



## designing the obvious



a common sense approach to web & mobile application design  
second edition

robert hoekman, jr.

**Designing the Obvious:****A Common Sense Approach to Web and Mobile Application Design, Second Edition**

Robert Hoekman, Jr.

**New Riders**

1249 Eighth Street  
Berkeley, CA 94710  
510/524-2178  
510/524-2221 (fax)

Find us on the Web at [www.newriders.com](http://www.newriders.com)

To report errors, please send a note to [errata@peachpit.com](mailto:errata@peachpit.com)

New Riders is an imprint of Peachpit, a division of Pearson Education

Copyright © 2011 by Robert Hoekman, Jr.

Editor: Wendy Sharp

Production Coordinator: Hilal Sala

Copyeditor: Jacqueline Aaron

Research Assistant: Sunny Thaper

Compositor: Danielle Foster

Indexer: Emily Glossbrenner, FireCrystal Communications

Cover design: Mimi Heft

Interior design: Joan Olson

**Notice of Rights**

All rights reserved. No part of this book may be reproduced or transmitted in any form by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. For information on getting permission for reprints and excerpts, contact [permissions@peachpit.com](mailto:permissions@peachpit.com).

**Notice of Liability**

The information in this book is distributed on an "As Is" basis, without warranty. While every precaution has been taken in the preparation of the book, neither the author nor Peachpit shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the instructions contained in this book or by the computer software and hardware products described in it.

**Trademarks**

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Peachpit was aware of a trademark claim, the designations appear as requested by the owner of the trademark. All other product names and services identified throughout this book are used in editorial fashion only and for the benefit of such companies with no intention of infringement of the trademark. No such use, or the use of any trade name, is intended to convey endorsement or other affiliation with this book.

ISBN 10: 0321749855

ISBN 13: 9780321749857

9 8 7 6 5 4 3 2 1

Printed and bound in the United States of America

# Contents

**Acknowledgments** x

**Author Biography** xi

<b>Chapter 1: Defining the Obvious</b>	<b>3</b>
What Is ‘the Obvious’?	6
Qualities of a great application	8
How Do You Design the Obvious?	10
Turn qualities into goals	10
The Framework for Obvious Design	12
Know what to build	14
Know what makes it great	14
Know the best way to implement it	15
<b>Chapter 2: Lead with Why, Follow with What</b>	<b>17</b>
Know Your Motivation	19
What follows Why	22
Make Authentic Decisions	24
Audit the user experience	24
Define the vision	28
Plan the new design	31
Implement it	32
Measure everything	32
Having vision	34
<b>Chapter 3: Ignore the User, Know the Situation</b>	<b>35</b>
Designing for the User	37
Designing for the Activity	44
Solve for the Situation	47
Understand How Users Think They Do Things	55
Understand How Users Actually Do Things	57
Find Out the Truth	61
Contextual inquiry	63
Remote user research	66
Surveys	67
Write Use Cases	68

Task-flow diagrams	74
My advice	76
<b>Chapter 4: Build Only What Is Absolutely Necessary</b>	<b>77</b>
More features, More frustration	78
So what's a geek to do?	79
Think Different	80
The dashboard and New Invoice screen	81
The finished invoice	82
The result	83
Think Mobile	84
Hey, it's your life	87
Not present at time of photo	87
Drop Nice-to-Have Features	88
The Unnecessary Test	89
The 60-Second Deadline	90
Aim low	92
Interface Surgery	93
Reevaluate nice-to-have features later	98
Let them speak	99
<b>Chapter 5: Support the User's Mental Model</b>	<b>101</b>
Understanding mental models	103
Design for Mental Models	104
Making metaphors that work	107
Interface Surgery: Converting an implementation model design into a mental model design	111
Eliminate Implementation Models	119
Create wireframes to nail things down	119
Prototype the Design	127
Test It Out	130
<b>Chapter 6: Turn Beginners into Intermediates, Immediately</b>	<b>139</b>
Use Up-to-Speed Aids	141
Provide a welcome screen	145
Fill the blank slate with something useful	147
Give instructive hints	149
Interface Surgery: Applying instructive design	153

Choose Good Defaults	160
Integrate preferences	163
Design for Information	163
Card sorting	166
Stop Getting Up to Speed and Speed Things Up	168
Reuse the welcome screen as a notification system	169
Use one-click interfaces	170
Use design patterns to make things familiar	171
Provide Help Documents, Because Help Is for Experts	173
<b>Chapter 7: Be Persuasive</b>	<b>175</b>
Draw a Finish Line	176
Ownership	177
Solve a Significant Problem	178
Make It Explainable	180
Know Your Psychology	181
Reciprocity	181
Commitment and consistency	182
Social proof	184
Authority	185
Liking	186
Scarcity	187
Ethical persuasion	188
<b>Chapter 8: Handle Errors Wisely</b>	<b>189</b>
Prevent and Catch Errors with Poka-yoke Devices	191
Poka-yoke on the web	192
Prevention devices	193
Detection devices	195
Turn errors into opportunities	200
Feeling smart	202
Ditch Anything Modal	203
Redesigning rude behavior	204
Replace it with modeless assistants	205
Write Error Messages That Help Instead of Hurt	207
Interface Surgery	209
Create Forgiving Software	211
Good software promotes good practices	213

<b>Chapter 9: Design for Uniformity, Consistency, and Meaning</b>	<b>217</b>
Design for Uniformity	220
Be Consistent Across Applications	229
Understanding design patterns	230
Intelligent inconsistency	233
Leverage Irregularity to Create Meaning and Importance	234
Interface Surgery: Surfacing the bananas in a process	237
<b>Chapter 10: Reduce and Refine</b>	<b>243</b>
Cluttered task flows	244
The path to simplicity	245
Clean Up the Mess	246
Reducing the pixel-to-data ratio	247
Minimizing copy	248
Designing white space	251
Cleaning up task flows	255
Practice Kaizen	258
The 5S approach	259
Eliminate Waste	262
Cleaning up your process	263
Put Just-in-Time Design and Review to Work	265
<b>Chapter 11: Don't Innovate When You Can Elevate</b>	<b>269</b>
Innovation	270
The problem with innovative thinking	270
Elevation	272
Elevate the User Experience	272
Elevation is about being more polite	273
Elevation means giving your software a better personality	274
Elevation means understanding good design	276
Seek Out and Learn from Great Examples	277
Inspiration	278
Elevate the standards	278
Take Out All the Good Lines	279
Get in the Game	280
<b>Index</b>	<b>283</b>

*This page intentionally left blank*

# 4

## Build Only What Is Absolutely Necessary

- ▶ Think Different
- ▶ Think Mobile
- ▶ Drop Nice-to-Have Features

When applications evolve based on the demands of users (or of CEOs), they tend to take a bad turn. Features used by only 10 percent of users or used only 10 percent of the time are added and get in the way of the remaining 90 percent of features. They clutter an otherwise clean interface. They interfere with the features used most often.

And when “featuritis” takes over, you quickly find yourself permanently providing tech support for things that shouldn’t be in the tool to begin with, fixing more bugs, writing more Help material, and neglecting other, more important features. And while this may sound like a lot of fun to certain (slightly crazy) programmers, it’s clearly the wrong approach.

The focus should not be on features, the focus should be on *focus*. An obvious application is a focused application. It’s easy to explain to other people. It makes sense to those using it because the purpose of the tool is self-evident, and nothing in it strays from that purpose. Every feature supports the single situation the application is designed to support.

## More Features, More Frustration

A user’s frustration level doesn’t map directly to the number of difficult features in an application. Frustration increases *exponentially*. For every additional feature, there is more to learn, more to tweak and configure, more to customize, more to read about in the Help documentation, and *more that can go wrong*.

For you, it’s one more feature. For users, it’s one more thing that adds to the already long list of frustrating things to deal with while using a computer. It’s not just your application—it’s everything else, too. It’s the operating system, which hides files away in obscure directories and is constantly popping up little dialog boxes and error messages that you are forced to address before you can get on with your real work. It’s the browser, which has no graceful way of indicating whether or not the link you just clicked is going to show you what you want to see. It’s the email client, which offers no insights into how long it will take for the message you just wrote, with the rather large photo of your cat, to be sent to your grandmother.

Users contend with all these things and more during the same stretch of time they try to deal with your application. And the frustrations add up quickly.

I know, I know—none of these things bother you at all. They don't really bother me either. But that's a sad fact. It means we've become desensitized to things that are otherwise maddening. It means we've gone numb.

In short, we've become "computer-savvy."

## So what's a geek to do?

You have to have killer features or your application won't be able to stand up to its competitors. Right? You have to keep adding things to new versions or no one will purchase upgrades and sales will stagnate. And you have to match the competition one-for-one so no one can ever say your application is light where the other guy's is robust. *Right?*

To paraphrase Alan Cooper, however, trying to match competing products feature-for-feature is like running through a battleground under cover fire. You can run all you want, but you have to keep shooting to get anywhere. Dishing out cover fire keeps you alive for a few minutes at a time. Long enough to hide. Companies that fight all the time to stay ahead fall into the endless cycle of trying to outdo the enemy (if the enemy has a big gun, you need a bigger gun). This goes on and on until someone falls. It's not a fun way to do things. It's a method that works only as long as the people fighting the battle continue to come up with bigger guns. They spend all their time spraying out cover fire while they run 3 feet to the next safe position.

Many companies live and die this way. To get into the fight, you have to stock up on venture capital, go into major debt, hire a bunch of rock star developers, go straight to code because there's no time to plan or design anything, and rush, rush, rush to market with a 27-page list of features. And if the enemy starts to catch up, you have to add more features, call the next version "the most robust release ever," and try to maintain your market share. Until, of course, the enemy puts out a new version with even *more* features.

It's exhausting.

It's also exhausting for users. The more features you offer, the more the user has to learn. The more options you provide, the more users have to do to get anything done. The more you allow customization, the more users have to fidget and tweak and manipulate your application. They spend more time configuring the tool than using it. As a result of fighting the fight, complicated applications often end up much less usable than one would hope.

To stay alive, you eventually have to get out of the line of fire. It's the only real option.

## Think Different

A few years ago, Firewheel Design ([www.firewheeldesign.com](http://www.firewheeldesign.com)) got out of the line of fire by creating Blinksale ([www.blinksale.com](http://www.blinksale.com)), a web-based invoicing system. The simple application contains only the features that are absolutely necessary for the largest percentage of its users to successfully create, submit, and track invoices.

Firewheel's decision to minimize Blinksale's feature list might look like a mistake because it seems as if it won't be able to compete in the rat race with the big boys of invoicing systems. But the small crew at Firewheel did something the big boys hadn't done: it created something that stood out.

Blinksale is aimed at contractors who don't need to do anything fancy with their invoices. Many people who need to submit and track invoices need only a few basic tools. These include a way to create the invoice, submit it, mark it as closed when payment is received, and perhaps send a receipt confirmation to the client. When the folks at Firewheel Design set out to create Blinksale, they realized they could keep it simple and satisfy the vast majority of user needs. They may have even realized that making it more complicated would decrease their chances of satisfying user needs. So they designed a web application that does one thing, and does it very well: it gives people a fast and effective way to create, submit, and track invoices.

(See how easy it is to explain? That's a good sign.)

Since Firewheel created Blinksale, it's been taken over by Doublewide Labs. It's even been completely redesigned. Amazingly, it's *still* one of the best applications around.

The system can be used by plenty of people besides contractors because it's so stripped down that a trained monkey could use it (assuming the trained monkey could type). The application contains just a few key features.

## The dashboard and New Invoice screen

When you sign in, Blinksale shows you a summary of your recent activity (open invoices, past-due invoices, and so on) so you get a quick, at-a-glance, dashboard-style view of the state of your invoices. It also offers an easy-to-spot New button, to start creating a new invoice.

You simply choose the client the invoice is for or create a new one—right there, on the same page—and hop over to the New Invoice screen. This page actually *looks* like a real invoice, so you maintain context the whole time you're creating it. All the fields you need to complete are displayed as form elements, so you can simply edit the invoice onscreen and click the big Save button.

**Edit Invoice**

Date: 2011 January 3  
 Invoice ID: 0010 (Last used: 0010)  
 P.O. Number:

Unit	Qty	Description	Price	Total
Service	1	Some really hard work	430.00	\$430.00
Service	1	That special thing I did last week	200.00	\$200.00
Service	1	The new, wicked-cool feature	560.00	\$560.00

Subtotal: \$1190.00  
**Total Due: \$1190.00**

**CLIENT**  
 The Biggest Client Ever  
 123 Mondo Street  
 Big Town, BG 33333  
 US

**INVOICE DETAILS**  
 Currency: U.S. Dollars  
 Tax & Freight:  
 Sales Tax  
 Freight  
 Payment is Due:  
 15 days (January 3, 2011)  
 Days from invoice date

Blinksale's main invoice-editing screen is easy to use.

When you're done, you see the final version of the invoice and a few new buttons, which let you send the invoice, edit it, or delete it. One click of the Send Invoice button produces an in-line form in which you checkmark all the people in the client company to whom you want to send the invoice and write an optional message.

## The finished invoice

The invoice itself is an HTML-formatted email that looks great right out of the box (well, the browser), and you don't have to configure anything at all to send off a professional invoice to a client in five minutes or less.

The screenshot shows the Blinksale software interface. At the top is a navigation bar with tabs for DASHBOARD, ESTIMATES, INVOICES (selected), PURCHASES, RECURRING, and PEOPLE. A notification in the top right corner says '249 Invoices left'. Below the navigation bar is a green banner with a checkmark and the text 'Your invoice has been saved.' The main content area displays 'Invoice 0010' for 'The Biggest Client Ever' at '123 Mondo Street, Big Town, BG 33333, US'. The due date is '03 Jan 2011' with the note 'Due 03 Jan 2011 (Due Upon Receipt)'. A table lists three services: 'Some really hard work' for \$430.00, 'That special thing I did last week' for \$200.00, and 'The new, wicked-cool feature' for \$560.00. The subtotal is \$1,190.00 and the total due is \$1,190.00. On the right side, there is a 'NEW INVOICE' button, a 'Send Invoice' button, and a list of actions: Edit Invoice, Print Invoice, Change Status to Open, Duplicate Invoice, and Delete Invoice. Below these is a 'TAGS' section with an 'Add tags' button. At the bottom left, there is a 'History & Comments' section with an 'Add Comment' button.

Qty	Unit	Description	Price	Total
1.0	Service	Some really hard work	\$430.00	\$430.00
1.0	Service	That special thing I did last week	\$200.00	\$200.00
1.0	Service	The new, wicked-cool feature	\$560.00	\$560.00
Subtotal:				\$1,190.00
Total Due:				\$1,190.00

Blinksale generates easy-to-read invoices and lets you email them to your clients in a click.

Simple as that.

Blinksale offers a few basic templates from which to choose how you want your invoices to look. You can also send reminders to clients about late payments and create thank-you messages to send to clients who pay their bills on time.

The whole application takes less than 30 minutes to learn inside and out, and just about pays for itself every time you create an invoice (at the time of this

writing, Blinksale offers a \$6 per month plan for up to 6 invoices, with plans of up to \$24 per month for 250 invoices.)

Firewheel built only what was absolutely necessary for most people to successfully handle the activity of invoicing clients. And Doublewide Labs has lovingly maintained that tradition. There are no obscure configuration options, no redundant functionality (there's exactly one way to complete each task in the tool, which makes it easy to learn), and no fancy interface widgets to figure out. It just does exactly what it should, and does it within a simple, clean interface that somehow makes invoices seem friendly, like someone you'd want to take to lunch. (We'll talk more about software personality in Chapter 10.)

## The result

Josh Williams, one of the creators of Blinksale, was justifiably proud of how things turned out. Back in 2004, he told me:

*As a small design company we did our fair share of client billing. Unfortunately we've always been less than enamored with the off-the-shelf invoicing and billing software that is available at your local office supply store. After a few years of frustration we set out to build our own web-based invoicing service. Goal number one was ease of use. Goal number two was keeping our cost of design and development of the service low. Remarkably, these two goals often go hand in hand.*

Firewheel could have designed Blinksale to be chock-full of features that did everything from integrate with Intuit QuickBooks in 12 easy steps to preparing tax information and letting you export it to Intuit TurboTax at the end of the year. They could have built a product that rivaled its competition feature for feature. They didn't. They built the 20 percent people actually need. Nothing more. Nothing less. Even after a change in ownership, with Doublewide Labs at the helm, Blinksale is still hyper-focused on only the most essential features.

While there are a few extra gadgets thrown in for more computer-savvy users, Blinksale keeps things simple and focused. If all you want to do is create

an invoice and send it to a client—the single task most people will spend most of their time completing in Blinksale—you can do it in just a few minutes and be on your merry way.

## Think Mobile

One of the best ways to avoid feature battles is to focus your attention on designing for mobile platforms such as Apple’s iOS and Google’s Android. Strategically, it’s also one of the best things you can do for your business. Consider this story:

Recently, after boarding a flight to San Francisco for the Voices That Matter conference (hosted by New Riders, of course), a man in his early 30s sat next to me and pulled out his iPhone. The older gentleman who sat on the other side of him asked about it. What was it like to use it? How easy was it *really*? Wow, it sure does look fast, and neat. The younger man answered every question with growing enthusiasm. I’d seen it a hundred times before—the iPhone frequently turns otherwise perfectly jaded people into vehement, adept Apple sales representatives. But then the younger man said something that surprised me. When the older man asked the younger man what he did for work, he replied:

“I’m a cop.”

He wasn’t a designer. Or a marketing guru. Or a social media expert. Or an entrepreneur. He wasn’t heading to a tech conference or a sales seminar. He wasn’t at all the kind of person I normally see have a conversation like this one. He was a cop—a middle-class guy who puts on a blue uniform every day and relies on walkie-talkies to communicate—heading off to meet some old college friends for the weekend.

“I hardly ever use my computer anymore. I can do it all on this thing.”

Touchscreens and gestural interfaces have taken rise. Long gone are the days when the Internet was considered the new frontier. Personal tech has taken over. Devices are the *new* new frontier.

And this, my friend, is a very good thing.

Over the next few years, as the obstacles to the adoption of mobile devices are eliminated, more and more people will trade in their desktop and laptop computers and start using devices exclusively. Apple and its competitors in the smartphone and tablet markets will make sure of that. And while many people in the tech industry still see some of these gadgets as luxury items—often even wondering what on Earth they would do with a tablet—these devices are designed for the other 99 percent. They're designed for that large segment of the population that uses computers for paying bills, social networking, making plans, watching videos, checking the news, listening to music, digging up recipes, learning new skills, creating spreadsheets for work, writing memos, and of course, checking email. These people use computers primarily for media consumption, web browsing, and basic document-creation. And that's exactly what the iPad and other tablets are designed to do best.

As these products evolve and get cheaper, it will simply be more affordable and more useful to buy a touchscreen tablet backed by a catalog of cheap and easy-to-use apps, with its ever-expanding array of possible use cases, than it will be to buy a desktop, laptop, or even a netbook. More people than ever before will be able to empower themselves through the Internet, and they'll be able to use it anywhere they want. The air will be completely filled with Wi-Fi signals, and all the world's information—all *your* information—will be quite literally at your fingertips, anytime, anyplace..

Personal tech is now affordable by the masses, useful for the masses, and usable by the masses. And this will only become more true in the years to come. If you're not designing for it now, you're already late.

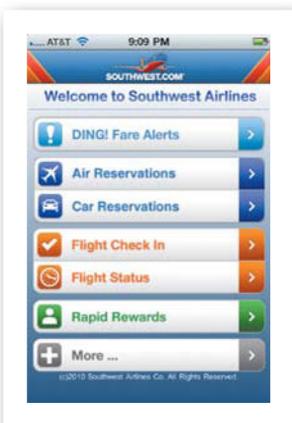
But even if you ignore this fundamental shift in personal computing, devices are good for application designers for other reasons—specifically, because they force designers to follow the principles of good application design.

The people designing for devices right now are doing a better job of embracing these principles than most people designing for the desktop *ever* have. The constraints of the medium—the limited screen space, risk of network slowdowns, difficulties of multitasking, and so on—are having the happy side-effect of encouraging designers to design more concise applications.

Southwest Airlines offers a great example. Here's the site in a desktop browser.



Naturally, the site offers a way to book a flight, car, or hotel. In fact, in this desktop version, there are two sets of tabs that offer this, one of which lets you fill out the reservation form on the homepage, the other of which takes you to another page to do it. Why? Well, because the larger tabs at the top are more than just links—they're menus. The Air tab, for example, offers a menu chock-full of links to other information, including a list of destinations Southwest flies to. And yes, plenty of people will seek out this information. But the natural effect is that the design takes up a lot of space, offers a lot of options, and requires a lot of scanning and decision-making.



The Southwest Airlines iPhone app, however, focuses only on what's absolutely necessary. On the iPhone, users can find this information by checking the To Where option on the Book a Flight screen.

Now, let's say you want to see how your Rapid Rewards stockpile is coming along. On the desktop version, there is a large Rapid Rewards tab-like link near the top of the page, an accordion tab labeled My Rapid Rewards, and a sign-in form that asks for your Rapid Rewards number. Which one is right? Which step do you take first?

In the iPhone app, you tap *Rapid Rewards*. It leads to a sign-in screen.

One of these things is so much clearer than the other.

## Hey, it's your life

So if you still wonder what on Earth you'd do with a tablet, there's your answer. You'd secure your own future as a designer. As a marketing guru. A social media expert. An entrepreneur.

Don't neglect to see the significance of mobile computing because you're busy sitting at a desk with a souped-up PC, a killer video card, and 8 feet of monitor space displaying 75 open Photoshop files. Yes, you'll have to continue doing a lot of your design work there. Yes, it will continue being *easier* to do design work there. But don't delude yourself that your customers will always and forever be sitting at a desk when they use your products. They won't be.

Use your desktop computer. Just use it to design for *devices*. And make sure a tablet is sitting next to it.

Throughout the rest of this book, I'll discuss mobile-specific considerations alongside our discussion of design principles for effective applications.

## Not present at time of photo

Sadly, at the time of this writing, there is no device-friendly version of Blink-sale. Hey, nobody's perfect.

A competing invoicing application, Ballpark, does offer a mobile-friendly dashboard. It's not much, but it offers a cursory view of your recent activity.

## Drop Nice-to-Have Features

Almost every mature application in existence contains at least a few features that were probably first described in a statement that started with “Something that would be really nice to have is <insert description here>.” But most of these things are exactly what clutter up interfaces all over the web and on our devices, and it’s our job to fend these things off with a big stick. They need to be removed from your next application before it’s even built. An obvious interface is one that is focused on what’s most important and leaves out the things that are simply nice to have.

In its book *Getting Real*, 37signals has this to say about focusing on only the important features:

*Stick to what’s truly essential. Good ideas can be tabled. Take whatever you think your product should be and cut it in half. Pare features down until you’re left with only the most essential ones. Then do it again.*

The statement is similar to something Steve Krug said in his book *Don’t Make Me Think*, one of the greatest books out there on web usability. It’s Krug’s Third Law of Usability:

*Get rid of half the words on each page, then get rid of half of what’s left.*

And Krug’s law can be traced back to William Strunk, Jr., and E. B. White’s *The Elements of Style*:

*Vigorous writing is concise. A sentence should contain no unnecessary words, a paragraph no unnecessary sentences, for the same reason that a drawing should have no unnecessary lines and a machine no unnecessary parts.*

Say it again, brother.

All these people are in the business of simplicity. Simplicity makes the point clear. It lets messages stand out. It offers communication that cuts through the noise.

## The Unnecessary Test

To create applications that cut through the noise, you have to be willing to slice your application's feature list down to its bare bones, and you have to recognize what's most important.

With that in mind, try the following exercise, which I call the Unnecessary Test:

Open an application you've worked on recently and find a feature you thought was really important a long time ago, perhaps before you started building the application.

Ask yourself the following questions:

1. Is there more than one way to complete the task this feature supports?
2. Does this feature contribute directly to the completion of the task?
3. Is the task this feature supports vital to the activity this application supports?

If you answered no to any of these questions, the feature may be unnecessary. You've found yourself a likely candidate for the cutting room floor.

If, on the other hand, you answered yes to all of these questions, you're either looking at a rock star feature or you're not looking hard enough at the feature to be objective. Try your best to detach yourself from all the work you did and ask these questions from a more objective point of view.

Regardless of your answers, it's likely there are several features in your application that could be scrapped, so you should take the time to go through every feature and run each one through the Unnecessary Test.

When you're done with the testing, close the application and ask yourself three more questions.

1. What are the circumstances of the situation my application is meant to support?
2. If this application didn't exist and I needed to handle the same situation, and I could wave a magic wand to create an application that helped me with this situation with the greatest of ease, what would the application do? (Hint: You should limit this answer to a few very big-picture statements that relate to the principal desired outcome.)
3. How long will it take to rebuild my application to make it do that?

Sorry—that last question is a joke (sort of). After all, you're likely to have answered one of the first two questions in a way that prevents you from having to admit you were wrong. I know—I've done this myself. It's difficult to admit your application may not be living up to its promise.

If this is true, have someone else answer the same set of questions and see if the answers are different. Even better, ask one of your users.

I'm not suggesting you start ripping functionality out of an existing application. Doing this could have the rather negative side effect of making some of your users extremely upset. To the people using the more obscure features, removing them would be a huge mistake. I'm only suggesting you learn from what you've already done so you can create more focused applications in the future.

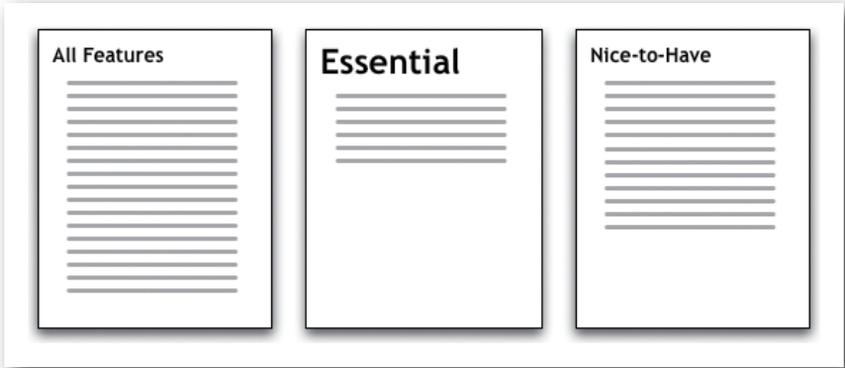
## The 60-Second Deadline

Here's another quick way to learn to effectively aim low and keep your application focused on the 20 percent that matters:

Pretend I'm your boss. I walk into your office and very matter-of-factly state, "The project time line has been cut in half. We have about 60 seconds to decide what to keep and what to throw away before we meet with the client in the conference room."

How do you respond to this statement?

Whatever you do, don't impulsively offer up the theoretical answer—the one where you say how much you'd love the low-carb sandwich. *Figure out the real answer.*



Grab a notepad and a pen, write down the list of features you have planned for an upcoming application, and see what you can cut in 60 seconds. Draw a line through each feature you can cut without completely destroying the application.

The goal is to leave yourself only with what is most essential for the application to serve its purpose.

Bells? Gone.

Whistles? Gone.

Show me only the pieces you absolutely have to keep for the tool to do its job.

When you're done, cut one more feature, just for good measure. Cut the one you're holding onto only because it's really cool. C'mon, I know there's at least one on your original list. Draw a line through it.

Your 60 seconds are up. Good job.

Now, take out a second sheet of paper and write a new list that shows only what you have left, just so you can see it sitting there all nice and clean. Looks much better, doesn't it? I know, it probably hurts a bit to have lost so much stuff, but I bet your application is now easier to explain.

Finally, take out another sheet of paper and write down the list of things you drew a line through earlier. Title this page "Nice-to-Have Features," stick it in your filing cabinet, and forget about it. We'll look at it again later.

The first time you do this, it can be quite revealing. You may find you've been wasting a lot of your time and energy on things that don't really contribute to the application in any meaningful way. Of course, this may be a bit unsettling, but hey, knowing is half the battle. Next time around, you can use the Unnecessary Test and the 60-Second Deadline exercise before you start coding, to see what really needs to be built—and you can spend all your time working to make those things as good as they can be.

And since building what's most important takes much less time than building what's not important, you can get more sleep, take more vacations, get more weekends off, and live a happier, healthier life.

Or you could do what I do and use all that saved time to design more applications. I know that's what you *really* want to do.

## Aim low

Regardless of how you do it, the ultimate goal is to determine what's most important to the application by whittling your list of features down to about 20 percent of what was built or what you were planning to build. Yes, some of the remaining 80 percent of your features may be useful somehow, to someone, some of the time, but they are most likely useless to 80 percent of your users, 80 percent of the time. And you probably spent 80 percent of your development time building things that aren't essential to the application.

This is because the 80-20 rule has made its way into the world of software.

Known formally as the Pareto principle (named for Vilfredo Pareto), the 80-20 rule was originally suggested to indicate that 80 percent of the effects come from 20 percent of the effort.

In terms of good, clean application design, it means that 80 percent of an application's usefulness comes from 20 percent of its features. It also works the other way around, to illustrate that 20 percent of the development work produces 80 percent of an application. The other 80 percent of the work satisfies only 20 percent of the outcome.

To create more focused applications, stick to building the 20 percent of features that are essential—the ones you'll stick on the mobile version—and you'll

take care of 80 percent of the user's needs. Let your competitors worry about the rest. While they're floundering around trying to one-up you by fleshing out the other 80 percent of the application, you could be taking 80 percent more vacations and enjoying 80 percent of the market share.

Less is more. Aim low.

## Interface Surgery

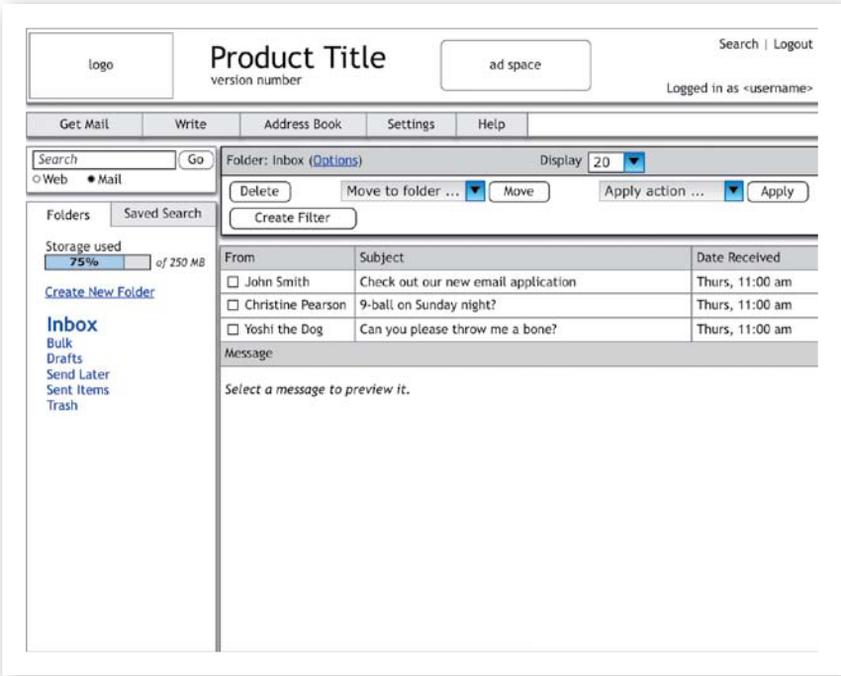
A job application form I saw once was composed of two windows. One window got the user through the first few screens of the process, and then it launched a second window to complete the bulk of the application. The first window was connected to the user's log-in session, which was timed and was designed to log out the user automatically if the system remained inactive for 20 minutes. However, the second window was not tied to the session. So, when a user tried to complete the job application in the second window—the part of the process that took the longest amount of time—the system invariably logged the user out after 20 minutes, rudely doing so without any notification whatsoever.

The company's solution was to add a bit of text in the original window warning users that they would be logged out after 20 minutes—a weak attempt to get their pesky users to stop complaining. This was a band-aid. It did not solve the problem, it just told people what to expect. Users would still have to complete the job application in 20 minutes or less. The company was essentially saying, "Sure, we've created a terrible system that will likely terminate your session before you can complete your job application, but hey, we're warning you before you start, so it's okay!"

I don't like band-aids.

Instead of putting band-aids on problems, I perform surgery on them. Interface surgery.

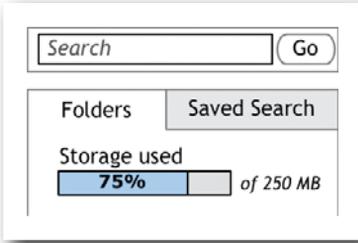
In this first installment of Interface Surgery, we'll cut out a bunch of unnecessary features from a fictitious web-mail application. Instead of finding ways to make a ton of unnecessary gadgets easier to present and use, we're going to rip them out and leave only what's absolutely essential for the application to do its job.



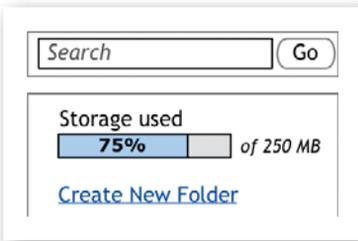
This application has a ton of features. In addition to being able to simply check your email, you can search the web, see how much storage space you've used, make sure you're logged in using a particular user name, reuse saved searches, apply actions (such as set up an automatic response email), move email to other folders you create yourself, configure options for the Inbox (such as font settings), and even change how many messages should be displayed in the list before having to switch to a new page.

Some of these things are necessary, some are not.

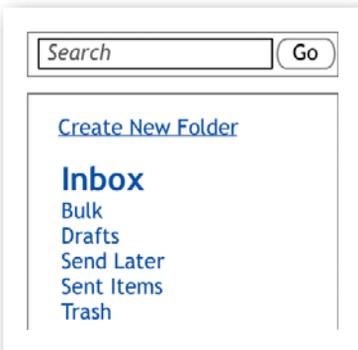
To get started, let's strip out the part of the Search feature that lets users search the web. There are already plenty of ways to search the web, and most modern browsers feature a built-in search bar, making this action accessible 100 percent of the time the user has the browser open. There's no need to replicate what's already ubiquitous. And since we're leaving only the option to search mail, we can remove the two radio buttons and shrink down the space this piece takes up.



Let's also get rid of the ability to save searches. It's more difficult to save a search, find it again later, and rerun it than it is to simply reenter a few keywords. This might be nice for some users, but it's not going to seriously benefit *most users, most of the time*. And since we're getting rid of it, we can lose the tabbed interface that displays it. Since the Folders view is now the only option, it no longer needs a label or a tab.



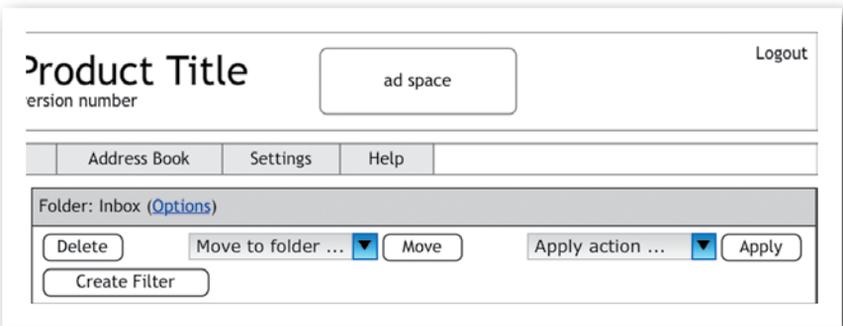
Next, let's get rid of the percentage indicator that tells users how much storage space has been used up. If we decide this is essential later, we can move it into the Settings screen. There's no reason to give it a permanent position in the main interface.



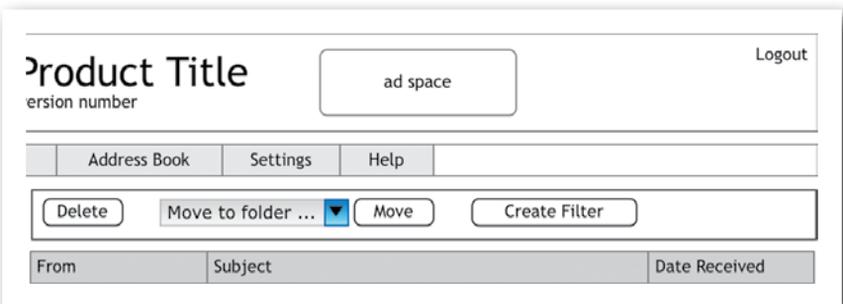
Next, let's get rid of the text that indicates which user is currently logged in. This is unnecessary most of the time, because most users will only ever have a single account, and since they have to manually log themselves in before they can see this screen, it's pointless to show them something they already know.

Also, let's kill the option to change how many messages display in the list at once. This can certainly be retained as a feature, but it's not the kind of thing users are going to use every day, so we can move it to the Settings screen.

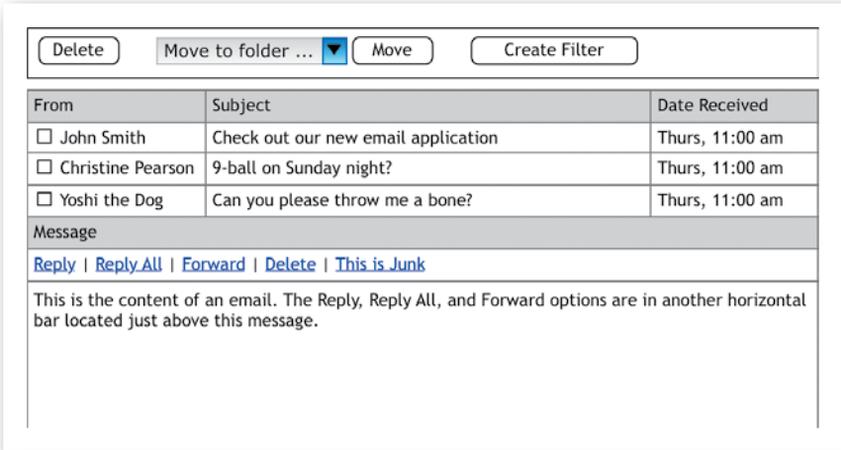
And since a Search bar is provided in the left-hand sidebar, we can remove the Search link from the top of the page.



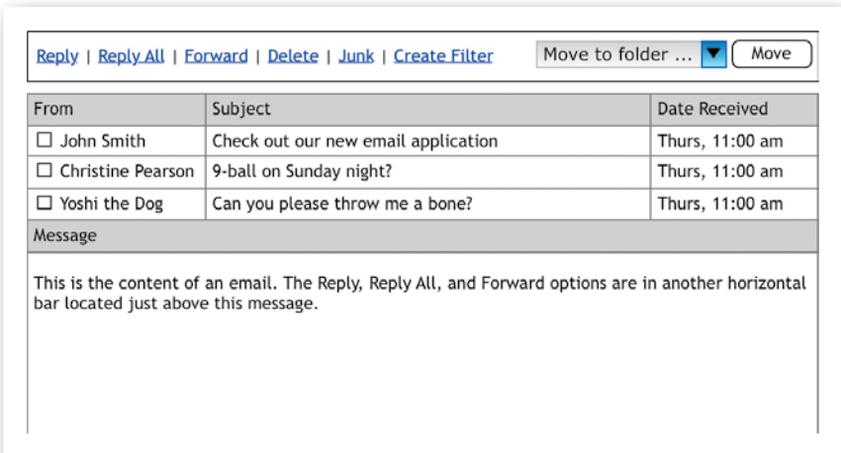
Showing a title bar for which folder is currently being displayed is redundant, because the label for the folder in the sidebar is made larger and bold when that folder is displayed. And if we remove the Folder title bar, we can free up some vertical space for more important content—like *mail*.



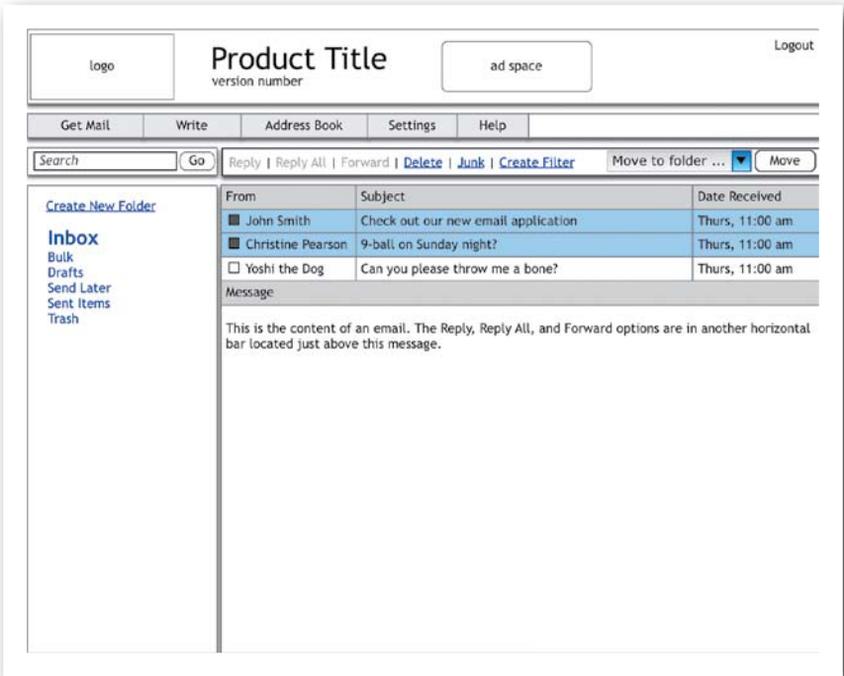
When an email is being displayed, another small bar appears above the email offering Reply, Reply All, Forward, and Delete functions, as well as a way to mark an email as junk.



But there's already a Delete button in the bar above the message list. If we remove the button and tidy things up a bit, we can consolidate the bar and unify the message options into a single interface element, which means less code, less interface, and less confusion.



Finally, let's add some logic to the application and have it disable the Reply, Reply All, and Forward links if more than one message is selected at a time. Delete, Junk, and Create Filter can all be applied to multiple messages, so we'll leave those active. In doing this, we make the message options more functional while still taking up less space.



Ahh, that's much better. We stripped out a few features, removed a few interface elements, cleaned things up, and came out with an application interface that is easier to understand at a quick glance and easier to use on a daily basis.

We'll perform interface surgery throughout this book as a way of improving applications one step at a time.

## Reevaluate nice-to-have features later

So, when is it time to take the list of nice-to-have features back out of the filing cabinet? The simple answer is this: not one second before your application has been released.

Once your application is out there, being used by real users, and you've given it some time to stabilize by fixing a lot of the immediate bugs that have inevitably come up since the release, then it's time to review the list of nice-to-haves. It's also time for a good laugh.

What usually happens is that users start to speak up about what they wish your application did, things that bother them, and so on, and no one ever mentions the items on your list of nice-to-haves. Users very quickly form different perspectives on your application than you may have ever had, and since none of them use the application exactly the way you thought they would, the complaints and wish lists that emerge are usually different than what you thought was important.

If this is the case for you, feel free to put that list of nice-to-haves into the *other* filing cabinet—the one shaped like a trash can—and call it a day. The things we often think are so important at the beginning of a project usually prove to be about as useful as adding another color to a logo. And more often than not, adding them way back when would have meant putting the rock star features at risk by making them harder to find, harder to configure, *harder to use*.

## Let them speak

Once your application is being used out in the wild and you want to hear all the little screaming voices of your users, you need to give them a way to talk to you. This means providing a way for users to offer feedback about your product or talk to others about it, getting out of the way so they can speak freely while you take notes and carefully interpret.

Something as simple as setting up a forum on your site and directing people there from your Support page can dramatically lower your customer-support costs (a forum costs extremely little to maintain), while greatly increasing the amount of information you get from customers.

Note, however, that you will probably not like everything that gets posted. Invariably, there will be some dissatisfied and possibly rude users who scream about your “horrible” application and say nothing constructive, but you have to let this happen. If you moderate user comments to filter out the negative, you'll defeat the purpose of the forum, which is to hear the complaints. The goal is to feel the pain.

When you allow your users to speak up, you'll quickly come up with a whole new list of nice-to-haves. Put those in the filing cabinet as well.

Avoid bending to users' whims if the high-demand features don't fit into your grand vision for the application. You might try pooling a few beta users together and have them try out a prototype of the proposed functionality to see how it really works before unleashing it to all your customers. There's no shame in pulling the feature back out if it just doesn't work. Better now than later.

Focus only on the features that are the most essential. *Build only what is absolutely necessary.*

# Index

- 5S approach, 259–261
- 5-second test, 133–134
- 30 Days, 53
- 37signals
  - and Backpack, 107, 108
  - and feature lists, 88
  - and self-design, 51–52
  - sources of inspiration for, 278
  - up-to-speed aids, 147
  - and Writeboard, 213
- 60-Second Deadline exercise, 90–92
- 80-20 rule, 92–93

## A

- ACD, 45–48, 74
- ActionScript, 130
- active users, 33
- activity, designing for the, 44–47
- Activity-Centered Design, 45–48, 74
- add-on offers, 233
- Adobe
  - Dreamweaver, 129, 145
  - Flash (See Flash)
  - Illustrator, 38
  - Photoshop, 38
- Agile development, 265
- airline app, 85–87
- alert messages, 110–111, 153, 204, 205
- alignment, 225–227
- Amazon
  - customer recommendations, 186
  - email campaigns, 171
  - one-click purchase button, 170–171
  - social influence tools, 184–185
- Anderson, Tom, 22
- Android, 84
- animated buttons, 272
- Apple
  - human-interface design team, 46, 49
  - Human Interface Guidelines, 203, 212
  - iBooks app, 176–178
  - iPad (See iPad)
  - Keynote, 120, 125, 129
  - mobile platform, 84
  - Photos app, 115–116
  - word processing app, 102
- application design. *See also* design;  
obvious design
  - and card sorting, 168
  - good vs. great, 4
  - and Just-in-Time Design, 265
  - and kaizen, 258
  - main goal of, 245
  - and mobile devices, 85
  - and Pareto principle, 92
  - process of, 13
  - role of users in, 37
  - simplicity in, 245
  - situation-centered nature of, 50
  - and vision, 19, 34
- application designers. *See also* designers
  - and demographics, 42
  - and forms, 155
  - and mobile devices, 85
  - recommended book for, 231
  - role of, 264
  - and setup wizards, 142
  - vs. strategists, 264
- application elements, 15
- application errors. *See* error messages;  
errors
- application form, 93–98
- application-modal dialog boxes, 204
- applications. *See also* software; web  
applications
  - adding to vs. improving, 19, 259
  - applying 80-20 rule to, 92–93
  - being consistent across, 229–233
  - card sorting for, 166–168
  - characteristics of well-designed, 277
  - choosing colors for, 235–236
  - competitive analysis of, 26–28
  - detailing features of, 262
  - detecting errors in, 195–203
  - eliminating implementation models in,  
119–127
  - establishing set style for, 261
  - getting user feedback on, 99–100, 128
  - getting users up to speed on, 141–152,  
242
  - how different people approach, 140–141
  - importance of simplicity in, 89, 147,  
245–246
  - knowing motivation for, 34
  - learning from great, 277–279
  - measuring effectiveness of, 32–33
  - minimizing users' frustration with,  
78–80
  - mistake-proofing, 191 (See also poka-  
yoke devices)
  - outlining goals for, 262

applications (continued)  
 preventing errors in, 193–195  
 reducing complexity of, 266  
 removing good lines from, 279–280  
 reviewing/refining, 267–268  
 taking inventory of, 25  
 testing, 130–138 (*See also* usability testing)  
 writing design criteria for, 30–31  
 application settings, 142, 163  
 application strategy, 24  
 attention, getting users', 176  
 attribution error, fundamental, 48  
 audits, user experience, 24–28  
 authority, 185–186  
 Axure RP Pro, 120

## B

Back button, 271  
 Backpack, 107–111  
 creators of, 107  
 as example of metaphor that works, 107, 111  
 getting inspiration from, 278  
 inline editing features, 246  
 inspiration for, 108  
 JavaScript alert message, 110–111  
 purpose of, 107  
 sample pages, 107, 109  
 up-to-speed aids, 147  
 backup application, 181  
 badges, 188  
 Ballpark, 87, 246  
 banana principle, 237–242  
 banking app, 178–179  
 Basecamp  
 blank-slate filler, 169–170  
 purpose of, 51  
 and self-design, 51–52  
 up-to-speed aids, 147–148  
 BBC News app, 224–225  
 beginners  
 designing for, 140–141  
 and instructive design, 159  
 and up-to-speed aids, 168  
 beta sites, 187  
*Big Red Fez, The*, 237  
 billing software, 83. *See also* Blinksale  
 blank-slate fillers, 147–149, 168–169  
*Blink*, 48  
 Blinksale  
 creators of, 80–81, 83  
 design process for, 123  
 getting inspiration from, 277, 278  
 how it works, 81–83

main invoice-editing screen, 81  
 mobile version of, 87  
 payment options, 83  
 purpose of, 80  
 sample invoice, 82  
 simplicity of, 82–84, 246  
 blogging service, 198  
 blogs, 105, 174, 181, 253  
 bookmarking tool, 193  
 book-preview feature, iBooks, 177  
 boxes, 248  
 Boxes and Arrows, 164  
 brainstorming, 31, 265, 266  
 brands, 22, 229, 248  
 Brin, Sergey, 23  
 browser incompatibility, 197  
 browserless self-tests, 131–133  
 browsers, hiding tools in, 131  
 bug reports, 104, 267  
 bulleted lists, 250  
 bulletproofing, 136  
 buzzwords, 250

## C

Camtasia Studio, 135  
 Cancel button, 233  
 card-sorting exercises, 166–168  
 Chase Mobile, 178–179  
 chrome, 222, 223, 224  
 Cialdini, Dr. Robert, 178, 181  
 Cisco Systems, 66  
 Citrix Systems, 66  
 cleaning up  
 application-development process, 263–264  
 task flows, 255–258  
 click-through mockups, 129  
 cloud storage service, 180–181  
 clutter  
 interface, 122, 246  
 task flow, 244–245, 255–258  
 CNN.com, 163  
 Cocoa Box Design, 66  
 code base, 233  
 cognitive walkthroughs, 25  
 color, 235–236, 241, 246, 252–253  
 commitment, 182–184  
 company brands, 22, 229, 248  
 company strategy, 20  
 company vision, 19, 21, 23  
 competitive analysis, 26–28  
 conceptual elements, 14  
 configuration options, 160, 163  
 confirmation messages, 110–111  
 consistency. *See also* uniformity

- across applications, 229–233
- and design patterns, 230–233
- and first impressions, 219
- leveraging standards to enable, 261
- persuasion via, 182–184
- vs. intelligent inconsistency, 233
- when creating wireframes, 122
- consistent design, 122, 234. *See also* consistency
- consistent layout, 228
- consistent messaging, 229–230
- consistent typography, 227
- contact information, 155
- content. *See also* information
  - adopting conversational tone for, 275
  - allowing sufficient space for, 222–225
  - defining structure for, 222
  - how users search for, 164
  - identifying structure of, 261
  - importance of, 223, 281
  - reducing clutter in, 248–250
  - separating presentation from, 261
- contextual inquiry, 63–65, 137
- contextual usability testing, 136–137
- contrast, 237, 246
- conversion rate, 33
- Cooper, Alan, 37, 79, 140
- corporate logos, 247–248. *See also* brands
- correspondence bias, 48
- courseware, 270
- CSS, 15, 72, 159, 222, 261
- customer reviews, 186
- customer satisfaction rates, 33

## D

- Dashboard HQ, 193–196
  - Add Content panel, 193
  - Drag and Drop Modules pattern, 233
  - poka-yoke detection devices, 195–203
  - poka-yoke prevention devices, 193–195
  - purpose of, 193
- data layer, 5
- deception, 188
- defaults
  - choosing good, 160–163
  - in form fields, 150
  - for Google Page Creator, 214–215
- design. *See also* application design
  - consistent, 122, 234 (*See also* consistency)
  - getting second opinion on, 126–127
  - improving, 33
  - instructive, 149, 153–159
  - Just-in-Time, 265–268
  - leveraging irregularities in, 220, 234–242
  - measuring effectiveness of, 32–33
  - obvious (*See* obvious design)
  - persuasive, 176, 179
  - prototyping, 127–130 (*See also* prototypes)
  - situation-centered, 50
  - success/failure of, 281–282
  - symptoms of weak, 19–20
  - user-centered, 40–41
- design criteria, 30–31
- designers. *See also* application designers
  - and competitive analysis, 28
  - and demographics, 42
  - and error pages, 196 (*See also* errors)
  - essential book for, 231
  - and mobile devices, 85
  - and modal errors, 205
  - role of, 176, 182, 264
  - and self-design, 50
  - and user-centered design, 40, 46
  - and wizards, 142
- design implementation, 32
- designing
  - for the activity, 44–47
  - for information, 163–168
  - for intermediate users, 140–141
  - for mental models, 104–119
  - for mobile platforms, 84–87
  - for touchscreen devices, 111
  - for uniformity, 220–228
  - for the user, 37–44
  - white space, 251–255
- Designing Interfaces*, 173, 231
- [designinginterfaces.com](http://designinginterfaces.com), 231
- Designing Visual Interfaces*, 245
- design pattern libraries, 231–233, 261
- Design Pattern Library, Yahoo, 232–233
- design patterns
  - collections of, 231–233
  - defined, 172
  - for displaying/hiding HTML page, 209–211
  - for editing within pages, 172
  - making things familiar with, 171–173
  - for pagination, 171–172, 230
  - purpose of, 172, 231
  - recommended book on, 173, 231
  - understanding, 230–233
- design thinking, 31
- desktop computers, 87, 102
- detection devices, error, 195–203
- DeWolfe, Chris, 23
- dialog boxes, 203–204
- digital cameras, 266
- dimension, 237, 242
- discount codes, 187

displaying/hiding HTML page, 209–211  
 document-modal dialog boxes, 203  
 document-modal errors, 205  
 dog food, eating your own, 137–138  
 domain name, changing, 137  
*Don't Make Me Think*, 88, 248  
 Doublewide Labs, 81, 83  
 drag-and-drop interactions, 14–15  
 Drag and Drop Modules pattern, 233  
 Dreamweaver, 129, 145  
 Dropbox, 180–181  
 dry-erase boards, 124. *See also* whiteboards  
 dummy text, 150

## E

ease of use, 46, 83, 105  
 eating your own dog food, 137–138  
 e-books, 49, 176–178  
 Einstein's rule, 122  
 e-learning application, 270–272  
 e-learning companies, 270  
*Elements of Style, The*, 88, 231, 248  
 elevation  
   and good design, 276–277  
   and polite applications, 273–274  
   and software personality, 274–275  
   of standards, 278–279  
   of user experience, 272–277, 280  
   vs. innovation, 272, 279  
 elevator pitch, 14  
 email addresses, 156  
 email applications, 187, 192, 206, 255–256  
 email campaigns, 171  
 emotional connections, 274  
 e-reader interface, 177. *See also* e-books  
 error-detection devices, 195–203  
 error messages  
   as bug reports, 104  
   in desktop-installed applications,  
     203–204  
   examples of good and bad, 208  
   for forms, 156, 158  
   how users react to, 190, 195–196  
   and implementation-model designs, 104  
   writing effective, 195–196, 207–208  
 error pages, 196–198  
 error-prevention devices, 193–195  
 errors, 190–215  
   converting to opportunities, 200–201  
   detecting, 195–203  
   how users react to, 190  
   preventing, 193–195, 203  
 ethical persuasion, 188  
 ethnographic research, 63  
 Excel, Microsoft, 59–60

exceptions, 70–72  
 exclusivity, 187  
 expert users  
   designing for, 140  
   and help documents, 173  
 exploratory searching, 164

## F

Facebook, 152  
 failure, product, 36–37  
 FAQs, 209–211  
 features  
   dropping nice-to-have, 88–98, 121, 279  
   reevaluating, 98–99  
   sorting through, 260  
 featuritis, 78  
 feedback  
   for forms, 153  
   soliciting, 99–100  
   via prototyping, 128  
 FEMA website, 196–197  
 Fettig, Abe, 130  
 file backups, 181  
 filing systems, 102, 180  
 Firewheel Design, 80, 83, 123  
 five-second test, 133–134  
 fivesecondtest.com, 134  
 Flash  
   and mobile devices, 31  
   prototypes, 130  
   UI Component Set, 130  
   welcome screen, 145  
 Flipboard, 152–153  
 flowcharts, 74–75  
 focused applications, 78, 92–93  
 fonts, 122, 227  
 forgiving software, 211–215  
 forms  
   alignment of fields in, 225–227  
   applying instructive design to, 153–159  
   inline validation of, 157–158, 198–200, 203  
   providing instructive text in, 150  
   typical problems in, 156  
   ways of using, 197–198  
 form vs. function, 22  
 forums, 99, 174  
 Forward button, 272  
 “Four Modes of Seeking Information...”  
   article, 164  
 Foursquare, 188  
 Framework for Obvious Design, 12–16  
 frameworks, interaction design, 233  
 Frequently Asked Questions, 209–211  
 Fried, Jason, 108, 147, 278  
 frustration levels, user, 78–80

functionality, 281  
 functional specifications, 262. *See also*  
 specifications  
 function vs. form, 22  
 fundamental attribution error, 48

## G

Gadgets, 15  
 gaming techniques, 183–184  
 gestural interfaces, 84, 112, 115–116  
*Getting Real*, 88  
 Getting Started guides, 141–142  
 Gladwell, Malcolm, 48  
 Gmail, 187, 206, 256  
 goals  
   application, 262  
   obvious design, 10–12  
 Godin, Seth, 237, 239, 242  
 good practices, promoting, 213–215  
 GoodReads, 182  
 Google  
   Drag and Drop Modules pattern, 233  
   founders, 23  
   and JotSpot, 130  
   and misspelled words, 201–202  
   mobile platform, 84  
   pagination in search results pages, 171, 230  
   and scarcity principle, 187  
   Search box, 235  
   site-builder tool, 161, 214–215  
   as source of help information, 173–174  
 Google Gmail, 187, 206, 256  
 Google Page Creator, 161–162, 214–215  
 Google Personalized, 233  
 Google Sites, 130, 161  
 GoToMeeting, 66  
 graphical dividers, 255  
 graphical elements, 223, 244, 246, 247, 260  
 graphic designers, 71, 127

## H

happy talk, 250  
 Harris, Jensen, 173  
 help documents, 173–174, 249  
 help systems, 161, 173  
 Hemingway, Ernest, 279  
 hiding/displaying HTML page, 209–211  
 homepages  
   defining vision for, 28–31  
   design considerations, 222  
   typical problems with, 26  
   use of happy talk on, 250  
 Hopkin, Michael, 218

## HTML

  headings, 221, 222, 261  
   prototypes, 129  
   purpose of, 222, 261  
 human-centered design, 45  
 Human Interface Guidelines, Apple, 203, 212  
 humor, 275  
 hyperlinks, 252–253

## I

iBooks app, 176–178  
 ID3 tags, 165  
 iGoogle, 15  
 Illustrator, 38  
 image libraries, 135, 169  
 immersion, situational, 52–54  
 implementation, design, 32  
 implementation models  
   converting into mental models, 111–119  
   and ease of use, 105–106  
   eliminating, 119–127  
   and error messages, 104, 110  
   and Netflix, 115, 119  
   and user frustration, 106  
   vs. mental models, 103–104  
 importance, leveraging irregularity to  
   create, 234–242  
 inconsistency, intelligent, 233  
*Influence: The Psychology of Persuasion*, 178, 181  
 information. *See also* content  
   architecture, 163–164, 166  
   designing for, 163–168  
   helping users find, 251  
   how people learn, 270  
 information-based websites, 164  
 “Information Design Using Card Sorting”  
   article, 168  
 informative text, 250  
 inline-expand design pattern, 209–211  
 inline tips, 150, 151  
 inline validation, 157–158, 159, 198–200, 203  
*Inmates Are Running the Asylum, The*, 37, 140  
 innovation, 19, 270–272, 279  
 inspiration, 278  
 instructional designers, 64  
 instructive copy, 249  
 instructive design, 149, 153–159  
 instructive hints, 149–153  
 intelligent inconsistency, 233  
 interaction design, 34  
 interaction designers, 280  
 interaction design frameworks, 233  
 interaction element, 15

- interactions
    - drag-and-drop, 14–15
    - round-trip, 257
  - interface layer, 5–6
  - interfaces
    - designing visual, 245
    - one-click, 170–171
    - providing tips in, 150
    - reducing clutter in, 122
    - reusing, 229
    - reviewing/refining, 267–268
    - testing, 130 (*See also* usability testing)
    - uniformity in, 221
    - using color in, 235–236, 241, 246
    - using wireframes to design, 121
  - interface surgery
    - applying instructive design, 153–159
    - designing page elements to stand out, 237–242
    - for displaying/hiding HTML page, 209–211
    - implementation- to mental-model conversion, 111–119
    - on job application form, 93–98
  - intermediate users
    - designing for, 140–141
    - and help documents, 173
  - interview testing, 134–135, 136
  - Intuit QuickBooks, 83
  - inventory, application, 25
  - invoicing applications, 80, 87, 246. *See also*
    - Ballpark; Blinksale
  - iOS, 84
  - iPad
    - BBC News app, 224–225
    - Flipboard app, 152–153
    - iBooks app, 176–178
    - Kayak app, 212
    - most popular uses of, 85
    - Netflix app, 112–119
    - notes app, 66
    - NPR app, 224, 246
    - Photos app, 115–116
    - prototyping apps for, 129
    - sketchbook app, 148
    - Twitter app, 276–277
    - word processing app, 102
  - iPhone
    - Chase Mobile app, 178–179
    - iBooks app, 176–178
    - Pandora app, 150, 151
    - Phone app, 202
    - popularity of, 84–85
    - prototyping apps for, 129
    - Shazam app, 249
    - Southwest Airlines app, 86–87
    - Things app, 151
    - to-do list app, 151
  - iPod, 49
  - irregularities, leveraging, 220, 234–242
  - iteration, 259
  - Ive, Jonathan, 47
  - iWork, 102, 120
- ## J
- Japanese improvement processes, 72, 259–261. *See also* kaizen
  - JavaScript
    - alert messages, 110–111, 153, 203, 204, 205
    - for inline-expand design pattern, 209–211
    - inline validation script, 198
  - JIT Design, 265–266, 268
  - JIT Review, 265, 267–268
  - job application form, 93–98
  - Jobs, Steve, 47
  - JotSpot, 130
  - jQuery, 13, 15
  - Just-in-Time Design, 265–266, 268
  - Just-in-Time Review, 265, 267–268
- ## K
- kaizen
    - benefits of, 259
    - defined, 72, 273
    - original purpose of, 258
    - practicing, 258–261
    - and use cases, 72–73
    - and wireframes, 123–127
  - Kayak, 212
  - Keynote, 120, 125, 129
  - Keynotopia, 129
  - known-item search method, 164, 165
  - Krug, Steve, 88, 248, 250
  - Krug’s Third Law of Usability, 88
- ## L
- labeling systems, 102, 106
  - layers, web, 5–6
  - libraries, design pattern, 231–233, 261
  - liking principle, 186
  - Lindgaard, Gitte, 218
  - LinkedIn, 184
  - links, 252–253
  - log-in widgets, 138
  - logos, 247–248

## M

## Macintosh

- and Internet Explorer, 197
- and online forms, 196–198
- wireframe-creation tool, 120

## Macromedia

- Flash, 130 (*See also* Flash)
- weblog aggregator, 209

## markup code, 221

## meaning, leveraging irregularity to create, 234–242

## mental maps, 228

## mental models

- converting implementation models into, 111–119
- defined, 103
- designing for, 104–119
- examples of, 102–103
- helping users form, 220
- vs. implementation models, 103–104

## messages

- alert, 110–111, 153, 204, 205
- confirmation, 110–111
- error (*See* error messages)

## messaging, consistent, 229–230

## metadata, 165

metaphors, 102–103, 107, 237. *See also* mental models

## metrics, success, 32–33

## Microsoft

- Excel, 59–60
- help documents, 173
- Office team, 173
- PowerPoint, 120, 231
- Visio, 120
- Word, 58–59

## minimizing copy, 248–250

## Mint.com, 142–145, 146, 277, 278

## misaligned objects, 227

## misspelled words, 201–202

## “mistake-proofing” device, 191

## mobile devices

- designing for, 84–87, 279
- and Flash, 31
- most popular uses of, 85
- prototyping apps for, 129

## mockups, 129

## modal dialog boxes, 203–204

## modal errors, 203–207

## modeless assistants, 205–207

## modeless dialog boxes, 203

## Morae, 135

## motivation, 34

## MSN, 171

## Mullet, Kevin, 245

## MySpace, 23, 178

## My Yahoo, 172, 233

## N

## Nature.com, 218

## navigation

- and card sorting, 166
- devoting too much space to, 222, 223, 224
- persistent, 132, 133, 198, 214, 244
- for search-engine sites, 172
- and spatial memory, 228
- testing, 131–133
- and user experience, 26

## nested boxes, 248

## Netflix, 112–119

## Next button, 270–272

## nice-to-have features, 88–100, 121, 279

## Nielsen, Jakob, 160, 161, 247

## Norman, Donald, 45, 46, 47, 49, 74

## notes app, 66

## note-taking tool, 193

## NPR app, 224, 246

## NYTimes.com, 163

## O

## Obama, Barack, 182–183

## obvious design

- and design patterns, 171–173
- and element of familiarity, 171–173
- and featuritis, 78
- framework for, 12–16
- goals for, 10–12
- qualities of, 6–10, 219–220
- and web layers, 5–6

## Odeo, 154–159

## OmniGraffle, 120, 125

## one-click interfaces, 170–171

## organizational graphics, 248

## ownership, 177–178

## P

## Page, Larry, 23

## Pages for iPad, 102

## pagination, design patterns for, 171–172, 230

## Pandora, 150, 151

## paper prototypes, 128–129

*Paper Prototyping*, 129

## Pareto, Vilfredo, 92

## Pareto principle, 92–93

## participation, measuring, 33

## passions, 18

## pattern libraries, 231–233, 261

patterns, 172, 232. *See also* design patterns

- peer pressure, 184
  - Penultimate, 66, 148, 149
  - people layer, 5
  - PeopleSoft, 40–41
  - perpetual intermediates, 140
  - persistent navigation, 132, 133, 198, 214, 244
  - persona descriptions, 37, 38, 43, 48, 76.
    - See also* personas
  - personal computing, 85
  - personal-finance management tool, 142
  - personality, software, 274–275
  - personas, 37–44
    - and Activity-Centered Design, 47–48
    - components of, 37
    - defined, 37
    - example of, 38–40
    - purpose of, 37–38
    - and user-centered design, 40–44
  - persuasion
    - book about psychology of, 178, 181
    - ethical, 188
    - principles of, 181–188
    - as quality of great application, 9
  - persuasive design, 176, 179. *See also* persuasion
  - persuasive statements, 180, 186
  - Phone app, 202
  - Photos app, 115–116
  - Photoshop, 38
  - pixel-to-data ratio, 247–248
  - podcast services, 154, 165–166
  - poka-yoke devices, 191–203
    - defined, 191
    - downside of using, 195
    - examples of, 191–192
    - pronunciation of, 191
    - simplicity of, 195
    - types of, 193
    - on the web, 192
  - polite applications, 273–274
  - pop-up windows, 204, 209
  - portal pages, 242
  - “Power of Defaults” article, 160
  - PowerPoint, 120, 231
  - preferences, 163
  - prevention devices, error, 193–195
  - previewing process, iBooks, 177
  - pricing, 188
  - private beta sites, 187
  - problems. *See also* errors
    - detecting, 195–203
    - preventing, 193–195
    - solving, 49, 178–179, 266
  - product failure, 36–37
  - product logos, 247–248
  - product research, 53–57
  - professional networking site, 184
  - profile badges, 188
  - progress meters, 184
  - project management tool, 51. *See also* Basecamp
  - promotion codes, 187
  - proportion, 222–224
  - prototypes, 127–130
    - benefits of using, 127–128, 138
    - defined, 127
    - ways of creating, 128–130
  - Prototyping: A Practitioner’s Guide*, 41
  - purchase paths, 233
  - purchases, measuring, 33
  - Putorti, Jason, 142, 145
- ## Q
- quality assurance testing, 138
  - QuickBooks, 83
  - quotes, 186
- ## R
- reciprocity, 181–182
  - reconnaissance testing, 136
  - recurring activity, 33
  - redirecting users, 137
  - reducing clutter, 122
  - reduction, 245, 261, 280
  - refactoring code, 72
  - refinement, successive, 245
  - registration forms, 154–159, 203, 226.
    - See also* forms
  - regularity, 122
  - Remember the Milk, 200
  - remote user research, 66–67
  - requirements, project, 121
  - research
    - situation, 54–55
    - user, 66–67
  - responsiveness, 276–277
  - retention rate, 33
  - Retweet button, 185
  - rhjr.net, 282
  - Robertson, James, 168
  - round-trip interactions, 257
  - RP Pro, 120
  - RSS feeds, 152
  - rude behavior, 204–205, 273–274

## S

sample values, 149–150  
 Sano, Darrell, 245  
 scanability, web-page, 222  
 scarcity principle, 187–188  
 screen-recording tools, 135  
 screen shots, 250  
 scrolling pages, 222  
 Search boxes, 234–235  
 search engines, 172, 230  
 search methods, 164  
 seiketsu, 261  
 seiri, 260, 261  
 seiso, 260, 261  
 seiton, 260, 261  
 self-congratulatory statements, 250  
 self-design, 50–52  
 self-tests, browserless, 131–133  
 semantic code, 221, 222  
 semantic markup, 261  
 settings, application, 142, 163  
 setup wizards, 142  
 shadowing, 65–66  
 sharing activity, 33  
 Shazam app, 249  
 shining, 260  
 shitsuke, 261  
 signposts, 237, 242  
 simplicity, 89, 147, 195, 245–246  
 Sinek, Simon, 21  
 site-builder tool, 161–162, 211, 213–215  
 situation, solving for the, 48–55  
 situational immersion, 52–54  
 situational inquiry, 63  
 situation-centered design, 50, 54–55, 76  
 situation research, 54–55  
 sketchbook app, 148  
 sketches, 124, 127, 138, 264. *See also*  
 wireframes  
 smartphones, 85. *See also* mobile devices  
 Snyder, Carolyn, 129  
 social influence tools, 185  
 social networking, 23, 33, 184  
 social proof, 184–185  
 social psychology studies, 178  
 software. *See also* applications  
 adding to vs. improving, 19, 259  
 creating forgiving, 211–215  
 detecting errors in, 195–203  
 giving personality to, 274–275  
 preventing errors in, 193–195  
 promoting good practices with, 213–215  
 qualities of great, 8–10  
 testing, 130–138 (*See also* usability testing)

software designers, 230. *See also* application  
 designers  
 Software Requirements Specification, 262  
 Software Usability Research Laboratory, 251  
 sorting  
 application features, 260  
 card, 166–168  
 Southwest Airlines app, 85–87  
 spatial memory, 126, 220, 227–228  
 specifications, 262, 263–264  
 spelling mistakes, 201–202  
 Spencer, Donna, 164  
 Spool, Jared, 233, 251  
 spread gesture, 115, 117  
 Spurlock, Morgan, 53  
 Squidoo, 201  
 SRS, 262  
 standardization, 261  
 standards, 278–279  
*Start with Why: How Great Leaders Inspire  
 Everyone to Take Action*, 21  
 stock photography sites, 135, 169  
 straightening, 260  
 strategic thinking, 34  
 strategists, 264  
 strategy  
 application, 24  
 company, 20  
 user experience (*See* user experience  
 strategy)  
 Strunk, William, 88, 248  
 style guidelines, 261  
 success metrics, 32–33  
*Super Size Me*, 53  
 SURL, 251  
 SurveyMonkey.com, 67  
 surveys, 67–68  
 sustaining work, 261  
 system messages, 229. *See also* messages

## T

tablet computers, 85, 87. *See also* mobile  
 devices  
 tag clouds, 105–106  
 Target.com, 133  
 task-flow diagrams, 25, 74–75  
 task flows  
 applying kaizen to, 258  
 cleaning up, 255–258  
 designing, 233, 258  
 examples of cluttered, 244–245  
 task-management system, 244–245, 255, 257  
 tax software, 83  
 technology, adapting to, 44

TechSmith, 135  
 testimonials, 186  
 testing. *See also* usability testing  
   with browserless self-test, 131–133  
   compensating participants in, 136  
   with contextual usability testing, 136–137  
   by eating your own dog food, 137–138  
   with five-second test, 133–134  
   with interview-style sessions, 134–135, 136  
   quality assurance, 138  
   recruiting testers for, 135–136  
 text fields, 149–150  
 Thaper, Sunny, 124  
 Things app, 151  
 Three Rs, 121–123  
 Tidwell, Jenifer, 173, 231  
 to-do list app, 151, 200  
 tooltips, 145  
 touchscreens, 84, 111, 115  
 Toyota, 72  
 trash metaphor, 102  
 travel-booking app, 212–213  
 trigger words, 165, 250  
 trust, 186, 220  
 Try It Now button, 250  
 Tumblr, 226  
 TurboTax, 83  
 tutorials, 145, 146  
 Twitter, 152, 185, 186, 276–277, 282  
 typefaces, 227. *See also* fonts  
 TypePad, 198–199  
 typography, 227

**U**

UCD, 40–41, 48  
 UI Component Set, Flash, 130  
 undo functions, 110–111, 205–206, 212  
 uniform alignment, 225–227  
 uniformity, 122, 220–228, 234. *See also* consistency  
 uniqueness, 234  
 Unnecessary Test, The, 89–90  
 up-to-speed aids, 141–173  
   blank-slate fillers, 147–149, 168–169  
   choosing good defaults, 160–163  
   designing for information, 163–168  
   getting rid of, 168–173  
   Getting Started guides, 141–142  
   instructive hints, 149–153  
   and intermediate users, 140–141  
   Mint.com example, 142–145  
   welcome screens, 145–146, 168

URLs, 194  
 usability. *See also* ease of use  
   and screen designs, 222  
   testing (*See* usability testing)  
   and white space, 251  
 Usability, Krug's Third Law of, 88  
 usability experts, 160, 280, 281  
 usability studies, 222. *See also* usability testing  
 usability testing, 130–138  
   benefits of, 130  
   with browserless self-test, 131–133  
   compensating participants in, 136  
   contextual, 136–137  
   by eating your own dog food, 137–138  
   and help documents, 173  
   with interview-style sessions, 134–135, 136  
   with paper prototypes, 128–129  
   recruiting testers for, 135–136  
   on use of white space, 251–253  
 use cases, 68–74  
   applying kaizen to, 72–73  
   argument against, 74  
   defined, 68  
   exceptions for, 70–72  
   refining ideas for, 74  
   writing, 68–72  
 useit.com, 247  
 user-centered design, 40–41, 48  
 user experience  
   auditing, 24–28  
   elevating, 272–277, 280  
   strategy (*See* user experience strategy)  
 user experience strategy  
   core activities for developing, 24–34  
   defined, 19, 20, 23  
   importance of, 18  
   revisiting, 32  
 User Interface Engineering, 133–134  
 User Interface Library, Yahoo, 233  
 user research, 45, 66–67  
 users  
   designing for, 37–44  
   getting feedback from, 99–100, 128  
   how websites are judged by, 218–219  
   identifying needs of, 61–68  
   recruiting for usability tests, 135–136  
   shadowing, 65–66  
   understanding, 55–60  
 user surveys, 67–68

## V

values, 18  
 VaultPress, 238, 246, 277  
 version tracking, 213  
 video tours, 141, 146  
 Visio, 120  
 vision  
   and company success/failure, 19, 21,  
   22–23  
   defining, 28–31  
   importance of, 34  
 Vision Document, 262  
 visual design, 258, 281  
 visual hierarchy, 219, 221–222  
 visual interfaces, 245  
 visual metaphors, 102–103, 107. *See also*  
   mental models

## W

walkthroughs, cognitive, 25  
 Warfel, Todd Zaki, 41  
 waste, eliminating, 262–264  
*Web Anatomy: Interaction Design Frameworks  
 That Work*, 233  
 web applications. *See also* applications  
   choosing good defaults for, 160–163  
   eliminating waste in, 262–264  
   getting inspiration from great, 278  
   process for creating, 262–263  
   simplicity in, 245–246  
   testing, 131–133  
 web-based invoicing applications, 80, 87, 246.  
   *See also* Ballpark; Blinksale  
 web-based site-creation tool, 211, 213–215  
 web browsers, hiding tools in, 131  
 WebEx, 66  
 web forms. *See* forms  
 web layers, 5  
 webmail applications, 187, 192, 206,  
   255–256  
 web pages. *See also* websites  
   choosing colors for, 235–236, 241, 246  
   hiding/displaying, 209–211  
   making elements stand out on, 234–235,  
   237–242  
   scanability of, 222  
   scrolling, 222  
   surfacing bananas in, 239  
   use of white space on, 222, 251–255  
   visual hierarchy for, 221–222  
 web resources, 174  
 websites. *See also* web applications  
   homepages for (*See* homepages)  
   how users judge, 218–219

  importance of content for, 222–225  
   tool for building, 211, 213–215  
 web standards, 261, 278–279  
 web usability, 88, 160, 247, 251. *See also*  
   usability  
 “Web Users Judge Sites in the Blink of an  
 Eye” article, 218  
 welcome screens, 145–146, 169  
 well-designed apps, 277  
 What’s This? documentation, 249  
 White, E.B., 88, 231, 248  
 whiteboards, 124, 125, 266  
 white space, 251–255  
   as alternative to graphics, 253–255  
   benefits of using, 251  
   as design element, 222  
   and file size, 255  
   separating areas of page with, 253–254  
 Wichita State University, 251  
 wikis, 141, 231  
 Williams, Josh, 83, 123  
 wireframes, 119–127  
   and application process, 262  
   applying kaizen to, 123–127  
   benefit of using, 121  
   purpose of, 119–120, 138  
   revising, 127  
   for simple log-in screen, 120  
   Three Rs for creating, 121–123  
   tools for creating, 120  
   vs. coding, 127  
 Wireframe Toolkit, Keynote, 129  
 wizards, setup, 142  
 Woot, 275  
 Word, Microsoft, 58–59  
 Writeboard, 213  
 WYSIWYG editors, 129

## X

X-factor, 15

## Y

Yahoo  
   Design Pattern Library, 232–233  
   editing design patterns, 172  
   pagination in search results pages, 171  
   User Interface Library, 233  
 Yelp, 234–235  
 YUI Library, 233

## Z

Zeldman, Jeffrey, 185  
 zip codes, 156