche. Basically, popular blogs like Techcrunch.com will get lots of links naturally. Also there is the term "link bait" to consider, which means writing articles that are more likely to be linked to. For example, articles that compile and list a bunch of related resources are known for getting lots of links. One on-site SEO thing you can do here is provide a field where users can copy/paste an article's URL--this will inspire people to link to you by making it easy to do so. Share buttons for Facebook, Twitter, etc, do the same thing basically.

So for #1, the main idea is that your competitors will have links you want. By analyzing what links they have, you will figure out where you can also get links. How to go about it is a little more complex, and is what we're going to cover in the next tutorial.

12-3 SEO | LINK ACQUISITION RESEARCH FORMULA

The following is a precise formula to find the best pages and sites to get links from. These links FYI are called "backlinks" or "inbound links." It's a combination of keyword research (i.e. finding the keywords that will send you the most and best highest converting traffic), competitive research, and generally cross-referencing data to find the most powerful links.

1) *Do keyword research using keyworddiscovery.com, semrush.com , and Google's "Keyword Tool": https://adwords.google.com/select/KeywordToolExternal . The goal is to find the keywords that get the most monthly searches on Google at this point. Those tools will help you find more and more keywords by entering the keywords you know your users search for. You'll most likely be surprised at how many other keywords you find, especially when you find keywords you never thought of that get more monthly searches than your initial seed keywords you started with. SEM Rush will help you find keywords by looking at what keywords send the most traffic to your competitors. You can use all these tools in combination to expand your keyword list, continually expanding your list of keywords, and learning new ones from the previous ones you discovered. Just think about it as a never-ending synonym finding game where one keyword will help you find another and so on.*

2) *Find the most effective keywords, measured by the ratio of # of monthly searches to the amount of pages indexed for those searches.* Google tells you the # of pages that contain your search phrase. You already know the # of searches your keywords get from the tools listed in step #1. So simply build this ratio here by putting the # of searches your keywords get in the denominator. Link Assistant's Rank Tracker application will automatically do all this for you:

http://www.link-assistant.com/rank-tracker/

That ratio by the way is called "KEI," which stands for "Keyword Effectiveness Indicator." The net result will be you can sort a list of all your keywords with the ones with the highest KEI at the top.

 3) *Take your keywords, and find your top competitors in Google.* You can either eye-ball it by seeing which sites come up the most often across your keywords (specifically your keywords with the

highest KEI), or automate it with a script. The script, which I can't share at this time, should grab the top 20 listings from google for each of your keywords, and record their position in terms of points (i.e. where a position of 1 which is great is equal to 20 points), and then whenever the same site is found again, it would add more points for that site. Then at the end, you can sort your list of sites discovered by those that earned the most points--those are your biggest competitors. You'll want to remove sites like about.com and wikipedia.com since they're generic and cover many niches (not just yours), and therefore not true competitors.

4) *Find as many backlinks as you can from your top competitors*. The list of competitors for which you find backlinks can be as big or small as you want. I usually use no less than 10 competitors and no more than 100. In this step, you're now finally finding potential pages and sites to link to you. But there is more to it...

5) *Then find the "link hubs" that link to the largest number of your competitors*. You will find that the same site is linking to multiple of your competitors' sites. These sites are therefore very likely to link to you too. If you find individual pages that link to your competitors, there is a high chance you can get a link on that page, as it's usually a list of competing/similar sites. You can eye-ball this by comparing each competitor's list of backlinks, or run them through a script that finds the backlinks that occur the most.

6) *Take all the backlinks that link to at least 2 of your competitors, and collect the following data on them by importing them into Link Assistant's SEO SpyGlass application (*http://www.link-assistant.com/seo-spyglass/ *) :*

-Alexa Rank
-domain Page Rank (Google's formula for the amount of link juice a site has received)
-Webpage Page rank (Google's formula for the amount of link juice an individual web page has received)
-yahoo link popularity
-their page titles
-the # of inbound and outboud links on each page (if they link out to too many places, they don't have much link juice to pass on to you)

SEO SpyGlass will list all these backlink pages and let you filter by the above columns of data to find the best pages to get links from. You can also export the data into a spreadsheet, and write a formula to combine all these pieces of information into cells in one column that you can then filter by. The idea is that this single column would contain the master score. The pages at the top are the first ones you want to go after to get links from.

Note: if you didn't find many link hubs web pages, it's fine to produce the above list in SEO SpyGlass using all your backlinks. You'll want to do both, starting with link hubs you find.

7) ***Now you need to contact all these webmasters and request a link on their site***. LinkAssistant's self-titled program, LinkAssistant, will help you do this. Check it out: http://www.link-assistant.com/linkassistant/ .

That program makes it very easy to email out requests to the sites found in the previous steps. Basically you can import all the previous backlinks and linkhubs, and start an email thread with the owners of those sites, and keep the thread associated with the initial backlink so you can track communication with many webmasters. You'll want to checkout the actual backlink pages and the sites that contain them to get to know the webmaster you will be requesting a link from. The email you write will need to be smart if you plan to get a link. Like, don't just ask for a link on a given page, but rather try to add value to their site. If you know you're link may go good in an article that discusses several other complementary sites (that aren't your competitors), then you may want to write that article and provide a list of those sites and suggest the webmaster add the article to his site.

That all said, LinkAssistant also has several tools itself to find potential link partners and skip the last few steps. It can automatically find sites that link to your competitors, find sites by keyword, and even find sites that have article submission forms and then automate the submission of the content you want to post to it. These tools aren't as thorough as my above process, but it can make it a lot easier if you want to skip a few steps and just start getting links. The benefit of doing my complete research techniques above is that you'll basically leave no potential link unturned.

8) The last step is to use LinkAssistant's Rank Tracker application: http://www.link-assistant.com/rank-tracker/ . That application will produce graphs of your rankings (in Google, Yahoo, etc) for all the keywords you import. Simple as that. You can of course also filter lists of your keywords by their ranking position, KEI, # of searches, competition, etc. Rank Tracker will also

generate a mini site you can send to clients summarizing how your keywords are doing. Of all the applications mentioned in this tutorial, you'll want to use Rank Tracker no matter what, even if you go hire an Indian firm to get you tons of links from user generated social networking sites. There are many applications that help track your ranking, and I've always found Rank Tracker to be the most feature-packed over the years.

FINAL NOTES: the above formula can be modified and you can skip steps. The point is that it highlights key roundabout ways to get unique and helpful information, such as the part about finding link hubs; and it also explains how to do all this research thoroughly. If you understand how to do it thoroughly, you can start with bite-size chunks of whatever parts you find easiest. Inevitably you'll end up with huge lists of backlinks, and you'll need to find various ways to filter the lists down to just those that will give you the most link juice and are most likely to link to you. Sites that let you submit to them are highly likely to link to you because you can do it yourself. Pages that list tons of sites are likely to squeeze you in there, and often direct you to contact them if you'd like to be added. Small blogs with personal webmasters are also likely to link to you, but only if you can give them a worth wild reason to do so. Coming up with that reason will be up to you to try and figure out until you determine what aspect of your business these guys are likely to write about. This will also be very effective from a PR standpoint.

12-4 SEO | ONSITE SEO

Now that you understand the hard parts of SEO, which is the theory of how it works and the complexities of getting quality backlinks, let's discuss the simplest aspect: ONSITE SEO. Onsite SEO is mainly about the following things:

1) the keywords in your URLs, and that they are formatted nicely with hyphens

2) the title of each page, as embedded within the <title> element of your <head> element in the HTML

3) headers and sub-headers of your content within <h1> and <h2> tags, etc.

4) the body's of your articles being keyword rich

5) your description met tag providing an enticing description that will make google searchers want to visit your site when they see the description in your search listing
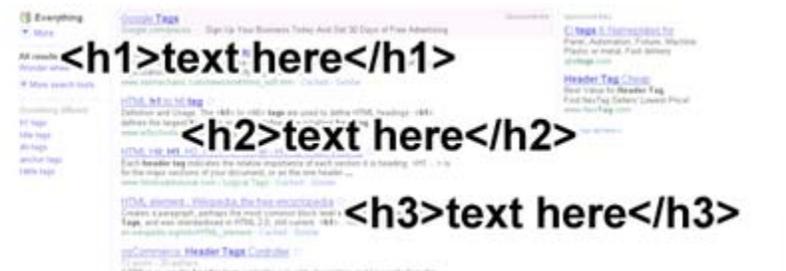
I won't teach HTML here because you can learn it in the HTML Tutorials. But the idea is you need to focus each of your web page on a set of keywords, and then title the page according to it, make the URL match, make the titles above paragraphs of content (the h1, h2, and h3 tags) full of those keywords, etc.

You're not just going to want to repeat the same keywords over and over. For the overall page title and URL, sure, but for the sub-headers above paragraphs, you'll want to vary up the keywords to similar and complementary phrases so you can get some "long tail" traffic from different combinations of your core keywords.

For example, if one your keywords is "New York City Hair Stylists," you may have a paragraph header called: "Best Female Hair Stylists in New York." capiche. That will help you rank for more combinations of keyword phrases that contain your core keywords, which in this case is "Hair Stylists."

Overall, the main thing is that the code of your site generates all the urls, <title> elements, etc properly, and then that you intelligently supply nifty keyword combinations for the titles and sub-headers, etc. I usually don't worry about anything except the main page title, and let the article come naturally. If you're a blog site,

you're churning out tons of content all the time anyway, so you don't need to worry so much per article about capturing the most number of complementary long-tail keyword phrases. However, if you're a small company site with only so many pages on your site, you'll want to go so far to apply all the keyword research you did to collect backlinks (as described in the previous tutorial) to determine what keyword phrase combinations you should flush your pages out with.

12-5 SEO | SPECIAL SEO TECHNIQUES

This tutorial will cover a few nifty & unconventional ideas you can do to get more keyword rich links pointing at your site.

One of my favorite techniques is building tools that get other people to do all the linking work for you. For example, if you have a video site, you can make your video player shareable and embeddable on another site, and when visitors do that, a link is pasted below the flash player. You can of course do something similar with any copy/paste widget you want to provide, for example a little widget that shows your latest articles or tweets. Generally, provide a box to copy/paste some HTML to another site, and put your links in it.

In general, you can make it so each of your articles has a field to copy/paste a link to another site. And of course you want your Facebook Like, Tweet buttons, and Google +1 buttons, which ultimately do the same thing, though all to the same 3 sites, which isn't as good as getting links from new unique domains.

You can write content that calls for action on behalf of your readers. For instance, you could ask your readers to review their favorite articles on your site, or perhaps rank recent products listed on your site. You can do things like tell them you're tracking who links to your site, and give out some sort of prize to the best articles that link to you.

Techcrunch, for example, all the time gives out prizes to people that tweet at them. Do the same for full fledged blog articles. You could also find a bunch of sites in your niche, and give them coupons if they write about your site/service (and of course link to you). You can give them enough coupons to offer them up as prizes to their own readers.

You can guest write for other blogs, and get yourself a link while doing it in exchange. You can also grow your own content by allowing other people to guest write on your site. You can build relationships with other webmasters/bloggers this way, and generally grow your network, and accordingly have more relationships with experienced internet experts you can eventually get links from.

In general, there are lots of things you can do if you're creative. The purpose of this tutorial was to get your mind thinking so you can invent your own techniques. You just need to participate with other people and sites on the web. Show them love without asking (i.e. link to them), and eventually you'll get links without asking. They'll find you when you link to them in one way or another (i.e. through tracking traffic from google analytics, doing their own SEO research, when their visitors mention they came from your site, etc). So in essence, link out to get linked to. Be a player in the web game, and it will come back to you.

12-6 SEO | BUILDING SEO TRAFFIC & CONVERSION PROJECTIONS

One of the hardest parts of SEO is projecting what results you will get out of it. If you're an SEO service provider you'll be sure to convert more clients, and if you're the head of your SEO department at your company, you'll be sure to please your superiors. Below is our process for producing such projections at FaceySpacey. Each step below describes a column in a spreadsheet you will produce. Here is a sample spreadsheet with all the formulas built in (feel free to download it in Excel format to your own computer):

https://spreadsheets.google.com/spreadsheet/ccc?key=0AoURnKZh8J-SclRCd0NKTWROWlFVTWVtUkUtVXBUX1E&hl=en_US

And here's a Prezi presentaton that will help you see what each column in the spreadsheet means: http://www.faceyspacey.com/resources/seo/seo-6-building-seo-traffic-and-conversion-projects

 Here is the in depth version of what it all means (refer back to that spreadsheet and Prezi presentation as you read this):

1) We list all your keywords potential clients/leads are likely to search for in the first column of the spreadsheet.

2) For each keyword, we go to Google's "Keyword Tool" previously described in the last SEO tutorial, and discover how many people are searching monthly for each of your keywords and list them in this column.

3) We realize that google isn't the only way people search the web and use the coefficient of 1.5 as a multiplier (across all cells in this column) to determine how much traffic a search phrase gets across all search engines.

4) We prepare what conversion rate we think we'll get out of your visitors, and enter it across all cells in this column. For existing sites, you should have this data. For new sites, you'll have to give your best guess based on your knowledge of conversion rates in your industry.

5) Next, enter the average conversion amount in all cells in this column. If you sell a service where a client keeps coming back over a long period of time, just estimate how much the average client spends with you per year.

6) Now Before you even get any clients or customers from search engines, you need to determine what percentage of them actually click your link in a google, yahoo or bing

search results page. This is called the "Click Through Rate" or CTR. Top listings get the most clicks, and the lower your listing is the lower the CTR percentage is. So if 200 people search for "New York Plumbers" per month, and 23% click the top listing (i.e. the top listing in google has a 23% CTR), then that listing gets 56 visitors per month from google. In three columns here we'll assume we have a 1st position ranking, a 2nd position and a 3rd position. Industry standards are 23%, 6.5% and 4.6%, respectively. Fill up all cells in each of these columns with those values respectively.

7) Now, that we know the CTR, average conversion rate, and average conversion amount, we can determine the number of visitors you will get from search result pages, the number of conversions and the dollar amount sum of all those conversions. Add 3 columns for these metrics next to each of the 3 columns in step #6 above--this way we can predict these things for each potential ranking position you may get for your keywords.

Over time you can plug in more and more accurate conversion rates and average sale amounts as you see the real thing in google analytics--and the whole spreadsheet will update if you've properly used math formulas between cells. To do this, you of course need to make, for example, the total sales column multiply the values of the total visitors times the conversion rate times the average conversion amount.

Also, you'll be able to hone in on the precise metrics you need to improve. For example, you may realize that it's more important to increase the average sale amount than it is to increase your positions in google search result pages or conversion rates.

8) Now, in a separate sheet, we add up the totals for the number of visitors, conversions and total sales amount across ALL your keywords in order to project you're aggregate results. We break this down into 3 groups based on the the top 3 positions you'll be shooting to get, like how we produced similar metrics for each position in step #7 above.

9) Finally, you will have your spreadsheet tell you--through the use of formulas across cells--what your monthly and daily totals are. It should also let you put in some coefficients that correspond to the fact that their might be more keywords you're targeting than in this sample pool of keywords. For example, in the sample spreadsheet provided, you're looking at only 30 keywords. More often than not you will be working to rank for hundreds of keyword phrases not here, which means additional traffic. The yellow rows in the sample sheet let us say that there will be 75% more traffic from other high traffic keywords, 50% more traffic from from keywords with a medium traffic level, and 25% more traffic from keywords with a low traffic level. The idea in this example is that the keywords that will get only 25% more traffic are what we call "long tail" keywords and might end up being something like: "free online new york city plumbing." A Short tail keyword would be "new york plumbing." After that in the orange rows, we add the complete aggregate up of all your potential keywords.

So that's our technique. The following visualization, using Prezi.com's presentation tools, may help you grasp it better. See it on :

http://www.faceyspacey.com/resources/seo/seo-6-building-seo-traffic-and-conversion-projects

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 .

Á

V@ą\•Ać*æą Á¦[{ ÁÐœ&^ˆÙ]æ&^ˆÅ€ą åÅ₳^Á˘¦^Å{Æ&@&\Å˘oÆFaceySpacey.comÅ[℮^}Á
₳¦Å˘¦c@¦Á}[¸|^å*^Á¸^Á₳\Å{Æ₳�^Á[˘¦ÅÛœ₊cˇ]Å{Æc@ÅÛœ₊•Ã

Á