

THE

UX

READER

The UX Reader

Insights on process and purpose from MailChimp's User
Experience team

© 2014 MailChimp

TABLE OF CONTENTS

Introduction

Collaboration

Research

Design

Development

Refinement

Resources

Contributors

Introduction

Here at MailChimp, we like to show our work—and magical things seem to happen when we do.

It forces us to reflect. “Let’s see. How did I do that? And why is that the best way? Oh!”

Reflection helps us see the best parts of our process. “We saved ourselves so much time when we built it that way.”

Sharing the best parts of our process sparks conversations. “You did it that way? Interesting. Why?”

These conversations keep us humble. “You know, you’re right. We probably could’ve done that differently.”

And by remaining humble and open to new ideas, we grow. “We used to do it this way, but after some consideration and lots of experimentation, we discovered that there’s a better way.”

We show our work, hoping to get better at what we do. This book is the embodiment of our quest to reflect, refine, and grow. We made it in hopes it’ll help you do the same.

But there's another reason we made this book: It gives us an opportunity to walk a mile in our customers' shoes.

In 2013, we started the UX Newsletter as an excuse to use the MailChimp app on a regular basis. It helped us see countless things we could improve, and the refinements came fast. Now we're taking our little experiment further by creating and connecting an e-commerce site to MailChimp, just like so many of our customers have done.

Incidentally, the revenue from this book won't land in our pockets, but will go to [RailsBridge](#). We love their commitment to increasing diversity in the tech world by teaching women and people from other underrepresented groups to code. And it's especially gratifying to see our efforts to improve our own work do the same for others.

Longtime UX Newsletter readers will recognize in this book some of our most popular articles, refined and given new life. But this isn't just a greatest hits collection. We've peppered new content throughout, and the resources section at the end will serve as an evergreen reference as you hone your own practice.

Our perspective on user experience design is reflected in the organization of the book. We start with an examination of collaboration and teamwork. The troika of UX follows: research, design, and development. We conclude with a selection of articles on refinement and other processes that never really end.

As you read, you'll surely have questions, observations, and feedback. In fact, we're counting on you to get in touch and [share your perspective](#). Really. We're staring at our inbox right now.

We're showing you our work, work that we love. If it informs your work, even in a small way, we'll be delighted.

Aarron Walter

Director of UX at MailChimp

COLLABORATION

Building a UX Team

Aarron Walter

Collaboration by Design

Aarron Walter

All Hands on Deck

June Lee

Asking for Help

Fernando Godina

On Good Terms

Gregg Bernstein



COLLABORATION

Building a UX Team

AARRON WALTER

“User experience” is such a nebulous term. The UX container can hold almost any discipline: research, design, development, even customer support. So how do you put together an effective team when the profiles of the members are so vastly different?

Back in 2008, when the MailChimp UX team was just me, I had to be a generalist. I designed UIs, wrote front-end code, built prototypes, interviewed customers, and conducted usability tests. Working on so many different things early on helped me see how it all fits together and shaped my views of the type of UX team I wanted to build.

I knew I wanted to grow a team of thinkers and makers. At big companies, UX teams often focus on wireframes and workflows

but don't have the power to design or build the UI. Things get lost in translation as ideas are passed to other departments. A team filled with specialists but no generalists creates silos and division, which lengthens iteration cycles, causes entropy and confusion, and absolutely pollutes the user experience. I wanted to build a team of generalists with a broad vision of the user experience and specialists that could hone in on the details.

Over the past 6 years, my team has grown slowly and methodically. I brought on someone to design, then someone to code, then someone to run usability tests, and on and on. But they weren't just "someones"—they were folks who could contribute, inspire, and thrive. We're small by design, just 12 people strong. We have 4 design researchers, 2 UI designers, 4 front-end developers, 1 expert HTML email designer/developer, and myself. Small teams make communication easier and leave no room for dead weight. Collaboration is easier when you know you're working with smart, capable people.

When I think back on how we've grown, I can derive a few key lessons about building a sharp UX team. These lessons aren't a

magic formula, but by combining a bit of rigor with a lot of intuition, we've been able to get a lot done with relatively little.

Easy to hire, hard to fire

Hiring is hard. It sucks up so much time, and it often feels like you'll never find the right person for the job. But it's the most important task of anyone building a team, UX or otherwise. And that's because it's easy to hire someone who's mediocre, but it's really hard to fire them.

Mediocre people are a drag on quality and morale, but they tend to do just enough good work to stick around—managers have a tough time justifying letting them go because there's no actionable offense. The scent of mediocrity on your team can also scare off talented candidates. Mediocrity is an albatross we tether ourselves to when we don't give the hiring process our full attention.

When you hire, look for skill fit, but don't make it your primary evaluation criteria. Look for passion, curiosity, selflessness, openness, confidence, communication skills, emotional intelligence, and intrinsic motivation, too. These things can't be taught—most skills can.

It sounds a bit nebulous, but I always look for the right energy fit too. I once interviewed a candidate and knew from his crushing handshake and deafening greeting that he would be too overbearing for my team. As the interview proceeded, my hunch was borne out. This guy's alpha energy would instantly squelch the open collaboration on my team. We spend more time with our colleagues than we do our families. Why wouldn't we listen to our gut when deciding who to hire?

When building a new UX team, your instincts might draw you to industry rock stars. Unfortunately, these folks often struggle to collaborate and can intimidate colleagues, however unintentionally. When hiring, ask yourself, "Can this person say 'we' instead of 'me'? Can they let someone else have the glory? Can they put in their best work, then relinquish control to someone else to take it further?" If yes, you've got a strong candidate worth considering.

Respect

Seeing your work mistreated can be demoralizing. Researchers throw themselves into studies that uncover insights that can change a company, but their findings go unheeded. Designers labor over pixel-perfect UIs, but the build-out cuts corners.

Developers write brilliant code, but the release gets spiked. You probably have a few similar stories of your own—it's the stuff that resignation letters are made of.

This dynamic is bad for individuals, bad for teams, and bad for the companies they serve. So why does it happen? Simple: A lack of respect between peers.

Respect comes from taking time to understand one another and our individual areas of expertise. Specialists who have mastered their craft are ineffective if they don't have a general understanding of how their contributions relate to those of their colleagues. Great designers need to understand engineering enough to empathize with, listen to, and respond to the people who build what they design. Great developers see value in making an app as usable as it is powerful, and they're willing to go the extra mile to make the UI a little more elegant, a little more efficient for users.

Respect fosters a can-do attitude. When colleagues value one another, they're more open to sharing ideas, even not-so-great ones. We could do a lot better if we started with **“Yes, and...”** instead of **“No.”**

Respect between design and development is the most critical ingredient in making great products, but it rarely happens organically. It has to be a core value of a company, demonstrated daily by leaders.

Autonomy

Even if they operate in a respectful environment, teams can become demoralized when they have to beg for permission. It's hard to experiment if you have to get sign-off first, and it's hard to make giant leaps forward if you don't experiment. If failure is not an option in your organization, then neither is success.

Teams need autonomy to make decisions quickly, to follow their gut, to explore. Autonomy empowers people to do their best work on tight deadlines and with limited resources. You have to hire great people, provide vision, then get the hell out of their way. And to do that, you have to trust each other.

Our MailChimp UX team is tiny, but our size isn't a shortcoming—it's an advantage. Communication is easier when teams are small. We're able to make plans quickly, then get back to making. Each little team has the autonomy to make decisions

about their own work, and when there's uncertainty, they discuss with another autonomous team that can provide more definitive direction.

There's a balance to be struck, though: Absolute autonomy can lead to chaos. Though each team operates independently, we never forget that we're part of a larger organism. Our decisions often affect the work happening elsewhere. So when big decisions need to be made—like when we redesigned the MailChimp app—team autonomy has to give way to the perspective of the whole company.

Meetings are often vilified, but there's great value in short, regularly scheduled check-ins. Individual members of autonomous teams need to be aware of their colleagues' activities. A quick round-table report of what everyone's working on creates opportunities to collaborate and inform. Brief chats in the hallway or by the espresso machine are also effective at keeping people moving on a common trajectory. These are common occurrences around the MailChimp HQ.

Parallel cycles

Lean and Agile methodologies are the religion of technology companies trying to gain competitive advantage by iterating quickly on their products. The problem is, fast iteration too often happens at the expense of research. Things get done fast, but they're not always the right things.

The MailChimp team is agile and lean, but in lowercase. We believe in many of the tenets of these methodologies, but we've never been keen on dogma. "Move fast and fix things" pretty much sums up our approach.

For a while we tried to tie one research workflow to the 5 week release cycle followed by our UI design, front-end devs, and engineering teams. That works OK when doing usability testing to find refinement possibilities, but it greatly restricts deeper studies. The design research team is often engaged in long-term studies about key issues that might shape company strategy or make us rethink pieces of the app. This work requires lots of customer interviews, surveys, ethnographic research, and—most of all—time. The cycle for deep research is considerably longer, and happens parallel to the rapid cycles of app development.

Though the 2 cycles operate autonomously, the teams' work has to remain connected. Research that goes unapplied is of little value. So the research team shares salient insights with designers and developers as they find them, rather than waiting for months to report back. And they don't do this expecting immediate action, just to offer context and meaning for the work already underway.

When big studies are completed, we talk about how to fit recommendations into our overall roadmap. Having in-depth research continually trickling into rapid iteration cycles helps ensure that we're not only moving fast, but that we're also moving in the right direction.

Create a culture of empathy

Making new features is fun. Fixing bugs and refining workflows is not. But to make a great product, you have to do both with equal measures of enthusiasm.

Refinement requires motivation and, for a UX team, nothing is more motivating than watching users struggle with your app. From time to time we invite users to the MailChimp office to conduct usability tests. The UX, engineering, and QA teams

watch real customers doing real work in MailChimp, and we squirm in our seats when they struggle. The outcome of these tests is always the same: We're made so uncomfortable on their behalf that we're compelled to make things better immediately.

We also do remote usability testing, conduct customer interviews, answer some support-related tweets, and read thousands of account closing surveys and customer feedback emails. The research team is largely responsible for this sort of work, but even the front-end devs visit customers in person to watch them use the app.

As your customers' experiences become more visible, your team will become more empathetic. They'll not only be motivated to refine, they'll do it with speed and passion, and without being asked.

Tell stories

When we started to fold design research into our UX practice, I thought it was enough to collect the findings and make recommendations. But I was wrong. Research can't create change in an organization until it has been turned into a compelling story.

MailChimp has been known to write 50 page documents filled with interesting insights about our customers and how they use our app. It's stuff that can help us make a better product—and could even shape the direction of the company. But very few people are willing to pore over giant tomes of research. It's time consuming, and it's not much fun. So we thought, while we're on a quest to understand how to make our customers' experience better, why not do the same for our colleagues?

We're experimenting with creative ways to turn high-level findings into posters, videos, and elegant web page layouts. In these forms, our research can be grokked in seconds and shared easily between teams. Research is much more influential when it's made accessible to others.

Keep going

I've been working on building and managing a UX team for years now, and I still don't have everything figured out. It always seems like there's some magic formula just around the bend, and from what I've heard from UX team leaders at other companies, I'm not the only one out here looking for it. But maybe, just maybe, that formula doesn't exist. There's no one way to describe “user experience,” so why would there be one

way to build and manage a UX team? Getting research, design, and development to operate in harmony is what our ambiguous craft is all about. This much I know for sure: When smart, capable people with complementary skills are united by a deep desire to help customers, great things happen.

COLLABORATION

Collaboration by Design

AARRON WALTER

A few summers ago I visited the [Stanford d.school](#) to give a guest lecture to [Enrique Allen's d.media class](#). Students were busy working on prototypes, gathering feedback, and iterating—it was a classroom with the spirit of a startup.

During my lecture, an attentive audience sat on chairs and couches scattered around the room. Afterwards, the seating was rolled out to the perimeter of the room and small tables were rolled in for collaborative conversations about student work. I was struck by how the dynamic of the room and the behavior of its occupants changed quickly and effortlessly along with the furniture.

As the class wrapped up, co-instructor [Scott Doorley](#) gave me a copy of his book [Make Space](#), which details his methods for creating collaborative spaces. As it turns out, the flexibility of the classroom was no accident. Scott told me they'd been experimenting with ideas to improve the program's learning experience and teach students how to think more creatively through the design of spaces.

He took me on a tour of the school, showing me spaces for ideation, productivity, and quiet reflection. The artifacts of creative thinking were plastered all over the walls and scattered throughout rooms. It was clear that ideation and collaboration were ingrained in student work—not only through the design of the curriculum, but also because of the design of the physical space.

The experience changed the way I think about workspaces. And the timing couldn't have been better: Back home in Atlanta, [Ron Lewis](#), our creative director, and [Mark DiCristina](#), our marketing director, were hatching plans for a new, unified design studio.

Designing the MailChimp studio

Ron was already doing some research of his own, visiting startups and design spaces in the Southeast to collect ideas that might influence our plans. After much debate, Ron, Mark, and I arrived at a core set of principles that shaped the design of our studio, and in turn, how we collaborate.

1. COMMINGLE AND CROSS-POLLINATE

Because collaboration was our central motivation for creating the new studio, we carefully considered the seating chart. Between our teams we have design researchers, writers, graphic designers, photographers, illustrators, developers, and interaction designers. It's a diverse set of skills and personalities that make up this creative soup.

We put everyone in a wide-open space on the office's fourth floor. Rather than segregate the teams, we mixed people up. We looked for complementary skills and personalities, then positioned accordingly. We put analytics people next to design researchers, UX front-end devs next to marketing devs, and designers next to writers. Commingling across disciplines encourages us all to look at projects from different perspectives. Regardless of what we each do day in and day out, it's all connected and equally important.

2. FACILITATE MOVEMENT

Sitting still for hours on end not only atrophies muscles, it softens minds. So we made our workspace open and filled it with desks, couches and standing tables to make it easy for people to move around and collaborate, or just to get a change of scenery.

Our open workspace also provides a landing place for smart designers who pass through Atlanta and need a place to be productive. [James Victore](#), [Dan Benjamin](#), [Brad Frost](#), and many more have dropped by our studio and claimed a spot for a few hours. We love the energy and ideas guests bring to the studio—it keeps us inspired.

3. LET IDEAS HAPPEN EVERYWHERE

Ron visited [The Iron Yard and CoWork](#) in Greenville, South Carolina, and loved the [Polygal](#) walls that defined the office's workspaces and served as giant whiteboards. We borrowed that idea when we built our studio. And we've discovered that, when the tools are always at hand, people are more apt to draw out their ideas together for all to see. It's a great way to encourage collaboration.

We combined the Polygal idea with another we discovered at the Stanford d.school. The rolling racks you find near dressing rooms at clothing stores can easily be retrofitted to hold a sheet of Polygal, creating a movable whiteboard. We roll them together to define space for creative exploration, then move them back into place when it's time to build out our ideas.

4. CREATE CONVERGENCE

We also have a common space for lunching and chatting where we all converge throughout the day. The heart of this space is a fancy coffee bar where we craft caffeinated indulgences on a [LaMarzocco Linea](#). Great coffee draws in not only the designers, but folks from throughout the entire company. We see engineers, accountants, support staff, and everyone in between pulling shots at the espresso machine, so there's always an opportunity to find out what other departments are up to.

Convergence points like this keep a company connected, as our senior design researcher [Gregg Bernstein](#) wrote in his recent article in [UX Mag](#). All of these design choices were made to encourage connections and conversations that lead to deeper collaboration. They might seem like frivolous extras, but we've

found that expenditures on anything that encourages convergence quickly pay for themselves.

5. CREATE RETREATS

Of course, there are times when large open spaces filled with inspiring conversations can be a distraction. So we furnished a few offices near the design studio with desks and chairs to give people a place to hole up if they need to have a private meeting, take a phone call, or just work in silence. We want to keep our teams working together in the same space, but let them adjust their work space as needed to be as productive as possible.

Flexible spaces, fuzzy teams

It's not hyperbole to say our new design studio has changed the way we work. Walls divide minds as much as spaces, and when the UX, design and marketing teams worked in separate spaces, we collaborated only occasionally. These days, the lines between teams are so fuzzy that outsiders would have a hard time telling us apart.

Cross-disciplinary discussion is a daily occurrence now. Writers talk to interaction designers, photographers talk to design

researchers. We're all talking to each other and making our best work because of it. We've always known that making the best digital products requires a deep synthesis of skills, but now we're living proof.

COLLABORATION

All Hands on Deck

JUNE LEE

All established cultures have defining moments or rites of passage where current members welcome new members through shared experiences. These initiation rituals foster a deep sense of camaraderie and a dedication to fulfilling shared goals. For many folks here at MailChimp, myself included, that rite of passage was our time spent on our support team.

Our support agents train together for extended periods of time, work closely with mentors, and put in long hours together—kind of like rushing a fraternity, without all the beer-funnelling.

With 7 million users—and with thousands of new customers added per day—MailChimp's support staff is growing all the

time. This has been useful on a few fronts: It means we have readily available customer service, which means happier customers. And it means we have a direct pipeline of experienced, dedicated talent. So when other departments—like UX or partnerships—need to fill a position, we can begin by looking within our support ranks. (This was my path to UX research.)

Recently we saw a rush in the number of customers awaiting a chat or email response from our support team. This made us nervous, as it's important to respond to requests as quickly as possible. But with a flood of new support staff joining our ranks around the same time, we had a chance to introduce them to the MailChimp community by organizing an “all hands on deck” day. All available support agents, along with former chimps now working in different departments, were invited to join together one Saturday to answer customer emails.

It was a work day, but it felt more like a great big family reunion. Former deskmates reunited to achieve a common goal. Breakfast, lunch, and child care were provided, but were considered extras in the minds of the MailChimp employees. All of them gathered to help the company and their colleagues.

As MailChimp grows, it's important for us to retain the community-based values we had when we were smaller. The social aspect of our culture is something we invest in: We host regular company-wide lunches (nacho bar!) and after-work socials, and we have common areas with ping pong tables, pool tables, and snacks.

All of these are fun and cherished parts of our company identity, but there's a special kind of energy generated from an all-hands event. When the queue hit zero at the end of the day, there were high fives all around and “[You're The Best Around](#)” rang through the office. Few things bring people together like working toward a common goal and meeting it.

COLLABORATION

Asking for Help

FERNANDO GODINA

MailChimp is made of people with diverse backgrounds. We have advanced degrees; we're self taught. We're biologists, advertisers, rhetoricians, industrial designers, painters, and poets. We come from all over the globe and speak dozens of languages. These differences are our primary strength. Our diverse perspectives give us an array of knowledge and experiences to pull from when we're collaborating. If there's one thing we have in common, it's that we're all deep thinkers who enjoy trying to solve several problems at once.

These are things I've always known, but a while back I saw first-hand how our diverse skills contribute to a collaborative workflow. The UX team was working together on 3 different projects with looming deadlines. The research team was

working on parallel studies that required sophisticated surveys and emails to thousands of customers. The design and development teams were busy crafting the next issue of our UX Newsletter, and the research team was jumping in to help. We were hustling to get these projects done, but obstacles kept popping up that we couldn't solve on our own. The only way we were going to meet our deadlines was to ask for help.

Passing the baton

While working on a study of our [Mandrill](#) customers, fellow design researcher [Laurissa Wolfram-Hvass](#) and I were struggling to figure out the right way to [inject merge tags into a survey email](#). We needed the merge tags to pass customer information from MailChimp into our SurveyMonkey data. But we couldn't get it to work quite right. Instead of pounding our heads on our desks, we asked some of our fellow researchers to lend a hand. They took a few moments to test the emails for us, and eventually the metadata we needed to pass from MailChimp over to SurveyMonkey was working.

Later, we were ready to send our survey email to thousands of customers when we noticed some coding issues with our template. A few people from our team jumped in to help, but in

the end we had to call in our email developer, [Fabio Carneiro](#), to sort things out. What was difficult for us was mere seconds of work for Fabio. We would have burned precious time had we continued to pound on a problem that was outside our area of expertise. By passing the baton to a colleague with the skills to solve the problem, we kept the project moving and ended up with better results.

Meanwhile, the rest of the UX team was working on [an issue of the UX Newsletter](#), and the articles needed edits before sending. A researcher and a designer dove right in to refine and smooth out a few bumpy narratives. We still shipped with [one mistake](#), but hey, we're only human.

When every member of the UX team contributes in their own way, tasks moved fluidly between us. We have confidence in the abilities of our colleagues, so we know that when a task is passed to a peer, it will get done—and this makes work easy, elegant, and fun.

No islands

As we worked on 3 parallel projects that day, 2 important skills empowered our team: communication and collaboration.

Whenever someone had a question, they would IM other team members to find an answer or walk up to their desks and start a conversation. In return, each question was met with an open and honest response and a willingness to help. Even while working on another task, colleagues made a point to respond in a few minutes—and if they didn't have an answer they'd pass it on to someone who did. No matter the situation, there was never a dead end. We always kept projects moving forward.

Each member of the UX team has their area of expertise, but no one is an island. Of course, this dynamic isn't limited to our team. It happens throughout MailChimp, within and between all teams. Thanks to our multidisciplinary, collaborative approach, we know we can tackle hairy problems concurrently, and that our individual weaknesses are mitigated by our collective strengths. When it comes to making MailChimp better, we never hesitate to ask for help.

COLLABORATION

On Good Terms

GREGG BERNSTEIN

An attorney, a writer, a front-end developer, a designer, and a researcher walk into a room... Sorry, this isn't a bad joke, just the story of a collaborative effort to rethink MailChimp's [Terms of Use](#).

Traditionally, legal documents have been drafted to protect clients and minimize risk, not facilitate ease of use. But that tide is turning. Recently, changes some companies have made to their terms of use have [fomented outrage and confusion](#) among users, more generally pushing agreements towards [clarity and friendliness](#).

At MailChimp, we put a lot of thought into every piece of user-facing content. And in 2013, MailChimp writer/editor [Kate](#)

Kiefer Lee and attorney Valerie Warner Danin agreed it was time to give the same amount of consideration to our Terms of Use. And so a team including UX, legal, and marketing folks began to hash out a plan for legal documents across the MailChimp family, including TinyLetter and Mandrill.

We all brought something different to the table. My goals for user experience focus on clarity and ease of use; meanwhile, Valerie serves to protect the company, and Kate seeks to achieve the proper **voice and tone**. But after working together and questioning everything in our then-current Terms of Use, we created some guidelines to suit everyone: the terms needed to be clear, strike the right tone, cover our legal bases, and provide a considerate user experience.

Here's how we made it work.

Use plain(-ish) language

It was important to all of us that we write the terms as if they were meant for other humans to read—because they are. To be sure, some legal terms, like “represent and warrant,” are so specific and protective that, to ensure their validity, they must

be used. But Valerie and Kate worked to cut all the unnecessary “wherefores” and “hereunders.”

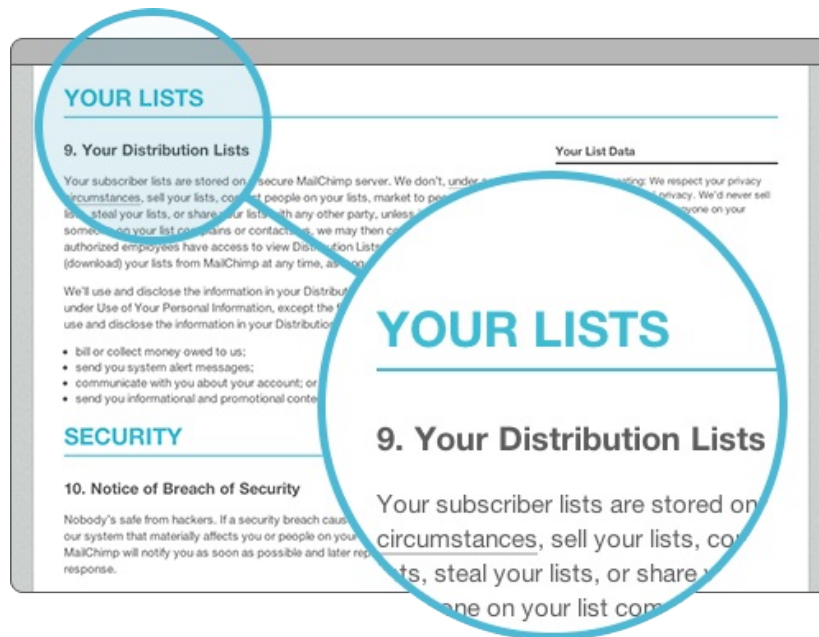
Writing in plain language also meant veering away from the humor sprinkled elsewhere on our site. Legal terms are serious business, and we had to consider the perspective of our customers. If someone is visiting a legal page, they likely want information, not [jokes or silly links](#).

Maintain legibility

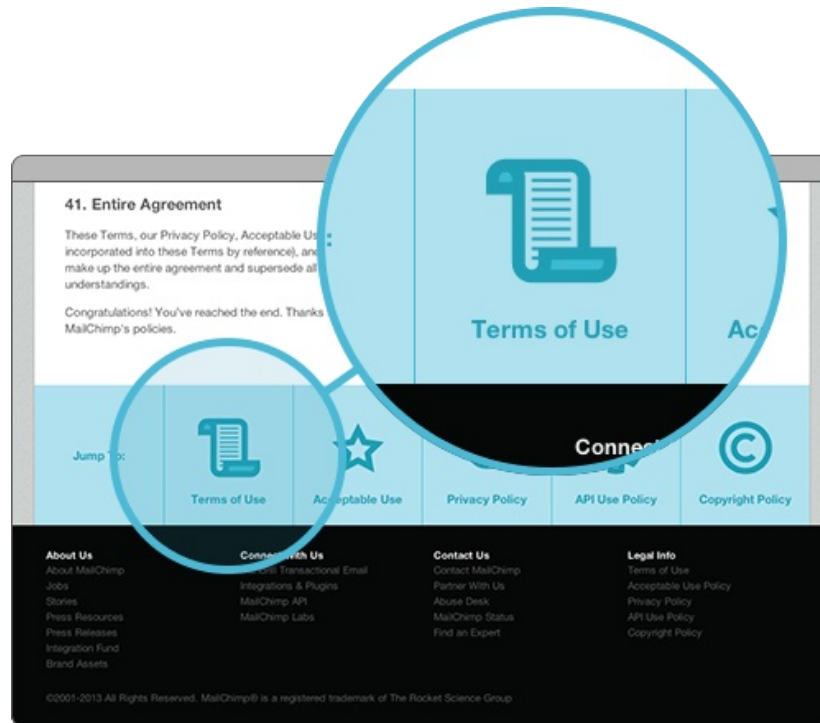
In the interest of avoiding the trappings of typical legalese (and our previous Terms of Use), we refrained from capitalizing entire sentences and paragraphs. By now, we all have a visceral reaction to all-caps emails or tweets (“AGH, STOP YELLING AT ME!”). Surely a legal document can provide a safe haven from such gauche behavior! When we had to emphasize entire sections, we bolded them instead.

We also set our type large enough to be read comfortably, generously spaced paragraphs, and used bullets and numerals to improve readability. This wasn’t rocket science, just good form.

Chunk the content



By organizing the content into intuitive groupings, we imposed hierarchy on the legal documents. Our readers can now refer back to specific sections, rather than searching through an entire body of text. It's one of those things that doesn't add much at first glance, but becomes very handy when a user is faced with an important task—like determining who owns their content. We had already separated our Terms of Use, Privacy Policy, and Copyright Policy, and we further separated our Acceptable Use and API Use policies for ease of reference and navigation, as these are areas where our customers look most often.

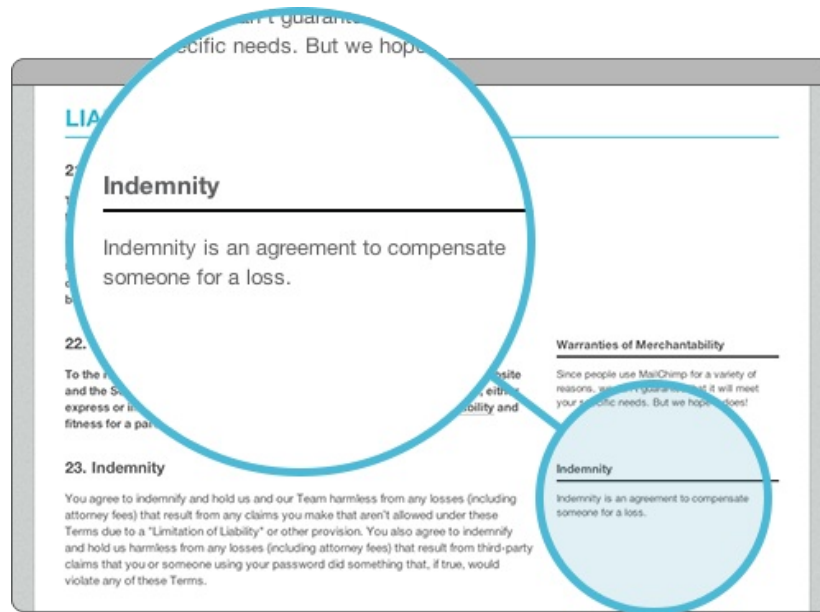


Taking things a bit further, we created an entire Legal landing page, which serves as a hub for all legal-related MailChimp matters.

Lend a helping hand

Even after making our language plainer and our text more legible, we still saw areas that readers might find a bit vague. To remedy this, we provided helper text in the sidebar. The goal of the sidebar isn't to summarize the terms, but to give context for a particular provision. In some cases, the helper text shares a concrete example of what we're talking about, like our [Email Genome Project](#).

In other cases, the helper text explains tricky legal terms.



With those guidelines established, we extended our circle of collaborators to include MailChimp’s art director, [David Sizemore](#), and front-end developer [Steven Sloan](#). David was charged with developing a design system for the different sections of the [Legal landing page](#), which he accomplished with iconography. Steven experimented with various methods of disclosing helper text, ranging from popovers to the sidebar text we ultimately went with.

The number of visitors to a website’s terms of use page will pretty much always be small. But just because less people will see a page doesn’t mean it deserves less care. In the end, our motley, cross-disciplinary team collaborated together to

transform a suite of almost totally unreadable documents into something inviting and accessible, providing our users with a carefully-considered—and illuminating—experience.

RESEARCH

Radicalizing Data

Gregg Bernstein

Sharing Research by Every Means Necessary

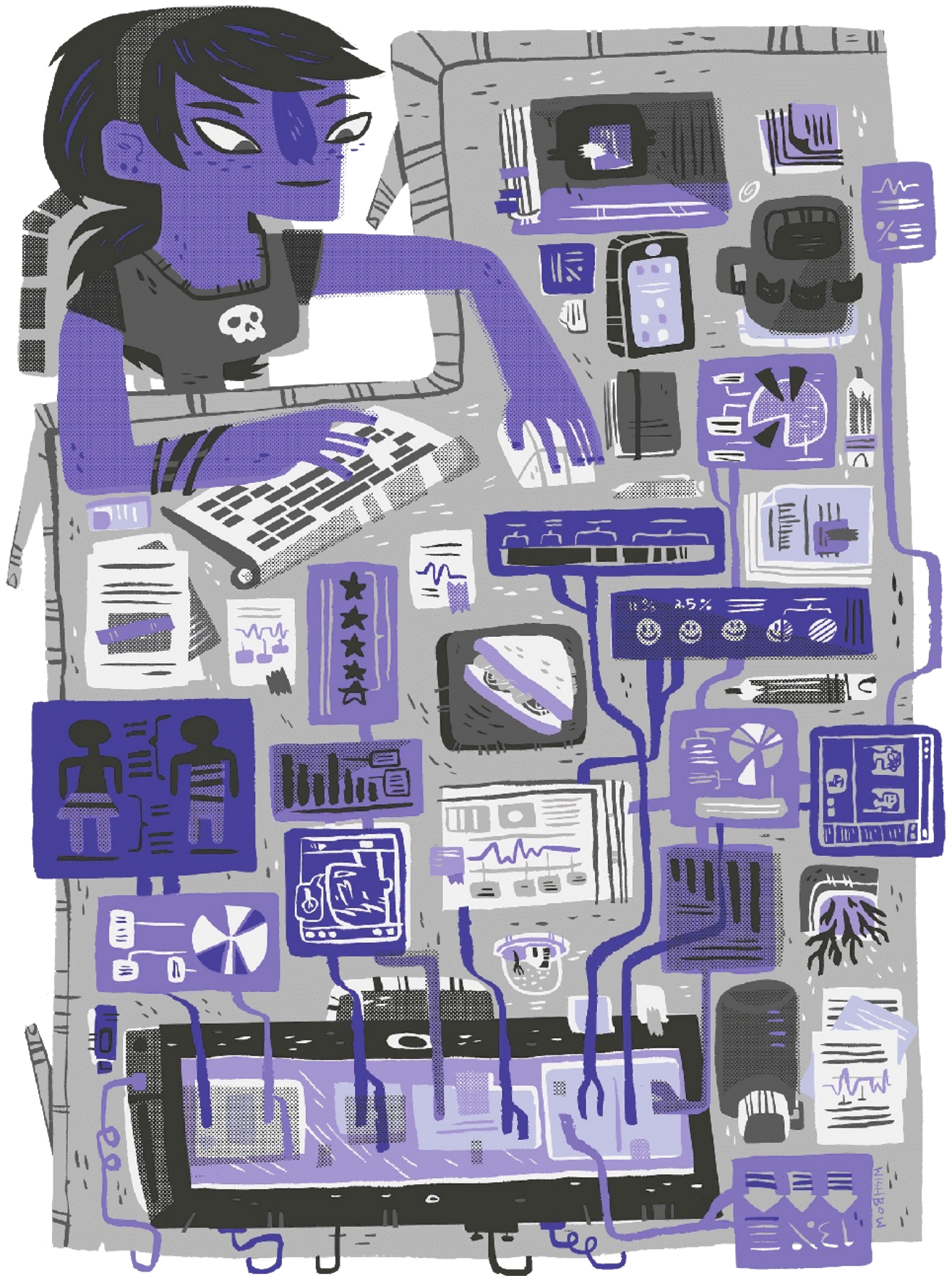
Laurissa Wolfram-Hvass

We Sorted 506,000 Data Points and Lived to Tell

Fernando Godina, Laurissa Wolfram-Hvass

The Open-Minded Interview

Steph Troeth



MICHAEL
W.

RESEARCH

Radicalizing Data

GREGG BERNSTEIN

A popular corollary in design and writing posits that by emphasizing everything, you end up emphasizing nothing. As a researcher at MailChimp, though, I've learned that my team can uncover meaningful and unique insights by flipping that idea on its head. We start with the premise that everything is important, and that every data point tells a story. By listening to all of these stories, then following them until they become epics, we achieve a mastery of our data sets and an ability to focus, prioritize, and emphasize. We've radicalized our data, allowing it to dictate the direction and scope of our research.

Dumb yourself down

To radicalize data is to become subservient to it, no matter the form. At MailChimp, we draw from a number of research channels: unsolicited email feedback, surveys, interviews, usability testing, and analytics, to name a few. By purposefully not making assumptions about what our research should uncover, we empty our minds of preconceived ideas and open ourselves to the data.

It's like we're unleashing our inner [Columbo](#). Embracing befuddlement, we assume that everyone else is an expert and that every sentiment carries weight, and we trust that from this process the patterns and hierarchies will emerge.

We receive emails every day from customers with suggestions and complaints about MailChimp, and we read every single one. Why? Because we assume this data is important and that our customers know far more about using our app than we ever will. If we're building for our customers, who better to learn from than them?

Last year an email came through our app feedback channel that requested a "Notes" function within our subscriber profiles. It was the first and only time we saw this request, but we didn't

discount it outright—it was data, and therefore important. We responded to the customer directly, learned of the use case, and this one piece of feedback turned into an app development. The data we received was minimal, but by granting it legitimacy by default, we learned something new and added a feature with potential benefit for all customers.

We take this same approach with our customer interviews. Once our data points us in one direction—towards the realm of e-commerce, let's say—we visit customers to learn more about the topic and how it affects them individually. We don't barge in and ask, "Can you tell me about e-commerce and how it relates to your use of MailChimp?" Instead, we take the long view and ask about users' typical and atypical days, how they develop their products, or what brought them into their particular line of work.

By focusing on narratives and tangents, we gain insight that lends context to our original data sets. This is analogous to the "switch" methodology espoused by [The ReWired Group](#): Customers hire products to perform a task, and our job as researchers is to uncover everything that influenced that choice. When we attribute expert status to our customers, we

open ourselves up to empathizing with their situations and rationales.

Then wise up

Of course, for every piece of data you follow, there always comes an end point. We complete our research mantra by claiming that each data point holds value until proven otherwise. Once we see enough evidence to contradict the data in question, we move on.

Here's an example: A while back, through our feedback channels, we saw data that seemed to suggest it was too easy to inadvertently unsubscribe from a mailing list with errant clicks. But once we looked more closely at the data (examining who provided it, the frequency and the rationale) we determined that this wasn't necessarily a point worthy of further research and development. We could empathize with the feedback, but the use cases in question did not add up to a development that would improve our app—instead, it would appease just a few users and possibly upset even more.

By radicalizing our data, every data point comes to represent continuing education about our products and customers. With

this principle firmly in mind, we can follow every lead and know that, even if proven unimportant, our research time was well spent.

RESEARCH

Sharing Research by Every Means Necessary

LAURISSA WOLFRAM-HVASS

Good research takes time and effort! Our research team spends days setting up tests, coming up with survey questions and interview prompts, tweaking protocols, scheduling participants, and running pilots. Research studies can take days or weeks to complete—and let's not forget all the time it takes to re-watch video footage, transcribe interviews, filter through responses, conduct follow-up interviews, and pull out meaningful statistics.

But what happens after that? How do we move from the reflections of research to the actions of development? How do

we transform all of our hard-sought insights into accessible knowledge we can use to improve our products or services?

No matter how methodologically sound, relevant, or interesting it may be, user research is essentially worthless if it's not used to influence improvements to our products or services. One of the most important parts of a researcher's job is making sure findings get to the people who need them to make decisions. This means translating findings and sharing them with designers, developers, marketers, writers, and support staff—anyone involved in constructing and influencing our users' experiences.

A research report is the most common way to share findings and insights. Reports have their place, but they tend to be long, dense, and usually require more time to read (and write) than most people are willing to commit. So here at MailChimp, we've been experimenting with other ways to share research with the company beyond printed reports. There's no perfect communication method, of course. But each of the channels we use serves a different purpose.

Posters

In 2013, we made [persona posters](#). The UX team spent months developing a group of “personas” that represented our primary customer types. We wrote a formal report that discussed each persona in-depth, and we had our designers create posters for each, with descriptive words that summarized each customer type.

We hung the posters in a common area so that they might spark conversation and remind people to consider the customers we serve. During our major redesign last year, we taped pages under each poster with verbatim feedback from customers who fit within that persona type. That direct feedback, paired with its corresponding persona, served as a visual reminder of our users and their needs.

By nature, posters can't offer very deep understanding, but they can provide a snapshot—a chunk of information that can be digested quickly.



Coffee Hours and Lunch & Learns

On Friday mornings, we have company-wide [Coffee Hours](#) featuring speakers who talk about things like the creative process, journalism, security, comedy, and design.

Occasionally, the research team takes the floor and shares some of the insights we've discovered about our users. Usually these talks focus on big trends and themes that affect our entire customer base—which, in turn, affect our entire company.

Information from Coffee Hours is usually available to our colleagues in other places, like a Google Drive report or [in Evernote, where a lot of our research is housed](#). The in-person presentation, though, is often more dynamic and engaging than a report—and more likely to reach a wider audience of people.

Lunch & Learns are less formal than Coffee Hours and usually have fewer than 20 people in attendance. The research team has used this time to invite designers and developers to watch usability testing footage with us. These lunches are a chance for us to get the designers, developers, and researchers together in a relaxed setting where we can chat about design challenges and collectively come up with solutions.

Coffee Hours and Lunch & Learns are more visual and dynamic ways of presenting research. They cater to folks who are more engaged when someone speaks to them directly and they have the opportunity to ask questions or respond.

Internal research newsletters

On a semi-regular basis, the research team uses MailChimp to send an internal newsletter that summarizes our current work. Usually it's a collection of interesting stats and numbers on a

particular topic (like international growth rates, integrations usage, or certain customer behaviors) paired with a brief explanation or analysis. It's information that's not really enough for a formal presentation but is still important to share with the company. We also encourage other departments to use this as a channel for research or insights they've gathered. And we always include links or directions on where to find additional information, if the reader wants to learn more.

Mini-documentaries

On occasion we've made short, 5 minute documentary videos that give the whole company a glimpse at our customers' everyday working lives. Seeing customers' faces, their office environments, and the technology they use helps us develop empathy. Listening to customers talk about their organizations, how MailChimp fits into their overall workflows, and what their daily struggles are like has more impact than simply reading a quote on a printed page. Plus, videos can be watched anytime, paused, and replayed.

Google Drive reports

Though we try not to rely on reports as our sole communication method, we do still use them. Google Drive helps us make reports a little more collaborative, since they're easy to share and the comments feature facilitates asynchronous discussion. Reports are usually read by just a fraction of the company; they're first shared with leads, who then might pass them along to someone on their team if the research directly pertains to a current project. A couple of designers and developers have told us that reports have become a handy reference for them when they're working on specific projects. These reports are packed with important information, so they do require a greater time commitment and focused concentration. They are rich with detail, and they can be sat with, re-read, and digested slowly.

Research evangelists

Each of those communication forms has its place, and the MailChimp UX research team is learning that the best way to reach all of our colleagues is to repeat information in different ways through multiple channels. An unspoken part of our jobs as researchers at MailChimp is just “being around” and promoting research whenever an opportunity presents itself. We keep ourselves open to those serendipitous moments when

we can contribute research tidbits to a conversation we're already a part of. This doesn't mean monopolizing conversations and droning on and on about a current study—nobody wants to be around that person. It just means being aware of when our research might inform the work of others.

Here at MailChimp, we try to keep the walls between teams low, so ideas can move and flow freely throughout the company. As researchers, we work across many different groups and can see how the work of one team might impact another. So, when we're talking to colleagues, we're in a great position to not only share information, but foster connections. We love being able to say things like, "Oh, you're working on a feature that affects e-commerce customers? The e-commerce brand manager just met with several customers last week who mentioned some pretty interesting struggles with that feature. She'd be a great person to talk to."

Keeping research fresh and minds curious

Right now, we're tossing around new ideas for how to keep things fresh and share research effectively. We've worked with our designers to make internal, IP-restricted websites for reports that are more dynamic and visually interesting than

formal, long-form documents. There's also been talk of collaborating with our marketing team to turn some of our research into comics that can be shared internally.

Ultimately, we want to spark conversation, get people interested, and inspire curiosity about our users and how we can create better products for them. We have teams of smart, talented people working with us at MailChimp, and as researchers, we want to empower them with information so they can make better decisions about their work.

We Sorted 506,000 Data Points and Lived to Tell

FERNANDO GODINA, LAURISSA WOLFRAM-HVASS

Customer feedback is very important to our work at MailChimp —it helps guide our decisions and improve our product and process. In early 2013, we tried something new (for us) and sent out a massive, 46-question survey. After one week, we had received responses from more than 11,000 customers. That's 506,000 pieces of feedback! And we had to go through them all.

Fernando: Sorting the results

The answers to the multiple choice questions were fairly easy to analyze. [SurveyMonkey](#) has some nifty tools that let us filter the data based on responses to specific questions, so we could

chart mobile device usage based on size of company or see which industries reported collaborating in teams the most.

We also asked our chief data scientist, [John Foreman](#), to do some data digging to find valuable insights. He shared with us the magic of Excel pivot tables, but that's a story for another day.

The thousands of unique answers to open-ended questions were a bit harder to comb through. We approached this feedback like a puzzle: Our job was to put the pieces together to find bigger patterns. Instead of thinking about the responses individually, we read through them looking for commonalities and then categorized them. Instead of having 1,966 unique answers to one question, we ended up with 14 buckets, each expressing a certain type of answer.

Bucketing is a simple idea, but making it happen takes time and concentration. Once we were done, we were able to rank-order buckets and find answers to our open-ended questions. This helped us quantify feedback patterns and share that information internally.

Laurissa: Finding meaning in the mess

Tagging and categorizing responses helps us see trends much more clearly than if we try to identify them by reading responses in isolation. But, as Fernando mentioned, this can be challenging when you have thousands of responses to filter. How do we dig through all of that data to find meaning?

While there are many different approaches to analyzing open ended survey responses, I'm going to share a few tips for how I move this process along using SurveyMonkey.

FIGURE OUT WHAT YOU REALLY WANT TO LEARN

Before you read the very first response, remind yourself why you conducted this survey. You'll look at your responses very differently if your goal is, for example, understanding how many of your customers express frustration about a particular feature, versus identifying commonly used workflows or features. Regularly refreshing yourself on what you're trying to learn will help you stay focused and move through the data more quickly.

FILTER AND CATEGORIZE

Like Fernando said, open-response questions are a great way to gather in-depth answers—but when you have several hundred of them, they can be a little tricky to dig through.

I'm one of those people who can't wait to see what our customers have to say, so I usually start by skimming through the responses, just to ease my curiosity. But once I've gotten a feel for what people are saying, my first priority is to filter out all the responses that aren't particularly helpful. For example, I usually get a big handful of "n/a"s from folks who either have nothing to say or just want to skip to the next question. I do a quick search for "n/a" and all of its variations (none, na, n.a., nada, nope) and categorize them in SurveyMonkey with a tag, something like "Unanswered" or "Nothing."

After those are cleaned out, I start skimming through the responses for obvious patterns and keywords that I can use as search terms. Depending on what I need to learn, this can be a MailChimp feature (template, user profile, autoresponder), common task (importing, segmenting, syncing), or software (Excel, EventBrite, CRMs). SurveyMonkey does all the hard work for me by searching for responses with these keywords, listing all the responses together, and highlighting the search

term in each. From there, I can select multiple responses and categorize them all at once.

SurveyMonkey's text analysis tab can also identify common themes or patterns—it pulls out the most common words used across responses and displays them as a word cloud. These words aren't always very useful, but sometimes text analysis uncovers a few gems.

I can usually categorize several hundred responses fairly quickly by using SurveyMonkey's search and text analysis tools. Once I've sorted and categorized all of these responses, I begin the more tedious task of reading through and categorizing everything that's left.

REVIEW AND ADJUST CATEGORIES

Once I have all of my responses tagged, I review the responses in each category. Sometimes I find that a category is too broad, so I split it into multiple smaller categories. I also might move responses from one category to another that seems more appropriate—for example, I might have initially identified a response as a “UI Design” problem, when it actually belonged in the category for “Content/Copy.” There's always some

shuffling that goes on before I feel like everything is categorized accurately.

IDENTIFY PROSPECTIVE INTERVIEWEES

Surveys are great for helping us understand what a group of users might be struggling with, but they don't always help us understand why they're struggling. To learn more, our research team usually follows surveys with a round or 2 of customer interviews. As I'm going through survey responses, I try to tag people I want to reach out to later. Usually I'm looking for an interesting problem or scenario that I want to know more about. Flagging potential interviewees while I'm sorting data saves me from having to search for them again later.

KEEP TRACK OF OUTLIERS

Surveys can help us see overall trends across large groups of users, but they're also good at drawing attention to users who really stand out from the crowd. It's particularly useful to flag these "outliers" and reach out to them individually for a follow up conversation. Our UX team is always on the lookout for interesting use cases, for people who are pushing the limits of

our app or employing interesting workarounds. These are often people we can learn from and are well worth the time to chat.

LEARN TO LET GO AND MOVE ON

This is a hard one. I used to feel like a negligent researcher when I didn't explore every single pattern, trend, or finding that came up in my surveys. But eventually I realized that following all those leads didn't make me a better researcher—it only made me a busier researcher.

If you come across something that's interesting but outside the scope of your current research goals, great! Start a running list of “things to explore” and come back to it later.

Parting thoughts

Your job doesn't end once you've filtered all of your responses into categories. Sure, you might be able to say “22.31% of respondents mention they use mobile tools,” but that statistic doesn't tell you who these users are, which mobile tools they use, the tasks they are trying to accomplish with these tools, or why they prefer to work with a mobile device. It's our job as researchers to draw meaning from this data by putting it into

context or digging deeper for more information. Will organizing and categorizing data do this for us? No, but it does give us a very good place to start.

The Open-Minded Interview

STEPH TROETH

There are more ways to conduct a design research interview than to cook a potato: over the phone, remotely through online meeting software, face-to-face in any variety of social settings —sterile meeting room, a noisy office, a bustling café... and the list goes on.

And that's just the setting. What about the questions to ask? Should we be structured in our interviews? Would it make our data easier to analyze? That would be the data-serving approach. However, what we look for in design research is naturally complex. To get closer to the true picture of our users' contexts and the motivations that lead to their behavior and patterns of use, we ought to be prepared for messy, unexpected situations and data we can't predict.

At MailChimp, when the research team started looking beyond product features and interface issues to the stories behind these problems, we began to refine our interviewing skills. Now we aim to be focused yet opportunistic in what we might learn. We want to gather consistent enough data that enables us to do meaningful comparison across interviews and other data, but also allow for unexpected surprises.

Before the interview

As researchers, our work begins long before we're sitting across from an interviewee. Here are steps we try to take and things we think about before we even begin those conversations.

ASK, "WHAT DO WE WANT TO LEARN?"

Walking into an interview with some clarity about what we want to learn is crucial to ensuring we'll come out the other end with usable data. For a set of interviews, we want to identify our research objectives. Are we trying to profile a type of customer more profoundly, or are we trying to compare various customers' workflows? Are we just trying to explore the

environment a particular set of customers operate in within a particular industry?

Regardless of the type of interview we conduct, it's good to have a handy checklist of "areas of knowledge." This is the baseline of what we want to learn from a set of interviews and serves as a guide to keep us from straying too far down rabbit holes.

Sometimes we write these as a list; other times they're more naturally expressed as a set of questions. Giving some upfront thought to the angle and the desired outcomes helps us straighten out how we want to approach any given interview, and it irons out any potential tricky nuances. I liken this to a mental rehearsal of how we imagine an interview might play out.

Depending on your customers, establishing what you want to learn may also influence who you recruit to interview—what kind of customers (or even non-customers) might be best suited to answer your questions?

ESTABLISH RAPPORT DURING RECRUITING

There are at least 2 parts to recruiting: selecting potential interviewees and actually contacting them. How we select

interviewees truly depends on what we intend to study, and how we interact with our interviewees from the outset can help bridge any awkwardness that might arise in the unnatural social setting that is the interview.

When I'll be doing an interview in person, I find it helpful to reach out directly to the user I'll be talking with. This lets me begin our conversation before we meet, and it allows me to put some expectations in place—already, in such a situation, we are setting ourselves up to be trusted.

During interviews

A rehearsal is fundamentally different from a performance. When you're sitting across from the user, it's the real thing. There's a bunch of great advice out there on how we should come into interviews with guidelines—but don't be afraid to stray from it.

With my ever-faithful interview guidelines as a security blanket, I prefer to approach interviews from a more open standpoint and focus as necessary.

SET UP THE ATMOSPHERE

Some interviewees like small talk, some do not. I often let them guide me on how much they want to chatter, and try and establish a comfortable middle. As you are getting to know one another, this is usually the best time to tell your interviewee what you are interested in learning (I keep this bit short and general, so as to leave it open to interpretation). Also, this is the time to sort out formalities, like getting those release forms signed and signaling how you intend to record the interview so that your interviewee is forewarned.

Often, I may prompt them to talk broadly about themselves, and let the interviewee hone in on issues most important to them. Already, I set the expectation that I'm here to listen to their stories. By keeping the conversation open, I can let the user take the lead on what they want to say. If we struggle to get started, I might ask them to tell me a story. Usually, things tend to just flow from there.

ALLOW FOR BEING HUMAN

I lean towards a loose, unstructured format because human memory is associative and, as people recall situations, they need to jump around a little. It's through going back over the same ground several times that we are usually able to uncover

deeper insights. More formal, structured interviews have always seemed strange to me, the way they can treat our fellow human beings as “black boxes” through a mechanistic, question-answer format. If you keep repeating things in the same sequence, your interview risks becoming more of an interrogation. By spreading the depth of questioning out a little, you have far better control over the tone of the conversation without being too confrontational.

ASK YOURSELF, “WHAT’S THE STORY?”

When I first started running interviews, I tried to remember all the types of questions I had to ask. Then I simplified my approach: just be naturally curious. That’s all there is to it.

This has mostly worked well for me out in the field, but being ever pragmatic, I set about looking for a framework. Eventually, I realized that if we treat every interview as a story that we want to retell, we can quite easily get at the core facts and the nuances. By establishing key characters (who are the main people you work with?), timelines (what happens when?), severity of struggles (how much? how frequent?), and emotional impact, we can extract a lot of information in a short amount of time.

KEEP A CONVERSATION CHECKLIST

Interviewees will throw details at you that are out of chronological order—it's your job as the interviewer to make sense of the story. I usually use a conversation checklist to keep myself on track. As the conversation goes on, I keep an eye on my checklist to make sure everything is covered. If it's not, I'll find a convenient way to segue into something that might be on my list, or go back to a point in their story for a point of clarification.

The other tool I sometimes use is what I called the “conversation stack”—in my notes, I keep track of topics that the interviewee may have touched upon that I want to dive back into later. This allows me not to interrupt the flow too much.

As we close up the interview, I will go over my notes and double check some points to make sure my impressions are accurate and complete.

When it's over

It can be difficult to immediately process the content of an unstructured interview, but some reflection can be helpful. The best way to deal with the data is to write down the key points

that you remember immediately after the interview. When we interview in teams, we allow a little time between interviews to put together a quick summary of what jumped out at us, and trends we found most significant. Our memories are great filtering tools in their own right—why not exploit that property of our minds?

After more interviews have been conducted, and after we've had a bit more time to reflect, we also do periodic reviews of sets of past conversations. If too few interviews were conducted, there is a strong danger of bias, so obviously this doesn't replace the more in-depth analysis. However, over time, key takeaways from interviews distilled in this way become a body of knowledge in its own right.

Also, maybe I'm a little old-fashioned, but I always write a thank-you email to an interviewee after our conversation, leaving an open invitation for any further comments or questions. After all, a customer's story begins and ends beyond the limits of their interaction with us, and certainly beyond the timeframe of an interview. It seems fitting to keep conversation lines open—there might always be more to say, or another occasion to meet.

DESIGN

Why You Should Sketch, And How

Federico Holgado

Evolution of a Pattern Library

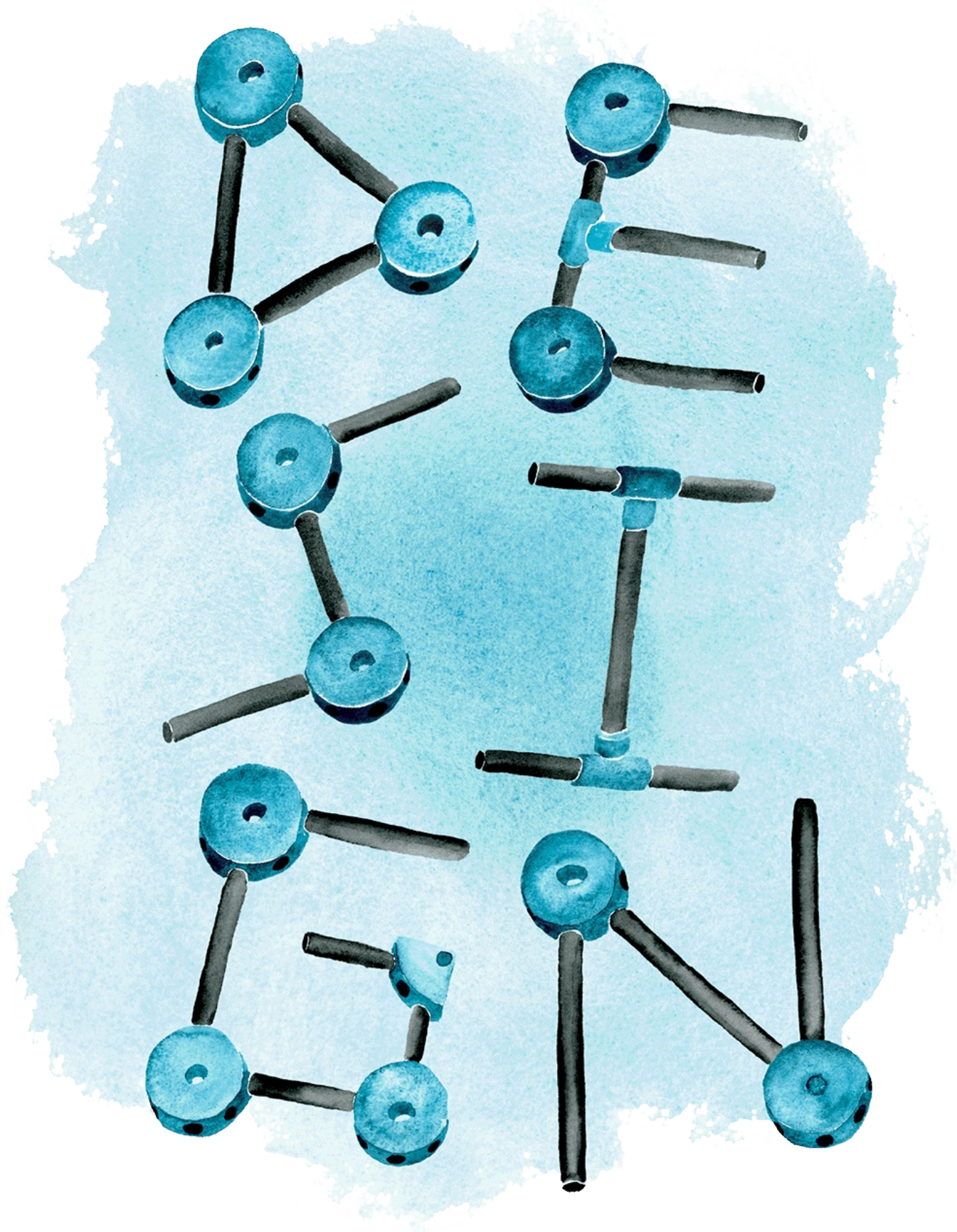
Jason Beard

Building a Better Pattern Library

Federico Holgado

High Five for SVGs

Caleb Andrews, Alvaro Sanchez



Why You Should Sketch, And How

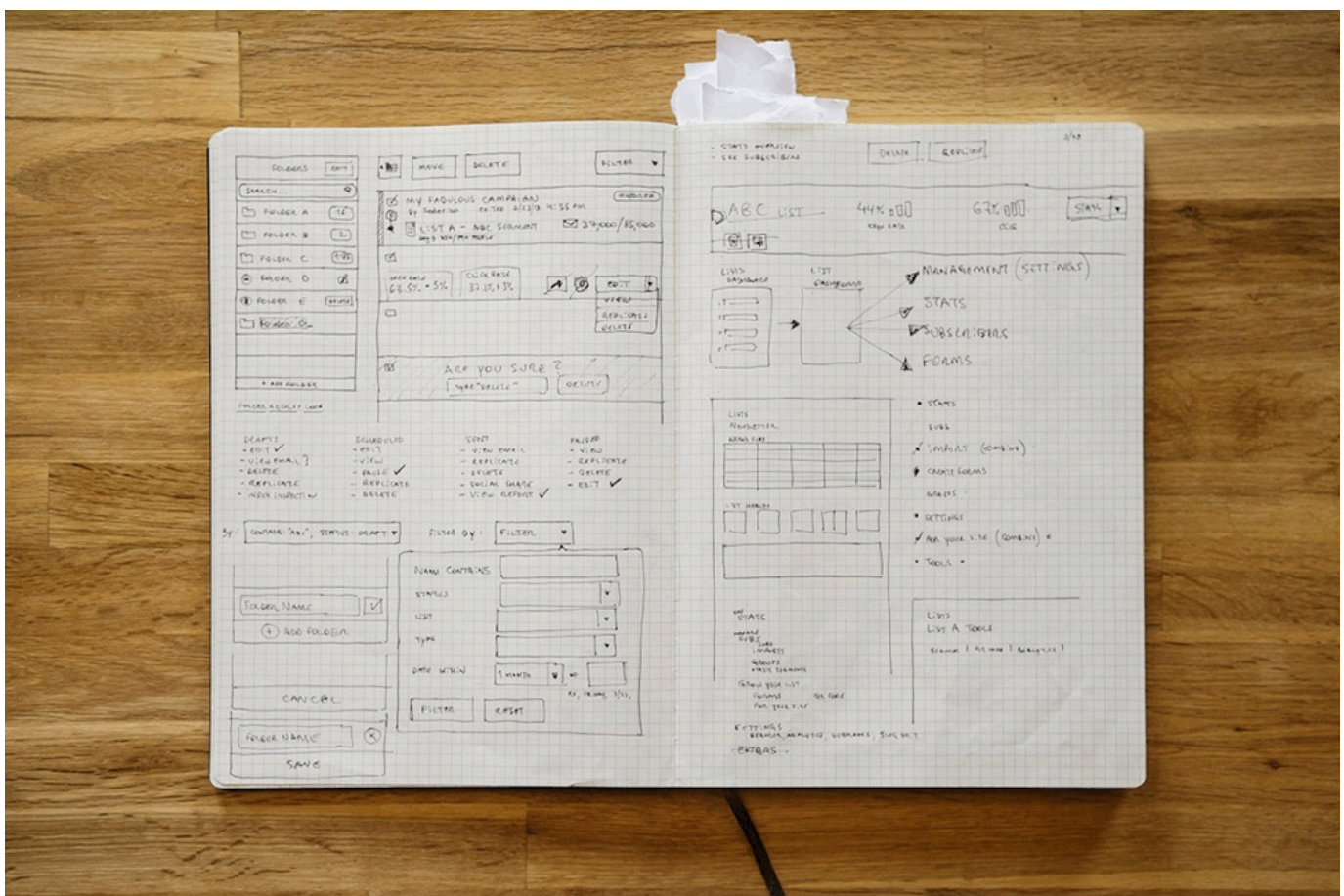
FEDERICO HOLGADO

Sketches are the building blocks of the language designers use to communicate. If you see me walking around MailChimp HQ, you probably also see a sketchbook or an iPad in my hand. I knew sketching was important before I came to MailChimp, but over the last few years I've been reminded of that fact again and again. Plus, I've learned a few tricks that have saved me countless hours of mucking around in Photoshop or wrestling with CSS to explore UX and UI ideas. Here's what works for me.

More sketching, less hand waving

I studied industrial design in [school](#), and sketching is a big deal in those classes. We had the idea drilled into our heads that if you have to use words or hand gestures to describe a feature,

your rendering or physical model isn't good enough. Same for when you're talking about a UI feature—it's much easier to bust out a quick sketch to show your team exactly what you're thinking about. And when it's easier for them to understand your thoughts, they can more quickly jump in and make edits or ask questions.



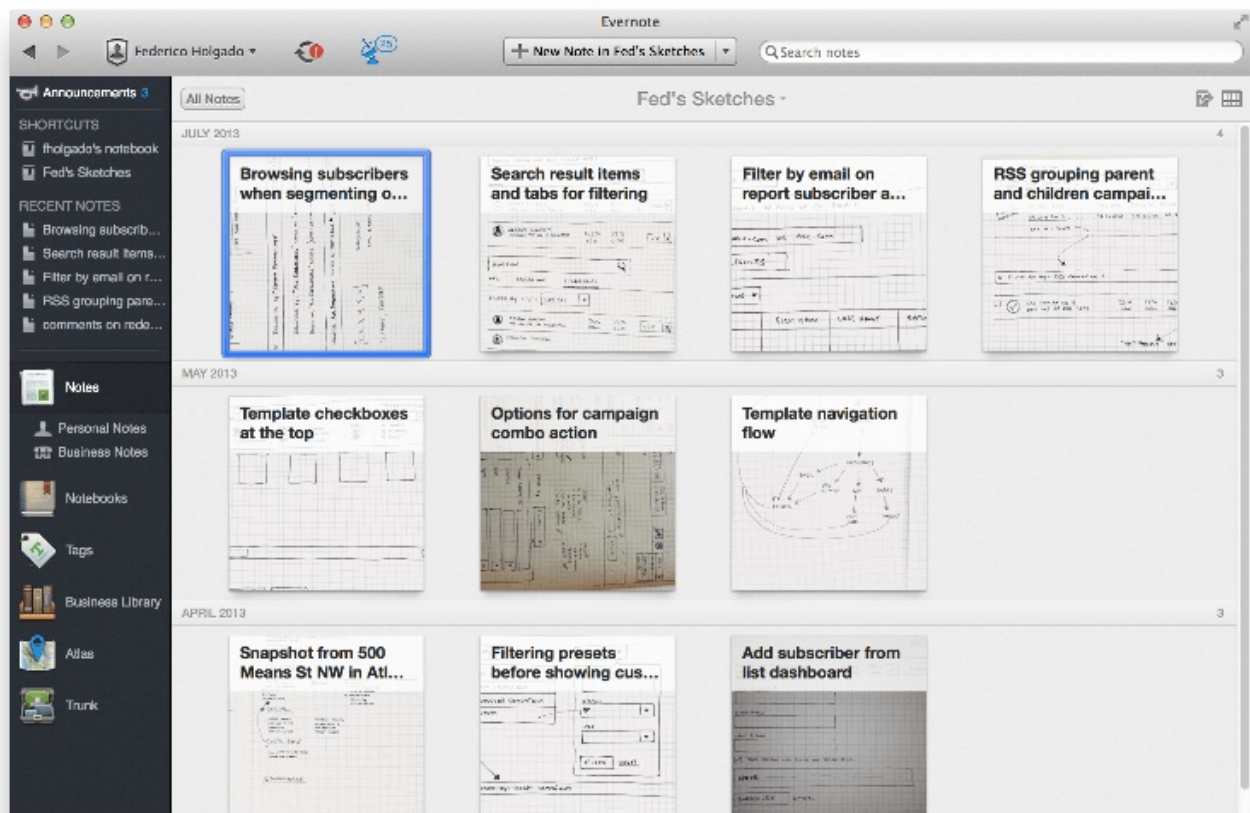
Have a purpose

Another thing I learned very early is that there's a process of refinement with sketching. In the early stages, you'll produce

tens or hundreds of very quick sketches to explore forms and ideas. Once you start to clarify the direction you want to take, you can start filling in more of the pieces. Many times, my sketches focus on big things and leave out the smaller details. You'll see squiggles for text and rectangles that represent buttons and images. But your sketches don't have to be perfect or beautiful—they just need to get your idea across.

Sharing is caring

At MailChimp, I work very closely with our UI designers [Tyrick Christian](#) and [Caleb Andrews](#), and we've figured out a quick process for producing beautiful feature comps in Photoshop. We start by sitting down and talking through concepts. We'll usually sketch as we talk, and generally get a polished idea in 5 to 10 sketches. Next I'll photograph those sketches with my iPhone and upload them to [Evernote](#). There, it's easy to add notes and discuss them as a team. When Tyrick moves on to create a more detailed comp in Photoshop, he has a nice blueprint to work from filled with notes and examples about the interactions to be designed.



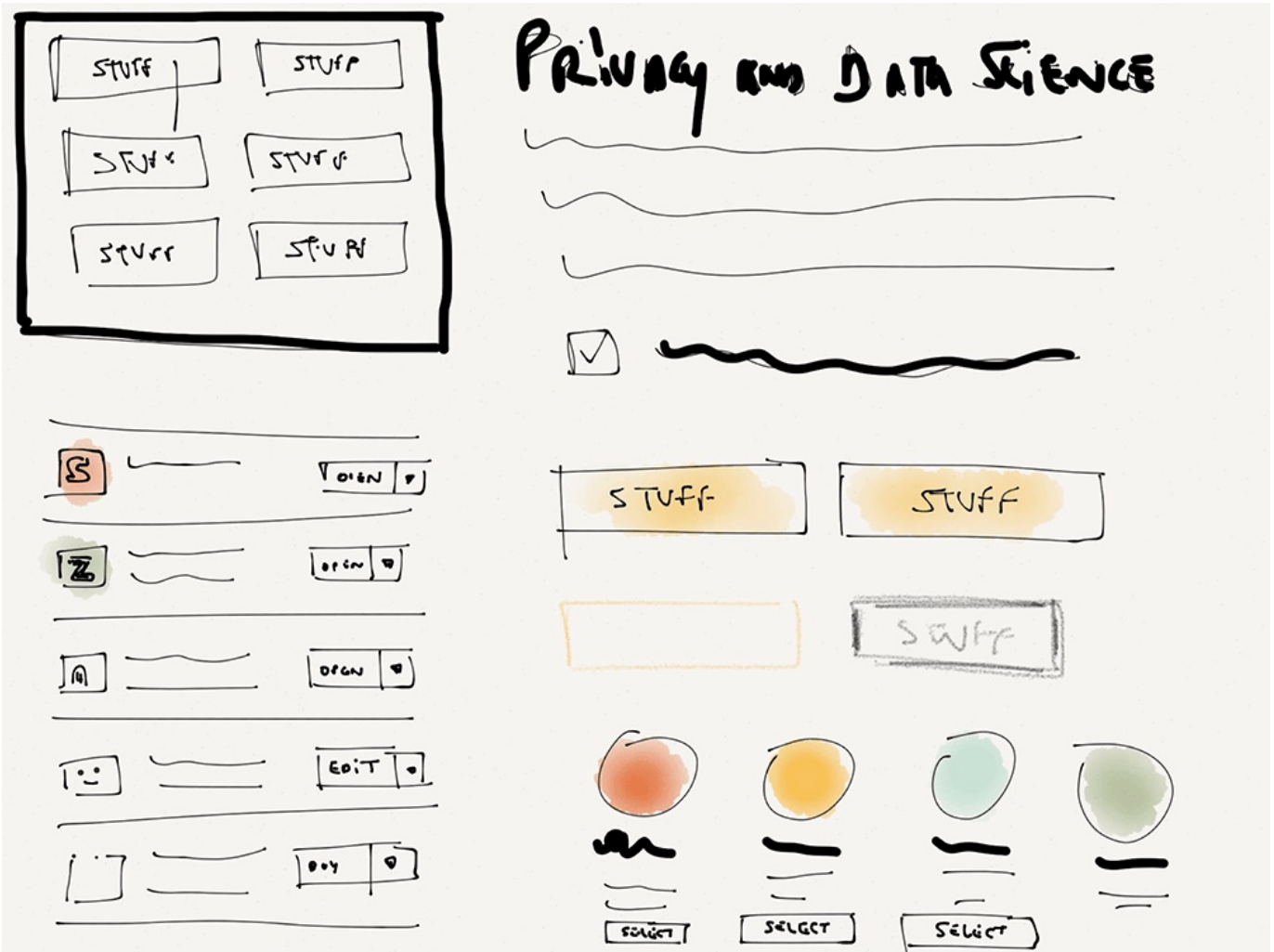
Select your tools

My notebook of choice is a [large Moleskin journal with squared paper](#). This is the largest Moleskine notebook I could find, and the extra real estate helps when I'm working on more detailed sketches. The squared pages also keep things in proportion, since sketching software usually involves drawing boxes upon boxes of stuff. Using a notebook also helps keep my sketches in one place, which makes it easier to refer to previous projects. I am constantly looking back at sketches from previous months,

and having a bunch of loose sheets would make this difficult if not impossible. (Related: I'm kind of a spaz.)

I use a [Pilot Razor Point](#) pen. It's a felt-tip pen, which makes it less prone to drying. The line weight is thin enough to write in small letters, but still dark and consistent from stroke to stroke. I buy them by the box.

Of course, pen and paper have their limitations. Snapping photos of your sketches and sharing them digitally can be a drag, and the photographed sketches often lose some contrast and detail. Not so on an iPad, though. My current app of choice is [Paper](#). I also use a Retina iPad (not an iPad Mini) because I want to sketch over the largest possible area. The Adonit Jot Pro is the most accurate stylus I've tried on the iPad, almost as accurate as a normal pen. And sharing sketches with Paper and adding them to Evernote is a breeze.



First sketching experiment on Paper for iOS.

Pros and cons

When I sketch on paper, I use a single pen and no color whatsoever. With the Paper app, adding color and different line weights to a sketch is simple, and the results are beautiful. It's certainly a different aesthetic than a hand-drawn sketch, but the fine-line pen and watercolors make a killer combo. Also,

the sketches maintain a lo-fi feel to them, which tells people they're looking at ideas, not finished prototypes.

There are some minor disadvantages to digital sketching, though. So far, I've found it to be a little slower than traditional paper sketching. Using a stylus can be awkward at first, though it gets easier with practice. Sometimes I trigger unwanted actions, like turning the page or bringing up the iOS notification center. Also, having the power to undo a line or stroke means that I end up being more nitpicky about getting things right.

The future of sketching

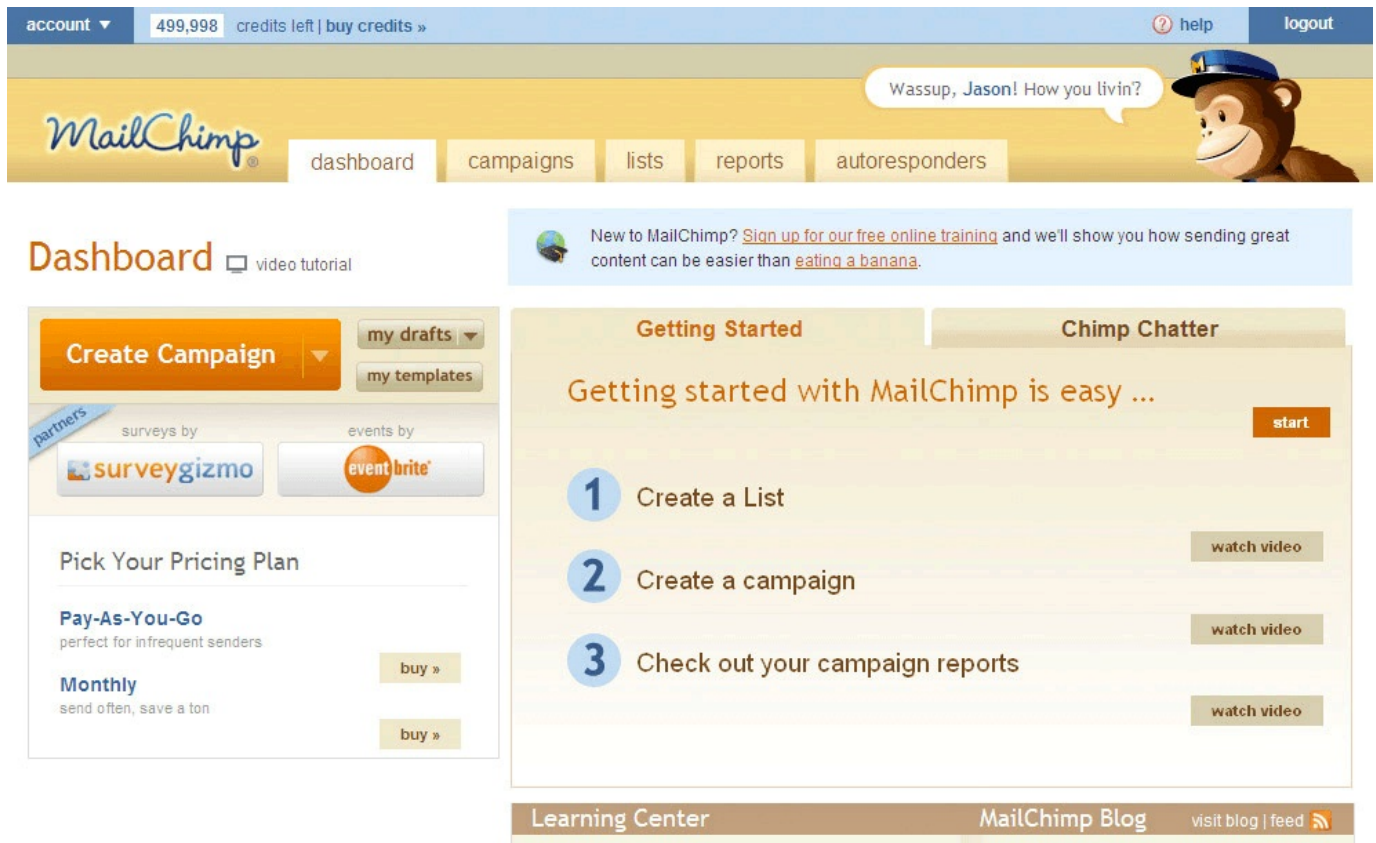
My foray into iPad sketching with Paper is only a couple of weeks old, but I'm already loving it. I'll still do a quick sketch here and there in my Moleskin, but unlimited pages, better looking sketches, and easy sharing has really won me over.

DESIGN

Evolution of a Pattern Library

JASON BEAIRD

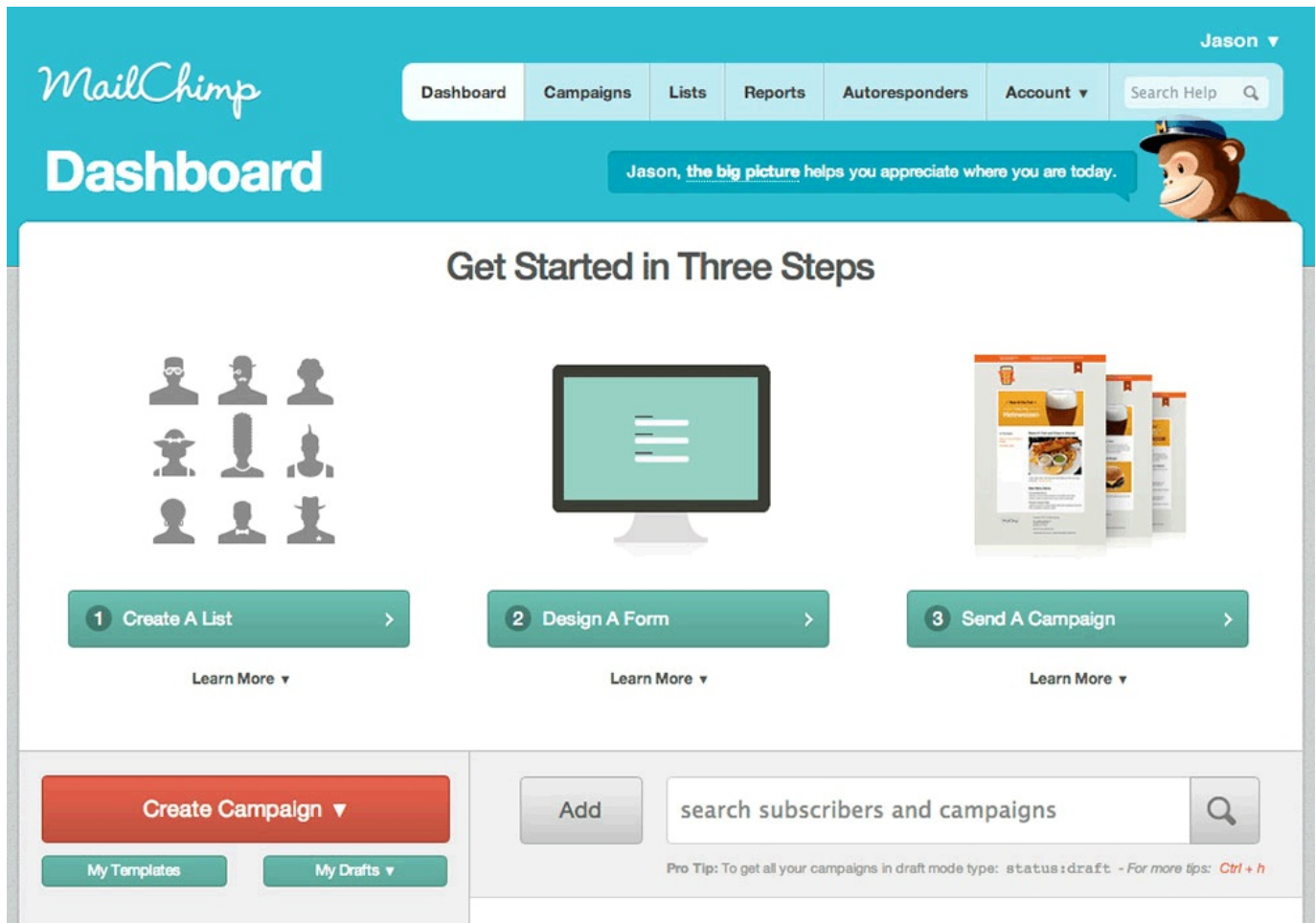
Life in the web industry sometimes feels like it's measured in dog years. While time does fly when you're having fun, the trends, techniques, and technologies change so fast that a 2-year-old website or application design can feel like it's a decade old. When I started at MailChimp in early 2010, this is what our application dashboard looked like:



Up until that point, the markup and styles for every component of the application had been (quite lovingly!) coded from scratch. We used common classes for things like buttons and grids, but the dated auburn tones, font styles, and gradients in the design above were baked in all over the CSS. In short, it was far from what our friend Brad Frost would call [Atomic Design](#). So when we started talking about a full redesign in mid-2010, we knew things had to change.

We started working months before we ever saw a single screenshot of what the design team was conjuring up for MailChimp's new look. Our primary focus for this "pre-design"

was to reduce repetition in our CSS to make the actual redesign process easier. We did this by combining similar interface elements into reusable patterns. The metric we used to gauge our success was total CSS size, and over the 6 months before we released the [2011 redesign](#), we managed to cut about 120k from our CSS. Here's what the dashboard looked like after we launched in February 2011:



As we were creating these reusable patterns, we started archiving them in an internal “cheat sheet” for a couple reasons. First, we wanted to create a style guide that anyone writing front-end code for the app could use as a shortcut. Second, we also wanted a test page to show whether or not the changes we made to our pattern CSS would break things in the app.

Fast forward to November 2012...

Knowing that we were going to be embarking on another big redesign, we started crafting a brand new pattern library. We already had a very modular approach to working on the app, so our focus the second time around was making things responsive, more user friendly, and even more flexible. As with the previous redesign, our team was working for several months before the actual design was settled on. Here's how it looked when we launched in 2013:

The image displays two side-by-side screenshots of the MailChimp dashboard interface from 2013. Both screenshots show a user profile for 'Jason' and a search icon. The left screenshot is titled 'Dashboard' and features a 'Recent Campaigns' section for 'MailChimp UX Issue 15 - Responsive Email'. It displays metrics: 1,840 Opens (56.9%), 345 Clicks (10.7%), List avg (51.5%), and Industry avg (14.1%). Below this is a 'List Growth' chart and a 'Top 5' section listing 'MailChimp UX in San Francisco' with an open rate of 76.5% and 62 opens. The right screenshot is also titled 'Dashboard' and shows the same campaign details in a more compact layout, including 'Recipients 3,237', 'List MailChimp UX Newsletter', and 'Delivered 10/22/13 2:41PM'. It also displays the 1,840 Opens (56.9%) metric and a simplified table for List avg (51.5%) and Industry avg (14.1%).

By the time we launched [New MailChimp](#), we had a much more extensive collection of common interface patterns. Unlike our old pattern library, though, we made this one public. We didn't

make a big deal about it, but it's been available [online](#) since our most recent redesign launched in 2013.

Now, you may be asking: why would we publish something like this when it probably won't benefit anyone outside of MailChimp? Well, for the same reason we do our UX Newsletter. We want to be transparent, share our challenges, and show our work. As with just about everything on the web, our pattern library was inspired by the struggle of many brilliant people, and we hope it'll inspire others as well. It's not perfect, and you might even find it broken at times, but it's served us well for the last year—which is really like 7 in web industry dog years.

DESIGN

Building a Better Pattern Library

FEDERICO HOLGADO

Stephen Kieran and James Timberlake's [Refabricating Architecture](#) profoundly impacted me as a designer. The book compares architecture to the automotive, aerospace, and shipbuilding industries, discussing how building construction processes haven't fundamentally changed in the last 80 years or so, while other industries have radically shifted how they create, design, and build.

“The traditional building paradigm is to gather all the parts of a building on site and then assemble them piece by piece,” Kieran and Timberlake write. First, an architect surveys the land, taking careful measurements and creating 2D drawings based on the designs. Next, a team using specialized machinery and devices interprets these drawings to prepare the land, and

further specialized machinery and teams start the process of pouring the foundation. Once the structural components of the building finally go up, then comes plumbing and, finally, the interior.

Kieran and Timberlanke contrast that rigid, linear process with other industries. The automotive industry has taken advantage of tools to share information, and has combined multiple processes to optimize production, cut costs, and make higher quality automobiles. Instead of manufacturing and assembling every piece of every car under the same factory roof, those pieces are subcontracted to outside manufacturers. In the shipbuilding industry, too, technology has adapted to the changing world. Modular construction allows shipbuilders to defeat gravity, and concurrently work on different modules of the ship that are not limited by the location or proximity to the ground.

In early 2013, when MailChimp's UX team began working on our big redesign, we took some of these ideas about modularity and multi-threaded development and applied them to the way we build software. The end result is a system of individual and interchangeable parts that start at the smallest possible level


and grow from there. Components are then assembled from these individual parts, and these components make up some of the core parts of our app. Once a group of components are assembled, we get a fully functioning page inside our application that is consistent in design and architecture with the other pages of our app—all because the pages share the same architecture.

Our [pattern library](#) consists of both fundamental pieces and ready-made components built from these small parts, and provides developers with a great toolset to build interfaces that work with our systems. Here, I'd like to show you how they've helped us in our work—specifically, in developing our “slats” system, one of the most fundamental and far-reaching parts of MailChimp.

Slats in the real world

When we began our redesign, we started with the most visited pages of our app, the main section dashboards. In the old MailChimp, we used a simple table to display these items. Upon further examination, we had a hunch that there was a better way of visualizing this data. So we spent a lot of time thinking, sketching, designing, and prototyping ideas.

For a while, nothing made sense. We explored a dual-mode interface with “cards” that were visually appealing and tables to satisfy the more information-dense requirements of some of our users. But a visual approach doesn’t work because of the similarity of the campaigns our users send—they all look generally the same. We also realized that one of the important things our users did in the campaigns table was sort by different attributes to “find” campaigns. The sortable attributes available didn’t make any sense, and this freed us from the notion that we had to display this data in a tabular manner.

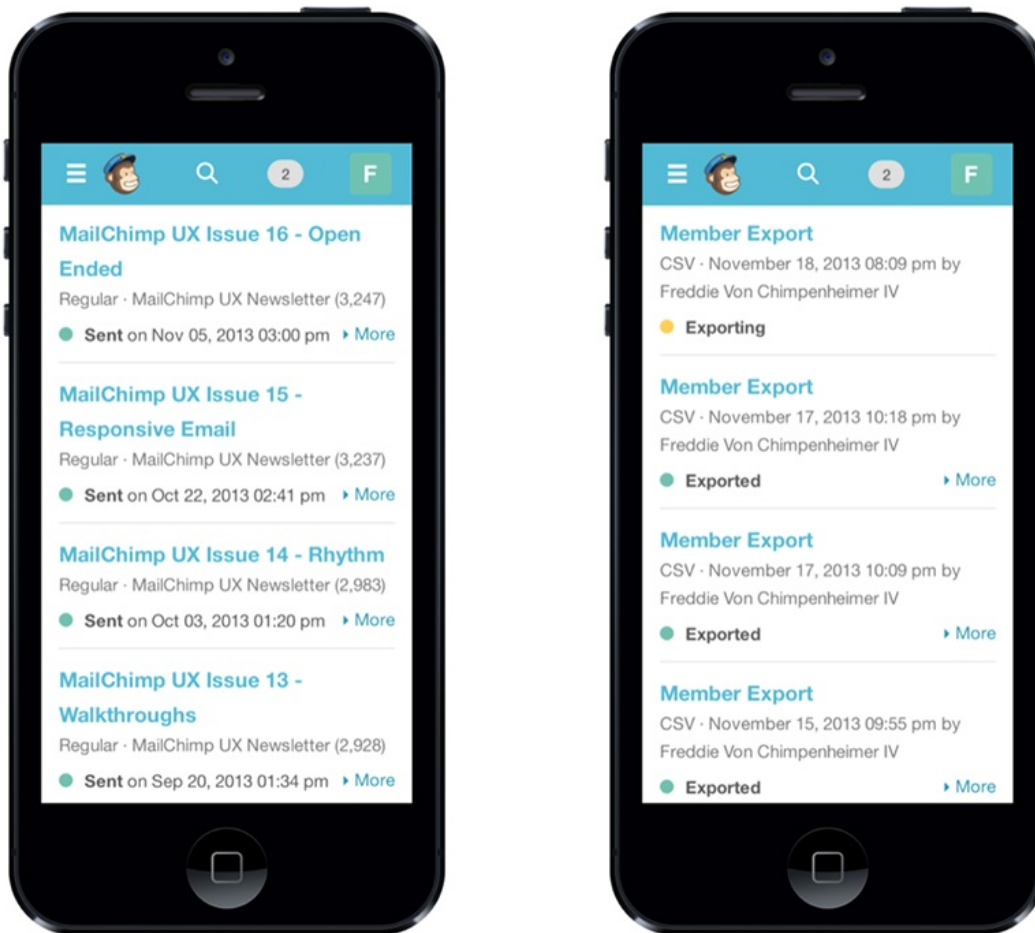
<input type="checkbox"/>	<input checked="" type="checkbox"/>	MailChimp UX Issue 16 - Open Ended Regular · MailChimp UX Newsletter Sent on Nov 05, 2013 03:00 pm	3,247 Subscribers	46.1% Open Rate	16.0% Click Rate	View Report ▾
⋮	<input type="checkbox"/>	MailChimp UX Newsletter Created Mar 05, 2013 09:17 pm ★★★★☆	3,884 Subscribers	50.9% Open Rate	11.2% Click Rate	+ Ⓞ Stats ▾
<input type="checkbox"/>		Re: MailChimp UX Issue 15 - Responsive Email Sue Johson (suejohnson@gmail.com) Hi Fernando I have an issue with Outlook just same as menti...				3 weeks ago Reply ▾
<input type="checkbox"/>	<input checked="" type="checkbox"/>	Member Export Exported as CSV · November 17, 2013 10:09 pm by Freddie			2,511 Subscribers	Download
<input type="checkbox"/>		Users without free emails (gmail, yahoo, etc.) Created 11/13/13 02:31 pm · Auto-Updated				Edit ▾

One example of the slats found throughout the MailChimp app.

The slat system was born out of efforts to make these connections between different sections of our app, and eventually we landed on something that worked for our 3 main sections. We added filtering instead of sorting (where it was appropriate), and, before we knew it, we had created the foundation of our dashboards.

The months we spent designing the original slats paid off in more ways than we could've imagined. After the initial time investment, we were able to reuse the same code to beautify

other parts of the app that would have otherwise been boring tables. Here are the mobile versions of the Campaigns and Exports sections side by side—you can clearly see the resemblance. Adapting our slat system to the Exports dashboard took me all of 45 minutes, and this included a mobile view that functions and looks great.



Mobile campaign slats on the left vs. export slats on the right.

Slats also now appear in Segments, the list of Conversations, and, most recently, our Exports dashboard. These parts of an application generally don't receive the same level of attention as a main dashboard, design-wise. But with our patterns, they fit within our system, they look nice, they're vetted, and they're familiar. They even respond to smaller screen sizes.

The beauty of this process was that while a small team splintered to work on the slat design, the rest of the team worked on implementing other parts of our system (like our forms, overall page structure, and navigation). They were working on the foundation at the same time we were working on the interior, and we all kept each other in the loop.

The old way, the new way

With this system we were able to modularize much more than just our slats. In fact, after our initial launch, we modularized most of the app. Forms, buttons, tables, and pieces of our editor all started falling into place.

I have to admit that, at first, it was tough to think in this way. I vividly remember working “the old way”—that is, crafting custom pieces of UI for every new feature we dreamed up. The CSS was unmanageable, and as much as we tried, things never felt like a true family of components. Everything had a slightly different makeup, both outwardly and in the underlying code.

Even though the time investment was significant in the beginning, we are still reaping the benefits. Taking a cue from our shipbuilding friends, we’ve created modular pieces that can

be developed at any point in the process. From the auto industry example, we've learned how to share requirements, how to break the QA cycle into smaller chunks, and how to share and tweak components.

Our teams work independently, but always from a common place. We have a platform to discuss our changes and ideas that allows not just the front-end devs, but also our engineers, to comment on code and design decisions. Our visual designers have helped us establish rules of [vertical and horizontal rhythm](#) that help us minimize the number of joints when we assemble our pages. If the system is working properly, it means that the designers and front-end devs are creating in harmony.

Our back-end developers have also benefitted from this system, since they are now encouraged to create general systems that can be reapplied. They have to learn how to implement less tooling, which speeds up our development time. Our standards now allow them to work on the data layer fully cognizant of what our UI expects.

And overall, the level of consistency and cohesiveness that we've reached feels good. The people that use our software now

have a beautiful, responsive, cohesive app that lets them work faster, and the people that build our software have a toolbox with which to craft our software in new and better ways.

In the end, we're in a better place now than we were before the redesign, and that makes the hard work and long hours we put in worth it. We've created a system that will be the basis of our application for future iterations. Based on the lessons of our sister industries, we're headed in the right direction.

High Five for SVGs

CALEB ANDREWS, ALVARO SANCHEZ

We spend a lot of time watching people use MailChimp, not only to learn how we can improve but also to get a sense for how people feel as they use the app. From doing this, we know sending a campaign can be stressful—users are often under deadline, and the fear of sending an email with errors is ever present. People twist their mouths, wring their hands, even break out in a sweat when they're about to send. We feel this stress ourselves every time we create our own UX Newsletter.

Knowing the emotional context of sending, we wanted to convey to our customers that we understand how they feel by acknowledging their moment with a congratulatory high five from Freddie. Previously, we displayed a picture of Freddie's hand, frozen in mid-air awaiting a high five, but we felt like we

could do more to provide our users with a sense of relief. In a recent release, we experimented with [SVG](#) animations to make this moment even more triumphant.

The SVG (short for “scalable vector graphic”) has been around for a while, but some folks are only now learning about its exciting advantages. SVG animation is a subject that’s hard to do justice in just 1 article, so we’re going to break it into 2 parts: [Caleb Andrews](#) will discuss preparing SVGs for animation, and [Alvaro Sanchez](#) will get into the nitty-gritty of how we made Freddie’s animated high five.

Caleb: Fitter, happier SVGs

We start our SVG animation journey with some tips on file cleanup and preparation that proved to be helpful when we were getting customers excited about our high five animations.

ESTABLISH LAYER HIERARCHY

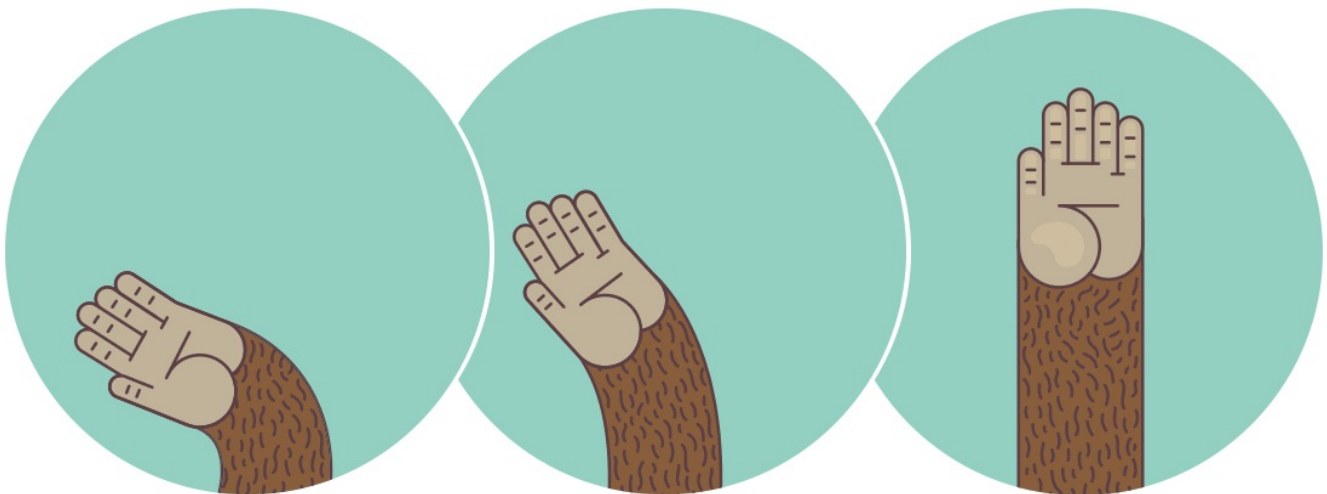
When you’re preparing an SVG illustration in Illustrator, layer hierarchy is key. We opted to place animation frames in groups so each illustration was clean and consistent. Properly naming groups goes a long way when you need to identify the output of

each group or path in a text editor, which we'll need to do when scripting the animation. For example, Freddie's watch is broken down into paths within a group so we can animate the time of a scheduled campaign.

REMOVE EXCESS ANCHORS

Creating fewer groups, paths, and points in a layer helps keep the SVG code light and legible. By installing [Hiroyuki Sato's scripts pack for Illustrator](#), it's easy to remove excess anchors under File > Scripts > Remove Anchors.

NO MASKS BEYOND A CERTAIN POINT



Above, Freddie's high five is merged into a single SVG without masks as 3 groups. These groups are considered the frames in

the animation and can be hidden or revealed when necessary. Sticking with the intended position, we chose to create the masks via code to have more control and a cleaner SVG.

DESELECT ILLUSTRATOR EDITING CAPABILITIES

Next, you'll need to export your art out of Illustrator as SVG. This option is easy to miss, but when saving your file in the SVG format, be sure to deselect "Preserve Illustrator Editing Capabilities" in "Options." Doing this will trim a lot of extra weight from the SVG code that you don't need in the browser. However, make sure you do this with the final version of your illustration because, as you might guess, you won't be able to easily edit this file in Illustrator again.

STROKES VERSUS SHAPES

```
<rect id="rectangle" x="193.5" y="50" style="fill:#249AB9;"  
<line id="stroke" style="fill: none; stroke: #F2F2F2; stroke
```

You can open your SVG assets in a text editor and take a look at the code generated. In the example above, I have SVG shapes for a rectangle and a stroke. In theory, they're the same thing—a rectangle—but the code to draw them is entirely different. If

working with stroked paths isn't your style, a stroke can be easily expanded by heading to "Object" and selecting "Expand" in Illustrator.

CONCLUSION



Not too difficult, huh? With just a little bit of thoughtful preparation, it's pretty simple to output an SVG that lives up to the hype and doesn't bog down the browser.

Alvaro: Behind the high five

Caleb has explained how we prepared our SVGs for animation; now I'll take it a step further and explain how we animated the

SVGs. I've simplified my descriptions here, but I link to the source code at the end, if you want to see all the components for the final animations.

GETTING STARTED

When I took on the Freddie animation project, I was given a prototype of the animation, made in After Effects and then exported as a GIF. Caleb helped me re-create in Illustrator the shapes used in the prototype—hands, arms, fur, etc. After I labeled the layers and organized the assets, I exported them as SVGs. This left us with 3 SVG files, one for each image of Freddie's arm in a different position. Next, I started to think about how to animate the shapes.

After doing some research on JavaScript libraries, I decided to go with [Snap.svg](#). Snap supports features like masking, clipping, patterns, full gradients, and groups. It also provides a simple and intuitive JavaScript API for animation.

The implementation for the animation functions (called “mina”—that’s “anim” backwards) uses requestAnimationFrame, which makes the animations perform really well in the browser, as [Paul Irish has explained](#).

(As a side note, if you use a library that doesn't support requestAnimationFrame to animate elements, be mindful of [layout thrashing](#). It occurs when JavaScript writes and then reads from the DOM over and over again, creating document reflows, which affect browser performance.)

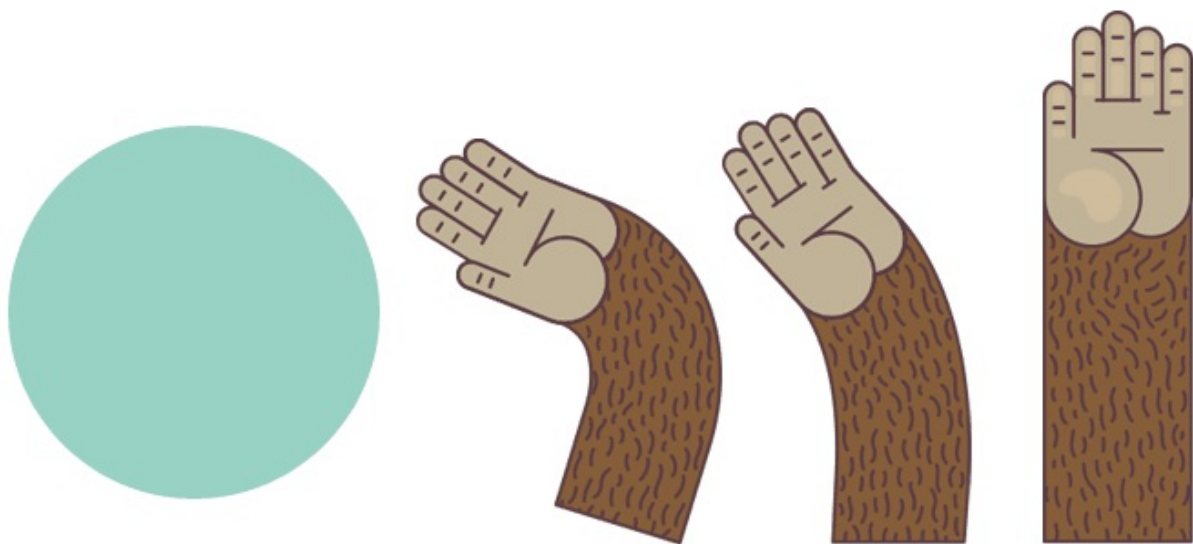
ANIMATING MOVEMENT

I created the high five animation with 3 separate images of Freddie's arm, each in a different position. For the final animation to look like a single arm in continuous motion, I needed to create a smooth transition between each image. To do this, I used a step animation technique and stitched the images together with transitions in between. (Why not just animate one image? Calculating all the points in Freddie's hairy arm and morphing the arm curvature was just too complex for the scope of this project.)

To load the 3 variations of Freddie's arm, I used the Snap.load function. On my first attempt, I used a separate Snap.load call for each arm. While the animation loaded just fine in Firefox, it broke in Google Chrome. Eventually I realized that Firefox was able to keep all the transitions and initial values, regardless of how many Snap.load calls were made. Chrome, however,

couldn't keep the transitions and initial values because each Snap.load call creates a new SVG fragment with its own viewport and coordinate system. I solved this issue by keeping all the assets (all 3 variations of Freddie's arm) in one SVG, so I only had to make one call instead of 3.

The next step was ordering the stack elements in the final animation.



To do that, I used Snap.svg's function to group elements. For example:

```
// Order of grouping is important!!!  
// s = SVG canvas created by Snap  
var group = s.group(  
  circleBG,  
  hand1,  
  hand2,  
  hand3  
);
```

Grouping also defines the stack order of those elements, with circleBG on the bottom and hand3 at the top.

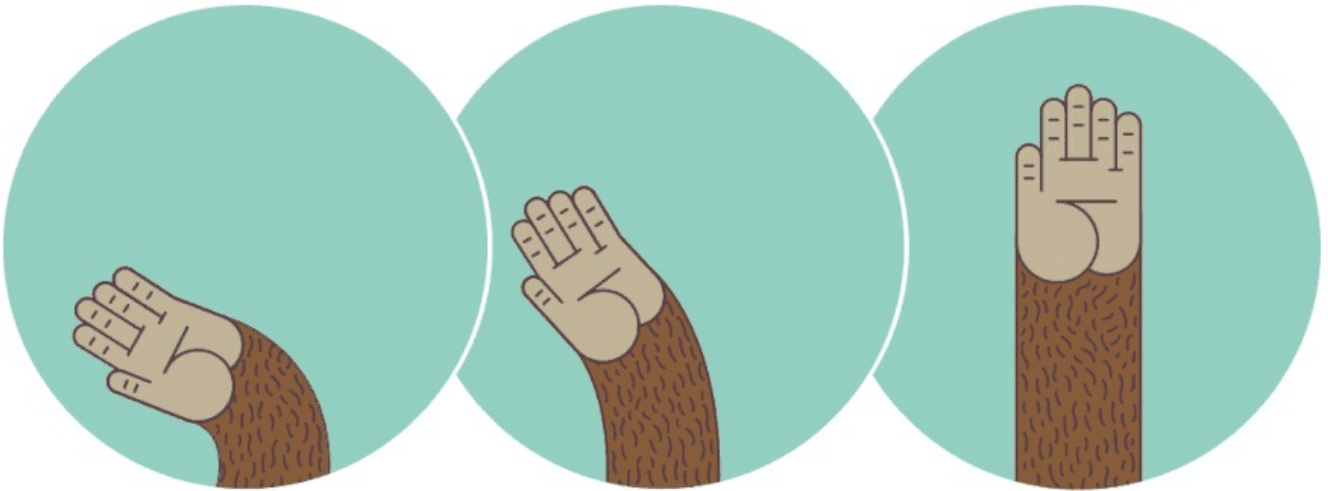
Then we defined and applied a mask to the group, which animates the elements within the mask's shape—in this case, a circle.

Here's how I defined a mask in Snap.svg:

```
// Create a circle at x:200 and y:200  
// and 200px radius  
circleMask = s.circle(  
  200,  
  200,  
  200  
);  
// Fill with white  
circleMask.attr({ fill: "#FFFFFF" });
```

To apply the mask to the elements in the group, I added the mask attribute to the group:

```
group.attr( {mask: circleMask} );
```



Once the mask was defined and applied, I initialized the position and visibility of all the elements. In our high five animation, I loaded 3 elements that I animated, one after the next, at different intervals.

To initialize the position of the arms, I used a [transformation string notation](#):

```
// Initialize position.  
arm1.transform(  
  "s0.6r-30t-100, 280"  
);
```

Each letter is a command: “s” is for scale, “r” is for rotate, and “t” is for translate.

With this notation I made arm1 40 percent smaller, rotated it -30 degrees, and placed it away from the mask so it is no longer visible.

I applied similar initializations to arm2 and arm3, but I hid them by setting the opacity attribute to 0:

```
// Set opacity to 0  
arm2.attr( {opacity: 0} );
```

CONNECTING ELEMENTS

Once I had all my assets in place and initialized, it was time to start animating and chaining transitions. This is where things got interesting.

Snap.svg has a function called Animate. This function receives the following attributes:

```
Element.animate(  
  attrs,  
  duration,  
  [easing],  
  [callback]  
);
```

Here is an example of the first animation triggered for the high five:

```
arm1.animate(  
  {transform: 't-50,60'},  
  400,  
  mina.backout,  
  function(){  
    // callback code here  
  }  
);
```

By passing the transform attribute and new values as the first parameter for the function, I told Snap.svg to move arm1 from -100px to -50px in the X-axis and from 280px to 60px in the Y-axis.

The second attribute (400), is the total duration of the animation in milliseconds.

The third attribute (`mina.backout`) is a timing function, just like `mina.linear`, `mina.easein`, `mina.easeout`, etc. The `mina.backout` timing function creates a whip effect, making the arm's movement appear more natural.

The fourth attribute is the callback function that executes as soon as the animation is complete. This is very important because with callback you can chain animations together and maintain control of when you're able to create concurrent new animations.

For example, in the transition between `arm1` and `arm2`, I fade `arm1` using the callback from the initial transformation. Once the fadeout is finished, `arm1` is hidden (`display:none`) and the reveal animation for `arm2` begins.

This all happens very quickly—the fadeout happens in 30 milliseconds, and the fadein in 10 milliseconds.

Here is the snippet of nested callbacks for what I just described:

```
arm1.animate(  
  {transform: 't-50,60'},  
  400,  
  mina.backout,  
  function(){  
    //400ms  
    arm1.animate(  
      {opacity: 0},  
      30,  
      mina.linear,  
      function(){  
        // hide arm1  
        arm1.attr(  
          {display: "none"}  
        );  
        // Chain/Start  
        // animation for arm2  
        _highFive.animate02();  
        // fadein 10ms  
      });  
  }, 100);
```

For a more detailed view of the code behind the High Five animation, check out the [source code](#) for yourself.

DEVELOPMENT

Building a System for Responsive Email

Fabio Carneiro

Tightening Type and Relative Font Sizing

Mardav Wala

Creativity in Front-End Development

Jason Beard

Release Cycles and Roadmaps

Federico Holgado



Building a System for Responsive Email

FABIO CARNEIRO

Creating a modular [responsive email blueprint](#) for email designers to learn and work from was a unique challenge. Though there are plenty of articles and code snippets in MailChimp's [email design reference](#) library, getting into ready-to-use, working code puts all of that information into a practical frame of reference.

The responsive blueprint wasn't MailChimp-specific, so I had to consider the ways in which different email designers might actually use the code. It came down to 2 things: The emails had to be adaptable to many purposes, and they had to be robust across a wide variety of email clients.

Achieving robustness was the easy part. Many email clients are a nightmare to deal with, but things get easier once you become familiar with the **ins and outs** of each. From there, it's just a matter of working within those constraints.

Adaptability was trickier. Designing and building an adaptable email for someone with a defined set of purposes is simple, because you can take liberties with design and use specialized HTML or CSS to make it all work. That's because there's context: knowledge of what the email is for, what sort of content it'll be stuffed with, and—hopefully—which email clients it will be viewed in. But there's zero context to work from when designing and building emails that might be used by large groups of people, many of them in ways that we can't foresee.

To work around this lack of context, I followed a process that's similar to the one I apply to the email templates I build for use in MailChimp:

1. Anticipate common use cases
2. Determine design patterns for those use cases
3. Create loose hierarchies and modular HTML structures

4. Reinforce the code with client-specific HTML or CSS
5. Educate within the code with commented, easy-to-use markup

Here's how I worked it out.

Anticipate common use cases

By anticipating common use cases, I was able to establish a rough idea of just what I needed to code and how I needed to code it.

Pretty much all email falls into 4 broad categories: Read Me, like newsletters or RSS-to-email campaigns, which deliver information and are text- or image-based; Buy Me, related to e-commerce, entice the reader to spend money; Join Me, like event notices and invites, highlight a particular occasion and urge an action towards it; and Understand Me, which are transactional, like receipts and order summaries, and convey important details or data, and little else.

In the case of this email blueprint, I had to account for all of those scenarios.

Determine design patterns

All of those use cases share components with each other. For instance, Buy Me, Join Me, and Understand Me emails might all contain a call to action in the form of a button, while Read Me and Buy Me emails rely on interesting content, most commonly in the form of multiple images.

By examining the common threads between each type of email, shared design patterns began to emerge, and I could then compile a list of the blocks to be built. Ultimately, 17 different patterns—like buttons, image groups, callouts, boxed content, highlight calendars, and captioned images—were included in the blueprint.

Create modular HTML

Once I knew what I was coding, it was time to figure out how to code it.

With this blueprint, I felt that anyone should be able to grab the HTML and understand how the code is structured, arrange design patterns to suit different needs, style the email without

worrying about the framework, and send campaigns almost immediately.

To meet those requirements, I wrote the HTML so that all content blocks shared a common supporting structure, which made the code easy to read and understand. Each of the content design patterns was sequestered in its own table row, which allowed for modularity within the template—content areas could be duplicated or moved simply by copying or moving the containing row.

Additionally, the entire framework was run by very minimal CSS, just 4 rulesets in the standard CSS and another 4 in the media query. There was some placeholder styling for the content itself, but it only affected text styles and background colors, so a designer could go in and write their own CSS without having to worry too much about how the functionality of the email would be affected.

Reinforce code

Once I finished building, I ran the email through my usual series of tests in desktop, browser, and mobile email clients. In the first run-through, the email rendered without major issues

in all the popular email clients. I wasn't quite finished, however, because of 2 email clients.

Outlook 2007/2010 and the Android Gmail mobile app are, in a word, terrible. Outlook has a host of issues due to its Microsoft Word-based rendering engine, and the Gmail app, despite being a mobile app, doesn't support media queries. Writing email HTML for the former requires very rigid, carefully crafted code, while the latter requires “spongy” code: not responsive, or even fluid, but pliable.

We normally err on the side of caution and make our emails more robust in order to support Outlook—after all, it has a stranglehold on the desktop email client market. But because this responsive blueprint was meant for email designers with a bit of technical know-how, I took the opposite tack and went for maximum mobile support. This meant coding many of the content blocks using **aligned tables**. Now, the Android Gmail app is happy.

You might be thinking, “But why only the Android Gmail app? There's one for iOS, too!” And there is, but the iOS Gmail app is

even worse—it supports nothing at all. Emails will render just fine, but for now, you can forget about mobile friendliness.

Regardless, using aligned tables as a solution for Gmail is great for flexibility, but extremely brittle and prone to layout mishaps in (you guessed it) Outlook 2007/2010. The problems all stem from a particularly frustrating bug in Outlook that automatically inserts a page break into any document that measures longer than 22 inches (or 1700px) in length. When that happens, the aligned tables, which are essentially floated, get all sorts of wonky. Fortunately, it seems like the problem was solved in Outlook 2013.

One solution was to use conditional CSS and write a stylesheet to override problems in Outlook. In this case, I kept it simple and just collapsed the layouts into a single-column stack—the same thing that’s done on mobile devices via the media query. The conditional CSS is entirely optional, but it is a nice fallback.

With support for Outlook 2007/2010 and the Android Gmail app shored up, the email is now pretty robust across multiple clients and reasonably flexible, too.

Educate with comments

Some of these techniques may verge into “way out there” territory, particularly for new email designers. To that end, I made sure to include explanations within the code for why certain bits of code are used and exactly what they do.

I also took steps to indicate where a content section begins and ends, and whether or not it requires some kind of special treatment.

By keeping the responsive email blueprint focused on common uses, including design patterns ubiquitous in email, ensuring the code is both robust and flexible, and taking the time to clarify murky code, anyone should be able to dig into the responsive email blueprint HTML and start working with ease.

Combined with our [email template reference](#) library, we’re hoping to dispel some of the mystery and confusion surrounding HTML email design. We’re not finished, though—not by a long shot. Both the reference library and the email blueprints are projects that I plan to improve and evolve based on feedback.

Tightening Type and Relative Font Sizing

MARDAV WALA

MailChimp's 2013 redesign gave us the opportunity to carefully consider the typography in the app. Laying out content in a grid is a piece of cake these days, but attaining vertical rhythm on the web (that's a fancy way of saying "aligning content to a baseline grid") is still nothing short of harrowing, especially when the content varies wildly. But after addressing the initial inconsistencies between design and code, which were mainly due to the differences in how the line height and leading is perceived (or not) in print and web media, our front-end and design teams are finally speaking the same language.

Tightening type

As [Richard Rutter has noted](#), the vertical rhythm on any web page can be achieved by carefully applying line height, margin, and padding to the content. The trick lies in finding a suitable line height, which forms the basis of calculating margins and paddings.

Although the MailChimp app is content-heavy, very little of its content consists of paragraphs of text—almost all of the information is displayed as either lists, forms, tables, charts, or data blocks. So instead of starting with a large value for a line height, which would translate to large margins and paddings, we started with the smallest possible margin value of 6px for any element in the app. Our 6px baseline grid is derived from this value.

Why 6px? We experimented with a number of base units, but found that 6px multiplied elegantly to 12px, 18px, 24px, and so on, giving us a nice range of type sizes and margins. It also worked well when applied to small elements like buttons and form fields. It offered us the flexibility we needed to construct any UI.

Change is a constant in MailChimp—we release new features and refinements for the UI every 4 weeks—so in our quest for vertical rhythm we had to design flexibility into our system. Early in our design process we decided to apply margin spacing only to the bottom of elements to make it easier to maintain vertical rhythm. This way, new modules could be accommodated without disrupting the visual hierarchy on a page. Single-direction margins helped us achieve this goal.

SIMPLE MATH

Because our UI is designed on a 6px baseline grid, all line heights, margins, and padding had to be applied in multiples of 6 to maintain vertical rhythm. Fonts, however, can be set to any size without breaking rhythm. Our base font size is 15px, a value that we found to be legible in all situations without making UIs feel oversized.

Best practices suggest setting the line height of type at one and a half times the size of the type to improve legibility. With our base font size set at 15px, that would result in line height at 22.5px. But because our baseline grid is set on a 6px base, we tweaked the line height to 24px, creating a relationship between the padding, margin, and line height in all layouts.

With the math sorted, we started applying these proportions to the elements in our app.

For all headers and other font sizes, the line height is again a multiple of 6 and is calculated based on the font size itself. The examples use pixel units for the sake of simplicity:

```
h1 {
  font-size: 40px;
  line-height: 48px;
}

.small-meta {
  font-size: 13px;
  line-height: 18px;
}
```

EXCEPTIONS TO THE RULES

Images and charts march to the beat of their own drummers, often breaking the baseline grid. Their heights are unpredictable and can't easily be made to fall in step with a baseline. But the vertical rhythm remains unaffected because the vertical spacing defined by the margins and padding doesn't change.

Elements with borders can also throw off the baseline grid, because they're added to, not included in, line height calculations. But there's an easy workaround: Simply include the border as part of the overall element height. (This may require reducing the top or bottom padding based on the border height.)

Of course, purists may argue that this results in visual imbalance if the border height is more than 1px—and they are absolutely right. Striving to align horizontal rules and elements with borders to the baseline grid is not always the right thing to do, because the vertical rhythm is never affected with borders and horizontal rules as long as the margins, line heights, and paddings are correct.

Here's an example of how we trimmed padding in list items to account for a 1px border separating elements:

```
@base-unit: 6px;

.dotted-list {
  margin-bottom: (@base-unit * 2);
  li {
    padding-top: (@base-unit * 2)
    padding-right: 0;
    // 1px less padding bottom
    padding-bottom: ((@base-unit * 2) - 1);
    padding-left: 0;
    border-bottom: 1px dotted #c0ffee;
  }
}
```

Charts can also be adjusted to follow your baseline proportions. If the chart maintains the aspect ratio upon resizing the browser window, set the chart height and vertical margins to be a multiple of your base unit. If not, only set the vertical margins and leave the height unchanged.

Because an image's height will scale as the width changes, we can't depend on the height aligning to the baseline grid. In these cases, we ensure that the margins surrounding the image are specified to maintain the vertical rhythm.

Maintaining code with such special rules and exceptions can get messy. A quick word with the team and a simple comment in

the code might alleviate some difficulties. Documenting the change in a live pattern library also goes a long way.

We've also found that tweaking the vertical spacing at a modular level—designing from the “content out,” as **Mark Boulton recommends**—makes more sense than making refinements on a global, per-page basis. If the individual modules adhere to the baseline grid, no matter where they're placed on the page the vertical rhythm and the visual hierarchy will automatically be maintained.

Relative font sizing

When using relative font sizing with em units for **fluid, scalable web design**, the inevitable **cascade** may be more frustrating and complicated than using absolute pixel-based font size rules.

Translating pixel-based design comps to em-based markup used to be a chore. Luckily, we can simplify the process by using the combined powers of CSS pre-processors and **Ethan Marcotte's** now ubiquitous **target ÷ context = result**.

The key to maintaining sanity amid this em-related cascade madness is to focus on context. For nested em-based elements,

the context is their parents' font-size in pixels; for the parent elements, the context is the default body font-size (16px, if you leave the default body font-size alone).

Consider the following HTML code based on this [example](#) from Treehouse:

```
<h1>Title: <span>Tagline</span></h1>
```

The font sizes in ems for the heading and the nested span using a LESS mixin can be easily obtained as:

```
// default body font-size
@baseFontSize: 16;
@h1Size: 24;
@h1SpanSize: 18;

h1 {
  #pxtoem > .font-size(@h1Size, @baseFontSize);
  // 24 ÷ 16 = 1.5em
  > span {
    #pxtoem > .font-size(@h1SpanSize, @h1Size);
    // 18 ÷ 24 = .75em
  }
}
```

While it may get difficult to keep track of the font sizes as the nested levels increase (lists within lists with nested divs,

headings, paragraphs, hyperlinks, and spans), a little effort up front can help achieve a small piece of the ever-elusive CSS nirvana.

DEVELOPMENT

Creativity in Front-End Development

JASON BEAIRD

We've written a lot in the UX Newsletter about our [pattern library](#) and how it helps us iterate quickly and ensure consistency within MailChimp. Building with and extending existing patterns is a bit like working with Legos: When you start building, you know exactly how the pieces should go together— but sometimes it's fun to break out of the pattern box and do things a little differently. I'd like to share a few examples of how we've done that at MailChimp.

Text based text-align icons



We needed to add icons for left, center, right, and justified text alignment. I could have easily added 4 new characters to our [icon font](#), but they just seemed too simple. I decided to take a creative approach and build them with CSS. Basic [CSS Shapes](#) are common practice, but they're usually created by adding background colors to modified block elements. Instead, I used aligned em and en dashes as pseudo elements to allow them to inherit text size and color the same way an icon font would. It was a fun little exercise and only took a few lines of CSS. ([View the icon demo on Codepen.](#))

D3 animated clock icons

 **MailChimp UX Issue 22 - Experimentation**
Regular · MailChimp UX Newsletter
Scheduled for Feb 28, 2014 01:45 pm
Scheduled ▼

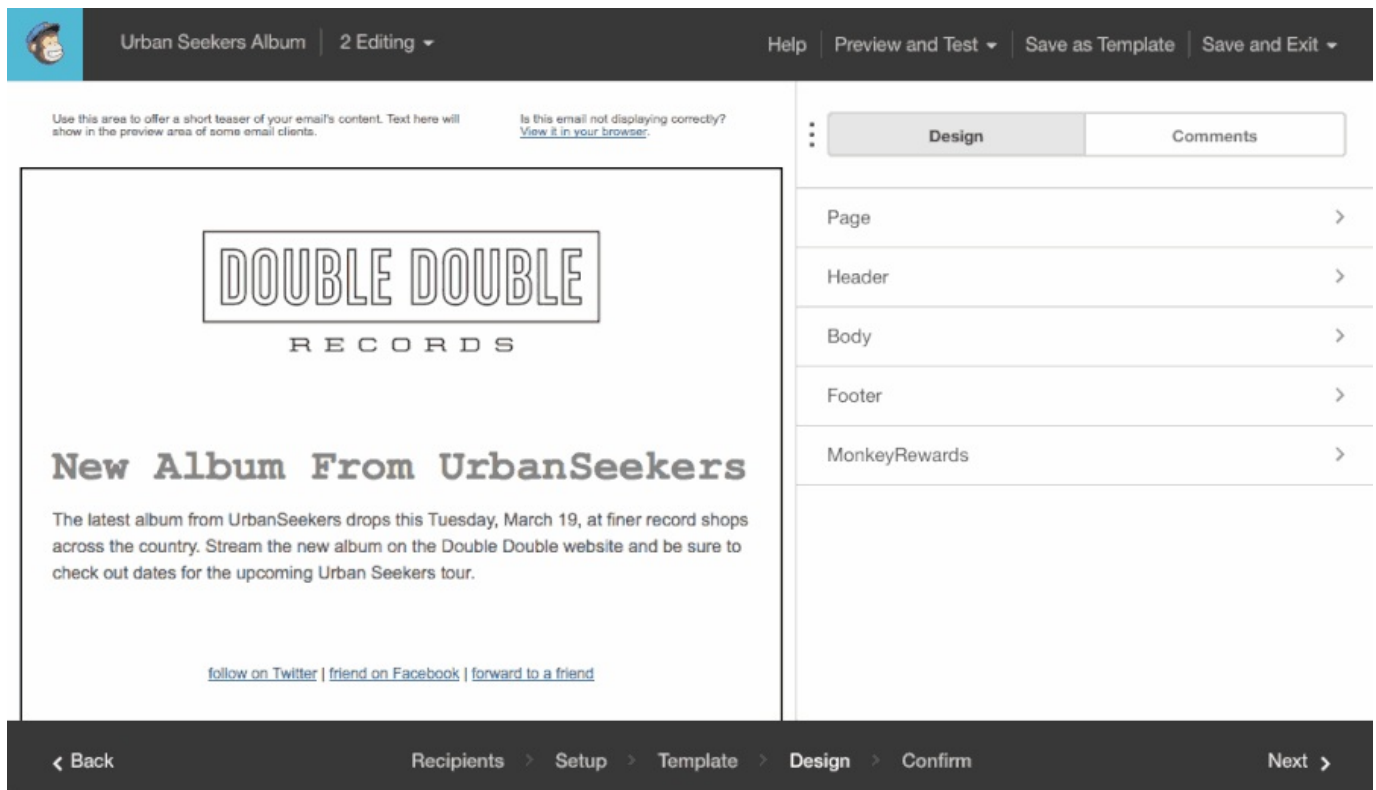
A more complex example of front-end creativity was the result of a single tweet. Mardav took it as a personal challenge when Thierry Blancpain **said**, “Given their amazing copywriting I’m always a bit disappointed that MailChimp’s icon for scheduled campaigns does not reflect the send time.”

Mardav had been tinkering for a while with a JavaScript library called **D3.js**, which is typically used to create complex charts and data visualizations. We already planned to start using D3 for making charts, so it ended up being a great excuse to try it out on these clock icons. Mardav whipped up a prototype and, with a little help from our awesome engineering team, it was converted into a **custom Dojo widget** that we now use all over the app.

If you set your campaign to send at 1:45p.m., or 7:30a.m., the clock can now reflect that time.

It’s a fun little detail that we didn’t think people would notice, so we were pretty excited when it was **featured on Little Big Details**. Thanks for the idea, Thierry. (**[View the clock demo on CodePen.](#)**)

Animated GIFs for onboarding



We've always used video tutorials to teach new users how MailChimp works. Most of these videos are less than 2 minutes long, but we've found that sometimes you need less time than that—say, only a few seconds. Animated GIFs are perfect for that, and since our 2013 redesign, we've been using them on occasion to show how new interface patterns work.

Unfortunately, when converting screen capture footage to GIFs in Photoshop, the file size can be pretty huge. When Federico created a series of short animations as part of an onboarding process for our new editor, even with fairly aggressive

compression settings, the images each weighed in at over 1MB. That's pretty tiny for video, but way too heavy for a quick animated tutorial. So he started looking for other examples of animated GIF screencasts and stumbled across **an email he received** from the photography platform **Exposure**.

The 2 GIFs in Exposure's email had a bit of photographic content, were sized for retina displays, and weighed in at a mere 171KB and 401KB. Federico asked Exposure's **Luke Beard** what they used to accomplish such a feat, and Beard replied: "only the best, **cockos.com/licecap/**."

At first we thought we might have been punked: The **table-based site** looks like something out of the 1990s, the company that makes it is called "Cockos Inc.," and the name "Licecap" (ew) didn't inspire much confidence. But it was GPL free software, so Federico gave it a try and re-recorded the GIF in question, taking the output from 1MB down to a minuscule 182KB. Problem solved! (Thanks, Luke. Sorry we doubted you.)

Do over-think it (sometimes)

Budgets and deadlines sometimes require you to rely on the most obvious solution to a problem, but it shouldn't always be

that way. Next time you're working out a front-end challenge, try second-guessing your first solution. Do something a little different. Take every opportunity you can to apply new techniques and technologies, especially when it will delight your users. And don't be afraid to ask how someone else solved the same problem—you never know what new tips and tricks you might learn.

DEVELOPMENT

Release Cycles and Roadmaps

FEDERICO HOLGADO

At MailChimp, we have to move fast—and that has as much to do with how we work as what we work on. We have to plan things that we can build efficiently, picking the right things to work on at any given time. If you combine the right ideas with a solid but flexible process for building software, you can achieve some serious speed.

Our release cycle is constantly evolving, but right now, it's 5 weeks long: 4 weeks of feature development and 1 week of code freeze and QA. We limit our development time to keep ourselves from building monolithic features that could break MailChimp in major ways. Compared to code developed over 6 months or a year, 4 week chunks of code are smaller in scope and therefore infinitely easier to test. Also, because we're

constrained by time, it forces us to pare down pie-in-the-sky ideas to something more manageable. But it wasn't always like this.

The old way: ad hoc planning

When I first started managing releases for MailChimp development, features were generally designed and built within the same release cycle. At the time, our release cycles were 4 weeks instead of 5, which meant we only had a week or so to get together a design that we could pass off to the UX devs and the engineers to build. When designing and building in the same release cycle, it was hard for people to mentally prepare: Details were missed, scope increased because of unknown unknowns, and it made our process more stressful. It was also difficult to change our mind about something mid-release, since we'd effectively have to start designing all over again.

The new way: knowing what comes next

Things have changed a bit since our early days of release planning. One of our recent goals was to get better at knowing what was coming next. Now we push to have an entire release cycle's worth of lead time to design. Engineers and CEOs are

much more comfortable committing to a large feature after it's been carefully dissected by UX, designers, and engineers, because that time gives us a chance to find as many of the “gotchas” as we can.

Large features are now designed with engineers providing valuable insight from the start. Getting them involved in our design process as early as possible means we're less likely to waste energy chasing down ideas that just aren't feasible to build. It also turns out that we have some really brilliant engineers that are just as capable as UX team members when it comes to suggesting ideas, workflows, and features. (That took me a while to realize!)

Aiming in the right direction

The trick to moving fast is making every effort count. We have to carefully consider what we commit ourselves to, because backing out of an idea costs a lot of time. And the only way we can quickly come to an agreement about what to build is by talking with the people who are the most qualified to help us find the answers. These questions are generally along the lines of, “Is this idea feasible to build?” “Is it the right time to build

this idea?” And, the most important: “Is this idea worth building?”

To kick off a release cycle, I’ll get together with MailChimp’s lead engineer, **Eric Muntz**, and our CEO, **Ben Chestnut**, to discuss possible features and refinements to work on.

Sometimes ideas come from past research and design on the UX side, sometimes from engineering-related discoveries, and sometimes straight from Ben as he talks to customers and formulates ideas. Between the 3 of us, we settle on high-level features and try to estimate the scope of what we’re going to build. And we essentially ask and answer those same questions for every big idea we discuss.

The key here is that these are high-level ideas, but they’re based on things that we’ve been thinking about for a while. For example, we decided to build comments and collaboration into MailChimp months after we launched a small labs app called OnStage, which mimicked some of the functionality we were after. We figured out what worked in OnStage and what didn’t, and then we took those ideas and crafted them into a plan for the app. Once we decided to proceed, we had a clear idea of what

to build, and that focused energy let us move quickly and release a big feature that we knew was going to work.

Taking wrong turns

Sometimes, of course, a feature doesn't end up making the cut for the current release. That decision is made early in the release cycle so that we can shift resources quickly. Usually, if we end up punting on a feature we already started working on, it's a matter of timing and not because it's something we don't want to build. Whenever we scrap a feature, we add it to the "parts bin," which we can always reach back into later.

Let's build some stuff

Here's a more detailed look at what it feels like to go through these 5 week cycles of building stuff. (For me, it's reminiscent of writing a research paper in college: In the beginning, I think I have so much time to work on it. Eventually, I realize I've got to get some work done, and the night before the paper's due, I hustle and produce a good piece of work.)

WEEK 1: RELEASING/PLANNING

We spend the first couple days deploying code from the last release. We've also got our eyes and ears glued to our support team, our inboxes and Twitter so we can listen hard to feedback coming in and change fast in response. We've surely missed some bugs that will need to be hot-patched to production during the first couple of days. We're also talking and trying to refine ideas about what we're building next. We have our "release" meeting this week, too.

WEEK 2: MORE PLANNING, SOME BUILDING

We usually have a few tasks left over from the previous cycle that we can jump on right away, while we're still figuring out some of the design details of the big features we'll be working on for the next release. Sometimes, if we have design details finalized, we're actually being productive and building new stuff.

WEEK 3: PRETEND WE'RE BUILDING STUFF

This is where things start taking shape and features start to get fleshed out. That includes lots of talking and figuring out implementation details.

WEEK 4: ACTUALLY BUILD STUFF

We're getting close to the deadline. We better start coding.

WEEK 5: CODE FREEZE! AND MORE PLANNING

We always have the best intentions to totally freeze feature development during this last week, but our QA team is so awesome at finding bugs and errors that this week is when things actually become presentable enough for other humans to look at.

I've heard stories of companies who dedicate entire teams of people to build things, only to have those projects or features killed at the last minute. Killing off a product team's feature is probably the most demoralizing thing you can do to a group of people who build stuff, and it's also incredibly unproductive. We take the utmost care to pick our commitments carefully. We respect the time and energy our teams put into everything they do—but also, we just love the thrill of speed.

REFINEMENT

Iteration and the Feature/Refinement Balance

Jason Beard

Making MailChimp More Accessible

Mardav Wala

Fixing a Leaky Funnel with Google Analytics

Allison Urban

Learning from Our Mistakes

Tyrick Christian

Refining via Redesign

Tyrick Christian



REFINEMENT

Iteration and the Feature/Refinement Balance

JASON BEAIRD

Recently, a company sent a rebate check they owed me to my old address—for the second time. “It’s a bug, and they need to fix it!” my wife exclaimed, annoyed. While I was also pretty peeved, I couldn’t help but feel a bit of sympathy for the team that would have to troubleshoot the issue.

When a user encounters a bug, or when part of your application confuses them and costs them time, it erodes their trust in your product. It also increases the load on your support team. Like holes in a bucket, big ones will drain trust quickly, and lots of small ones will kill it slowly over time. Either way, once that trust is gone, it’s gone.

Even though we're constantly working hard to improve our app, MailChimp is not immune to bugs. And I'll be the first to admit that our bucket has experienced its share of both big and small holes.

Find the holes, protect your bucket

When you work on something every day, it can be easy to overlook its shortcomings. But sometimes a seemingly innocuous oversight can have devastating consequences.

This is why the UX team relies on interviews, surveys, analytics, and other research methods to reveal the weak spots in our app. We also try to pay attention to every feedback channel our users might use to express opinions or frustrations. We watch social outlets like Twitter, Facebook, and blog comments, and our support team helps by tagging all of the chat and email correspondence to help us discover common problems. And, as Aarron Walter, our UX director, [has written](#), we've found some of the biggest leaks in our trust bucket through account closing surveys.

Prioritize and patch

Collecting feedback is easy, but filtering and determining what to act on is tough. If we made changes based on every comment, tweet, email, chat, and fax (OK, we don't really collect feedback by fax), we'd have more tickets than our team could address. Many companies just hire more people to address every bug, but that approach doesn't scale and it can introduce usability issues. Not every suggestion we get is worth implementing, as they may be contradictory or would help only a few people. Instead, our research team keeps a finger on the pulse of our feedback and forwards only the most repeated and interesting requests.

If you look at our commit logs for a given 5-week release, a large number of the comments include words like “fixed,” “tweaked,” “refined,” “adjusted,” and “changed.” This is where a lot of those suggestions are getting made. Some bugs, typos, or display issues that affect people's workflows get hot-patched in between release cycles, too.

Turn weakness into strength

MailChimp gets a lot of attention when we introduce new, innovative features, so it's easy to focus on that instead of making minor refinements. Oftentimes, though, a new feature

is simply a rethinking of part of the app that hasn't been updated in a while. This is the kind of stuff that everyone from UX researchers to our awesome dev engineers most enjoys working on. It's exciting to build new things, even if those new things are just reimagined incarnations of existing features.

Make time to refine

Sometimes you need to slow down, set aside new ideas, and work only on maintenance and refinement. Reducing the ticket queue isn't a particularly exciting task, and minor refinements don't make for popular blog posts, but both are necessary.

At least once a year, we have an entire release focused solely on cleanup and maintenance. We find that the holidays are a good time for this. We'll also use that time to plan new features for the next release and the upcoming year.

Think small—and big

When we're working on bugs and refinement, it's easy to become nearsighted, missing the big picture of how our changes affect MailChimp's overall UX. Working on big, new features allows us a broader perspective—but sometimes we

miss small details, fail to predict how people will use them, or discover edge cases we didn't consider before. The most important thing is that we keep doing both, so our bucket keeps getting bigger and better with as few leaks as possible.

REFINEMENT

Making MailChimp More Accessible

MARDAV WALA

We're lucky that our customers provide valuable feedback, not only about what works well in MailChimp, but also about what doesn't. Here's an example: A little while ago, customer **Justin Romack**, who is blind, let us know that he was struggling with the accessibility of our dashboards and subscriber table.

“Email marketing is an insanely important part of my business, and the clients I work with each and every day,” Justin told us. “Being a totally blind guy and depending on screen reading technology, it's not always easy to find a dynamite option that meets my needs and is fully accessible. MailChimp most certainly comes the closest, though.”

Lucky for us, Justin had consulted on accessibility issues before and offered to help us improve MailChimp. He was kind enough to send us some videos showing the problem areas in our app. By watching those videos and **reading some articles**, we found that we could greatly improve accessibility in the app by focusing on 2 objectives: Establish and provide context on elements using either the WAI-ARIA **roles and properties** or the HTML **title attribute**, and ensure that all markup is semantically correct. (This wasn't true for the MailChimp subscriber table, which used a div-based layout at the time.)

Here's how we made it work.

Dashboard improvements

Before we heard from Justin, we had already started making accessibility improvements to the MailChimp dashboards. These easy but effective fixes add context to the dashboard elements, which screen readers use to provide audible descriptions.

All dashboard views in MailChimp are based on the **slats pattern**, which is made up of semantically correct list elements containing a checkbox, a linked heading with supplemental

information, and a combo button with action items. Some dashboards also include a subscriber count, along with open and click rates.

At the time, the dashboard's supplemental information about campaign type and list name didn't provide enough information to explain what these elements actually mean. Campaign type "Regular" and list name "Testing 1" weren't very descriptive or self-explanatory. To fix this, we used **aria-label** to add descriptions to the campaign and list information. This instructs screen readers to announce text preceded by either "campaign type" or "list name." We also used an **aria-label** for the dashboard icons to provide audio cues about campaign status.

Although the subscriber count and open/click rates are written in semantically correct markup, a screen reader would announce the numbers and corresponding labels separately, making the information difficult to understand. To prevent this from happening, we hid the subscriber and open/click rate information from the screen reader, then added the information back as an **aria-label** on the parent container. This

prompts the screen reader to announce the numbers and their labels together.

Subscriber table improvements

As we continued to work on accessibility improvements, the subscriber table got a much-needed refactor. It is now defined in semantically correct data table markup.

As shown in this [this example](#), our subscriber table scrolls horizontally behind the single fixed column on the left. Originally, we used `th`, `thead`, and `tbody` in conjunction with this absolutely-positioned column, but this made [VoiceOver](#) announce 2 column names per column. For VoiceOver users, this shifted the entire table to the left. To fix this, we now only use `td` elements in the table and assign the [columnheader](#) role to the headers. We found [Yahoo! Accessibility's video](#) on implementing accessible tables very useful while we worked through these changes.

We added more context using `title` attributes to the subscriber profile checkboxes, which allows screen readers to announce the subscriber email address when a checkbox gets focused.

We also improved the member rating list for accessibility with ARIA roles and labels. ([View subscriber table on CodePen.](#))

Takeaways

Ensuring that markup is semantic and providing descriptive context to elements are both very easy things we can do to make an app screen reader-friendly.

For complex JS-based interactions, it's best to use one of the available frameworks. We use [Dojo](#) and we get all the UI widgets with built-in accessibility features from its [UI library](#).

After we made our adjustments, we asked Justin for a follow-up. We were happy to receive a positive response: “Nothing wonky. Nothing weird,” he said. “It reads straight across perfectly—just as expected. Loving it! :)”

REFINEMENT

Fixing a Leaky Funnel with Google Analytics

ALLISON URBAN

As a web analyst, I spend a lot of time thinking about how to translate data into design insight. One of my favorite tools for this is Google Analytics' Next Page Paths. This handy little report tells you where users go after viewing a particular page. In terms of UX, it's one of the quickest ways to identify where things are going wrong in a funnel.

Here's an example from our Account Activation funnel. After a user signs up, they need to complete a CAPTCHA before they can log into their account. The navigation pattern we'd expect to see in Google Analytics goes like this:

signup > signup/success > signup/confirm >
login.mailchimp.com

However, we noticed that 32% of next page paths from signup/success go to login.mailchimp.com, completely bypassing signup/confirm.

If you could log into your account without completing the CAPTCHA first, this wouldn't be an issue. The problem is, you can't.

It turns out, you used to be able log in before completing a CAPTCHA, but our developers patched the loophole for security reasons. So why were some people still trying to log in rather than following our instructions to confirm their account? Well, because we were encouraging them with a shiny Log In button.

Since removing the Log In button from /signup/success/, the percentage of people going directly to /signup/confirm/ has increased from 63% to 79%. Our support department also reports receiving 25% fewer emails related to account activation, which puts them on track for the lowest number of emails related to account activation all year.

Not bad for a few minutes of research.

REFINEMENT

Learning from Our Mistakes

TYRICK CHRISTIAN

In golf, a hole in one is rare. Even after training for years to obtain a perfect swing, golfers still face obstacles out of their control: wind, topography, turf texture. Golfers know this, which is why they don't get their hearts set on a hole in one every time—they just do their best to move the ball closer to the hole.

Designers face similar obstacles when they're staring at a blank page. Getting layout, workflow, language, and style just right in one try is almost impossible. Instead, we break each problem down and, piece by piece, move them closer to meeting the needs of our users. We're not trying to nail it the first time out. We're iterating.

In 2013, the MailChimp UX team started an enormous redesign of our web app. Mobile web traffic was growing quickly and user research showed that people wanted access to MailChimp from a range of devices and contexts throughout the day, no matter where they were. Customer interviews and a broad study of how mobile devices shape culture led to MailChimp's vision of the future. To show our team the story of the experience we wanted to build, we wrote a script, hired actors, and created a short video portraying the use cases and workflow our future app would enable. The video didn't show a new UI, just a way of working.

From that point on, the goals were clear and it was time to start designing. We were working on concepts for a range of screen sizes, from phone to desktop monitor, so we decided to build a pattern library, which my teammates have described in detail elsewhere in this book. During this time, we were less concerned with creating a single, perfect idea, and more concerned with iterating through several ideas until we found the best solution.

Inside the MailChimp app, users get an overview of important account data on a series of dashboards. During the redesign, we

experimented with using tables and cards as a means of better displaying this data. After weeks of experimentation, we found ourselves looking at interesting ideas, but we weren't convinced that they were good solutions. Tables were clear, but boring. Cards were visually interesting—they made lists feel human, reports exciting, and campaigns visually recognizable—but they took up lots of space, and the information density was poor. They were attractive, but not practical.

We took a step back and looked at the way information was presented on the dashboards at the time. It wasn't bad: Each slat was easy to read. The actions were clear, and the content could be easily scanned. We realized that this pattern could be improved with just few adjustments, rather than a complete overhaul.

So we sketched each slat from the Reports, Campaigns, Lists, and Autoresponder dashboards, where we started with basic typography for titles and metadata and slowly added other elements like buttons, checkboxes, and status indicators. After a few sketch iterations, we were ready to move to Photoshop.

Creating the initial version of each slat was easy because we had already established patterns for each element. The most troublesome slat to design was on the Campaigns page: We wanted the status indicator to be visually interesting, but we also had to consider other factors, such as content hierarchy and color-blindness. After several iterations, we finally arrived at an idea where we used icons to indicate the status for each campaign and a color to reinforce the icon.

After we built them and made a few adjustments in the browser, we've hardly touched those slats since the redesign. In fact, they work so well that we've implemented them on other pages throughout the app, adding them to the Templates dashboard and File Manager when we redesigned those sections. Since the content on those pages is similar, it made sense to apply a consistent style across the entire app.

Of course, the slats weren't perfect. After the initial release, our customers offered feedback. We listened and responded. For instance, before the redesign the dashboard showed the total clicks and opens; after the redesign, the slats on the reports dashboard showed only the open and click rate for each campaign. After users spoke up about the change, we

compromised by showing the total number of clicks and opens when the user hovered over the click and open rates.

Creating and refining slats is just one example of iteration within our redesign. The same process continues for each piece of the pattern library and, in turn, each workflow and page within MailChimp.

Today when we make changes within the app, we continue to listen closely to the customer feedback that started us down this path. Sometimes there are cheers, sometimes there are moans. Either way, we learn something new about the changes we've made and we iterate on that feedback. In the end, the dialogue between the user and our team makes this process more effective—and, hopefully, more gratifying for the user.

REFINEMENT

Refining via Redesign

TYRICK CHRISTIAN

In November of 2012, I was asked to re-imagine MailChimp as a responsive application. Since so few applications as big as MailChimp are responsive, my sources of inspiration were somewhat limited. I packed the few items on my desk and moved from our UX office to a room with our [DesignLab](#), marketing team, and a few dry-erase boards. Here, our creative director, [Ron Lewis](#), asked me to begin by researching, exploring, and designing with no limitations.

This was my first responsive project, MailChimp is a large application, and I was the only UI designer on the team. How was I going to tackle this?

First, I needed to do some research. I started by re-reading [Ethan Marcotte's Responsive Web Design](#), browsing [Brad Frost's collection of responsive design patterns](#), researching responsive grids, and looking at examples of responsive sites. I also posted design inspiration to our office walls, covering them with examples of illustration styles, typography, and responsive patterns.

After doing all this research, I realized that it would be easier to begin the process by redesigning the MailChimp pattern library—a system of components, rules, and interactions that are used to build and design the application.



Variations of basic components in the pattern library.

New Typeface Friend Helvetica Neue

Headline

Subheadline

Repellat dolorum rerum neque provident omnis labore ea voluptatibus. recusandae autem sed voluptas sequi eos consequatur sunt praesentium vitae neque nisi eveniet et. odit minima saepe excepturi eos beatae est eveniet dicta maiores. a autem aliquam ut repellendus sequi beatae vitae quasi ut vel dolorum eum. et voluptatum consequatur ducimus dolor est nesciunt deleniti corrupti rem voluptates

Quia animi voluptatibus sed totam sint assumenda et voluptates eum beatae ditiis. saepe nulla omnia maiores sapiente est

New Typeface Friend f Helvetica Neue

Headline

Subheadline

Repellat dolorum rerum neque provident omnis labore ea voluptatibus. recusandae autem sed voluptas sequi eos consequatur sunt praesentium vitae neque nisi eveniet et. odit minima saepe excepturi eos beatae est eveniet dicta maiores. a autem aliquam ut repellendus sequi beatae vitae quasi ut vel dolorum eum. et voluptatum consequatur ducimus dolor est nesciunt deleniti corrupti rem voluptates

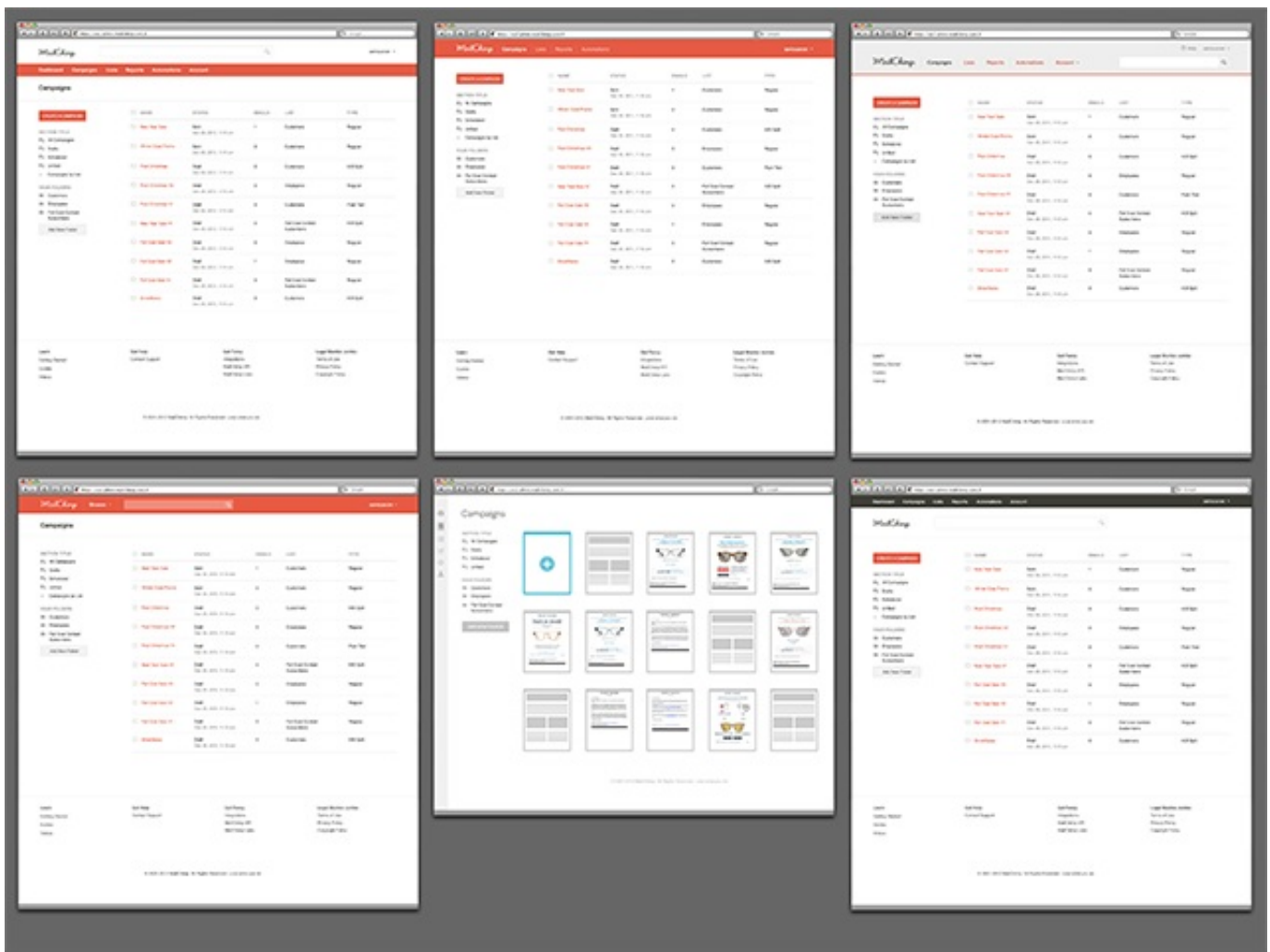
Quia animi voluptatibus sed totam sint assumenda et voluptates eum beatae ditiis. saepe nulla omnia maiores sapiente est

Typography samples.

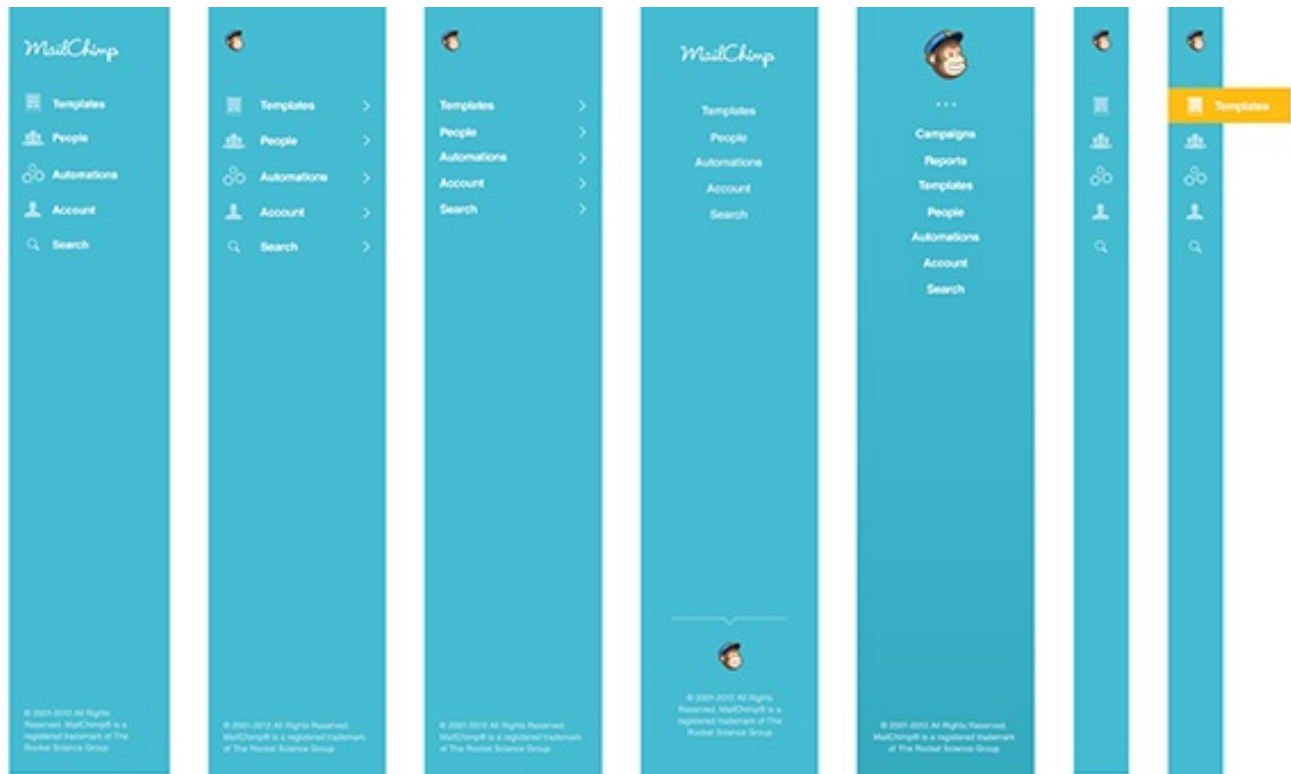
Once we all agreed on a small set of patterns, I started designing the navigation, which would give our team the framework for the app. The goal was to make the navigation flexible enough to work on all devices, support some version of co-branding, and make search a primary action so our users' data is always within their reach.

After trying several variations of horizontal navigation, I realized it didn't scale well across devices. I eventually tried a vertical navigation which worked well for every display size

because the simple column of navigation on large screens could easily hide in a menu on smaller displays. After realizing this, I tried several more variations, the team agreed on a direction, and the navigation was passed to the developers so we could see a working prototype.

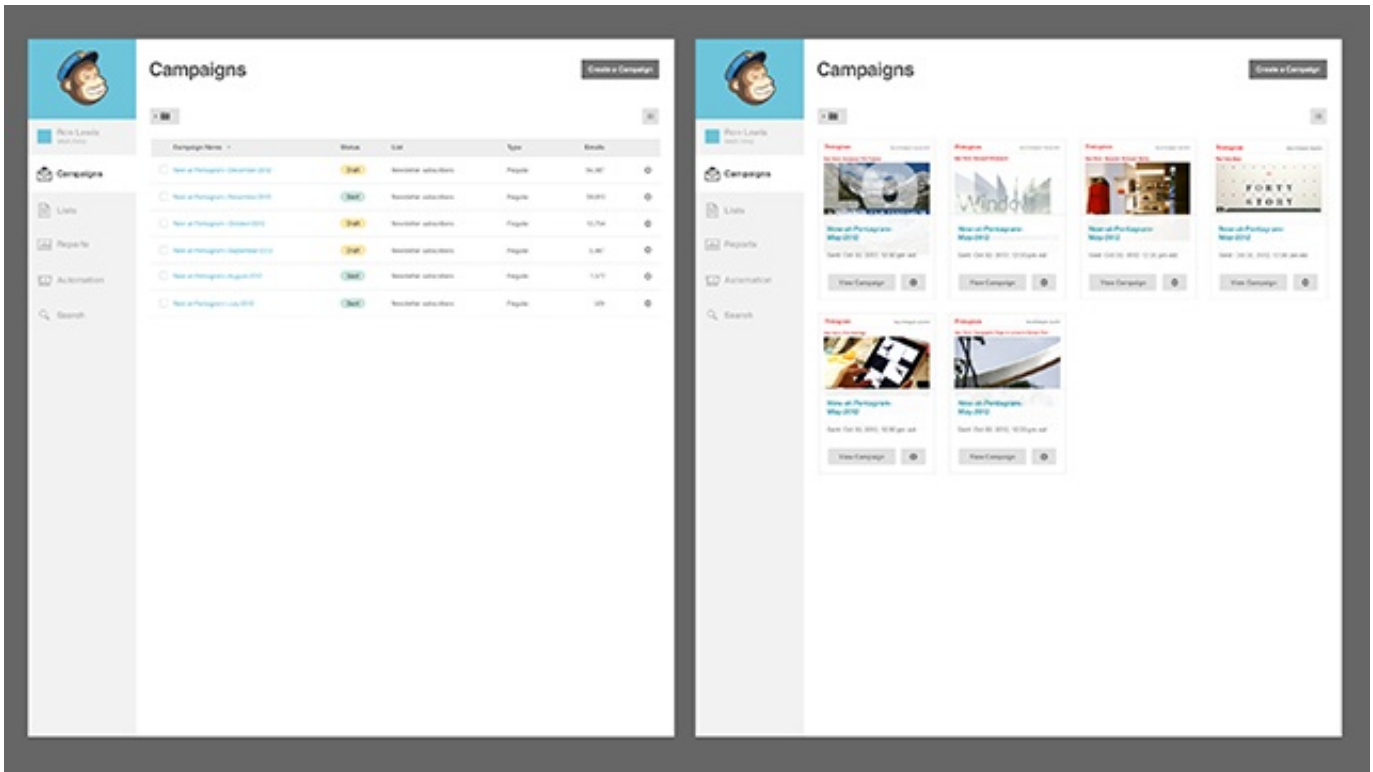


Examples of horizontal navigation from early in the redesign process.

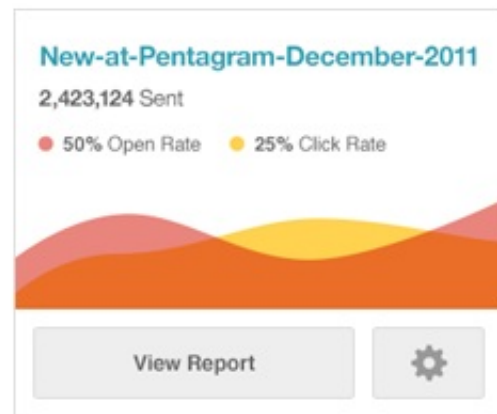
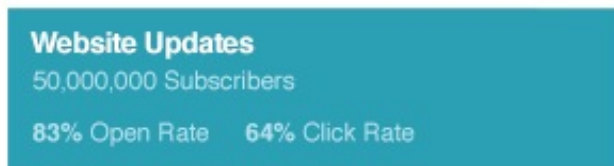
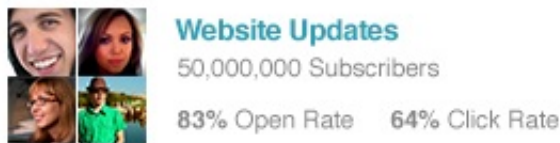


Explorations of design treatments for vertical navigation.

At this point, I started experimenting with different ways to display content in MailChimp. After working on the pattern library, I was thinking about grids and modules that could be reshaped for different screen sizes. My first idea was to create a grid view and a list view for the dashboard pages—the grid view would give new users a beautiful visual experience, while the list view would provide power users with a simple and efficient table of information.



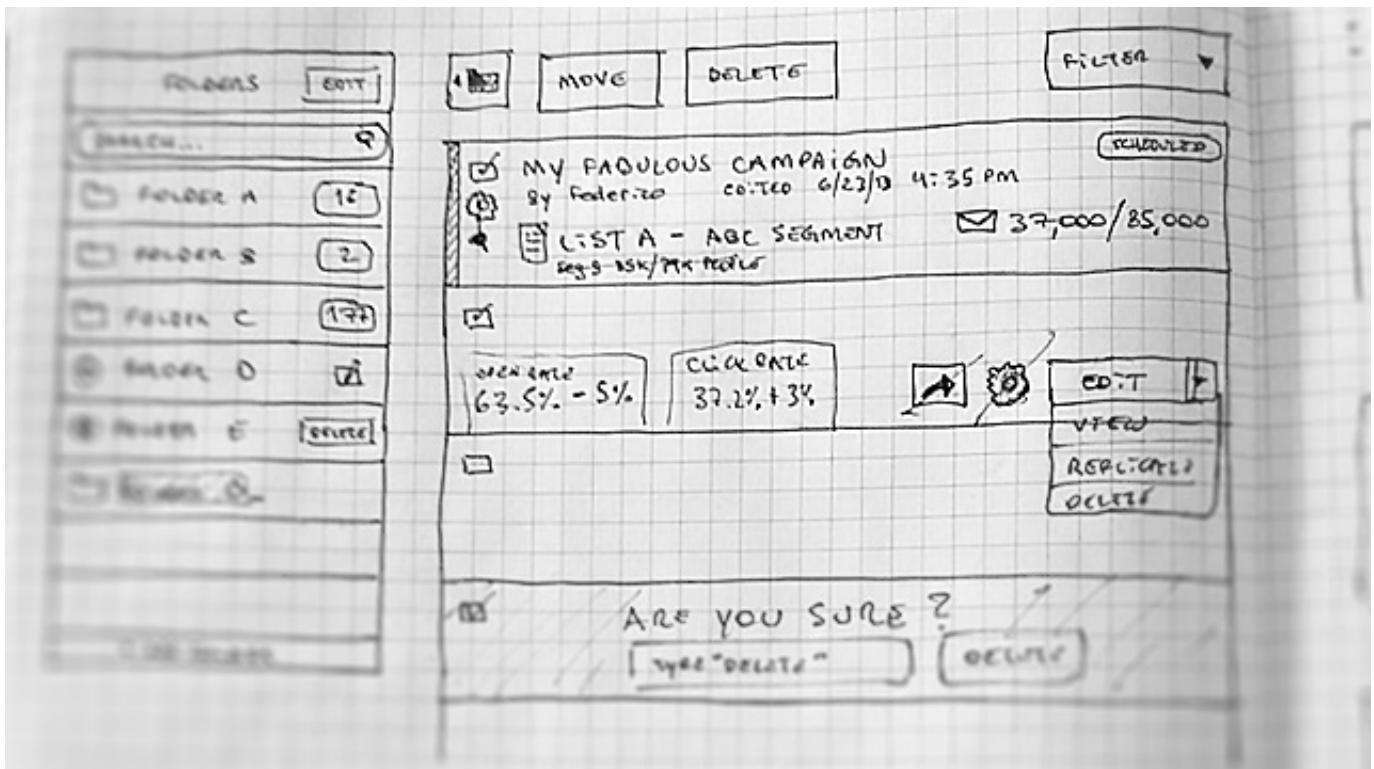
An example of grid and list view on the campaign dashboard.



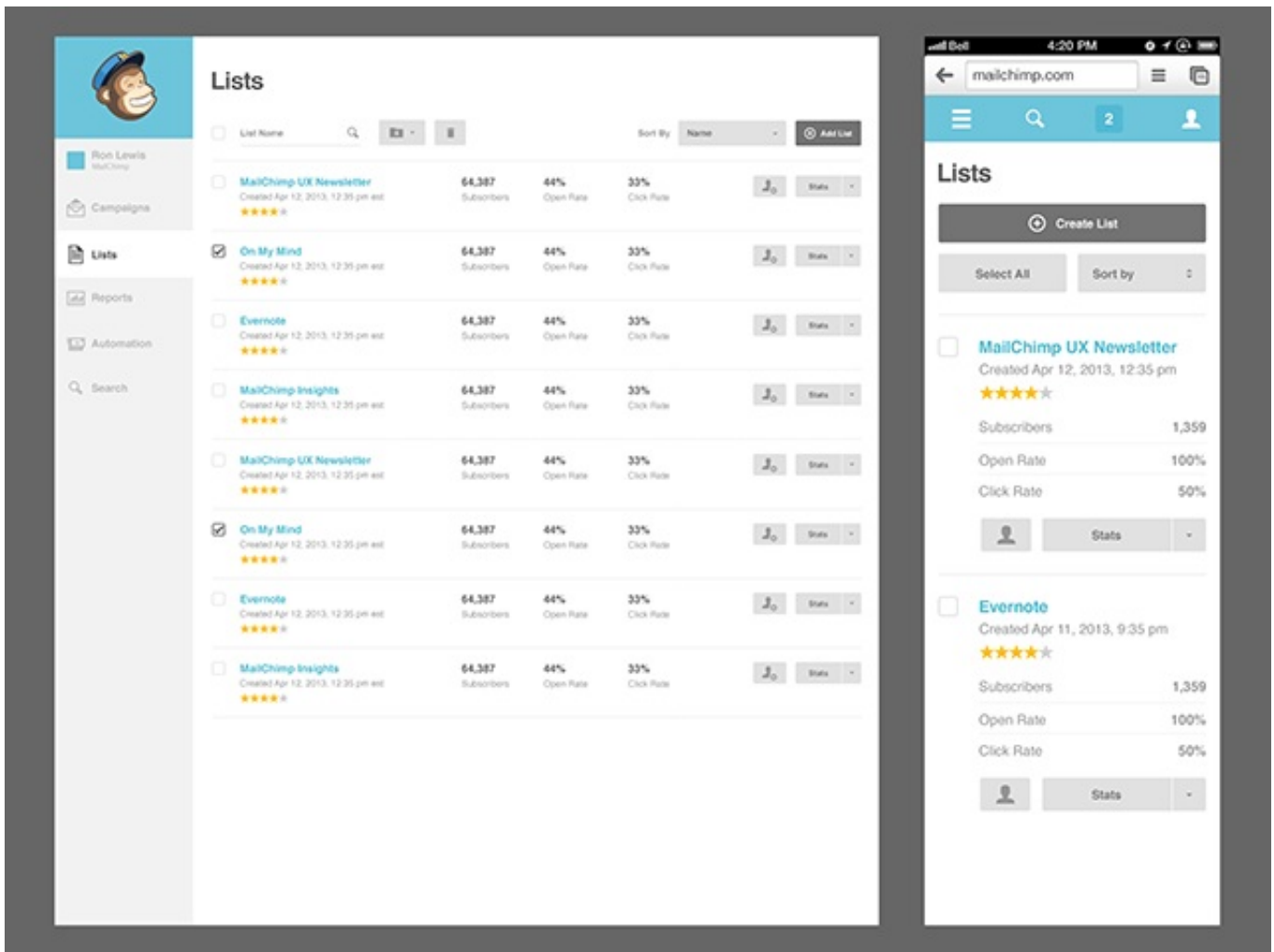
Samples of tiles for the grid view on the campaign, lists, and reports dashboards.

I experimented with tile treatments for the grid view for weeks while other members of the team were building prototypes of the pattern library. We wanted lists to feel more human, but that particular direction required subscriber avatars, which we weren't prepared to add to the app. We thought about including a display of each customer's campaigns, but that would only be useful if each email looked noticeably different from the rest, which isn't always the case. In the list view, responsive tables weren't attractive and didn't scale well for smaller screens.

After weeks of experimentation and discussion, we decided to change course. I discussed the previous designs with Federico, and we agreed on a few points. We liked the idea of an efficient table, and agreed that the content should be sortable, scannable, and have the ability to grow vertically and horizontally. After realizing this, it was clear that we needed lists styled as tables, or “slats.” (Fed talked about these slats earlier.) We quickly developed these ideas from sketches to designs to prototypes.



Federico's sketches.



Post-sketch designs were passed to other members of the team for prototyping.

While the prototypes of the lists, reports, campaigns, and autoresponders pages were being developed, I moved on to design some interior pages. After designing the pattern library, this was much easier than I expected: Most of the heavy lifting was done, and I only had to design small pieces of a few pages.

We went through countless iterations to get to the final design of New MailChimp. We quickly abandoned ideas that looked

great but ultimately didn't serve our users' needs. It's sometimes hard to let go of ideas you're invested in and move in a new direction, but that's just part of the creative process. Big, collaborative projects like this will certainly teach you humility.

Redesigning MailChimp was an invaluable learning experience that forced us to examine our design process and how it could be improved. We learned how to more effectively **collaborate with other teams**, and think about the user experience holistically. New MailChimp is now live, but there's still work to be done. That's part of what we love about working on apps—there's always room for improvement.

Resources

BY THE MAILCHIMP CREW

- [The UX Newsletter](#)
- [@MailChimpUX on Twitter](#)
- [MailChimp's Pattern Library](#)
- [DesignLab Blog](#)
- [Voice & Tone](#)
- [The Principles of Beautiful Web Design](#) – Jason Beard
- [Nicely Said](#) – Nicole Fenton and Kate Kiefer Lee
- [Data Smart](#) – John Foreman
- [Designing for Emotion](#) – Aarron Walter
- [Connected UX](#) – Aarron Walter

BOOKS WE LOVE

- [Remote Research](#) – Nate Bolt and Tony Tulathimutte
- [The Elements of Typographic Style](#) – Robert Bringhurst
- [Just Enough Research](#) – Erica Hall
- [Universal Methods of Design](#) – Bruce Hanington and Bella Martin
- [Don't Make Me Think](#) – Steve Krug
- [Universal Principles of Design](#) – William Lidwell, Kritina Holden, and Jill Butler
- [Interviewing Users](#) – Steve Portigal
- [Storytelling for User Experience](#) – Whitney Quesenbery and Kevin Brooks
- [A Project Guide to UX Design](#) – Russ Unger and Carolyn Chandler
- [Web Form Design](#) – Luke Wroblewski

HANDY TOOLS AND REFERENCES

- [JankFree.org](#): Let's make the web silky smooth!
- [A11Yproject.com](#): The Accessibility Project
- [The Google Ventures Library](#)
- [User Interface Engineering Blog](#)
- [PatternLab](#)
- [User Onboarding](#)
- [Jobs To Be Done](#)
- [Invision](#)
- [What fuels great design](#) – Braden Kowitz
- [10 Usability Heuristics for User Interface Design](#) – Jakob Nielsen

Contributors

WRITERS

Caleb Andrews, UI designer

Jason Beard, senior UX developer

Gregg Bernstein, senior design researcher

Fabio Carneiro, email developer

Tyrick Christian, UI designer

Fernando Godina, design researcher

Federico Holgado, lead UX developer

June Lee, design researcher

Alvaro Sanchez, UX developer

Steph Troeth, design researcher

Mardav Wala, UX developer

Aarron Walter, UX lead

Laurissa Wolfram-Hvass, design researcher

Allison Urban, software engineer

EDITORS

Kate Kiefer Lee

Rachael Maddux

Austin L. Ray

ART DIRECTOR

David Sizemore

ILLUSTRATORS

Jane Song

Cover

This Paper Ship

Collaboration

Pam Wishbow

Research

Karen Kurycki

Design

Vaughn Fender

Development

Lydia Nichols

Refinement