DESIGN'S IRON FIST

and other essays

By Jarrod Drysdale

Part 1: Business

Part 2: Design

Foreword

Thank you for reading this ebook. It's a collection of my most popular and well-received articles. I sincerely hope you learn something useful.

You'll find more writing like this is you stay subscribed to my newsletter.

I'm so excited to be able to share this with you for free. If you enjoy it, I'd really appreciate your support. You can learn about my products, or hire me for consulting work.

Best wishes for you and your projects,
Jarrod

@studiofellow

Part 1:

Business

Reboot. Relaunch. Redesign.
Pivot. Sunset. Shutter.

The Knack, a web app, story.

Editor's note: This article started it all. It was my first public post about design and business, and laid the groundwork for what eventually became my first ebook and the very newsletter you are subscribed to. This article hit the front page of Hacker News, and brought me 3 full time job offers, and connections with many smart people who have taught me a lot. Some have claimed this article sounds bitter, and in hindsight, that's probably somewhat true. However I still believe the experience is worth sharing. I sincerely hope you don't make the same mistakes I did, and that you never have to shut down a business you've poured your heart into.

This article is about the demise of my labor of love for over a year, Knack.

(Boring back story: I quit a full time job I loved when a freelance opportunity fell into my lap. It was my chance to strike out on my own. Building a web app on the side had been a goal ever since I read Getting Real, and this freelance project would fund it. I decided to design and build the thing myself after having friends and co-founders bail on me at various stages of

4 prior projects. The first version of Knack took me ~2 months to build, spread over 5 months of freelancing. Launch was August 2010. I worked on it another 4 months of the following year, and today I am shutting it down.)

Entrepreneurship, and the act of creation vs. the pit of despair!

My first year of entrepreneurship was spectacular and melodramatic. It was unlike anything else I have done in my life or my career. I made something awesome. Not everyone can say that.

Throughout the year after launch, I struggled to find users and make sales. I tried every marketing tactic I could afford. I wrote personal emails to bloggers asking for coverage, or at least feedback. I took a stance. (That teachers are unfairly blamed for the problems in education). I lucked out and had a chance to write about my stance for GOOD. My thoughts and writing were a bit undeveloped, but it was exciting. I got a few hundred visits. This was the high point of the year, and came a month or so after launch. It didn't last long; the traffic stalled. I struggled to get users. I added features, relaunched, redesigned, repositioned. I rewrote the marketing website

over and over. I ran Facebook, LinkedIn, StumpleUpon, and AdWords ads. I wrote link bait blog posts. I tweeted. I facebooked. Sent more emails. I offered free subscriptions to every teacher I met.

A few people loved what I had to say, but none of them used my web app. (Almost none.) Over the year I had about 120 free trial sign ups (100 of which I bought with AdWords), and a total of exactly 10 users who paid for at least one month. I lost about $2000 out of pocket, which I easily financed through freelance work. Monthly costs were about $150 total for my Braintree merchant account/gateway, Spreedly, and Heroku. The price in time was more substantial. Knack occupied about 6 months of work stretched out over more than a year. At my standard $100/hr rate, that's $96,000 worth of work uncompensated. This figure is misleading and even deceptive, so let's take it for what it is: a number I pulled out of thin air. (I don't even make that much in a full year, much less 6 months.) That said, it does show how much effort I put into this app.

My great idea.

Knack was supposed be an alternative in an enterprise software dominated market. It would underdo the competition. It would save users measurable time every day.

Knack would empower downtrodden educators who are blamed for the problems in the public education system.

My goal was to carve off a niche of young, tech-savvy educators who are passionate about education reform. I'd build an app for them. I wasn't going to revolutionize an industry. I didn't need every teacher everywhere to use it or even like it. Just a few.

I focused on building a great app. I reworked UI features until they were right—sometimes 4 or 5 times. I ignored the laundry list of standard features most gradebooks have, and just built what I knew teachers needed. The only blogger who graciously covered Knack called it a "beta version." That stung, but it was fair. I had launched early on purpose, following the advice of tech industry maxims. I knew I could always fix it up later (and eventually I did).

I charged money for my software. I could not risk freemium and getting stuck with server bills I couldn't afford for people who didn't want to pay me. I set the price low—only $4.99/mo. That's cheap for a web app.

The same month I launched, Learnboost, a funded silicon valley darling, launched too. I felt blindsided. Worried. Then I reread Getting Real and Rework. Learnboost had just validated my market, or so I thought. I put it out of my mind and kept working.

(More recently another funded app probably borrowed from my design. Quite a compliment. This event and my

ensuing desperate tweets led me to Amy Hoy's 30×500.)

I thought Knack was a viable business. It was the best work of my career, built for people who could reap tangible benefits. The hard truth that it was doomed from the start took that full year to set in.

Sometimes people don't want their problems solved.

Before I built the app, I talked to friends and family who are educators. Trying to be supportive, they innocently told me what I wanted to hear. They didn't know much about business or software. I didn't know how to get real answers out of them. (Now that I've read Running Lean, I do.)

I should have done more research about teachers before I started coding. I thought what I'd learned was enough. There were quite a few online gradebooks that charged money and seemed to be healthy businesses. I had found research that said most teachers spend their own money on their classrooms. I asked educators I knew if they'd pay for a web app. None of this was enough. I didn't learn about the simple psychology that drives teachers' desires and purchasing decisions. I didn't realize there is a stigma amongst teachers—

that they deeply resent having to spend money on their classrooms and careers. Many businesses, both online and off, have special discounts and freebies for teachers. This has warped teachers' sense of value and fostered a sense of entitlement. (Whether teachers deserve free stuff is a different debate. I personally don't think anyone deserves a free lunch.)

Beyond this, I've learned that teachers do not want technology solutions for their everyday problems. Teachers say they love tech. Some blog about it. They tweet about it in #edchat and #edtech. They even coin their own special tech terms. (PLN anyone?) This is a farce. Talking about tech and being on the Twitter make teachers look good to administrators and to the public. They can add "Technology Committee Member" to their resumes and congratulate themselves for being innovative. But using tech to do work requires a small minimum of effort and change, and any amount of these is too much for teachers.

To be fair, teachers are really busy. A hell of a lot busier than you'd expect. Really. Visited a local school lately? Most teachers are busy putting out fires or trying not to get fired. (Or lose their collective bargaining rights.) I feel for them. I really do. But they're still terrible customers.

We ruined the web.

Teachers and education are just one online market we ruined by giving everything away for free. I made the mistake of thinking I could carve off a little niche of teachers who were passionate about education reform. I couldn't. You can't either. Don't waste your time on people who won't pay. They aren't going to change their minds.

People are going to comment and tell me mobile is our savior. That I should have built an iPad app. Those people are wrong. If mobile proves anything, it's that technologists have a poor collective memory. We can build apps for every imagined need on every emergent platform. Just like we do on the web. We can create a boring app for every boring aspect of our boring lives, and then pat ourselves on the back just as each lands in the dead pool. We can even create a few good apps, but the majority of consumers will not assign monetary value to them. That's the tech industry we earned, and it won't change anytime soon.

(You could write me off as bitter. To be completely truthful, I'm not! I had a blast building Knack. I had to learn some lessons the hard way, but it was the best thing I've ever done.)

The way forward.

After a year, I ended up with around 10 Google Docs with variations of the titles "The Way Forward" or "Reboot." They are filled with ideas of how I could pivot, respin, relaunch, better market, redesign, or add to my app. None of these could have saved Knack.

The way forward is without my great ideas or my selfish audience. The way forward is finding people who are willing to pay for an honest day of work. These people deserve my attention, and I deserve their money if I do a good job. (I'm going to.)

If you're interested in what I come up with, follow me on twitter.

Funding or bootstrapping:

The too-simplistic paths to startup success.

After my 15 minutes in the spotlight, I was fortunate enough to speak with a handful of experienced and successful founders. I've condensed what I learned down to a few important points I haven't seen mentioned elsewhere. This knowledge is nothing new for experienced entrepreneurs, but those of us just getting started don't see this kind of advice often enough.

One important note: the definition of success is relative. For some, it means breaking even on monthly revenue. For others, it means being the next Facebook. I'm not going to proselytize about this here, so, as you read, feel free to insert your own definition for success.

It's business time.

Most tech startup founders don't know the first thing about business. I certainly don't. This is the biggest reason our startups fail. Not because we chose the wrong business

model, market, or executed poorly. Failure comes from not understanding the relationship with customers and the principles that guide every kind of business.

From reading tech blogs and listening to gurus, you'd gather there are only two strategies for success: charge money or pursue an acquisition. People with business degrees (or just more experience) know better. You can make money many other ways, such as from partnerships, charging other companies for access to your users, and transaction fees. Even a combination works. (Also, advertising and IPOs, but these have become more challenging in recent years.) The only limit to the possibilities is your creativity.

No matter how inventive or brilliant your strategy, more important is learning to gauge the likelihood of success for that strategy given market context. Not every strategy will work in every market. For example, consumers leap at the chance for special deals, which are in essence nothing more than ads. If you really think about that, it's shocking—most people complain about ads. Business is counter-intuitive, and often the most obvious approach is dead wrong.

Business fundamentals will determine your strategy. Research is critical. You need an understanding of customers and their desires, habits, beliefs, and needs. Before you build, you need to know how you will market. You need to plan how to present your product not only so that people will understand, but also in a way that gets people excited. Great

ideas fail us. Ideas are all about us, and not our customers. All great businesses focus on the customer.

One-sided views of business fall flat. As newbies, we binge on blogs, books, podcasts, and conference talks. We assume the advice we glean is correct; after all, the people giving it are successful. However, every business is unique. No two businesses operate or grow the same way. Just because a tactic worked for one successful dude, don't expect it to work for you. What we newbies need is not stories of how so-and-so earned her millions, or how some guru thinks you should do things his way. We need a mature and rational understanding of business.

This sounds obvious, but how many tech entrepreneurs to you see discussing business? We talk about features, code, design, and "monetization." We read up on marketing, optimization, and SEO. Occasionally we add a new popular business model to our cannon, but these are all surface issues. We're missing the most important part: a solid foundation built on centuries-old, proven business principles.

Success takes more than building software and selling it.

Many bootstrappers, like me, read Getting Real and think all we need to do is build something great and charge for it. We're missing the point; just charging for your software product won't bring you automatic success. Getting people to pay you is hard work.

Boostrapping appeals to me for its fundamental and obvious nature. Plus, if you charge customers, you have revenue. This makes sense. It's logical and understandable. Starting small gives you freedom to experiment and get it right.

However, charging money for your software still requires the right combination of target customer, software product, and market environment. As I learned with my first product, not everyone is willing to pay. Not every product is worth a monthly payment. Not every market welcomes a simple transactional structure. Before you write a single line of code, do some research. Talk to people. Learn to ask the right questions, and don't quit until you prove or disprove every assumption. Bail if you don't like what you learn.

(I'd like to add that I'm in no way trying to argue against the advice of the 37signals guys. I owe them a huge debt for the good advice and inspiration to strike out on my own. Their books are essential reading. My point is that entrepreneurs need more knowledge than a couple of books can provide.)

Funding is complicated.

Saying VC is evil or funding is somehow less altruistic sounds great when you're standing on a soapbox. I understand that people who say this are just trying to establish that it's not the only choice for tech startups. However most of this dialog also suggests it's one or the other; that business must be either purely bootstrapped or purely funded. The Lean approach is the first voice of dissent; here, funding is seen as a means to spur growth rather than start. This is constructive. The popularity of the Lean methodology is encouraging. The startup community is beginning to mature; we're recognizing the complex nature of business. It's a good start.

If you're a newbie, you need to understand a few things about funding:

1. To get funding, you need to prove the viability of your startup. This means users, numbers, and a sound strategy.

2. To get funding, you need connections. If VCs and angels don't know you or you don't get an introduction, they won't even listen to your pitch.

3. Funding is not going to make your startup successful. Success is in the fundamentals. You need to get that right first. Funding can be an option later on, but you probably don't need it when you're just starting out.

4. Incubators are glamorous and exciting. They give you connections and advice. However insiders will tell you the advice from various advisors often conflicts. Even in this concentrated environment, you face the same dilemma—which guru should you listen to? The benefits provided from incubators will not replace a whole understanding of business. After all, not every incubator startup succeeds.

For these reasons, funding is an insurmountable challenge for most newbies. Boostrapping is often more attractive as a first step. If you still want to swing from the fences from the onset and need funding to do so, go for it. I

commend your ambition. Just know what you're facing.

Grow up already.

I've realized my understanding of business was immature. Join me: let's put all this simple-minded, one-sided business advice behind us. Let's stop wasting our efforts on businesses that can't succeed and stop repeating each other's mistakes. I, for one, am exhausted of post-mortems.

$30,000 eBook Sales.

In 2 Months.

Editor's note: This is the most popular article I've ever written. I wrote it over 2 years ago, but I'm including it here because the lessons I learned are still relevant.

Just over 2 months after I launched my design ebook for startup founders, I broke $30,000 in sales volume. Here's what I learned.

Research customers.

6 months ago, the idea of writing a book was inconceivable. I've never wanted to write a book. I didn't think I had anything to say.

When I started—with research—what would become my next project, I was surprised. Not surprised at myself, that I had discovered some new ambition, but surprised at what people needed and how well I could meet that need. Me, a designer, not a writer.

See, by beginning my project with research rather than an idea, I found an opportunity. It wasn't an opportunity I could have imagined nor one for which I would have intentionally searched. It was an opportunity that already existed out

there—on the web, in tweets, on blogs, and appearing in the frustrations of certain people. (The audience I researched was programmers who are bootstrapping software businesses.)

The primary reason for the success of this eBook is that the idea came from my customers, not from me.

My business is succeeding because it began with an understanding of customers. This understanding includes not just what they need, but also more important insights: what they buy, what they value, how they communicate, and where they hang out.

You think you already know these things about your own customers, but you don't. Your assumptions are wrong.

My first business was based on such assumptions, and it crashed and burned in silence. Don't make the same mistake. Research first.

Price by value.

I set a price for my ebook that some consider too high. Their opinion demonstrates that these people are not really in the audience for my book.

My audience is composed of professionals—they're good at what they do and they are paid well for it. While many in my audience can't afford to hire a designer outright to work on their bootstrapped side projects, they are comfortable

paying for products and services as part of doing business. They donate time and money to open source projects and they enjoy supporting products they like. For them, it doesn't matter that much if the eBook costs $12 or $39. All that matters is that it helps them to build a more viable business.

Read my guest blog post on A Smart Bear about my value pricing strategy for a more detailed rationale.

The numbers prove my strategy worked well enough. I earned $30,286 in total sales volume (revenue) as of 66 days after the launch.

Say sorry.

The power of an apology or the cost of a mistake?

I made an honest mistake after launch. The coupon I had promised everyone on my mailing list expired before I said it would.

I reactivated the coupon, extended the expiration by an extra week, and sent an email to the list apologizing. This apology email drove about $1,200 in sales. Other newsletters have not converted quite so well.

Would I have earned more sales if I hadn't made that

mistake? Did the apology completely close the gap? I have no way of knowing. Regardless of speculation, the apology email made an impact.

Later on, my payment system had an issue where credit cards were being declined for no reason. A couple of customers were very kind to notify me, and I got in touch with the support teams for those third-party systems and they fixed the issue quickly.

Rather than leave it there, I sorted through the logs to find about 10 people who had been declined when trying to purchase. I wrote a personal email to each of them, and included a $5 discount coupon as an apology. A few of them had already purchased after the error was fixed, so I refunded them $5 instead.

These apologies have led to customer relationships and great feedback, and I know that several of these customers recommended my book to others just because of this experience. Apologizing and fixing the problem was not only the right thing to do, but it drove more sales by word of mouth.

So, what now?

The last couple of months have been strange because I have no idea how to run a profitable business. I'm making it up as I go.

I've completely skipped marketing practices like SEO and A/B testing. Many people claim these techniques are essential to running a business, but the truth is they are long-term strategies for optimizing something that already works. SEO and multivariate testing can't do much if people don't want your product. This time out, I learned to focus on getting the product right before I worried about following every best practice. I can always work on that stuff later.

Next, the revised edition of the eBook is in the works. I have dozens of lengthy feedback emails and even full copies of the ebook annotated by customers to inform the revisions.

After that, I'll start researching the next project. I have an awesome customer base to learn from, and they're already asking for more.

The Innovation Fad

If you read startup blogs or listen to entrepreneurship podcasts, the only consistent advice you will find is that the thing you build must be innovative. It must be a completely new and never-before-seen idea.

This requisite newness carries many implications for your business. The most damaging implication is that past models for success no longer apply. If you're building something truly new, you must cut a new path and find a different way to achieve success. You must employ av unique mash-up of tactics. Otherwise, what you've built is boring and lacks ambition. Why bother doing it the same way someone else did? You're a brilliant thinker. You can do better.

If you allow this mindset to seep into your brain, suddenly, nothing is good enough.

Any design you could create on your own isn't good enough. If you know PHP already, you shouldn't use that because it's not as cool as Node.js. No one would take a startup seriously if it uses PHP. Or, just building another project management app is laughable. There are so many project management apps—how could you possibly have something new to add?

The proposition that newness and innovation drive success is a farce. Furthermore, it's illogical. If you use an

approach that has earned success for others in the past, it is much more likely to earn success for you. Feeling your way around in the dark and hoping you'll stumble upon a new insight is a fool's errand.

So think about the most successful businesses and what they do. Are founders of recent tech startups the most successful entrepreneurs? Or, are old-school business types more successful? The answer is obvious.

Knowledge about business success has been available for a century. Tech celebs can't hold a candle to it.

As for design, programming, marketing, and practically every other component of your business, the same is true. Look to what works historically, not to what sounds innovative.

Get off my soapbox:

Don't let trolls hobble your marketing.

Some months ago, I decided to change how I write. I'd realized that much of my writing came across as preachy. "People just want helpful information," I theorized. "People don't want to be preached at."

So, I wrote posts geared towards being helpful and did my best to avoid sounding judgemental.

And something strange happened. These posts didn't get so much traffic. People started unfollowing me on Twitter. My writing frequency decreased.

I had a difficult time deciding what to write. I'd often open up Twitter, then immediately close it because, well, I didn't have anything to say.

My decision to avoid sounding preachy had removed the best aspect of my writing: the passion. And people lost interest.

When I changed back to my old, ranty style on Twitter, people immediately began engaging me. Even though I was

standing on a soapbox. Even though my bold claims had some slight inaccuracies, or didn't apply to everyone. Even though some of my statements were a bit idealistic and simplistic.

Looking back over my blog and tweets, the difference is amazing. Bold, controversial content always gets more interest. Some people get mad, but even more thanked me for speaking up. And then they subscribed to my newsletter, followed me on Twitter, or bought my ebook.

I'd changed my writing because of my fear of dissenters. I was tired of having people nitpick my arguments. I was afraid the only reason people were reading was because of the controversy—that they didn't really care what I had to say. I didn't want people to think I was just rocking the boat so they'd buy my stuff.

I was wrong. Developing a voice and taking a stance are what earn you an audience. It's how you get people to care about what you're doing.

> *"If no one's upset by what you're saying, you're probably not pushing hard enough. (And you're probably boring, too.)*
>
> *—Jason Fried and David Heinemeier Hansson in Rework*

So when you're writing landing pages, blog posts, and emails for your business, take a stance. Say something bold. Make some people mad. You'll find that people who share your worldview will gravitate toward you.

And do yourself a favor: spend your time talking to those people, the people you like, rather than arguing with grumpy strangers on the internet. You and your audience will be a lot happier.

Is your product a tourist trap?

Have you ever been to a public, museum-sized aquarium? I have been to a few, but never one with a real tiger and a robotic chimp like I saw at the Denver aquarium last Sunday.

Aquariums are a lot like bootstrapped products: to most people they sound extremely boring, but to a small group they're really exciting.

I get really excited about aquariums—maybe more than is appropriate for an adult—and while it was a strange experience, I had a blast.

The ticket booth sold tickets for kiddie rides outside the building, plus behind-the-scenes tours, yearly passes, local restaurant discount cards (?), and something called the "Aquarium Adventure Experience". I couldn't find the price just for tickets to see the aquarium.

At the window, I asked "We just want to see the aquarium, how much is that?" The attendant said, "Two Aquarium Adventures? $36," in a dry tone that seemed to imply I was the biggest idiot in the world. I bought my tickets, ready for adventure.

Just inside, someone shouts "Want a free photo?" as we push through the confused crowd blocking the door. You can

have your likeness superimposed over a menacing shark, etc. No thanks, iphoneslol. We walk past.

Then we get lost. Walk in circles looking for the entrance to the exhibits. Turns out it's right behind the person taking photos, with no signage communicating this whatsoever. Clearly, the management doesn't want you entering the place until you've recorded how much fun you are already having.

Once we found the big glass boxes with water and fish in them, it got to be more fun. But around each corner, a booth selling souvenirs begged for attention. Each had a crowd of understandably manic children swirling around it. Those kids would walk right past each exhibit, looking for the next souvenir booth, totally missing the actual fish and stuff. I couldn't blame them—I love snow cones bigger than my face too.

Later, we stumbled upon a fake orangutan robot puppet thing that made the most horrific screeching sounds as it held its creepy fake dead-looking baby, expelling a shockwave of dust with each twitchy movement.

Next was a tiger exhibit. Me: "Wait, there's a tiger here? I thought this was an aquarium." Rachel: "Yeah but everybody loves tigers."

The path dead-ends into a little plaza, offering photo ops with bikini-clad mermaid actresses. The photographer approaches me assuming I'd definitely be interested. (No thanks, I'm not a creep.) Heading toward the exit sign, we

somehow find ourselves back at the entrance. We finally spot the real exit obscured behind one of the souvenir booths. Then a final decision: exit through the gift shop or the second restaurant?

Tourist Trap Marketing

You and I have both tried to sell our products like the Denver aquarium does.

Falling into that sales and marketing approach is easy, because as consumers we see it constantly. But it's the wrong way for bootstrappers, authors, and founders to sell a product online.

"I'm ready to buy, but I can't find it"

At the Aquarium, I was sold on the idea of seeing fish. (Which if you think about it is an accomplishment. I could see the exact same stuff at a tropical fish store for free.) But I still couldn't figure out which ticket to purchase.

User experience is too often determined by "what I want to sell you" and not "what you wanted in the first place". When designing a sales site, it's important to consider how someone arrived. For example: was it through a tweet, a blog post, or a retargeting ad?

The Aquarium ticket booth offered products for every possible context. If I had visited previously, maybe I'd be interested in seeing something extra this time. But as a first-time-customer, I was completely overwhelmed.

Don't make the same mistake and write your pitch to accommodate every possible context. You should already know who your target customers are. Focus on your primary audience first. It's better to serve one small group well than to barely serve a larger group at all.

Over-branding your products

Often in the process of selling—explaining all your awesome features—the customer gets lost.

Sometimes we get clever in explaining how awesome our products are. To us, it sounds great because we're proud of what we've built. But this kind of overly clever writing can also cause the customer to be confused about what they'd receive upon purchase.

In a customer's mind, "aquarium adventure experience" != tickets to see an aquarium. The former sounds like it costs more and includes something extra.

Similarly, don't name your most basic tier "Pro" and your upgraded tier "Elite". It's confusing. If someone wants your ebook, what do Elite and Pro have to do with that? It seems

like there isn't an ebook, and you might have just lost that sale.

"How many more things do you want me to buy?"

Conceiving product upgrades and up-sells also sounds wise. Value-based, tiered pricing can increase your earnings.

To be clear: there's nothing wrong with asking people to pay for a product. Offering different tiers or packages is good for customers because different levels of need can be met.

However the example of the numerous aquarium souvenir booths illustrates what it looks like if you go overboard. Walking through this fascinating place, I could barely see past all the extra sales pitches to enjoy it. Even worse, seeing all that junk I doubted the value of the whole experience in the first place.

When you do offer packages or upgrades, make sure they're worth buying. Also, ensure the pitches for the different versions of your product don't compete, or all you'll have earned is frustrated, confused people leaving your website.

If you aren't sure how to add a tier that makes sense, don't tack on something half-assed. Just invest that time into the core product. Sure, we all want a perfect pricing strategy, but simply selling a quality product is a great position to have.

I've sold my ebook without any tiered pricing for a couple years (a new edition is coming soon too, by the way), and have reached over $70k in profit. You don't need tiers to start making money.

Who wins: fish, tiger, or chimp-bot?

Your product isn't selling as well as you'd like. You could try to fix it like the aquarium did: add a tiger, or worse, an unintentionally nightmarish chimp-bot.

Even if you build your product the right way, you can still experience slower sales than hoped. This happened with my product, a design framework for coder-founders called Cascade. I launched Cascade to a newsletter with 4500 subscribers, 1800 of which had already bought my ebook.

When initial sales were slow I was tempted to build a new feature; to add a tiger, something that made the product seem really cool to a new set of potential customers.

But making a substantial change to the product would deviate from the research about my audience and their needs. I'd end up with a confusing product: a robochimp in an aquarium.

Eventually I realized that I simply needed to do a better job of selling. My current page is the 6th or 7th complete

redesign & rewrite (I lost count). A trickle of sales has started back up. Even better, customers love my aquarium-esque product. I'm so glad I didn't end up with chimp-bot instead.

What does the purchasing experience really look like to customers?

The analogies from the aquarium may seem obvious to you. None of us wants to seem a scammy tourist trap or greedy in the least.

However, as the people building the products, it's tough to separate ourselves and get sufficient distance to accurately see what it's like for customers.

Our community is learning that people-focused products, instead of sales-focused products, are better for everyone. **But even after you've found a great audience and built a product for them, don't lose sight of the people when you start selling.**

Part 2:

Design

Design's iron fist.

Finding that perfect type setting for a logo, or just the right amount of contrast to draw the eye to the signup button is satisfying. The right combination will have a huge impact upon how people read and explore the composition.

However, sometimes while working on a design, you discover a dilemma. You realize you might need to sacrifice the integrity of the message to make the design look pretty.

Maybe the headline doesn't fit in the font size you want to use. Or there are too many form fields to fit in the area you've reserved for them. In these situations, it's easy to cut the content to save the design. I couldn't count the times I've trimmed a few words or form fields so I could keep the design the way I wanted it.

But when you and I do this, **we're doing it wrong.**

Why is changing the content to fit the design wrong? It seems harmless enough.

You probably expected me to say this: design should serve the content. I've harped on this point in my ebook and in other writing. And I know you understand that point: all design decisions should support the content and goals of the project. But the implications are big.

"Design serves content" is a cute aphorism. It sounds nice and even intelligent enough to be correct. But I'm here to

tell you that this claim is not a harmless one. It will cause you much strife while working on a design.

Are you willing to murder your darlings in order to support the content? Are you willing to change your favorite aspect of a design in order to deliver the message with more clarity?

Design's iron fist rules the content, but it also rules you.

Be willing to sacrifice the aspects of your vision that impede project goals. Don't cut form fields that are essential to your business just so it all fits in a certain spot. Don't remove words that form a better connection with readers just so you can use that particular font setting.

Instead, rework the design. Change the layout. Make more space by shrinking less important elements. Go back and sketch until you find a solution that fits the content and goals, but that will also look pretty. You can have it all if you're willing to put in the effort.

Don't get me wrong. It's not easy. But the affect on your conversion rate (or other key metric) will make the effort worthwhile.

The choice between design and content integrity is a false dilemma. You can have both.

You are a designer
(Yes, you!)

Are you building software or launching a startup? Are you a coder, thinking you'll never feel confident in your own designs?

Start calling yourself a designer right now.

"Designer" isn't a title you unlock at some threshold of skill, or a title you earn. People of all skill levels are designers.

If you begin calling yourself a designer, all the sudden you can take that work seriously. You might not have a lot of experience, but at least the time you spend working on a design is serious work. You're not messing around, dipping your toes in the water. You are committed.

That commitment makes all the difference: now you are allowed to make mistakes, because you are going to stick with it. Making mistakes is how you learn and improve your skill.

Too often, fear of making mistakes gets in the way of learning. Make a commitment and allow yourself to make mistakes without beating yourself up about how bad it looks. With each attempt, your skill will improve.

Those like me who have made a career out of design started poorly skilled. My work wasn't great when I began my career, but I still called myself a designer. Back then, I wasn't any different from a developer who is just starting to learn

design right now.

It's easy to earn the right to call yourself a designer: just go design things.

> *We assign mystical reverence to the work of professional designers. Their elegant color schemes, provocative typography, and eye-scorching aesthetics leave us dumbfounded. Only "creative types" can achieve this; only near-savants who were born with a special talent.*
>
> *…you'd think design is difficult. You'd think it's complicated, and that gaining basic skill requires hours of studying a multitude of advanced topics. And you'd be wrong.*
>
> *Anyone can be a great designer with practice. It's both at once liberating and frightening: your future as a designer depends only on how hard you're willing to work. Design is a skill and a trade; you get better at it by practicing. First, learn the basics and go design something. Then, call yourself a designer. The more things you design, the better you will get and the more lovely and insightful your creations will become. No magic knowledge hidden away in design books, blogs, or classes will teach you to be a great designer. All you have to do is practice. Learning design is that simple.*

—Excerpt from Chapter 2 of my ebook, Bootstrapping Design

How to
Start a design

New designers are terrified of that blank screen. How are you supposed to fill it with design? Where should you start?

Somehow, that empty white space instills fear. Why? Because it holds power over us. Because we have no reliable method for starting a design.

The cure to this fear is process.

Now, you hear designers talking about process, and you've seen those pretty diagrams that explain their processes. Every single one is different. How can that be? Even worse, those diagrams are incredibly complex. How can a person internalize so many steps and be able to follow them instinctively?

I've got good news for you: you're being deceived. No serious designer works by following dozens of linear steps.

A correct design process is simple and it doesn't try to replace critical thinking. Design process isn't meant to think for you; it's meant to help you think clearly and decide what deserves your attention.

Here's the process you should use:

1. Gather

Find all the elements you need to begin the design, including content, requirements, goals, and examples.

2. Edit & Sketch

Revise your content and begin rough sketches that help visualize how you can organize the content and meet goals. Your sketches should be fast and ugly.

3. Mock Up

In Photoshop or HTML/CSS, bring your sketch to life. The mock up should reveal how the sketch functions and whether each feature fulfills its purpose.

4. Build

Build the design into a fully functioning application. Be faithful to the solution you planned.

5. Test & Correct

Use data and metrics to evaluate the design. When you uncover problems, and you will very often, return to previous steps to explore smaller solutions to those problems.

Now when looking at the blank screen, you know you can't proceed because there's so much to do before you even need a computer. This process makes that blank screen less intimidating, almost like you're not looking at a blank screen at all anymore, but a screen filled with kitten photos. #catpicslol.

Are you stuck?

The previous article explained how process makes the blank screen you face at the start of every project less intimidating. But sometimes, even when you follow process, you still get stuck.

When you get stuck, that usually means you are on the wrong step. Move back to the previous step and evaluate your choices.

Here are some examples:

You are struggling to sketch an interface.

This usually means you don't have enough information about how it should function, what it needs to accomplish, or how it fits into the rest of the application. Go back and Gather.

When mocking up a feature, you keep thinking of extra features to add to the page and get distracted thinking about them.

You didn't do enough sketching. Go back and sketch out all those little features, and make sure they really work well enough to deserve a mock up.

When building a page, you realize the design cannot be supported by the server-side code without substantial, deadline-breaking changes.

When this happens, it means the gathering, sketches, and/or mock ups were insufficient. Before you continue building, research that aspect of the software and then sketch and mock up a few different solutions and see if you can find one that doesn't cause huge implications for the server-side code.

You're done building and are ready to test, but you don't know what you should test.

In the Gather phase, you didn't think about the goals for the project. Now, you're not sure how to measure whether the project is meeting those goals.

So, to recap, process is how you get out of ruts. If you're not sure what do to, think about process. Take a step back and try to identify that missing piece that's holding you up. Sometimes your instincts will just make the problem worse—that's when process has your back.

How To Stop Design Guesswork

When you try to design, do you ever feel like you are making things up?

For example, you are trying UI elements you've never used before. You're not even sure if they will make a difference—it's just a guess. You thought it sounded like a good idea. But how do you really know if it's a good idea or a waste of time?

Design is just like any other kind of work: every time you venture out into new territory, you take a risk of getting stuck. When facing an issue, the natural reaction is to keep working until you fix it. Somehow a week passes, and you realize there was a simpler solution all along. Then you find yourself wishing you could get that week back.

While you're still new to design, getting lost like this is easy. Here are some tips to avoid that trap.

My definition of designer is broad. I write for developers learning design, and encourage you to start calling yourself a designer right now! Keep that in mind as you read.

Become a dirty thief

The design problem that's vexing you isn't new. Someone has solved it before, and they are probably more experienced than you. So before you try to reinvent the wheel, do some looking around to see how other designers have solved your problem.

Look at software you use regularly for similar situations. Sign up for a few free trials to see what the interfaces look like.

Once you figure out how someone else solved that sticky issue, copy them. Steal their solution.

This is what designers often refer to as "finding inspiration", "design convention", and "best practice". These are all fancy euphemisms for copying.

So while I call it stealing, what I'm really suggesting is that you learn from others' experience. That's the best way to solve the most challenging design problems.

(To be clear, I'm not suggesting you infringe on someone else's rights, break the law, or take intellectual property. Don't copy someone's entire design, just the one aspect that solves your challenge.)

Further Reading: check out my ebook. I wrote a whole chapter called "How To Steal," that gets into more specific ways to copy design ideas.

Sketch first and you won't waste so much time

Sketching is magic. There's no better way to find and test a multitude of ideas than to sketch.

Sure, you could draw a wireframe in Balsamiq. You could build a prototype in code or even create a full-blown mockup in Photoshop. But none of these is nearly so fast as sketching.

Sketches get exponentially more efficient the more concepts you explore. Your first concept will rarely be correct. To find the best solution, you need to explore lots of concepts. If you try each of these by making a prototype or mockup, you'll end up burning a lot of time.

New designers often want to skip sketching. They want to jump into the exciting part—the real design. They want to get something built. But sketches are important because they prevent you from committing to a solution too soon. If you jump into creating a polished, detailed version of the concept before you are certain it's correct, that's a risk. Better to spend 15 minutes sketching to ensure that time won't be wasted by having to start over.

You might think sketching is a bad idea unless you have some drawing skill. I also wrote in my ebook that many

designers' sketches look like works of art, but yours shouldn't. Your sketches should be quick and ugly. The only purpose of these sketches is to explore ideas quickly. The more beautiful your sketches, the more time you wasted. Let them be ugly. You don't have to show them to anyone else.

Oh, and just use a pen and paper. Don't try to get fancy with iPad apps or other software. Remember, it's all about cutting distractions to explore ideas as quickly as possible.

Don't start from scratch

Frameworks and libraries are awesome. Never before has our time spent coding been so efficient, and it gets better all the time. However you might not expect that using a front end coding framework can save you a lot of design time too.

Frameworks like Bootstrap and Foundation solve common interface problems for you. You don't have to research the best way to design a dropdown menu, or ways to lay out a form. Just choose from implementations built into the framework. Each time you do this, your design burden gets lighter.

Unfortunately, there aren't many tools like that specifically for design. *(*Cough* Sure would be nice if someone would build a design framework.)*

But joking and shameless plugs aside, even if you don't

use my framework, using traditional design tools isn't quite so difficult as you'd expect (or have been led to believe).

Basic Photoshop and Illustrator can be a good use of time when you are ready to build the real concept. Not necessarily for mocking up the whole design, but just for adding some nice graphics. Learning just a few simple techniques can make you look like a bona fide digital artist.

Yes, a lot of designers advise steering clear of graphics software altogether and designing in the browser instead. That's good advice (usually), but often sites still need graphics. Learn a bit of 'shop and you won't have to use stock photos so frequently.

If you want to learn some Photoshop, set a limit. A few simple techniques will be sufficient. If you tried to learn every feature in Photoshop, you'd end up with a lot of useless knowledge.

(Also remember you don't need an expensive license to use Adobe software anymore. You can pay to use it by the month these days.)

Nathan Barry is teaching Photoshop, and the topics covered are nearly identical to what I'd recommend (samples here and here).

If you're hesitant about creating the entire design yourself from scratch, I designed a framework that includes logos, color schemes, font pairings, and layouts to get you started. It's called Cascade. You can see a full demo too.

When you make an assumption, test it

Inevitably there will times you can't find a perfect solution, usually because you don't have enough information to inform your decision. And because of that, you will have to make something up.

Too many designers stop there. They launch that assumption and never think about it again.

As designers, it's easy to come to think that our assumptions are correct. We get comfortable making assumptions because design, as scientific as it can be, isn't always precise.

However, when your assumption can have significant implications, make sure to test it. Real data is always better than a guess. Implement an A/B test or simple analytics to see how people use it out in the wild.

Then, launch it and observe what happens. But keep your sketches handy. You might find that you need to try one of your other concepts.

Personally, I find CrazyEgg to be the best analytics for evaluating design. The scrollmap is especially useful for seeing how people interact with your content and interface. It's also less time-consuming to implement than event-based analytics.

Remember: if you are guessing, you need information

Do everything you can do avoid making a pure guess. Steal, sketch, and use tools that provide common answers. If you do have to make a guess, test it, and be ready to switch to a backup solution.

Put the logo below the fold:

Breaking design rules for profit.

Editor's note: I wrote this article after the first version of the Cascade landing page. I've since moved to a different design that does have the logo above the fold. Just keep in mind that this article is not simply about logo location. It's about considering the implications of every design decision.

I've had to relearn certain aspects of design now that I'm running my own business. I'm making decisions I'd have considered a major error only a short time ago. Like putting the logo below the fold. Or using a color that doesn't match the rest of the palette on purpose. On purpose!

The proof it works is in the numbers. Cascade is earning a 12-15% conversion rate, depending on the traffic source. This is higher than Bootstrapping Design's landing page conversion rate of 8-12%.

Designing for your own business is different than designing for clients. What follows are my hard realizations after living on product income for a year.

(Also note, I'm discussing professional products only.

Consumer products are a wild goose chase, as far as I'm concerned.)

The logo below the fold:

Customers are more important than me.

Designers revere conventions. Putting the logo in the top left corner of the site is supposed to be an important convention.

But for small startups like mine, it's the wrong decision.

Putting the logo in the top left corner is wrong because it makes the statement that the person or company operating the website is the most important information on the page. It implies: "Hey visitor, you should know who I am before you read anything. I'm kind of a big deal."

Brand awareness doesn't exist if your customers number in the hundreds. Designing for a small business is totally different than designing for a large brand (or a brand that wants to become large).

Visitors don't care about your logo and they don't even care who you are (yet). They are looking for the answers to two questions: What problem does it solve? Do I have that problem?

You'll notice that in my design for Cascade, I placed the logo below the fold and low on the visual hierarchy. The answers to those two important questions are right at the top where the logo would normally be. The only people who even see the logo are those who identify with the problem and intended audience. People who don't leave the page immediately. That's great because throughout the rest of the page, I know exactly who I am writing to.

So stop putting the logo at the top of the visual hierarchy. Instead, write about the problem your business is proposing to solve. Write about what type of person usually has that problem. Then, lower down on the page, explain how your business is the answer and ask them to buy. And place the logo there too.

Your job as a designer and writer is to show the visitor that they have a problem and to convince them to continue reading about your solution. If you succeed in this, people will pay you.

The ugly button:

Business goals are more important than good taste.

(I bet you never thought you'd see a designer write that!)

Cascade's landing page sports a shiny green button. The button isn't ugly in and of itself. But set amongst the intense orange that dominates the composition, it clashes—badly. And, it clashes on purpose.

I chose the bright, clashing green color for the button because I wanted visitors to see it. Originally, the color scheme for the site included a complimentary blue. I used that blue in several of the illustrations, but the buttons were blue too. When evaluating the design, I realized that this was a problem. The buttons had similar visual prominence to the other elements on the page, and that didn't fit my goals for the visual hierarchy.

My plan was to place the buttons lower on the page. After all, no one is going to sign up before I've explained why they'd want to. But, when the visitor does scroll to a point where the button is visible, I wanted it to be the most obvious element on screen.

A pretty blue button that matches the rest of the design did not accomplish this. So, I stripped the blue from the design and chose a clashing green color for the buttons. Now when you scroll far enough to see a button, it is impossible to miss. The ugly aesthetic of the button supports my goals perfectly.

With this change, I made a conscious decision: my business goals were more important than making the design attractive. I don't care if Cascade wins design awards. It doesn't matter if other designers like the page. It's not for them. All that matters is that the page connects with technical startup founders who struggle with design.

Furthermore, rather than merely assuming I met my goals, I used analytics. Every metric shows that my design is successful.

It's okay to break the rules.

I'm not trying to convince you to always place the logo below the fold or to always use an ugly button. But I do hope these examples give you the confidence to break best practice when you discover it's necessary.

Doing this is not nearly as big of a risk as you'd think. Designers might point out the flaws in rude tweets, but they could surprise you. Some might even be smart enough to

catch onto what you're doing and write a blog post about it, like Sacha Grief's writeup of the Cascade landing page.

Regardless, be proud that you're prioritizing efficiency over vanity. Making these kinds of decisions is the right way to run a business, and your customers will be grateful. And they'll show that gratitude by paying you.

Why do coders think they can't design?

"I'm not creative enough." / "I wasn't born with the artistic gene."

"I don't know how to start. Everything I read about design confuses me more."

"I recognize good design when I see it, but everything I design looks bad."

The answers are always the same. I write about design specifically for developers, and have even written a book on the topic. I talk to developers about design a lot.

Developers want to design—that's apparent from conversations. However, many are confused about how to start learning. And even worse, some developers don't think they are capable in the first place.

But that's not even remotely true. Design is a skill anyone can learn. You don't have to be born with a special talent,

or acquire a wealth of knowledge before you can design something beautiful.

Developers *are* creative

"I'm not creative enough." / "I wasn't born with the artistic gene."

You might consider coding to be logical and analytical, while design is creative and artistic. That's completely inaccurate. Coding is every bit as creative as visual design.

I've seen code that was art. I know you have too. The kind of code that just seems so brilliantly creative—a solution to a problem that was too elegant to believe or a new technique that completely changed the way people work.

Programmers are creative. When you are debugging code or writing tests, you are imagining possible outcomes and causes. That's creativity—the ability to imagine.

Which means you already have the creativity you need to design.

The only problem is you haven't learned to be creative in the ways necessary for design. You are creative in problem solving, but maybe not in visual communication. But that's ok. For now, what's important is that you know you are capable of creativity. You can learn to express your ideas visually, just like

you learned to express them in code.

Stop reading design education that's intended for experienced designers

"I don't know how to start. Everything I read about design confuses me more."

You've read design books, blogs, and/or tutorials, but you don't understand how to use that information. None of it seems to help you make a design, or demonstrate what your very first step should be.

Learning to code, you install libraries and a text editor, then start typing. The tutorial teaches you what each line of code means, and introduces concepts one at a time.

If learning design seems more difficult, it's because the tutorials/books you are reading are intended for experienced designers, not people just starting to learn. (And often, even if a resource claims to be written for developers, it still covers advanced topics too soon.)

Would you try to teach someone learning to code about OOP or TDD before they understood what a variable or a function is? Of course not! It would be impossible.

So don't make that mistake while learning design. Don't read about advanced topics like responsive design, color theory, or typographic genres before you have grasped the basics.

When you try to learn design, if you find yourself getting lost, take a step back and try to find a source for more basic topics. (I'd suggest that the most important topics to start learning are alignment, proximity, and visual hierarchy. My book could be a starting point for you.)

Your designs will look bad at first, but keep practicing

"I recognize good design when I see it, but everything I design looks bad."

Look back at a sample of your code from just a year ago. I'm sure you could find ways to improve it. In a year's time, you've practiced programming and you've increased your skill. You also have new knowledge.

Was the first program you ever wrote from scratch incredibly elegant, or was it buggy and gnarly?

Starting out with design is exactly the same. Your early

designs will be ugly and gnarly. But each design will be better than the one before it.

Design is a skill. You have to design things to get better at designing things. It's exactly the same as learning to code.

You have to be willing to let yourself make mistakes. Know that the first few times you try to design something, it's going to be flawed. That's ok. Look closely at those flaws and learn how to fix them. You'll get better at it.

Practicing design is not as much work as you're thinking

Practicing design sounds like a lot of work. After all, if it's just practice, you might think you're not actually producing real work.

Practicing on fake projects is great, but if that feels like a waste of time, practice on a real project. Redesign your personal blog, or finally design a concept for that side project you keep putting off. Or, take some minor feature of a bigger project and redesign it. Just do something.

You don't have to make 100 websites before you get good at design. (Who has the time for that?) It happens in small

steps. However, if you don't start taking steps, you'll never get better.

5 tips to pick:

Web Fonts

Choosing fonts means sorting through a massive collection. It's overwhelming if you're not sure what you're searching for.

1) Pick fonts based on intended use

You should always evaluate fonts for two kinds of uses:

Display font settings use large font sizes, for content like headlines and logos. You can use more decorative or complex-looking fonts for display settings because the font size is larger. Headlines are supposed to grab attention, so using a more decorative font is a good idea.

Text font settings are what you'd expect: paragraphs and longer sections of content. Here, you want to avoid decorative fonts because, at smaller sizes, they are more difficult to read.

Searching through a huge library is much easier if you know whether you need a **Display** or **Text** font. But, even separating uses like this doesn't keep you from having to look at each font, one at a time.

2) Ignore fonts that don't have enough styles

To filter out even more picks, look at each font family's available weights and styles. If you are looking for a font for a text setting, make sure it includes at least a normal weight, bold, italic, and bold italic.

If a text font doesn't include these at a minimum, you can run into problems when using web fonts. Sometimes, when a web browser doesn't have a font to use for a passage of bold italic text, it will use a default font, which can make your design look broken. Even worse, I've even seen browsers replace those passages with garbled text—as if that section of text was mapped to the wrong characters.

However, with display settings, you can be more lax about which fonts to choose. It's okay if you set a headline in a font that doesn't include a normal weight or italic, for example. It's a headline, so all you need is that one bold weight.

3) Check rendering across browsers

Before you decide on a web font, make sure you test it in all the browsers you plan to support. Many problems can arise when the web fonts are created or converted by the type designer. Some fonts might have bad hinting or kerning that only appears in certain browsers. These issues can even make large display text unreadable.

Don't assume that every font out there is equal. Fonts are software and can suffer from the same issues as any other kind of software.

4) Look up the type designer or foundry

Many free fonts, like on the amazing Google Fonts service, are designed by students or inexperienced designers. There's a lot of junk, but if you're willing to look, you can find some hidden gems.

Sometimes I look up the designer or foundry and see what other fonts they've made. If they've made several fonts or released under one of the more well-known type foundries, chances are the font files will be well made and will render

correctly.

(Note that a lot of the well-known foundries use Google Fonts for marketing purposes, so you'll find a lot of single-style font families from these.)

5) Notice how others use fonts, and imitate them

Typography genres can be confusing. Sometimes, genres are based upon history, but others refer to appearance. Plus, many new font designs are genre benders, which makes classifications nearly useless.

Picking fonts without knowing a lot about typography can be tricky.

To make this easier, take a look at site designs created by experienced designers. Note how they use fonts in various places, then find similar fonts to use in a similar way in your own design.

This is a lot more foolproof than trying to invent it all yourself.

It's good to doubt

Doubt. With creative pursuits, it's inevitable.

Not just with design, but anything creative—starting a business, writing a blog, or even choosing a color to paint your bedroom walls.

I just wanted to share a quick story about doubt. Since you are learning design, you might think that once you get some more experience, you'll be more comfortable and will doubt your work less and less.

Over the past couple of weeks, I've been writing, designing, and recording for the second edition of Bootstrapping Design.

As I was creating a video series to accompany the new edition, I was a little worried about the design example that's the core focus of the series. I was worried it might not turn out well.

Why? I've probably designed a hundred websites over the past 10 years. Designing one little example landing page, rationally, should be no big deal, right?

But I've never had to show every step of a design on camera—in front of a bunch of people. With creative work, mistakes are an essential part of the process because that's how you find the right solutions. Opening up and showing those mistakes to the entire world (or in my case thousands of

people reading my newsletter) was, to put in mildly, outside my comfort zone.

I started having thoughts like:

"I'm going to look like a complete idiot if this design doesn't look good."

And doubt started creeping into the back of my mind.

Even though I've worked on all kinds of high pressure design projects and have years of experience in this kind of work, I still doubt myself.

But what we forget is that doubt is a good thing. **Doubt makes your creative work better.** If you are so confident that you never question your ideas, how could you improve?

It's good to doubt.

When I started to doubt that example landing page design, you know what I did next? I worked harder. I paid more attention to details. I made sure the design wouldn't turn out horrible by addressing every one of my doubts. The result was a better design.

So when you work on your own design, embrace the doubt. Don't beat up on yourself for being unsure—doubt your ideas and know it's okay to make mistakes. Work through the questions, and you will know when you arrive at the right idea.

The trick to managing your doubt is to **avoid making decisions based on fear**. Instead, find the exact reason you are doubtful about the idea, and consider how to fix it. Make a rational decision about how to make the design better.

Also, be honest with yourself: is your doubt really about the design, or are you just fearful in general? Part of creative work is knowing when to push ahead, even when you aren't confident. Confidence comes after you take a risk and put that design out there, despite your fear, and find that it was a good design all along. It's exhilarating. You deserve that kind of experience.

Designing is writing.

When working on a design project for your business, dealing with the look and feel is challenging. You have so many aspects to consider. There are so many decisions to make. Because of that, it's easy to get lost in those design details and end up ignoring more important aspects of the project.

That's how it goes when you are a new designer. It's hard to know which details you need to get right and which you can safely ignore. But I'm here to help!

Writing is one of those important components you need to get right. In fact, **the quality of your writing is more important to the success of your design than the quality of the aesthetics**. After all, even the most beautiful form is unusable if the labels are unclear.

So before you worry about color scheme, fonts, layout, and other design considerations, start thinking about the most important design consideration of all: content. (I'm not the first to say this. You should read what the 37signals guys had to say about it too.)

Never, ever use Lorem Ipsum or fake content. When you design, you need to to be working with real content so that you can form a strategy around how to best deliver and

emphasize. Doing that is impossible if the content isn't real.

Further, while creating a composition using real content, you'll start to identify sections of the content that are weak and need editing or rewriting. Your design process and your writing process will start to merge, until you find yourself jumping back and forth between editing and designing. That's when the magic happens. That's how you create a beautiful, concise composition that makes customers sit up and think *"Wow, it's like this design is reading my mind!"*

The best writing reflects the reader, not the writer. To write using your reader's viewpoint, you need two things:

1. *You need to know who your readers are.*

2. *You need to understand your readers through research.*

Many people start businesses before they even know who the business is supposed to sell to. **But instead of an idea, it should all start with research.** Without the knowledge about your customer base that you can only get from research, you won't be able to write copy that makes sense to people. And that copy won't sell the product.

So do yourself a favor. Before you start your next business, start researching. Sift through your research and find an opportunity to build a product. Learn to understand your specific audience. (Learn more about this technique

from Amy Hoy.) Then, **writing copy will become so much easier because you'll know who you are writing to and how to talk to them.** Your design will be immensely more successful, and paying attention to the aesthetic details will be worthwhile because you'll have a solid foundation of great content to build upon.

Better copy means a better design, which means your business will have a better chance to succeed.

On timeless design.

It's time again for all those year-end blog posts about design trends. They're fun to read, but they contain a hidden implication you might miss: design and fashion are easily confused.

Now, any designer will denounce design-as-fashion and proudly state that great design is timeless. We can point to any number of famous works in architecture, industrial design, and graphic design and show rightly that they are equally effective today as they were upon creation.

But what about digital work, like web design? Technology turns over so quickly that digital design seems to be forgotten or deemed ineffective much more quickly than physical counterparts. No one points to a great website from 5 years ago and says "Yeah, that would still do the job today."

Is digital design disposable?

The idea that digital design is disposable can be disheartening. You put the same hours, care, and passion into a digital design as you would the design of a physical object. But if you don't update that site design every couple of years,

you start to feel like it isn't doing its job anymore.

We feel this way because we forget about the purpose of the digital design in the first place: to deliver a message, experience, or connection. That purpose is fundamentally different than the purpose for designing a physical object. Further, realizing this difference brings the core of the matter into focus. It's not the design that matters. It's the message.

So, when you're working on a digital design, don't worry about whether it's timeless. Don't concern yourself with earning web design awards or whether you're committing a grievous design mistake by following a trend. Instead, focus on the goal of your project. Create a great experience for customers. Deliver your message in a way that people can really connect. And, exploit those trends if they serve the goals of your project.

Let the design fashionistas snicker by themselves in the corner. Our work is bigger than fashion.

Good enough design.

When you're building a business, you read about all kinds of best practices you aren't supposed to neglect: design, user testing, marketing, security, etc. So you start bookmarking blog posts and make a to do list. That list gets long really quickly, and you start to wonder. How can I do all this? How could anyone do all this?

Many, many disciplines are required to build a business, and you're learning about how to trim your idea down into a minimum viable product. But even that tiny remainder is intimidating if you aren't already an expert in each discipline. How will you develop a marketing strategy when you've never done that before? How will you pull off a great design?

The problem is, everyone has an opinion about how you should build your business, but no one is willing to stop short and say "That's good enough for launch." The marketing experts say you should have a flawless marketing plan. The design blogs say nothing short of the best will do.

Then, what started as the minimum gets bigger. And bigger.

If you're bootstrapping a business, there is such a thing as good enough design. Furthermore, *there's such a thing as*

too much design, too.

On launch day, your design doesn't need to be perfect because your goal is to sell a product, not to win a design award. Similarly, you can still earn sales if your marketing is imperfect.

If you keep refining the design, the marketing, the elegance of the code, and whatever else, you won't get to launch. **So, you have to learn to be comfortable with good enough.** You have to learn to filter all that advice, and decide which pieces are essential for your business.

Work towards good enough and launch your product. You can fix and tinker later.

Don't try to be creative.

Getting good at design means cultivating your taste. Right now, you don't have taste that you can trust. Eventually you will, but for now you cannot trust your creativity. It will only lead you down the wrong paths. While you are still learning basic principles, don't try to be creative. Instead, focus on simplicity, clarity, and the cold, hard science of what works.

As you become more comfortable with design fundamentals, like those in Bootstrapping Design, allow yourself to branch out and experiment. You will make mistakes. However, making mistakes is part of creativity, so don't beat yourself up. Instead, try a different decision next time.

We all think our creative ideas are great, but design is not just about having ideas—it's about choosing the correct ones. Doing that takes experience. Through practice and hard work you will gain that experience, and you will then be able to indulge in the luxury of your creative ideas.

Until then, don't try to be creative. Instead, observe what works and imitate it. Practice. When you look back at older work and see its flaws, don't be ashamed. Seeing the flaws in your past work only means your skill has increased. Be proud. Use that knowledge to do better on your next project.

Knowledge vs. skill.

How does one become a good designer? Is it by gaining knowledge or by improving a skill?

You're thinking it's both. And, I'm sorry to say, you're wrong. Here's why.

The knowledge required to become a better-than-average designer is miniscule. In fact, you could easily become a great designer and produce insightful, effective work without any formal training or knowledge of design theory. Many revered designers know nothing about various design topics that the larger community deems essential, but they still produce great work.

This is because designers are communicators. Designers do not exist merely to make pretty pictures or to imbue everything they touch with fashion. Designers work to make communication more efficient. This communication can take a variety of forms: communication between a human and a computer, a company and a consumer, or software and its user, for example.

Of course, this is not to say design knowledge is worthless. Knowledge will help a person with skill make informed decisions and thus increases the quality of the work.

However, you've probably observed that reading up about design doesn't make much difference in the quality of your design attempts. That's because when it comes to design, skill is more important than knowledge. You haven't found a way to build your skill yet.

Why does all that matter for your startup?

It matters because if design is a skill more than a knowledge, it's also more difficult to achieve as a bootstrapper.

This is why you've had such a hard time pulling of a great design for your business. Design isn't something you can just grab off the shelf and plug in. It's also not something you can just read about then implement the next day. There is no quantifiable threshold that makes great design: instead there are a million degrees between awful and incredible, with skill being the determining factor. This is also why great design can be so expensive.

So if you're bootstrapping, what are you supposed to do with this? Don't be discouraged. One of those degrees of design quality between awful and incredible will be good enough for your business. Even if your design skill is developing, you can still produce a design to strengthen your business. You aren't aiming for perfect—you're aiming for good-enough-to-launch.

Break the grid.

You know what grids are, and why you should use them. Designers won't shut up about how awesome grids are. So you follow suit.

But then, you implement something like 960.gs in your project, and the layout still doesn't look right. After scratching your head a bit, you write off the issue because, after all, you aren't a designer.

This is happening because you don't know when to break the grid.

Now, some designers will disagree. To them, *The Grid* is a force of nature not to be trifled with. I say, the grid is there to serve you—not the other way around. So, if it gets in the way, break it. (Or switch to a different grid.)

Sometimes, **you have an element with dimensions you can't change**, but it doesn't fit the grid. What do you do? *Break the grid.*

Or, you have **two column text and the grid's built-in padding almost makes the lines run together.** What do you do? *Break the grid.*

Other times, **positioning elements correctly within a grid just looks bad.** What do you do? You should know by now.

When you break the grid, no one will notice. And even if

they do, here's what they'll think: *"This designer broke the grid, but the design still looks nice, so they know what they're doing."*

Great designers sometimes break their grids. Other times, they forgo a grid altogether. Being a designer doesn't mean adhering to one prescribed way of working with religious zeal. Design is about critical thinking. If a tool doesn't fit the job, put it back in the toolbox.

Should you design in Photoshop or HTML?

Should you design in Photoshop or HTML? It doesn't matter. That's the wrong question.

Gurus tend to take a firm stance to establish their points, but this can be a distraction to those of us who operate outside the monastery.

We're building stuff. Right now. Doctrinal debates don't help us get stuff done.

So, the 37signals designers are more efficient when they design in code. Learning from these talented designers' experience is a good thing, but there also a problem with it: just because it works for them, doesn't mean it will work for you.

The correct question is: **which tools help *you* to be the most efficient?** People who suggest that you shouldn't use Photoshop or that every design should be responsive are just trying to help you be effective. They're sharing their experience to help you out.

But when reading this advice, remember that each of us has different knowledge and experience. We each have

different strengths and will find varying tools to match those strengths.

That might seem obvious, but letting advice from gurus dominate your thinking is easy. You have to be the filter because no one else understands your situation. So when you hear this kind of advice, instead of agreeing immediately because you respect the person, think: "Ok, they're suggesting that a different way of working might be more productive. Is that true for me?"

Think for yourself. People become gurus by finding success one way, but there are many paths to success. There are many ways to design a project. You have to find your way. If you blindly follow celebrity advice, you will end up with a patchwork of incompatible and even conflicting advice (and maybe you'll write a bitter-sounding post about your experience like I did once).

So, if Photoshop helps you to be more productive when designing a project, use it. If you can mock up your design concepts more efficiently in HTML & CSS, do that. Pick the tools that work best for you, and not some guy out there on a soapbox. Still, go ahead and listen to him. Seek advice anywhere you can get it and know the intentions are good, but people sometimes get a bit too zealous.

Not designers.
Not coders.

Just builders.

Designers should learn to code. We've been talking about it for years, and the attention this issue rightfully earned brought about a change: designers are writing code now more than ever.

This movement has produced an interesting and unintended consequence: the line between designer and coder is blurring.

I propose we remove the line altogether.

Why? Say it with feeling: design concerns and code concerns are the same concerns. Internalize that for a moment. Designers and coders contribute to the project at hand equally—we both want the project to succeed. Because of that, our goals and concerns are the same.

Designers don't want to build an app that runs slow because of agonizingly complex SQL. Coders don't want to build an app that is hard to use. An incredible app requires both incredible code and incredible design. Without one, it's not incredible.

Designers are tired of being the gatekeepers for visual detail and user experience. They're tired of having to explain to the team that their concerns are not petty but are instead a

major factor in a project's success.

Coders are tired of dealing with designers who don't understand how things get built. They're tired of turning down silly features that have dire coding implications or are otherwise untenable.

Too often, designers and coders are at odds. Their interests compete. They disagree. If our goals are the same, the classic face-off between designers and coders needs to end.

The influx of coding designers proves we can do it. We can align our goals and start to alleviate all that mutual frustration by sharing our duties and skill sets.

Of course, specialization is important. If each of us were to just dabble in every possible skill rather than pursuing mastery, nothing would get done. However, we need to stop defining ourselves by too-narrow sets of concerns, and instead become well-rounded professionals who each excel in a specific area.

But there's a problem: coders need to catch up. I could make a broad generalization that most coders don't know about design, and it would be unfair but perhaps a tiny bit true. Instead, I'll say that I don't anecdotally know of many coders who understand even design basics. Do you?

Here is my plea: let's all start learning about the opposite discipline and continue blurring the line between them. Designers: learn to code and understand why technical

proficiency brings about great software. Coders: learn about design and understand why the little details affect the entire project. Let's stop assuming our colleagues are out to undermine our interests and get on the same side.

Let's stop being coders and designers—let's become builders.

Perhaps too conveniently, I've written a design ebook for technical founders. If you're a coder who wants to start learning design, it just might be a good place for you to start.

Design's Iron Fist and other essays

By Jarrod Drysdale

First Edition, August 2014

This ebook is available for free at http://studiofellow.com

If you enjoyed this ebook, please support me by purchasing my products:
http://bootstrappingdesign.com
http://cascade.io

Email me anytime at: hello@studiofellow.com