A DECADE OF MAKING APPS

BY MICHAEL SCHWARTZ

michaelsboost.com

Introduction:

Hello my name is Michael Schwartz. I've designed and built over 100 free and open source applications over a wide variety of categories over the years such as design, development, fitness and even games for well over a decade. I've developed many apps that have exceeded thousands and even a million downloads such as CamDesk and even my simple Alphabetizer application. If you'd like to see all the apps I've developed you can visit my website at michaelsboost.com Where I also give you a free source on beginner software development as a gift just for visiting my website. No Strings Attached! My work has also been featured on popular sites such as OMGUbuntu, MakeUseOf, Infotonics Media, Web Designer Depot, etc:

I've also contributed to several open source projects such as Firefox, GIMP, Blender, Codemirror & others. I've also been interviewed and featured by Daniel Davis from tinkernut.com for my work. So I think it's safe to say that after spending over a decade of my life designing and developing apps I'm not only credible but I've also learned the pros and cons of what to do and what not to do.

You see when I first was introduced to the World Wide Web. The internet was a very different place back then. There was no Instagram, SnapChat or TikTok. As a matter of fact cell phones weren't even able to browse the web. That all came years later. Interactivity and animation now is done almost entirely through HTML5 and Javascript, while back in the early 2000's it was almost all done in Adobe's Flash. I still remember the days of when I got a MySpace account and was introduced to the world of "DIV Overlays" which allowed designers like myself to customize my profile however way I wanted. It really gave us complete creative freedom to show off our profiles anyway we wanted. Which sparked my interest in learning how to code. Back then I recall watching how to videos where people would record their desktop and have their Webcam overlayed above all other apps. I thought that was really cool which sparked me learning how to code and eventually develop CamDesk. Of course when starting out I had some bad habits which I later had to break and adjust to changing conditions within the market. Such as redesign my apps to fit the modern "flat design" style and even make them responsive. You see I got into this reality because I really love it. I love innovation! I really do. I didn't make over 100 apps for the acquisition of money, that's why they're all free and open source. I wanted to do my part to help drive innovation even more by making them open source. So people can not only learn from my work but contribute to it and/or even fork the project with their own updates and renditions. I truly believe the world is a much better place as long as we're cooperative. Sure we all have differences of opinion, different styles and cultures but that's the beauty of it. That's what makes this world so beautiful.

Table Of Contents:

Chapter 1: Learn From My Mistakes

1. Sketching and Writing Down Goals and Workflow

If I could go back in time knowing the mistakes from my past one thing I wish I did more of (especially in the beginning of me starting to make apps) is sketching the app idea out along with writing down functionalities, goals, objectives, etc:

2. Ask for Help

By doing that I would've saved a lot of time. I never had a problem asking for help when needed, although there was a lot of times I asked for help prematurely and later I had fixed whatever the problem was either by resolving the problem myself in code or from finding a helpful resource searching Google.

3. Do Your Research

Becoming familiar with other frameworks and libraries is imperative to evolving and progressing both as a designer and as a developer. Sometimes the usage of such libraries may otherwise be relevant to your project and sometimes it/they may. It really all depends on your project, timeline, budget, etc: In addition it's imperative to be upto date with the latest W3C standards. For example the center element/tag has been obsolete for MANY years as it's been replaced in favor of using CSS for centering. Therefore it's important that you understand what these languages mean, what they're for, how to use them and what attributes and elements are standard vs obsolete. You don't need to know all as I don't know all even after making apps for over 10 years. Some elements you will almost never use like the picture tag as the img tag has been the standard since I began coding years ago.

4. Failing Isn't Always Failure

I'm glad for the mistakes I've made as they helped me become a better designer and programmer. We need these pains to progress not just in technology but in life as well like in the gym. We need to realize that failing isn't always failure as we need those stresses to help strengthen our growth. So always remember to embrace failure.

5. Keep Your Code Neat

This should be obvious but common sense isn't so common now a days. If your code is neat and easily readable, collaboration will be significantly easier.

In addition document how users can contribute to your code either it be your team or if it's open source. Make sure that whomever is working with you either closed or open that the information is easily understood as the best apps tend to be those that a child can use. No point in overcomplicating it. Facebook beat MySpace because kids and most adults generally don't wanna learn how to code. They made a stylish page in which everybody can still express themselves while having the same profile placements.

6. Take Time Away For Your Mental Health

Sometimes projects can become overwhelming. It could be a deadline, budget, client, a bug, etc: I found (at least for me) that oftentimes if I become stressed to get off the computer and find something else to do to clear my mind and relax. That could be reaching a book, taking a nap or even going for a walk. Really whatever works best for you. After I did that I was able to come back level headed and fixed whatever the problem was.

7. Adjust To Technological Changes

Some apps eventually die off. If you're using an app that does you need to find another real quick and simply adjust to those changing conditions. The same way us web designers had to adjust to make our websites and apps to responsive after the advent of smart phones.

8. Available In Multiple Languages

If you want to increase the availability of your app to reach more people it's imperative that your app isn't only available in English. If you do not know any other language either learn the language or hire someone to help it's as simple as that. If you're learning you can have a friend proofread in which he/she can find any errors in your grammer or punctuation. I learned this myself when making my apps WebDGap and Budjut to be available in both English and Spanish.

9. Don't Repeat Yourself

There's an acronym called DRY which stands for "Don't Repeat Yourself". It's meaning is simple. Don't repeat writing the same code multiple times in your app. For example, It's far more of a time saver to write it in a variable or function and just call the function.

10. Lastly Don't Do It For The Money

Problems are often hidden as opportunities which I'll discuss later in this book but don't get so focused on the money and not enjoy the process. When I turned my hobby into a job I got burnt out and didn't wanna do it again or at least not for a long time. That's because I was doing it for the money and focusing on that rather than just enjoying the process that I had Initially loved before.

It reminds me of a quote from Bruce Lee that if you focus too much on the finger you'll miss all that heavenly glory and is still relevant to the niche of design and development but even within life itself.

Chapter 2: The Boring Stuff

It sounds exciting to make an app that generates different character designs or even a social network. However before you can do that you need to learn the boring stuff first. So this chapter is decided to getting the boring stuff out of the way before we begin making a simple website.

HTML (Hypertext Markup Language)

Is the most basic building block of the web. It simply defines your web pages content.

CSS (Cascading Style Sheets)

Is used to define the appearance or presentation of your website.

Javascript

Handles the behavior or functionality of your website.

You can store data and information a few different ways. The most common is saved on a server which could be a MySQL server or a MongoDB server. Saving to the server is usually done in PHP or Node.JS. You can also save data by cashing in the browser using a Javascript object called localStorage. You can also use .json files to store information saved from the HTML5 FileSystem API and loaded in the browser using the FileReader API. You may also want to utilize Google's Firebase tool to store information on a server hosted by Google rather than yourself. It all depends on the needs, focus and goal(s) of your project.

API (Application Programming Interface)

An API lists a bunch of operations that developers can use, along with a description of what they do. The developer doesn't necessarily need to know how, for example, an operating system builds and presents a "Save As" dialog box. They just need to know that it's available for use in their app.

Tools:

There's an extremely large and diverse selection of tools available on the web to complete small and simple tasks as well as large and complex tasks. Everything from designing and development. Before you get started making apps or designing websites I think it's important to distinguish the difference between technical and Non-technical programmers. You see technical programmers use code while Non-technical ones use an IDE or Integrated Development Environment such as Bubble where you can actually create a Twitter clone without writing a single line of code as long as you're running a desktop operating system such as Windows, Linux or Mac.

IDE's (Integrated Developmwnt Environments):

Now not all IDE's are the same. Codepen for example allows you to code in html, css, Javascript and/or even utilizing preprocessors such as Pug, SASS, TypeScript, etc: (which we will not be going over in this ebook. As a developer it's essential to do your own due diligence in order to grow as a designer and/or developer) what I love about Codepen is you can not only see the preview of your project as you code it but you can share it with the community as well. There wasn't anything like this available for mobile users let alone as an offline app for desktop users which inspired me to create my app kodeWeave.

There are many other IDE's people use for coding such as ATOM or Brackets (my personal fav) on a desktop. For Chromebook users I recommend the Amazon's Cloud9. Android users I recommend ACode because it has a built in console which is essential to find and solve any potential problems or "bugs" in your app. Now for those users who prefer to work within a terminal then Termux on Android is a great app which you can use the app Vim as your ide for coding your apps. Keep in mind all these apps I'm recommending are free and I've used myself. I would never recommend ANYONE to use any app I have not personally used myself.

Platforms:

Making your app available for multiple platforms sounds pretty attractive. I've made apps such as CamDesk, kodeWeave and WebDGap cross platform. Meaning these apps run on more than 1 operating system. You can share them on sites such as Sourceforge or Github but in order to make web apps a desktop app would require you to use a tool like Electron. WebDGap is a wrapper I developed that makes the conversion process fast and easy but it doesn't utilize Electron instead it uses NW.js. If you want to release your app as a mobile app you can use PhoneGap or PhoneGap Build which WebDGap also speeds up that process a little bit for you as well. Now of course like I said before technology changes and it changes fast so it's important to be familiar with the code more so than the tools being used because sometimes you may need to move to a different tool like I did from Aviary's vector editor Raven to Gravit Designer.

I must also express that sometimes making 1 interface design available for all devices is fine and sometimes it's not. It really depends on the complexity of your app. Last thing I want to add with platforms is the easier your app is to use (both for kids and the elderly) the more downloads it'll get. If it looks ugly such as bad color combinations or very technical and hard to understand then the less likely people will download and use your app. Which I had learned the hard way.

Now there's a lot more I didn't go over such as html elements, attributes, classes, id's, etc: The reasoning behind this is so you do your own due diligence and read up as much as you can and gain the experience required to grow. Which can easily be done on the Mozilla Developer Network and Codepen.

Chapter 3: Problems Are Opportunities

Majority of the apps I've developed I made to solve a legitimate problem I myself was having at the time or a problem I heard others were having and I took advantage of it while still making it free and open source. Some examples I already mentioned but some others are my workout app, Eye Workout, svgMotion, SVGAnimFrames, Character Party, among others.

I'm gonna give you a little background as to the reasoning behind writing this book because it correlates with the topic being discussed. In the year of 2017 I joined a big Multi Level Marketing Business and I got hooked on the vision and dream they had preached, looking back it felt as if I was tunneled in a trance. Now back then I was already swing trading in the stock market and stopped to focus on building my so called "business" which in all honestly I was nothing but a sub contractor to this big business who was merely selling a dream vision or as we called a "business opportunity". I was never a big energy drink person and even though fitness has always been a major factor in my life I never got soaked into wasting my hard money on protein bars, pre-workout or post workout recovery drinks because I would always make them at home. However just like in stocks I knew in order to make money you need to spend it. I wasted 2 years of my life attempting to grow this "business" and got no where and there's many reasons why. I didn't have a written budget for the month, I stopped investing, had over 20k in medical debt, I spent my emergency fund trying to build my "business", grow a romantic relationship, all while I had no license, no car and was living in my parents home. Later I had got fired from my job at a call center (and rightly so as my mental health wasn't in a good state). I was trying to grow my finances with no solid foundation or mentorship which caused me to go down a rapid spiral. My relationship fell, I had to quit and abandon that business and start from scratch. I was an emotional wreck. At this point I'm 28, no job, no car, no license, in massive debt, living with my parents in which it was not a healthy home. I applied at every job I could find and had multiple interviews but no one hired me and that hurt. That hurt me bad. I would cry every night asking God why did he do this to me? What was the reasoning? Thankfully UPS hired me and for that I am and will forever be grateful. I began to go back to what initially worked was trading in the stock market and began following various strategies I learned following people such as Timothy Sykes, Ross Cameron, Ricky Gutierrez and investing advise I learned from Dave Ramsey and the book The Intelligent Investor by Benjamin Graham (whom was Warren Buffet's mentor).

You see market crashes are a blessing is disguise and with budgeting, stock trading, living frugally, forcing myself to run and bike 25 miles a day to and from work for months on end during the 2020 COVID-19 pandemic. I was able to become debt free beginning of 2021 while starting the new year with my emergency fund saved and stored in an interest earning savings account. While it was a huge relief I still at this point was only working part time at UPS and part time trading stocks. As making only an average of $200 a week at UPS I knew in order for me to be stable I have no choice but to invest as much as I possible can to leverage my way to financial freedom where I can become my own boss.

Chapter 4: The Disguise Of Insecurity

I've tried a variety of different brokers and quote a few I've used offer free stocks. Remember when I said that problems are often hidden as opportunities? I began this battle in my head of seeing the need for making a page on my website dedicated to brokers that offer a free stock but then the thoughts of taxes come up, pros and cons of each broker, explaining what a broker is, a stock is, etc: just sounded too complex and strays from the focus for such a simple book on what I learned about making apps.

You see ideas are so easy but making that idea a reality is a very different challenge. You see I wasted too much time just planning ideas of apps to make instead of actually executing on them. Don't get me wrong it's important to sketch out the user interface, write down core functions and goals but if you spend too much time planning for the "perfect" app you will never actually execute on making that app a reality. I had learned the hard way not just with myself but even conversing with other developers that we make up perfection and that perfection itself is the disguise of insecurity. So don't stay focused on your ideas too long or you will never execute.

Chapter 5: Stop Trying and Start Training

I've always been a designer at heart. Everything from photo manipulations, vector art, 3d modeling, animations, etc: It's very therapeutic for me. As I've progressed as an artist over the years I learned many different styles and learned my own design style. This didn't come by trying but came by training. When you try you're only putting half the effort (I'd even argue less than half) instead of really training. Training is when you are spending a good amount of not just time but a good or even extreme amount of effort everyday on the craft that you're determined to become good at, great at or even excel at.

In this technological day in age it's unbelievably easy to get distracted. Everyday need to remained disciplined and remain undistracted amongst distractions. These distractions oftentimes for me did more harm than good on me progressing in design because I was simply wasting time. For me there's a difference in wasting time and taking time away for my own mental health as it's all about balance. For me sometimes stress is good but there can also be a point where it's unhealthy. The threshold will vary from peer to peer and from mood to mood. It's still important to remember to take a step back now and then and remember not to judge ourselves for not being productive while we're being emotional. I wish I had learned that lesson earlier in my 20's rather than later but regardless I learned it and am grateful I did so that I can share it with you.

Chapter 6: Don't Sell Assets Prematurely

If I could go back in time knowing the mistakes of my past I wouldn't of sold my MacBook prematurely in 2020. At that time I seriously believed it accelerated me getting out of debt but in reality looking back it only slowed me down because I had to adjust to trading stocks on my phone which made my execution times significantly slower meaning I had lost a lot of potential gains from 1 mistake of living a minimalist lifestyle to the extreme.

If I learned anything from coding and making apps on nothing but my phone it's that I really missed my laptop because I could've easily cut design and development time significantly. It all boils down to using the right tool for the job.

Chapter 7: Learn What Works For You

I've used many different operating systems over the years. I've tried too many Linux distributions that I can't even remember all the names for. I remember justifying the reasoning because of the development of WebDGap. I honestly believe now that such technologies exist to make your apps cross platform the reasoning to get a MacBook and a windows laptop or just dual booting isn't really necessary anymore.

For me I found I my productivity speeds always depended on the tools I used. For example I found I was best at video editing with Camtasia Studio but if I wanted to add special effects. I would use Blender because I didn't wanna spend the money on any other software that could've potentially of decreased render times.

Of course this isn't always relevant towards software but hardware as well. For example I remember wanting to use the DaVinci Resolve video editor really bad but it just wouldn't run on my Mac as it just didn't support the old outdated hardware I was using. Of course there's many things I could've done but I don't dwell upon the "could a, would a, should a" spiral. I apply lessons learned after an event like a trade loss occurs. For example say I'm trading the stock market while making French toast for breakfast (which really happened). I've already setup my watchlist for the day and did my technical analysis for the stocks I plan on day trading. This particular day on 2/1/2021 I was day trading $AUMN. While leaving my phone on the charger and delivering the French toast to the table. I missed my alert for a buy in opportunity at $1.10 a share. Instead I got in upon the upward momentum and bought 736 shares at 9:46:16AM at $1.23 a share with only $1k in my account. So I'm investing $905.28. I end up selling all my shares at $1.30 a share making a 5.63% profit or a profit of $51.52. This particular day my lesson learned was to never step away from my phone laptop or tablet while trading as while this case I gained profit I could've actually lost just as much if not more. Which I added to my risk management strategy. So again don't dwell upon "could a, would a, should a". It's easy to look back and think these things but at the end of the day we learn to adjust to changing conditions. It's as simple as what most of us did as children when our mommy and daddy said not to touch the stove and yet we did cause it looked cool and fun.

Chapter 8: In Closing

I sincerely hope you've learned from my mistakes of a decade of making apps and maybe even learned something new about yourself either as a person of when to take a break when needed, to not judge yourself while being emotional or even embracing failure by understanding failing isn't always failure. Maybe you learned of new tools you never heard of before and want to try. Whatever it is I really hope I've brought you some good value out of reading my ebook that you would be able to apply to your own life.

With Love and Grace By Yours Truly,

❤ Michael Schwartz ❤