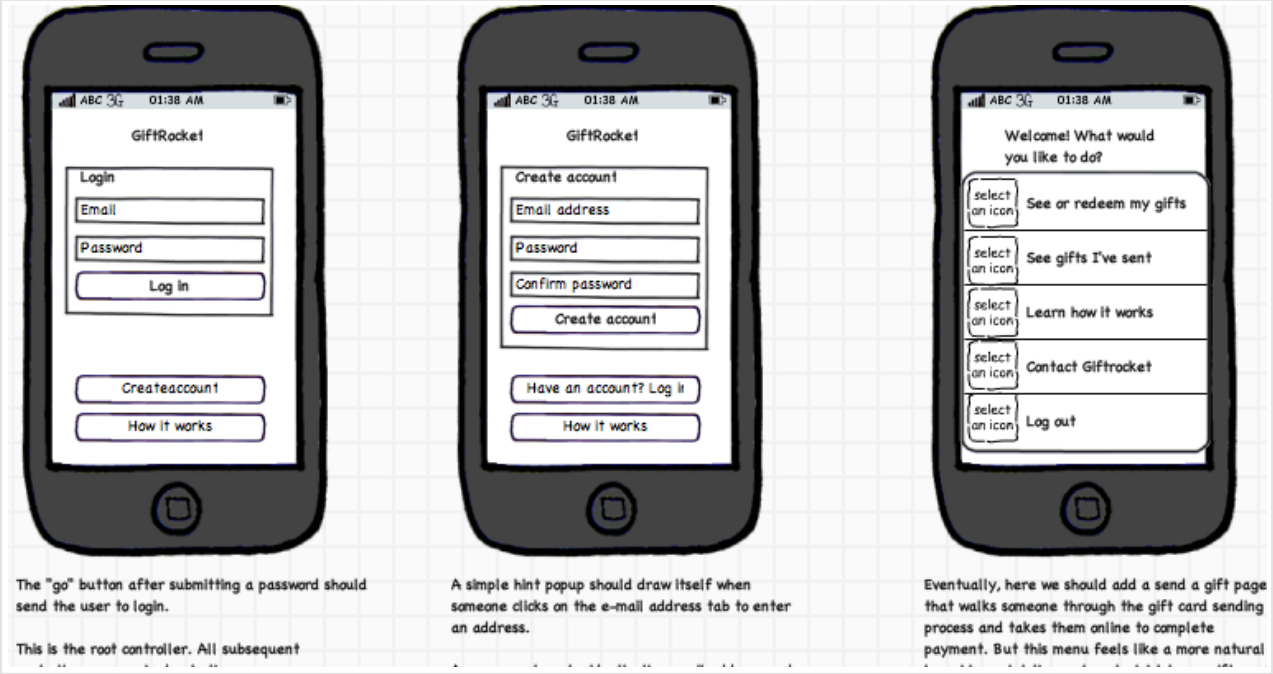
giftrocket.com

Learning to Design: How I Bombed Art Class But Still Designed A Remarkable Website

6 min read • original



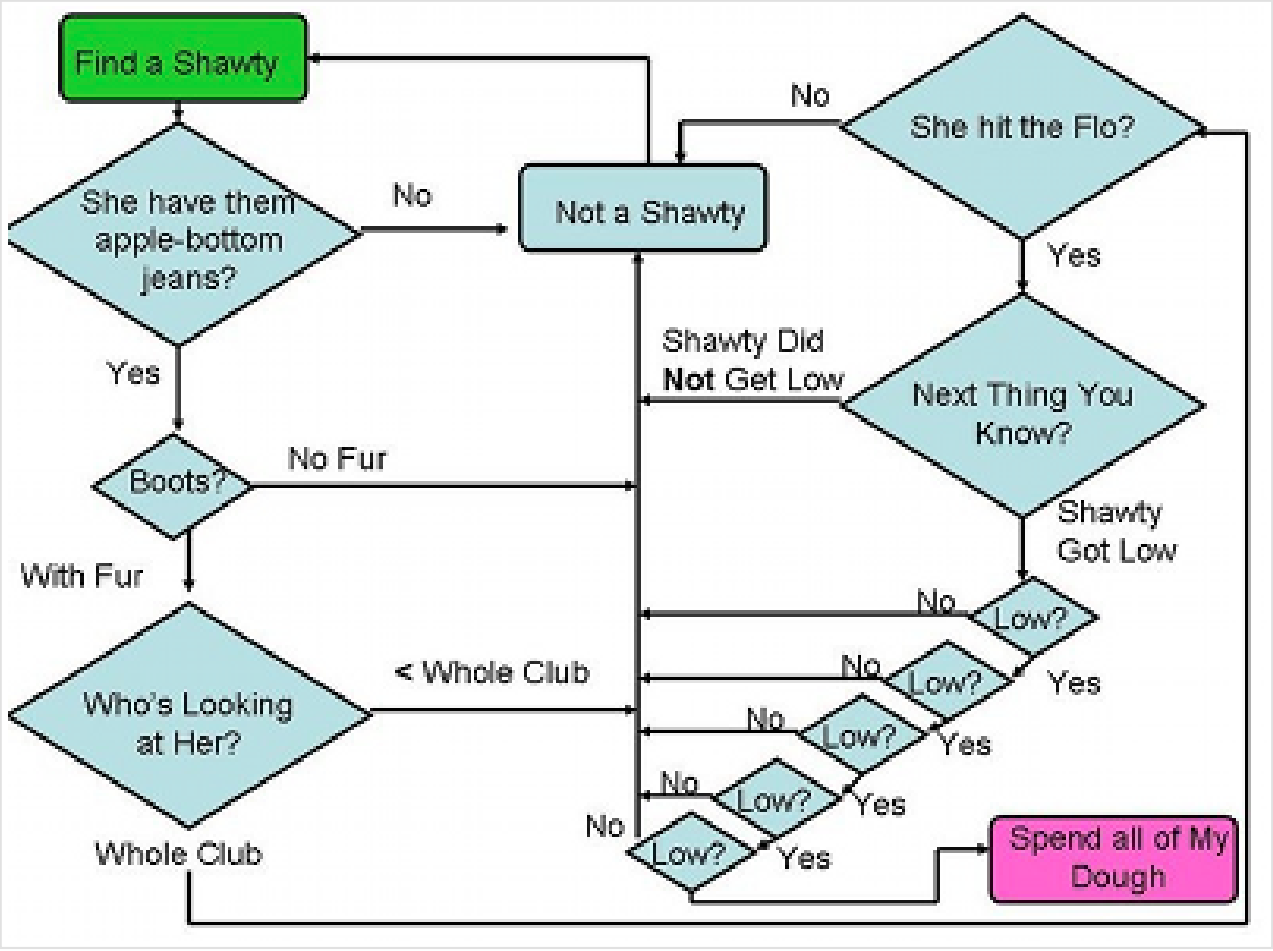
In December 2010, I was tasked with designing GiftRocket's original website. As a former management consultant, my only design credentials were making too many powerpoint slides and getting a C in middle school art class.

So I spent a considerable amount of time learning to design things for the web. What I ended up building for GiftRocket was generally well received, and in one isolated unexplainable incident, praiseworthy. My design wasn't the prettiest thing, but it was usable and intuitive.

But because of my lack of formal background in the topic, I also **learned how to learn** web design. In that now, I can break things down and explain how a new designer should approach the topic.

So, the time has come to **document the process by which I learned to design, along with the resources I found most helpful**. This post is really intended as an article to my former self, as an instructional on how to approach the topic. Here's the three step process I would advise my former self to follow:

1. Determine your user flows.

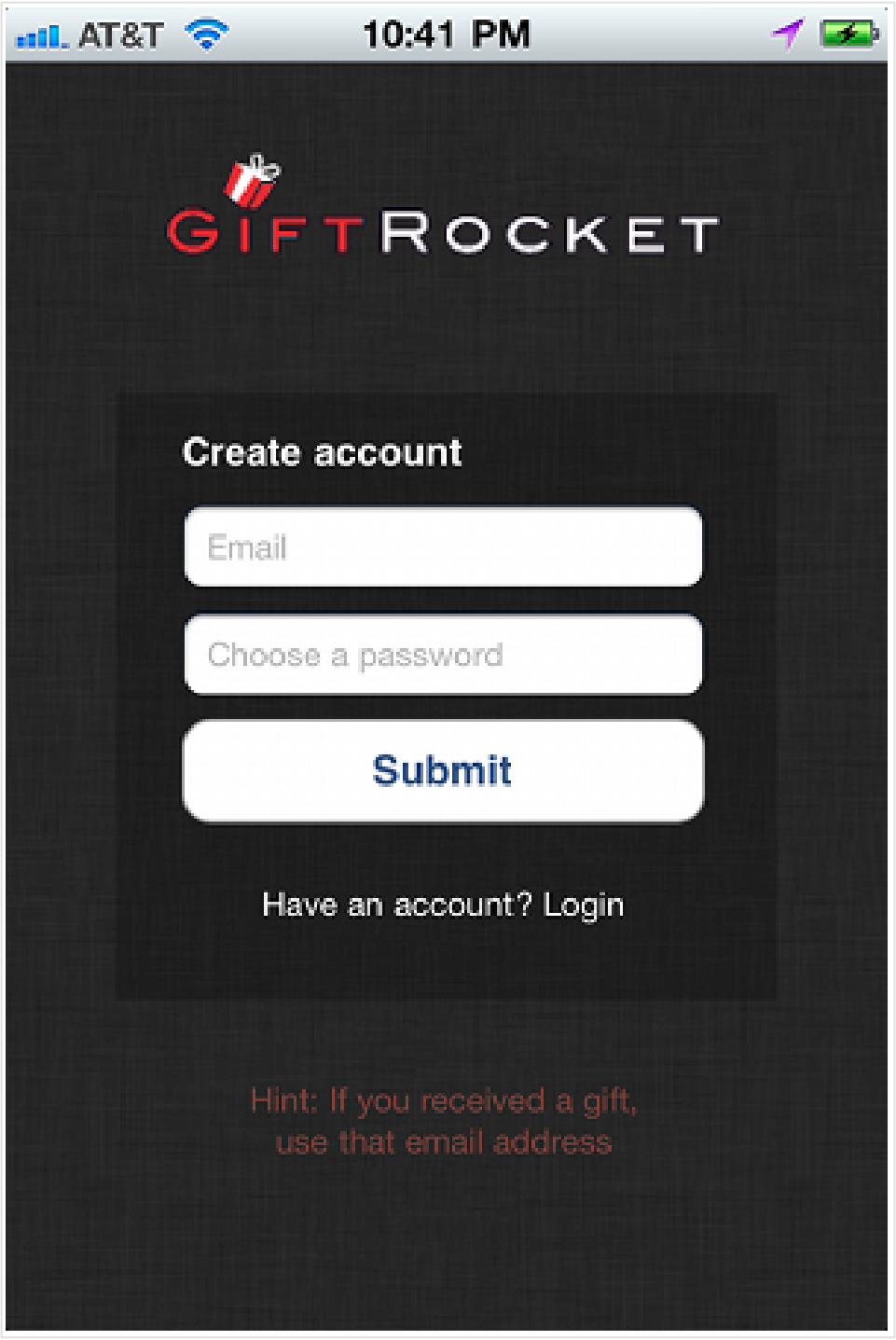


When I started thinking about design, we only had a general concept for GiftRocket. We had to define how the product would work what features it would include, and how those features would translate into pages.

User flows are an imagination exercise in how users will interact with your product, put onto paper. In the process, a lot of important questions about product features come up. Here are two examples of some questions we had:

Should we have a shopping cart? Would users be purchasing for multiple people, or just for one person at a given time? If the answer was multiple people, then it'd be a pain for someone to re-enter credit card information. If the answer was a single person, we'd be creating an additional feature that no one used that would unnecessarily present an option to the user before checkout. We decided that most users would be buying one at a time (and ended up being right).

Should users redeem via native apps or a web app? Most mobile developers face this choice. We decided initially to build native apps. Below is a screenshot of the GiftRocket app:



Authentication is a problem with native apps. With web apps, you can send a user an email with a custom URL that automatically authenticates them. But with native apps, it isn't possible to launch an app and provide it a packet of information, elegantly at least (If anyone has a good solution here, I'd love to hear it). In most situations, the user has to create an account and confirm it via email. Moreover, the user has to download the app as well.

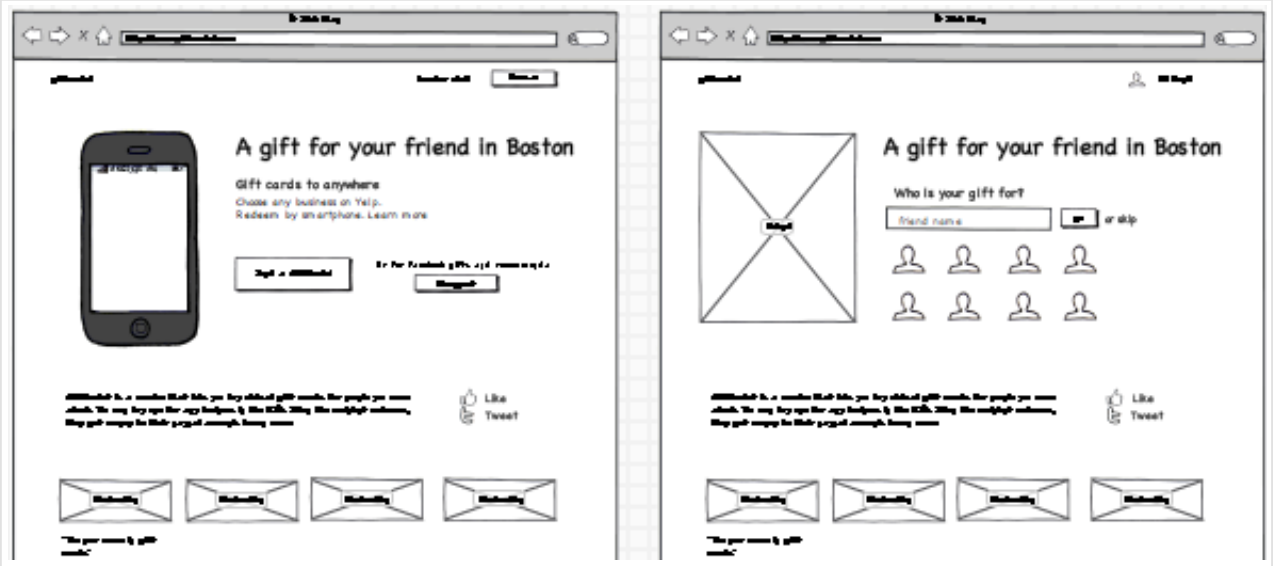
We took the native app version of our product to Garry Tan during office hours, and he said "this is too complicated." He was right– many users didn't want to download an app, create an email and password, and confirm it– just to redeem their gift card. So we shifted to the web app model.

We should have invested more time here. It would have saved us engineering work and saved me from having to learn Objective C. Despite that though, there were probably 30 other small decisions we made about our product at the time. The hope is that based on this step, you have a well-defined set of flows, and a rough idea of how they translate to pages within your site.

Resources on User Flows

- A Project Guide to UX- this book had some excellent information on flowcharting, but was more directed at a UX professional. Worth skimming.
- 12 Examples of Lazy Registration- A great overview of lazy reg. (not asking for information from a user unless it is absolutely necessary)
- Rebecah Cox's Quora Design notes- Rebekah defines how she got medieval on information hierarchy on Quora. My favorite quote is "A lot of decent features have been dropped or hidden or otherwise cut in order to pursue this goal but it's helped ensure that only the most important information is on any given page. And with fewer elements comes fewer decisions a user has to make as they interact with the interface."

2. Create wireframes



Now that you know the pages you have to create, and the point of each page– its time to create wireframes representing the content on the pages. The biggest challenge here is structuring the information in such a way that is logical, easy to follow, and hierarchical.

It's important that your user be able to quickly identify what they're looking at and where they are on your website. There should be a set of actions and accompanying text that guides them to the next step in the flow.

The first step is to remove every piece of content that doesn't help a user progress along the flow (or at least take those pieces of information and set them to the side). Good design is reductionist. After that, the focus is laying out the information in a logical order.

Visually, follow the C.R.A.P principles (contrast, repetition, alignment, and proximity) to arrange the remaining information. And don't be afraid of whitespace. Look to other sites for inspiration.

Here's an example of a page I wireframed in Balsamiq:

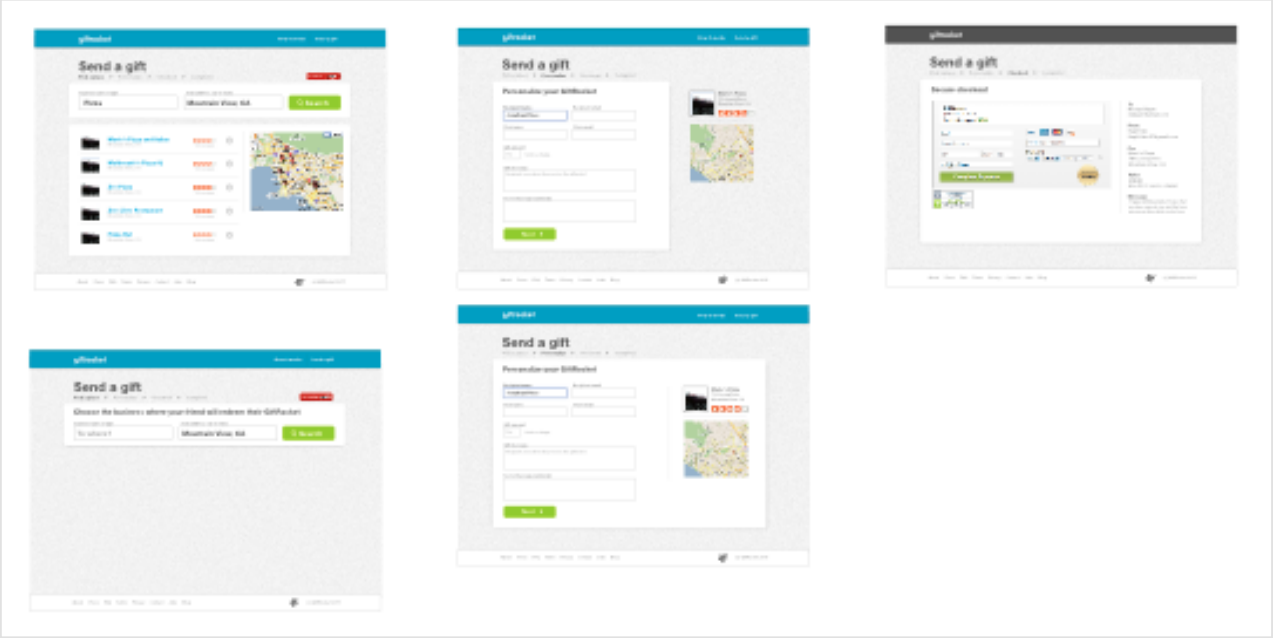
There's a lot of good stuff happening in this wireframe. The header is big, and there's a progress tracker that leaves the user clearly grounded in the purchase process.

However, the problem with this wireframe is that it isn't specific enough. It isn't a blueprint– its a vague definition of what the page should look like. Where do the labels on the text fields go? What are they? How is the stuff on the right, which seems like ancillary information about the gift, being separated from the form on the left? What does the footer look like? In retrospect, I would have invested more time into this phase.

Resources on layout and wireframing:

- Don't Make Me Think- This is a seminal work that defined much of our creation of user flows and wireframes. This is a must read for anyone learning design.
- The Non-Designer's Design Book- this book laid out the four principles of design (contrast, repetition, alignment, proximity) that helped us create a clean look for our site.
- Whitespace Is Not Your Enemy- Helped me shake the habit of filling space with stuff. The title is now a favorite battle cry.
- Redesigning Mailchimp- A great read on how MailChimp enforced hierarchy and consistency within their site by creating a set of rules and restrictions for the way that their site would appear.
- Subtraction- Khoi Vinh's Blog- The design itself is beautiful, but it is a sheer joy to watch him decompose it and show how he uses a grid system to maintain alignment and consistency with spacing.

3. Create mockups



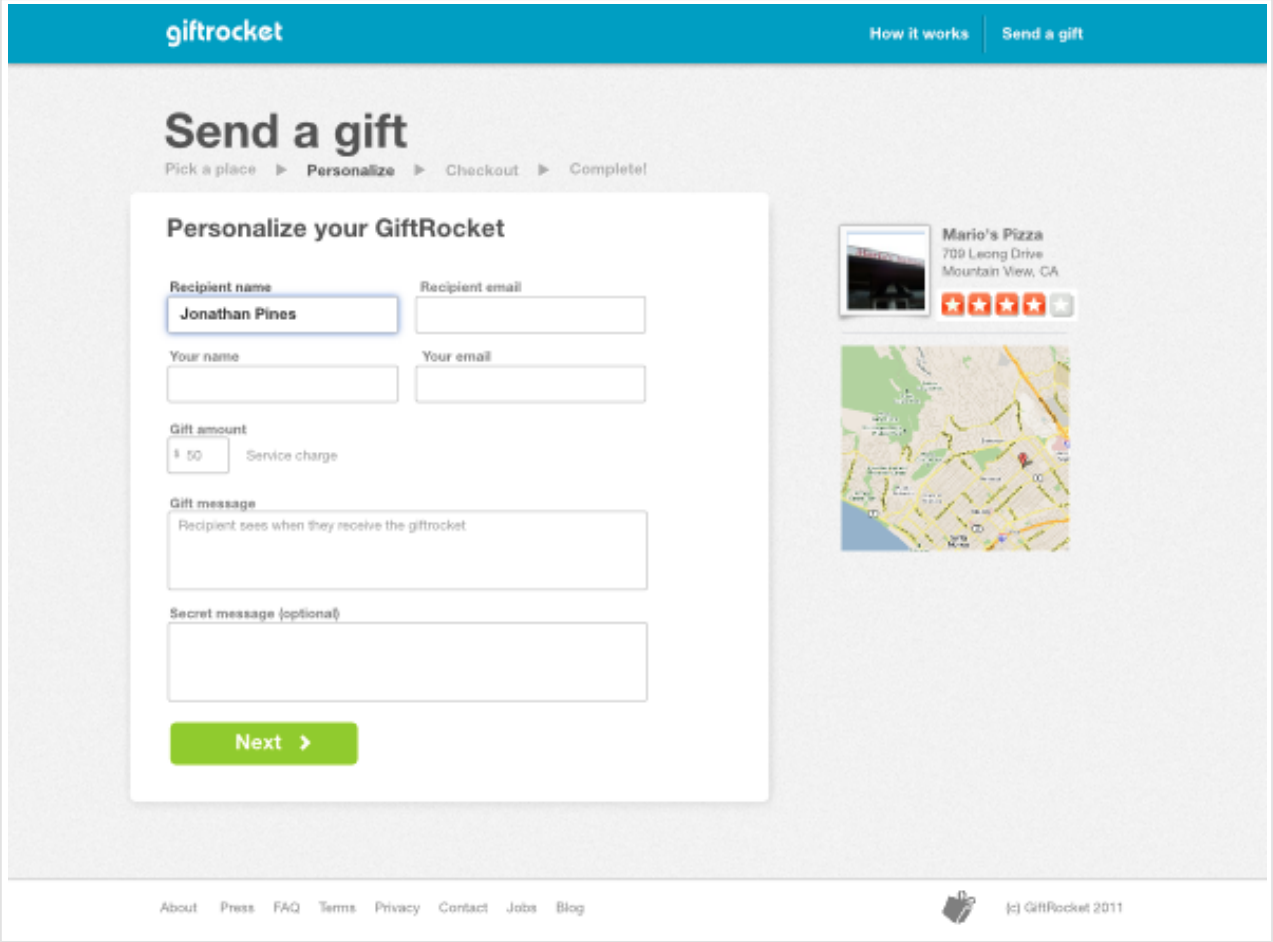
If the wireframes serve as the blueprints for the actual site, the mockups are the interpretation of those blueprints. They should either be a real site, or be sufficiently detailed that you could open the images in a browser and get a sense for how the site would look.

I designed the site in Gimp & Inkscape (free relatives of Photoshop and Illustrator), but more talented front-end engineers can probably build directly in HTML / CSS.

There are probably three things to focus on:

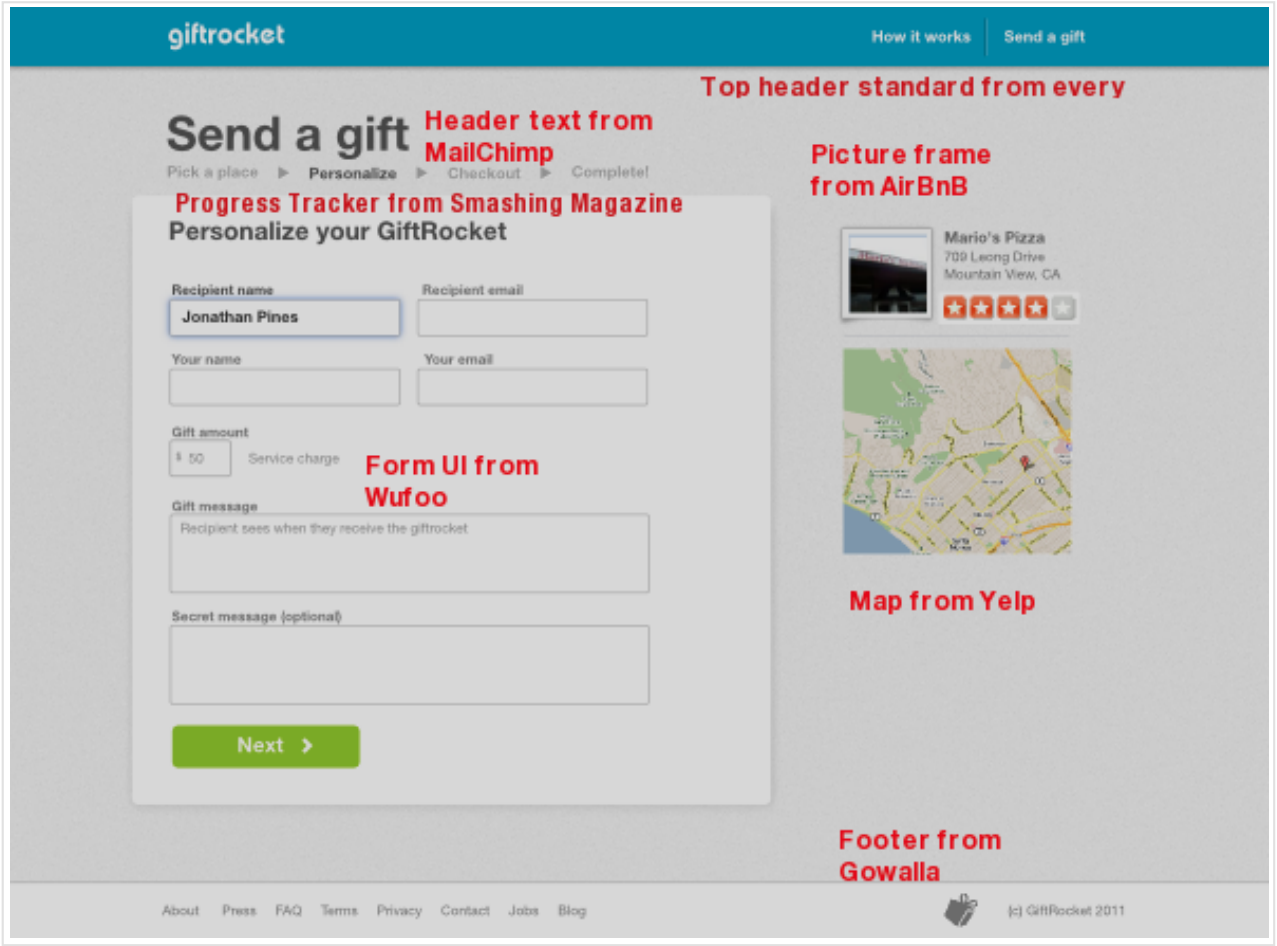
1. **Colors and feel.** For the first time, you'll be touching color. Focus on using a color palette that reflects your product. There are lots of resources online that helped us do this. Do things that are different contrast properly? Is the eye still being guided the way you'd like it to?
2. **Usability.** Do buttons look like things to click? Do things feel responsive and intuitive? Does everything feel legible?
3. **Overall polish.** Do things look pretty? This starts being far more subjective, but I'm still convinced that there's a strategy here.

This was the mockup I created from the gift page wireframe.



I've found that this portion takes the most artistic talent and experience. You can get to the previous stage having only browsed the web a lot.

But if you're new to this sort of thing– I suggest taking UI elements from other sites. Much like a new guitarist starts piecing together a song from licks he's heard elsewhere. I fell into this category, and drew from a variety of influences:



Resources on creating mockups:

- The Pantone Guide to Communicating with Color- the most important page is 63, which covers the different vibes that colors have.
- Kuler- a great resource that helps you create and look at color swatches
- The Smashing Book- excellent for learning a lot of the web 2.0 styles that pervade web design these days.

- Design for Hackers- I've been impressed with David Kadavy's blog posts, and have high expectations for his forthcoming book.

4. Iterate

Your first design won't be right. Design isn't too different than writing or coding. I generally agree with Paul Graham's advice that it is best to bang out a quick first version, get feedback, and revise. I probably created a new design of the site every couple days for a few months, often showing off the latest mockups at YC dinners. The biggest change I'd make is investing in the user flows and wireframes more– it would have saved a lot of iteration on the mockups.

What design advice would you now give a younger version of yourself?
