A Brain-Friendly Guide

2nd Edition
Updated for HTML5

Head First HTML and CSS

Launch your web career in one chapter

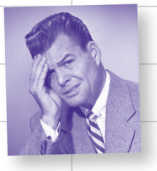
A learner's guide to creating standards-based web pages

Watch out for common HTML & CSS traps and pitfalls

Learn why everything your friends know about style is probably wrong

Bend your mind around 100 puzzles & exercises

Avoid embarrassing validation mistakes



O'REILLY®

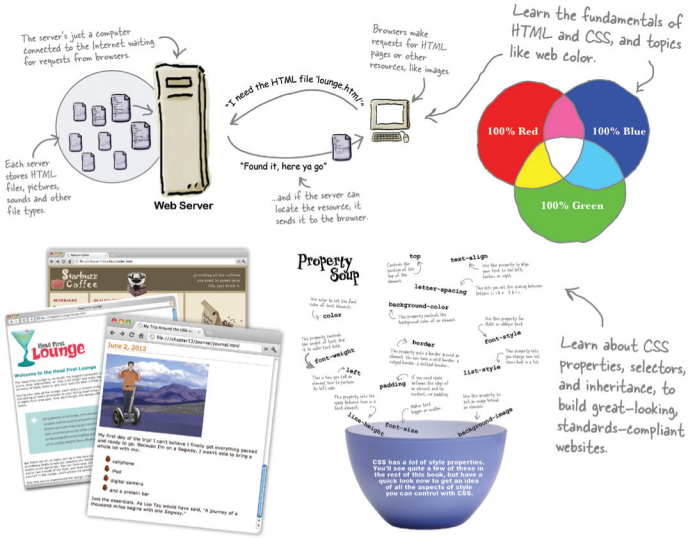
Elisabeth Robson & Eric Freeman

Head First HTML and CSS

Web Design & Development/HTML

What will you learn from this book?

Tired of reading HTML books that only make sense after you're an expert? Then it's about time you picked up the newly revised *Head First HTML and CSS* and really learned HTML. You want to learn HTML and CSS so you can finally create those web pages you've always wanted, so you can communicate more effectively with friends, family, fans, and fanatic customers. You also want to do it right, using the latest HTML5 standards, so you can actually maintain and expand your web pages over time so they work in all browsers and mobile devices.



What's so special about this book?

In this book, you'll learn the real secrets of creating web pages, and, most importantly, you'll learn them in a way that won't put you to sleep. If you've read a Head First book, you know what to expect: a visually rich format designed for the way your brain works. Using the latest research in neurobiology, cognitive science, and learning theory, this book will load HTML and CSS into your brain in a way that sticks.

US $39.99 CAN $41.99
ISBN: 978-0-596-15990-0

twitter.com/headfirstlabs
facebook.com/HeadFirst

O'REILLY®
oreilly.com
headfirstlabs.com

"*Head First HTML and CSS* is a thoroughly modern introduction to forward-looking practices in web page markup and presentation. It correctly anticipates readers' puzzlements and handles them just in time. The highly graphic and incremental approach precisely mimics the best way to learn this stuff: make a small change and see it in the browser to understand what each new item means."

— *Danny Goodman, author of* Dynamic HTML: The Definitive Guide

"The information covered in this book is the same material the pros know, but taught in an educational and humorous manner that doesn't ever make you think the material is impossible to learn or you are out of your element."

— *Christopher Schmitt, author of* The CSS Cookbook *and* Professional CSS

Praise for *Head First HTML and CSS*

"*Head First HTML and CSS* is a thoroughly modern introduction to forward-looking practices in web page markup and presentation. It correctly anticipates readers' puzzlements and handles them just in time. The highly graphic and incremental approach precisely mimics the best way to learn this stuff: make a small change and see it in the browser to understand what each new item means."

— **Danny Goodman, author of *Dynamic HTML: The Definitive Guide***

"Eric Freeman and Elisabeth Robson clearly know their stuff. As the Internet becomes more complex, inspired construction of web pages becomes increasingly critical. Elegant design is at the core of every chapter here, each concept conveyed with equal doses of pragmatism and wit."

— **Ken Goldstein, Executive Vice President and Managing Director, Disney Online**

"The Web would be a much better place if every HTML author started off by reading this book."

— **L. David Baron, Technical Lead, Layout and CSS, Mozilla Corporation *http://dbaron.org/***

"I've been writing HTML and CSS for 10 years now, and what used to be a long trial-and-error learning process has now been reduced neatly into an engaging paperback. HTML used to be something you could just hack away at until things looked okay on screen, but with the advent of web standards and the movement toward accessibility, sloppy coding practice is not acceptable anymore…from a business standpoint or a social responsibility standpoint. *Head First HTML and CSS* teaches you how to do things right from the beginning without making the whole process seem overwhelming. HTML, when properly explained, is no more complicated than plain English, and the authors do an excellent job of keeping every concept at eye level."

— **Mike Davidson, President and CEO, Newsvine, Inc.**

"The information covered in this book is the same material the pros know, but taught in an educational and humorous manner that doesn't ever make you think the material is impossible to learn or you are out of your element."

— **Christopher Schmitt, author of *The CSS Cookbook* and *Professional CSS*, *schmitt@christopher.org***

"Oh, great. You made an HTML book simple enough a CEO can understand it. What will you do next? Accounting simple enough my developer can understand it? Next thing you know, we'll be collaborating as a team or something."

— **Janice Fraser, CEO, Adaptive Path**

More Praise for *Head First HTML and CSS*

"I *heart* *Head First HTML and CSS*—it teaches you everything you need to learn in a 'fun coated' format!"

— **Sally Applin, UI designer and fine artist, *http://sally.com***

"This book has humor and charm, but most importantly, it has heart. I know that sounds ridiculous to say about a technical book, but I really sense that at its core, this book (or at least its authors) really care that the reader learns the material. This comes across in the style, the language, and the techniques. Learning—real understanding and comprehension—on the part of the reader is clearly topmost in the minds of the authors. And thank you, thank you, thank you, for the book's strong and sensible advocacy of standards compliance. It's great to see an entry-level book, that I think will be widely read and studied, campaign so eloquently and persuasively on behalf of the value of standards compliance in web page code. I even found in here a few great arguments I had not thought of—ones I can remember and use when I am asked (as I still am)—'what's the deal with compliance and why should we care?' I'll have more ammo now! I also liked that the book sprinkles in some basics about the mechanics of actually getting a web page live—FTP, web server basics, file structures, etc."

— **Robert Neer, Director of Product Development, Movies.com**

"*Head First HTML and CSS* is a most entertaining book for learning how to build a great web page. It not only covers everything you need to know about HTML and CSS, it also excels in explaining everything in layman's terms with a lot of great examples. I found the book truly enjoyable to read, and I learned something new!"

— **Newton Lee, Editor-in-Chief, ACM Computers in Entertainment *http://www.acmcie.org***

"My wife stole the book. She's never done any web design, so she needed a book like *Head First HTML and CSS* to take her from beginning to end. She now has a list of websites she wants to build—for our son's class, our family…If I'm lucky, I'll get the book back when she's done."

— **David Kaminsky, Master Inventor, IBM**

"Beware. If you're someone who reads at night before falling asleep, you'll have to restrict *Head First HTML and CSS* to daytime reading. This book wakes up your brain."

— **Pauline McNamara, Center for New Technologies and Education, Fribourg University, Switzerland**

Praise for other books by Eric Freeman and Elisabeth Robson

"From the awesome *Head First Java* folks, this book uses every conceivable trick to help you understand and remember. Not just loads of pictures: pictures of humans, which tend to interest other humans. Surprises everywhere. Stories, because humans love narrative. (Stories about things like pizza and chocolate. Need we say more?) Plus, it's darned funny."

— **Bill Camarda, READ ONLY**

"This book's admirable clarity, humor, and substantial doses of clever make it the sort of book that helps even nonprogrammers think well about problem solving."

— **Cory Doctorow, co-editor of Boing Boing and author of *Down and Out in the Magic Kingdom* and *Someone Comes to Town, Someone Leaves Town***

"I feel like a thousand pounds of books have just been lifted off of my head."

— **Ward Cunningham, inventor of the wiki and founder of the Hillside Group**

"This book is close to perfect, because of the way it combines expertise and readability. It speaks with authority and it reads beautifully. It's one of the very few software books I've ever read that strikes me as indispensable. (I'd put maybe 10 books in this category, at the outside.)"

— **David Gelernter, professor of computer science, Yale University, and author of *Mirror Worlds* and *Machine Beauty***

"A nosedive into the realm of patterns, a land where complex things become simple, but where simple things can also become complex. I can think of no better tour guides than these authors."

— **Miko Matsumura, industry analyst, The Middleware Company former Chief Java Evangelist, Sun Microsystems**

"I laughed, I cried, it moved me."

— **Daniel Steinberg, Editor-in-Chief, java.net**

"Just the right tone for the geeked-out, casual-cool guru coder in all of us. The right reference for practical development strategies—gets my brain going without having to slog through a bunch of tired, stale professor-speak."

— **Travis Kalanick, founder of Scour and Red Swoosh, member of the MIT TR100**

"I literally love this book. In fact, I kissed this book in front of my wife."

— **Satish Kumar**

Other O'Reilly books by Eric Freeman and Elisabeth Robson

Head First Design Patterns

Head First HTML with CSS & XHTML (first edition)

Head First HTML5 Programming

Other related books from O'Reilly

HTML5: Up and Running

HTML5 Canvas

HTML5: The Missing Manual

HTML5 Geolocation

HTML5 Graphics with SVG and CSS3

HTML5 Forms

HTML5 Media

Other books in O'Reilly's *Head First* series

Head First C#

Head First Java

Head First Object-Oriented Analysis & Design (OOA&D)

Head First Servlets and JSP

Head First SQL

Head First Software Development

Head First JavaScript

Head First Ajax

Head First Rails

Head First PHP & MySQL

Head First Web Design

Head First Networking

Head First iPhone and iPad Development

Head First jQuery

Head First
HTML and CSS

Elisabeth Robson
Eric Freeman

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

Head First HTML and CSS

by Elisabeth Robson and Eric Freeman

Printed in Canada.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly Media books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://my.safaribooksonline.com*). For more information, contact our corporate/institutional sales department: (800) 998-9938 or *corporate@oreilly.com*.

Series Creators: Kathy Sierra, Bert Bates

Editor: Brett McLaughlin (first edition), Mike Hendrickson (second edition)

Cover Designer: Karen Montgomery

HTML Wranglers: Elisabeth Robson, Eric Freeman

Production Editor: Kristen Borg

Indexer: Ron Strauss

Proofreader: Rachel Monaghan

Page Viewer: Oliver

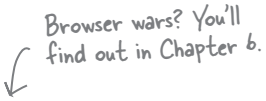
Printing History:

December 2005: First Edition.

September 2012: Second Edition.

Thanks to Clemens Orth for the use of his photo, "applestore.jpg", which appears in Chapter 5.

ISBN: 978-0-596-15990-0
[TI]



To the W3C, for saving us from the browser wars and for their brilliance in separating structure (HTML) from presentation (CSS)…

And for making HTML and CSS complex enough that people need a book to learn it.

Authors of Head First HTML and CSS

Eric Freeman

Elisabeth Robson

Eric is described by Head First series co-creator Kathy Sierra as "one of those rare individuals fluent in the language, practice, and culture of multiple domains from hipster hacker, corporate VP, engineer, think tank."

Professionally, Eric recently ended nearly a decade as a media company executive—having held the position of CTO of Disney Online and Disney.com at the Walt Disney Company. Eric is now devoting his time to WickedlySmart, a startup he co-created with Elisabeth.

By training, Eric is a computer scientist, having studied with industry luminary David Gelernter during his Ph.D. work at Yale University. His dissertation is credited as the seminal work in alternatives to the desktop metaphor, and also as the first implementation of activity streams, a concept he and Dr. Gelernter developed.

In his spare time, Eric is deeply involved with music; you'll find Eric's latest project, a collaboration with ambient music pioneer Steve Roach, available on the iPhone App Store under the name Immersion Station.

Eric lives with his wife and young daughter on Bainbridge Island. His daughter is a frequent vistor to Eric's studio, where she loves to turn the knobs of his synths and audio effects.

Write to Eric at `eric@wickedlysmart.com` or visit his site at `http://ericfreeman.com`.

Elisabeth is a software engineer, writer, and trainer. She has been passionate about technology since her days as a student at Yale University, where she earned a master's of science in computer science and designed a concurrent, visual programming language and software architecture.

Elisabeth's been involved with the Internet since the early days; she co-created the award-winning website, the Ada Project, one of the first websites designed to help women in computer science find career and mentorship information online.

She's currently co-founder of WickedlySmart, an online education experience centered on web technologies, where she creates books, articles, videos and more. Previously, as Director of Special Projects at O'Reilly Media, Elisabeth produced in-person workshops and online courses on a variety of technical topics and developed her passion for creating learning experiences to help people understand technology. Prior to her work with O'Reilly, Elisabeth spent time spreading fairy dust at the Walt Disney Company, where she led research and development efforts in digital media.

When not in front of her computer, you'll find Elisabeth hiking, cycling or kayaking in the great outdoors, with her camera nearby, or cooking vegetarian meals.

You can send her email at `beth@wickedlysmart.com` or visit her blog at `http://elisabethrobson.com`.

Table of Contents (summary)

Table of Contents (the real thing)

Intro

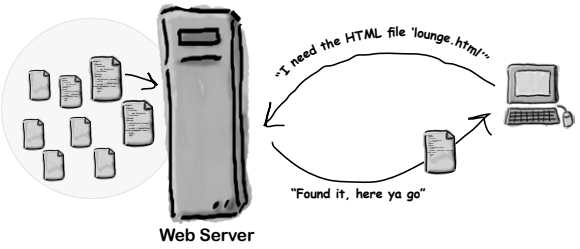
Your brain on HTML and CSS. Here *you* are trying to *learn* something, while here your *brain* is doing you a favor by making sure the learning doesn't *stick*. Your brain's thinking, "Better leave room for more important things, like which wild animals to avoid and whether naked snowboarding is a bad idea." So how *do* you trick your brain into thinking that your life depends on knowing HTML and CSS?

getting to know html

1 The Language of the Web

The only thing that is standing between you and getting yourself on the Web is learning to speak the lingo:

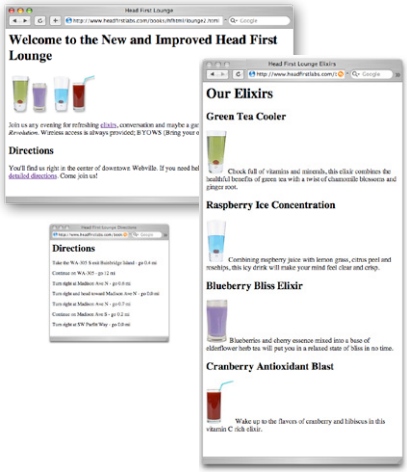
HyperText Markup Language, or HTML for short. So, get ready for some language lessons. After this chapter, not only are you going to understand some basic **elements** of HTML, but you'll also be able to speak HTML with a little **style.** Heck, by the end of this book, you'll be talking HTML like you grew up in Webville.



going further with hypertext

2 Meeting the "HT" in HTML

Did someone say "hypertext?" What's that? Oh, only the entire basis of the Web. In Chapter 1 we kicked the tires of HTML and found it to be a nice markup language (the "ML" in HTML) for describing the structure of web pages. Now we're going to check out the "HT" in HTML, hypertext, which will let us break free of a single page and link to other pages. Along the way we're going to meet a powerful new element, the <a> element, and learn how being "relative" is a groovy thing. So, fasten your seat belts—you're about to learn some hypertext.

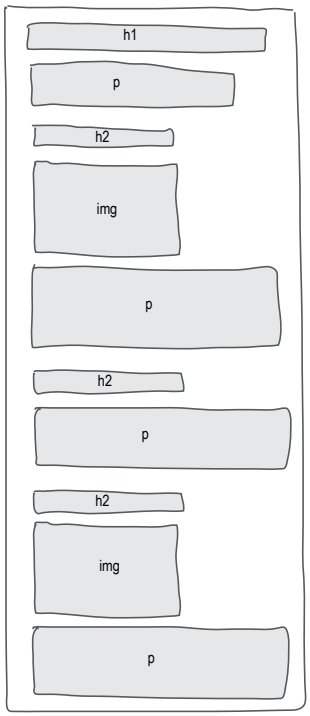


building blocks

3 Web Page Construction

I was told I'd actually be creating web pages in this book?

You've certainly learned a lot already: tags, elements, links, paths…but it's all for nothing if you don't create some killer web pages with that knowledge. In this chapter we're going to ramp up construction: you're going to take a web page from conception to blueprint, pour the foundation, build it, and even put on some finishing touches. All you need is your hard hat and your toolbelt, as we'll be adding some new tools and giving you some insider knowledge that would make Tim "The Toolman" Taylor proud.



getting connected

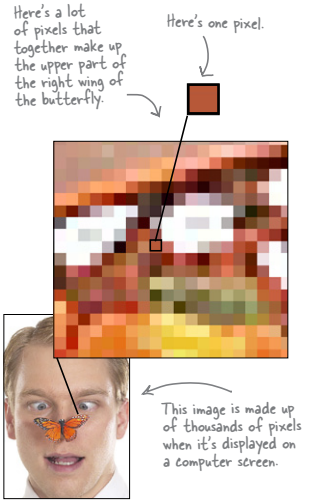
4 A Trip to Webville

Web pages are a dish best served on the Internet. So far you've only created HTML pages that live on your own computer. You've also only linked to pages that are on your own computer. We're about to change all that. In this chapter we'll encourage you to get those web pages on the Internet where all your friends, fans, and customers can actually see them. We'll also reveal the mysteries of linking to other pages by cracking the code of the h, t, t, p, :, /, /, w, w, w. So, gather your belongings; our next stop is Webville.

adding images to your pages

5 Meeting the Media

Smile and say "cheese." Actually, smile and say "gif," "jpg," or "png"—these are going to be your choices when "developing pictures" for the Web. In this chapter you're going to learn all about adding your first media type to your pages: images. Got some digital photos you need to get online? No problem. Got a logo you need to get on your page? Got it covered. But before we get into all that, don't you still need to be formally introduced to the <img> element? So sorry, we weren't being rude; we just never saw the "right opening." To make up for it, here's an entire chapter devoted to <img>. By the end of the chapter you're going to know all the ins and outs of how to use the <img> element and its attributes. You're also going to see exactly how this little element causes the browser to do extra work to retrieve and display your images.



standards and all that jazz

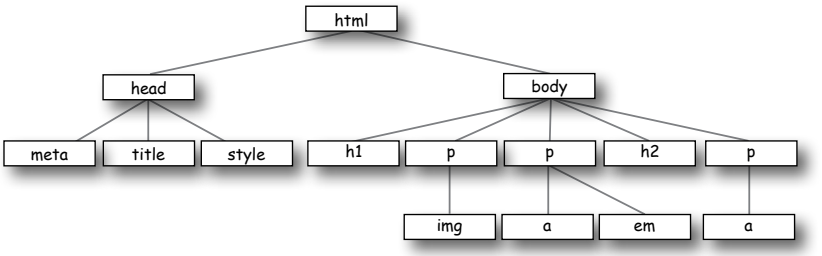
6 Serious HTML

What else is there to know about HTML? You're well on your way to mastering HTML. In fact, isn't it about time we move on to CSS and learn how to make all this bland markup look fabulous? Before we do, we need to make sure your HTML is really ready for the big leagues. Don't get us wrong, you've been writing first-class HTML all along, but there are just a few extra things you need to do to make it "industry standard" HTML. It's also time you think about making sure you're using the latest and greatest HTML standard, otherwise known as HTML5. By doing so, you'll ensure that your pages play well with the latest i-Device, and that they'll display more uniformly across all browsers (at least the ones you'd care about). You'll also have pages that load faster, pages that are guaranteed to play well with CSS, and pages that are ready to move into the future as the standards grow. Get ready, this is the chapter where you move from web tinkerer to web professional.

getting started with CSS

7 Adding a Little Style

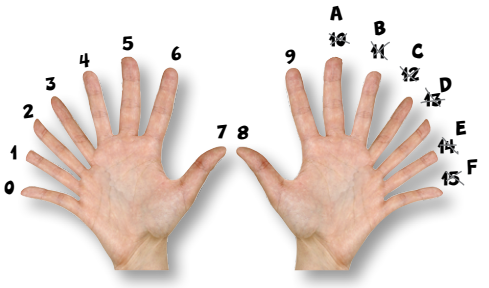
I was told there'd be CSS in this book. So far you've been concentrating on learning HTML to create the structure of your web pages. But as you can see, the browser's idea of style leaves a lot to be desired. Sure, we could call the fashion police, but we don't need to. With CSS, you're going to completely control the presentation of your pages, often without even changing your HTML. Could it really be so easy? Well, you *are* going to have to learn a new language; after all, Webville is a bilingual town. After reading this chapter's guide to learning the language of CSS, you're going to be able to stand on *either* side of Main Street and hold a conversation.



styling with fonts and colors

8 Expanding Your Vocabulary

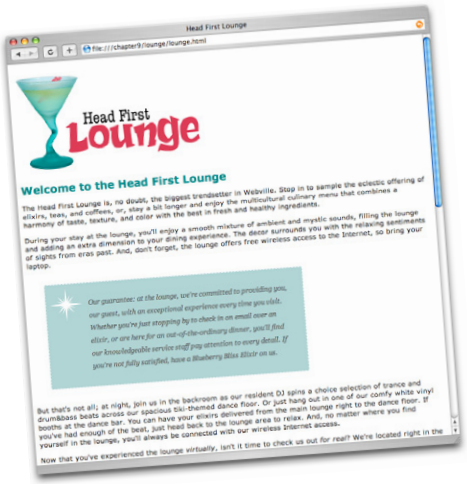
Your CSS language lessons are coming along nicely. You already have the basics of CSS down, and you know how to create CSS rules to select and specify the style of an element. Now it's time to build your vocabulary, and that means picking up some new properties and learning what they can do for you. In this chapter we're going to work through some of the most common properties that affect the display of text. To do that, you'll need to learn a few things about fonts and color. You're going to see you don't have to be stuck with the fonts everyone else uses, or the clunky sizes and styles the browser uses as the defaults for paragraphs and headings. You're also going to see there is a lot more to color than meets the eye.



the box model

9 Getting Intimate with Elements

To do advanced web construction, you really need to know your building materials. In this chapter we're going to take a close look at our building materials: the HTML elements. We're going to put block and inline elements right under the microscope and see what they're made of. You'll see how you can control just about every aspect of how an element is constructed with CSS. But we don't stop there—you'll also see how you can give elements unique identities. And, if that weren't enough, you're going to learn when and why you might want to use multiple stylesheets. So, turn the page and start getting intimate with elements.



divs and spans

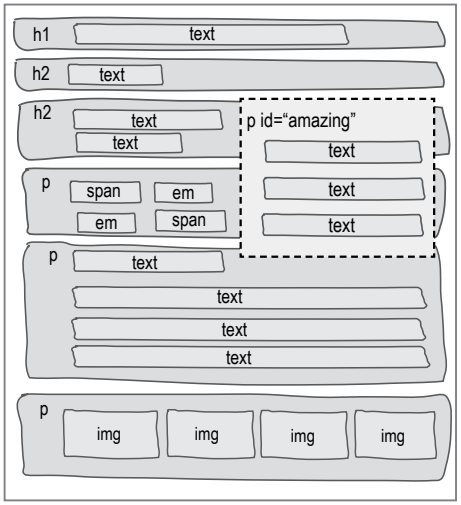
10 Advanced Web Construction

It's time to get ready for heavy construction. In this chapter we're going to roll out two new HTML elements: and . These are no simple "two by fours"; these are full-blown steel beams. With and , you're going to build some serious supporting structures, and once you've got those structures in place, you're going to be able to style them all in new and powerful ways. Now, we couldn't help but notice that your CSS toolbelt is really starting to fill up, so it's time to show you a few shortcuts that will make specifying all these properties a lot easier. And we've also got some special guests in this chapter, the *pseudo-classes*, which are going to allow you to create some very interesting selectors. (If you're thinking that "pseudo-classes" would make a great name for your next band, too late; we beat you to it.)

layout and positioning

11 Arranging Elements

It's time to teach your HTML elements new tricks. We're not going to let those HTML elements just sit there anymore—it's about time they get up and help us create some pages with real *layouts*. How? Well, you've got a good feel for the and structural elements and you know all about how the box model works, right? So, now it's time to use all that knowledge to craft some real designs. No, we're not just talking about more background and font colors—we're talking about full-blown professional designs using multicolumn layouts. This is the chapter where everything you've learned comes together.



html5 markup

12 Modern HTML

So, we're sure you've heard the hype around HTML5. And, given how far along you are in this book, you're probably wondering if you made the right purchase. Now, one thing to be clear about, up front, is that everything you've learned in this book has been HTML, and more specifically has met the HTML5 standard. But there are some new aspects of HTML markup that were added with the HTML5 standard that we haven't covered yet, and that's what we're going to do in this chapter. Most of these additions are evolutionary, and you're going to find you are quite comfortable with them given all the hard work you've already done in this book. There's some revolutionary stuff too (like video), and we'll talk about that in this chapter as well. So, let's dive in and take a look at these new additions!

tables and more lists

13 Getting Tabular

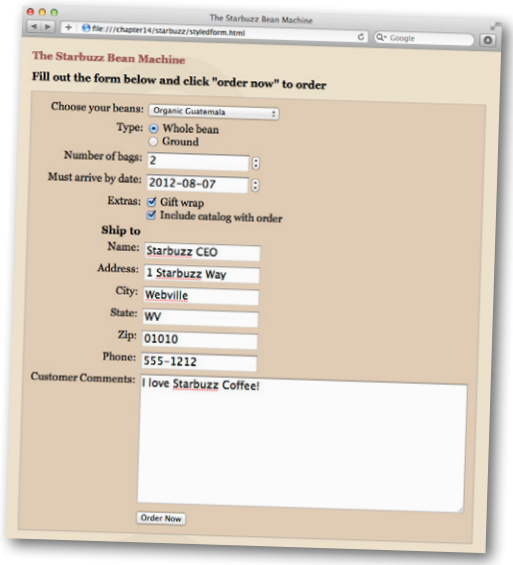
If it walks like a table and talks like a table… There comes a time in life when we have to deal with the dreaded *tabular data*. Whether you need to create a page representing your company's inventory over the last year or a catalog of your vinylmation collection (don't worry, we won't tell), you know you need to do it in HTML, but how? Well, have we got a deal for you: order now, and in a single chapter we'll reveal the secrets that will allow you to put your very own data right inside HTML tables. But there's more: with every order we'll throw in our exclusive guide to styling HTML tables. And, if you act now, as a special bonus, we'll throw in our guide to styling HTML lists. Don't hesitate; call now!

City	Date	Temperature	Altitude	Population	Diner Rating
Walla Walla, WA	June 15th	75	1,204 ft	29,686	4/5
Magic City, ID	June 25th	74	5,312 ft	50	3/5
Bountiful, UT	July 10th	91	4,226 ft	41,173	4/5
Last Chance, CO	July 23rd	102	4,780 ft	265	3/5
Truth or Consequences, NM	August 9th	93	4,242 ft	7,289	5/5
	August 27th	98			Tess 5/5 Tony 4/5
Why, AZ	August 18th	104	860 ft	480	3/5

html forms

14 Getting Interactive

So far all your web communication has been one-way: from your page to your visitors. Golly, wouldn't it be nice if your visitors could talk back? That's where HTML forms come in: once you enable your pages with forms (along with a little help from a web server), your pages are going to be able to gather customer feedback, take an online order, get the next move in an online game, or collect the votes in a "hot or not" contest. In this chapter you're going to meet a whole team of HTML elements that work together to create web forms. You'll also learn a bit about what goes on behind the scenes in the server to support forms, and we'll even talk about keeping those forms stylish.

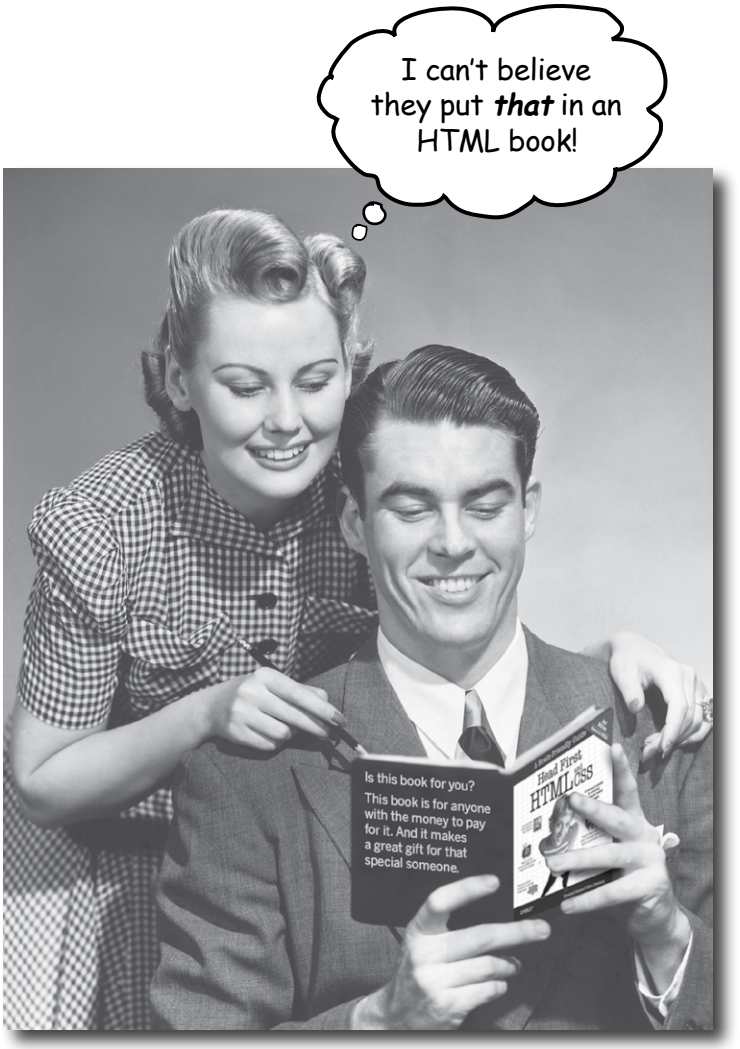


15 appendix: leftovers

The Top Ten Topics (We Didn't Cover)

We covered a lot of ground, and you're almost finished with this book. We'll miss you, but before we let you go, we wouldn't feel right about sending you out into the world without a little more preparation. We can't possibly fit everything you'll need to know into this relatively short chapter. Actually, we *did* originally include everything you need to know about HTML and CSS (not already covered by the other chapters), by reducing the type point size to .00004. It all fit, but nobody could read it. So, we threw most of it away, and kept the best bits for this Top Ten appendix.

Intro



In this section, we answer the burning question:
"So, why DID they put that in an HTML book?"

Who is this book for?

If you can answer "yes" to all of these:

1. Do you have access to a computer with a **web browser** and a **text editor**?

2. Do you want to **learn, understand,** and **remember** how to **create** web pages using the best techniques and the most recent standards?

3. Do you prefer **stimulating dinner-party conversation** to **dry, dull, academic lectures?**

this book is for you.

If you have access to any computer manufactured in the last decade, the answer is yes.

Who should probably back away from this book?

If you can answer "yes" to any one of these:

1. **Are you completely new to computers?**

   (You don't need to be advanced, but you should understand folders and files, simple text editing applications, and how to use a web browser.)

2. Are you a kick-butt web developer looking for a ***reference*** **book?**

3. Are you **afraid to try something different**? Would you rather have a root canal than mix stripes with plaid? Do you believe that a technical book can't be serious if HTML tags are anthropomorphized?

this book is not for you.

[Note from marketing: this book is for anyone with a credit card.]

We know what you're thinking.

"How can this be a serious book?"

"What's with all the graphics?"

"Can I actually learn it this way?"

And we know what your brain is thinking.

Your brain craves novelty. It's always searching, scanning, *waiting* for something unusual. It was built that way, and it helps you stay alive.

Today, you're less likely to be a tiger snack. But your brain's still looking. You just never know.

So what does your brain do with all the routine, ordinary, normal things you encounter? Everything it *can* to stop them from interfering with the brain's *real* job—recording things that *matter*. It doesn't bother saving the boring things; they never make it past the "this is obviously not important" filter.

How does your brain *know* what's important? Suppose you're out for a day hike and a tiger jumps in front of you—what happens inside your head and body?

Neurons fire. Emotions crank up. *Chemicals surge.*

And that's how your brain knows…

This must be important! Don't forget it!

But imagine you're at home, or in a library. It's a safe, warm, tiger-free zone. You're studying. Getting ready for an exam. Or trying to learn some tough technical topic your boss thinks will take a week, 10 days at the most.

Just one problem. Your brain's trying to do you a big favor. It's trying to make sure that this *obviously* non-important content doesn't clutter up scarce resources. Resources that are better spent storing the really *big* things. Like tigers. Like the danger of fire. Like how you should never again snowboard in shorts.

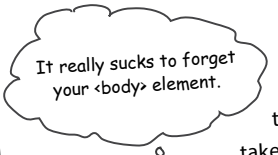
And there's no simple way to tell your brain, "Hey brain, thank you very much, but no matter how dull this book is, and how little I'm registering on the emotional Richter scale right now, I really *do* want you to keep this stuff around."

We think of a "Head First" reader as a learner.

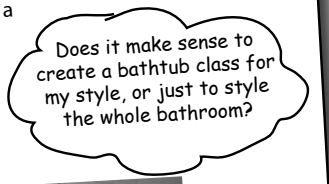
So what does it take to *learn* something? First, you have to *get* it, then make sure you don't *forget* it. It's not about pushing facts into your head. Based on the latest research in cognitive science, neurobiology, and educational psychology, *learning* takes a lot more than text on a page. We know what turns your brain on.

Some of the Head First learning principles:

Make it visual. Images are far more memorable than words alone, and make learning much more effective (up to 89% improvement in recall and transfer studies). It also makes things more understandable. **Put the words within or near the graphics** they relate to, rather than on the bottom or on another page, and learners will be up to *twice* as likely to solve problems related to the content.



Use a conversational and personalized style. In recent studies, students performed up to 40% better on post-learning tests if the content spoke directly to the reader, using a first-person, conversational style rather than taking a formal tone. Tell stories instead of lecturing. Use casual language. Don't take yourself too seriously. Which would *you* pay more attention to: a stimulating dinner-party companion, or a lecture?



Get the learner to think more deeply. In other words, unless you actively flex your neurons, nothing much happens in your head. A reader has to be motivated, engaged, curious, and inspired to solve problems, draw conclusions, and generate new knowledge. And for that, you need challenges, exercises, and thought-provoking questions, and activities that involve both sides of the brain, and multiple senses.

Get—and keep—the reader's attention. We've all had the "I really want to learn this, but I can't stay awake past page one" experience. Your brain pays attention to things that are out of the ordinary, interesting, strange, eye-catching, unexpected. Learning a new, tough, technical topic doesn't have to be boring. Your brain will learn much more quickly if it's not.

Touch their emotions. We now know that your ability to remember something is largely dependent on its emotional content. You remember what you *care* about. You remember when you *feel* something. No, we're not talking heart-wrenching stories about a boy and his dog. We're talking emotions like surprise, curiosity, fun, "what the…?", and the feeling of "I rule!" that comes when you solve a puzzle, learn something everybody else thinks is hard, or realize you know something that "I'm more technical than thou" Bob from engineering *doesn't*.

Metacognition: thinking about thinking

If you really want to learn, and you want to learn more quickly and more deeply, pay attention to how you pay attention. Think about how you think. Learn how you learn.

Most of us did not take courses on metacognition or learning theory when we were growing up. We were *expected* to learn, but rarely *taught* how to learn.

But we assume that if you're holding this book, you really want to learn how to create web pages. And you probably don't want to spend a lot of time. And you want to *remember* what you read, and be able to apply it. And for that, you've got to *understand* it. To get the most from this book, or *any* book or learning experience, take responsibility for your brain. Your brain on *this* content.

The trick is to get your brain to see the new material you're learning as Really Important. Crucial to your well-being. As important as a tiger. Otherwise, you're in for a constant battle, with your brain doing its best to keep the new content from sticking.

So how *DO* you get your brain to think HTML & CSS are as important as a tiger?

There's the slow, tedious way, or the faster, more effective way. The slow way is about sheer repetition. You obviously know that you *are* able to learn and remember even the dullest of topics, if you keep pounding on the same thing. With enough repetition, your brain says, "This doesn't *feel* important to him, but he keeps looking at the same thing *over* and *over* and *over*, so I suppose it must be."

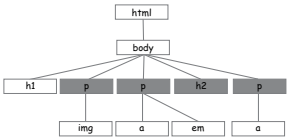
The faster way is to do ***anything that increases brain activity,*** especially different *types* of brain activity. The things on the previous page are a big part of the solution, and they're all things that have been proven to help your brain work in your favor. For example, studies show that putting words *within* the pictures they describe (as opposed to somewhere else in the page, like a caption or in the body text) causes your brain to try to make sense of how the words and picture relate, and this causes more neurons to fire. More neurons firing = more chances for your brain to *get* that this is something worth paying attention to, and possibly recording.

A conversational style helps because people tend to pay more attention when they perceive that they're in a conversation, since they're expected to follow along and hold up their end. The amazing thing is, your brain doesn't necessarily *care* that the "conversation" is between you and a book! On the other hand, if the writing style is formal and dry, your brain perceives it the same way you experience being lectured to while sitting in a roomful of passive attendees. No need to stay awake.

But pictures and conversational style are just the beginning.

Here's what WE did:

We used ***pictures***, because your brain is tuned for visuals, not text. As far as your brain's concerned, a picture really *is* worth 1,024 words. And when text and pictures work together, we embedded the text *in* the pictures because your brain works more effectively when the text is *within* the thing the text refers to, as opposed to in a caption or buried in the text somewhere.



We used ***redundancy***, saying the same thing in *different* ways and with different media types, and *multiple senses*, to increase the chance that the content gets coded into more than one area of your brain.

We used concepts and pictures in ***unexpected*** ways because your brain is tuned for novelty, and we used pictures and ideas with at least *some* ***emotional*** *content*, because your brain is tuned to pay attention to the biochemistry of emotions. That which causes you to *feel* something is more likely to be remembered, even if that feeling is nothing more than a little ***humor***, ***surprise***, or ***interest.***

We used a personalized, ***conversational style***, because your brain is tuned to pay more attention when it believes you're in a conversation than if it thinks you're passively listening to a presentation. Your brain does this even when you're *reading*.

We included more than 100 ***activities***, because your brain is tuned to learn and remember more when you ***do*** things than when you *read* about things. And we made the exercises challenging-yet-doable, because that's what most *people* prefer.

We used ***multiple learning styles***, because *you* might prefer step-by-step procedures, while someone else wants to understand the big picture first, while someone else just wants to see a code example. But regardless of your own learning preference, *everyone* benefits from seeing the same content represented in multiple ways.

We include content for ***both sides of your brain***, because the more of your brain you engage, the more likely you are to learn and remember, and the longer you can stay focused. Since working one side of the brain often means giving the other side a chance to rest, you can be more productive at learning for a longer period of time.

And we included ***stories*** and exercises that present ***more than one point of view,*** because your brain is tuned to learn more deeply when it's forced to make evaluations and judgments.

We included ***challenges***, with exercises, and by asking ***questions*** that don't always have a straight answer, because your brain is tuned to learn and remember when it has to *work* at something. Think about it—you can't get your *body* in shape just by *watching* people at the gym. But we did our best to make sure that when you're working hard, it's on the *right* things. That ***you're not spending one extra dendrite*** processing a hard-to-understand example, or parsing difficult, jargon-laden, or overly terse text.

We used ***people***. In stories, examples, pictures, etc., because, well, because *you're* a person. And your brain pays more attention to *people* than it does to *things*.

We used an ***80/20*** approach. We assume that if you're going to be a kick-butt web developer, this won't be your only book. So we don't talk about *everything*. Just the stuff you'll actually *need*.

Cut this out and stick it on your refrigerator.

Here's what YOU can do to bend your brain into submission

So, we did our part. The rest is up to you. These tips are a starting point; listen to your brain and figure out what works for you and what doesn't. Try new things.

1 Slow down. The more you understand, the less you have to memorize.

Don't just *read*. Stop and think. When the book asks you a question, don't just skip to the answer. Imagine that someone really *is* asking the question. The more deeply you force your brain to think, the better chance you have of learning and remembering.

2 Do the exercises. Write your own notes.

We put them in, but if we did them for you, that would be like having someone else do your workouts for you. And don't just *look* at the exercises. **Use a pencil.** There's plenty of evidence that physical activity *while* learning can increase the learning.

3 Read the "There Are No Dumb Questions."

That means all of them. They're not optional sidebars—***they're part of the core content!*** Don't skip them.

4 Make this the last thing you read before bed. Or at least the last *challenging* thing.

Part of the learning (especially the transfer to long-term memory) happens *after* you put the book down. Your brain needs time on its own, to do more processing. If you put in something new during that processing time, some of what you just learned will be lost.

5 Drink water. Lots of it.

Your brain works best in a nice bath of fluid. Dehydration (which can happen before you ever feel thirsty) decreases cognitive function.

6 Talk about it. Out loud.

Speaking activates a different part of the brain. If you're trying to understand something, or increase your chance of remembering it later, say it out loud. Better still, try to explain it out loud to someone else. You'll learn more quickly, and you might uncover ideas you hadn't known were there when you were reading about it.

7 Listen to your brain.

Pay attention to whether your brain is getting overloaded. If you find yourself starting to skim the surface or forget what you just read, it's time for a break. Once you go past a certain point, you won't learn faster by trying to shove more in, and you might even hurt the process.

8 *Feel* something!

Your brain needs to know that this *matters*. Get involved with the stories. Make up your own captions for the photos. Groaning over a bad joke is *still* better than feeling nothing at all.

9 *Create* something!

Apply this to something new you're designing, or rework an older project. Just do *something* to get some experience beyond the exercises and activities in this book. All you need is a pencil and a problem to solve…a problem that might benefit from using HTML and CSS.

Read me

This is a learning experience, not a reference book. We deliberately stripped out everything that might get in the way of learning whatever it is we're working on at that point in the book. And the first time through, you need to begin at the beginning, because the book makes assumptions about what you've already seen and learned.

We begin by teaching basic HTML, then standards-based HTML5.

To write standards-based HTML, there are a lot of technical details you need to understand that aren't helpful when you're trying to learn the basics of HTML. Our approach is to have you learn the basic concepts of HTML first (without worrying about these details), and then, when you have a solid understanding of HTML, teach you to write standards-compliant HTML (the most recent version of which is HTML5). This has the added benefit that the technical details are more meaningful after you've already learned the basics.

It's also important that you be writing compliant HTML when you start using CSS, so we make a point of getting you to standards-based HTML before you begin any serious work with CSS.

We don't cover every single HTML element or attribute or CSS property ever created.

There are a *lot* of HTML elements, *a lot* of attributes, and *a lot* of CSS properties. Sure, they're all interesting, but our goal was to write a book that weighs less than the person reading it, so we don't cover them all here. Our focus is on the core HTML elements and CSS properties that *matter* to you, the beginner, and making sure that you really, truly, deeply understand how and when to use them. In any case, once you're done with *Head First HTML and CSS*, you'll be able to pick up any reference book and get up to speed quickly on all the elements and properties we left out.

This book advocates a clean separation between the structure of your pages and the presentation of your pages.

Today, serious web pages use HTML to structure their content, and CSS for style and presentation. Nineties-era pages often used a different model, one where HTML was used for both structure and style. This book teaches you to use HTML for structure and CSS for style; we see no reason to teach you outdated bad habits.

We encourage you to use more than one browser with this book.

While we teach you to write HTML and CSS that are based on standards, you'll still (and

probably always) encounter minor differences in the way web browsers display pages. So, we encourage you to pick at least two modern browsers and test your pages using them. This will give you experience in seeing the differences among browsers and in creating pages that work well in a variety of them.

We often use tag names for element names.

Rather than saying "the a element," or "the 'a' element," we use a tag name, like "the <a> element." While this may not be technically correct (because <a> is an opening tag, not a full-blown element), it does make the text more readable, and we usually follow the name with the word "element" to avoid confusion.

The activities are NOT optional.

The exercises and activities are not add-ons; they're part of the core content of the book. Some of them are to help with memory, some are for understanding, and some will help you apply what you've learned. ***Don't skip the exercises.*** The crossword puzzles are the only things you don't *have* to do, but they're good for giving your brain a chance to think about the words in a different context.

The redundancy is intentional and important.

One distinct difference in a Head First book is that we want you to *really* get it. And we want you to finish the book remembering what you've learned. Most reference books don't have retention and recall as a goal, but this book is about *learning*, so you'll see some of the same concepts come up more than once.

The examples are as lean as possible.

Our readers tell us that it's frustrating to wade through 200 lines of an example looking for the two lines they need to understand. Most examples in this book are shown within the smallest possible context, so that the part you're trying to learn is clear and simple. Don't expect all of the examples to be robust, or even complete—they are written specifically for learning, and aren't always fully functional.

We've placed all the example files on the Web so you can download them. You'll find them at **`http://wickedlysmart.com/hfhtmlcss/`**.

The Brain Power exercises don't have answers.

For some of them, there is no right answer, and for others, part of the learning experience of the Brain Power activities is for you to decide if and when your answers are right. In some of the Brain Power exercises, you will find hints to point you in the right direction.

Tech reviewers (first edition)

Our reviewers:

We're extremely grateful for our technical review team. **Johannes de Jong** organized and led the whole effort, acted as "series dad," and made it all work smoothly. **Pauline McNamara**, "co-manager" of the effort, held things together and was the first to point out when our examples were a little more "baby boomer" than hip. The whole team proved how much we needed their technical expertise and attention to detail. **Valentin Crettaz**, **Barney Marispini**, **Marcus Green**, **Ike Van Atta**, **David O'Meara**, **Joe Konior**, and **Corey McGlone** left no stone unturned in their review and the book is much better for it. You guys rock! And further thanks to **Corey** and **Pauline** for never letting us slide on our often too formal (or we should just say it, incorrect) punctuation. A shout-out to JavaRanch as well for hosting the whole thing.

A big thanks to **Louise Barr**, our token web designer, who kept us honest on our designs and on our use of HTML and CSS (although you'll have to blame us for the actual designs).

Acknowledgments (first edition)*

Even more technical review:

We're also extremely grateful to our esteemed technical reviewer **David Powers**. We have a real love/hate relationship with David because he made us work so hard, but the result was *oh so worth it*. The truth be told, based on David's comments, we made significant changes to this book and it is technically twice the book it was before. Thank you, David.

Esteemed Reviewer David Powers

Don't let the sweater fool you—this guy is hardcore (technically of course).

At O'Reilly:

Our biggest thanks to our editor, **Brett McLaughlin**, who cleared the path for this book, removed every obstacle to its completion, and sacrificed family time to get it done. Brett also did hard editing time on this book (not an easy task for a Head First title). Thanks, Brett; this book wouldn't have happened without you.

Brett McLaughlin

Our sincerest thanks to the whole O'Reilly team: **Greg Corrin, Glenn Bisignani, Tony Artuso,** and **Kyle Hart** all led the way on marketing and we appreciate their out-of-the-box approach. Thanks to **Ellie Volkhausen** for her inspired cover design that continues to serve us well, and to **Karen Montgomery** for stepping in and bringing life to this book's cover. Thank you, as always, to **Colleen Gorman** for her hardcore copyedit (and for keeping it all fun). And we couldn't have pulled off a color book like this without **Sue Willing** and **Claire Cloutier**.

No Head First acknowledgment would be complete without thanking **Mike Loukides** for shaping the Head First concept into a series, and to **Tim O'Reilly** for always being there and his continued support. Finally, thanks to **Mike Hendrickson** for bringing us into the Head First family and having the faith to let us run with it.

Kathy Sierra and Bert Bates:

Last, and anything but least, to **Kathy Sierra** and **Bert Bates**, our partners in crime and the BRAINS who created the series. Thanks, guys, for trusting us *even more* with your baby. We hope once again we've done it justice. The three-day jam session was the highlight of writing the book, we hope to repeat it soon. Oh, and next time around, can you give LTJ a call and tell him he's just going to have to make a trip back to Seattle?

Bert Bates

Kathy Sierra

Hard at work researching Head First Parelli

Kara

*The large number of acknowledgments is because we're testing the theory that everyone mentioned in a book acknowledgment will buy at least one copy, probably more, what with relatives and everything. If you'd like to be in the acknowledgment of our *next* book, and you have a large family, write to us.

Tech reviewers (second edition)

We couldn't sleep at night without knowing that our high-powered HTML & CSS reviewer, **David Powers**, has scoured this book for inaccuracies. Truth is, so many years had passed since the first edition that we had to hire a private detective to locate him (it's a long story, but he was finally located in his underground HTML & CSS lair and research lab). Anyway, more seriously, while all the technical faults in this book sit solely with the authors (that's us), we can assure you in every case David tried to make sure we did things right. Once again, David was instrumental in the writing of this book.

We're extremely grateful for everyone on our technical review team. **Joe Konior** joined us once again for this edition, along with **Dawn Griffiths** (co-author of *Head First C*), and **Shelley Powers** (an HTML & CSS "power"house who has been writing about the Web for years). Once again, you all rock! Your feedback was amazingly thorough, detailed, and helpful. Thank you.

David Powers

Less pink, more HTML & CSS power!

Joe Konior

Dawn Griffiths

Mike Hendrickson

Lou Barr

Acknowledgments (second edition)

Our biggest thanks to our chief editor, **Mike Hendrickson**, who made this book happen in every way (other than actually writing it) , was there for us the entire journey, and more importantly (the biggest thing any editor can do) totally trusted us to get it done! Thanks, Mike; none of our books would have happened without you. You've been our champion for well over a decade and we love you for it!

Of course it takes a village to publish a book, and behind the scenes a talented and friendly group at O'Reilly made it all happen. Our sincerest thanks to the whole O'Reilly team: **Kristen Borg** (production editor extraordinaire); the brilliant **Rachel Monaghan** (proofreader); **Ron Strauss** for his meticulous index; **Rebecca Demarest** for illustration help; **Karen Montgomery**, ace cover designer; and last but definitely not least, **Louise Barr**, who always helps our pages look better.

Safari® Books Online

Safari® Books Online is an on-demand digital library that lets you easily search over 7,500 technology and creative reference books and videos to find the answers you need quickly.

With a subscription, you can read any page and watch any video from our library online. Read books on your cell phone and mobile devices. Access new titles before they are available for print, and get exclusive access to manuscripts in development and post feedback for the authors. Copy and paste code samples, organize your favorites, download chapters, bookmark key sections, create notes, print out pages, and benefit from tons of other time-saving features.

O'Reilly Media has uploaded this book to the Safari Books Online service. To have full digital access to this book and others on similar topics from O'Reilly and other publishers, sign up for free at *http://my.safaribooksonline.com*.

1 getting to know HTML

The Language of the Web

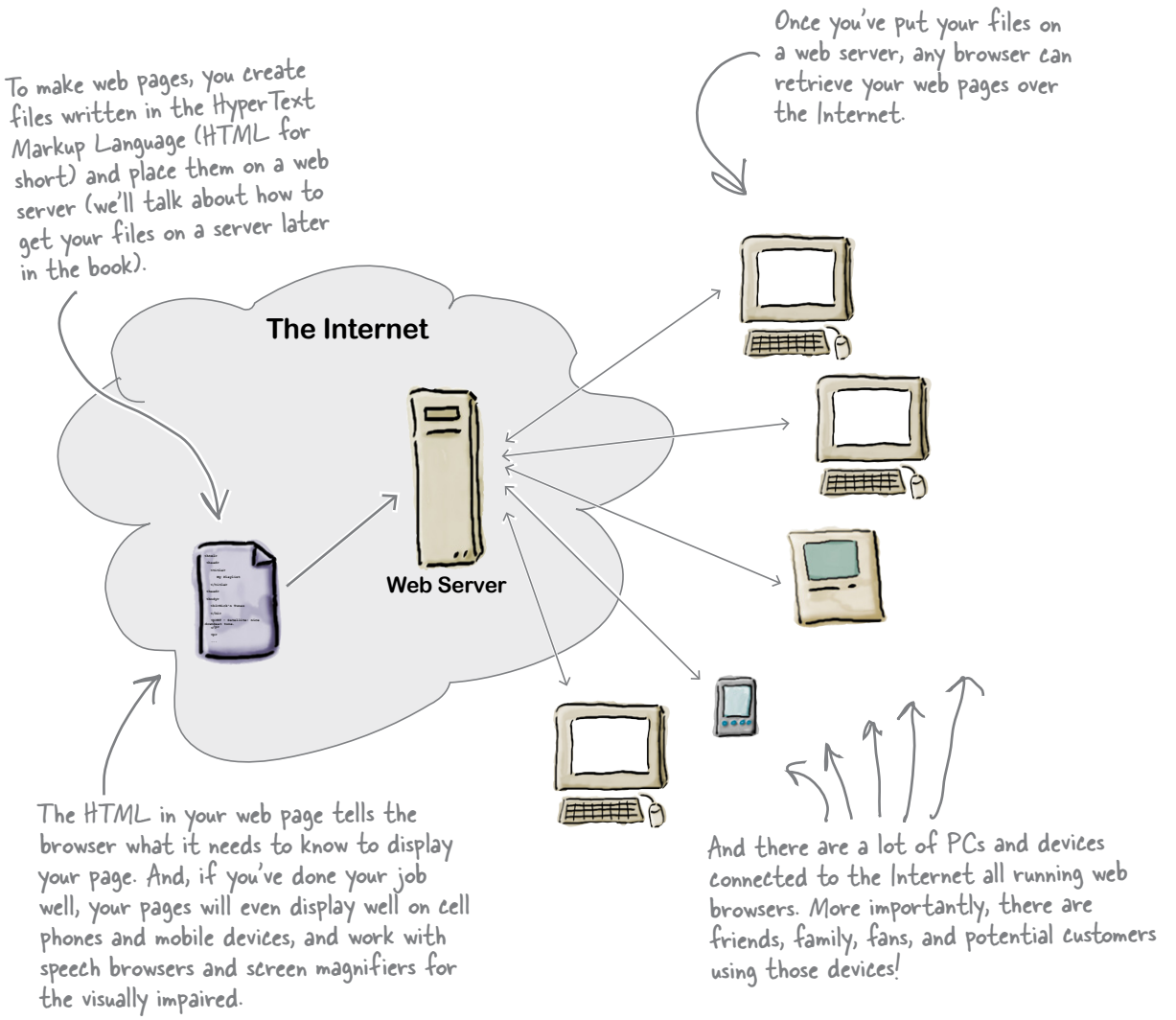
The only thing that is standing between you and getting yourself on the Web is learning to speak the lingo:

HyperText Markup Language, or HTML for short. So, get ready for some language lessons. After this chapter, not only are you going to understand some basic **elements** of HTML, but you'll also be able to speak HTML with a little **style.** Heck, by the end of this book you'll be talking HTML like you grew up in Webville.

The Web
~~Video~~ killed the radio star

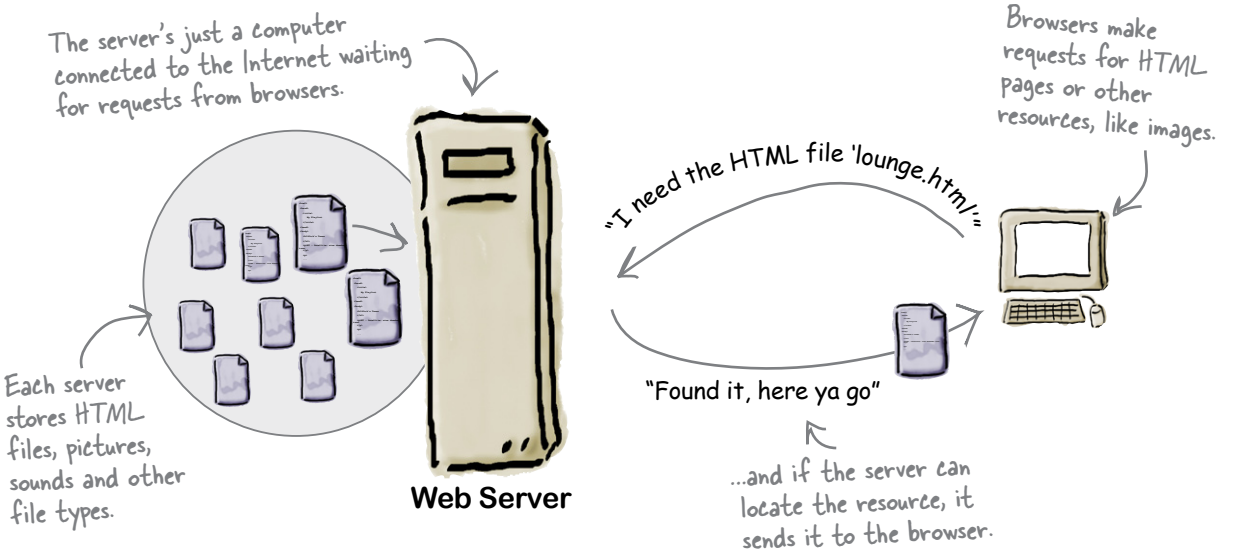
Want to get an idea out there? Sell something? Just need a creative outlet? Turn to the Web—we don't need to tell you it has become the universal form of communication. Even better, it's a form of communication **YOU** can participate in.

But if you really want to use the Web effectively, you've got to know a few things about **HTML**—not to mention, a few things about how the Web works too. Let's take a look from 30,000 feet:



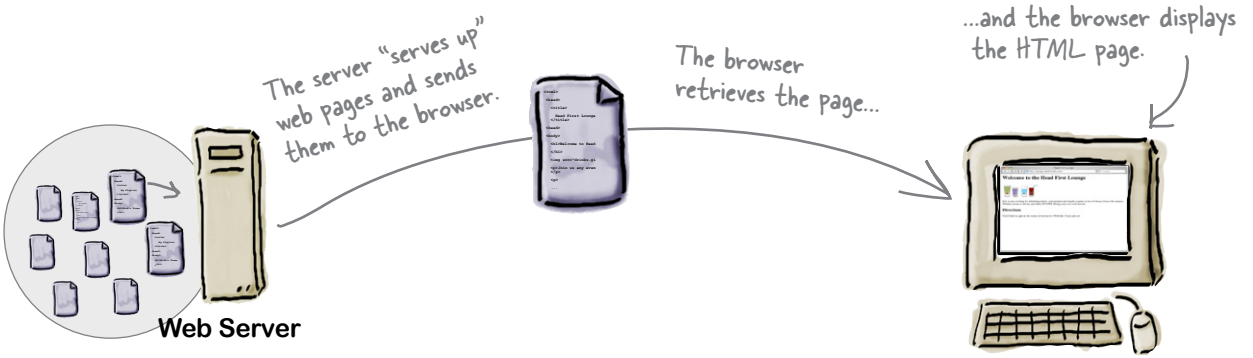
What does the web server do?

Web servers have a full-time job on the Internet, tirelessly waiting for requests from web browsers. What kinds of requests? Requests for web pages, images, sounds, or maybe even a video. When a server gets a request for any of these resources, the server finds the resource, and then sends it back to the browser.



What does the web browser do?

You already know how a browser works: you're surfing around the Web and you click on a link to visit a page. That click causes your browser to request an HTML page from a web server, retrieve it, and display the page in your browser window.



But how does the browser know how to display a page? That's where HTML comes in. HTML tells the browser all about the content and structure of the page. Let's see how that works…

What you write (the HTML)

So, you know HTML is the key to getting a browser to display your pages, but what exactly does HTML look like? And what does it do?

Let's have a look at a little HTML…imagine you're going to create a web page to advertise the *Head First Lounge*, a local hangout with some good tunes, refreshing elixirs, and wireless access. Here's what you'd write in HTML:

```
<html>
  <head>
    <title>Head First Lounge</title>  (A)
  </head>
  <body>
    <h1>Welcome to the Head First Lounge</h1>  (B)
    <img src="drinks.gif">  (C)
    <p>
  (D)  Join us any evening for refreshing elixirs,
       conversation and maybe a game or
       two of <em>Dance Dance Revolution</em>.  (E)
       Wireless access is always provided;
       BYOWS (Bring your own web server).
    </p>
    <h2>Directions</h2>  (F)
    <p>
  (G)  You'll find us right in the center of
       downtown Webville. Come join us!
    </p>
  </body>
</html>
```

We don't expect you to know HTML yet.

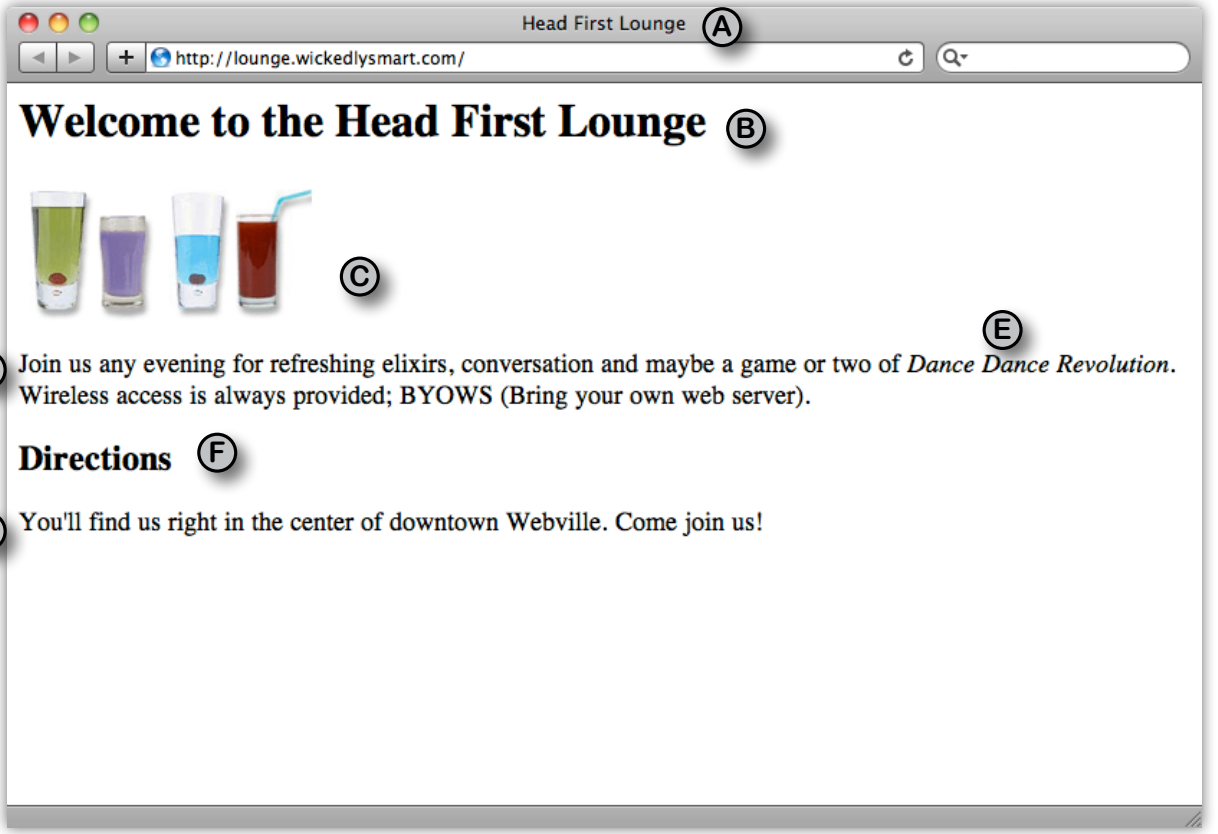
At this point you should just be getting a feel for what HTML looks like; we're going to cover everything in detail in a bit. For now, study the HTML and see how it gets represented in the browser on the next page. Be sure to pay careful attention to each letter annotation and how and where it is displayed in the browser.

What the browser creates

When the browser reads your HTML, it interprets all the *tags* that surround your text. Tags are just words or characters in angle brackets, like <head>, <p>, <h1>, and so on. The tags tell the browser about the *structure and meaning* of your text. So rather than just giving the browser a bunch of text, with HTML you can use tags to tell the browser what text is in a heading, what text is a paragraph, what text needs to be emphasized, or even where images need to be placed.

Let's check out how the browser interprets the tags in the Head First Lounge:

Notice how each tag in the HTML maps to what the browser displays.



there are no Dumb Questions

Q: So HTML is just a bunch of tags that I put around my text?

A: For starters. Remember that HTML stands for HyperText Markup Language, so HTML gives you a way to "mark up" your text with tags that tell the browser how your text is structured. But there is also the HyperText aspect of HTML, which we'll talk about a little later in the book.

Q: How does the browser decide how to display the HTML?

A: HTML tells your browser about the structure of your document: where the headings are, where the paragraphs are, what text needs emphasis, and so on. Given this information, browsers have built-in default rules for how to display each of these elements.

But you don't have to settle for the default settings. You can add your own style and formatting rules with CSS that determine font, colors, size, and a lot of other characteristics of your page. We'll get back to CSS later in the chapter.

Q: The HTML for the Head First Lounge has all kinds of indentation and spacing, and yet I don't see that when it is displayed in the browser. How come?

A: Correct, and good catch. Browsers ignore tabs, returns, and most spaces in HTML documents. Instead, they rely on your markup to determine where line and paragraph breaks occur.

So why did we insert our own formatting if the browser is just going to ignore it? To help us more easily read the document when we're editing the HTML. As your HTML documents become more complicated, you'll find a few spaces, returns, and tabs here and there really help to improve the readability of the HTML.

Q: So there are two levels of headings, <h1> and a subheading <h2>?

A: Actually there are six, <h1> through <h6>, which the browser typically displays in successively smaller font sizes. Unless you are creating a complex and large document, you typically won't use headings beyond <h3>.

Q: Why do I need the <html> tag? Isn't it obvious this is an HTML document?

A: The <html> tag tells the browser your document is actually HTML. While some browsers will forgive you if you omit it, some won't, and as we move toward "industrial-strength HTML" later in the book, you'll see it is quite important to include this tag.

Q: What makes a file an HTML file?

A: An HTML file is a simple text file. Unlike a word processing file, there is no special formatting embedded in it. By convention, we add an ".html" to the end of the filename to give the operating system a better idea of what the file is. But, as you've seen, what really matters is what we put inside the file.

Q: Everyone is talking about HTML5. Are we using it? If so, why aren't we saying "HTML-FIVE" instead of "HTML"?

A: You're learning about HTML, and HTML5 just happens to be the latest version of HTML. HTML5 has had a lot of attention recently, and that's because it simplifies many of the ways we write HTML and enables some new functionality, which we're going to cover in this book. It also provides some advanced features through its JavaScript application programming interfaces (APIs), and those are covered in Head First HTML5 Programming.

Q: Markup seems silly. What-you-see-is-what-you-get applications have been around since, what, the '70s? Why isn't the Web based on a format like Microsoft Word or a similar application?

A: The Web is created out of text files without any special formatting characters. This enables any browser in any part of the world to retrieve a web page and understand its contents. There are WYSIWYG applications out there like Dreamweaver, and they work great. But in this book we're going to take it down to the bare metal, and start with text. Then you're in good shape to understand what your Dreamweaver application is doing behind the scenes.

Q: Is there any way to put comments to myself in HTML?

A: Yes, if you place your comments in between <!-- and --> the browser will totally ignore them. Say you wanted to write a comment "Here's the beginning of the lounge content." You'd do that like this:

```
<!-- Here's the beginning of
the lounge content -->
```

Notice that you can put comments on multiple lines. Keep in mind anything you put between the "<!--" and the "-->", even HTML, will be ignored by the browser.

You're closer to learning HTML than you think…

Here's the HTML for the Head First Lounge again. Take a look at the tags and see if you can guess what they tell the browser about the content. Write your answers in the space on the right; we've already done the first couple for you.

```
<html>
  <head>
    <title>Head First Lounge</title>
  </head>
  <body>
    <h1>Welcome to the Head First Lounge</h1>
    <img src="drinks.gif">
    <p>
       Join us any evening for refreshing elixirs,
       conversation and maybe a game or
       two of <em>Dance Dance Revolution</em>.
       Wireless access is always provided;
       BYOWS (Bring your own web server).
    </p>
    <h2>Directions</h2>
    <p>
       You'll find us right in the center of
       downtown Webville. Come join us!
    </p>
  </body>
</html>
```

Tells the browser this is the start of HTML.

Starts the page "head" (more about this later).


```
<html>                                                    Tells the browser this is the start of HTML.
  <head>                                                  Starts the page "head".
    <title>Head First Lounge</title>                      Gives the page a title.
  </head>                                                 End of the head.
  <body>                                                  Start of the body of page.
    <h1>Welcome to the Head First Lounge</h1>             Tells browser that "Welcome to..." is a heading.
    <img src="drinks.gif">                                Places the image "drinks.gif" here.
    <p>                                                   Start of a paragraph.
       Join us any evening for refreshing elixirs,
       conversation and maybe a game or
       two of <em>Dance Dance Revolution</em>.            Puts emphasis on Dance Dance Revolution.
       Wireless access is always provided;
       BYOWS (Bring your own web server).
    </p>                                                  End of paragraph.
    <h2>Directions</h2>                                   Tells the browser that "Directions" is a subheading.
    <p>                                                   Start of another paragraph.
       You'll find us right in the center of
       downtown Webville. Come join us!
    </p>                                                  End of paragraph.
  </body>                                                 End of the body.
</html>                                                   Tells the browser this is the end of the HTML.
```

Your big break at Starbuzz Coffee

Starbuzz Coffee has made a name for itself as the fastest growing coffee shop around. If you've seen one on your local corner, look across the street—you'll see another one.

In fact, they've grown so quickly, they haven't even managed to put up a web page yet…and therein lies your big break: By chance, while buying your Starbuzz Chai Tea, you run into the Starbuzz CEO…

BRAIN POWER

Decisions, decisions.
Check your first priority below (choose only one):

- ☐ A. Give dog a bath.
- ☐ B. Finally get my checking account balanced.
- ☐ C. Take the Starbuzz gig and launch BIG-TIME web career.
- ☐ D. Schedule dentist appointment.

Sharpen your pencil

Take a look at the napkin. Can you determine the *structure* of it? In other words, are there obvious headings? Paragraphs? Is it missing anything like a title?

Go ahead and mark up the napkin (using your pencil) with any structure you see, and add anything that is missing.

You'll find our answers at the end of Chapter 1.

* If by chance you chose option A, B, or D on the previous page, we recommend you donate this book to a good library, use it as kindling this winter, or what the heck, go ahead and sell it on Amazon and make some cash.

Creating the Starbuzz web page

Of course, the only problem with all this is that you haven't actually created any web pages yet. But that's why you decided to dive head first into HTML, right?

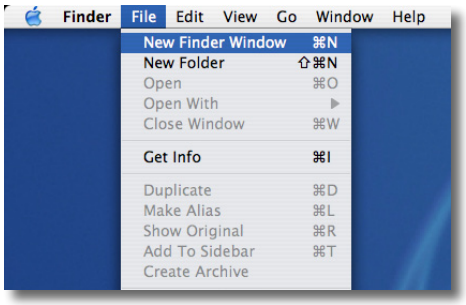
No worries, here's what you're going to do on the next few pages:

1. **Create an HTML file using your favorite text editor.**
2. **Type in the menu the Starbuzz CEO wrote on the napkin.**
3. **Save the file as "index.html".**
4. **Open the file "index.html" in your favorite browser, step back, and watch the magic happen.**

Creating an HTML file (Mac)

All HTML files are text files. To create a text file, you need an application that allows you to create plain text without throwing in a lot of fancy formatting and special characters. You just need plain, pure text.

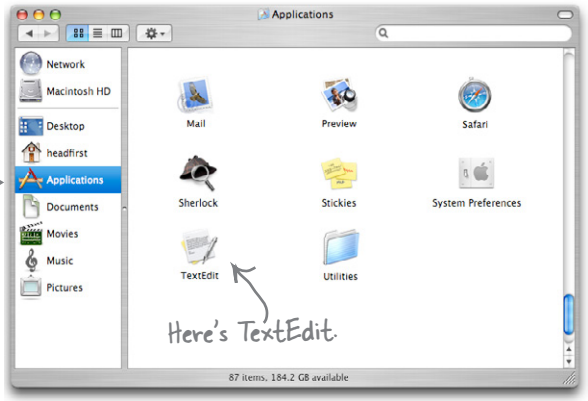
We'll use TextEdit on the Mac in this book; however, if you prefer another text editor, that should work fine as well. And, if you're running Windows, you'll want to skip ahead a couple of pages to the Windows instructions.



Step one:

Navigate to your **Applications** folder

The TextEdit application is in the *Applications* folder. The easiest way to get there is to choose New Finder Window from the Finder's File menu and then look for the Application directly in your shortcuts. When you've found it, click on Applications.



Step two:

Locate and run **TextEdit**

You'll probably have lots of applications listed in your *Applications* folder, so scroll down until you see TextEdit. To run the application, double-click on the TextEdit icon.

Step three (optional):

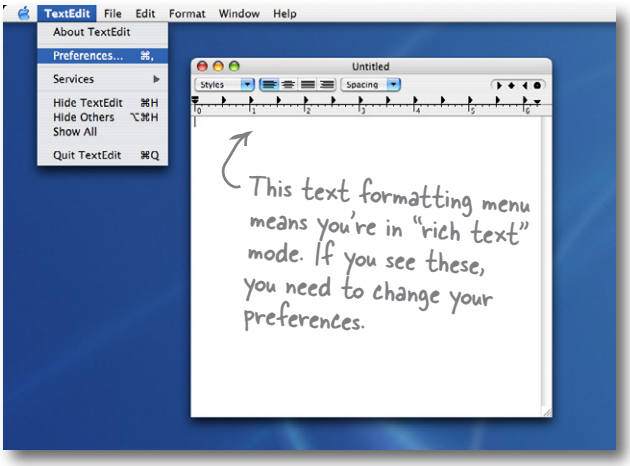
Keep **TextEdit** in your **Dock**

If you want to make your life easier, click and hold on the TextEdit icon in the Dock (this icon appears once the application is running). When it displays a pop-up menu, choose Options, then "Keep in Dock." That way, the TextEdit icon will always appear in your Dock and you won't have to hunt it down in the *Applications* folder every time you need to use it.

Step four:

Change your TextEdit **Preferences**

By default, TextEdit is in "rich text" mode, which means it will add its own formatting and special characters to your file when you save it—not what you want. So, you'll need to change your TextEdit Preferences so that TextEdit saves your work as a pure text file. To do this, first choose the Preferences menu item from the TextEdit menu.



Step five:

Set Preferences for **Plain text**

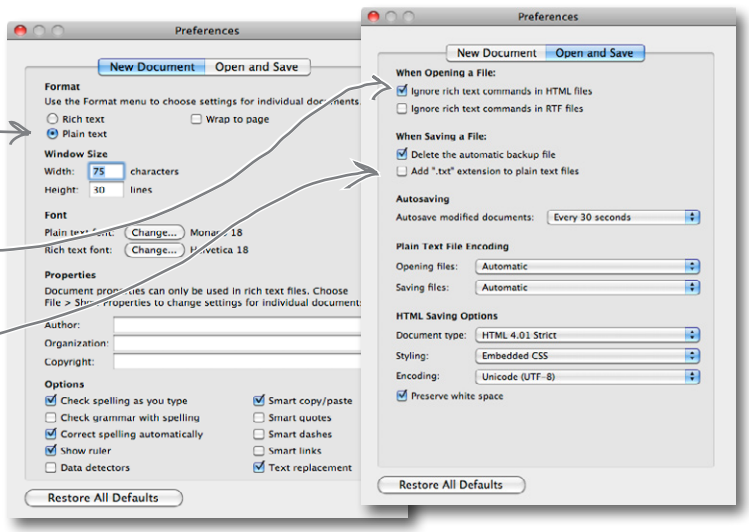
Once you see the Preferences dialog box, there are three things you need to do.

First, choose "Plain text" as the default editor mode in the New Document tab.

In the "Open and Save" tab, make sure "Ignore rich text commands in HTML files" is checked.

Last, make sure that the "Add .txt extension to plain text files" is **un**checked.

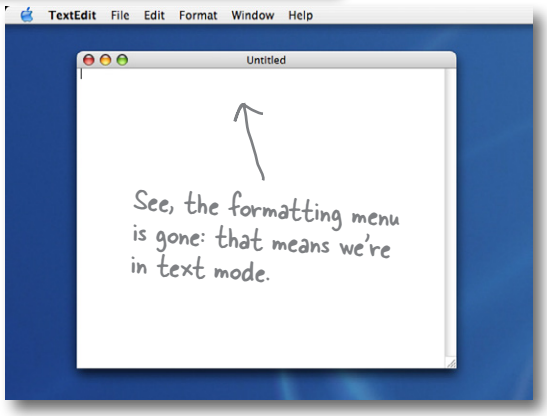
That's it; to close the dialog box, click on the red button in the top-left corner.



Step six:

Quit and **restart**

Now quit out of TextEdit by choosing Quit from the TextEdit menu, and then restart the application. This time, you'll see a window with no fancy text formatting menus at the top. You're now ready to create some HTML.



Creating an HTML file (Windows)

Or another version of Windows

If you're reading this page you must be a Windows 7 user. If you're not, you might want to skip a couple of pages ahead. Or, if you just want to sit in the back and not ask questions, we're okay with that too.

If you're using another version of Windows, you'll find Notepad there as well.

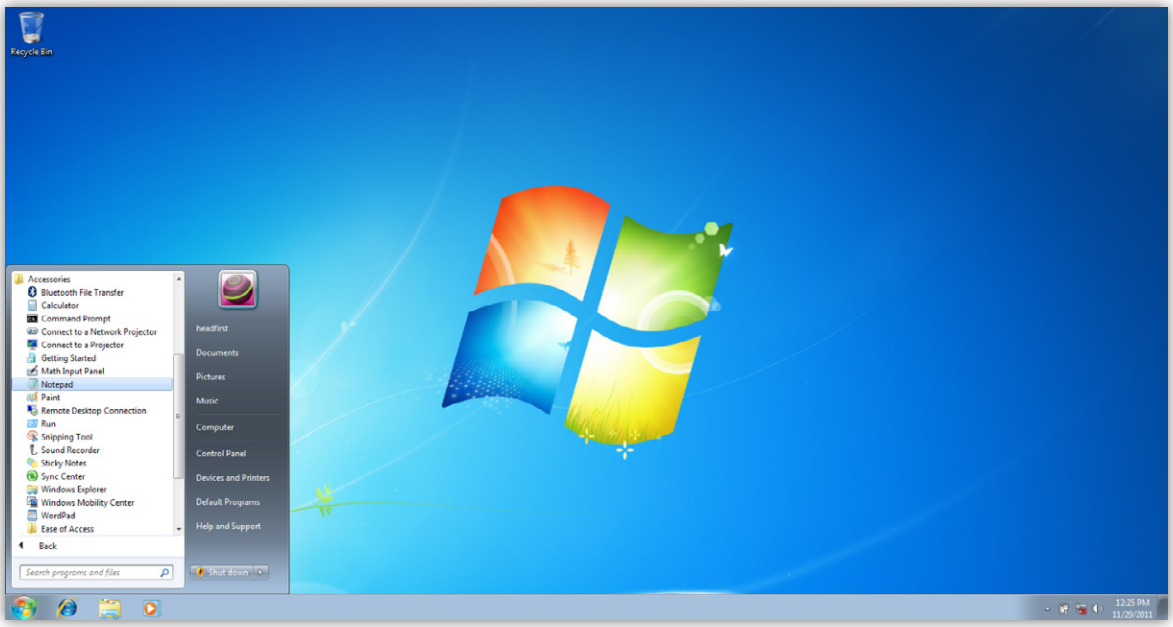
To create HTML files in Windows 7, we're going to use Notepad—it ships with every copy of Windows, the price is right, and it's easy to use. If you've got your own favorite editor that runs on Windows 7, that's fine too; just make sure you can create a plain-text file with an ".html" extension.

Assuming you're using Notepad, here's how you're going to create your first HTML file.

Step one:

Open the **Start** menu and navigate to Notepad.

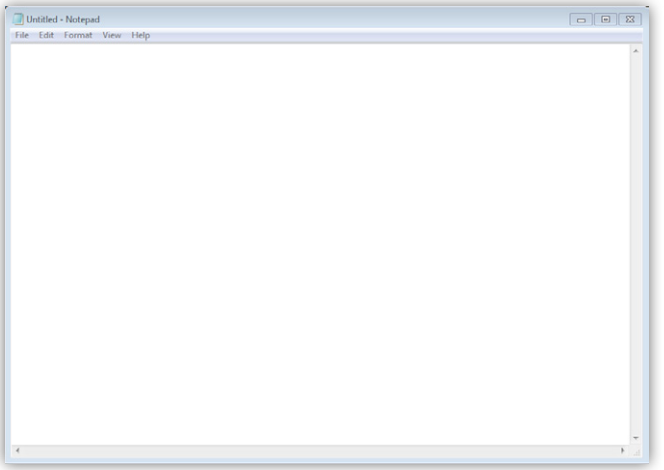
You'll find the Notepad application in *Accessories*. The easiest way to get there is to click on the Start menu, then on All Programs, then Accessories. You'll see Notepad listed there.



Step two:

Open **Notepad**.

Once you've located Notepad in the *Accessories* folder, go ahead and click on it. You'll see a blank window ready for you to start typing HTML.



But recommended

Step three (optional):

Don't hide extensions of well-known file types.

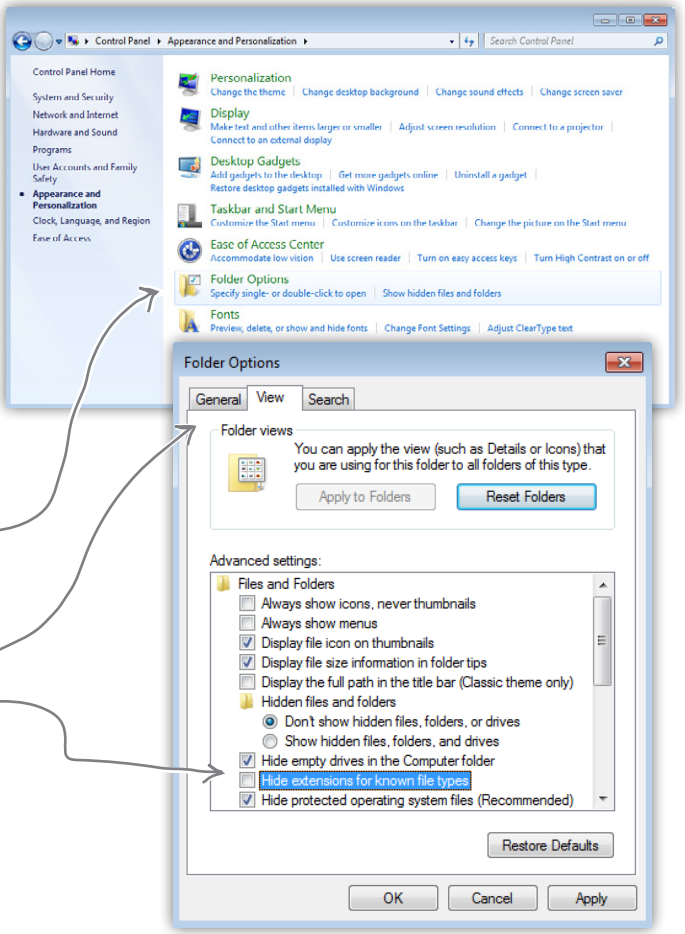
By default, Windows File Explorer hides the file extensions of well-known file types. For example, a file named "Irule.html" will be shown in the Explorer as "Irule" without its ".html" extension.

It's much less confusing if Windows shows you these extensions, so let's change your folder options so you can see them.

First, open Folder Options by clicking the Start button, clicking Control Panel, clicking "Appearance and Personalization," and then clicking Folder Options.

Next, in the View tab, under "Advanced settings," scroll down until you see "Hide extensions for known file types" and *uncheck* this option.

That's it. Click on the OK button to save the preference and you'll now see the file extensions in the Explorer.

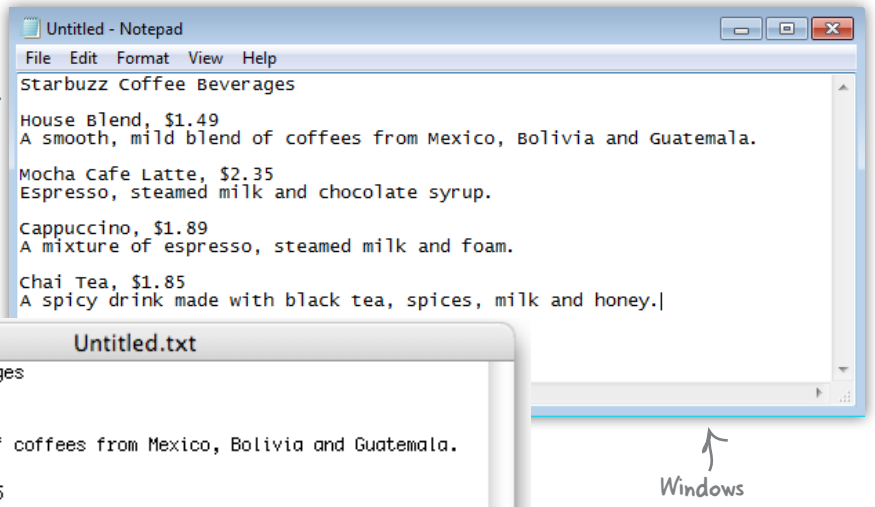


Meanwhile, back at Starbuzz Coffee...

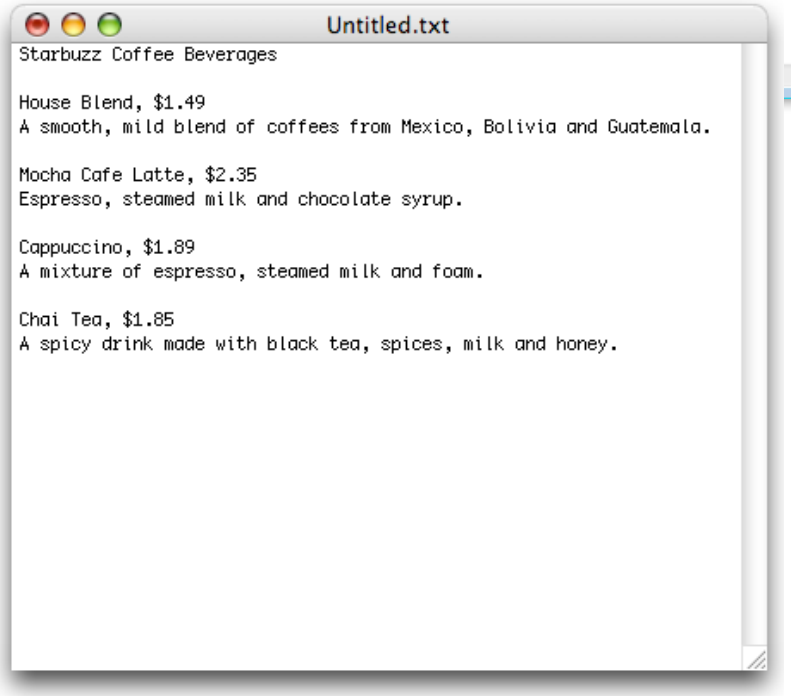
Okay, now that you know the basics of creating a plain-text file, you just need to get some content into your text editor, save it, and then load it into your browser.

Start by typing in the beverages straight from the CEO's napkin; these beverages are the content for your page. You'll be adding some HTML markup to give the content some structure in a bit, but for now, just get the basic content typed in. While you're at it, go ahead and add "Starbuzz Coffee Beverages" at the top of the file.

Type in the info from the napkin like this.



Windows



Mac

there are no Dumb Questions

Q: Why am I using a simple text editor? Aren't there powerful tools like Dreamweaver and Expression Web for creating web pages?

A: You're reading this book because you want to understand the true technologies used for web pages, right? Now those are all great tools, but they do a lot of the work for you, and until you are a master of HTML and CSS, you want to learn this stuff without a big tool getting in your way.

Once you're a master, however, these tools do provide some nice features like syntax checking and previews. At that point, when you view the "code" window, you'll understand everything in it, and you'll find that changes to the raw HTML and CSS are often a lot faster than going through a user interface. You'll also find that as standards change, these tools aren't always updated right away and may not support the most recent standards until their next release cycle. Since you'll know how to change the HTML and CSS without the tool, you'll be able to keep up with the latest and greatest all the time.

There are many more fully featured editors that include great features like clips (for automatically inserting bits of HTML you write often), preview (for previewing directly in the editor before you test in the browser), syntax coloring (so tags are a different color from content), and much more. Once you get the hang of writing basic HTML and CSS in a simple editor, it may be worth checking out one of the fancier editors, such as Coda, TextMate, CoffeeCup, or Aptana Studio. There are many out there to choose from (both free and not).

Q: I get the editor, but what browser am I supposed to be using? There are so many—Internet Explorer, Chrome, Firefox, Opera, Safari—what's the deal?

A: The simple answer: use whatever browser you like. HTML and CSS are industry standards, which means that all browsers try to support HTML and CSS in the same way (just make sure you are using the newest version of the browser for the best support).

The complex answer: in reality there are slight differences in the way browsers handle your pages. If you've got users who will be accessing your pages in a variety of browsers, then always test your web page in several different browsers. Some pages will look exactly the same; some won't. The more advanced you become with HTML and CSS, the more these slight differences may matter to you, and we'll get into some of these subtleties throughout the book.

Any of the major browsers—Internet Explorer, Chrome, Firefox, Opera, and Safari—will work for most examples (except where noted); they are all modern browsers with great HTML and CSS support. And as a web developer, you'll be expected to test your code in more than one browser, so we encourage you to download and get familiar with at least two!

Q: I'm creating these files on my own computer—how am I going to view these on the Web?

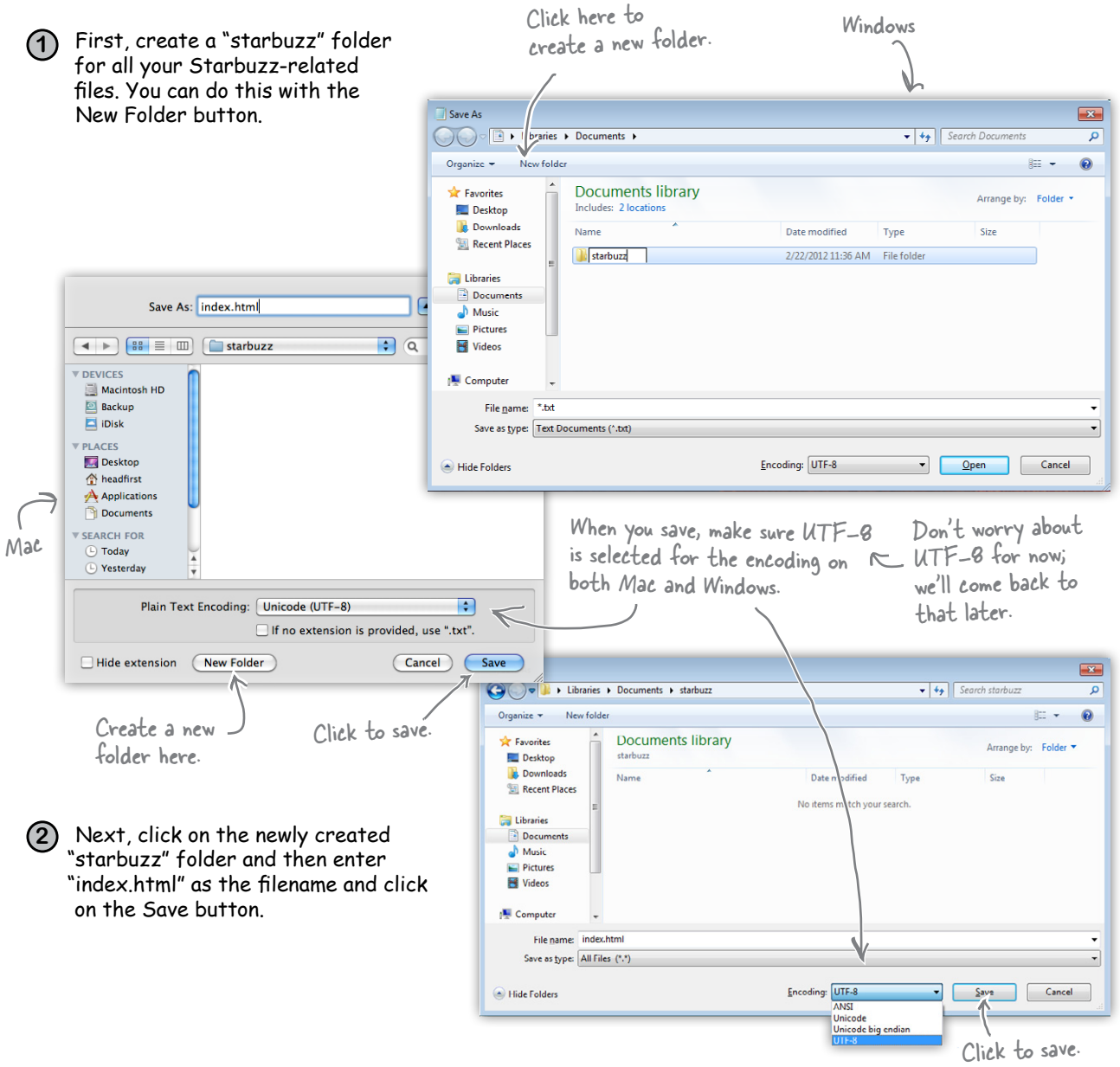
A: That's one great thing about HTML: you can create files and test them on your own computer and then later publish them on the Web. Right now, we're going to worry about how to create the files and what goes in them. We'll come back to getting them on the Web a bit later.

Saving your work

Once you've typed in the beverages from the CEO's napkin, you're going to save your work in a file called "index.html". Before you do that, you'll want to create a folder named "starbuzz" to hold the site's files.

To get this all started, choose Save from the File menu and you'll see a Save As dialog box. Then, here's what you need to do:

1. First, create a "starbuzz" folder for all your Starbuzz-related files. You can do this with the New Folder button.

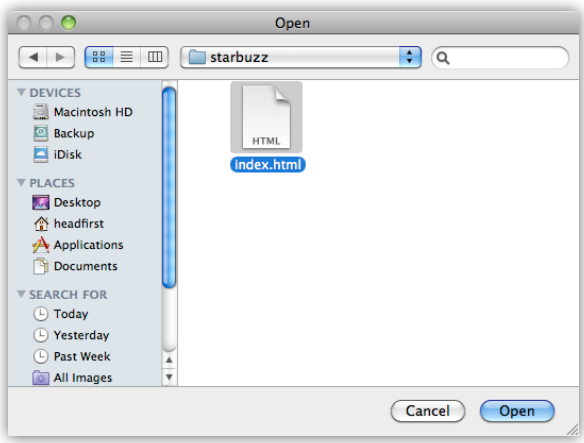


2. Next, click on the newly created "starbuzz" folder and then enter "index.html" as the filename and click on the Save button.

Opening your web page in a browser

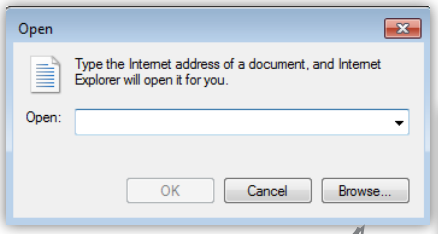
Are you ready to open your first web page? Using your favorite browser, choose "Open File…" (or "Open…" using Windows 7 and Internet Explorer) from the File menu and navigate to your "index.html" file. Select it and click Open.

Mac



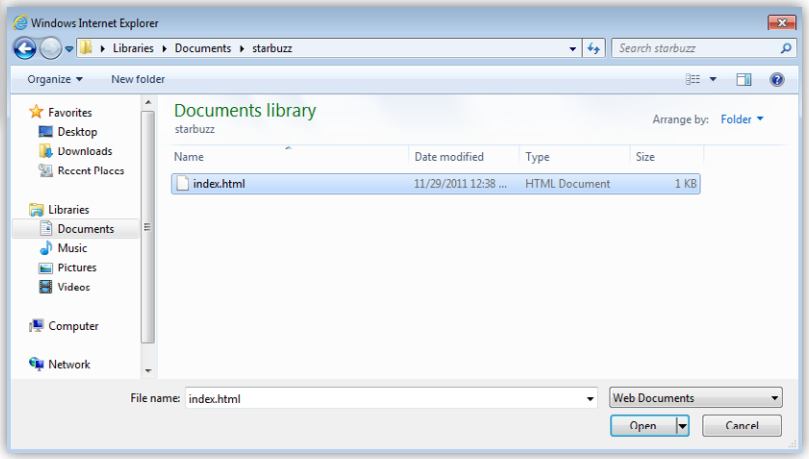
On the Mac, navigate to your file, and select it by clicking on the file icon and then on the Open button.

Windows



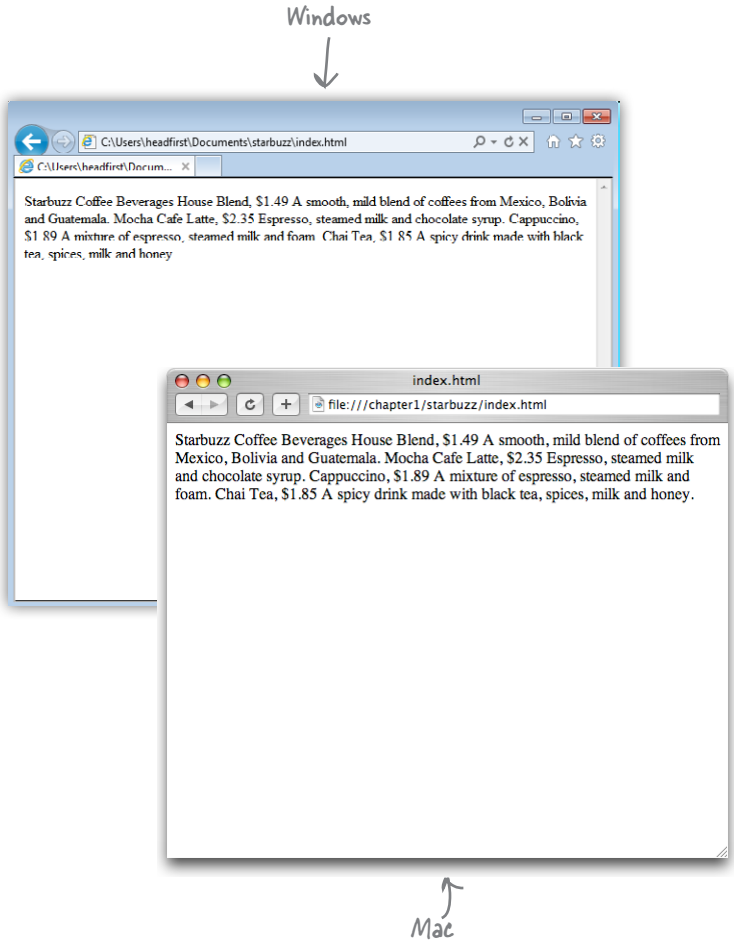
In Windows Internet Explorer it's a two-step process. First, you'll get the Open dialog box.

Then click Browse to get a browse dialog and navigate to where you saved your file.



Take your page for a test drive

Success! You've got the page loaded in the browser, although the results are a little…uh…unsatisfying. But that's just because all you've done so far is go through the mechanics of creating a page and viewing it in the browser. And so far, you've only typed in the *content* of the web page. That's where HTML comes in. HTML gives you a way to tell the browser about the *structure* of your page. What's structure? As you've already seen, it is a way of marking up your text so that the browser knows what's a heading, what text is in a paragraph, what text is a subheading, and so on. Once the browser knows a little about the structure, it can display your page in a more meaningful and readable manner.



Depending on your operating system and browser, often you can just double-click the HTML file or drag it on top of the browser icon to open it. Much simpler.

Markup Magnets

So, let's add that structure…

Your job is to add structure to the text from the Starbuzz napkin. Use the fridge magnets at the bottom of the page to mark up the text so that you've indicated which parts are headings, subheadings and paragraph text. We've already done a few to get you started. You won't need all the magnets below to complete the job; some will be left over.

<h1> Starbuzz Coffee Beverages </h1>

House Blend, $1.49

A smooth, mild blend of coffees from Mexico, Bolivia and Guatemala.

<h2> Mocha Cafe Latte, $2.35 </h2>

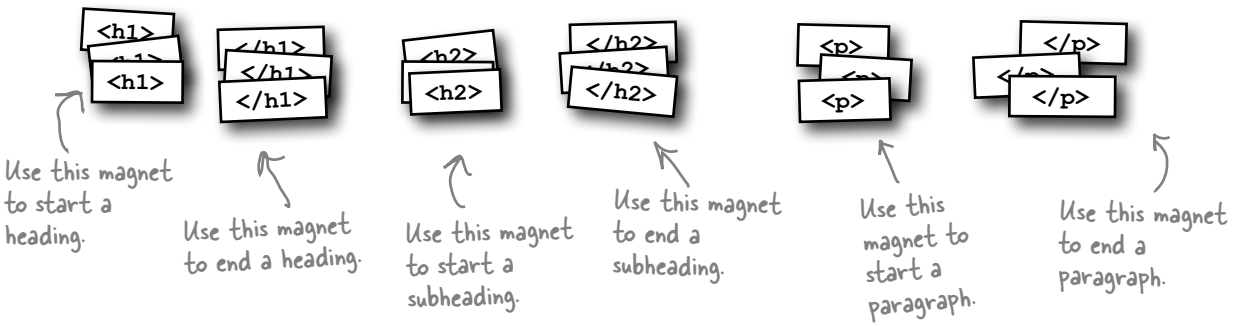
 Espresso, steamed milk and chocolate syrup.

Cappuccino, $1.89

A mixture of espresso, steamed milk and foam.

Chai Tea, $1.85

A spicy drink made with black tea, spices, milk and honey.



Congratulations, you've just written your first HTML!

They might have looked like fridge magnets, but you were really *marking up* your text with HTML. Only, as you know, we usually refer to the magnets as *tags*.

Check out the markup below and compare it to your magnets on the previous page.

Use the <h1> and </h1> tags to mark headings. All the text in between is the actual content of the heading.

```
<h1>Starbuzz Coffee Beverages</h1>

<h2>House Blend, $1.49</h2>
<p>A smooth, mild blend of coffees from Mexico, Bolivia
and Guatemala.</p>

<h2>Mocha Cafe Latte, $2.35</h2>
<p>Espresso, steamed milk and chocolate syrup.</p>

<h2>Cappuccino, $1.89</h2>
<p>A mixture of espresso, steamed milk and foam.</p>

<h2>Chai Tea, $1.85</h2>
<p>A spicy drink made with black tea, spices, milk and
honey.</p>
```

The <h2> and </h2> tags go around a subheading. Think of an <h2> heading as a subheading of an <h1> heading.

The and tags go around a block of text that is a paragraph. That can be one or many sentences.

Notice that you don't have to put matching tags on the same line. You can put as much content as you like between them.

Are we there yet?

You have an HTML file with markup—does that make a web page? Almost. You've already seen the <html>, <head>, <title>, and <body> tags, and we just need to add those to make this a first-class HTML page…

First, surround your HTML with <html> & </html> tags. This tells the browser the content of the file is HTML.

Next add <head> and </head> tags. The head contains information about your web page, like its title. For now, think about it this way: the head allows you to tell the browser things about the web page.

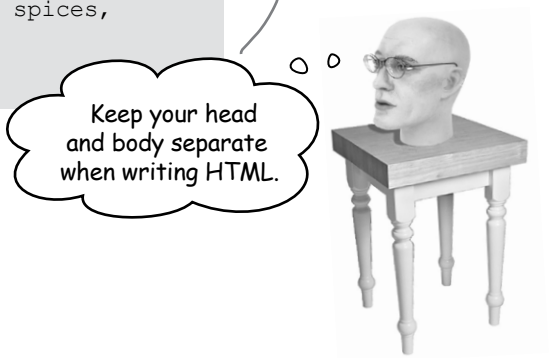
Go ahead and put a title inside the head. The title usually appears at the top of the browser window.

The head consists of the <head> & </head> tags and everything in between.

```
<html>

  <head>
    <title>Starbuzz Coffee</title>
  </head>

  <body>
    <h1>Starbuzz Coffee Beverages</h1>
    <h2>House Blend, $1.49</h2>
    <p>A smooth, mild blend of coffees from Mexico,
       Bolivia and Guatemala.</p>

    <h2>Mocha Cafe Latte, $2.35</h2>
    <p>Espresso, steamed milk and chocolate syrup.</p>

    <h2>Cappuccino, $1.89</h2>
    <p>A mixture of espresso, steamed milk and foam.</p>

    <h2>Chai Tea, $1.85</h2>
    <p>A spicy drink made with black tea, spices,
       milk and honey.</p>
  </body>

</html>
```

The body consists of the <body> & </body> tags and everything in between.

The body contains all the content and structure of your web page—the parts of the web page that you see in your browser.



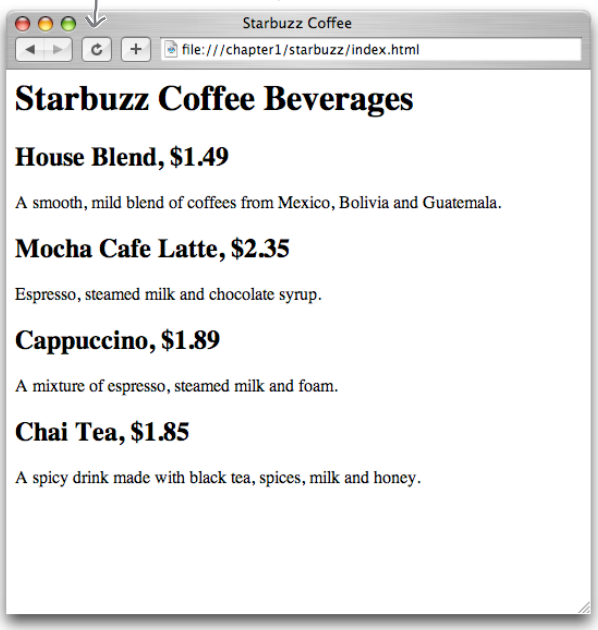
Another test drive

Go ahead and change your "index.html" file by adding in the <head>, </head>, <title>, </title>, <body> and </body> tags. Once you've done that, save your changes and reload the file into your browser.

You can reload the index.html file by selecting the Open File menu item again, or by using your browser's reload button.

Notice that the title, which you specified in the <head> element, shows up here.

Now things look a bit better. The browser has interpreted your tags and created a display for the page that is not only more structured, but also more readable.



Tags dissected

Okay, you've seen a bit of markup, so let's zoom in and take a look at how tags really work.

You usually put tags around some piece of content. Here we're using tags to tell the browser that our content, "Starbuzz Coffee Beverages," is a top-level heading (that is, heading level one).

Here's the opening tag that begins the heading.

This is the closing tag that ends the heading; in this case the </h1> tag is ending an <h1> heading. You know it's a closing tag because it comes after the content, and it's got a "/" before the "h1". All closing tags have a "/" in them.

```
<h1> Starbuzz Coffee Beverages </h1>
```

Tags consist of the tag name surrounded by angle brackets; that is, the < and > characters.

The whole shebang is called an element. In this case, we can call it the <h1> element. An element consists of the enclosing tags and the content in between.

We call an opening tag and its closing tag matching tags.

To tell the browser about the structure of your page, use pairs of tags around your content.

Remember:

Element = Opening Tag + Content + Closing Tag

there are no Dumb Questions

Q: So matching tags don't have to be on the same line?

A: No; remember the browser doesn't really care about tabs, returns, and most spaces. So, your tags can start and end anywhere on the same line, or they can start and end on different lines. Just make sure you start with an opening tag, like <h2>, and end with a closing tag, like </h2>.

Q: Why do the closing tags have that extra "/"?

A: That "/" in the closing tag is to help both you and the browser know where a particular piece of structured content ends. Otherwise, the closing tags would look just like the opening tags, right?

Q: I've noticed the HTML in some pages doesn't always match opening tags with closing tags.

A: Well, the tags are supposed to match. In general, browsers do a pretty good job of figuring out what you mean if you write incorrect HTML. But, as you're going to see, these days there are big benefits to writing totally correct HTML. If you're worried you'll never be able to write perfect HTML, don't be; there are plenty of tools to verify your code before you put it on a web server so the whole world can see it. For now, just get in the habit of always matching your opening tags with closing tags.

Q: Well, what about that <img src="drinks.gif"> tag in the lounge example? Did you forget the closing tag?

A: Wow, sharp eye. There are some elements that use a shorthand notation with only one tag. Keep that in the back of your mind for now, and we'll come back to it in a later chapter.

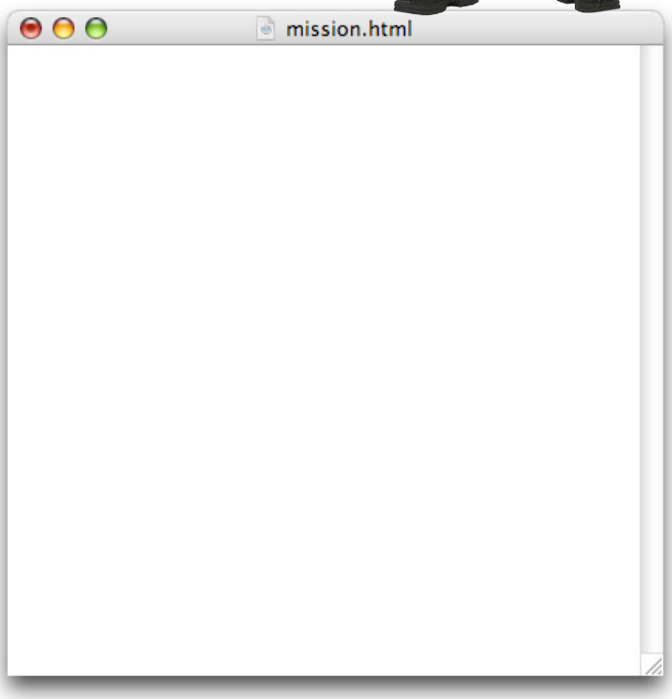
Q: An element is an opening tag + content + closing tag, but can't you have tags inside other tags? Like the <head> and <body> are inside an <html> tag?

A: Yes, HTML tags are often "nested" like that. If you think about it, it's natural for an HTML page to have a body, which contains a paragraph, and so on. So many HTML elements have other HTML elements between their tags. We'll take a good look at this kind of thing in later chapters, but for now just get your mind noticing how the elements relate to each other in a page.

Tags can be a little more interesting than what you've seen so far. Here's the paragraph tag with a little extra added to it. What do you think this does?

```
<p id="houseblend">A smooth, mild
blend of coffees from Mexico, Bolivia
and Guatemala.</p>
```

Exercise



1. Write the HTML for the new "mission.html" page here.

2. Type in your HTML using a text editor, and save it as "mission.html" in the same folder as your "index.html" file.

3. Once you've done that, open "mission.html" in your browser.

4. Check your work at the end of the chapter before moving on…

Okay, it looks like you're getting somewhere. You've got the main page and the mission page all set. But don't forget the CEO said the site needs to look great too. Don't you think it needs a little style?

Right. We have the structure down, so now we're going to concentrate on its presentation.

You already know that HTML gives you a way to describe the structure of the content in your files. When the browser displays your HTML, it uses its own built-in default style to present this structure. But relying on the browser for style obviously isn't going to win you any "designer of the month" awards.

That's where CSS comes in. CSS gives you a way to describe how your content should be presented. Let's get our feet wet by creating some CSS that makes the Starbuzz page look a little more presentable (and launch your web career in the process).

CSS is an abbreviation for Cascading Style Sheets. We'll get into what that all means later, but for now just know that CSS gives you a way to tell the browser how elements in your page should look.

Meet the style element

To add style, you add a new (say it with us) E-L-E-M-E-N-T to your page—the <style> element. Let's go back to the main Starbuzz page and add some style. Check it out…

The <style> element is placed inside the head of your HTML.

Just like other elements, the <style> element has an opening tag, <style>, and a closing tag, </style>.

The <style> tag also has an (optional) attribute, called type, which tells the browser the kind of style you're using. Because you're going to use CSS, you can specify the "text/css" type.

And here's where you're going to define the styles for the page.

```
<html>
  <head>
    <title>Starbuzz Coffee</title>
    <style type="text/css">


    </style>
  </head>
  <body>
    <h1>Starbuzz Coffee Beverages</h1>

    <h2>House Blend, $1.49</h2>
    <p>A smooth, mild blend of coffees from Mexico, Bolivia and
    Guatemala.</p>

    <h2>Mocha Caffe Latte, $2.35</h2>
    <p>Espresso, steamed milk and chocolate syrup.</p>

    <h2>Cappuccino, $1.89</h2>
    <p>A mixture of espresso, steamed milk and milk foam.</p>

    <h2>Chai Tea, $1.85</h2>
    <p>A spicy drink made with black tea, spices, milk and honey.</p>
  </body>
</html>
```

there are no Dumb Questions

Q: An element can have an "attribute"? What does that mean?

A: Attributes give you a way to provide additional information about an element. Like, if you have a <style> element, the attribute allows you to say exactly what kind of style you're talking about. You'll be seeing a lot more attributes for various elements; just remember they give you some extra info about the element.

Q: Why do I have to specify the type of the style ("text/css") as an attribute of the style? Are there other kinds of style?

A: At one time the designers of HTML thought there would be other styles, but as it turns out we've all come to our senses since then and you can just use <style> without an attribute—all the browsers will know you mean CSS. We're disappointed; we were holding our breath for the <style type="50sKitsch"> style. Oh well.

Giving Starbuzz some style...

Now that you've got a `<style>` element in the HTML head, all you need to do is supply some CSS to give the page a little pizzazz. Below you'll find some CSS already "baked" for you. Whenever you see the Reapy Bake logo, you're seeing HTML and CSS that you should type in as-is. *Trust us*. You'll learn how the markup works later, after you've seen what it can do.

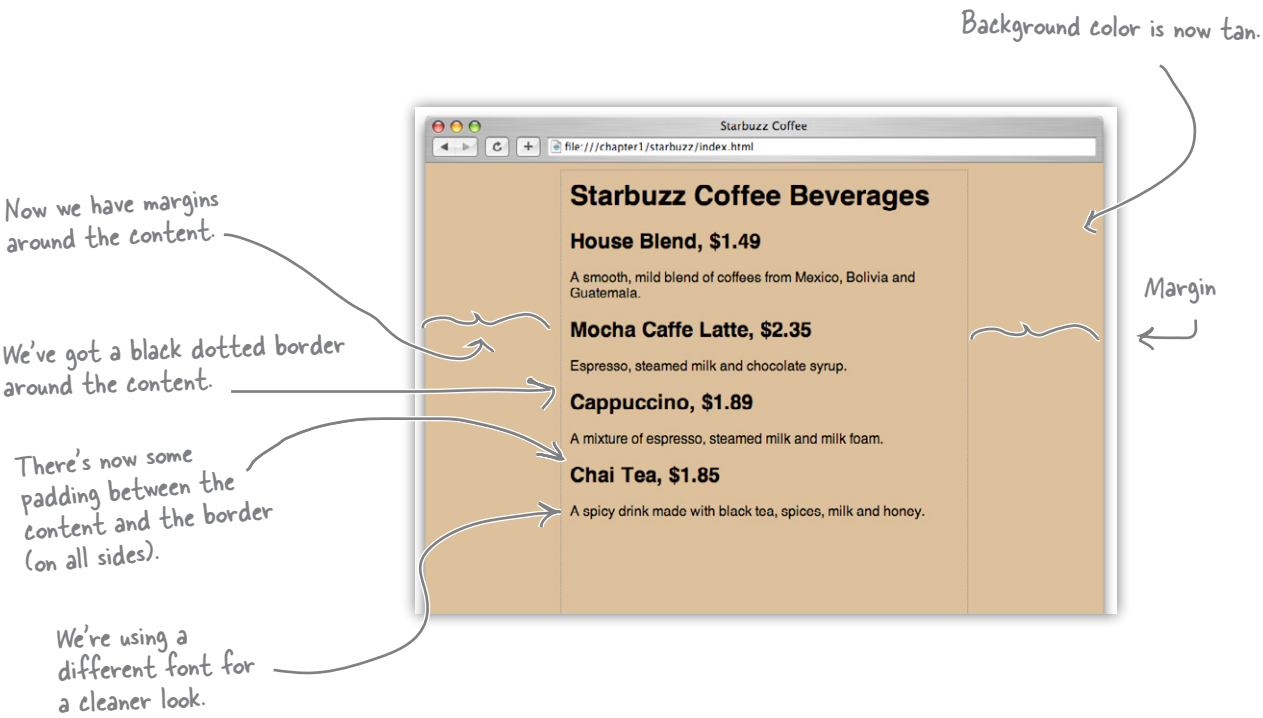
So, take a look at the CSS and then add it to your "index.html" file. Once you've got it typed in, save your file.


```
<html>
  <head>
    <title>Starbuzz Coffee</title>
    <style type="text/css">
      body {
        background-color: #d2b48c;
        margin-left: 20%;
        margin-right: 20%;
        border: 2px dotted black;
        padding: 10px 10px 10px 10px;
        font-family: sans-serif;
      }
    </style>
  </head>

  <body>
    <h1>Starbuzz Coffee Beverages</h1>

    <h2>House Blend, $1.49</h2>
    <p>A smooth, mild blend of coffees from Mexico, Bolivia and
    Guatemala.</p>

    <h2>Mocha Caffe Latte, $2.35</h2>
    <p>Espresso, steamed milk and chocolate syrup.</p>

    <h2>Cappuccino, $1.89</h2>
    <p>A mixture of espresso, steamed milk and milk foam.</p>

    <h2>Chai Tea, $1.85</h2>
    <p>A spicy drink made with black tea, spices, milk and honey.</p>
  </body>
</html>
```

CSS uses a syntax that is totally different from HTML.

Cruisin' with style...

It's time for another test drive, so reload your "index.html" file again. This time, you'll see the Starbuzz web page has a whole new look.



Watch it!

If you're using IE, you might not see the border.

Internet Explorer does not display the border around the body correctly. Try loading the page in Firefox, Chrome or Safari to see the border.

WHO DOES WHAT?

Even though you've just glanced at CSS, you've already begun to see what it can do. Match each line in the style definition to what it does.

```
background-color: #d2b48c;

margin-left: 20%;
margin-right: 20%;

border: 2px dotted black;

padding: 10px 10px 10px 10px;

font-family: sans-serif;
```

Defines the font to use for text.

Defines a border around the body that is dotted and the color black.

Sets the left and right margins to take up 20% of the page each.

Sets the background color to tan.

Creates some padding around the body of the page.

there are no Dumb Questions

Q: CSS looks like a totally different language than HTML. Why have two languages? That's just more for me to learn, right?

A: You are quite right that HTML and CSS are completely different languages, but that is because they have very different jobs. Just like you wouldn't use English to balance your checkbook, or math to write a poem, you don't use CSS to create structure or HTML to create style because that's not what they were designed for. While this does mean you need to learn two languages, you'll discover that because each language is good at what it does, this is actually easier than if you had to use one language to do both jobs.

Q: #d2b48c doesn't look like a color. How is #d2b48c the color "tan"?

A: There are a few different ways to specify colors with CSS. The most popular is called a "hex code," which is what #d2b48c is. This really is a tan color. For now, just go with it, and we'll be showing you exactly how #d2b48c is a color a little later.

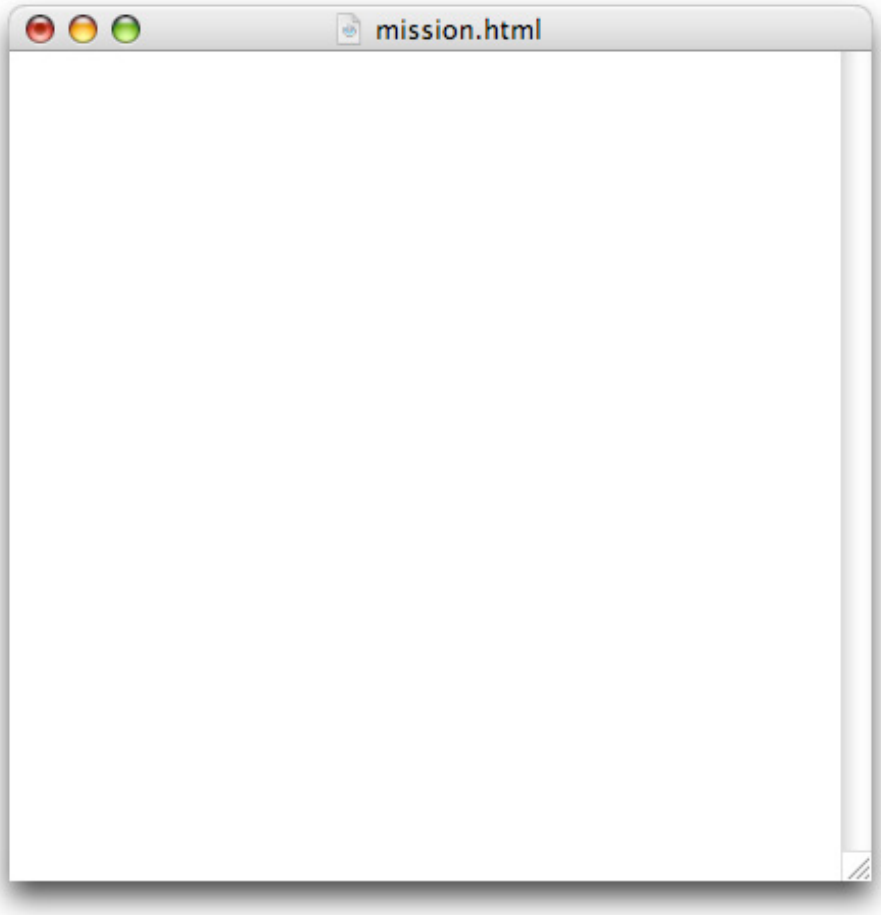
Q: Why is there a "body" in front of the CSS rules? What does that mean?

A: The "body" in the CSS means that all the CSS between the "{" and "}" applies to content within the HTML <body> element. So when you set the font to sans-serif, you're saying that the default font within the body of your page should be sans-serif.

We'll go into a lot more detail about how CSS works shortly, so keep reading. Soon, you'll see that you can be a lot more specific about how you apply these rules, and by doing so, you can create some pretty cool designs.

Now that you've put a little style in the Starbuzz "index.html" page, go ahead and update your "mission.html" page to have the same style.

1. Write the HTML for the "mission.html" page below, and then add the new CSS.
2. Update your "mission.html" file to include the new CSS.
3. Once you've done that, reload "mission.html" in your browser.
4. Make sure your mission page looks like ours at the end of the chapter.



Fireside Chats

Tonight's talk: **HTML and CSS on content and style**

HTML

Greetings, CSS; I'm glad you're here because I've been wanting to clear up some confusion about us.

Lots of people think that my tags tell the browsers how to *display* the content. It's just not true! I'm all about *structure*, not presentation.

Well, you can see how some people might get confused; after all, it's possible to use HTML without CSS and still get a decent-looking page.

Hey, I'm pretty powerful too. Having your content structured is much more important than having it look good. Style is so superficial; it's the structure of the content that matters.

Whoa, what an ego! Well, I guess I shouldn't expect anything else from you—you're just trying to make a fashion statement with all that style you keep talking about.

CSS

Really? What kind of confusion?

Heck yeah—I don't want people giving you credit for my work!

"Decent" might be overstating it a bit, don't you think? I mean, the way most browsers display straight HTML looks kinda crappy. People need to learn how powerful CSS is and how easily I can give their web pages great style.

Get real! Without me, web pages would be pretty damn boring. Not only that, but take away the ability to style pages and no one is going to take your pages seriously. Everything is going to look clumsy and unprofessional.

HTML

Right. In fact, we're totally different languages, which is good because I wouldn't want any of your style designers messing with my structure elements.

Yeah, that is obvious to me any time I look at CSS—talk about an alien language.

Millions of web writers would disagree with you. I've got a nice clean syntax that fits right in with the content.

Hey, ever heard of closing tags?

Just notice that no matter where you go, I've got you surrounded by <style> tags. Good luck escaping!

CSS

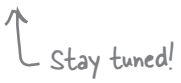
Fashion statement? Good design and layout can have a huge effect on how readable and usable pages are. And you should be happy that my flexible style rules allow designers to do all kinds of interesting things with your elements without messing up your structure.

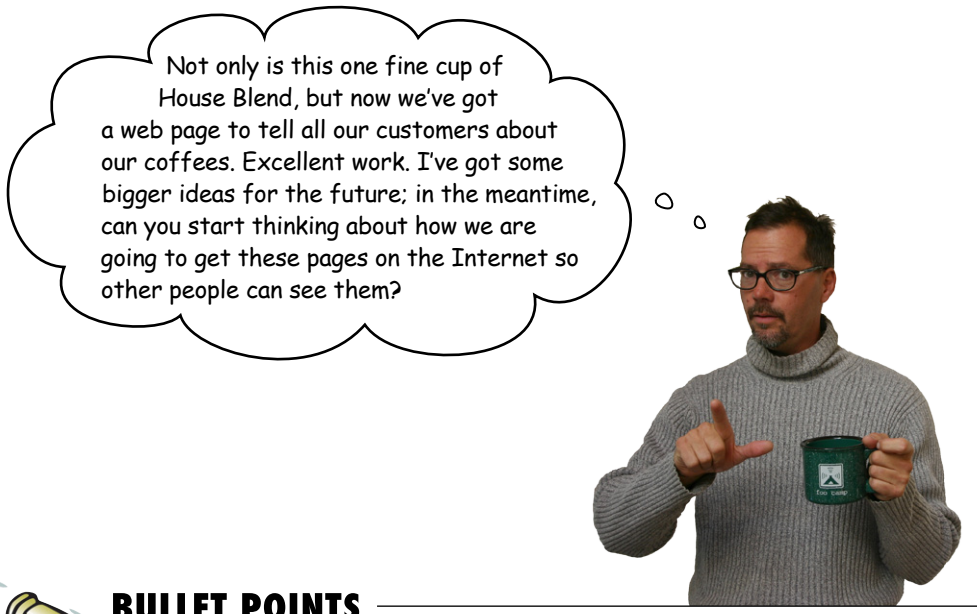
Don't worry, we're living in separate universes.

Yeah, like HTML can be called a language? Who has ever seen such a clunky thing with all those tags?

Just take a look at CSS; it's so elegant and simple, no goofy angle brackets <around> <everything>. <See> <I> <can><talk> <just><like><Mr.><HTML><,><look><at> <me><!>

Ha! I'll show you…because, guess what? I *can* escape…



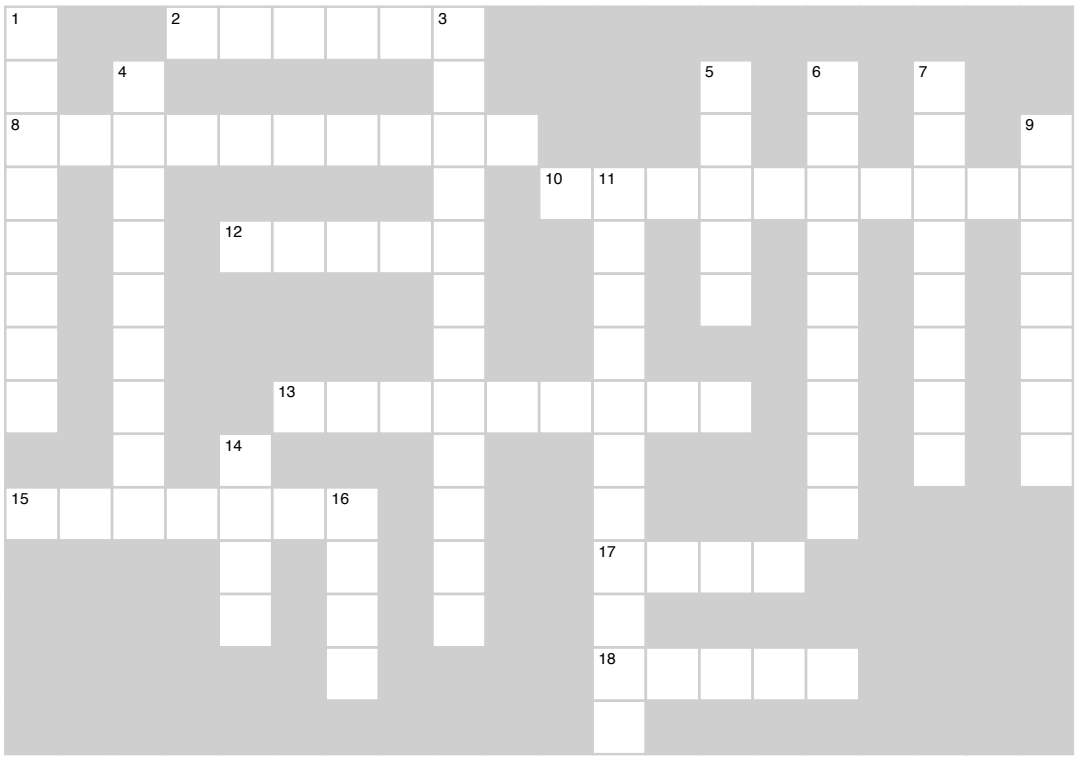


BULLET POINTS

- HTML and CSS are the languages we use to create web pages.
- Web servers store and serve web pages, which are created from HTML and CSS. Browsers retrieve pages and render their content based on the HTML and CSS.
- HTML is an abbreviation for HyperText Markup Language and is used to structure your web page.
- CSS is an abbreviation for Cascading Style Sheets, and is used to control the presentation of your HTML.
- Using HTML, we mark up content with tags to provide structure. We call matching tags, and their enclosed content, elements.
- An element is composed of three parts: an opening tag, content, and a closing tag. There are a few elements, like <img>, that are an exception to this rule.
- Opening tags can have attributes. We've seen one already: type.
- Closing tags have a "/" after the left angle bracket, in front of the tag name, to distinguish them as closing tags.
- Your pages should always have an <html> element along with a <head> element and a <body> element.
- Information about the web page goes into the <head> element.
- What you put into the <body> element is what you see in the browser.
- Most whitespace (tabs, returns, spaces) is ignored by the browser, but you can use it to make your HTML more readable (to you).
- You can add CSS to an HTML web page by putting the CSS rules inside the <style> element. The <style> element should always be inside the <head> element.
- You specify the style characteristics of the elements in your HTML using CSS.

HTMLcross

It's time to sit back and give your left brain something to do. It's your standard crossword; all of the solution words are from this chapter.



Across

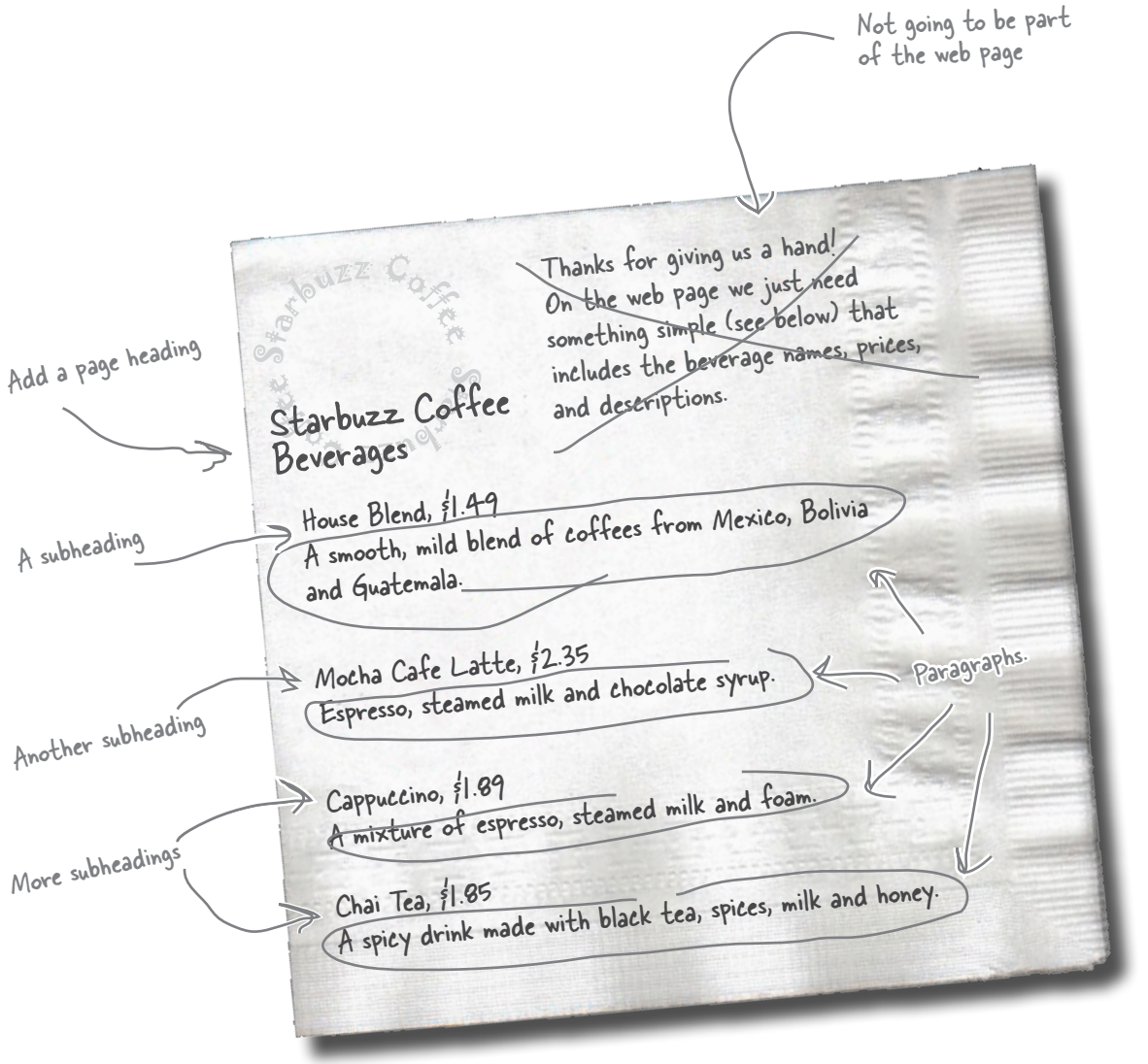
2. The "M" in HTML.
8. Tags can have these to provide additional information.
10. Browsers ignore this.
12. You define presentation through this element.
13. Purpose of <p> element.
15. Two tags and content.
17. What you see in your page.
18. We emphasized Dance _____ Revolution.

Down

1. There are six of these.
3. CSS is used when you need to control this.
4. You mark up content to provide this.
5. Appears at the top of the browser for each page.
6. Style we wish we could have had.
7. Company that launched your web career.
9. Only type of style available.
11. Always separate these in HTML.
14. About your web page.
16. Opening and closing.

Sharpen your pencil Solution

Go ahead and mark up the napkin (using your pencil) with any structure you see, and add anything that is missing.



Markup Magnets Solution

Your job was to add some structure to the text from the Starbuzz napkin. Use the fridge magnets at the bottom of the page to mark up the text so that you've indicated which parts are headings, subheadings, and paragraph text. Here's our solution.

```
<h1> Starbuzz Coffee Beverages </h1>

<h2> House Blend, $1.49 </h2>
<p> A smooth, mild blend of coffees from Mexico, Bolivia and
Guatemala. </p>

<h2> Mocha Cafe Latte, $2.35 </h2>
<p> Espresso, steamed milk and chocolate syrup. </p>

<h2> Cappuccino, $1.89 </h2>
<p> A mixture of espresso, steamed milk and foam. </p>

<h2> Chai Tea, $1.85 </h2>
<p> A spicy drink made with black tea, spices, milk and honey. </p>
```

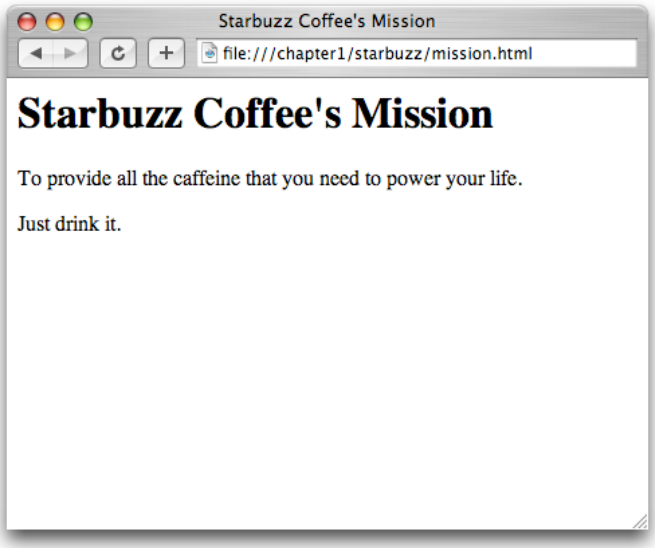
<h1>
<h1>
</h1>
</h1>

Leftover magnets

mission.html

```
<html>
  <head>
    <title>Starbuzz Coffee's Mission</title>
  </head>
  <body>
    <h1>Starbuzz Coffee's Mission</h1>
    <p>To provide all the caffeine that you need to
power your life.</p>
    <p>Just drink it.</p>
  </body>
</html>
```

Here's the HTML.



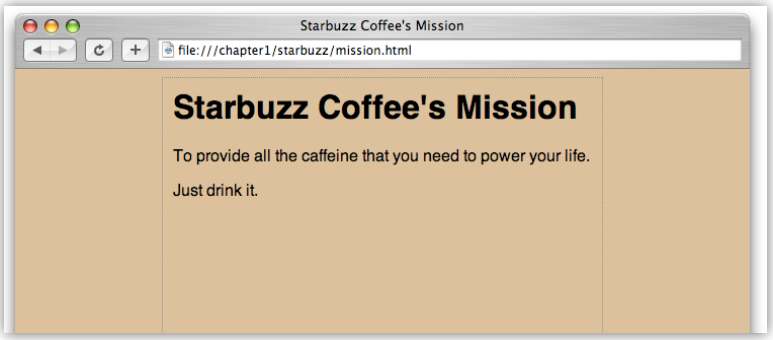
Here's the HTML displayed in a browser.

Here's the CSS in the mission page.

mission.html

```
<html>
  <head>
    <title>Starbuzz Coffee's Mission</title>
    <style type="text/css">
      body {
          background-color: #d2b48c;
          margin-left: 20%;
          margin-right: 20%;
          border: 2px dotted black;
          padding: 10px 10px 10px 10px;
          font-family: sans-serif;
      }
    </style>
  </head>
  <body>
    <h1>Starbuzz Coffee's Mission</h1>
    <p>To provide all the caffeine that you need to power
your life.</p>
    <p>Just drink it.</p>
  </body>
</html>
```

Now, the style matches the main Starbuzz page.

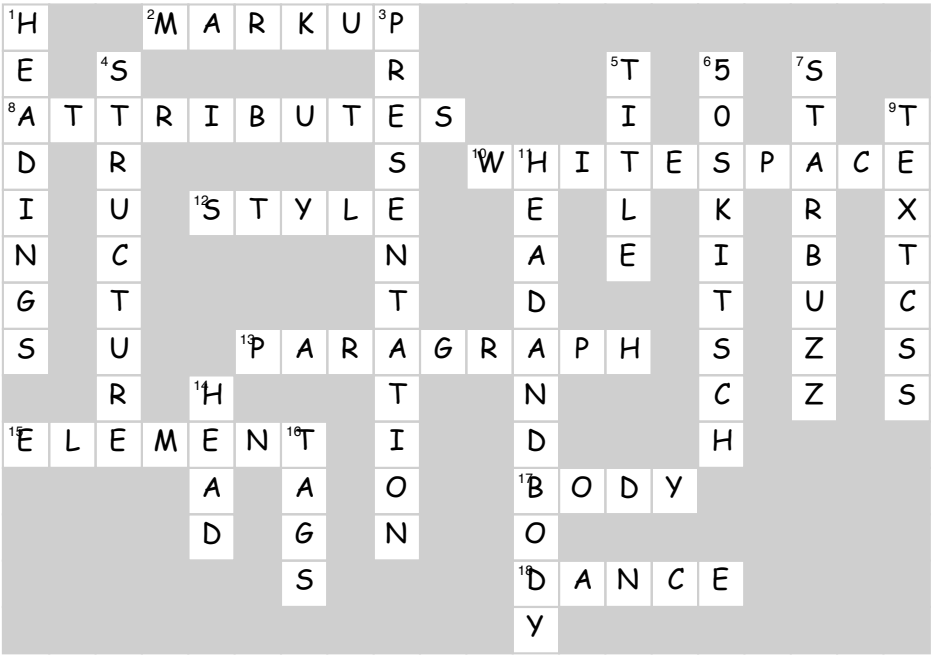


Who does what?

Even though you've just glanced at CSS, you've already seen the beginnings of what it can do. Match each line in the style definition to what it does.

```
background-color: #d2b48c;

margin-left: 20%;
margin-right: 20%;

border: 2px dotted black;

padding: 10px 10px 10px 10px;

font-family: sans-serif;
```

- Defines the font to use for text.
- Defines a border around the body that is dotted and the color black.
- Sets the left and right margins to take up 20% of the page each.
- Sets the background color to tan.
- Creates some padding around the body of the page.



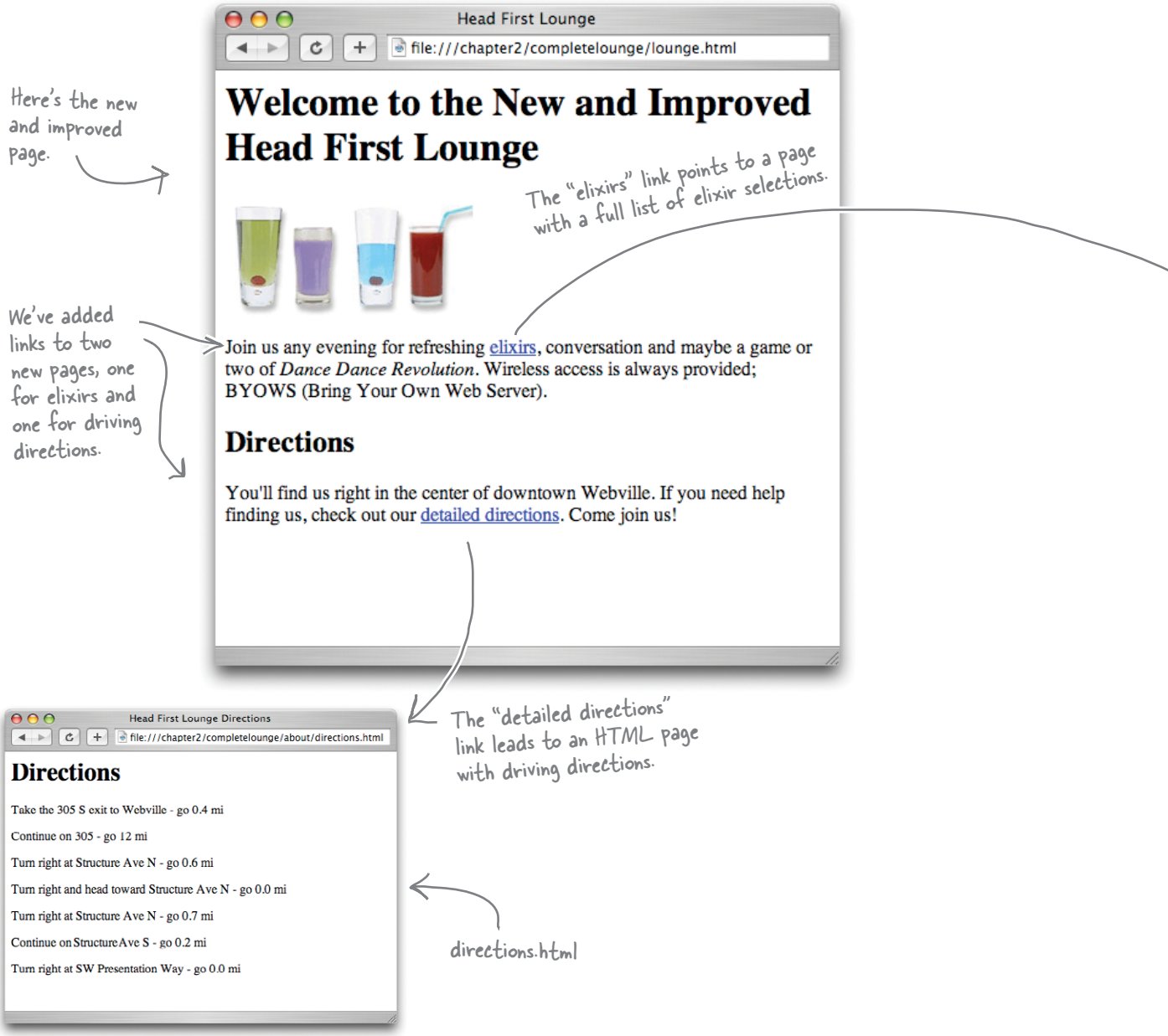
2 going further with hypertext

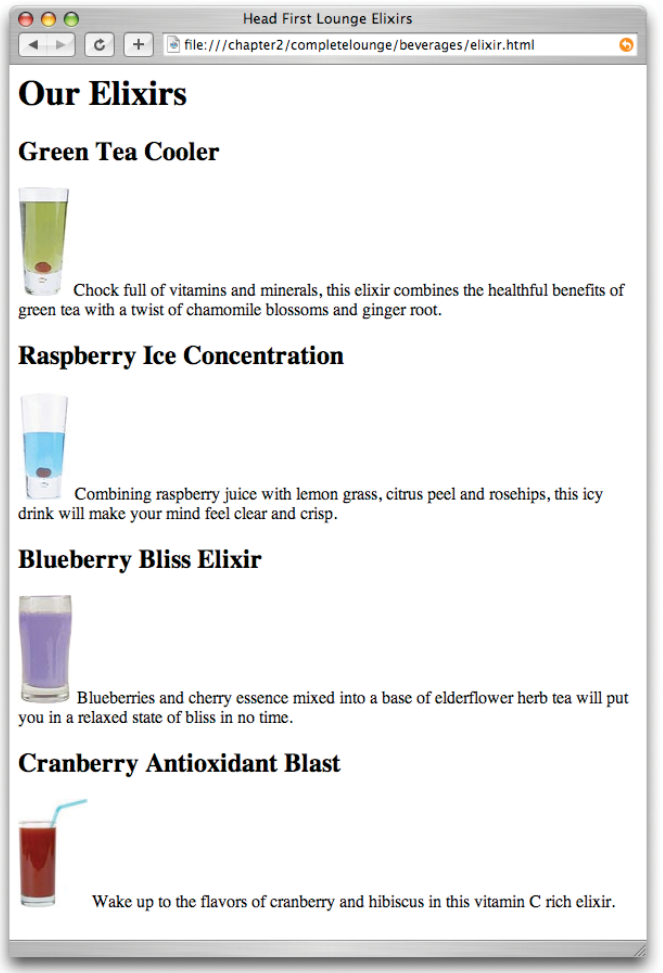
Meeting the "HT" in HTML

Did someone say "hypertext?" What's that? Oh, only the entire basis of the Web. In Chapter 1 we kicked the tires of HTML and found it to be a nice markup language (the "ML" in HTML) for describing the structure of web pages. Now we're going to check out the "HT" in HTML, hypertext, which will let us break free of a single page and link to other pages. Along the way we're going to meet a powerful new element, the <a> element, and learn how being "relative" is a groovy thing. So, fasten your seat belts—you're about to learn some hypertext.

Head First Lounge, *new and improved*

Remember the Head First Lounge? Great site, but wouldn't it be nice if customers could view a list of the refreshing elixirs? Even better, we should give customers some real driving directions so they can find the place.





elixir.html

A page listing some refreshing and healthy drinks. Feel free to grab one before going on.

Creating the new and improved lounge in three steps...

Let's rework the original Head First Lounge page so it links to the two new pages.

1. The first step is easy because we've already created the "directions.html" and "elixir.html" files for you. You'll find them in the source files for the book, which are available at `http://wickedlysmart.com/hfhtmlcss`.

2. Next you're going to edit the "lounge.html" file and add in the HTML needed to link to "directions.html" and "elixir.html".

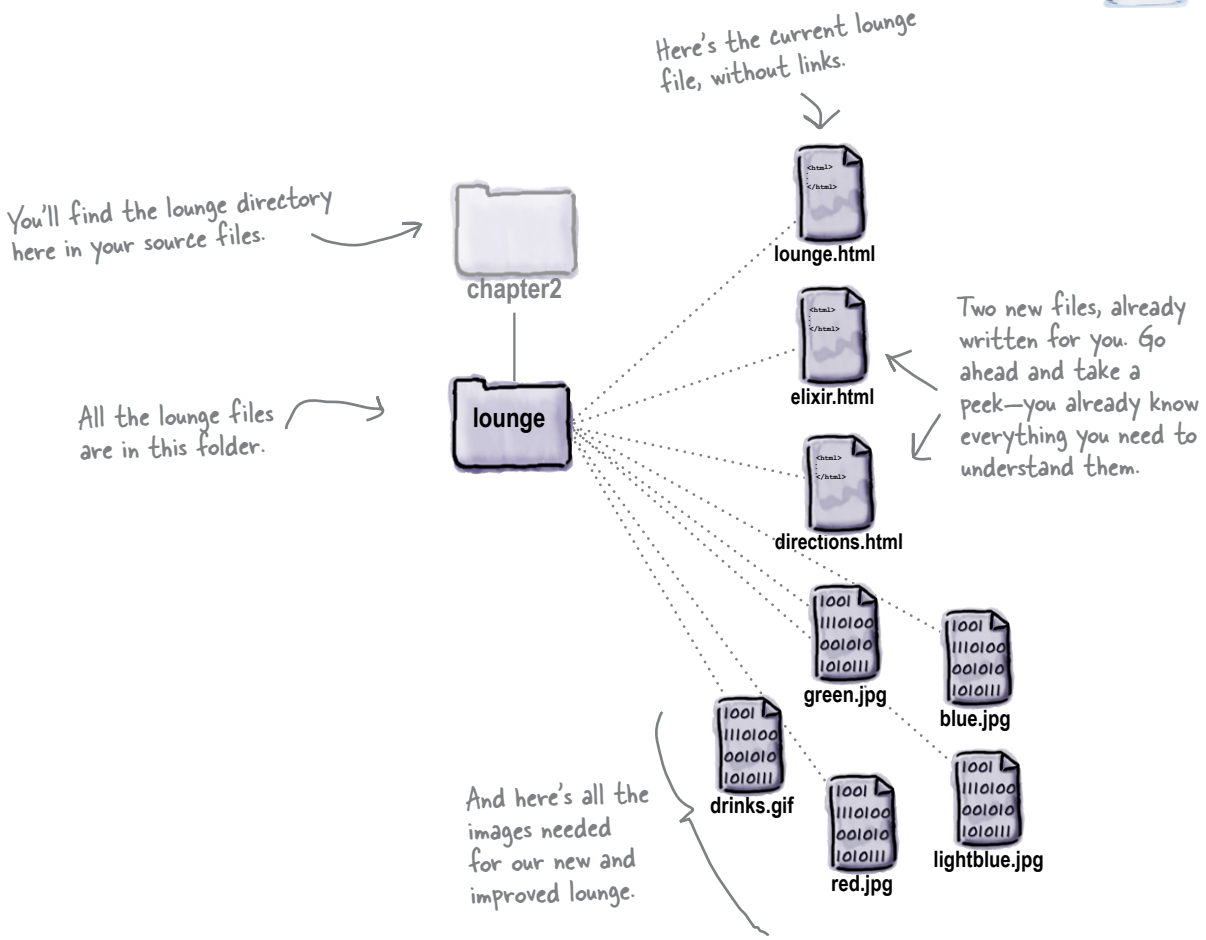
3. Last, you'll give the pages a test drive and try out your new links. When you get back, we'll sit down and look at how it all works.

Flip the page and let's get started…

Creating the new lounge

1 Grab the source files

Go ahead and grab the source files from `http://wickedlysmart.com/hfhtmlcss`. Once you've downloaded them, look under the folder "chapter2/lounge" and you'll find "lounge.html", "elixir.html", and "directions.html" (and a bunch of image files).



BRAIN POWER

The Head First Lounge is already growing; do you think that keeping all the site's files in a single directory is a good way to organize the site? What would you do differently?

2 Edit lounge.html

Open "lounge.html" in your editor. Add the new text and HTML that is highlighted below. Go ahead and type this in; we'll come back and see how it all works on the next page.

```
<html>
  <head>
    <title>Head First Lounge</title>
  </head>
  <body>
    <h1>Welcome to the New and Improved Head First Lounge</h1>
    <img src="drinks.gif">
    <p>
       Join us any evening for
       refreshing <a href="elixir.html">elixirs</a>,
       conversation and maybe a game or two of
       <em>Dance Dance Revolution</em>.
       Wireless access is always provided;
       BYOWS (Bring your own web server).
    </p>
    <h2>Directions</h2>
    <p>
      You'll find us right in the center of downtown Webville.
      If you need help finding us, check out
      our <a href="directions.html">detailed directions</a>.
      Come join us!
    </p>
  </body>
</html>
```

Let's add "New and Improved" to the heading.

Here's where we add the HTML for the link to the elixirs.

To create links, we use the <a> element; we'll take a look at how this element works in just a sec...

We need to add some text here to point customers to the new directions.

And here's where we add the link to the directions, again using an <a> element.

3 Save lounge.html and give it a test drive.

When you're finished with the changes, save the file "lounge.html" and open it in your browser. Here are a few things to try:

1. Click on the elixir link and the new elixir page will display.
2. Click on the browser's back button and "lounge.html" should be displayed again.
3. Click on the directions link and the new directions page will display.

Okay, I've loaded the new lounge page, clicked the links, and everything worked. But I want to make sure I understand how the HTML works.

Behind the Scenes

What did we do?

1. Let's step through creating the HTML links. First, we need to put the text we want for the link in an <a> element, like this:

```
<a>elixirs</a>
```

The <a> element is used to create a link to another page.

```
<a>driving directions</a>
```

The content of the <a> element is the link text. In the browser, the link text appears with an underline to indicate you can click on it.

2. Now that we have text for the link, we need to add some HTML to tell the browser where the link points to:

```
<a href="elixir.html">elixirs</a>
```

For this link, the browser will display the text "elixirs" that, when clicked, will take the user to the "elixir.html" page.

The href attribute is how you specify the destination of the link.

```
<a href="directions.html">driving directions</a>
```

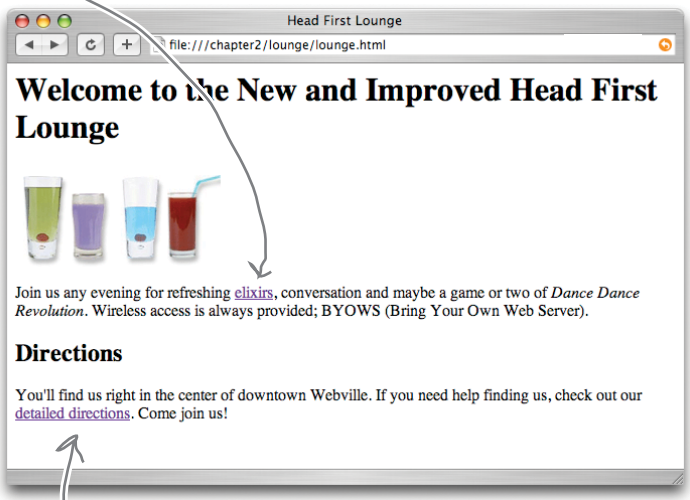
And for this link, the browser will display a "driving directions" link that, when clicked, will take the user to the "directions.html" page.

What does the browser do?

1. First, as the browser renders the page, if it encounters an <a> element, it takes the content of the element and displays it as a clickable link.

```
<a href="elixir.html">elixirs</a>
```

Both "elixirs" and "detailed directions" are between the opening and closing <a> tags, so they end up being clickable text in the web page.




```
<a href="directions.html">detailed directions</a>
```

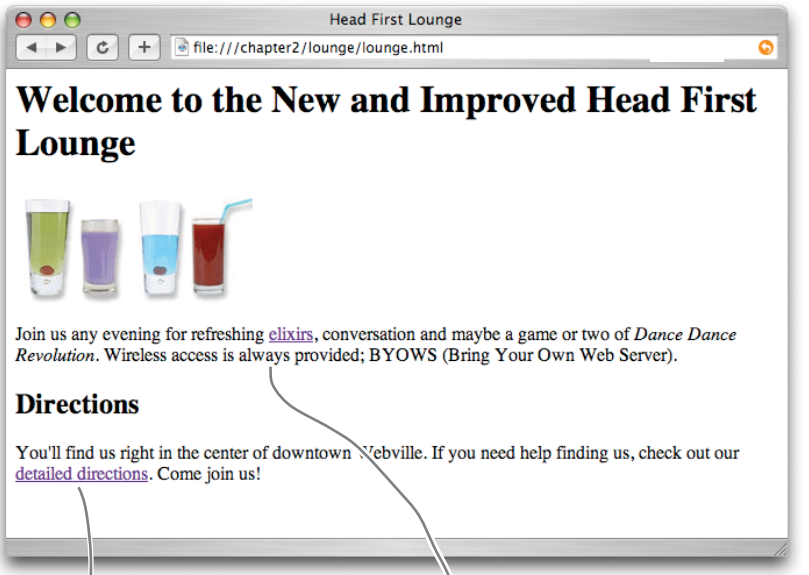
Use the <a> element to create a hypertext link to another web page. The content of the <a> element becomes clickable in the web page. The href attribute tells the browser the destination of the link.

Behind the Scenes

② Next, when a user clicks on a link, the browser uses the "href" attribute to determine the page the link points to.

The user clicks on either the "elixirs" link or...

...on "detailed directions".



When "detailed directions" is clicked, the browser grabs the value of the href attribute, in this case "directions.html"...

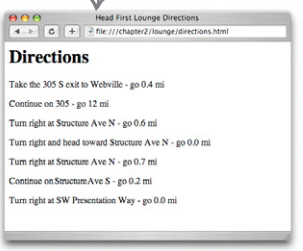
If "elixirs" was clicked, the browser grabs the href value "elixir.html"...

```
<a href="elixir.html">elixirs</a>
```

...and displays the "elixir.html" page.


```
<a href="directions.html">detailed directions</a>
```

...and loads "directions.html".



Understanding attributes

Attributes give you a way to specify additional information about an element. While we haven't looked at attributes in detail, you've already seen a few examples of them:

```
<style type="text/css">
<a href="irule.html">
<img src="sweetphoto.gif">
```

The type attribute specifies which style language we're using, in this case CSS.

The href attribute tells us the destination of a hyperlink.

The src attribute specifies the filename of the picture an img tag displays.

Let's cook up an example to give you an even better feel for how attributes work:

What if <car> were an element?

If <car> were an element, then you'd naturally want to write some markup like this:

```
<car>My Red Mini</car>
```

With no attributes, all we can supply is a descriptive name for the car.

But this <car> element only gives a descriptive name for your car—it doesn't tell us the make, precise model, whether it is a convertible, or a zillion other details we might want to know. So, if <car> were really an element, we might use attributes like this:

But with attributes, we can customize the element with all kinds of information.

```
<car make="Mini" model="Cooper" convertible="no">My Red Mini</car>
```

Better, right? Now this markup tells us a lot more information in an easy-to-write, convenient form.

SAFETY FIRST

Attributes are always written the same way: first comes the attribute name, followed by an equals sign, and then the attribute value surrounded in double quotes.

You may see some sloppy HTML on the Web that leaves off the double quotes, but don't get lazy yourself. Being sloppy can cause you a lot of problems down the road (as we'll see later in the book).

Do this (best practice)

```
<a href="top10.html">Great Movies</a>
```

attribute name

equals sign

double quote

attribute value

double quote

Not this

```
<a href=top10.html>Great Movies</a>
```

No double quotes around the attribute value

there are no Dumb Questions

Q: Can I just make up new attributes for an HTML element?

A: Web browsers only know about a predefined set of attributes for each element. If you just made up attributes, then browsers wouldn't know what to do with them, and as you'll see later in the book, doing this will very likely get you into trouble. When a browser recognizes an element or an attribute, we like to say that it "supports" that element or attribute. You should only use attributes that you know are supported.

That said, for programming web applications (the subject of *Head First HTML5 Programming*), HTML5 now supports custom data attributes that allow you to make up custom names for new attributes.

Q: Who decides what is "supported"?

A: There are standards committees that worry about the elements and attributes of HTML. These committees are made up of people ~~with nothing better to do~~ who generously give their time and energy to make sure there's a common HTML roadmap that all organizations can use to implement their browsers.

Q: How do I know what attributes and elements are supported? Or can all attributes be applied to any element?

A: Only certain attributes can be used with a given element. Think about it this way: you wouldn't use an attribute "convertible" with the element <toaster>, would you? So, you only want to use attributes that make sense and are supported by the element.

You're going to be learning which attributes are supported by which elements as you make your way through the book. After you've finished the book, there are lots of great references you can use to refresh your memory, such as *HTML & XHTML: The Definitive Guide* (O'Reilly).

Attributes Exposed

This week's interview:
Confessions of the href attribute

Head First: Welcome, href. It's certainly a pleasure to interview as big an attribute as you.

href: Thanks. It's good to be here and get away from all the linking; it can wear an attribute out. Every time someone clicks on a link, guess who gets to tell the browser where to go next? That would be me.

Head First: We're glad you could work us into your busy schedule. Why don't you take us back to the beginning…What does it mean to be an attribute?

href: Sure. Well, attributes are used to customize an element. It's easy to wrap some `<a>` tags around a piece of content, like "Sign up now!"—we do it like this: `<a>Sign up now!</a>`—but without me, the href attribute, you have no way to tell the `<a>` element the destination of the link.

Head First: Got it so far…

href: …but with an attribute you can provide additional information about the element. In my case, that's where the link points to: `<a href="signup.html">Sign up now!</a>`. This says that the `<a>` element, which is labeled "Sign up now!", links to the "signup.html" page. Now, there are lots of other attributes in the world, but I'm the one you use with the `<a>` element to tell it where it points to.

Head First: Nice. Now, I have to ask, and I hope you aren't offended, but what is with the name? href? What's with that?

href: It's an old Internet family name. It means "hypertext reference," but all my friends just call me "href" for short.

Head First: Which is?

href: A hypertext reference is just another name for a resource that is on the Internet or your computer. Usually the resource is a web page, but I can also point to PDF documents…all kinds of things.

Head First: Interesting. All our readers have seen so far are links to their own pages; how do we link to other pages and resources on the Web?

href: Hey, I gotta get back to work, the whole Web is getting gunked up without me. Besides, isn't it your job to teach them this stuff?

Head First: Okay, okay, yes, we're getting to that in a bit…thanks for joining us, href.

You've created links to go from "lounge.html" to "elixir.html" and "directions.html"; now we're going to go back the other way. Below you'll find the HTML for "elixir.html". Add a link with the label "Back to the Lounge" at the bottom of the elixir page that points back to "lounge.html".

```
<html>
  <head>
    <title>Head First Lounge Elixirs</title>
  </head>
  <body>
    <h1>Our Elixirs</h1>

    <h2>Green Tea Cooler</h2>
    <p>
       <img src="green.jpg">
       Chock full of vitamins and minerals, this elixir
       combines the healthful benefits of green tea with
       a twist of chamomile blossoms and ginger root.
    </p>
    <h2>Raspberry Ice Concentration</h2>
    <p>
        <img src="lightblue.jpg">
        Combining raspberry juice with lemon grass,
        citrus peel and rosehips, this icy drink
        will make your mind feel clear and crisp.
    </p>
    <h2>Blueberry Bliss Elixir</h2>
    <p>
        <img src="blue.jpg">
        Blueberries and cherry essence mixed into a base
        of elderflower herb tea will put you in a relaxed
        state of bliss in no time.
    </p>
    <h2>Cranberry Antioxidant Blast</h2>
    <p>
        <img src="red.jpg">
        Wake up to the flavors of cranberry and hibiscus
        in this vitamin C rich elixir.
    </p>

Your new HTML goes here. ↪

  </body>
</html>
```

When you are done, go ahead and do the same with "directions.html" as well.

We need some help constructing and deconstructing <a> elements. Given your new knowledge of the <a> element, we're hoping you can help. In each row below, you'll find some combination of the label, destination, and the complete <a> element. Fill in any information that is missing. The first row is done for you.

Label	Destination	What you write in HTML
Hot or Not?	hot.html	`<a href="hot.html">Hot or Not?</a>`
Resume	cv.html	
	candy.html	`<a href="...">Eye Candy</a>`
See my mini	mini-cooper.html	
let's play		`<a href="millionaire.html">...</a>`

there are no Dumb Questions

Q: I've seen many pages where I can click on an image rather than text. Can I use the <a> element for that?

A: Yes, if you put an <img> element between the <a> tags, then your image will be clickable just like text. We're not going to talk about images in depth for a few chapters, but they work just fine as links.

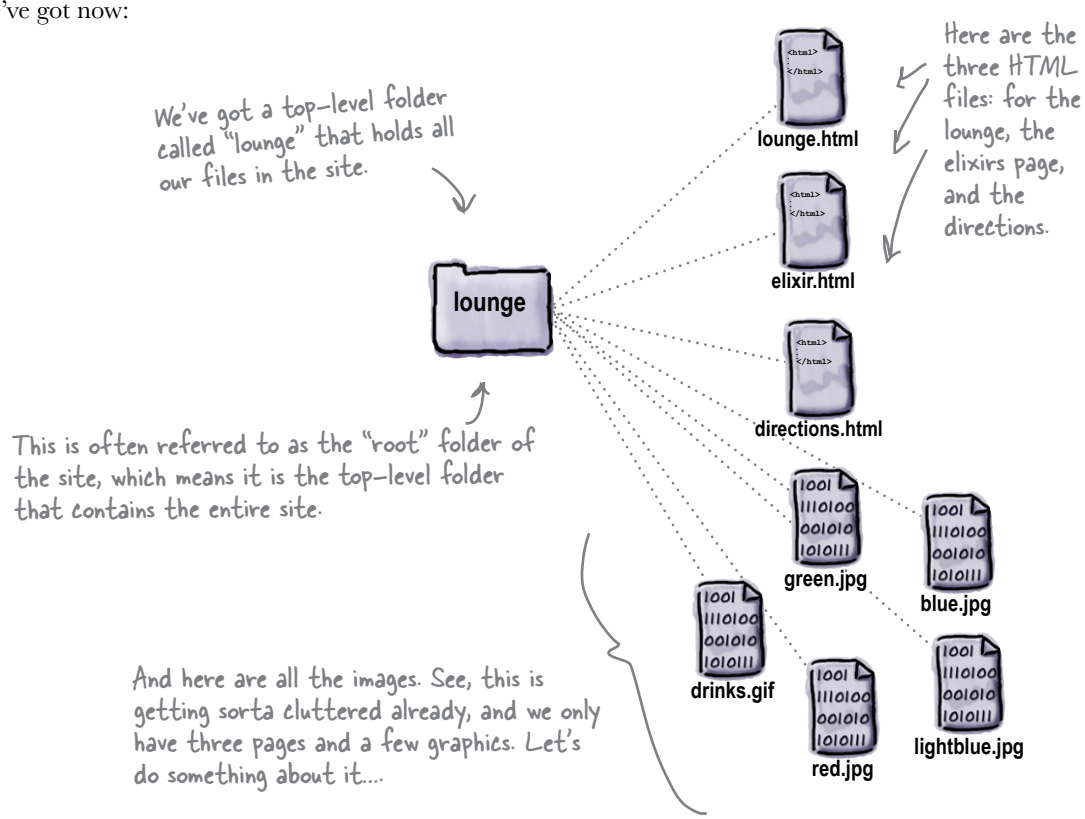
Q: So I can put anything between the <a> tags and it will be clickable? Like, say, a paragraph?

A: You can indeed put a <p> element inside an <a> element to link an entire paragraph. You'll mostly be using text and images (or both) within the <a> element, but if you need to link a <p> or a <h1> element, you can. What tags will go inside other tags is a whole other topic, but don't worry; we'll get there soon enough.

Your work on the Head First Lounge has really paid off. With those enticing elixirs and directions, lots of people are frequenting the place and visiting the website. Now we've got plans for expanding the lounge's online content in all sorts of directions.

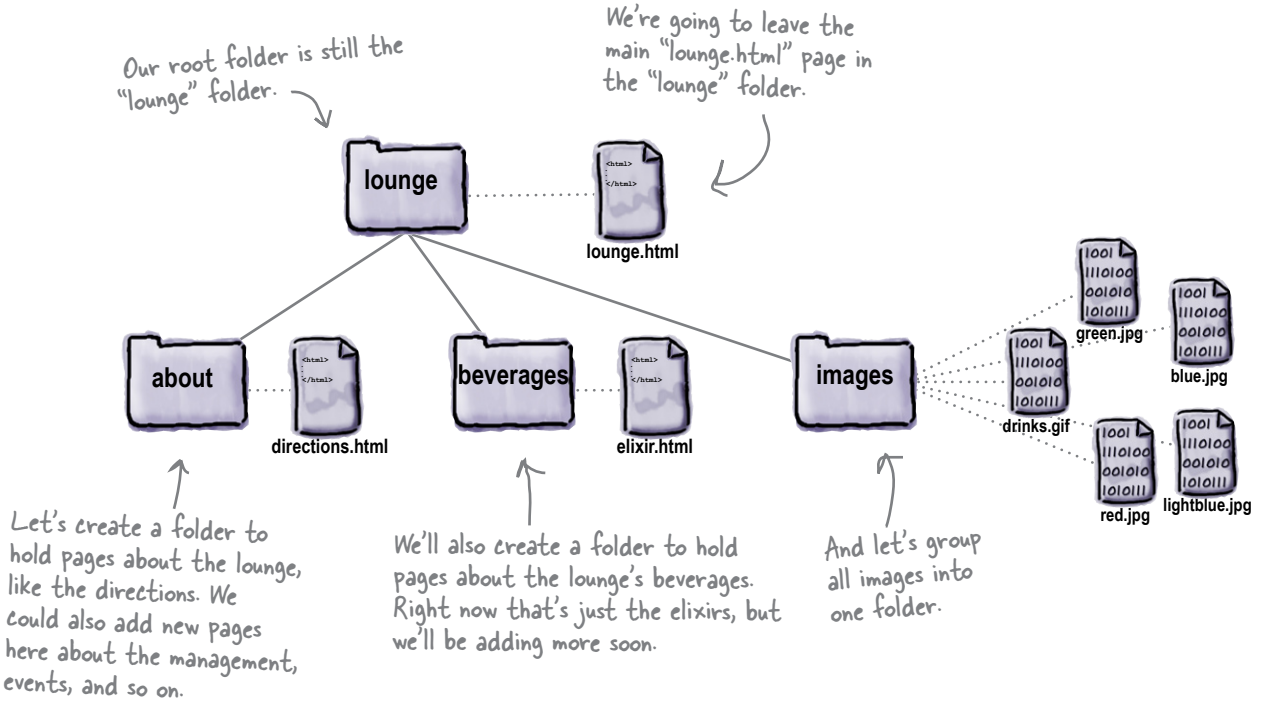
Getting organized

Before you start creating more HTML pages, it's time to get things organized. So far, we've been putting all our files and images in one folder. You'll find that even for modestly sized websites, things are much more manageable if you organize your web pages, graphics, and other resources into a set of folders. Here's what we've got now:



Organizing the lounge...

Let's give the lounge site some meaningful organization now. Keep in mind there are lots of ways to organize any site; we're going to start simple and create a couple of folders for pages. We'll also group all those images into one place.



there are no Dumb Questions

Q: Since you have a folder for images, why not have another one called "html" and put all the HTML in that folder?

A: You could. There aren't any "correct" ways to organize your files; rather, you want to organize them in a way that works best for you and your users. As with most design decisions, you want to choose an organization scheme that is flexible enough to grow, while keeping things as simple as you can.

Q: Or why not put an images folder in each other folder, like "about" and "beverages"?

A: Again, we could have. We expect that some of the images will be reused among several pages, so we put images in a folder at the root (the top level) to keep them all together. If you have a site that needs lots of images in different parts of the site, you might want each branch to have its own image folder.

Q: "Each branch"?

A: You can understand the way folders are described by looking at them as upside down trees. At the top is the root and each path down to a file or folder is a branch.

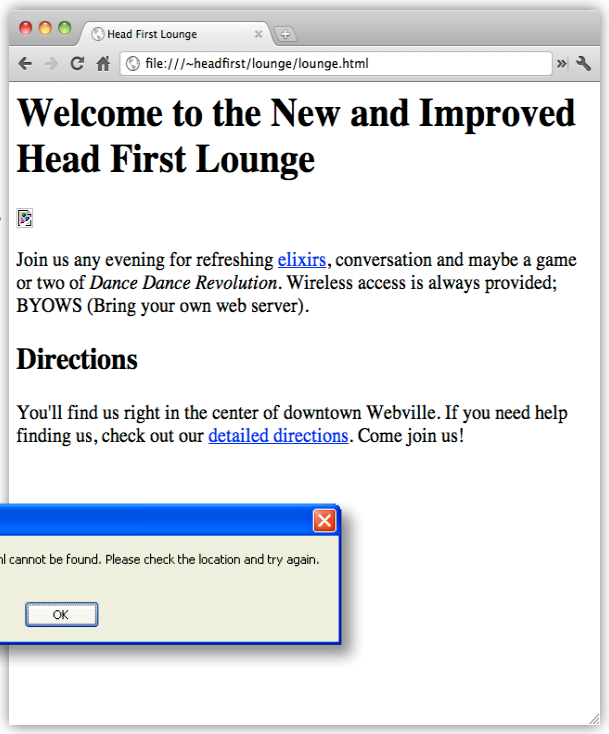
Now you need to create the file and folder structure shown on the previous page. Here's exactly what you need to do:

1. Locate your "lounge" folder and create three new subfolders named "about", "beverages", and "images".
2. Move the file "directions.html" into the "about" folder.
3. Move the file "elixir.html" into the "beverages" folder.
4. Move all the images into the "images" folder.
5. Finally, load your "lounge.html" file and try out the links. Compare with how ours worked below.

Technical difficulties

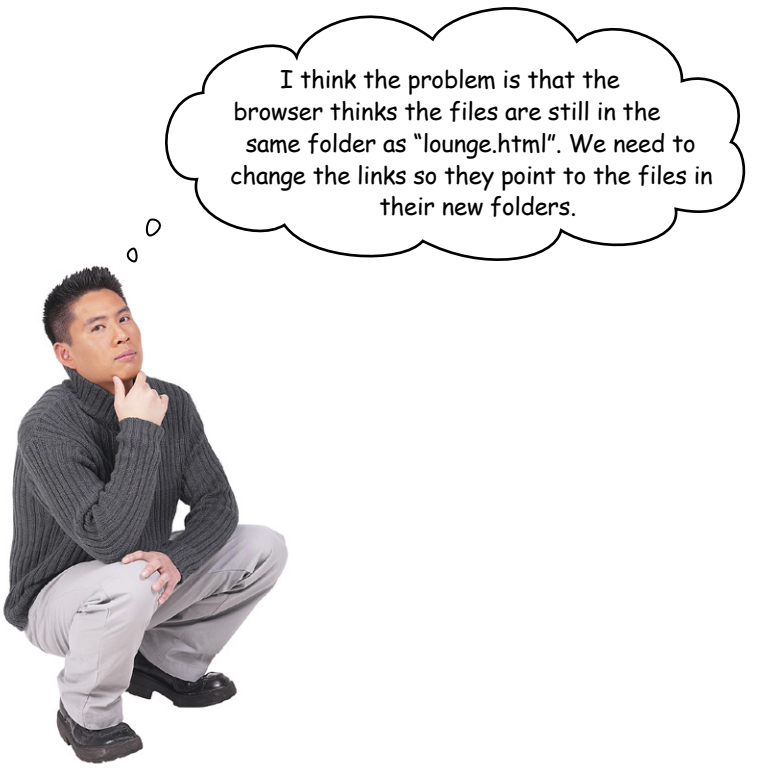
It looks like we've got a few problems with the lounge page after moving things around.

We've got an image that isn't displaying. We usually call this a "broken image."



And, when you click on "elixirs" (or "detailed directions") things get much worse: we get an error saying the page can't be found.

Some browsers display this error as a web page rather than a dialog box.



Right. We need to tell the browser the new location of the pages.

So far you've used `href` values that point to pages in the *same folder*. Sites are usually a little more complicated, though, and you need to be able to point to pages that are in *other folders*.

To do that, you trace the path from your page to the destination file. That might mean going down a folder or two, or up a folder or two, but either way we end up with a *relative path* that we can put in the `href`.

Planning your paths...

What do you do when you're planning that vacation in the family truckster? You get out a map and start at your current location, and then trace a path to the destination. The directions themselves are *relative* to your location—if you were in another city, they'd be different directions, right?

Okay, you'd really go to Google Maps, but work with us here!

To figure out a relative path for your links, it's the same deal: you start from the page that has the link, and then you trace a path through your folders until you find the file you need to point to.

There are other kinds of paths too. We'll get to those in later chapters.

Let's work through a couple of relative paths (and fix the lounge at the same time).

Linking down into a subfolder

1. Linking from "lounge.html" to "elixir.html".

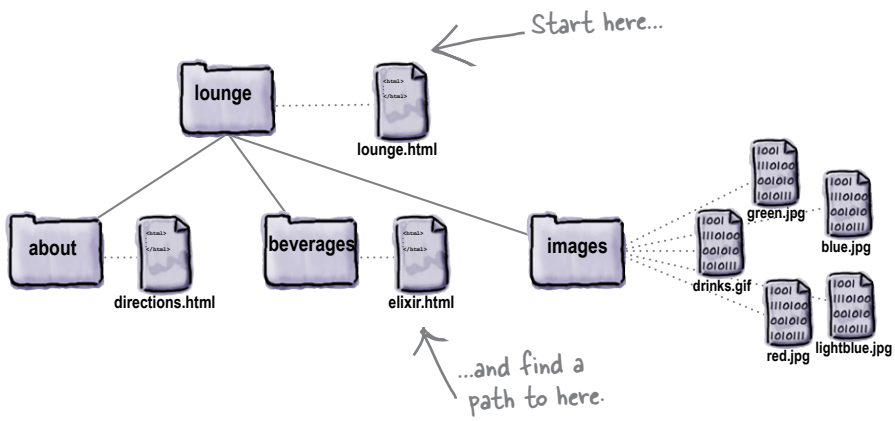
We need to fix the "elixirs" link in the "lounge.html" page. Here's what the `<a>` element looks like now:

```
<a href="elixir.html">elixirs</a>
```

Right now we're just using the filename "elixir.html", which tells the browser to look in the same folder as "lounge.html".

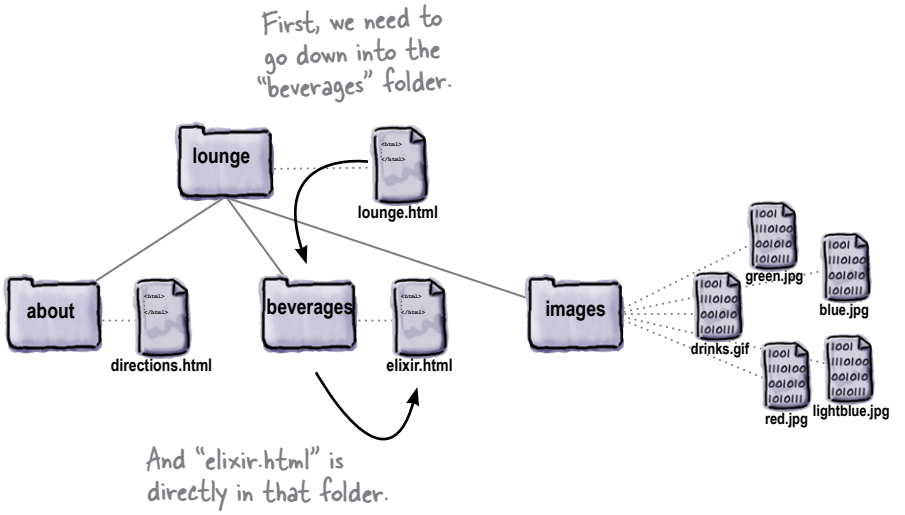
2. Identify the source and the destination.

When we reorganized the lounge, we left "lounge.html" in the "lounge" folder, and we put "elixir.html" in the "beverages" folder, which is a subfolder of "lounge".



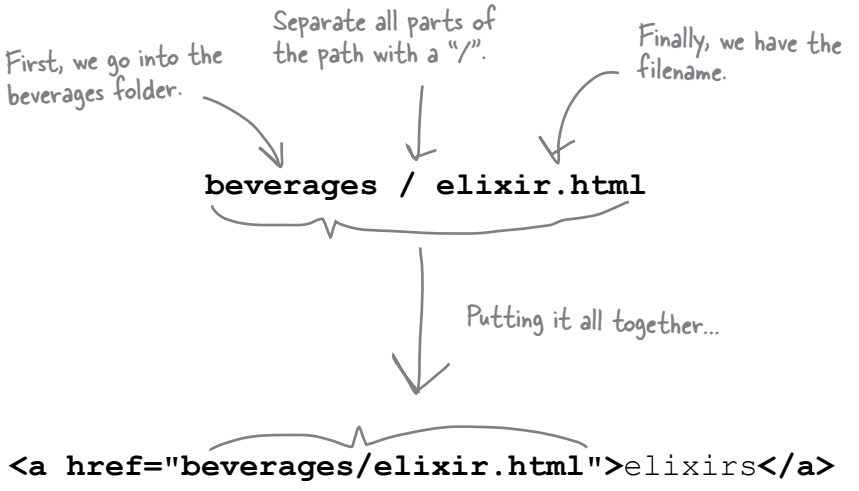
③ Trace a path from the source to the destination.

Let's trace the path. To get from the "lounge.html" file to "elixir.html", we need to go into the "beverages" folder first, and then we'll find "elixir.html" in that folder.



④ Create an href to represent the path we traced.

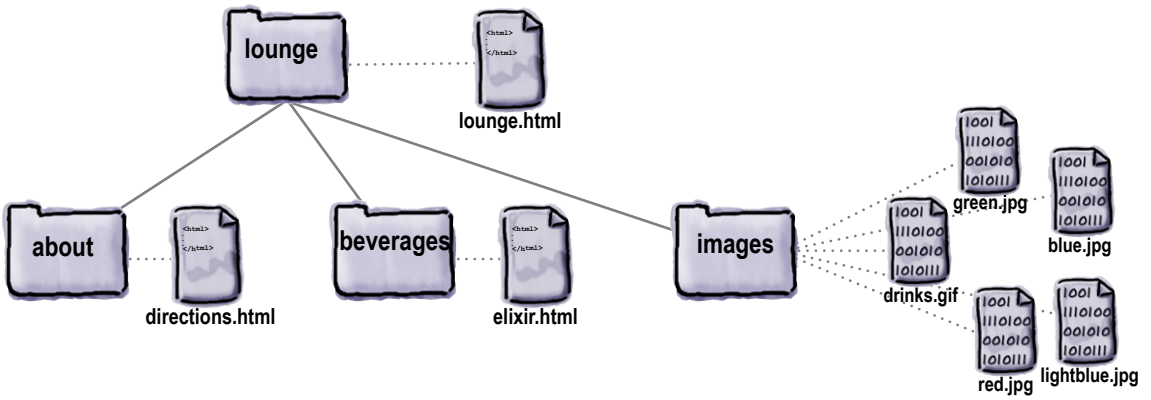
Now that we know the path, we need to get it into a format the browser understands. Here's how you write the path:



We put the relative path into the href value. Now when the link is clicked, the browser will look for the "elixir.html" file in the "beverages" folder.

Sharpen your pencil

Your turn: trace the relative path from "lounge.html" to "directions.html". When you've discovered it, complete the `<a>` element below. Check your answer in the back of the chapter, and then go ahead and change both `<a>` elements in "lounge.html."




```
<a href="                                    ">detailed directions</a>
```

YOUR ANSWER HERE

Going the other way; linking up into a "parent" folder

1 Linking from "directions.html" to "lounge.html".

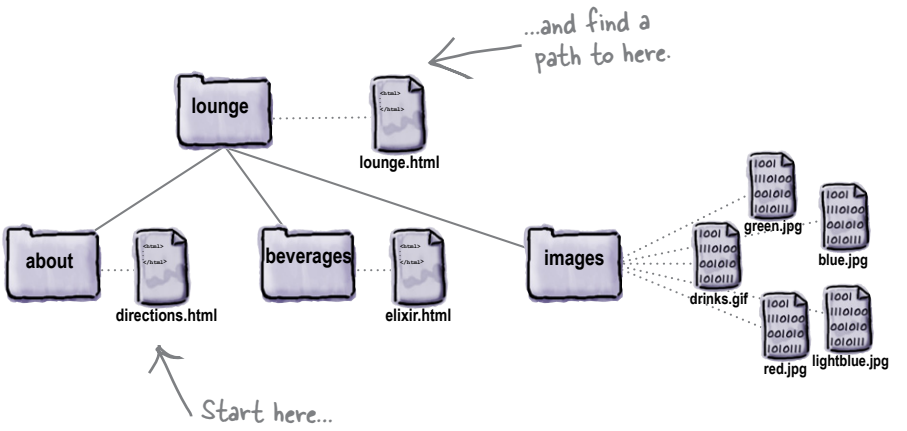
Now we need to fix those "Back to the Lounge" links. Here's what the <a> element looks like in the "directions.html" file:

```
<a href="lounge.html">Back to the Lounge</a>
```

Right now, we're just using the filename "lounge.html", which tells the browser to look in the same folder as "directions.html". That's not going to work.

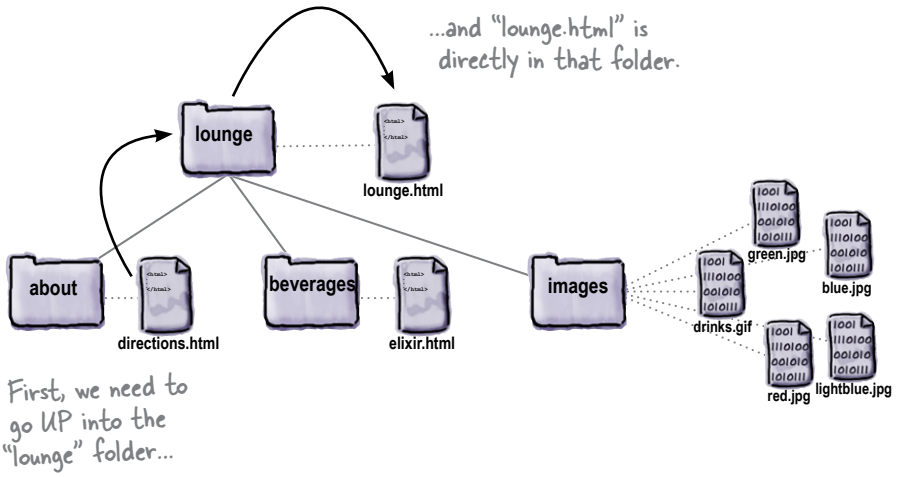
2 Identify the source and the destination.

Let's take a look at the source and destination. The source is now the "directions.html" file, which is down in the "about" folder. The destination is the "lounge.html" file that sits above the "about" folder, where "directions.html" is located.



3 Trace a path from the source to the destination.

Let's trace the path. To get from the "directions.html" file to "lounge.html", we need to go up one folder into the "lounge" folder, and then we'll find "lounge.html" in that folder.



4 Create an href to represent the path we traced.

We're almost there. Now that you know the path, you need to get it into a format the browser understands. Let's work through this:

First, you need to go up one folder. How do you do that? With a ".." . That's right, two periods. Go with it, we'll explain in a sec.

Separate all parts of the path with a "/".

Finally, you have the filename.

```
.. / lounge.html
```

Pronounce ".." as "dot dot".

Putting it all together...

```
<a href="../lounge.html">Back to the Lounge</a>
```

Now when you click on the link, the browser will look for the "lounge.html" file in the folder above.

there are no Dumb Questions

Q: What's a parent folder? If I have a folder "apples" inside a folder "fruit", is "fruit" the parent of "apples"?

A: Exactly. Folders (you might have heard these called directories) are often described in terms of family relationships. For instance, using your example, "fruit" is the parent of "apples", and "apples" is the child of "fruit". If you had another folder "pears" that was a child of "fruit", it would be a sibling of "apples." Just think of a family tree.

Q: Okay, parent makes sense, but what is ".."?

A: When you need to tell the browser that the file you're linking to is in the parent folder, you use ".." to mean "move UP to the parent folder." In other words, it's browser-speak for parent.

In our example, we wanted to link from "directions.html", which is in the "about" folder, to "lounge.html", which is in the "lounge" folder, the parent of "about". So we had to tell the browser to look UP one folder, and ".." is the way we tell the browser to go UP.

Q: What do you do if you need to go up two folders instead of just one?

A: You can use ".." for each parent folder you want to go up. Each time you use ".." you're going up by one parent folder. So, if you want to go up two folders, you'd type "../..". You still have to separate each part with the "/", so don't forget to do that (the browser won't know what "...." means!).

Q: Once I'm up two folders, how do I tell the browser where to find the file?

A: You combine the "../.." with the filename. So, if you're linking to a file called "fruit.html" in a folder that's two folders up, you'd write "../../fruit.html". You might expect that we'd call "../.." the "grandparent" folder, but we don't usually talk about them that way, and instead say, "the parent of the parent folder," or "../.." for short.

Q: Is there a limit to how far up I can go?

A: You can go up until you're at the root of your website. In our example, the root was the "lounge" folder. So, you could only go up as far as "lounge".

Q: What about in the other direction—is there a limit to how many folders I can go down?

A: Well, you can only go down as many folders as you have created. If you create folders that are 10 deep, then you can write a path that takes you down 10 folders. But we don't recommend that—when you have that many folder levels, it probably means your website organization is too complicated!

In addition, some browsers impose a limit on the number of characters you can have in a path. The spec advises caution above 255 characters, although modern browsers support longer lengths. If you have a large site, however, it's something to be aware of.

Q: My operating system uses "\" as a separator; shouldn't I be using that instead of "/"?

A: No; in web pages you always use "/" (forward slash). Don't use "\" (backslash). Various operating systems use different file separators (for instance, Windows uses "\" instead of "/") but when it comes to the Web, we pick a common separator and all stick to it. So, whether you're using Mac, Windows, Linux, or something else, always use "/" in the paths in your HTML.

Your turn: trace the relative path from "elixir.html" to "lounge.html" from the "Back to the Lounge" link. How does it differ from the same link in the "directions.html" file?

Answer: It doesn't; it is exactly the same.

Fixing those broken images...

You've almost got the lounge back in working order; all you need to do now is fix those images that aren't displaying.

We haven't looked at the `<img>` element in detail yet (we will in a couple of chapters), but all you need to know for now is that the `<img>` element's `src` attribute takes a relative path, just like the `href` attribute.

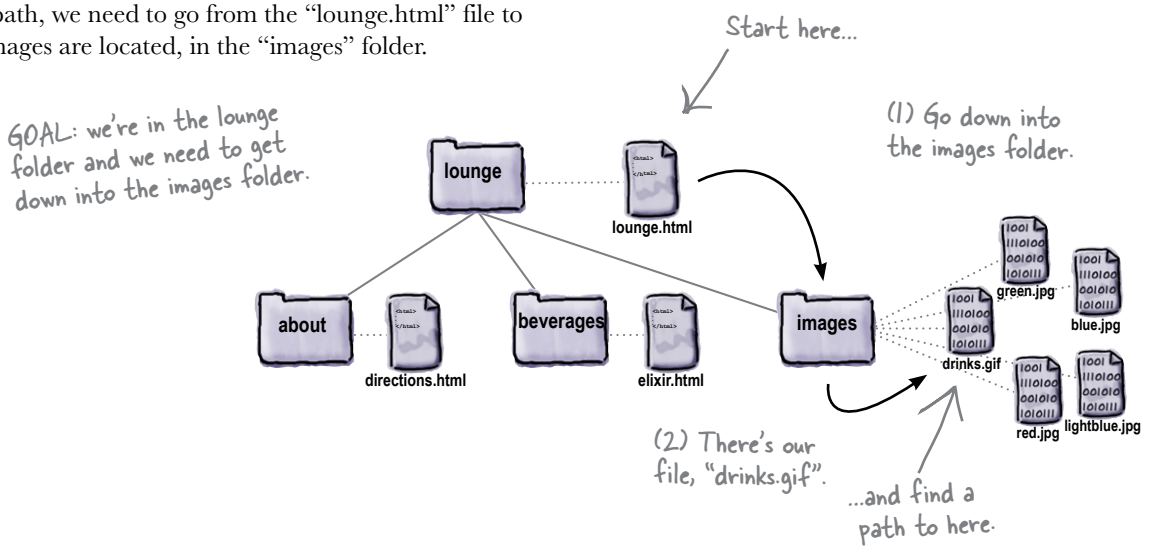
Here's the image element from the "lounge.html" file:

```
<img src="drinks.gif">
```

Here's the relative path, which tells the browser where the image is located. We specify this just like we do with the href attribute in the <a> element.

Finding the path from "lounge.html" to "drinks.gif"

To find the path, we need to go from the "lounge.html" file to where the images are located, in the "images" folder.

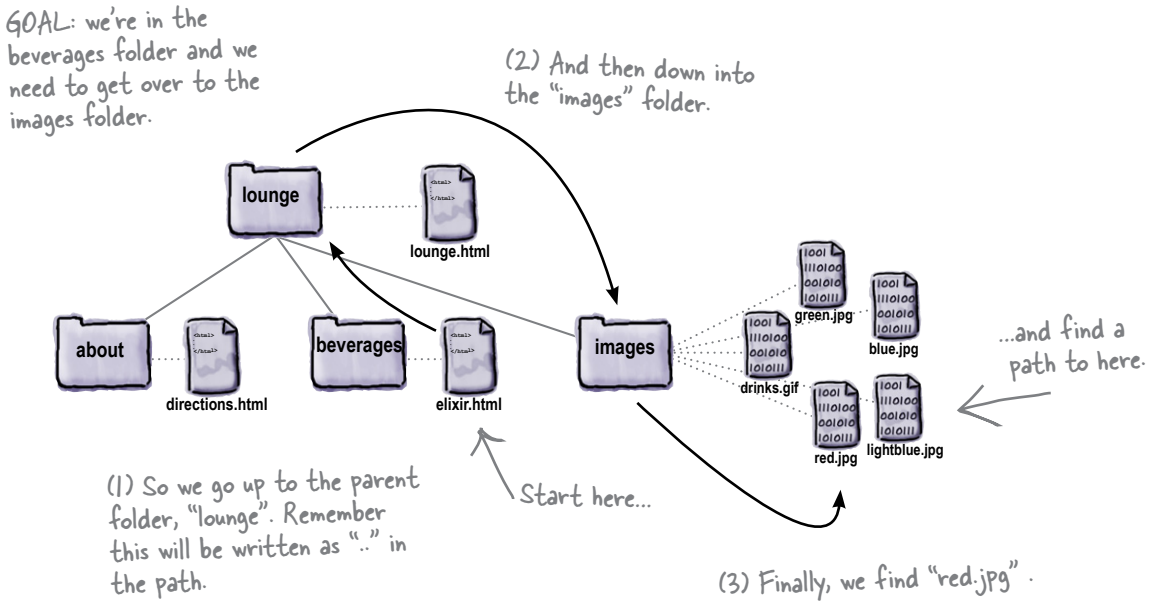


So when we put (1) and (2) together, our path looks like "images/drinks.gif", or:

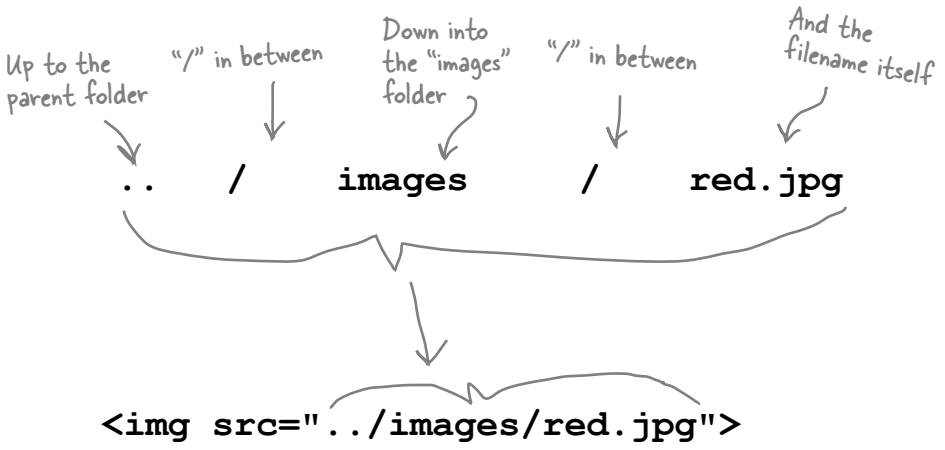
```
<img src="images/drinks.gif">
```

Finding the path from "elixir.html" to "red.jpg"

The elixirs page contains images of several drinks: "red.jpg", "green.jpg", "blue.jpg", and so on. Let's figure out the path to "red.jpg" and then the rest will have a similar path because they are all in the same folder:

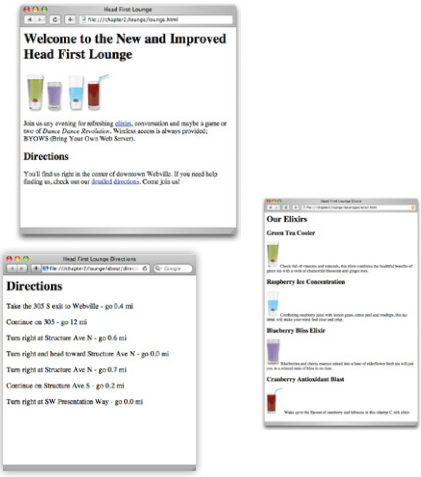


So putting (1) , (2), and (3) together, we get:



That covers all the links we broke when we reorganized the lounge, although you still need to fix the images in your "lounge.html" and "elixir.html" files. Here's exactly what you need to do:

1. In "lounge.html", update the image `src` attribute to have the value "images/drinks.gif".
2. In "elixir.html", update the image `src` attribute so that "../images/" comes before each image name.
3. Save both files and load "lounge.html" in your browser. You'll now be able to navigate between all the pages and view the images.



P.S. If you're having any trouble, the folder "chapter2/completelounge" contains a working version of the lounge. Double-check your work against it.

BULLET POINTS

- When you want to link from one page to another, use the <a> element.
- The href attribute of the <a> element specifies the destination of the link.
- The content of the <a> element is the label for the link. The label is what you see on the web page. By default, it's underlined to indicate you can click on it.
- You can use words or an image as the label for a link.
- When you click on a link, the browser loads the web page that's specified in the href attribute.
- You can link to files in the same folder, or files in other folders.
- A relative path is a link that points to other files on your website relative to the web page you're linking from. Just like on a map, the destination is relative to the starting point.
- Use ".." to link to a file that's one folder above the file you're linking from.
- ".." means "parent folder."
- Remember to separate the parts of your path with the "/" (forward slash) character.
- When your path to an image is incorrect, you'll see a broken image on your web page.
- Don't use spaces in the names you choose for files and folders for your website.
- It's a good idea to organize your website files early on in the process of building your site, so you don't have to change a bunch of paths later when the website grows.
- There are many ways to organize a website; how you do it is up to you.

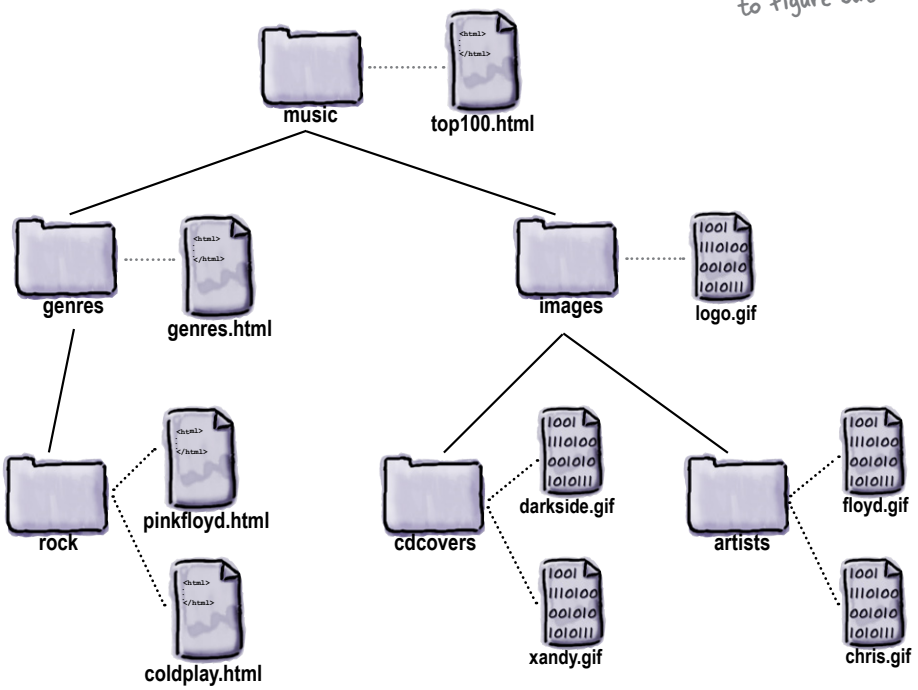
The Relativity Grand Challenge

Here's your chance to put your relativity skills to the test. We've got a website for the top 100 albums in a folder named "music". In this folder you'll find HTML files, other folders, and images. Your challenge is to find the relative paths we need so we can link from our web pages to other web pages and files.

On this page, you'll see the website structure; on the next page, you'll find the tasks to test your skills. For each source file and destination file, it's your job to make the correct relative path. If you succeed, you will truly be champion of relative paths.

Good luck!

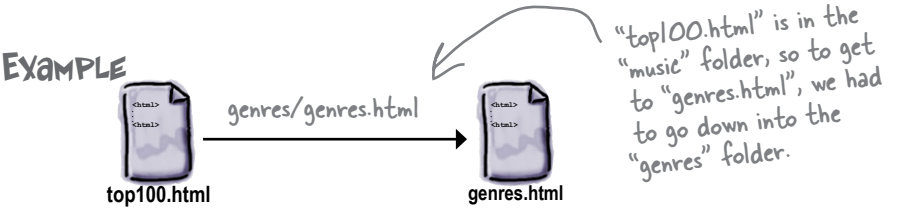
Feel free to draw right on this website picture to figure out the paths.



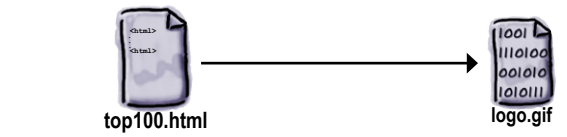
It's time for the competition to begin.

Ready...set...write!

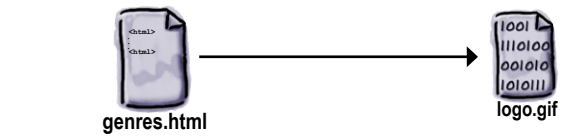
Example



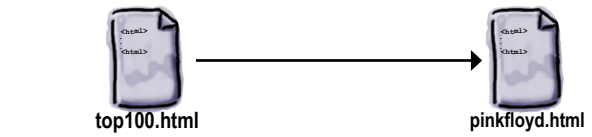
Round One



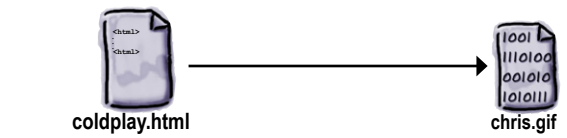
Round Two



Round Three

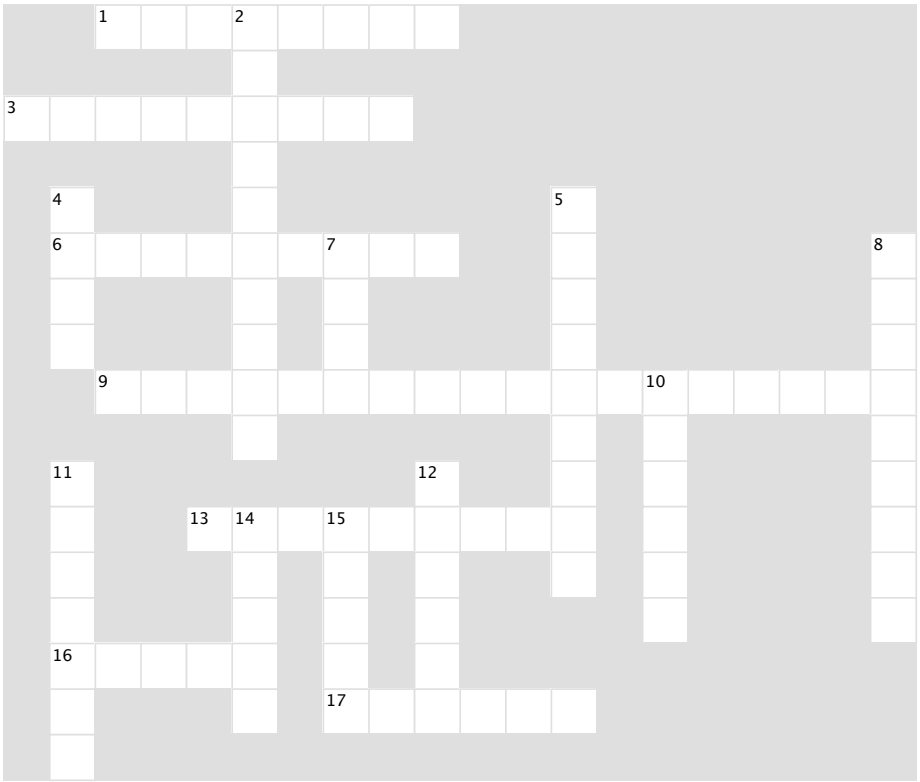


Bonus Round



HTMLcross

How does a crossword help you learn HTML? Well, all the words are HTML-related and from this chapter. In addition, the clues provide the mental twist and turns that will help you burn alternative routes to HTML right into your brain!



Across

1. "../myfiles/index.html" is this kind of link.
3. Another name for a folder.
6. Flavor of blue drink.
9. What href stands for.
13. Everything between the <a> and is this.
16. Can go in an <a> element, just like text.
17. Pronounced "..".

Down

2. href and src are two of these.
4. Hardest-working attribute on the Web.
5. Rhymes with href.
7. Top folder of your site.
8. The "HT" in HTML.
10. Healthy drink.
11. A folder at the same level.
12. Use .. to reach this kind of directory.
14. Text between the <a> tags acts as a _____.
15. A subfolder is also called this.

You needed to add a link with the label "Back to the Lounge" at the bottom of the elixir page that points back to "lounge.html". Here's our solution.

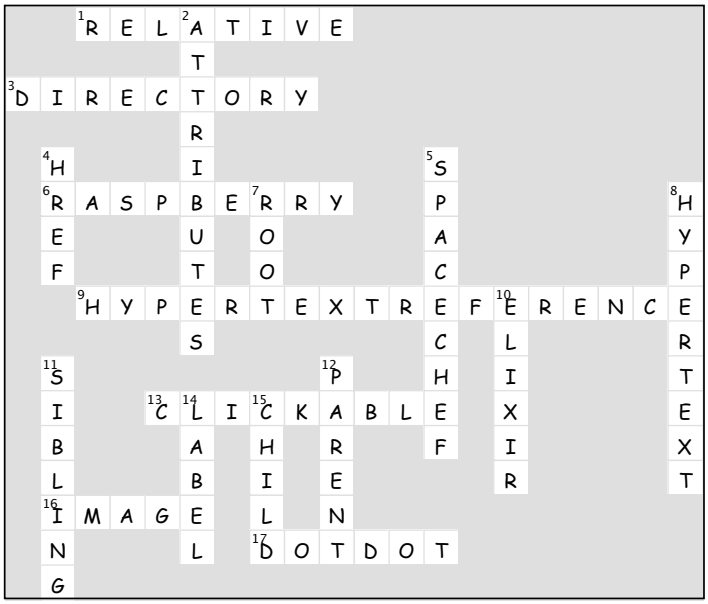

```
<html>
  <head>
    <title>Head First Lounge Elixirs</title>
  </head>
  <body>
    <h1>Our Elixirs</h1>

    <h2>Green Tea Cooler</h2>
    <p>
                <img src="green.jpg">
                Chock full of vitamins and mi
                combines the healthful benefi
                a twist of chamomile blossoms
    </p>
    <h2>Raspberry Ice Concentration</h2>
    <p>
                <img src="lightblue.jpg">
                Combining raspberry juice wit
                citrus peel and rosehips, thi
                will make your mind feel clea
    </p>
    <h2>Blueberry Bliss Elixir</h2>
    <p>
                <img src="blue.jpg">
                Blueberries and cherry essence mixed into a ba e
                of elderflower herb tea will put you in a relaxe
                state of bliss in no time.
    </p>
    <h2>Cranberry Antioxidant Blast</h2>
    <p>
                <img src="red.jpg">
                Wake up to the flavors of cranberry and hibiscus
                in this vitamin C rich elixir.
    </p>
    <p>
        <a href="lounge.html">Back to the Lounge</a>
    </p>
  </body>
</html>
```

Here's the new <a> element pointing back to the lounge.

We put the link inside its own paragraph to keep things tidy. We'll talk more about this in the next chapter.

Exercise solutions

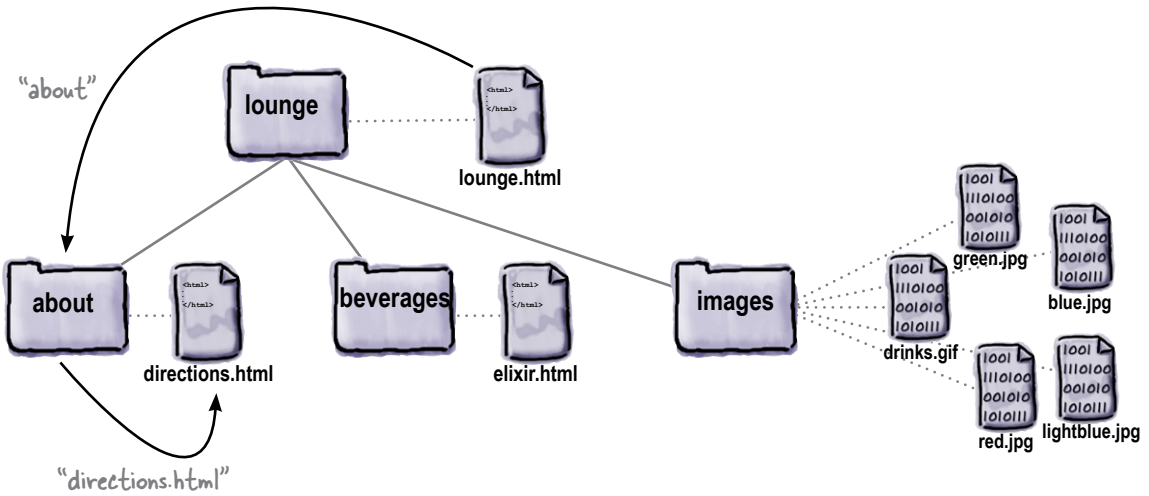
Label	Destination	Element
Hot or Not?	hot.html	Hot or Not?
Resume	cv.html	Resume
Eye Candy	candy.html	Eye Candy
See my mini	mini-cooper.html	See my mini
let's play	millionaire.html	let's play



Sharpen your pencil Solution

Trace the relative path from “lounge.html” to “directions.html”. When you’ve discovered it, complete the <a> element below.

Here’s the solution. Did you change both <a> elements in “lounge.html”?




```
<a href="about/directions.html">detailed directions</a>
```

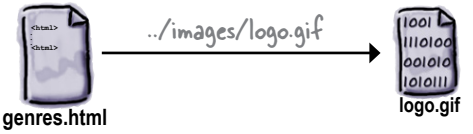
YOUR ANSWER HERE

The Relativity Grand Challenge Solution

Round One

top100.html is in the music folder, so to get to logo.gif, we had to go down into the images folder.

Round Two

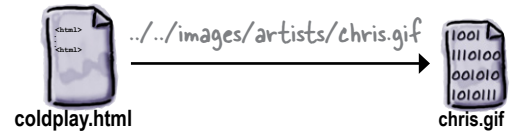


genres.html is down in the genres directory, so to get to logo.gif, we first had to go up to music, and then down into the images folder.

Round Three

From top100.html, we go down into genres, then down into rock, and find pinkfloyd.html.

Bonus Round



This was a tricky one. From coldplay.html, which is down in the rock folder, we had to go up TWO folders to get to music, and then go down into images, and finally artists, to find the image chris.gif. Whew!