

A large red arrow pointing downwards, serving as a background for the title.

EPUB

Straight to the Point

by Elizabeth Castro

© Peachpit Press

EPUB Straight to the Point: Creating ebooks for the Apple iPad and other ereaders

by Elizabeth Castro

Peachpit Press

1249 Eighth Street

Berkeley, CA 94710

510/524-2178

510/524-2221 (fax)

Find us on the Web at: www.peachpit.com

To report errors, please send a note to errata@peachpit.com.

Peachpit Press is a division of Pearson Education.

Copyright © 2011 by Elizabeth Castro

Editor: Clifford Colby

Production Editor: David Van Ness

Cover design: Aren Howell

Interior design: Elizabeth Castro

Notice of Rights

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

Notice of Liability

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

Trademarks

Apple, iBook, iBooks, iPad, iPhone, iTunes, and Mac are trademarks or registered trademarks of Apple Inc., registered in the U.S. and other countries

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

ISBN-13: 978-0-321-73468-6

ISBN-10: 0-321-73468-8

9 8 7 6 5 4 3 2 1

Printed and bound in the United States of America

Table of Contents

Introduction

Print vs. ebook vs. website	8
<i>Static vs. Dynamic</i>	9
<i>Appearance</i>	9
<i>How it's read</i>	10
<i>The order of things</i>	10
<i>Formats, durability, and batteries</i>	11
<i>Searchability</i>	11
<i>Highlighting and sharing passages</i>	12
<i>Copy protection</i>	13
<i>Buying new books</i>	13
What is EPUB?	14
<i>Navigating a sea of ereaders</i>	14
Anatomy of an iBooks page	16
Who is this book for?	18

Using Word to write EPUB

Styling your Word document	20
<i>Setting up styles in Word</i>	21
<i>Applying styles</i>	26
Saving Word files as HTML	28
Preparing HTML files for EPUB	31
<i>Using a text editor</i>	31
<i>Declaring the file to be XHTML, not HTML</i>	31
<i>Moving style data to its own file</i>	33
<i>Declaring the language used</i>	41
<i>Adding quotation marks around attributes</i>	41

using InDesign to create EPUB	45
About using InDesign for EPUB	46
<i>One file or many?</i>	47
Styles in InDesign	48
<i>Creating styles</i>	49
<i>Applying the main Body style</i>	50
<i>Applying headers, quotes, and other special styles</i>	52
<i>Replacing local formatting with styles</i>	53
<i>Drop Caps and Nested Styles</i>	57
Add images	61
<i>Placing an image</i>	62
<i>Creating text wrap within the flow</i>	64
Add links	70
<i>Creating a style for links</i>	70
<i>Hyperlinks</i>	71
<i>Cross-references</i>	73
Create a navigational TOC	76
<i>Preparing your book in order to create the navigational TOC</i>	76
<i>Creating a Table of Contents Style</i>	77
Add metadata to your ebook	80
Export EPUB from InDesign	82
<i>Exporting EPUB from InDesign CS4</i>	83
<i>Exporting EPUB from InDesign CS5</i>	87
Inside an EPUB file	91
Unzipping an EPUB	92
The files that make up an EPUB	94
<i>The mimetype file</i>	95
<i>The META-INF folder</i>	95
<i>The OEBPS folder</i>	95
<i>XHTML and CSS files</i>	96
<i>The toc.ncx file for the navigational TOC</i>	98
<i>Writing the content.opf file</i>	103
Creating the cover	113
Zipping and testing	119
<i>Organizing files before reziping</i>	119
<i>Reziping after edits</i>	120
<i>Getting the new EPUB file to the iPad</i>	122
<i>Further editing, reziping, and testing</i>	124
Validating your EPUB file	126
Converting EPUB to Kindle's Mobi	128

Advanced EPUB Formatting	129
Ensuring ereaders use your CSS	130
Cleaning up InDesign EPUB files	131
<i>How InDesign writes XHTML</i>	131
<i>How InDesign writes CSS</i>	132
Fonts in your ebook	138
<i>Choosing fonts</i>	138
<i>Fonts available for ebooks on the iPad</i>	141
<i>Ornaments, dingbats, and symbols</i>	150
<i>Using non-English fonts</i>	151
<i>Embedding fonts</i>	152
Controlling text alignment	154
Keeping elements together	155
<i>Controlling a header's position</i>	156
<i>Keeping captions with their images</i>	156
<i>Setting widows and orphans properties</i>	157
<i>Setting page break options</i>	158
Drop caps and small caps	159
<i>Having CSS mark the first letter and line</i>	159
<i>Tagging the first letter and first line explicitly</i>	161
Controlling spacing	163
Controlling indents	164
Formatting short lines	165
Borders and backgrounds	169
<i>Creating a sidebar</i>	169
Hyphenation	171
<i>Adding soft hyphens</i>	171
<i>Using left-aligned text</i>	172
Working with images	173
<i>Size</i>	173
<i>Wrapping text around images</i>	175
<i>Wrapping text around sidebars</i>	177
Creating links	179
Creating tables	181
Video in your ebook	183
<i>Creating your video</i>	183
<i>Adding code for the video</i>	184
Index	187

Thanks

As I was writing this book, I had the good fortune to join up with the knowledgeable and helpful folks who follow the [#eprdcn](#) hashtag on Twitter. This book would have been a much less valuable resource without the tips, thoughts, questions, suggestions and real world experience of Lindsey Martin (@crych), Joshua Tallent (@jtallent), David Blatner (@dblatter), Anne Marie Concepcion (@amarie), Tina Henderson (@tinahender), Colleen Cunningham (@BookDesignGirl), Rick Gordon (@rcgordon), Anthony Levings (@anthonylevings), David Mundie (@mundie1010), Titusz Pan (@Titusz), Koan-Sin Tan (@koansin), Adam Jury (@adamjury), Walt Shiel (@slipdown), Mike Cane (@mikecane), Jose Afonso Furtado (@jafurtado), and Guy L. Gonzalez (@glecharles) and the folks at @DigiBookWorld. There are many others. You can come join us too by following [#eprdcn](#).

Thanks also to Cliff Colby, my editor at Peachpit Press, who believed in this project, and who kept his calm when I moved things around at the last minute, and even let me move the last minutes themselves as Apple released new versions of iBooks.

And to David Van Ness, my production editor at Peachpit Press, whose help discovering all the widows and bad breaks help make this a much more pleasant read.

Finally, thanks to my friends (if I still have any) who endured months of “I’m almost done, can we get together *next* month” and to my family who quietly urged me on. I couldn’t have kept going without you. Writing a book is a crazy, intensive, very personal pursuit, which must both be done in solitude and with support. I have been very fortunate to have had both.

One of the things I love about EPUB is the same thing I loved about HTML in its infancy: anyone can do it. You, too, can use this technology to write and publish your own book. I hope you’ll tell me about it. You can find me at <http://www.twitter.com/@lizcastro>.

Introduction



With the announcement on January 27, 2010, of Apple's iPad, and its support for the EPUB format, the world of electronic books, or *ebooks* for short, took a quantum leap into the future. According to IDPF (International Digital Publishing Forum), ebook sales in the U.S. in January and February of 2010 were \$60.8 million, \$5 million more than the entire fourth quarter of 2009 (\$55 million). The iPad is not responsible for *all* of the excitement, but it sure isn't hurting things.

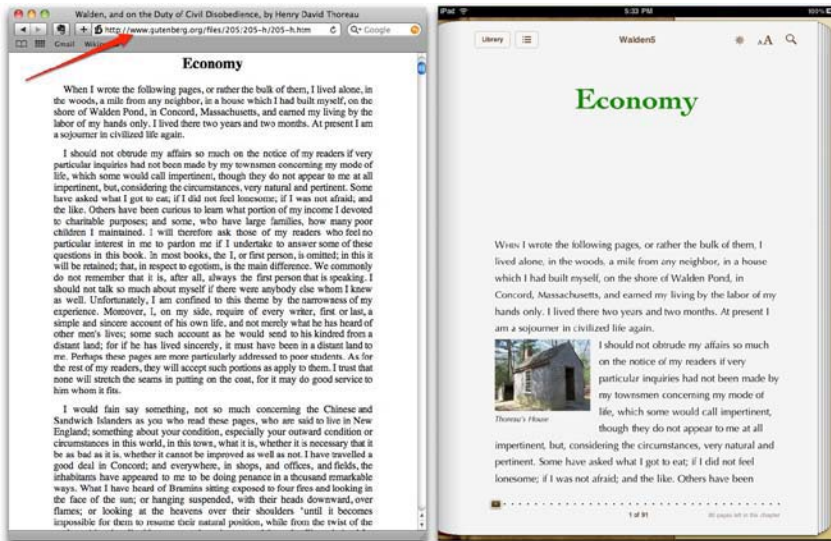
This book is for people who want to create their own EPUBs, and publish them for the iPad in particular, but also on other ereaders, like the Barnes and Noble Nook, Sony Reader, and desktop ereaders like Ibis and Stanza.

In this chapter we'll talk about:

- ▶ The differences and similarities among print books, ebooks and websites
- ▶ The EPUB format itself
- ▶ The size and structure of a page on the iPad
- ▶ Who this book was written for and where you can find updates, example code, and extras

Print vs. ebook vs. website

One of the first questions I thought about as I was writing this book was how an EPUB ebook was different from a website. Indeed, they have much in common. Both are written in (distinct but very similar flavors of) HTML and formatted with CSS, both adjust themselves to the constraints of the system in which they are viewed, and both are accessed electronically. Even their content might be similar; certainly there are websites (like Project Gutenberg) that make entire books available to readers. So what makes an ebook different?



The content is practically the same on the Gutenberg website (left) and the iBooks app on the iPad (right).

Another question that's helpful to explore is what distinguishes an ebook from a print book. Again, there are many similarities, most notably in the content, but also in the idea of the page as a unit of information. Although with an ebook, this might be changed once, as a reader adjusts the font size, for example, it would stay the same throughout the rest of the reading experience. The reader could go back and know that the section in the upper left of the previous page would still be in that section on the upper left of the previous page.

I have outlined a number of areas where ebooks, print books, and websites, overlap and diverge, which I hope will be illustrative.

Static vs. Dynamic

One principal feature of ebooks and websites that is distinct from printed books is how quickly they can be changed and indeed are *expected* to be updated. It is understood that a website will be frequently updated, while ebooks might be updated occasionally, and regular books only before a new printing. It's not only a question of correcting errors, or updating timely information, however. There is something about the static nature of a book that somehow gives it more solidness, more integrity. We judge a book, even an ebook, as a discrete collection of information, not as an ever-changing one.

Appearance

A printed book cannot change its appearance, although publishers sometimes offer various renditions of the same book: hard cover, paperback, large print, and so on. A single version of an ebook, however, can generally be modified by the reader to their own taste, by changing the font, text size, and sometimes even the color of the text and/or background. Although it's also possible for users to change the appearance of a website (perhaps by choosing a custom style sheet or increasing the font size), it's not nearly as widespread—or expected—as with an ebook.

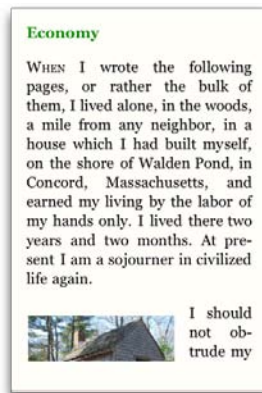


A reader might choose a different size or font in which to read the ebook. This is, of course, unheard of for a printed book.

How it's read

Printed books are generally read left-to-right or right-to-left. Websites tend to scroll up and down. Currently, many ebooks mimic the behavior of print books, surely in an attempt to make the transition to ebooks less uncomfortable for readers, who are accustomed to having a discrete amount of information on a page that doesn't change size or shape.

One of the cool things about an ebook is that it *reflows* to fit the size of the device in which it is being read. If you're reading it on an iPhone, the width of the page is a fair bit smaller than if you're reading it on an iPad, or on some other reader. The beauty of EPUB is that it flows the text to fit whatever screen it's on.



Here is the same file from the last few illustrations shown in Stanza on the iPhone.

This is different from a zoom function, that might let you magnify one area of a page, but doesn't flow the entire page to fit. With zoom, you can get the text big enough to read, but then it's a nuisance to navigate from one page to the next.

The order of things

With a printed book, you're used to opening up the cover, leafing through a title page, copyright, table of contents, dedication, and even a preface before diving into the main content of a book. With an ebook, the book designer can control where you start reading. The first time you open an ebook, you might be thrust onto the first page of the content (if the designer thinks you'll be annoyed by front matter). In this respect, ebooks are more similar to websites.

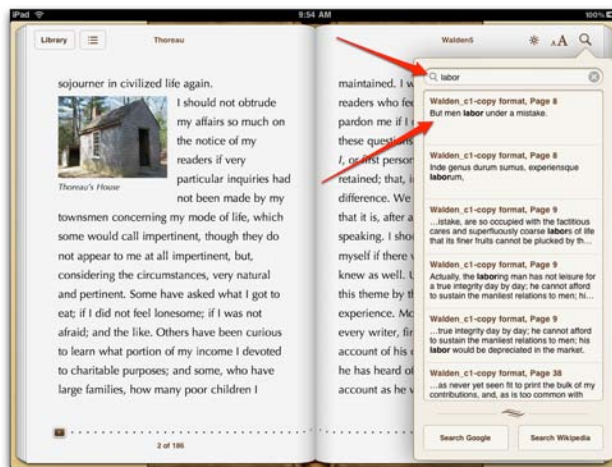
In a print book, you can often, but not always, consult a table of contents and then jump to a desired section. In ebooks (and websites), not only can you access the table of contents from any page, you often will find links in the text that transport you to other sections of the book, or even to related external websites.

Formats, durability, and batteries

Print books don't become obsolete, don't need batteries, and can be read in many different environments—including on a beach where sunlight and sand might make ereader devices less advantageous. They do require external illumination, however, if you're using them in the dark, something which some readers, including the iPad and iPhone, do not. Books are more sturdy than ereaders, and don't break when dropped or if they slip off the bed. And they certainly require a lot less outlay at the outset.

Searchability

A printed book's main search tools are a table of contents and an index, though the latter are only prevalent in nonfiction books. Most ereaders, however, offer some sort of electronic search of the full-text content of the ebook, in addition to a navigable table of contents, and less commonly, an index whose entries are linked to the referenced sections. Web browsers, too, commonly offer full-text search. Search, however, should not be seen as a reasonable substitute for a targeted index.



In iBooks on the iPad (and most other ereaders), you can search for words within your book and then click the found text to jump directly to that passage.

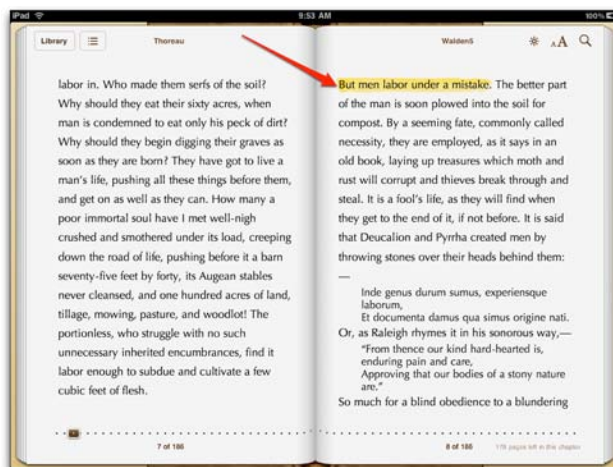


The table of contents in an ebook is navigable: if you click one of the entries, you automatically jump to the selected section.

Highlighting and sharing passages

Of course, one of the most popular things to do with any kind of book is share it. A printed book doesn't have any special tools for allowing highlighting and sharing, apart from being open to pencils and highlighters and to being handed to friends. The ability to highlight and share an ebook depends on the ereader's capacities.

Some ereaders, like the iPad, allow you to highlight a passage for future reference, but have no sharing abilities at all. Indeed, you can't even copy a passage!



You can highlight text in the iPad (by selecting the text, and choosing bookmark) for later reference.

Others, like Kindle, let you create both notes and highlights, that you can view either in the book itself or online. Currently, the most highlighted passages in Kindle books are published on Amazon's site, but personal notes and highlights are not yet shared among readers. The Barnes and Noble Nook lets you share entire books with friends, though you're currently only allowed to share a particular book one time and with one person.

There are many tools for copying and sharing passages from websites. Indeed, there are tools for including chunks of a website in a different site altogether.

Copy protection

The EPUB format allows for DRM (Digital Rights Management) encryption so that the file can be read in only one specific kind of ereader and only by authorized users. It's a shame, because while EPUB itself is very widely supported, the DRM severely limits that versatility, while it makes it much more difficult for the licensed reader to access the content, which they have come by legally. For example, if you buy a book through iBooks on the iPad, the added DRM will impede you from reading the book on the Sony Reader, B&N Nook, or even Stanza, even though they all support EPUB. Printed books have no such DRM. Websites don't have DRM as such though some are located behind firewalls or subscription services.

Buying new books

While print books have long had advertisements and excerpts from sequels or other related books that publishers hoped people would buy, ebooks can contain direct links to bookstores that facilitate the immediate purchase of another ebook. On the iPad, readers can access Apple's iBookstore from within the iBooks app. Ebooks can also contain links to external websites and other sources of marketing and information in order to generate additional sales. Websites can have links to other websites as well as to ebooks in all the various ebook stores.

What is EPUB?

The most widely accepted format for ebooks today is EPUB, which is developed and maintained by the IDPF. You can find the official specifications for EPUB documents on its website: www.idpf.org under, well, *Specifications*.

An EPUB document is a specially constructed zip file with the *.epub* extension. An ereader can reflow the content of an EPUB document into any size display screen, from a phone to a desktop monitor. EPUB also allows for the generation of a navigational table of contents.

The content of a book formatted with EPUB is contained in XHTML and CSS files, which may reference images and embedded fonts, and be encrypted with DRM. XHTML is a special flavor of HTML, which is the language that all web pages are written in. The EPUB file also contains a series of XML files that help format the book so that it can be properly read by an ereader.

There are a number of tools that can generate EPUB files for you, either from plain text, from XHTML, from Microsoft Word, or even from Adobe InDesign. Still, in these early days when EPUB tools are less than perfect, it's a good idea to know what's going on under the hood so that you can go in and make necessary adjustments. For example, Word doesn't export drop caps, but you can edit the XHTML files by hand to allow them. InDesign doesn't export text wrap with its EPUB documents, but you can set up the files so that a quick edit to the XHTML achieves that aim. In the rest of this book, I'll show you both how to use available tools, and how to handcode extra features.

Navigating a sea of ereaders

There are a number of reader applications, or *ereaders*, that can read EPUB documents. iBooks on the iPad, Barnes & Noble Nook, and the Sony Reader support EPUB, as do Adobe Digital Editions, Lucidor, and Stanza (on various platforms), Ibis Reader (which is web-based), Mobipocket on Blackberry and Aldiko on Android, and many more.

The most well-known ereader that *does not* support EPUB is Amazon's Kindle. I suspect that may change as more and more ereaders join forces behind EPUB, but only time will tell.

Unfortunately, not every ereader reads and interprets EPUBs in exactly the same way. Because the earliest popular ereaders (like Stanza) did not support any formatting at all, later ereaders felt forced to compensate by adding default formatting of their own and ignoring the formatting of the EPUB documents they displayed.

As EPUB designers have gotten more savvy, however, they have chafed under the sometimes overbearing nature of these ereaders, who instead of following the standards laid out in the EPUB specs, insist on overriding EPUB designs to make up for old issues. Ebook designers are discouraged from even *choosing the font* for their book—something a print book designer would never stand for—in the supposed interest of a good user experience.

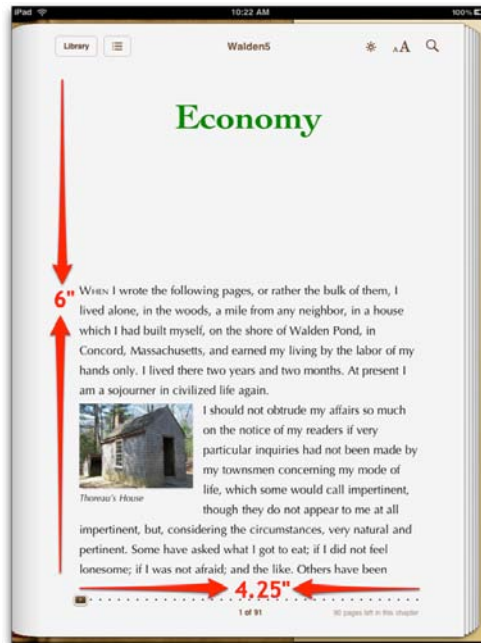
Personally, I fail to see how properly designing a book takes away from a good user experience. Quite the contrary. Instead, in this book, I encourage you to follow the standards laid out in the EPUB specifications and to speak up for standards support in all ereaders.

Anatomy of an iBooks page

While this book explains how to create standard EPUB documents, it is also particularly focused on how to take best advantage of these standard features to make beautiful ebooks on the Apple iPad.

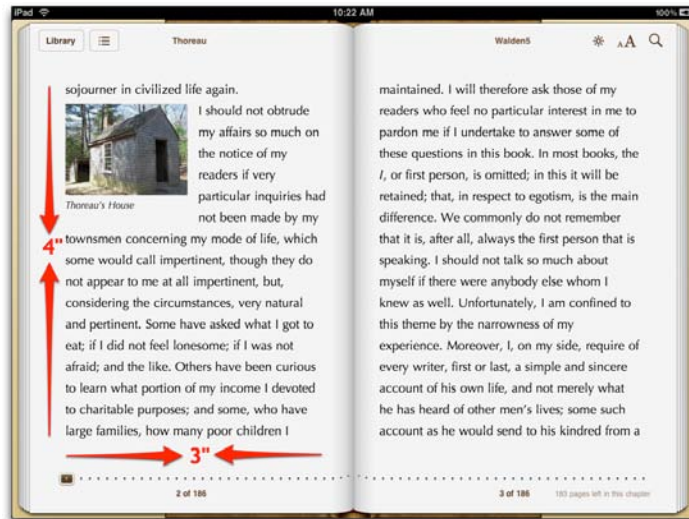
The iPad displays ebooks in two sizes: a single large page if you hold the iPad vertically and a spread of two smaller pages if you rotate the display to a horizontal orientation.

The size of the single vertical page is about 5.5 x 7.5" (about 15 x 19 cm), although a fair bit of that space is taken up with navigation tools and margins, leaving a content frame of about 4.25 x 6" (11 x 15 cm).



When rotated vertically, iBooks displays a single page of the ebook.

If you rotate the iPad to a horizontal orientation, you get two vertically oriented pages, each of which measures about 3.75 x 5.5" (10 x 14 cm), which, when you take away the navigation buttons and margins, leaves you with a content area about 3 x 4" (7.5 x 10 cm).



When rotated vertically, iBooks on the iPad shows two smaller vertically oriented pages.

The resolution of the iPad is 132 dpi, which is considerably higher than the average 98 dpi of a desktop monitor. This means that text and images will be physically smaller on the iPad, although they may seem about the same size since everything will be in proportion. If you call for text at 16 pixels, the iPad will display it at $16/132$ pixels, which is about .12" or about 9 points. Curiously, if you specify a size of 12 points, it will also be displayed at 16 pixels or 9 actual points. So much for absolute measurements.

The iPad displays text in Palatino, by default, but the reader can also choose Baskerville, Cochin, Times New Roman, Verdana, and Georgia. All but Verdana are serif fonts. The reader can also choose from 10 different font sizes, with size 4 being the default.

The iBooks application does not yet support embedded fonts, even though the iPad itself does (for example, in Safari), as long as these are in SVG format. One might presume that iBooks will support embedded SVG formats as well at some point.

The iPad does have a number of system fonts that can be used in ebooks viewed both on Safari and in the iBooks application. I will show you what fonts are available and how to call them in "Fonts in your ebook" in Chapter 4.

Who is this book for?

This book is for anyone who wants to publish an ebook in EPUB format, particularly on the iPad, but for any ereader that supports EPUB, including the Sony Reader, Barnes & Noble Nook, Ibis Reader, and Stanza. It explains how to use Word and InDesign—software you may already own and which might already contain your formatted books—to generate the files necessary that make up the EPUB, as well as how to manually create or improve the files in order to take advantage of the capabilities of the most advanced ereaders, without leaving underperforming ereaders too far behind. I believe strongly in following standards so that a book that works today will continue to work tomorrow in the next new ereader that comes along.

You don't need to have either Word or InDesign to create an EPUB document; it is very possible to write the files by hand. Nevertheless, these tools facilitate the creation of an EPUB, and take advantage of the formatted documents you may already have created with those programs.

It is essential to have a good text editor so that you can adjust and adapt the files once they are created. I provide details and recommendations in the corresponding sections.

Finally, it's very helpful if you have some knowledge of XHTML and CSS, since that's what EPUB is based on and I don't discuss XHTML or CSS basics much at all. If you are not familiar with HTML, XHTML, or CSS, I recommend taking a look at my bestselling *HTML, XHTML, and CSS: Visual QuickStart Guide, 6th edition*, also published by Peachpit Press. Many of the same techniques for designing websites are also valid for designing ebooks.

I will be posting updates, errata, extras, and more information on my website: <http://www.elizabethcastro.com/epub> as well as on my blog, Pigs, Gourds, and Wikis (<http://www.pigsgourdsandwikis.com>).

Using Word to write EPUB

1

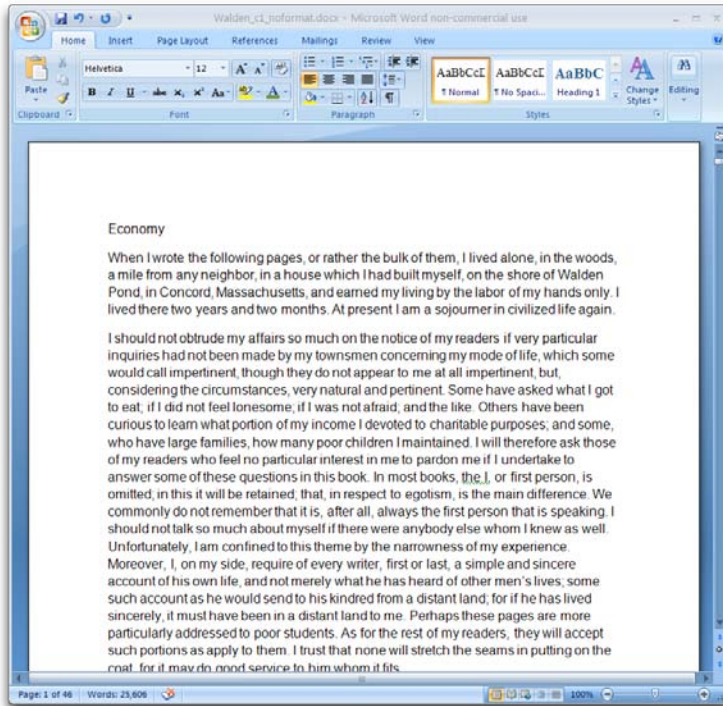
Microsoft Word is the most popular word processor in the world so it's fortunate that you can use Word files as the basis for generating an ebook in EPUB format. Unfortunately, EPUB can't recognize Word directly, so you'll have to save your documents in HTML format, and then adjust the files so that EPUB can use them. I'll show you the whole process in this chapter, particularly how to:

- ▶ Style your Word document
- ▶ Save Word files as HTML
- ▶ Prepare your Word-generated HTML files for EPUB

Of course, you don't have to use Word. If you have InDesign, you can skip Word completely (and begin in the next chapter). You can even use a text editor to write your XHTML and CSS files from scratch, following the guidelines in Chapter 3, "Inside an EPUB file".

Styling your Word document

Suppose for a minute that you're Henry David Thoreau and someone has kindly time-transported you a computer and a copy of Microsoft Word. You've just finished writing the first chapter of *Walden*.



The first chapter of Walden, without formatting.

It's a good idea to apply at least basic formatting, in order to distinguish the main body from chapter headers, quoted material, and the like. Indeed, in this document, there are three main styles to be applied: Normal, for the main text, Quote, for any of the single-spaced, short lines that Thoreau has set off from the main body of text, and Heading 1, for the chapter name.

Tip

Even if you don't care about the formatting itself in Word, you should apply styles in order to facilitate editing the CSS in the EPUB later. An empty style can serve as a tag to identify each kind of content.

Setting up styles in Word

Word supplies a set of Default styles with every new document. These might work for you as is, or you can modify them as desired. In this example, we're going to adapt Word's default Normal, Quote, Heading 1, and Emphasis styles to our needs.

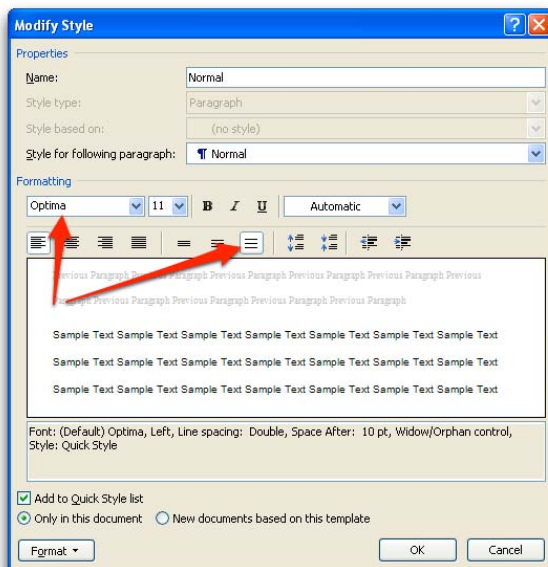
When you open a brand new document in Word, a series of styles are displayed by default:



Some of Word's default styles

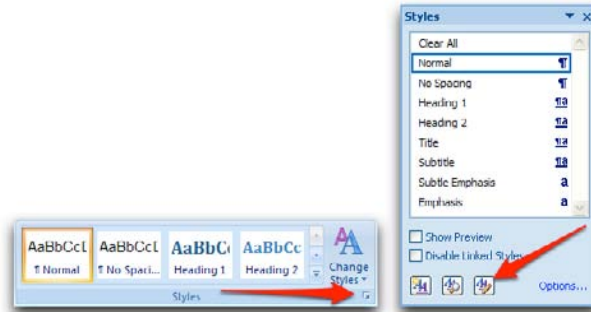
It's easy to change and reorder these styles so that they reflect and facilitate your own formatting needs.

- 1** First, right-click the Normal style and choose Modify from the flyout menu. Choose the desired formatting characteristics from the dialog box that appears. In this example, I want the Normal style to use the Optima font at 11 points and be double-spaced.



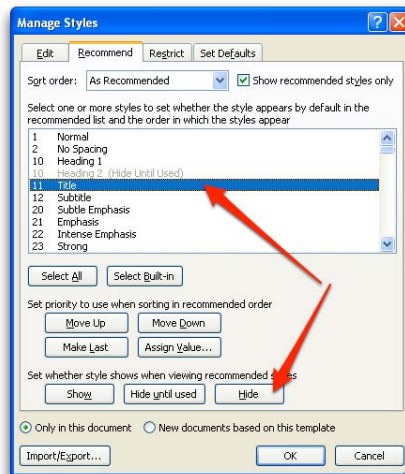
Choose the desired formatting characteristics from the Modify Style box and click OK.

- 2 Change the Heading 1 style so that it is centered, and uses a sans-serif font, at 36 points, in a green color, with 100 points of space after it (to separate the chapter name from the rest of the book). The Quote style should be single-spaced. No changes are needed for the default Emphasis style.
- 3 Next, click the little button underneath the Change Styles option in the Home toolbar in order to display the Styles palette. In the Styles palette, click the right-most button at the bottom to open the Manage Styles box.



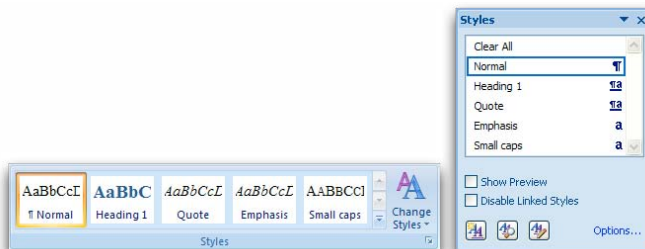
Click the tiny button in the lower right corner to reveal the Styles palette. Then, in the Styles palette, click the Manage Styles button.

- 4 In the Manage Styles box, select the styles you don't need and click Hide or Hide until used. This will remove them from the main palette and make the ones you do need easier to find and apply.



Hiding unused styles makes applying the ones you want faster and easier.

5 You can also reorder the styles to put the ones that you use most frequently at the top of the list. Select the style and either click Move Up or Move Down. (Or click Assign Value to give the style a number that determines its place in the list.) Click OK when you're satisfied.

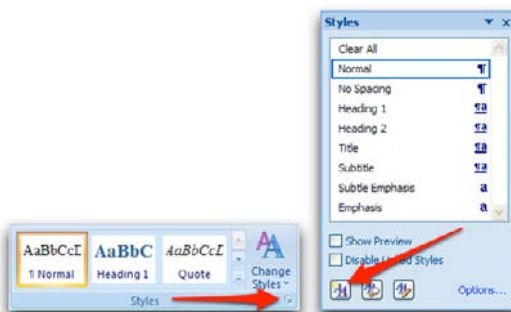


It's a lot easier to apply styles when you don't have to sift through the styles to find the right one.

Adding new styles

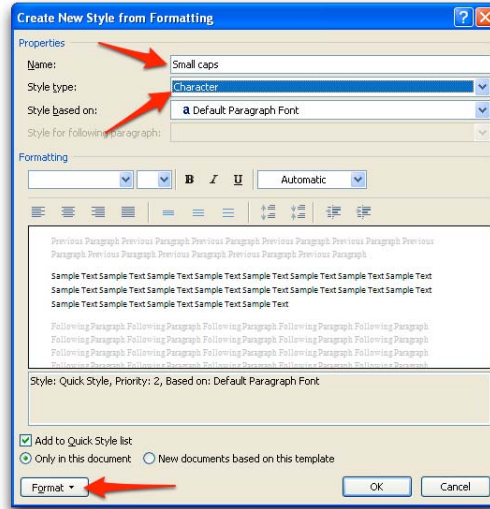
While it's probably easier to adapt Word's default paragraph styles (in part because they already use the most useful and descriptive names), it's often easier to create *character* styles from scratch. Of course, you can create brand new paragraph styles as well (just choose Paragraph instead of Character for Style type).

1 Begin by clicking the little button underneath the Change Styles option in the Home toolbar in order to display the Styles palette. Then, at the bottom of the Styles palette, click the New Style button (at the far left).



Click the tiny button to reveal the Styles palette and then click the New Style button in the Styles palette that appears.

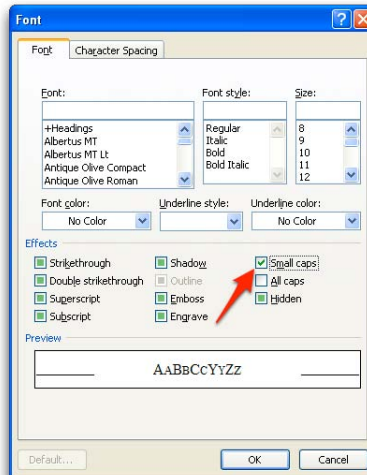
2 In the New Style box, type the name for the new style (*Small caps* in our example), and choose Character since we only want to apply this style to a selection of characters or words (and not to entire paragraphs).



The New Style box lets you define which formatting characteristics are applied with a given style.

3 Next, choose Format > Font from the menu in the bottom-left corner of the box to show the Font formatting options.

4 Click the Small caps checkbox and click OK twice.

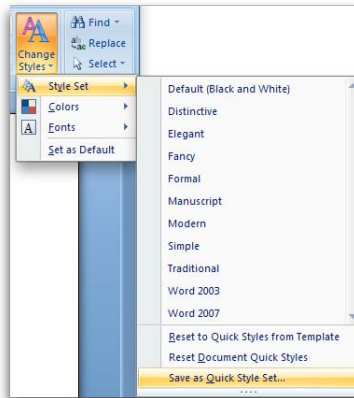


Check the Small caps option to add this formatting to the new style.

Loading styles from another document

Once you have the desired styles set up in one document, you can load them into other documents quite easily.

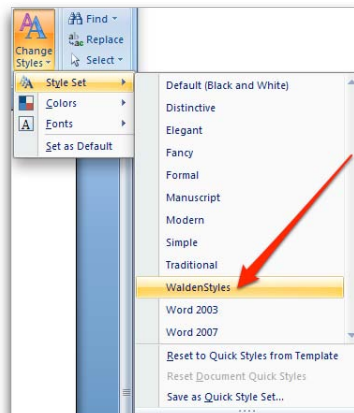
- 1 The first step is to save the styles into a template. Begin by choosing Style Set > Save as Quick Style Set... below the Change Styles icon in the Styles section of the Home toolbar. Give the Style Set a descriptive name. (In this example, I'll use *WaldenStyles*.)



This option is a bit hidden!

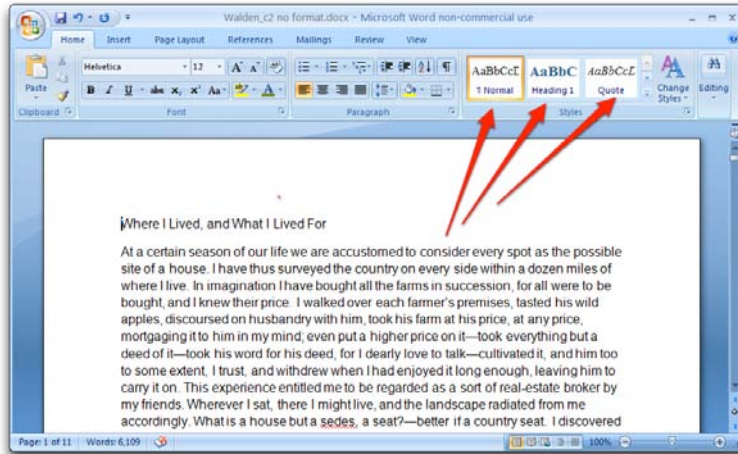
You can also download the styles from this example from the book's website.

- 2 To load styles into a new document, open the new document, click Change Styles, and choose Style Set > WaldenStyles from the pop-up menu.



Once you save your style set, it will appear in the Style Set menu and can be applied to different documents.

3 Now you can begin to apply styles to the newly loaded document, as we'll discuss in the following section.

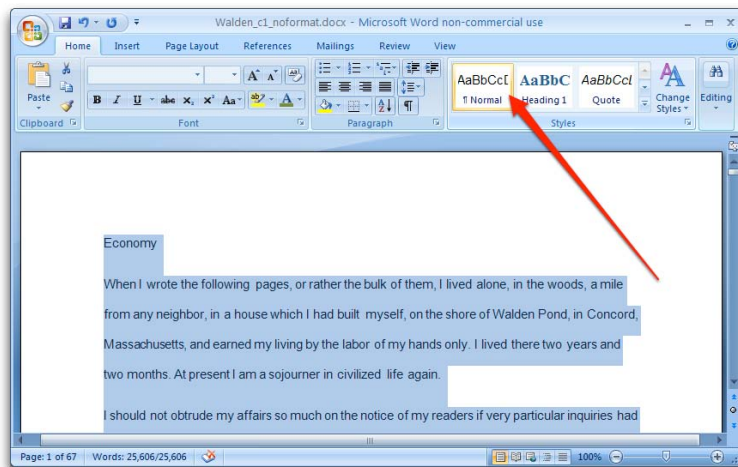


With the new styles loaded, it will be easy to format the second chapter.

Applying styles

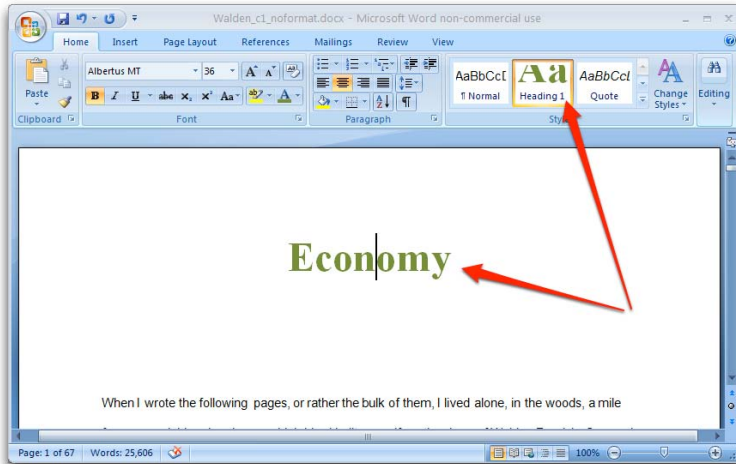
Since most of the content will be in the Body format, it makes sense to apply that style first.

1 Select the entire contents of the document (Control or Command-A), and then click the Normal style in the toolbar.



Style the most common type of paragraph first.

2 Next, style the chapter title by selecting it and choosing Heading 1.



The Heading 1 style centers the text and displays it in big green letters with a large space before the next paragraph.

3 Continue applying styles until the document is complete, for example, style the quotation sections with Quote, the italic words with Emphasis, and the first word of the first paragraph on the page with Small caps.

Tip

The more you use styles to format your documents instead of local formatting, the easier it will be to generate HTML files for use with EPUB.

Heading styles in Word (Heading 1, Heading 2, and so on) are automatically converted to heading styles in HTML (`h1`, `h2`, and so on). All other styles are created with `p` elements and classes.

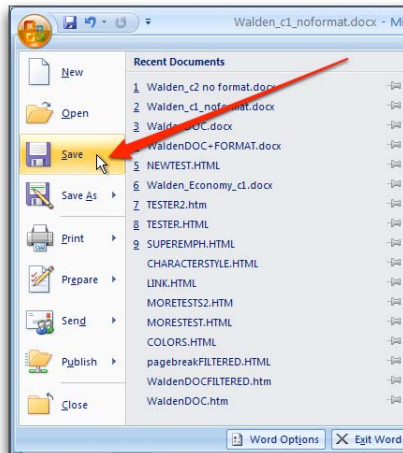
saving word files as HTML

Unfortunately, EPUB can't deal with Word files directly. The solution is to export the files from Word and then retouch them slightly so that EPUB can use them.

Word's HTML generation is both very robust, and very quirky. It does some things really well, and others very badly. We'll take advantage of Word's strengths, and then edit the documents manually to finish the job.

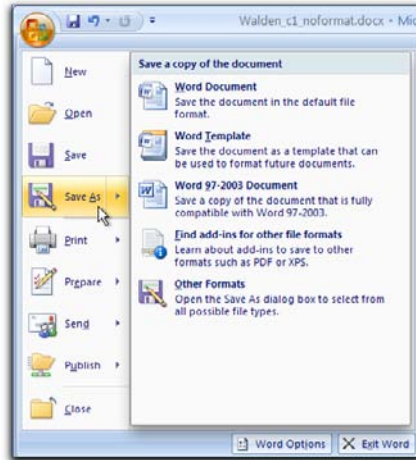
1 Apply formatting to your book's documents with styles as explained in the last section.

2 Make sure you save your document as a Word file before proceeding. (This is important since some formatting is permanently lost when you save as HTML.) To do so, click the Home button and then choose Save. If prompted to do so, choose the regular Word format (.doc or .docx).



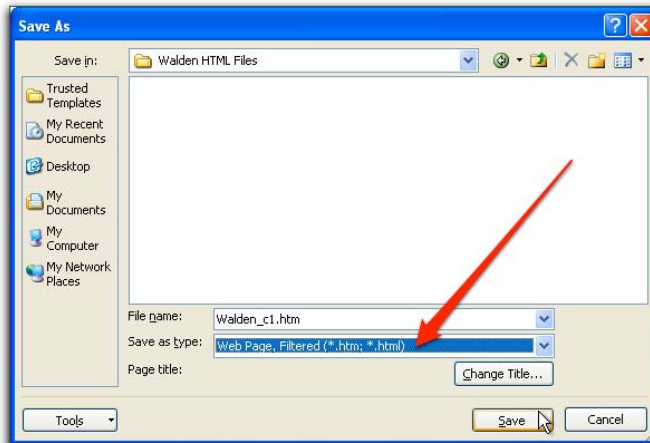
Make sure you save the changes in your Word document before saving as HTML.

3 Now return to the Home button, and this time choose Save As. You don't have to choose an option from the flyout menu.



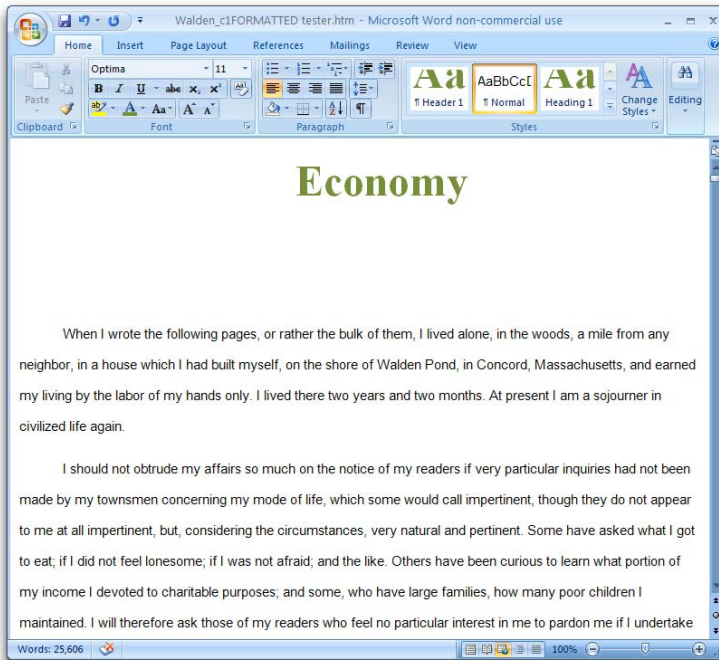
You'll now create an alternate version of your document in HTML format.

4 Choose Filtered HTML in the Save as type menu in the Save As dialog box. Remember, HTML is the necessary format for the content files of your EPUB document. Word offers both HTML and Filtered HTML. The latter has less extraneous information than the former and is the preferred option.

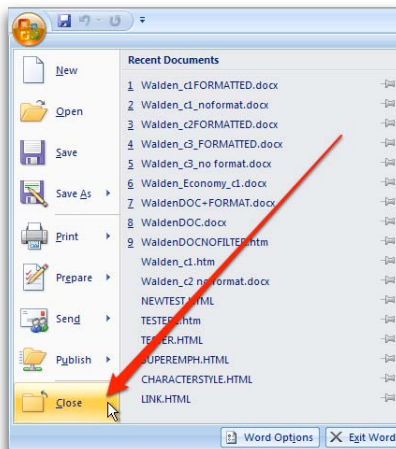


Choose Word's Filtered HTML option which generates files closer to what we need than its regular HTML option.

5 The document that is now displayed is Word's rendition of the HTML version of your document. It may look very much like your actual formatted document, but I don't recommend making any changes here. Instead, immediately choose Close from the menu that appears when you click the Home button.



The Filtered HTML version may look a lot like the regular Word version, but loses many of Word's features. It's better not to make changes here.



Once the HTML file is created, you should immediately close it!

Preparing HTML files for EPUB

I have to admit I'm not a big Word user. About 20 years ago, I was proud of being a master at Word 3, but I haven't used the program much since then. In the interim, I heard that the software had gotten rather bloated, that it performed every function under the sun, but added a lot of extraneous stuff you didn't need. In particular, I had heard that the HTML it generated was filled with garbage.

Thankfully, it's not quite as bad as all that. It's nowhere near perfect, or even what you need for EPUB, but if you're already a Word user, it can do part of the job of generating your files for you. But you'll still have to clean up the files and make them EPUB ready before going on to creating the EPUB itself.

Using a text editor

Microsoft Word is a *word processor*, and saves documents in its own proprietary format (.doc or .docx). In order to clean up the HTML files in a way that EPUB likes, you'll need to use a *text editor*, that can save the files in plain text format, with an .html or .xhtml extension.

For Macintosh, I am a longtime user of BBEdit (\$125), <http://www.barebones.com>. There is a free version called TextWrangler. Another good option is TextMate, from <http://macromates.com/> which costs about \$50. For Windows, I've heard good things about Notepad++, which you can download from <http://notepad-plus-plus.org/>. Be sure to get a text editor that supports GREP, which lets you use wildcards to turbocharge search and replace, and which is essential for massaging EPUB files. I'll show you how to perform many EPUB-specific GREP techniques in this book.

It's very important that you use a *text editor*, and *not* Microsoft Word, for the rest of the steps in this chapter.

Declaring the file to be XHTML, not HTML

Word creates HTML files but EPUB requires XHTML files. Luckily, the two formats are very similar. The first major difference between them is the header at the top of the document. The HTML generated by Word looks like this:

```
<html>  
<head>
```

- 1 Replace that heading with the code that every XHTML document for EPUB must begin with. To wit:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.
w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

This declares that the document is an XHTML file. Of course, declaring it is just a start. We will adjust the rest of the code so that it really will be XHTML and not HTML. Every XHTML file used for EPUB should begin with the above code.

Declaring the character set

The next requirement of an XHTML file that Word fails to meet is that it use the UTF-8 character set encoding. Word instead uses your system's default character set encoding, which for the English language version of Windows is Windows-1252.

In the Word generated HTML, it looks like this:

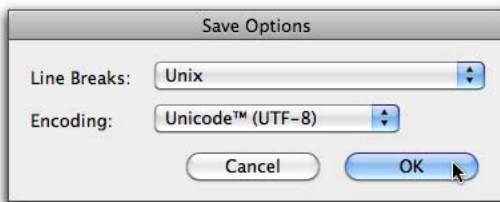
```
<meta http-equiv=Content-Type content="text/html;
charset=windows-1252">
```

- 1 Change the value of the `charset` variable to `utf-8`. Be careful not to eliminate the closing quotation marks by accident. Note that the opening quotation marks precede `text/html`, as they should.

```
<meta http-equiv=Content-Type content="text/html; charset=utf-8">
```

Those familiar with XHTML will notice that the first attribute is not quoted and there is a missing closing slash at the end of the line. We will complete these changes in an automated way further ahead, but feel free to change them now if you like.

2 If you're on a Mac, choose File > Save as and then click Options to choose Unix line breaks and the proper encoding (UTF-8). Now the actual encoding will match what we've promised. Replace the original file with the new properly encoded one. Windows users should use the default line endings.



Don't worry if the option in your program is worded slightly differently as long as it includes UTF-8.

Tip

Word also generates a meta element to proudly state that it generated your HTML code. You can remove or edit this as you like.

```
<meta name=Generator content="Microsoft Word 12 (filtered)">
```

(We'll add the missing quotation marks automatically later on in this chapter in "Adding quotation marks around attributes".)

Moving style data to its own file

When Word generates HTML, it outputs a style sheet right in the HTML file itself. Because most EPUB documents are made up of multiple XHTML files that should use the same style information, it makes more sense to copy the style sheet into a separate external file that can be accessed by all the XHTML files in the EPUB document.

Unfortunately, much of the style information is largely ineffective for EPUB documents since Word uses physical size units (points and inches) instead of relative size units (like ems and pixels). In addition, it adds a lot of extraneous font information that is likewise not used by the EPUB.

We'll go through it line-by-line.

- 1 Select the style sheet contents from `<style>` to `</style>`, inclusive, and choose File > Cut to remove it from the XHTML file.
- 2 Next, open a new document with a text editor and choose File > Paste to place the style content into it.
- 3 Save the newly independent CSS style sheet as plain text with the .css extension. Keep it open as there are more adjustments to be made.
- 4 Remove the opening and closing `<style>` tags. They are not required in a standalone CSS file. (If you did decide to keep the style information in the XHTML document, the opening tag should include `type="text/css"`.)
- 5 Finally, remove the commenting `<!--` and `-->` from the beginning and end of the style information, respectively.
- 6 Don't forget to save your changes!

Fixing embedded font information

Return to the CSS style sheet you created in the previous section. The code for embedding fonts that Microsoft includes in the style section both doesn't work on the iPad and has extra information you don't need.

```

/* Font Definitions */
@font-face
{font-family:Helvetica;
panose-1:2 11 6 4 2 2 2 2 4;}
@font-face
{font-family:"Cambria Math";
panose-1:2 4 5 3 5 4 6 3 2 4;}
@font-face
{font-family:Optima;
panose-1:0 0 0 0 0 0 0 0 0;}
@font-face
{font-family:"Albertus MT";
panose-1:0 0 0 0 0 0 0 0 0;}

```

I recommend eliminating the entire section. If you want to embed fonts, there's detailed and updated information in "Fonts in your ebook" in Chapter 4.

Removing extraneous word links

The next thing you can remove from Word's style information is anyplace where it says `mso-style-link` and the text that follows it, up to the first semicolon. So, it used to look like this:

```
h1
{mso-style-link:"Heading 1 Char";
margin-top:0in;
margin-right:0in;
margin-bottom:100.0pt;
margin-left:0in;
text-align:center;
page-break-after:avoid;
font-size:36.0pt;
font-family:"Albertus MT","serif";
color:#76923C;}
```

1

But the first declaration should be deleted altogether, ending up like this:

```
h1
{margin-top:0in;
margin-right:0in;
margin-bottom:100.0pt;
margin-left:0in;
text-align:center;
page-break-after:avoid;
font-size:36.0pt;
font-family:"Albertus MT","serif";
color:#76923C;}
```

2

Repeat the previous step and remove every similar instance of `mso-style-link`.

Collecting style information together

Because of Word's peculiar Linked styles, which apply the character portions of a style to a selection of characters or words and the paragraph portions of a style to a paragraph that just contains the cursor, Word divides the formatting for those styles into two chunks, roughly equivalent to the character features and the paragraph features.

```
h1
{margin-top:0in;
margin-right:0in;
margin-bottom:100.0pt;
margin-left:0in;
text-align:center;
page-break-after:avoid;
font-size:36.0pt;
font-family:"Albertus MT","serif";
color:#76923C;}

span.Heading1Char
{mso-style-name:"Heading 1 Char";
mso-style-link:"Heading 1";
font-family:"Albertus MT","serif";
color:#76923C;
font-weight:bold;}
```

There are a couple of things going on here. Notice that for every heading style in your Word document (Heading 1, 2, and so on), Word generates a heading style in your XHTML (`h1`, `h2`, and so on) as well as a corresponding set of property/value pairs in the CSS. But since Heading styles are created by Word by default as *Linked styles*, Word separates out some of the character information into a `Heading1Char` class and leaves some of it in the definition of the `h1` rule. Indeed, some of the properties are inexplicably duplicated in both declarations. Bloat indeed.

I would recommend consolidating all of the style definitions and applying them to the single `h1` selector. This is more clear, and much easier to edit and update.

In this example, there is only one rule for the `Heading1Char` class that doesn't already exist for the `h1` rule:

```
font-weight:bold;
```

I'm not exactly sure where Word draws the line in terms of determining if a given bit of formatting is "character" or "paragraph". It doesn't seem particularly logical to apply a color, font size, or font family to a paragraph but not the font weight. But what do I know?

1 Consolidate the style rules with the main selector and eliminate the other set of rules altogether.

```
h1
{margin-top:0in;
margin-right:0in;
margin-bottom:100.0pt;
margin-left:0in;
text-align:center;
page-break-after:avoid;
font-size:36.0pt;
font-family:"Albertus MT","serif";
color:#76923C;
font-weight:bold;}
```

2 In our example, we'll also have to consolidate MsoQuote and QuoteChar.

Eliminating extraneous Microsoft Word page information

Microsoft Word generates some page information in the style sheet that someday might be useful, but right now is not supported by any ereaders that I know of. In addition, it's based on the physical size of the page size of the Word document, and does not vary with the size of the ereader screen, should it recognize it at all. You should just get rid of it.

```
.MsoPapDefault
{margin-bottom:10.0pt;
line-height:115%;}
@page Section1
{size:8.5in 11.0in;
margin:1.0in 1.0in 1.0in 1.0in;}
div.Section1
{page:Section1;}
```

using relative units

Perhaps the most unfortunate habit of Word's style sheets is their insistence on using absolute measurements like points and inches instead of relative measurements like ems and pixels. I recommend using ems or pixels for text sizing; both are better and more regularly supported among ereaders.

```
h1
{margin-top:0in;
margin-right:0in;
margin-bottom:100.0pt;
margin-left:0in;
text-align:center;
page-break-after:avoid;
font-size:36.0pt;
font-family:"Albertus MT","serif";
color:#76923C;
font-weight:bold;}
p.MsoQuote, li.MsoQuote, div.MsoQuote
{margin-top:0in;
margin-right:0in;
margin-bottom:10.0pt;
margin-left:.2in;
font-size:11.0pt;
font-family:"Optima","sans-serif";
color:black;
font-style:italic;}

```

On the iPad, 12pts are roughly equivalent to 16 pixels or 1em, so 100pts are about 133px, or 8.3em. The font-size of 36pt would be 48px or 3em. In general, divide the points by 12 to get the number of ems, and then multiply by 16 to get pixels.

For the `margin-left` property, note that Word has specified a margin of .2in. Since there are 72 points to the inch, you can multiply .2 x 72 (to get 14.4 points), and then proceed as above to get 1.2em or 19.2px.

Eliminating quotes for generic font styles

Microsoft erroneously adds quotation marks around generic font styles like `serif`, `sans-serif`, `fantasy`, `cursive`, and `monospace`. They must be removed.

```
font-family:"Optima", sans-serif;

```

using shortcut rules

Word specifies each of the margin settings individually and takes up a lot of room doing so. I like setting the four margin values at once, in the form `margin: top right bottom left` (start at the top and go clockwise). Each value besides 0 should have a specified unit. This is what we started with:

```
p.MsoNormal, li.MsoNormal, div.MsoNormal
{margin-top:0in;
margin-right:0in;
margin-bottom:.83em;
margin-left:0in;
```

And this is the equivalent shortcut rule:

```
p.MsoNormal, li.MsoNormal, div.MsoNormal
{margin: 0 0 .83em 0;
```

You can apply similar shortcuts for the `padding` and `border` properties.

Eliminating rules for nonexistent styles

I'm not exactly sure why but instead of creating independent classes, Word iterates each possible use of the class. For example, it lists:

```
p.MsoNormal, li.MsoNormal, div.MsoNormal
{margin: 0 0 .83em 0;
line-height:200%;
font-size:.92em;
font-family:"Optima", sans-serif;}
```

This means that the specified formatting should be applied to `p` elements with the `MsoNormal` class, `li` elements with the `MsoNormal` class, and `div` elements with the `MsoNormal` class. However, I have yet to see it actually create `li` or `div` elements in the HTML that use these classes. Instead, it's better to create a set of rules for the `.MsoNormal` class selector in order to apply the specified formatting to *all* elements with a class of `MsoNormal`. It's shorter *and* more complete. Don't forget the initial period (.) to denote that these rules apply to the class `MsoNormal`.

.MsoNormal

```
{margin: 0 0 .83em 0;
line-height:200%;
font-size:.92em;
font-family:"Optima", sans-serif;}
```

The final style sheet

Our final CSS style sheet looks like this:

.MsoNormal

```
{margin: 0 0 .83em 0;
line-height:200%;
font-size:.92em;
font-family:"Optima", sans-serif;}
```

h1

```
{margin:0 0 8.3em 0;
text-align:center;
page-break-after:avoid;
font-size:3em;
font-family:"Albertus MT", serif;
color:#76923C;
font-weight:bold;}
```

.MsoQuote

```
{margin: 0 0 .83em 0;
margin-left:1.2em;
font-size:.92em;
font-family:"Optima", sans-serif;
color:black;
font-style:italic;}
```

span.Smallcaps

```
{font-family:"Helvetica", sans-serif;
font-variant:small-caps;}
```

It's cleaner, clearer, and shorter, and will thus be easier to edit and update, and faster to download. (If you like, you can use Find/Change to get rid of the Mso prefix that precedes style names, though you'll also have to remove it from the XHTML.)

Now that the CSS style sheet file is finished, you can close it and return to the XHTML file.

Declaring the language used

Word rather anemically declares the language used in your document by adding a `lang` attribute to the `body` element.

```
<body lang=EN-US>
```

In fact, XHTML 1.1, which is the version that is supposed to be used for EPUB, supports `xml:lang` but not `lang`. Therefore, you should both substitute `xml:lang` and add quotation marks.

```
<body xml:lang="EN-US">
```

You can find the two-letter language codes at Wikipedia: http://en.wikipedia.org/wiki/List_of_ISO_639-1_codes. Further, note that the `xml:lang` attribute is not required. If you prefer, you can eliminate the attribute altogether. On the other hand, you *should* declare your ebook's language in the `content.opf` file as described in "Adding info about your book (metadata)" in Chapter 3.

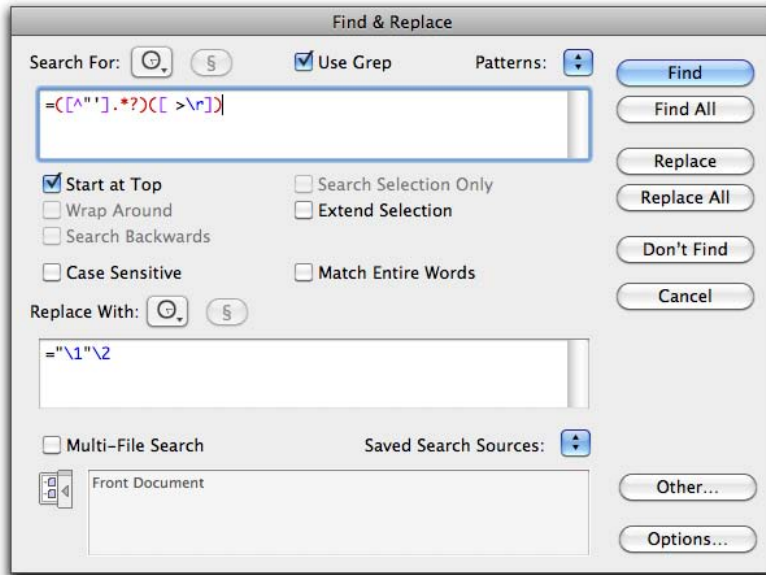
Adding quotation marks around attributes

Word only encloses multi-word attributes in quotation marks (which is all that HTML requires). Unfortunately, XHTML wants everything quoted. No exceptions.

As long as your text editor has GREP (wildcard) searches, you'll be able to automate adding quotes around every single attribute value. (I use BBEdit for these examples, but the GREP expressions will work in any text editor that supports GREP.)

- 1 Begin by opening the Find & Replace dialog box (generally with Control or Command-F).
- 2 Check the GREP box in order to use the special wildcard characters that make GREP so powerful.
- 3 Type `=(["'"].*?)([>\r])` in the Search for: box and `class="\1"\2` in the Replace with: box. This basically finds anything after an equals sign, except things that start with single or double quotes (that is, attributes that Word has already quoted) up until a space or `>`, and then replaces what it found with an equals sign and the same value within quotes. I love GREP!

The only line that this expression will catch erroneously is that first **meta** element that declares the character set, because of it's strangely worded second attribute. You should let it quote the first attribute but skip the second one.



Make sure Use Grep is checked at the top.

4 Click Find to begin the search. The text editor will highlight the first selection found.

```
<p class=MsoNormal>I see young
inherited farms, houses, barns,
easily acquired than got rid of
pasture and suckled by a wolf,
field they were called to labor
```

*In this example, BBEdit found **=MsoNormal>**.*

5 Replace the found text with the quoted version.

```
<p class="MsoNormal">I see young
inherited farms, houses, barns,
easily acquired than got rid of
pasture and suckled by a wolf,
field they were called to labor
```

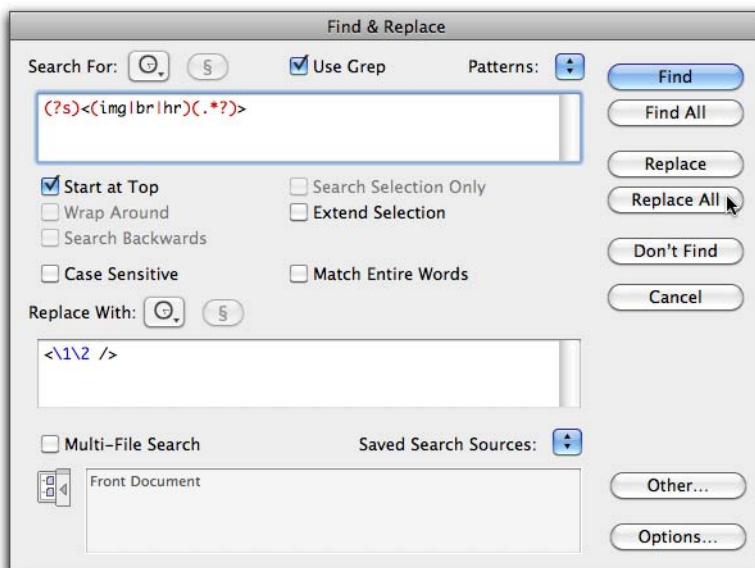
Now the MsoNormal attribute is quoted, as required by the XHTML spec.

6 Find the next occurrence, and continue adding quotes until the entire article has been searched.

Adding closing slashes to empty elements

XHTML requires that all elements have closing tags, even empty ones. That means that all the `<br>`, `<img>`, and `<hr>` tags that Word creates will need a closing slash.

- 1 Open the Find & Replace box. (Again, I'm using BBEdit for these examples, but you can use any text editor that supports GREP.)
- 2 In the Search for: box, type `(?s)<(img|br|hr)(.*?)>`. This says, "look for all of the `img`, `br`, and `hr` tags and remember what they contain."
- 3 In the Replace with: box, type `<\1\2 />`. This means, add a space and closing slash to what was found.



In BBEdit, you can save complicated sequences of Grep searches by clicking the pop-up menu next to Patterns and choosing Add....

- 4 Check the Start at Top option and then click Replace All.

Finishing cleaning up XHTML code

Word has several more unfortunate coding habits. These aren't things you can go and search for en masse. Unfortunately, you'll have to go after them one-by-one.

Here is a list of elements and attributes that Word adds to HTML files that are not compatible with XHTML 1.1 (the version required for EPUB), and thus must be eliminated:

- ▶ the `link` and `vlink` attributes in the `body` element (These have been deprecated for a very long time! Shame on you, Microsoft.)
- ▶ the `clear` attribute in `br`
- ▶ the `align` and `hspace` attributes in `img` elements
- ▶ and all attributes in `hr` must be eliminated.

In addition, you must *add* an `alt` attribute, with a quoted value, to all of your `img` elements:

```

```

Finally, note that although you are allowed to specify the height and width of images in XHTML 1.1, it's not recommended for ebooks. In particular, the iPad resizes images automatically between horizontal and vertical views only if *no size* is explicitly specified.

Using the text editor's syntax checker

As a final check, I recommend using your text editor's syntax checker to make sure that your document conforms to XHTML 1.1.

In BBEdit, you can activate the syntax checker by choosing Markup > Check > Document Syntax.

Making an EPUB

Once you've converted all of your Word documents into well-formed XHTML and CSS, you're ready to skip to Chapter 3, "Inside an EPUB file", to make it into an EPUB document.

Using InDesign to create EPUB

2

If you already own a copy of Adobe InDesign CS5 (or CS4 or even CS3), you've got probably the most powerful EPUB file creator to date in your hands. If you already know how to use it, even better. In this chapter, rather than give you a tutorial on InDesign, I'll show you how to use the features that you're already acquainted with in order to create the best EPUB file possible.

In this chapter you'll learn:

- ▶ About using InDesign for EPUB

and how to:

- ▶ Apply styles to an ebook
- ▶ Add images
- ▶ Add links
- ▶ Create a navigational TOC
- ▶ Add metadata to your ebook
- ▶ Export EPUB from InDesign

If you need an introduction—or even a refresher course—to using InDesign, I recommend getting a copy of *Real World InDesign*, by Olav Martin Kvern and David Blatner, also published by Peachpit Press. In this chapter I'll focus on those aspects of InDesign that deal specifically with exporting EPUB files.

About using InDesign for EPUB

It's a curious thing using InDesign to create an EPUB document. InDesign is a complex, industrial-strength piece of software often used to create complicated but static layouts for print. EPUB, on the other hand, focuses much more on the flow of text; the *order* of the text is much more important than the position of any particular word. The trick lies in not getting swept up by InDesign's myriad possibilities, but rather in using the power of its more basic tools in order to automate what can otherwise be a rather tedious process.

One of the most important differences between typical InDesign documents and those destined for EPUB is the emphasis on *pages*. It's no coincidence that InDesign's mother's name was PageMaker, whose ability to let designers choose the exact location of text and graphics on a page spawned the entire desktop publishing revolution.

InDesign even has *Master Pages*. Master Pages in InDesign are used as a template for the rest of the document. So, if you want to create a header at the top of every page, you would create it just once on the Master Page. If you want to place a page number on every page, you can add a special code on the Master Page that InDesign automatically updates for each page in your document. You also might use the Master Pages to set the guides and column margins. Master Pages make sense in InDesign because you are designing a static document that exists on a specific size of paper.

None of that translates to EPUB. EPUB is all about *flow*. It's designed to expand into bigger screens, adapt to larger or smaller text, and generally take advantage of the space allotted, whatever its size. Because EPUB documents are not predivided into pages, InDesign's emphasis on pages is less useful. Indeed, nothing that exists on an InDesign Master Page is exported with the EPUB file nor is the specific location of a bit of text on a given page ever passed on to EPUB.

You might even conclude that InDesign and EPUB are at such cross-purposes that it doesn't make sense to use the one to create the other. But actually, once you can forget about pages, and focus on the flow and the formatting of the text, InDesign becomes a great tool for creating EPUB documents. It has extremely powerful styles, excellent search and replace tools, including GREP, robust meta-information

collection, and automated table of content generation, all of which will facilitate the creation of your EPUB file. And if your book is already in InDesign, it makes even more sense to take advantage of this power.

One file or many?

You can either organize your book by separating chapters into individual files or by creating a single InDesign file that contains your entire book. The method you choose depends on the version of InDesign you have. InDesign CS5 has a new feature that can automatically divide a book into individual files for each chapter upon export to EPUB. This is important for two reasons. First, each file in an EPUB should be less than 300K (not counting embedded images). Second, individual files in an EPUB document always begin on a new page, and there is currently no other way to create page breaks that work in the iBooks app. In addition, CS5 currently has an unpleasant bug that breaks links across documents upon export to EPUB, but there is a script that can solve the problem for books contained in a single InDesign document. Therefore, if you are using CS5, it's probably better to keep your entire book in a single document.

If the size of the book makes housing it in a single file unwieldy, or if you have CS4 or earlier, you may prefer to divide your book into chapters to ensure that each chapter begins on a fresh page in the exported EPUB file.

If you decide to create each chapter in its own InDesign file, it's a good idea to create a template to base them on. One good way to create a template is to place a representative chunk of text from your book and then create the styles until you're satisfied with how it looks. If you're creating both print and electronic editions, you can design master items on Master Pages that will automatically appear on all the pages in the document, but note that master items are not exported to EPUB.

Once the styles are created in the template file, erase the sample text, and save the file as a template (choose InDesign template in the Save As box). Create new, empty files based on the template by choosing File > Open and then selecting the template, and then paste a chapter's worth of text into each individual InDesign file.

Once you've created all the individual files, create an InDesign book to hold them by choosing File > New > Book, and then clicking the plus sign at the bottom of the book palette.

Styles in InDesign

However you decide to create your book—whether it be in a single document or in a book of documents—styles should be paramount in your mind. InDesign converts your file into EPUB by creating XHTML/CSS approximations of the styles that you have applied in InDesign.

Because of the different natures of print and ebook, and because of the limitations of InDesign itself, not everything you choose in the Paragraph Style Options box will be exported to EPUB. Here are the features that InDesign will export properly from its Paragraph and Character Style options.

Style	Notes
font, including weight, style, and size	converted at 1em per 12pts; make sure name is correct; if font name is “Medium Italics” or “Bold Condensed”, the simpler style is applied: Italic or Bold respectively
leading	converted with respect to font size, (see “Controlling spacing” in Chapter 4)
case	small caps, and CS5 adds support for all caps
lines	underline, strikethrough, but not rules above or below
justification: right, left, center, and justified	but not Justified Left, Justified Center, Justified Right, or Full Justify
spacing (left indent, first line indent, right indent, space before, space after)	converted with respect to font size, (see “Controlling spacing” in Chapter 4)

That leaves a lot that is not yet supported (even in CS5), including rules above and below, kerning, tracking, horizontal and vertical scale, superscript, no break, baseline shift, hyphenation options, keep options, and spanning columns. Almost all of these style features can be re-created with CSS so there’s no good reason

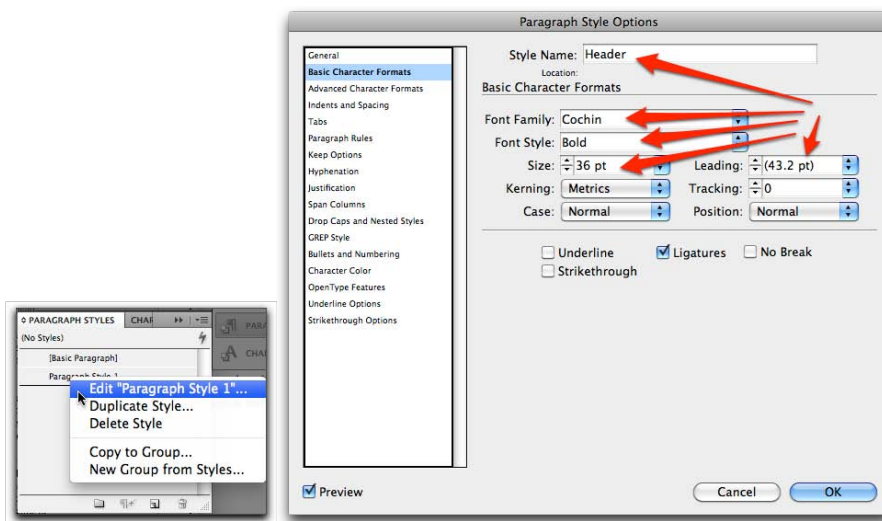
why InDesign doesn't support them. Perhaps it will in the future. The solution right now is to add these effects manually, as described in Chapter 4, "Advanced EPUB Formatting".

In addition, if you apply Bullets or Numbering to a style, you can convert those paragraphs to unordered or ordered lists in the EPUB document, but InDesign exports the styles erroneously, and thus all of the style characteristics (apart from the mere fact of being a list) must be edited manually.

Creating styles

I like to create styles in the Paragraph Styles palette.

- 1 Click the Create New Style icon at the bottom of the palette, and InDesign places in the palette a Paragraph Style 1 with the current attributes of the paragraph that contains the cursor. If you like, you can format the paragraph first so that those characteristics are automatically added to the style. Or you can do as I do, and choose the characteristics from the Paragraph Styles Option box that appears when you right-click the new style and choose Edit "Paragraph Style 1".



Create a new style and then right click the style and choose Edit from the pop-up menu. Choose the style characteristics in the Paragraph Style Options box and click OK.

- 2 To create the Header style for all of the chapter names in our book, type "Header" in the Style Name box at the top of the Paragraph Style Options box.

3 We want our header to be centered, bold, in a larger size, and in the Cochin font. Choose these characteristics from the various menus and click OK.

Next, you'd create the rest of the styles for the book. If you are creating the style sheet for the first time, you may want to experiment with the sample text and apply different formatting until you're satisfied. For this project, you can download the styles from the website.

Perhaps the most important thing to remember when you're formatting an ebook with InDesign CS4 and earlier is that InDesign will only export formatting that has been applied *with styles*. All local formatting is completely ignored. Although CS5 has been updated so that it does export local formatting, styles save you so much time that the tips in this section will probably still be worth your while, even though they don't seem as essential.

Applying the main Body style

When you first place your text, it is automatically styled with the default [Basic Paragraph] style. It's a good idea to reformat it with a defined style, like Body, particularly so that you can see the spacing between paragraphs and get an idea about what your pages will look like. Most InDesign experts recommend against using the default [Basic Paragraph] style permanently, myself included.

Economy

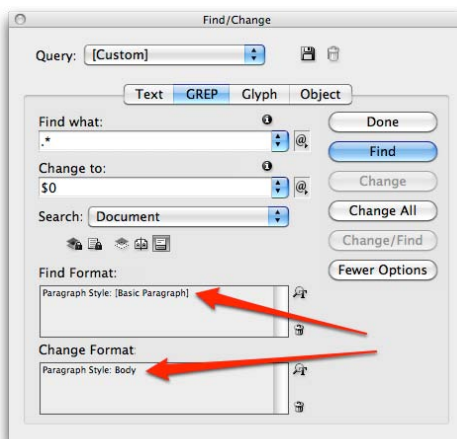
WHEN I wrote the following pages, or rather the bulk of them, I lived alone, in the woods, a mile from any neighbor, in a house which I had built myself, on the shore of Walden Pond, in Concord, Massachusetts, and earned my living by the labor of my hands only. I lived there two years and two months. At present I am a sojourner in civilized life again.

I should not obtrude my affairs so much on the notice of my readers if very particular inquiries had not been made by my townsmen concerning my mode of life, which some would call impertinent, though they do not appear to me at all impertinent, but, considering the circumstances, very natural and pertinent. Some have asked what I got to eat; if I did not feel lonesome; if I was not afraid; and the like. Others have been curious to learn what portion of my income I devoted to charitable purposes; and some, who have large families, how many poor children I maintained. I will therefore ask those of my readers who feel no particular interest in me to pardon me if I undertake to answer some of these questions in this book. In most books, the *I*, or first person, is omitted; in this it will be retained; that, in respect to egotism, is the main difference. We commonly do not remember that it is, after all, always the first person that is speaking. I should not talk so much about myself if there were anybody else whom I knew as well. Unfortunately, I am confined to this theme by the narrowness of my experience. Moreover, I, on my side, require of every writer, first or last, a simple and sincere account of his own life, and not merely what he has heard of other men's lives; some such account as he would send to his kindred from a distant land; for if he has lived sincerely, it must have been in a distant land to me. Perhaps these pages are more particularly addressed to poor students. As for the rest of my readers, they will accept such portions as apply to them. I trust that none will stretch the seams in putting on the coat, for it may do good service to him whom it fits.

I would fain say something, not so much concerning the Chinese and Sandwich

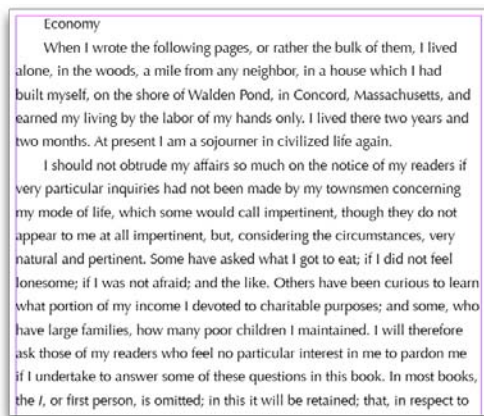
The [Basic Paragraph] style is InDesign's default style, and is applied automatically when you import a document to any unstyled paragraphs.

- 1 Choose Edit > Find/Change and click the GREP tab to begin. Then type `.*` in order to find anything and everything. In the Change to: box, type `$0`. This means you want to apply the changes to everything that you find.
- 2 Click the Find Format: box, and choose the [Basic Paragraph] style from the Paragraph Style menu.
- 3 Click the Change Format: box and choose the Body style from the Paragraph Style menu.



Change all Basic Paragraph styled text to Body.

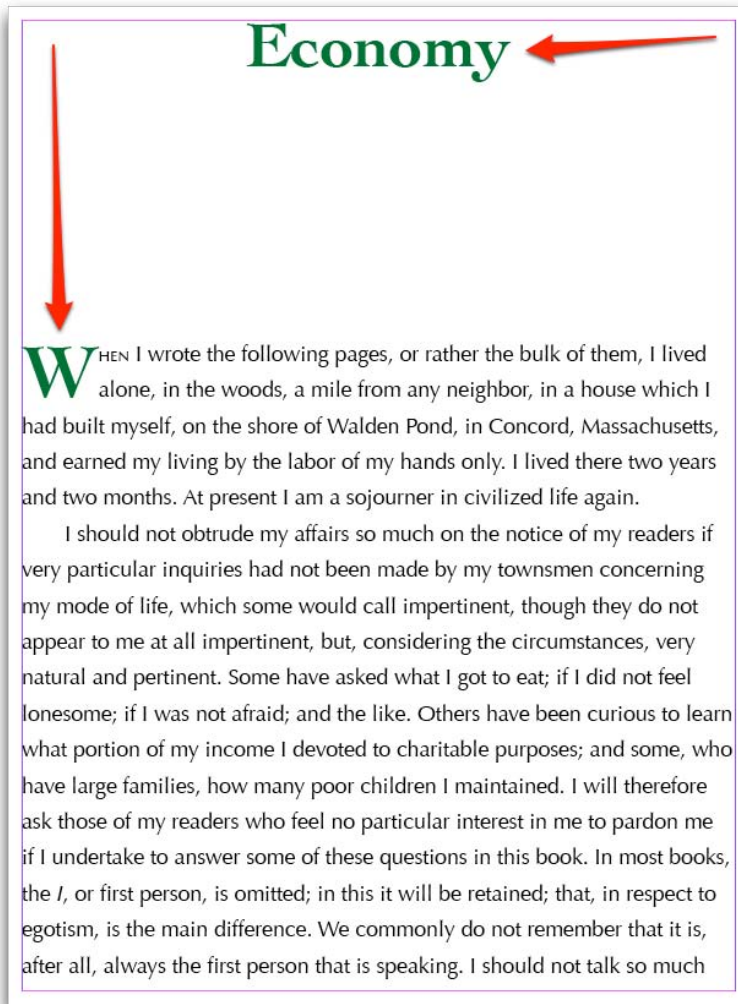
- 4 Click Change All to apply the Body style to everything previously styled with [Basic Paragraph].



The Body style applies the Optima font, as well as the size and leading that I've chosen to all the paragraphs previously styled with [Basic Paragraph].

Applying headers, quotes, and other special styles

There's no real secret to applying styles to headers, quotes, and other special styles throughout your book. It's just a question of flying through your document, and applying styles as required. The trick that will save the most time is to apply the most used style first, as we did in the previous section. Since there are many fewer headers than body paragraphs, applying a few particular styles shouldn't take too long.



Here I've applied the Header style to the chapter name and the First Para style to the first paragraph.

» Tips «

You can add keyboard shortcuts to InDesign paragraph and character styles in order to make them easier to apply, although unfortunately only with the number keys on the numeric keypad. David Blatner has developed a set of tools that extends this feature (<http://blatnertools.com>).

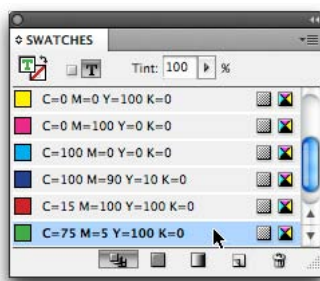
You can search for common terms in your special paragraphs to jump to them more quickly. For example, if all of your chapter names begin “Chapter...”, you could search for the word *Chapter* and apply the style.

Replacing local formatting with styles

InDesign CS4 and earlier won’t export any local formatting. Even if you’re using CS5, which does export local formatting, it’s still a better idea to format with styles. Styles make it much easier to update the formatting at a later date, and much easier to edit the resulting CSS files. In addition, it often makes for a smaller EPUB file, which makes for shorter download times and happier readers.

What is local formatting?

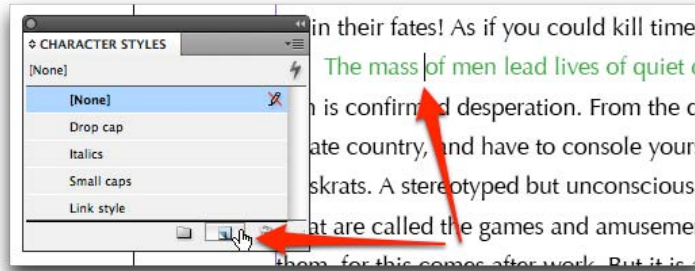
Local formatting happens when you select some text in your document and choose a specific kind of formatting, say green coloring, directly from the toolbar, and not by applying a character or paragraph style.



Applying local formatting involves selecting some text, and choosing formatting directly from the toolbar or Character or Paragraph palettes.

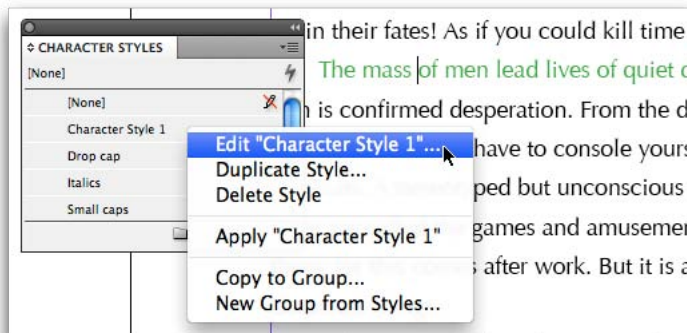
The first step to replacing local formatting with styles is to create the style that contains the desired formatting. InDesign makes this particularly easy.

- 1 First, place the cursor in the formatted text and open the Character Palette. Click the Create new style icon at the bottom of the palette.



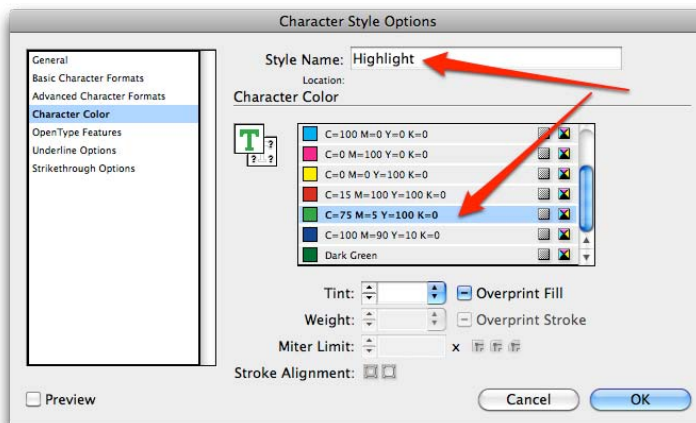
Place the cursor in the text (here: The mass of men...), and then click the Create new style icon at the bottom of the palette.

InDesign automatically creates a style with whatever characteristics were applied to the formatted text that contains the cursor.



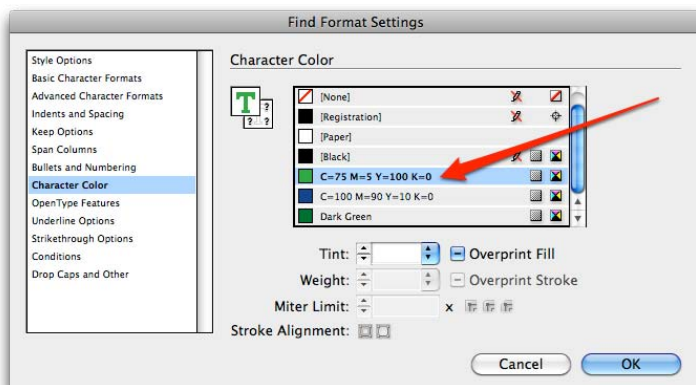
Right-click the new style to edit its name.

- 2 All you have to do is give the style a name. To do so, right or control-click the new style and choose Edit Character Style... in the fly-out menu. The Character Style Options box appears. If you click the appropriate options in the left-hand list—in this case, Character Color—you'll see that InDesign has already specified the formatting to be applied, using the formatted text as a guide. Enter the name for the new style in the Style Name box.



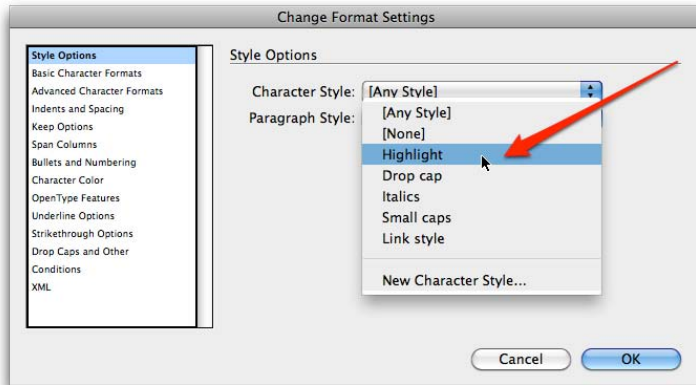
If you put the cursor in the formatted text before creating the style, the only thing you'll have to do is change the name of the style in the Style Name box.

- 3 Next, we'll use InDesign's Find/Change box to replace all the local green formatting with styled Highlight formatting. Of course, it won't look any different. Choose Edit > Find/Change to begin and click the GREP tab if it's not already selected.
- 4 In the Find what: box, type `.*` to match any text. In the Change to: box, type `$0`, to replace the text with whatever you find, that is, the content of the text won't be changed.
- 5 Next to the Find Format: box (click More Options if you don't see it), click the garbage icon to start with a fresh slate. Then click in the Find Format: box, click Character Color at left, and then choose the green color in the Character Color box. Click OK.



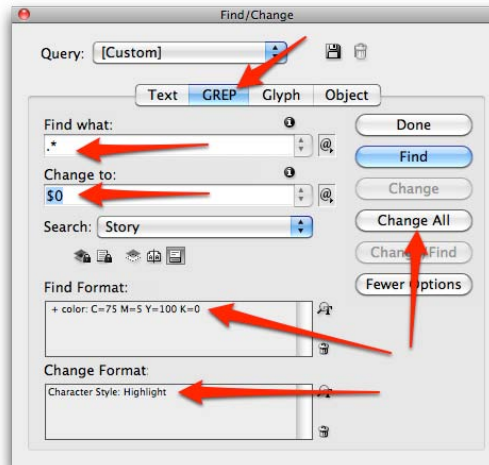
Choose the local format you want to search for.

- 6 Click the garbage icon next to the Change Format: box to start afresh here too. Then click in the Change Format: box, and choose the Highlight character style that you created previously. Click OK.



Choose the Character Style that we created earlier with which to replace the local formatting.

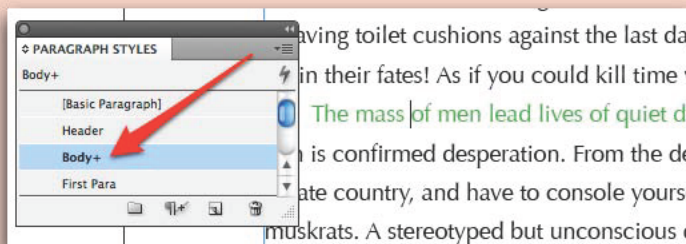
- 7 Finally, click Find to make sure your search is working correctly. Click Change to test the first found text. As long as it works properly, you can click Change All to change the rest of the local color formatting to styled Highlight formatting.



Click Change All once you've entered all the search criteria.

» Tips «

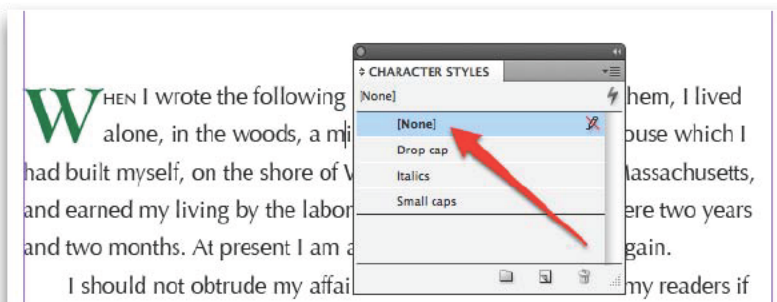
While you can't immediately tell by a given piece of text's appearance if it has been locally formatted or formatted with styles, you can get InDesign to tell you. Place the cursor within the text in question and then open the Paragraph Styles palette. The paragraph style applied to that text, even if it's just the default [Basic Paragraph], will be highlighted. If the text has any additional local formatting, InDesign will display a plus sign (+) next to the style's name.



With the cursor in the formatted text, a plus sign appears next to the paragraph style. This means local formatting has been applied.

Drop Caps and Nested Styles

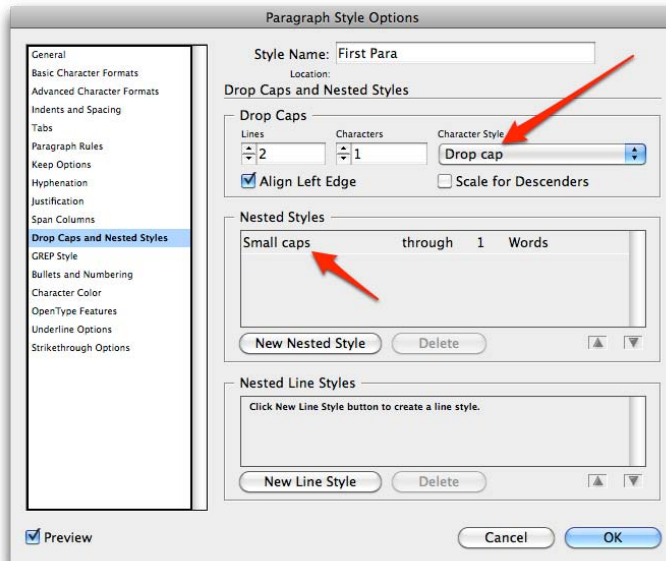
There is a special situation where you might think you've already applied styles but where InDesign will ignore them upon export. That is the case with Drop Caps and Nested Styles. Although InDesign will export the paragraph style, it does not—even in CS5—export the styles associated via the Drop Caps and Nested Styles tab.



In our example, the First Para paragraph style has a drop cap applied to the first letter and a nested small caps style applied to the first word. Note that no character style seems to have been applied.

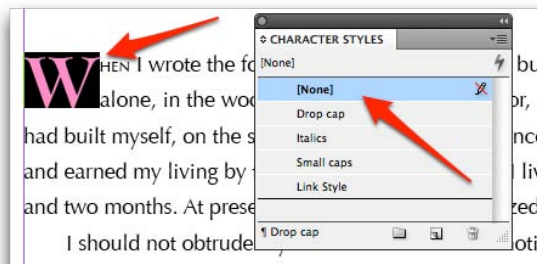
Instead, you have to apply the same styles called in Drop Caps and Nested Styles directly. I'll show you how.

- 1 Open the First Para style in our *Walden* document (by right or control-clicking its name in the Paragraph Styles palette and choosing "Edit First Para..."). Click the Drop Caps and Nested Styles option in the left list and notice that the First Para style includes a nested style which applies the Small caps character style to the first word in the paragraph, and the Drop cap style to the first letter.



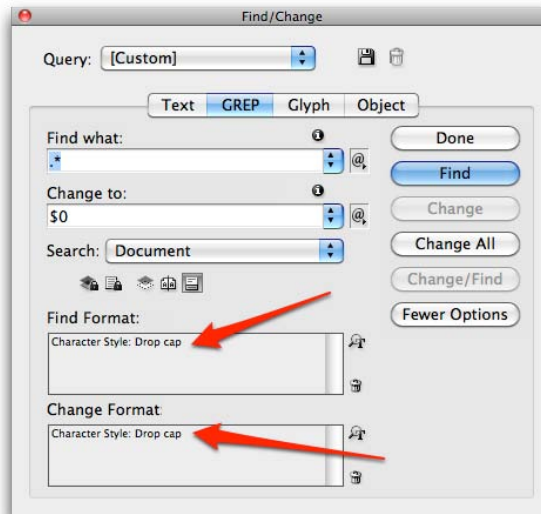
There are two character styles applied by InDesign to this Paragraph Style: Drop cap and Small caps.

- 2 Now click in the first word of a First Para paragraph. Notice that None is displayed in the Character Styles palette.



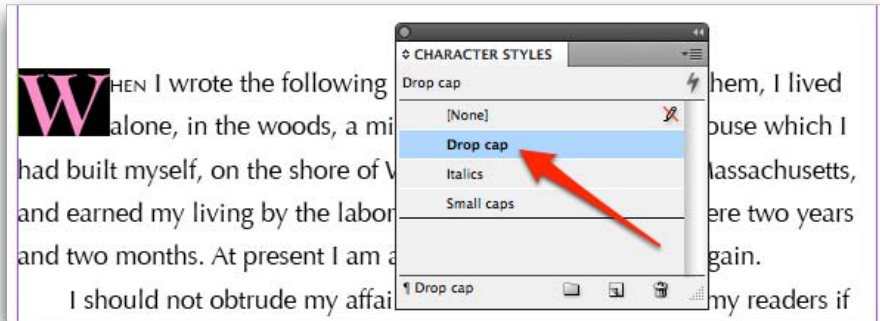
Because the character style is automatically applied through the paragraph style, it doesn't show up in the Character Styles palette (and is not exported to EPUB).

- 3 Close out of this box and open the Find/Change dialog box (choose Edit > Find/Change). Make sure GREP is selected.
- 4 In the Find what: box, type .* to make sure you search all the text. In the Change to: box, type \$0 to apply the changes to whatever is found.
- 5 Click the Find Format: box and choose the character style that is applied for the drop cap. In this example, it is Drop cap.
- 6 Click the Change Format: box and choose the same character style so that it is applied directly to the text and not just through the paragraph style.



It seems very nonintuitive to find something that's a Drop cap and apply the Drop cap style to it. The difference is that the first was applied through nesting styles, and the second is applied directly.

7 Click the Find button to find the first occurrence and make sure you've got the Find/Change criteria properly set up. Click Change to update the first occurrence. Notice that when you select the first character in a First Para paragraph, the Drop cap character style is now displayed in the Character Style palette, indicating that it has been applied directly. This will ensure that InDesign exports the style information correctly to the EPUB file.



Now when you select the first character in a First Para paragraph, you see the Drop cap character style has been applied. It won't look any different in InDesign, but it will export properly.

8 Repeat the same steps to apply the Small caps style to the first word in the First Para paragraph.

Tip

If you have all of the documents in your book open, you can choose All Documents in the Find/Change box to apply the changes to every document in your book.

Add images

Before we look at the nitty-gritty of adding an image, it's important to see how InDesign looks at the content of a document before exporting it to EPUB. It starts with the first frame on the first page and exports the entire contents of that frame, no matter where that text is placed. It doesn't matter if that frame spans five other pages, or if it is interrupted with other text frames. Indeed, it doesn't even matter if all of the text is placed at all. InDesign simply takes all of the content in the frame and exports it in the order in which it appears in that frame. It then goes on to the second text frame that it finds and does the same thing.

But how does InDesign determine which frame is *first*? Well, it begins at the highest, leftmost frame. Not just the highest, but the highest one that's furthest to the left. It may be that there's a higher frame that begins farther to the right, but that won't be the first one. It then looks to see if there are any other frames at the same left position lower down. If so, it goes through those. It then proceeds rightward until it finds the next highest frame. Again, if there are two at the same distance left, it grabs the topmost one first, and moves down through all the frames that begin at the same horizontal position. And so on.

If you are used to designing books with InDesign destined for print, you may have grouped images with captions in their own frames and then placed the whole group on top of the main content before applying text wrap. Because of the way InDesign orders exported content as described above, that won't work for EPUBs. Here is an example of an image and a caption, grouped together, placed next to a text frame, with text wrap enabled so that the text flows around the image and its caption.



An image and caption in their own frames, with a distinct text frame wrapped around them.

The problem here is that upon export to EPUB, InDesign will first deal with all of the text in the larger text frame (since it begins above the image), and then it will export the image and its caption. In this example, the image will end up appearing at the *end of the chapter*, which is not what we want at all.

The solution is to paste the image *into the principal flow* of text right before or after the paragraph to which it should be attached, that is, as an *inline graphic*. In addition, if you paste the image *before* the related paragraph, although InDesign won't automatically apply text wrap in an exported EPUB, it'll make it easier for you to create text wrap manually, as described in "Working with images" in Chapter 4.



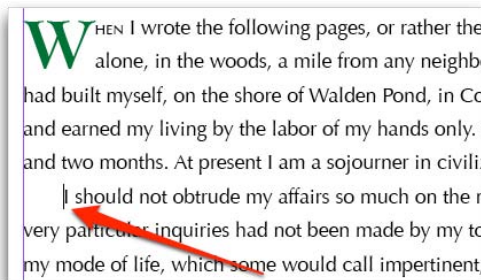
Tip

It's a really good idea to name your images as simply as possible, with no spaces or punctuation other than - or _. This will keep file names as simple and compatible as possible upon export.

Placing an image

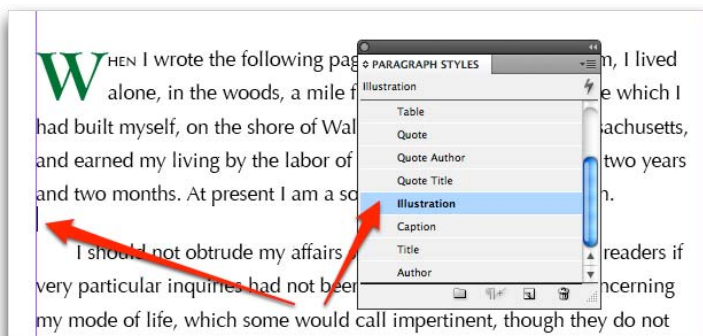
The most important thing to remember when placing images in an InDesign document that will be exported to EPUB is that everything must be in a single flow of content. You don't want to create any independent frames or blocks for images. Rather, you must place images right in the main flow of your book.

- 1 Place the cursor in the main flow of your book, right before or after the paragraph to which the illustration belongs. In our example, we'll place the illustration before the second paragraph, the one that begins "I should not obtrude my affairs..." so that we can later make the text wrap around the image.



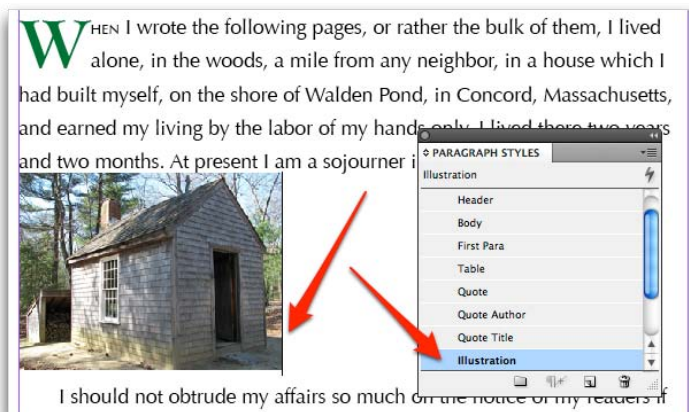
Place the cursor in front of the paragraph to which the illustration belongs.

- 2 Press the return key to create a separate paragraph for the illustration and then press the up arrow to return the cursor to the illustration paragraph.
- 3 Apply the Illustration style from the Paragraph Styles palette. (If you have trouble finding it, be sure that you've downloaded the project styles from the website.) Note that the Illustration style in our example has Auto leading applied so that the paragraph expands to whatever height the image has.



With the cursor in the new empty paragraph, apply the Illustration paragraph style.

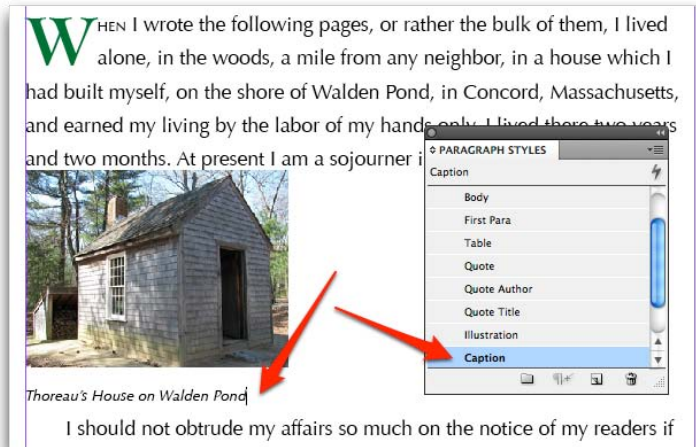
- 4 Choose File > Place and then select the desired image. For this example, we're going to place *Thoreau_House.jpg*. Click OK to place the image.



The image appears before the paragraph to which it is related.

- 5 If desired, resize, crop, shear, and rotate the image until you are satisfied with the results. You can either choose to have InDesign take your edits into account upon export, or ignore the modifications and export the image with its original size and other characteristics.

- 6 If desired, press the Return key to create a new caption after the illustration.
- 7 Apply the Caption style from the Paragraph Styles palette.
- 8 Type *Thoreau's house on Walden Pond*.



*The image is now in its own paragraph before the paragraph that it concerns.
The caption is set with the Caption style.*

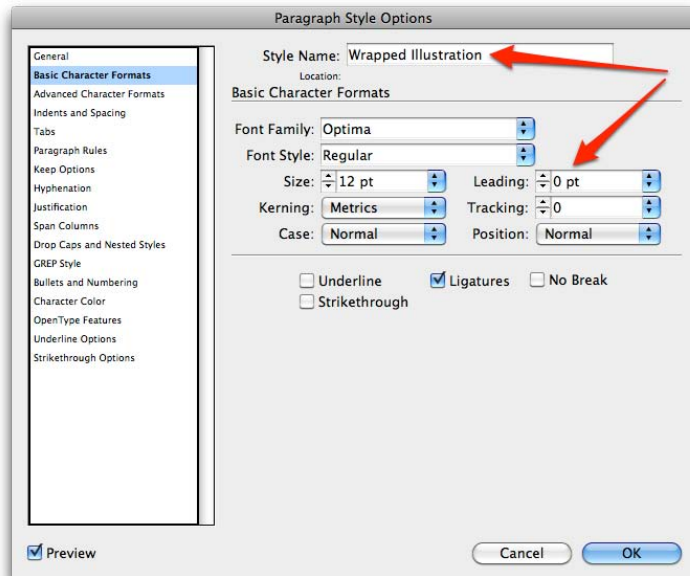
Creating text wrap within the flow

Although InDesign can't yet export text wrapped around images, you can adjust the CSS code later to achieve that effect. While it's fine to leave the images as described above, placed *before* the paragraphs that will flow around them, if you plan to create a print or PDF version of your book from the InDesign file, you'll want to wrap text around images right in InDesign as well. We've already seen that it won't help to place images in a separate frame, but there are techniques you can use to wrap text around images within the flow. You'll still have to manually adjust the CSS once you export the EPUB, but at least you'll have wrapped text in all versions of your book once you do.

Wrapping text around just an image

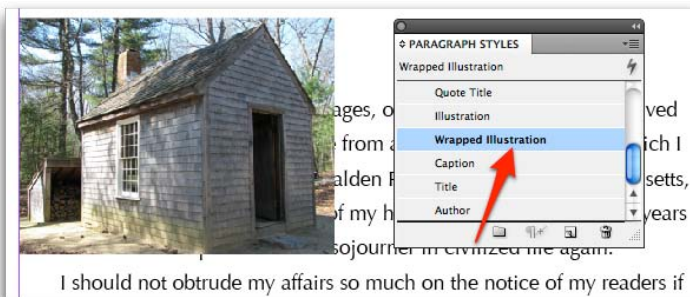
Wrapping text around an image by itself is simpler than wrapping text around an image and caption.

- 1 If it's just an image by itself that you want to wrap text around, you can begin by placing it as described in the previous section.
- 2 In order for the image not to displace the entire paragraph that follows it, its leading must be 0. I have created the Wrapped Illustration style, which has 0 leading but is otherwise identical to Illustration. If you open the Paragraph Style Options box for Wrapped Illustration, you'll see that the value for the leading is 0. Apply the Wrapped Illustration style to the paragraph.



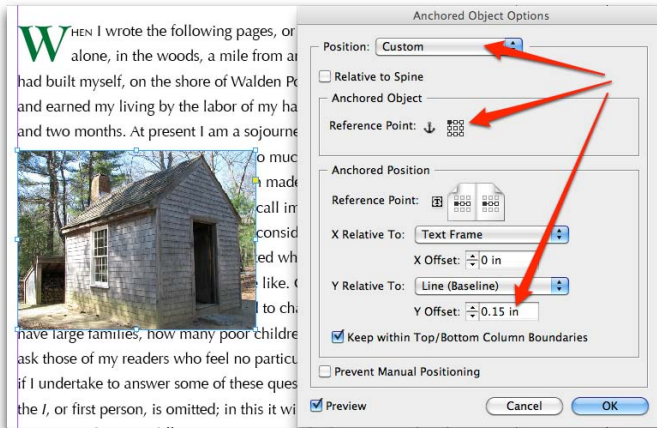
When you use a leading value of 0, the entire Illustration paragraph slides above the preceding paragraph.

You'll see that the illustration now overlaps the preceding paragraph.



- 3 Next, select the image with the selection tool and choose Object > Anchored Object > Options....

- 4 In the Anchored Object Options dialog box, choose Custom next to Position, and then click the upper-left box next to Reference Point in order to align the top of the image with the top-left corner of the paragraph.



When the Illustration paragraph's leading is set to 0, you can adjust the position of the illustration in the Anchored Object Options dialog box.

- 5 Next, add a small value to the Y Offset box to align the image with the top of the text line. This amount will vary according to the leading in your text paragraph. In this example, I have used .15".

- 6 Finally, click OK to see the changes.

- 7 Finally, we set the text wrap by choosing Window > Text Wrap to display the Text Wrap palette, and clicking the first Text Wrap option. I've also added .1" of space between the illustration and the wrapped text.



Choose a Text Wrap option and add a small amount of offset so that the illustration doesn't butt right up next to the text.

Because this image is in the main flow of the document, it will be exported to EPUB almost exactly like the previous one. The difference is that the print and PDF versions will show the text wrap.

Wrapping text around an image and a caption

Of course, many times you will want to wrap text around an image *and* its caption. Again, while InDesign doesn't automatically create the correct CSS so that such a layout works directly in EPUB, you can set the document up to display properly for print and PDF, and then adjust the CSS manually so that it displays properly in EPUB as well (as described in “Wrapping text around images” in Chapter 4.)

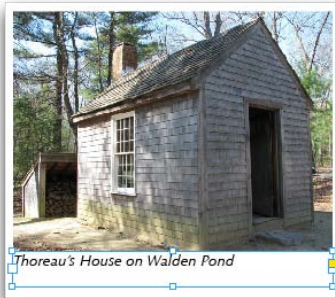
In order for an image and caption to have text wrapped around them, they have to be grouped. But you can't group an image and text if they're already part of a larger text flow. The trick is to group them *before* you add them to the text flow.

- 1 Place the image on the pasteboard, outside of the main text flow.



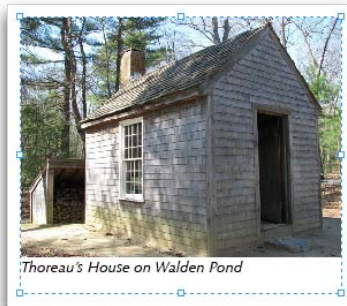
Place the image independently, out on the pasteboard.

- 2 Create the caption in its own text frame and situate it in the desired position near the image.



Add a caption in its own frame and place it beneath the image.

- 3 Now, group the image and its caption by selecting both items and choosing Object > Group.



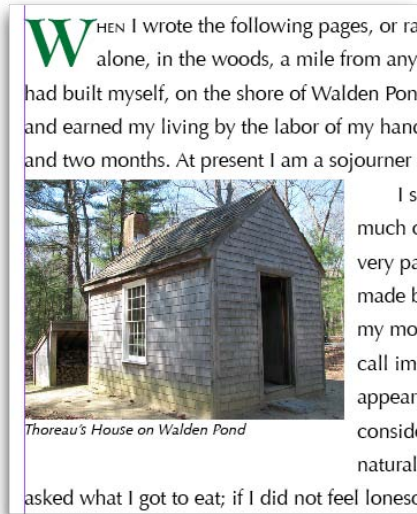
A group is displayed with a dotted frame.

- 4 To insert the new group into the text flow, so that it stays with the paragraph to which it belongs, you must cut it from the pasteboard and paste it into the text flow. Start by selecting the group and choosing Edit > Cut.
- 5 Then place the insertion point before the paragraph which will wrap around the image and caption, and create an empty paragraph.
- 6 Paste the grouped image and caption into that paragraph.
- 7 Choose the Wrapped Illustration style to format the group's paragraph with no leading as in the previous example.

8 As in the previous example, choose the group and then choose Options > Anchored Object > Options... and set the Custom, Reference Point, and Y Offset.

9 As in the previous example, choose the group and in the Text Wrap palette, choose a text wrap option and a right offset.

Now the image and the caption have text wrapped around them:



The grouped image and caption are part of the main text flow and so will be exported in the proper order.

InDesign will create a special **div** element with **group** class for the grouped elements and will export them in the proper order within the main flow. It doesn't yet create the proper CSS for the elements to have the following paragraph wrapped around automatically, but adjusting the CSS manually to create the effect is pretty straightforward. (See "Working with images" in Chapter 4 for more details.)

Add links

One of the advantages ebooks have over print books is the ability to insert links which enable your readers to jump between one section of the book and another. You can use InDesign to create and manage such links. InDesign can create both hyperlinks and cross-references. The principal difference is that a hyperlink adds a link to existing text whereas a cross-reference grabs text from the destination and uses that as the text for the link. That difference also explains why creating a hyperlink is a two-step process: you create a hyperlink destination and then link to it, while a cross-reference is all-in-one: you reference the destination that will be used to generate the current link.

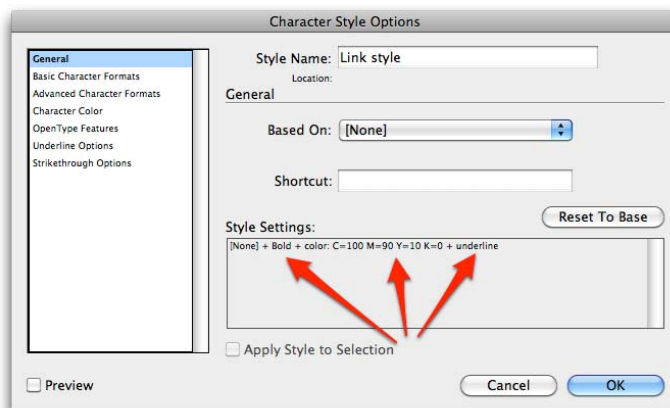
Tip

InDesign CS5 has an unfortunate and serious bug that creates broken links in the exported EPUB file from hyperlinks and cross-references that go from earlier chapters to later ones in multiple-chapter books. Curiously, internal links and backward referencing links continue to work. If you have links in your InDesign document that you want to export to EPUB, I recommend either avoiding CS5, or using a single file for your entire book. I will update the status of this bug on my website.

Creating a style for links

Before you create the links themselves, you may want to create a style for the links so that they stand out in a distinctive and easily identifiable way and readers know that they're supposed to click on them.

- 1** Create a new character style for links by clicking the Create new style icon at the bottom of the Character Styles palette.
- 2** Specify the desired characteristics for the link style—I'm going with the very standard bold, blue, and underlined in this example—give the style a name, and click OK. I've called the style "Link style".



Choose the desired characteristics for links in your ebook.

Hyperlinks

An InDesign hyperlink is much like a link in a web page. The text on the web page can say anything—“click here”, “see this page”, or whatever—and when you click on it, you arrive at a destination that is coded into the web page, but that isn’t necessarily visible before you click.

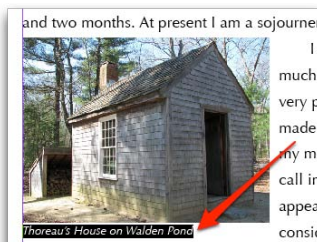
A hyperlink has two parts: what you click on—InDesign calls this a *hyperlink*—and where you arrive—in InDesign parlance, this is a *hyperlink destination*. InDesign also lets you create links to URLs, email addresses, and particular pages of a book, although the latter option is not useful for ebooks, whose pagination can fluctuate.

Creating a hyperlink destination

We’ll begin by creating a hyperlink destination on the first page of our document, where we inserted the image of Thoreau’s house.

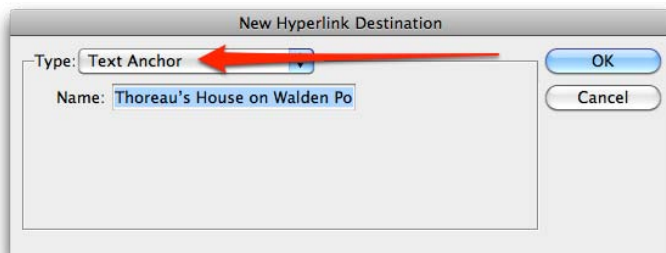
1

Select the text or image that should appear when the reader clicks the link.



We want the picture of the house to appear when the reader clicks the link, so we’ve selected the caption to be the hyperlink destination.

- 2 If necessary, choose Window > Interactive > Hyperlinks to display the Hyperlinks palette. Choose New Hyperlink Destination from the flyout menu.
- 3 In the New Hyperlink Destination box that appears, you'll see that the text you selected in step 1 appears automatically as the Name of the new destination. That's fine. Choose Text Anchor from the Type menu, and click OK.

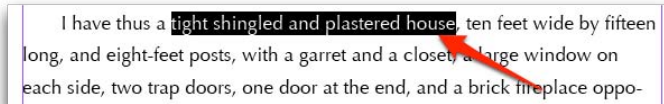


Choose Text Anchor in the Type menu. The text you selected in step 1 is automatically used in the Name field to identify the destination.

Creating a new hyperlink

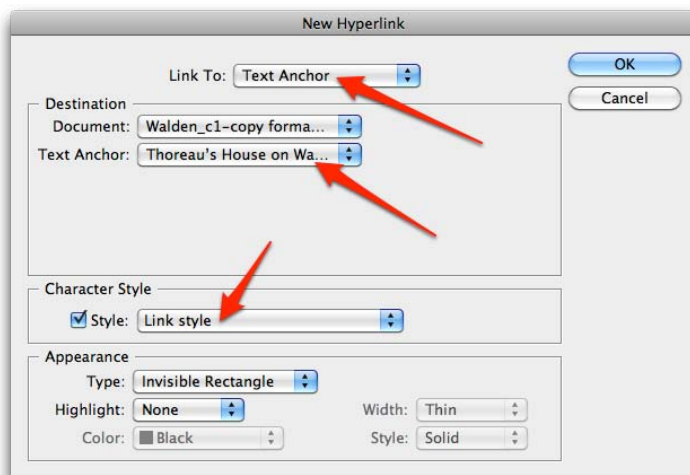
Once you have defined the destination for your link, you're ready to create the link itself and style it with the Link style you created earlier.

- 1 Select the text in your document that you wish to create a link from.



Select the text that the reader will click.

- 2 Choose New Hyperlink from the flyout menu in the Hyperlinks palette. In the dialog box that appears, choose Text Anchor from the Link to menu at the top.
- 3 Next, from the Text Anchor menu, choose the hyperlink destination that you created in the previous section.
- 4 In the Character Style section, check the Style box and choose the style that you created for links.
- 5 Under Appearance, be sure to choose Invisible Rectangle for Type so that InDesign doesn't enclose your link in an ugly frame.



The New Hyperlink box lets you choose what kind of link you're creating, where it should go, and what it should look like.

Cross-references

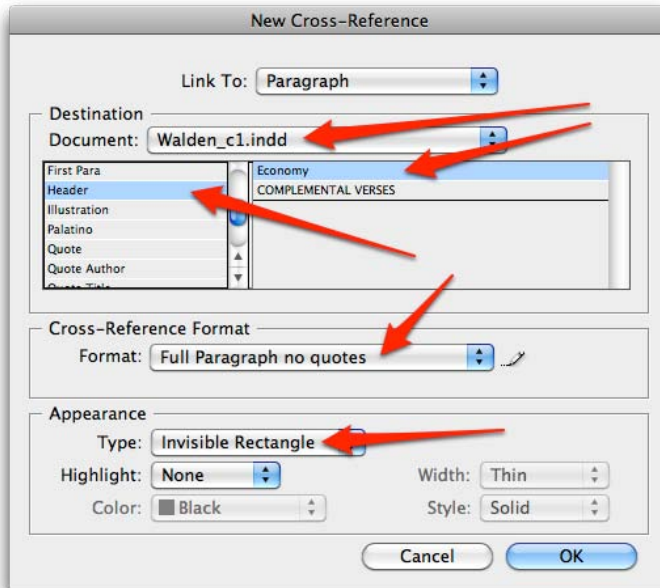
The main difference between a cross-reference and a hyperlink is that the content of a cross-reference—that is, the clickable text—comes directly from the content of its destination, and is automatically updated when the destination text changes. In printed books, cross-references are used to point to chapter names or pages, say, see *Chapter 3, “Life in the Amazon”*. Since they are generated automatically from the destination text, they are guaranteed to match the destination text. Although cross-references aren't dynamic when printed—they simply point you to a related section or chapter—InDesign can convert them into clickable links.

Cross-references are ideal for creating clickable tables of contents in the body of an ebook. While InDesign can facilitate creating a navigational table of contents, as described in the next section, it won't generate regular tables of contents that you can export with your EPUB. Instead, you can create these in-text tables of contents with cross-references.

Creating a cross-reference

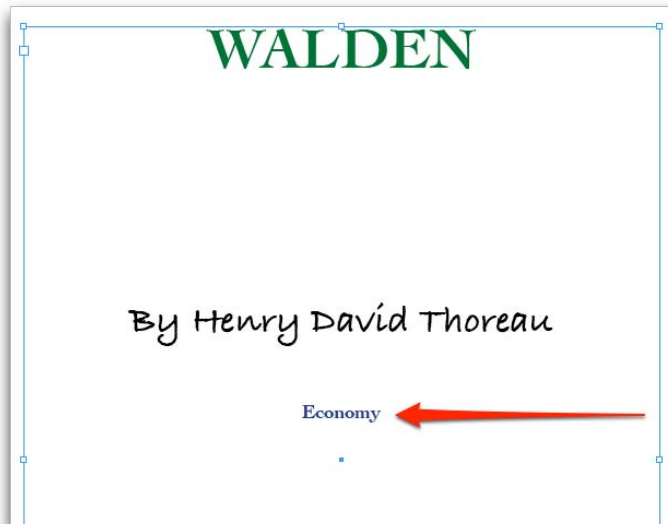
- 1 Place the cursor where the text from the cross-reference should appear. Sometimes that will mean constructing the rest of the sentence that contains the cross-reference. Since you'll be inserting the text from the destination, you'll want to make sure that it will fit into your current context.

- 2 Choose Window > Interactive > Hyperlinks to display the Hyperlinks palette, if necessary. Choose Insert Cross-Reference from the flyout menu.
- 3 Choose the document and paragraph style that contains the destination you want to cross-reference.

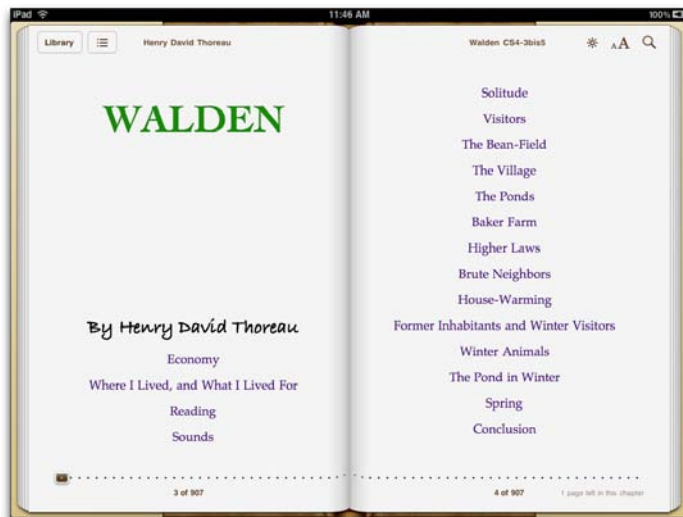


Once you choose the document that contains the destination in the Document menu, you'll see the paragraph styles in the left pane.

- 4 Make a selection in the Format menu to determine what text should be extracted from the destination to generate the cross-reference. Click the pencil to create or edit formats (or to apply a character style to a format).
- 5 Choose Invisible Rectangle in the Type menu under Appearance in the lower part of the window to keep InDesign from placing an ugly box around your links.
- 6 Click OK to generate the cross-reference.



The word Economy came directly from the selected paragraph; I didn't type it.

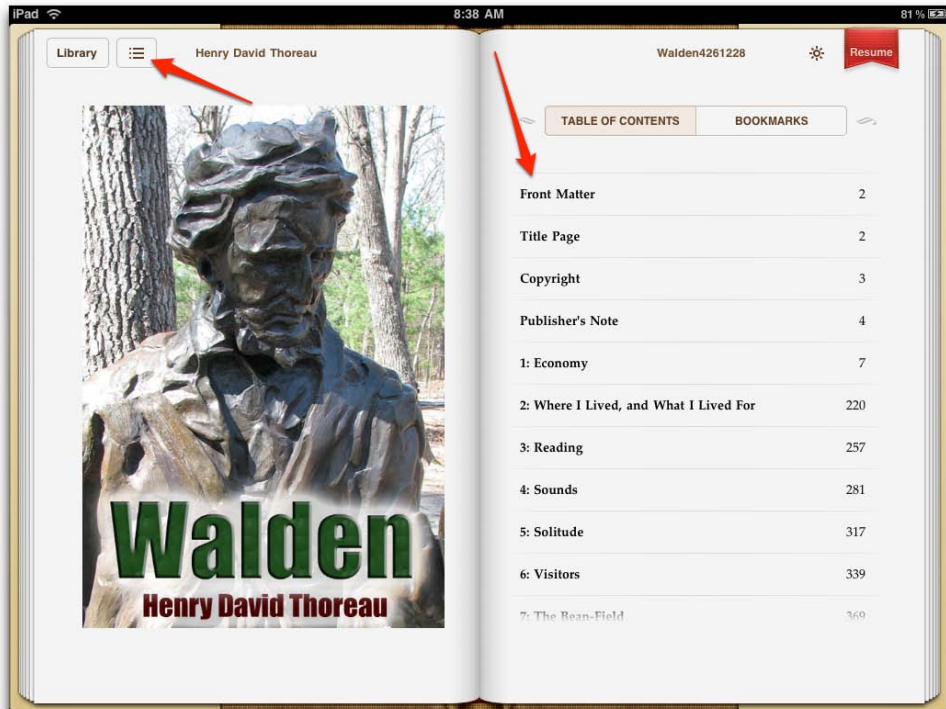


Once the cross-reference based table of contents is completed and exported it becomes a clickable portal to the rest of the book.

There is some controversy about whether you should create an in-book table of contents (like the one shown here) *as well as* an ereader-generated navigational table of contents (as described in the next section). First of all, only the former is optional. Second, some people think the more things you have at the beginning of the book between the front page and the content, the more you will frustrate your reader. I think giving your reader multiple ways to navigate the book is a good idea.

Create a navigational TOC

Most ereaders generate a *navigational table of contents* that is accessible from any point in the book, much like the menu option on a DVD, and that enables the reader to jump to whichever chapter they're interested in from any point in the ebook. InDesign can generate the necessary code for such a navigational table of contents.



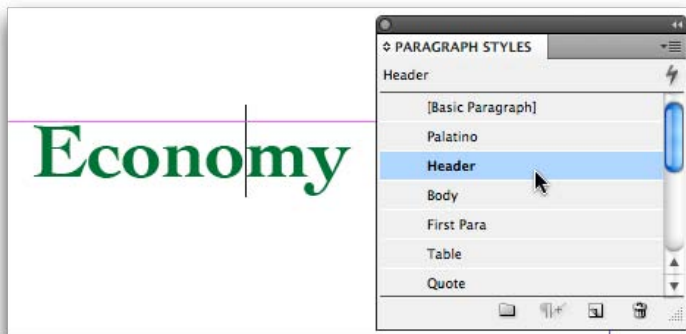
The iPad's table of contents menu appears when you click the icon next to the Library button.

Preparing your book in order to create the navigational TOC

Everything in InDesign is based on styles, and tables of contents are no exception. To begin, therefore, make sure that everything that should be contained in your table of contents is properly styled. In our example, there is a header at the beginning of each chapter. Our table of contents menu will be based on those chapter title headers.

You could also create subentries in the table of contents menu for subsections, or even illustrations or quotations. It's up to you. For this example, we'll keep it simple and just use the chapter titles.

- 1 Make sure your table of content entries are properly styled. In our example, that means every chapter title should be styled as a Header.



Place the cursor in the header paragraph for each chapter and make sure the Header style from the Paragraph Styles palette is selected.

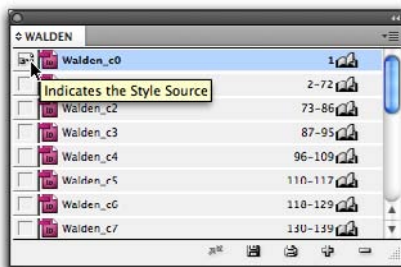
Creating a Table of Contents Style

Once you're sure your table of contents entries are properly styled, you create a *Table of Contents Style*. I find the terminology a bit obtuse here, so let me clarify. There are the paragraph styles, like Header, shown above that are applied to the text in the headers themselves. Then there are the styles that are applied to those items *when they appear* in a Table of Contents, these often have names like *toc-header 1*, and *toc-header 2*.

But a Table of Contents Style is neither of these. It's the collection of settings, or pre-sets, that define a *particular kind* of Table of Contents. Let's go create one so you can see what I mean.

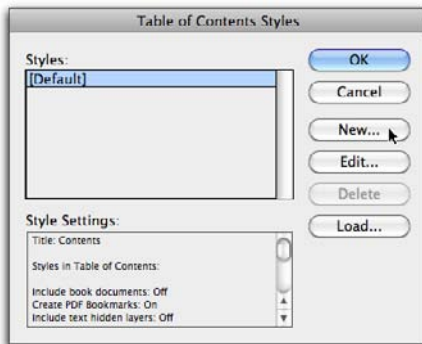
First, it's important to create the Table of Contents Style in the document which is the *Style Source* for your entire book. This is generally, but not always, the first document in your book.

- 1 Click the box to the left of the name of the first document in your book in the Book panel to designate it as the Style Source. (If you only have one document, it's automatically considered the Style Source.)



Click to the left of the first document to make it the Style Source.

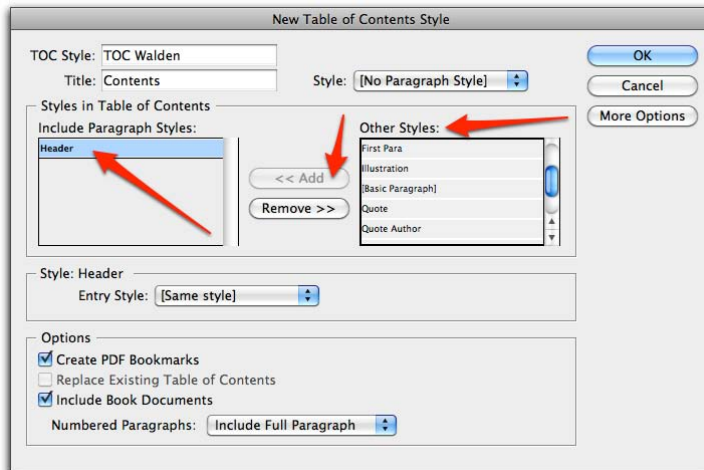
- 2 Next, open the Style Source document that you just designated.
- 3 Choose Layout > Table of Contents Styles... You'll see a dialog box that lets you create a new Table of Contents Style. Click New.



Click New in the Table of Contents Styles dialog box.

- 4 In the New Table of Contents Style box that appears, choose a name for the TOC Style and type it in the box at the upper left.

5 In the Other Styles box at the right, double-click the styled paragraphs that should be included in the table of contents. In our example, since the Header style denotes the name of the chapter, that's what we want included in our table of contents.



Choose a name for your TOC Style in the top-left. Then, most importantly, add the Paragraph Styles that should be included in the TOC in the center sections.

6 If your documents are organized in a book, make sure Include Book Documents is selected at the bottom of the dialog box.

7 Click OK to finish.

Now that you have created a Table of Contents Style, InDesign will be able to generate the code necessary for the navigational table of contents when you export to EPUB.

Add metadata to your ebook

Metadata is information *about* your ebook that can be used to help readers find your book, know who wrote it, who published it, and when, among many other things. You must, at the very least, add the title of your work to the metadata. Otherwise, some ereaders won't be able to read it. Indeed, it's a good idea to be as complete as possible with the metadata.

1 Make sure you're in the Style Source document for your book by viewing the Book palette and ensuring that the icon to the left of the document's name is checked. (A book contained in a single InDesign file is always its own Style Source.) Metadata, like the TOC Style, is created in your book's Style Source document.

2 Choose File > File Info to open the File Information box.

The screenshot shows the 'File Information for Walden_c0.indd' dialog box. It has tabs for Description, IPTC, IPTC Extension, Camera Data, Video Data, Audio Data, and Mobile SWF. The Description tab is active, showing fields for Document Title (Walden), Author (Henry David Thoreau), Author Title, Description (A tale of Thoreau's two years living in the woods near Walden Pond), Rating (5 stars), Description Writer, and Keywords. Below these fields is a note: 'Semicolons or commas can be used to separate multiple values'. The Copyright section includes Copyright Status (Public Domain), Copyright Notice, and Copyright Info URL. At the bottom, there is a 'Powered By xmp' logo and buttons for Preferences, Import..., Cancel, and OK.

The most important metadata fields are the Document Title and the Author.

- 3 Enter the title of your book in the Document Title field. Without this information, some ereaders won't be able to open your book.
- 4 If desired, enter the Author, Description, Copyright Status, and other metadata information.
- 5 Click OK to save the information in the Style Source document.

All of the metadata will be exported with the EPUB.

You'll notice that there's no field for entering information about the Publisher or the unique identifier in the File Information box. The unique identifier is a required value for the EPUB, but the Publisher information is also quite important.

InDesign will let you add these two pieces of information in the Export EPUB dialog box that we'll get to in the next section.

You can also add additional metadata by editing the *content.opf* file once the EPUB has been created. For more information, see "Writing the content.opf file" in Chapter 3.



InDesign CS4 and later do not add the required date to your EPUB document. In order to pass EpubCheck validation (and thus have your ebook in Apple's iBookstore), you'll have to add that information by hand in the *content.opf* file (see "Writing the content.opf file" in Chapter 3) or use the script written by Teus de Jong (<http://bit.ly/dfV19F>).

Export EPUB from InDesign

Once you have formatted your book with styles, placed your images as desired, created a Table of Contents Style, and added metadata, you're ready to export the book from InDesign in EPUB format.

InDesign added a number of significant improvements to its export to EPUB function with version CS5. Unfortunately, CS5 also has a major bug that creates broken links in multiple-file books. If that is a deal breaker for you, you may want to stay with CS4 until that bug is fixed. Thankfully, InDesign CS4 does a pretty good job exporting to EPUB.

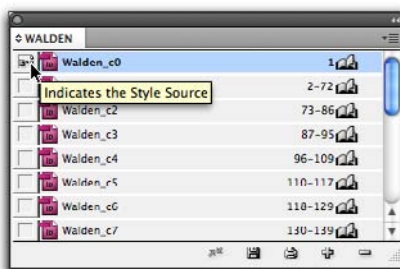
For those who have a single EPUB file or no links, I'll show you how to use CS5's new features as well.

Tip

If you are using the same InDesign file for both print and EPUB and you generally convert your RGB images into CMYK before sending your files to the printer, you should be sure to export your EPUB file *before converting the images*. That way, the EPUB will use the brighter, native RGB images, which will display perfectly well in ereaders that support color.

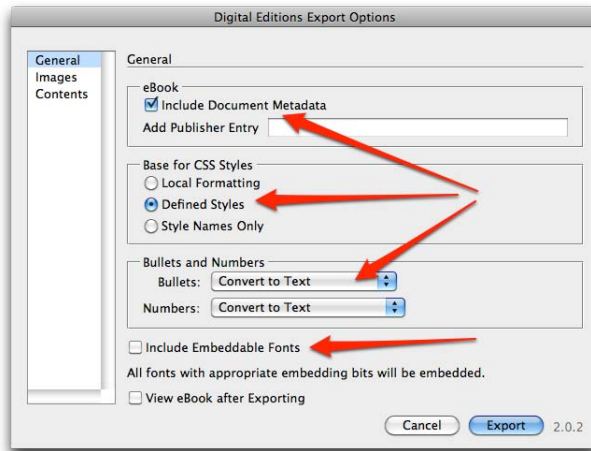
Exporting EPUB from InDesign CS4

- 1 Open the Book palette and make sure that the proper Style Source document is designated as such. In our example, we've selected the first document in the book as our Style Source, and it is here where we have created a Table of Contents Style and added metadata. If you don't have the Style Source document selected, neither the metadata nor the table of contents menu will be properly exported.



Make sure the proper document (in our case the first one) is selected as the Style Source for your book.

- 2 Choose Export Book for Digital Editions from the fly-out menu on the Book palette. If you choose the option from the File menu, only the current document will be exported, not the entire book.
- 3 Give the EPUB document a name and save it to your folder of choice. Once you click Save, the Digital Editions Export Options box will appear. There are three principal tabs: General, Images, and Contents.
- 4 In the General tab, make sure the Include Document Metadata option is selected, and add the Publisher name, if desired. The first time you export a document, InDesign generates a unique identifier. The Unique Identifier is a required element of the EPUB.

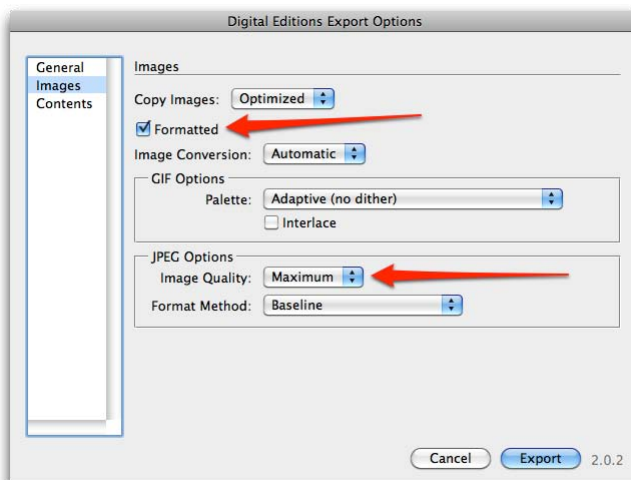


The General tab controls metadata, CSS, lists, and fonts.

Check Defined Styles in the Base for CSS Styles section to have InDesign export a CSS style sheet with (some of) the characteristics from the paragraph and character styles you defined for the document. You'll see how InDesign generates the CSS in "How InDesign writes CSS" in Chapter 4.

I don't recommend checking Include Embeddable Fonts because InDesign doesn't yet allow you to include the SVG fonts that are required by iPad. I prefer to embed fonts manually, if at all. For more details, see "Fonts in your ebook" in Chapter 4.

5 Click the Images tab to view the Image options. If you choose Formatted, the images placed in your InDesign document will be exported with their current size, cropping, rotation, border, and background, but at 72 dpi. That means if the image measures 4 inches by 2 inches on your InDesign page, it will be 288 pixels by 144 pixels. And on the iPad, which has a resolution of 132 pixels per inch, it will measure 2.2" x 1.1".



Use Formatted to export the images as they appear in InDesign, and uncheck it to export them as they appeared in their original files.

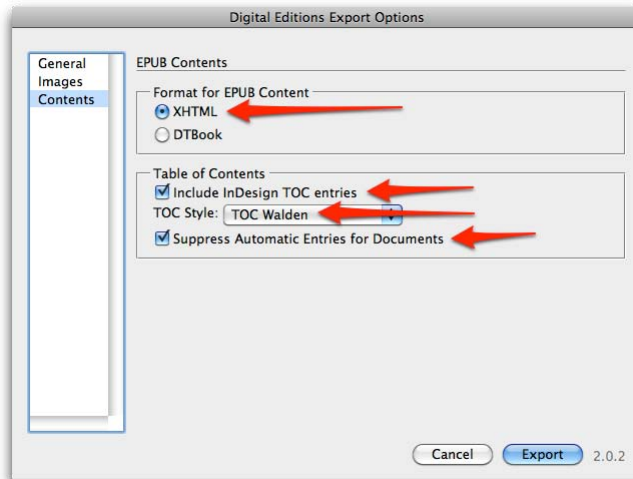
If you don't check Formatted, the image will maintain its original size. If it started out as 400 pixels x 300 pixels, that's what it will have in the EPUB document. InDesign also changes the resolution to 72 dpi, but that should not affect its display size, as iPad takes into account the pixel size, irrespective of the resolution.

You can also choose whether to force InDesign to export as GIF or JPEG in the Image Conversion menu, and the level of quality in those exported images.

6 Click Contents to get to the last tab. This is where we'll generate the navigational table of contents. Make sure XHTML is the Format chosen for EPUB Content at the top.

7 Next, make sure Include InDesign TOC entries is checked, and then choose the TOC Style that we created earlier. If your Table of Contents Style does not appear in the pop-up menu, it's probably because you didn't choose the Style Source document before going to export, assuming, of course, you created one.

Check the Suppress Automatic Entries for Documents so that InDesign does not add links to your navigational table of contents that point to the file name of each of your InDesign documents.



Be sure to choose the Table of Contents Style from the TOC Style menu so that InDesign exports the code necessary to generate the table of contents menu.

8 Finally, click Export and the file is created.

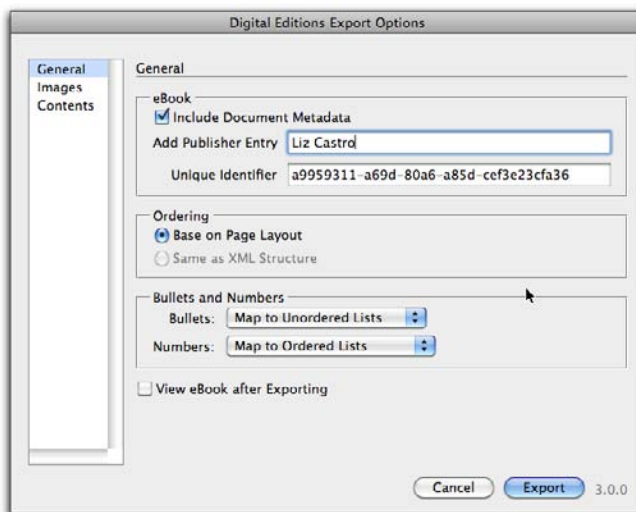


On my computer, the default application for opening EPUB documents is Adobe Digital Editions, and therefore, all EPUB documents will have the ADE icon. Yours may look different.

Exporting EPUB from InDesign CS5

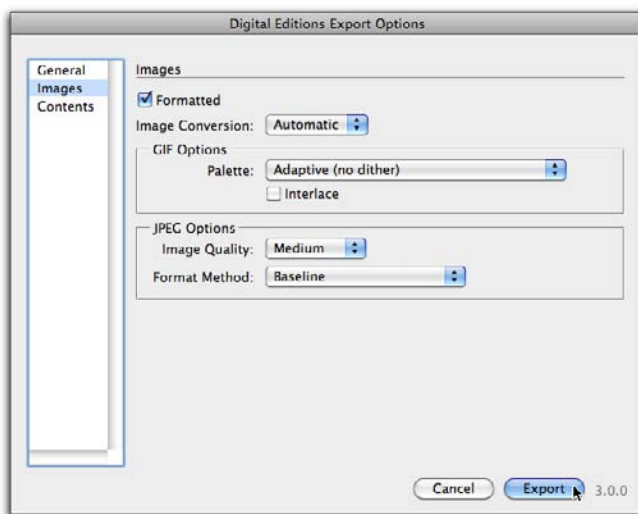
If your document has no links (hyperlinks or cross-references), or if it is contained in a single document, you're probably safe to use InDesign CS5 to export your EPUB documents.

- 1 Open the InDesign document that you wish to export to EPUB and choose File > Export Book to EPUB. If you have organized your files in a book, you'll want to select Export Book to EPUB from the Book panel's flyout menu.
- 2 Give the EPUB document a name and save it to your folder of choice. Once you click Save, the Digital Editions Export Options box will appear. There are three principal tabs: General, Images, and Contents.
- 3 In the General tab, make sure the Include Document Metadata option is selected, and add the Publisher name, if desired. The first time you export a document, InDesign generates a unique identifier, but you can add your own, for example, the book's ISBN, if you prefer. The Unique Identifier is a required element of the EPUB. The rest of the options are pretty straightforward.



The Unique Identifier field will be empty the first time you export an EPUB from this document. If you leave it empty, InDesign will generate a unique identifier for you, as well as save it for future exports.

4 Click Images to get to the next tab. If you choose Formatted, the images placed in your InDesign document will be exported with their current size, cropping, rotation, border, and background, but at 72 dpi. That means if the image measures 4 inches by 2 inches on your InDesign page, it will be 288 pixels by 144 pixels. And on the iPad, which has a resolution of 132 pixels per inch, it will measure 2.2" x 1.1".



Use Formatted to export the images as they appear in InDesign, and uncheck it to export them as they appeared in their original files.

If you don't check Formatted, the image will maintain its original size and format. If it started out as 400 pixels x 300 pixels, that's what it will have in the EPUB document. InDesign also changes the resolution to 72 dpi, but that should not affect its display size, as iPad takes into account the pixel size, irrespective of the resolution.

You can also choose whether to force InDesign to export as GIF or JPEG in the Image Conversion menu, and the level of quality in those exported images.

5 Click Contents to get to the last tab. This is where we'll generate the Table of Contents and the CSS file. Make sure XHTML is the Format chosen for EPUB Content at the top.

6 Next, make sure Include InDesign TOC entries is checked, and then choose the TOC Style that we created earlier. If your Table of Contents Style does not appear in the pop-up menu, it's probably because you didn't choose the Style Source document before going to export. Check the Suppress Automatic Entries for Documents so that InDesign doesn't add a link to the document's file name in the table of contents menu.

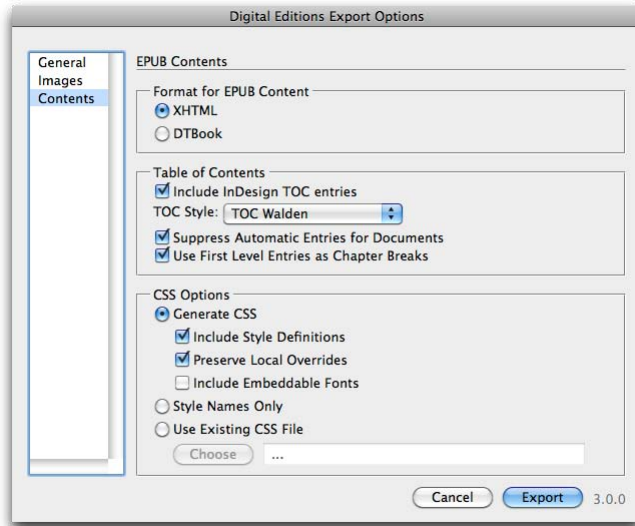
7 If your document contains multiple chapters that you would like to force to begin on a new page in the EPUB document, check Use First Level Entries as Chapter Breaks. Otherwise, each chapter will directly follow the preceding one.

8 In the CSS Options section, click the Generate CSS button. Check Include Style Definitions to have InDesign export the characteristics of each style along with the style name. Note that if you don't check the Include Style Definitions box, you'll get the same results as if you checked the Style Names Only radio button: a list of style names with no defined characteristics.

If you check Preserve Local Overrides (only available in CS5), InDesign will create a set of formatting rules to cover each instance of local formatting and apply it to a span with class `no-style-override-n`, where *n* is an integer. I think it's much more useful to apply all formatting with named styles so you don't have a lot of these no-style-override classes in your final EPUB document. Nevertheless, I still check this option just in case I missed any local formatting that has not been properly styled. At the very least, I can use search and replace later to find the local formatting in the exported EPUB file.

I don't check Include Embeddable Fonts because InDesign doesn't yet allow you to include the SVG fonts that are required by iPad. I prefer to embed fonts manually. For more details, see "Fonts in your ebook" in Chapter 4.

The Use Existing CSS File option is useful if you have already created a CSS style sheet that matches the style names used in your book. For example, once you adjust the CSS as shown in the next two chapters, you could use that polished CSS file for future projects.



The Use First Level Entries as Chapter Breaks is new to CS5.

9

Finally, click Export and the file is created.



*On my computer, the default application for opening EPUB documents is Adobe Digital Editions, and therefore, all EPUB documents will have the ADE icon.
Yours may look different.*

Inside an EPUB file

3

Although few people will write EPUB files completely from scratch—computers are much better suited to the task than humans—computers still aren’t quite up to doing it on their own. The best approach, until there is a robust, complete solution for creating EPUBs, is to use a variety of tools, typically InDesign or Word to generate the initial code, and then a text editor to polish it up.

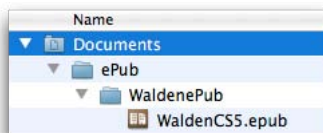
In this chapter you’ll learn how to:

- ▶ unzip an EPUB file from InDesign or elsewhere and look inside it,
- ▶ recognize and create the individual files within an EPUB document, including the files that contain the text and images in the ebook, the file that serves as the basis for the navigational table of contents and the file that lists an ebook’s contents
- ▶ designate an image file as the cover of your ebook,
- ▶ compress the individual files properly into (or back into) an EPUB document,
- ▶ check the EPUB file’s validity with EpubCheck.

Unzipping an EPUB

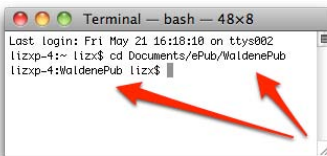
If you've created your EPUB with InDesign, or if you're curious about how someone else created an EPUB, you'll have to unzip it in order to get at the XHTML and CSS files and images that it contains.

- 1 Create a new directory for the EPUB file and place the EPUB file in it. (When you unzip the EPUB, there are a lot of files and it's nice if they're not all spread out on your desktop.)



I've placed the WaldenCS5.epub file in the WaldenePub directory, which is contained in the ePub directory inside the Documents folder.

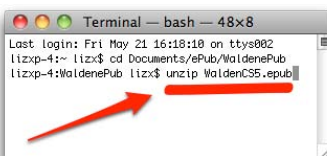
- 2 Open Terminal on the Mac. (You can use Zip on Windows—<http://bit.ly/aUXJLG>.)
- 3 Navigate to the directory that contains the EPUB file. For example, in this example, I'll have to type `cd Documents/ePub/WaldenePub`.



Once you navigate to the desired directory, its name appears after the colon in the command line.

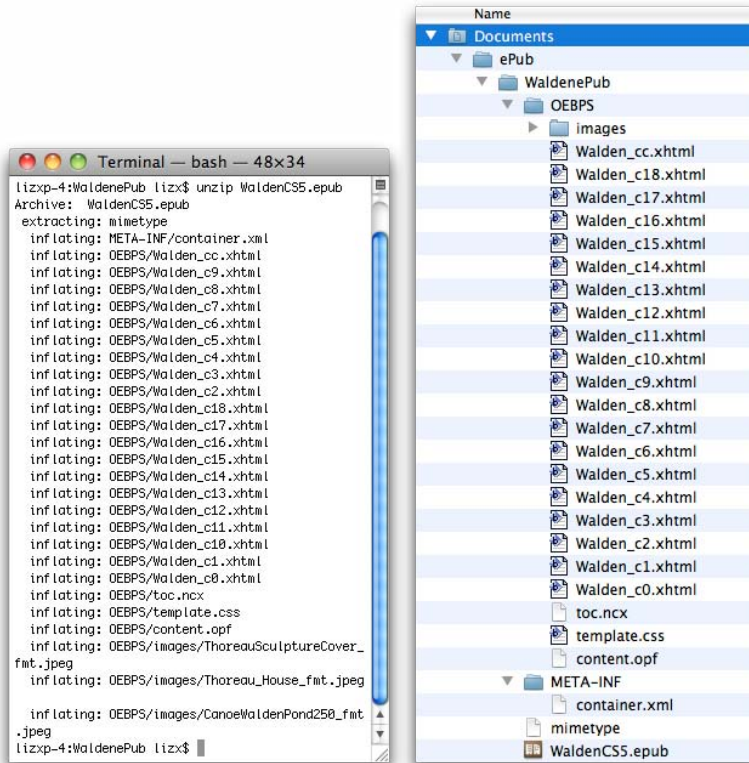
- 4 Once you're in the directory that contains the EPUB file that you want to unzip, simply type `unzip` followed by the filename and extension.

`unzip WaldenCS5.epub`



By navigating to the directory first, you don't have to type the path to the file.

5 Press Enter. The directory will fill with the XHTML, CSS, and other XML files, as well as any images and fonts that the EPUB contained.



*Once you hit enter after typing the **unzip** command, you'll see the contents of the EPUB file unfurl before your eyes (left) and in the Finder (right).*

» Tips «

I like to collect the three elements: the OEBPS and META-INF folders and the mimetype file and drop them into their own folder. Then, I can navigate to that folder, make changes, zip a new EPUB, and test, all without disturbing the original WaldenCS5.epub document that came from InDesign.

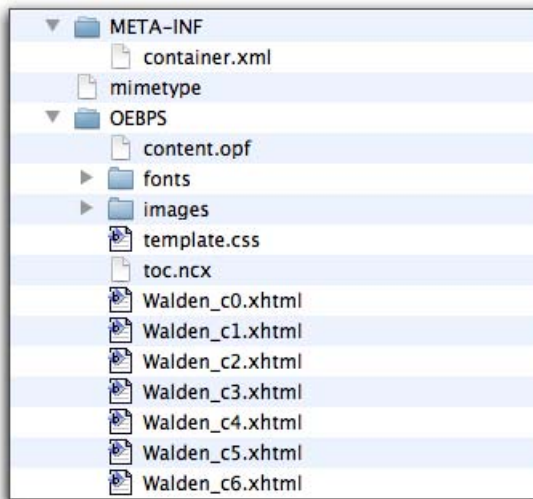
There is a program called Springy that can look inside an EPUB file and let you edit the contents of the XHTML and CSS files without unzipping anything. You can find it at <http://springyarchiver.com>.

You can also unzip an EPUB file by changing its extension to .zip and double-clicking, though you'll need Terminal or similar to zip it back up again.

The files that make up an EPUB

If you use InDesign, it will generate an EPUB for you which you can unzip in order to see (and then edit) all the files contained therein (as seen in the previous section). If you use Word or handcode the contents of your ebook, you'll have to organize the EPUB document's contents by hand. Either way, it's a good idea to know where each file should go so that you can reference them properly from the other files as you go along.

In our case, we are creating an ebook for Henry David Thoreau's *Walden*, and the final structure will look something like this.



The files that make up a typical EPUB document. You can download this EPUB from the book's website.

To edit an EPUB's files, you'll need a text editor, preferably with GREG capabilities. GREG lets you use wildcards to turbocharge the search and replace functions and can save you a huge amount of time as you adjust the contents of your EPUB file. While you can use the text editor that came with your computer, say, TextEdit on the Mac or Notepad on Windows, there are many better text editors, ideal for working with XHTML, including BBEdit (<http://www.barebones.com>) or TextMate (<http://macromates.com/>) for Mac, and Notepad++ on Windows (<http://notepad-plus-plus.org/>). Don't use Microsoft Word. It will mess with your HTML.

The *mimetype* file

The *mimetype* file is a simple text file, always titled *mimetype*, that contains a single line, which reads `application/epub+zip`, with no returns or line breaks. It does not change from one EPUB document to another, so you can just copy and paste it into each book folder. For your own use, copy one from any other EPUB file, including the one in the Tutorial folder from the book's website.

The META-INF folder

The META-INF folder, which contains the single *container.xml* file, looks like this:

```
<?xml version="1.0"?>
<container version="1.0"
  xmlns="urn:oasis:names:tc:opendocument:xmlns:container">
  <rootfiles>
    <rootfile full-path="OEBPS/content.opf"
      media-type="application/oebps-package+xml"/>
  </rootfiles>
</container>
```

The only value that might need adjustment is the `full-path` attribute in the `rootfile` element, which should point to the *content.opf* file for your EPUB document. If you always place the *content.opf* in the OEBPS folder, as is the convention, and as is done in this tutorial, you can leave this file the same for each EPUB document.

If you embed fonts in your EPUB, InDesign will place an *encryption.xml* file in the META-INF folder. The *encryption.xml* file is also used if you apply DRM to your EPUB. You can find more information about encryption in the OPS specification: http://www.idpf.org/ocf/ocf1.0/download/ocf10.htm#_Ref8795282. This book does not cover applying DRM to your ebooks.

The OEBPS folder

The OEBPS folder is where the contents of your ebook are found. *OEBPS* stands for *Open eBook Publication Structure*, and is the first in a rather long list of impossible to remember acronyms with which you must now contend. Although you could just call it the “book” folder—as long as you change the path in the *container.xml* file accordingly—the convention is to call it *OEBPS*.

At any rate, the OEBPS folder contains each of the XHTML files that hold the text of your book, a CSS file that contains the formatting instructions, a folder each for images and fonts, and two special XML files: *toc.ncx* and *content.opf*. The *toc.ncx* file is used by ereaders to generate the navigational table of contents. The *content.opf* file holds meta information about your book as well as a list of the individual files that make up its contents.

XHTML and CSS files

The content files in the OEBPS folder can appear in any order, but it's probably easiest to start with the book's contents, which are written in XHTML 1.1 format and styled with CSS. XHTML is a stricter variant of HTML, the same language that is used to write web pages. CSS is the language used to format the contents of an XHTML document.

The beginning of an XHTML document for EPUB should begin with an XML declaration and the proper DOCTYPE, as shown below. Every XHTML document in your EPUB file should begin with this code and should be saved with a utf-8 encoding.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.
    w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

After the initial information about the XHTML document comes the head section, which will contain metadata about the contents themselves, including the title and the location of the style sheet:

```
<head>
<title>Walden, by Henry David Thoreau, originally published in 1854</title>
<link href="Walden.css" rel="stylesheet" type="text/css" />
</head>
```

Note that the title of any one XHTML file *does not determine* the title of your book in ereaders or online. That function is covered in the *content.opf* document. The [link](#) element tells the reader where to find the CSS style sheet file that will be used to format this document ("Walden.css").

After the head section, comes the body. Here is a sample of what the body of an average XHTML document—the first chapter of *Walden*—might look like within an EPUB document.

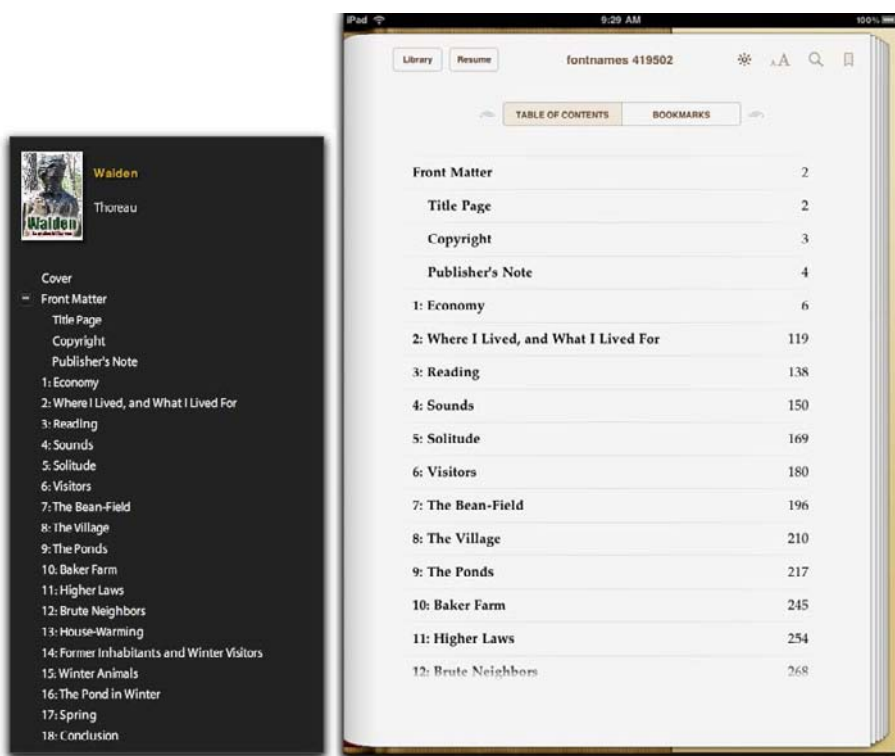
```
<body>
<div id="walden-c1">
  <h1 class="header"> Economy</h1>
  <p class="first-para"><span class="small-caps">When I wrote the
    following pages,</span> or rather the bulk of them, I lived alone,...</p>
  <div class="illustration">
    <p class="caption">Thoreau's house on Walden Pond</p>
  </div>
  <p class="body">I should not obtrude my affairs so much ... </p> ...
  <p class="quote">Inde genus durum sumus, experiensque </p>...
</div>
</body>
</html>
```

Note several things in this document:

- ▶ There are four classes of `p` elements (`first-para`, `caption`, `body`, and `quote`) so that these different types of paragraphs can be formatted differently.
- ▶ The file is called `walden_c1.xhtml` and is located directly in the OEBPS folder.
- ▶ This admittedly abridged version of `walden_c1.xhtml` is just the first chapter of the book. Separating a book into separate files for each chapter makes almost all ereaders begin each chapter on a new page. This is currently the best-supported way of creating a page break.
- ▶ You'll also want to create separate XHTML documents for the cover (as described in "Creating the cover" later in this chapter), for the front matter (title page, table of contents, preface, and so on), and for the back matter (index and blurbs, for example).
- ▶ I'll explain techniques for formatting ebooks in Chapter 4, "Advanced EPUB Formatting".

The *toc.ncx* file for the navigational TOC

The *toc.ncx* file is one of two files that describe the contents of your book to ereaders. It is a simple utf-8 encoded text file that should be located within the OEBPS folder. It can be called anything you like, as long as its extension, which reflects another abbreviation—Navigation Control file for XML—is *.ncx*. The *toc.ncx* file is used by ereaders to create the navigational table of contents, or menu, as shown below. (I call this a “navigational table of contents” to distinguish it from a table of contents in the main body of the contents of your book, which is also possible, though due to the existence of this menu, perhaps less necessary.)



Here is the navigational table of contents of our book, with two levels of hierarchy, shown in Adobe Digital Editions (left), and the iPad (right).

If you use InDesign to generate your EPUB document, you can tell it to generate a navigational table of contents—i.e., a *toc.ncx* file—automatically from the desired headers in your document. See “Create a navigational TOC” in Chapter 2 for more details. You generally won’t need to edit it, but you could—perhaps to add an

additional item to the menu—by following the instructions below. Or, if you have created your content by hand or with Word, you can construct the *toc.ncx* file by hand as well.

The *toc.ncx* file must be a valid XML file. It should begin with the following header:

```
<?xml version="1.0"?>
<!DOCTYPE ncx PUBLIC "-//NISO//DTD ncx 2005-1//EN" "http://www.
daisy.org/z3986/2005/ncx-2005-1.dtd">
<ncx xmlns="http://www.daisy.org/z3986/2005/ncx/" version="2005-1">
```

After the heading comes the *head* section of the *toc.ncx* file. The head section must contain four *meta* elements, *dtb:uid*, *dtb:depth*, *dtb:totalPageCount*, and *dtb:maxPageNumber*, although only the first two are relevant to ebooks.

The value of *dtb:uid*, the user id, should be a string or code that is unique to this ebook. For example, book publishers might use an ISBN. If your book doesn't already have a number or code associated with it, you can generate your own *uuid*, or universally unique id, and use that (just google “generate uuid” to find a generator). The convention is to identify the uuid by prefacing it with *urn:uuid:*. If you have created your EPUB with InDesign, the program will create the user id for you.

The value of *utb:depth* refers to how many levels and sub-levels of items there will be in the table of contents menu. In our example, we have two levels; level 1 is for front matter, back matter, and chapter names; level 2 is for subsections of certain chapters (often the front matter and back matter).

The last two *dtb* items refer only to paper books and can be left at 0, but should not be omitted.

```
<head>
  <meta name="dtb:uid" content="urn:uuid:2a0e205b-c754-317f-2fc8-
    24b9c93e83dc"/>
  <meta name="dtb:depth" content="2"/>
  <meta name="dtb:totalPageCount" content="0"/>
  <meta name="dtb:maxPageNumber" content="0"/>
</head>
```

The second principal part of the *toc.ncx* file is the `docTitle` element and the `text` element it contains. Although it should contain the title of your book, I haven't found a single ereader that pays attention to it at all. Nevertheless, it's probably a good idea to include it. It should look like this:

```
<docTitle>
<text>Walden</text>
</docTitle>
```

The third part of the *toc.ncx* file, the `navMap` element, is where you define the actual navigational table of contents that will appear in the ereader. We'll go through the `navMap` point by point. Begin with the opening `navMap` element:

```
<navMap>
```

Each item in the TOC is created with a `navPoint` element, which has two attributes. The first, `id`, is a unique code that identifies this entry in the navigational TOC and the second, `playOrder`, is a number that indicates the item's *order* in the TOC. Note that each `playOrder` value must be unique. (Note that the order in the actual book when a reader browses from one document to the next is not controlled here, but in the *spine* section of the *content.opf* file. For more details, see "Giving the order of your book's elements (the spine)" later in this chapter.)

```
<navPoint id="navpoint-1" playOrder="1">
```

The `navLabel` element defines the links aspects of the TOC item. The name of the TOC item (clickable text, if you will) is enclosed within a `text` element within the `navLabel` element, as shown below. The first item in our navigational table of contents is the *Cover*.

```
<navLabel><text>Cover</text></navLabel>
```

Next you define the target of the link, that is, which part of the book should be displayed when the reader chooses this item from the navigational table of contents. To do so, create a `content` element that points to the chapter's URL. In this case, fragment identifiers—that reference specific locations in a document—are allowed

(though not required). Since the first document is the cover, we'll point to the XHTML file for the cover. (You can find more information about creating the cover in "Creating the cover" later in this chapter.)

```
<content src="Walden_cc.xhtml" />
```

Finally, finish the cover entry in the navigational table of contents by closing its `navPoint` element:

```
</navPoint>
```

The whole TOC entry looks like this:

```
<navPoint id="navpoint" playOrder="1">
  <navLabel>
    <text>Cover</text>
  </navLabel>
  <content src="Walden_cc.xhtml" />
</navPoint>
```

Now you've got your first entry in the navigational table of contents. To begin the next entry, we'll create a new `navPoint` element for the *Front Matter* item.

```
<navPoint id="navpoint-2" playOrder="2">
  <navLabel><text>Front Matter</text></navLabel>
  <content src="Walden_c0.xhtml" />
```

Since the Front Matter comprises various files—the Title Page, Copyright page, and Publisher’s Note—we’ll have to create subentries on the navigational table of contents to give direct access to them. To do so, create a new, inner `navPoint` element before you close the *Front Matter* `navPoint`. Here are the three subentries for the *Front Matter*.

```
<navPoint id="navpoint" playOrder="3">
  <navLabel><text>Title Page</text></navLabel>
  <content src="Walden_c0.xhtml#toc-anchor"/>
</navPoint>
<navPoint id="navpoint" playOrder="4">
  <navLabel><text>Copyright</text></navLabel>
  <content src="Walden_c0copy.xhtml"/>
</navPoint>
<navPoint id="navpoint" playOrder="5">
  <navLabel><text>Publisher’s Note</text></navLabel>
  <content src="Walden_c0note.xhtml"/>
</navPoint>
```

Note that the Title Page links to an anchor on the same document as the Front Matter file. That is permitted as long as the two URLs are not identical, which they are not, thanks to the appended fragment identifier.

You could also divide each of those subentries into smaller subsections, though it seems rare that that you’d need more than two levels. When you’re done with the sublevels, close the outer `navPoint` element (in this example, the one that corresponds to *Front Matter*).

```
</navPoint>
```

Continue creating `navPoint` elements for each entry in the table of contents. When you’re done, add the closing tags for both the `navMap` and `ncx` elements:

```
</navMap>
</ncx>
```

You can see the entire *toc.ncx* file for this example on the book’s website.

One upcoming but not yet widely supported feature of the *toc.ncx* file is the `<pageList>` element which allows you to relate chunks of your ebook to the corresponding pages in a print book. I'll keep you updated on my website.

It can sometimes be difficult to pinpoint errors in the *toc.ncx* file since there are so many opening and closing elements. You can use the validator at http://www.w3schools.com/xml/xml_validator.asp to check your file for you.

Writing the *content.opf* file

The *content.opf* is the second of two files that describe the contents of your book to ereaders. It is a simple utf-8 encoded text file that should be located within the OEBPS folder. It can be called anything you like, as long as its extension, which reflects another abbreviation—Open Packaging Format—is *.opf* and it is referenced properly by the *container.xml* file in the META-INF directory. The file has four principal parts: the *metadata* section, which gives information *about* the book's contents, the *manifest*, which *lists* the actual files used for the ebook, including images and fonts, the *spine*, which lists the *order* in which the text files should appear in the ebook, and the *guide*, which describes the *role* that each XHTML file plays in the ebook. If desired, you can write the *content.opf* file by hand using the information in this section.

InDesign automatically creates the *content.opf* file when you export to EPUB as described in “Export EPUB from InDesign” in Chapter 2. However, it does not properly add a date, reference the cover file/image, nor does it create a *guide*. You can complete these steps manually as described in this chapter.

The *content.opf* file is in XML format, and begins with the XML declaration and the root *package* element in which all other elements are contained. Note that the *package* element contains three required attributes: *xmlns* for the namespace declaration, *version* (of OPF used), and *unique-identifier*, whose value must match the *id* of the also required *dc:identifier* element created in the *metadata* element later on.

```
<?xml version="1.0"?>
<package xmlns="http://www.idpf.org/2007/opf"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  unique-identifier="bookid" version="2.0">
```

Adding info about your book (metadata)

The metadata section of the *content.opf* file is where you will specify information about your book, including its title, creator, identifier, illustrator, publisher, subject, description, and more. Metadata is what makes it possible for potential readers to find your book. It is very important to add sufficient, accurate metadata for every ebook you publish.

The metadata section begins with the `metadata` element, which itself contains two further namespace declarations:

```
<metadata xmlns:dc="http://purl.org/dc/elements/1.1/"  
  xmlns:opf="http://www.idpf.org/2007/opf">
```

Those namespace declarations allow you to use elements and attributes from the Dublin Core (DC) Metadata specification (<http://dublincore.org/documents/2004/12/20/dces/>) and Open Packaging Format specification (<http://www.idpf.org/2007/opf/opf2.0/download/>), respectively, while keeping your XML document valid. Note that elements that come from the Dublin Core specification should be prefaced with `dc:` while attributes that come from the OPF specification should be prefaced with `opf:`.

Next, you must include a Dublin Core title, identifier, and language in your metadata section. We'll start with the title.

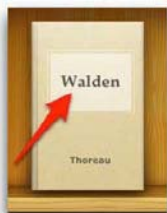
```
<dc:title>Walden</dc:title>
```

The title is not optional, and some ereaders won't even open a book that doesn't have one.

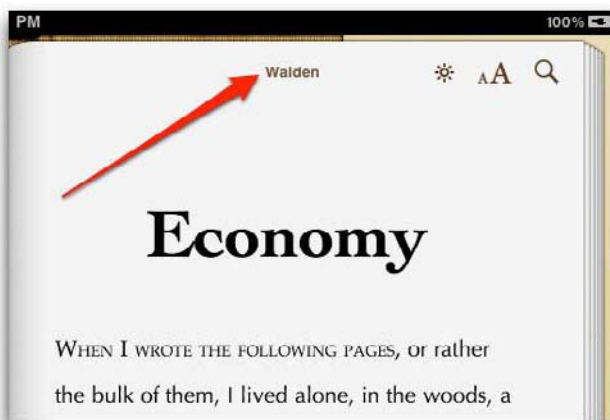
InDesign does not create a title by default. You must either add it within the program by following the instructions in "Add metadata to your ebook" in Chapter 2 or by following the instructions in the rest of this section to add it manually.

The specification says you must have at least one title (but possible more than one) and that though there is no current specification for how multiple titles should be handled by ereaders, they might take into account the `xml:lang` attribute, if present.

There are many places to specify a title of your ebook, but it is important to note that it is the content of the `dc:title` element that is used by most if not all ereaders to identify your book as shown in the following illustrations.



In iBooks on the iPad, the contents of the `dc:title` element are used for the cover if no cover image has been provided.



The title specified in the content.opf file is used to identify your book in the ereader. iBooks uses it on the cover if there is no cover image and also at the top of each right-hand page. ADE lists the title in the left navigation bar (right).

The second required element is `dc:identifier`, whose value should match the value you used in the `dtb:uid meta` element in the `toc.ncx` file and whose `id` should match the `id` of the `package` element's `unique-identifier` attribute. (InDesign can take care of this step for you when you export to EPUB.)

You can specify the *kind* of identifier used with the `opf:scheme` attribute. Although the specification does not yet define which attributes are allowed, I have seen *ISBN* (international standard book number), *DOI* (digital object identifier), *URN* (uniform resource name), *URI* (uniform resource identifier), and *URL* (uniform resource locator). Indeed, you can create multiple identifiers and identify the scheme of each.

Here is what the identifiers look like for our example. Notice that the second identifier has an *id*, *bookid*, that matches the *id* of the package element's *unique-identifier* attribute from the beginning of the *content.opf* file.

```
<dc:identifier opf:scheme="URI">http://www.elizabethcastro.com/epub/  
Walden</dc:identifier>  
<dc:identifier id="bookid">urn:uuid:2a0e205b-c754-317f-2fc8-24b9c93e83dc  
</dc:identifier>
```

The third required piece of metadata is *dc:date*. It must contain at least a four-digit year, but may be followed by an optional two-digit month, and an optional two-digit day. The date must conform to the *Date and Time Formats* note published by the W3C at <http://www.w3.org/TR/NOTE-datetime>. It is most often used for the date of publication. Multiple *date* elements are allowed, for example, to give the original date of publication as well as the date of the EPUB version. The *opf:event* attribute may be used to differentiate among multiple *date* elements.

```
<dc:date opf:event="original-publication">1854</dc:date>  
<dc:date opf:event="ops-publication">2010-02</dc:date>
```

Note that InDesign doesn't currently export the date with the EPUB document, either on CS4 or CS5, so you'll have to add it manually as described above.

The fourth required element in the metadata section is *dc:language*. Its value should conform to RFC 3066, and thus be one of those listed in ISO 639 (http://en.wikipedia.org/wiki/List_of_ISO_639-1_codes). In practice, two-letter codes are the most common. You may include more than one *dc:language* element, if desired. The code for English is "en". Note that iBooks' dictionary currently only works for English and *only* if you have specified "en" as the value for *dc:language*.

```
<dc:language>en</dc:language>
```

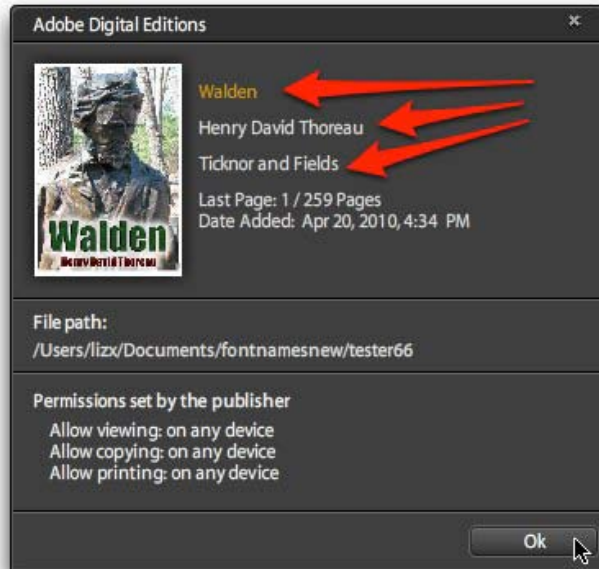
There are a number of optional Dublin Core (dc) elements which you can use to add information about your book to the EPUB file, the most important of which are listed here. The *dc:* prefix must not be omitted.

- ▶ **dc:creator**, used for the author of a book. Although the specification says you can use multiple creator elements to add information about multiple authors, I haven't found an ereader that displays more than one. The **creator** element has two optional attributes, **opf:role** and **opf:file-as**. The allowed values for the **opf:role** attribute (like **aut** for *author*), can be found at <http://www.loc.gov/marc/relators/relaterm.html>. The **opf:file-as** attribute can be used to show how to alphabetize the name. In ereaders like iBooks, if you don't use **opf:file-as**, when a reader lists a book by author, you'll find Henry David Thoreau under *H*, instead of under *T*.
- ▶ **dc:subject**, like those specified by the Library of Congress, though no vocabulary is defined or required by the specification.
- ▶ **dc:description**, could be a summary of the book's contents, or a historical note if the book is old and famous.
- ▶ **dc:contributor**, used for those who helped create the book to some lesser degree than the author, perhaps the illustrator, editor, and so on. The **dc:contributor** element can also contain the **opf:role** and **opf:file-as** attributes described under **dc:creator**.
- ▶ **dc:publisher**, again, you can use multiple elements for multiple publishers but current ereaders only display the first one.
- ▶ **dc:rights**, used to express the Copyright status of the EPUB document.
- ▶ **dc:source**, used to show where a book came from originally.

So, the rest of the metadata section of the *content.opf* file might look something like this:

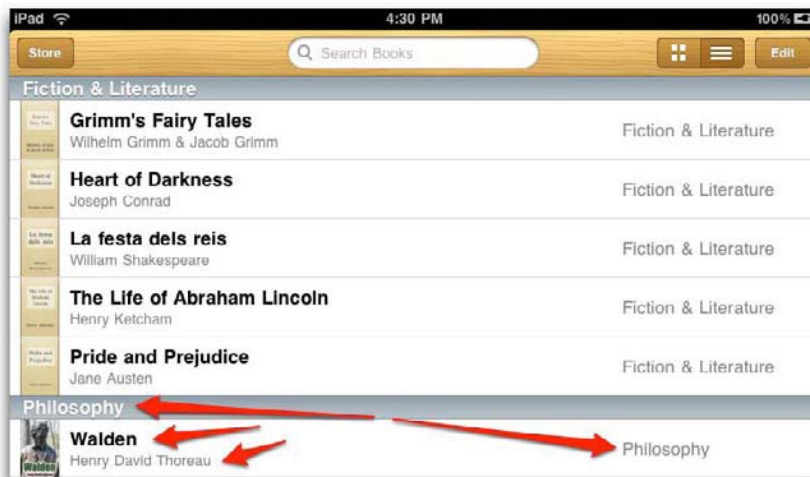
```
<dc:creator opf:role="aut">Henry David Thoreau</dc:creator>
<dc:contributor opf:role="bkp">Liz Castro</dc:contributor>
<dc:subject>19th Century, Philosophy</dc:subject>
<dc:description>Walden (first published as Walden; or, Life in the Woods)
    is an American book written by noted Transcendentalist Henry David
    Thoreau. The work is part personal declaration of independence, social
    experiment, voyage of spiritual discovery, and manual for self reliance.
    [from Wikipedia]</dc:description>
<dc:publisher>Ticknor and Fields</dc:publisher>
<dc:source>Project Gutenberg eText 205</dc:source>
```

The amount of metadata displayed by the ereader varies from one program to the next. Here is what you see if you choose Reading > Item Info... in Adobe Digital Editions for the preceding file:



Adobe Digital Editions shows the title, author, and publisher, as well as internal data about the current page count and date added.

The iBooks application on the iPad shows even less:



The only metadata shown in iPad's iBooks is the title, author, and subject.

Even though ereaders don't currently show all your metadata, it's still a good idea to add it. It may be that future ereaders do a better job, and even if they don't, the metadata's real target is computers that will be able to learn about your EPUB file and catalog it in a meaningful way, making it easier for prospective readers to find.

Listing the contents of your book (the manifest)

The second part of the *content.opf* file is the *manifest*, or listing of the files that make up the ebook, and includes the book's text (in XHTML format), the CSS style sheet used to format them, images, and embedded fonts, in short, anything that is contained in your EPUB document.

It is also necessary to reference the *toc.ncx* file that we created earlier in "The toc.ncx file for the navigational TOC".

The manifest begins with the *manifest* element. It contains an *item* element for each file in the EPUB document. Each item has an *id* attribute that uniquely identifies the file, an *href* attribute that gives the file's location relative to the manifest, and a *media-type* attribute that tells what sort of item it is, generally by giving its MIME type. The most common values for the media-type attribute are:

- ▶ *application/xhtml+xml* for all HTML and XHTML files,
- ▶ *image/jpeg* for jpeg images, *image/gif* for GIF images, *image/png* for PNG images, and *image/svg+xml* for SVG images,
- ▶ *font/opentype* for OpenType fonts, *font/svg* for SVG fonts,
- ▶ and *application/x-dtbnex+xml* for the *toc.ncx* file.

Note that the value of the *href* attribute must point to an entire file, and may not point to a particular anchor in that file (for example, */file.html#particular_section*). In other words, the *href* attribute may not contain a fragment identifier.

Here is what our manifest looks like:

```
<manifest>
<item id="ncx" href="toc.ncx" media-type="application/x-dtbnx+xml"/>

<item id="walden-cc" href="Walden_cc.xhtml" media-type="application/
  xhtml+xml"/>
<item id="walden-c0" href="Walden_c0.xhtml" media-type="application/
  xhtml+xml"/>
<item id="walden-c0copy" href="Walden_c0copy.xhtml"
  media-type="application/xhtml+xml"/>
<item id="walden-c0note" href="Walden_c0note.xhtml"
  media-type="application/xhtml+xml"/>
<item id="walden-c1" href="Walden_c1.xhtml" media-type="application/
  xhtml+xml"/>
<item id="walden-c2" href="Walden_c2.xhtml" media-type="application/
  xhtml+xml"/>
... [and 16 additional xhtml files here]
<item id="WaldenPond-jpg" href="images/WaldenPondwaves.jpg"
  media-type="image/jpeg"/>
<item id="Thoreau_House-jpeg" href="images/Thoreau_House.jpg"
  media-type="image/jpeg"/>
<item id="cover-image" href="images/ThoreauSculptureCover.jpg"
  media-type="image/jpeg"/>
<item id="css" href="template.css" media-type="text/css"/>
</manifest>
```

In this example, there is a *toc.ncx* file, 22 XHTML files (only six of which are shown above), three JPEG image files, and one CSS style sheet file.

Giving the order of your book's elements (the spine)

The third part of the *content.opf* file is the **spine**, which tells the ereader how to sequence the XHTML files. In other words, when a reader gets to the end of a file, the spine determines which one is displayed next. If you've created your EPUB with InDesign, it will complete this step for you.

Begin with a **spine** element, which must have a **toc** attribute whose value matches the value of the **id** in the item that referenced the *toc.ncx* file in the manifest. (I always use *ncx*.)

Next, create an `itemref` element for each XHTML file in the ebook. Each `itemref` element should have an `idref` attribute that matches the `id` of the matching `item` in the manifest.

Here's what it looks like for our file:

```
<spine toc="ncx">
  <itemref idref="walden-cc"/>
  <itemref idref="walden-c0"/>
  <itemref idref="walden-c0copy"/>
  <itemref idref="walden-c0note"/>
  <itemref idref="walden-c1"/>
  <itemref idref="walden-c2"/>
  ... [and so on until...]
  <itemref idref="walden-c18"/>
</spine>
```

Explaining what each file in your book is for (the guide)

Although the purpose of the fourth, optional section, called the `guide`, has mostly been usurped by the `toc.ncx` file, Apple recommends its use. The `guide` lets you identify the *role* of the different files that make up your book. You can choose among `cover`, `title-page`, `toc`, `index`, `glossary`, `acknowledgements`, `bibliography`, `colophon`, `copyright-page`, `dedication`, `epigraph`, `foreword`, `loi` [list of illustrations], `lot` [list of tables], `notes`, `preface`, and `text`.

Apple's documentation says that the first document that iBooks will visualize the first time an ebook is opened will be the first item in the guide whose type is `text`. It turns out, however, that iBooks will actually display the first item in the guide whose type is `preface`, `acknowledgements`, `dedication`, `epigraph`, `foreword`, or `text`.

I find it a bit disconcerting for a book to open up on any page but the very first one, though I suppose I do that all the time when perusing a new book. Still, it makes me feel like I'm missing something, and I immediately start to page backward toward the beginning. On the other hand, it's probably a good idea not to make your readers go through endless initial material before being able to begin to read. For this example, we'll make the book open first on the Publisher's Note (labeled as the `preface` in the guide).

The guide, rather predictably, begins with a `guide` element. Each XHTML file is then listed in order of appearance, along with a `type` (whose values can be one of those listed in the first paragraph of this section) and the location of the file with respect to the manifest.

```
<guide>
  <reference type="cover" href="Walden_cc.xhtml"/>
  <reference type="title-page" href="Walden_c0.xhtml"/>
  <reference type="copyright-page" href="Walden_c0copy.xhtml"/>
  <reference type="preface" href="Walden_c0note.xhtml"/>
  <reference type="text" href="Walden_c1.xhtml"/>
  <reference type="text" href="Walden_c2.xhtml"/>
  ... [and so on, until...]
  <reference type="text" href="Walden_c18.xhtml"/>
</guide>
```

You can find the full list of guide types in the OPF spec online: http://www.idpf.org/2007/opf/OPF_2.0_final_spec.html#Section2.6.

InDesign does not create a `guide` element. If you want one, you'll have to create it by hand as described above.

Finishing the content.opf file

Your *content.opf* file should finish with a closing `package` element:

```
</package>
```

Creating the cover

The cover of an ebook is just as important and serves the same function as the cover of a printed book. Almost all ereaders have some kind of bookshelf or library function, where you can see all of the ebooks that you own, and most also let you view the library either as a list of titles or as a series of images of the ebook covers. Ebook stores also use ebook covers to draw attention to particular books, just like bricks-and-mortar stores use beautiful printed book covers to attract readers.

Unfortunately, although you can create a cover document in InDesign, the program does not automatically designate it as the cover of the ebook, as you might clearly expect, and instead treats it like any other page of your book. You'll have to follow the instructions outlined in the rest of this section to use it as the cover.

Tip

You can add a cover to an EPUB document for iBooks once you have copied it to iTunes by selecting the EPUB document in the Books section of iTunes, choosing File > Get Info (Command I), clicking the Artwork tab, and then clicking Add before choosing the desired cover graphic. However, this will only work with iBooks, and not any other ereaders, as it uses a completely non-standard system. The rest of this section describes how to designate a cover for *all* ereaders, not just iBooks.

An ebook cover has a major disadvantage with respect to its printed counterparts: it's often a tiny icon, perhaps 100 x 150 pixels, automatically generated by the ereader from a larger image. An ebook publisher can supply the larger image, but has less control over how the icon itself is created. It's important, therefore, to make sure you create a cover that is pleasing when reduced in size, and particularly that any text is clear enough to be able to be read when miniaturized.

The larger image of the ebook cover is often also used as the title page of the ebook itself, though this is not required.

Once you have created the cover image, there's a particular system for designating that image as the cover. First, you must create an XHTML file to display the image. The iBooks app won't recognize an image document on its own as the cover of your document.

That document for the cover might look something like this:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.
w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Walden_cc.xhtml</title>
  <link href="template.css" rel="stylesheet" type="text/css" />
</head>
<body>
<div>
  
</div>
</body>
</html>
```

Notice that the body contains a single `div` element to hold the cover image. An `img` element cannot go directly inside the `body` in XHTML.

While ereaders usually generate the icon for the ebook from the cover document, they don't all use the same system (of course!). I've found that unless you add some style information to the image, Adobe Digital Editions selects just a piece of the upper left section of the image and goes from there:



This is only the upper left portion of the cover.

To force ADE to use the whole image for its icon, just add `style="max-width: 100%"` to the `img` tag in the XHTML document.

Now that the XHTML file is complete, you just have to add some information to the *content.opf* file in order for the ereader to recognize this file as the cover.

First, you must add a line to the metadata section of the *content.opf* file that alerts ereaders to which image should be used for the cover:

```
<meta name="cover" content="cover-image" />
```

The value for the **name** attribute should always be **cover**. The value for the **content** attribute must match the **id** of the actual image to be used for the cover in the manifest.

Don't remember the manifest? That's the list (described in "Writing the content.opf file") that gives the name of every file used in the EPUB, including the cover image. Each of these files is also assigned an **id**. The **id** for the image of the cover must match the **content** attribute in the **meta** element.

To wit, here is the entry in the manifest for the book cover's *image file*:

```
<item id="cover-image" href="images/ThoreauSculptureCover.jpg"
      media-type="image/jpeg" />
```

The entry in the manifest for the book cover's *XHTML page* isn't any different than the entry for other XHTML files. Take note of the value of its **id** attribute, however, as you will need it when you create the **spine**.

```
<item id="walden-cc" href="Walden_cc.xhtml" media-type="application/
      xhtml+xml" />
```

In the **spine**, there is also a reference to each XHTML file in the book. This determines the order of the items when a reader browses through the book. There are two reasons why the XHTML file for the cover page should be the first item in the **spine** section. First, this will make it the first item to appear in the book, which seems an appropriate place for the cover. Second, Adobe Digital Editions uses the first element listed in the **spine** to generate the cover icon.

Each `itemref` element in the spine has an `idref` attribute that should match the value of the `id` attribute for the corresponding `item` in the manifest section. Notice that in this case, the `idref` is `walden-cc` and that matches the XHTML file item in the manifest, as shown above.

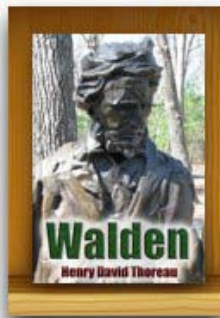
```
<itemref idref="walden-cc"/>
```

Finally, Apple recommends putting information about the cover in the `guide` section. I have successfully loaded a book into iBooks without the guide, but there's certainly a possibility that Apple or other ereaders will depend on this feature in the future, so I recommend adding it just in case.

```
<reference type="cover" title="Cover of Walden" href="Walden_cc.xhtml" />
```

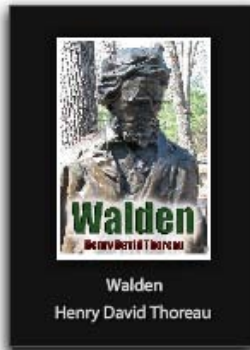
Note that you should use a value of `cover` for the `type` attribute, and that the value of the `href` attribute should point to the XHTML file that contains the cover page (not the image itself).

If you follow these steps, your book will have a beautiful icon in iBooks...



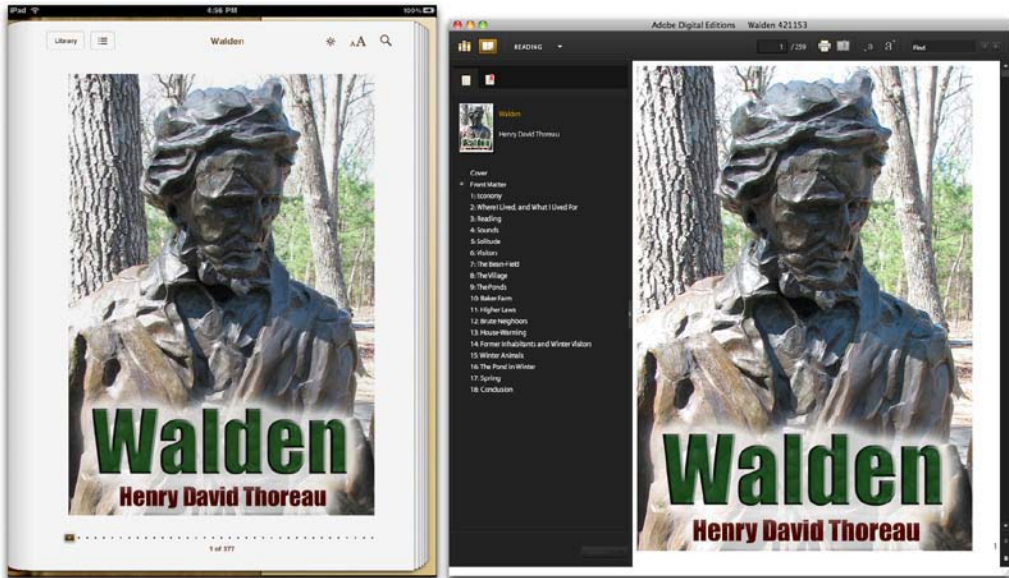
In Bookshelf mode, the iBooks app shows a fairly good miniaturized version of the cover image.

a reasonable one in Adobe Digital Editions...



Adobe Digital Editions creates a much more pixelized, but still recognizable, icon of the cover, using the cover's XHTML page as its basis.

and a beautiful looking cover page in all ereaders that support images:



iBooks is on the left, ADE on the right.

Essentials of content.opf in ePub

meta with name cover required for cover reference

```
<metadata>
<dc:title>Walden</dc:title>
...
<meta name="cover" content="cover-image" />
....
```

toc in spine matches id of toc.ncx in manifest

```
<spine toc="ncx">
<itemref idref="walden-cc"/>
<itemref idref="walden-c0"/>
<itemref idref="walden-c0copy"/>
<itemref idref="walden-c0note"/>
<itemref idref="walden-c1"/>
...
```

contents of dc:title in metadata determines title of eBook in readers

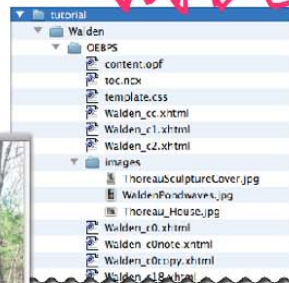
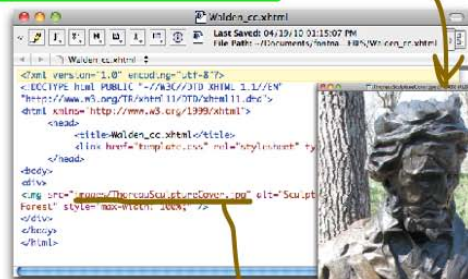
idrefs in spine match ids in manifest

```
<manifest>
<item id="ncx" href="toc.ncx" media-type="application/x-dtbncx+xml"/>
<item id="walden-cc" href="Walden_cc.xhtml" media-type="application/xhtml+xml"/>
<item id="walden-c0" href="Walden_c0.xhtml" media-type="application/xhtml+xml"/>
<item id="walden-c1" href="Walden_c1.xhtml" media-type="application/xhtml+xml"/>
...
<item id="cover-image" href="images/ThoreauSculptureCover.jpg" media-type="image/jpeg"/>
```

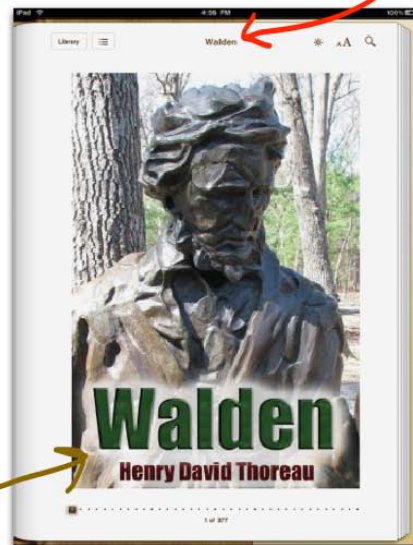
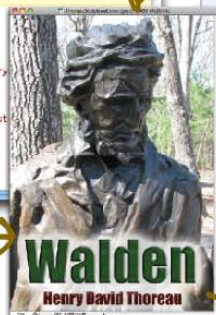
```
<guide> [optional]
<reference type="cover" href="Walden_cc.xhtml"/>
<reference type="title-page" href="Walden_c0.xhtml"/>
<reference type="copyright-page" href="Walden_c0copy.xhtml"/>
<reference type="preface" href="Walden_c0note.xhtml"/>
<reference type="text" href="Walden_c1.xhtml"/>
```

hrefs in guide and manifest match file name and path

content in cover meta matches id of cover image in manifest



XHTML file for cover references cover image



The relationship between different parts of the content.opf can be confusing. Hopefully this graphic doesn't make it worse!

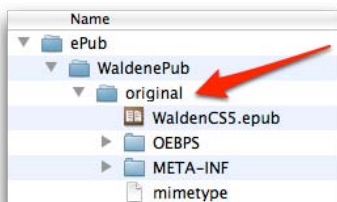
Zippping and testing

Once you have a file unzipped, you can make changes to its files, as described in the preceding pages and in the following chapter. To see the effect of those changes, you'll have to zip the file back up again and open it with an ereader.

Organizing files before rezippping

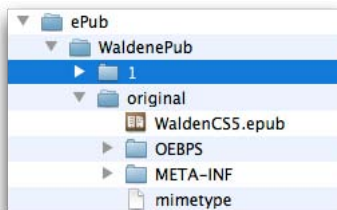
1 I recommend creating a separate directory for a given set of tests within your general EPUB directory. For example, in the section, “Unzipping an EPUB” earlier in this chapter, we created the *WaldenePub* directory to contain the files from the CS5-generated EPUB that we unzipped.

2 Next, create subdirectories so that you can make and test changes incrementally, and always get back to a version that worked. With that aim, I recommend creating a new directory for the original EPUB files, no matter how they were generated. I call this “original”.



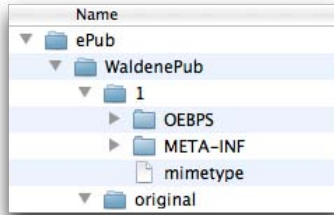
Enclose the original EPUB file and the files that have been newly unzipped from it into their own subdirectory, called “original”.

3 Select that directory and duplicate it and call it “1”. Using simple, short, names without spaces will facilitate zipping and unzipping.



Duplicate the original directory, and call it “1”.

- 4 Open the *1* directory and delete the original EPUB file, leaving only the *OEBPS* and *META-INF* directories and the *mimetype* file.



Delete the original EPUB file that has been duplicated within the 1 directory.

- 5 Next, edit the XHTML and CSS files as desired, perhaps using the techniques discussed in the rest of this chapter, or those in Chapter 4, “Advanced EPUB Formatting”.

- 6 The iBooks app has a curious caching behavior that prevents it from recognizing a new EPUB file unless you have changed its name. Therefore, before you rezip your EPUB, you must open the *content.opf* file (within the *OEBPS* directory) and change the value in the `<dc:title>` element. I generally just add a digit.



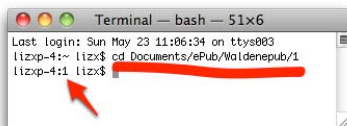
During testing, I add a new number to the title in the content.opf file to force iBooks to see it.

Rezipping after edits

Once you’ve edited your content files and updated the *content.opf* file so that the iBooks app will see it, you’re ready to rezip the files into an EPUB file.

- 1 Open the Terminal program (on the Mac) or Zip (on Windows; <http://bit.ly/aUXJLG>).

- 2 Navigate to the numbered directory that contains the current iteration of the test. In this example, it's the *1* directory.

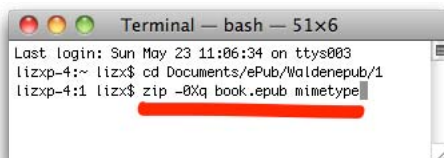


```
Terminal — bash — 51x6
Last login: Sun May 23 11:06:34 on ttys003
lizxp-4:~ lizx$ cd Documents/ePub/Waldenepub/1
lizxp-4:1 lizx$
```

A red arrow points to the prompt `lizxp-4:1 lizx$` in the terminal window.

Navigate to the directory that contains the edited files, in this case 1.

- 3 Type `zip -0Xq book.epub mimetype` to add the *mimetype* file with no compression and then hit Return.

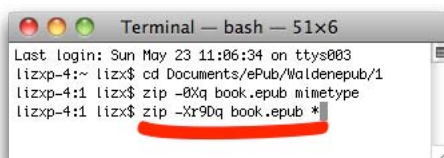


```
Terminal — bash — 51x6
Last login: Sun May 23 11:06:34 on ttys003
lizxp-4:~ lizx$ cd Documents/ePub/Waldenepub/1
lizxp-4:1 lizx$ zip -0Xq book.epub mimetype
```

A red underline is under the command `zip -0Xq book.epub mimetype` in the terminal window.

I've had the best luck when I zip in two parts. First, as shown here, add the mimetype file with no compression.

- 4 Type `zip -Xr9Dq book.epub *` to compress the remaining folders and their files into the EPUB document and then hit Return.

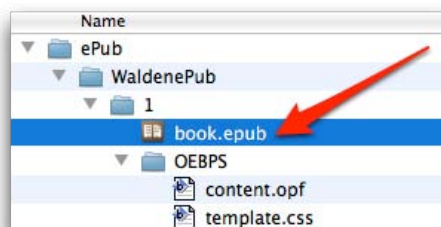


```
Terminal — bash — 51x6
Last login: Sun May 23 11:06:34 on ttys003
lizxp-4:~ lizx$ cd Documents/ePub/Waldenepub/1
lizxp-4:1 lizx$ zip -0Xq book.epub mimetype
lizxp-4:1 lizx$ zip -Xr9Dq book.epub *
```

A red underline is under the command `zip -Xr9Dq book.epub *` in the terminal window.

The second zip command compresses everything else in the current directory into the book.epub file.

- 5 Back in the Finder, you'll see that the new *book.epub* file appears in the *1* directory.

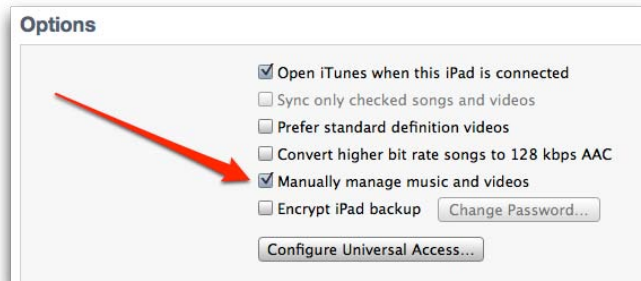


The newly created book.epub file appears in the 1 directory in the Finder.

Getting the new EPUB file to the iPad

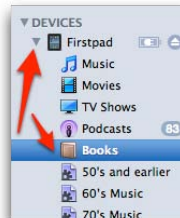
Once you've created the new EPUB file, you'll want to test it on the iBooks app on the iPad. The fastest technique I've found is detailed below.

- 1 Open iTunes and make sure your iPad is connected. If you have just plugged the iPad into your Mac, you can cancel the sync for now.
- 2 Select the iPad in the left-column of the iTunes window so that the Sync button is available, but don't sync it yet, and in fact, cancel the sync if it has started automatically.
- 3 In the main part of the window, check the Manually manage music and videos box so that you can sync books without syncing your entire iPad.



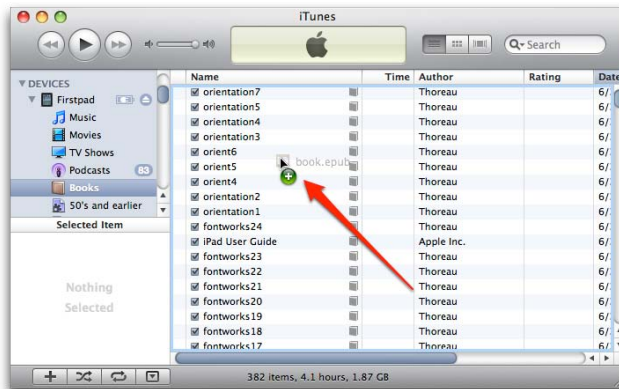
Choosing this option will keep the iPad from syncing automatically.

- 4 Click the arrow next to your iPad's name and then choose Books. All of the books (both EPUB and PDF) currently on your iPad are displayed.

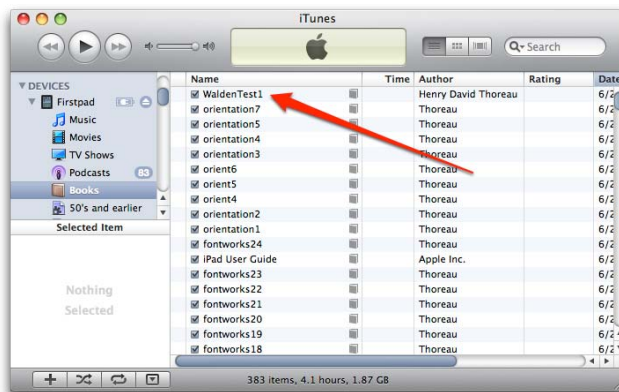


Click the arrow to the left of the iPad's name to reveal its contents. Then click the Books section. Your iPad's books will be displayed in iTunes' right pane.

- 5 Drag your EPUB document to the list of Books in the right pane of iTunes. Only the new ebook will be synced to the iPad.

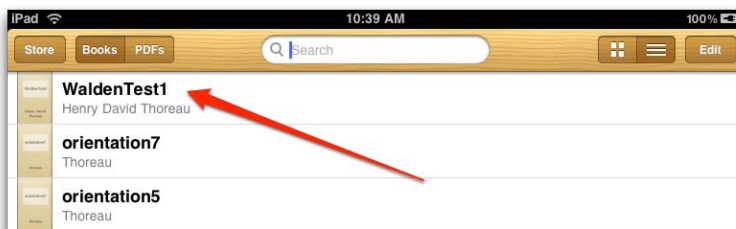


You'll only be able to drag an EPUB file to the Books section of your iPad if you've first chosen Manually manage music and videos in step 3.



The ebook soon appears both in the Books list and on the iPad.

- 6** On the iPad, open iBooks, if it's not already open. It will take a few seconds for the latest iteration of your test to appear. When it does, click it to open, and check your ebook to see if the changes have had the desired effect.

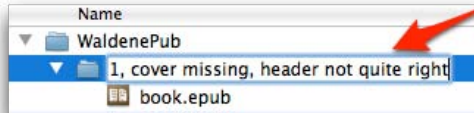


The new EPUB file appears in iBooks. Note that the title comes directly from the content.opf file. If your EPUB has a cover, you may wish to view the books in list form, in order to see the distinct title.

Further editing, re-zipping, and testing

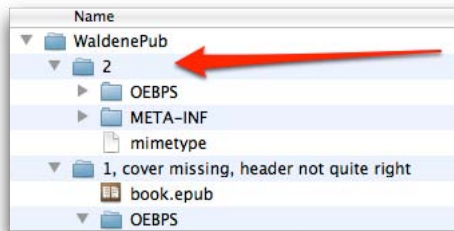
I hardly ever get the desired results on the first try. Perhaps you'll have better luck! At any rate, I've streamlined the re-editing, re-zipping, and retesting sequence in order to save time. Here's what I've found.

- 1 Go back to the 1 directory and change its name to indicate the results of your test. What did you add or remove? Was it successful?



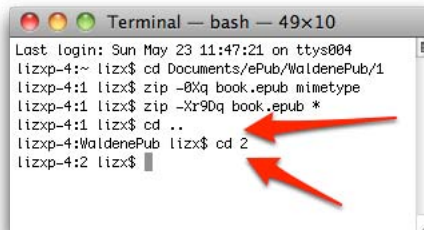
I like adding notes to each test so I remember what worked and what remains to be done.

- 2 Duplicate the 1 directory and call it "2". Delete the EPUB file that it contains. Edit the content files as desired and update the title in the *content.opf* file.



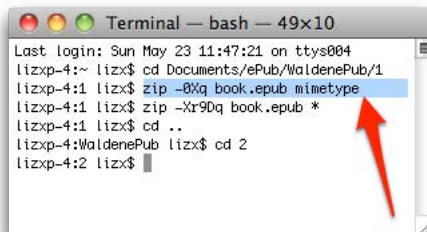
The 2 directory is a duplicate of 1, with the EPUB file deleted.

- 3 In Terminal or Zip, navigate to the 2 directory by typing `cd ..` and Return, followed by `cd 2` and Return.



In Terminal or Zip, type `cd ..` to go up one directory (out of 1) and `cd 2` to go into the 2 directory.

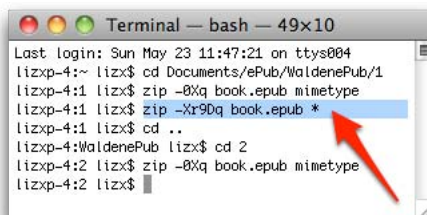
- 4 In Terminal, copy the `zip -0Xq book.epub mimetype` line from the earlier test and then Paste. You don't have to have the cursor in the last line, it will automatically go there. Take care not to copy any more than the actual command.

A screenshot of a macOS Terminal window titled "Terminal — bash — 49x10". The window shows a series of commands and their outputs. The command `zip -0Xq book.epub mimetype` is highlighted in blue on the line `lizxp-4:1 lizxp$ zip -0Xq book.epub mimetype`. A red arrow points from the right side of the terminal towards the highlighted command.

```
Terminal — bash — 49x10
Last login: Sun May 23 11:47:21 on ttys004
lizxp-4:~ lizxp$ cd Documents/ePub/WaldenePub/1
lizxp-4:1 lizxp$ zip -0Xq book.epub mimetype
lizxp-4:1 lizxp$ zip -Xr9Dq book.epub *
lizxp-4:1 lizxp$ cd ..
lizxp-4:WaldenePub lizxp$ cd 2
lizxp-4:2 lizxp$
```

Copy the earlier zip command. It doesn't matter that you used it in the 1 directory. Make sure you copy the return at the end. Then Paste (there's no need to place the cursor in the last line).

- 5 Copy the second zip line and paste again to execute.

A screenshot of a macOS Terminal window titled "Terminal — bash — 49x10". The window shows the same sequence of commands as the previous terminal, but now the command `zip -Xr9Dq book.epub *` is highlighted in blue on the line `lizxp-4:1 lizxp$ zip -Xr9Dq book.epub *`. A red arrow points from the right side of the terminal towards the highlighted command.

```
Terminal — bash — 49x10
Last login: Sun May 23 11:47:21 on ttys004
lizxp-4:~ lizxp$ cd Documents/ePub/WaldenePub/1
lizxp-4:1 lizxp$ zip -0Xq book.epub mimetype
lizxp-4:1 lizxp$ zip -Xr9Dq book.epub *
lizxp-4:1 lizxp$ cd ..
lizxp-4:WaldenePub lizxp$ cd 2
lizxp-4:2 lizxp$ zip -0Xq book.epub mimetype
lizxp-4:2 lizxp$
```

Copy the second zip line and paste (again, no need to place the cursor in the last line).

- 6 As before, copy the newly created EPUB file to iTunes.

- 7 Assuming the previous test was still open, click Library in iBooks (to exit the current test), and wait a few seconds for the newly copied EPUB file to appear in the iBooks list.

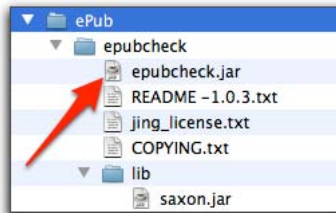
- 8 Verify the results and repeat as necessary.

Tip

In Windows, you can edit files without unzipping the file. However, you shouldn't overwrite old files with new ones in Windows 7 as it can mess up the compression.

validating your EPUB file

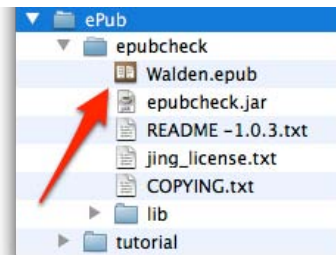
Once you've completed your EPUB file, it's a good idea to check it to make sure it's properly written. The standard for validating EPUB files—passage of which is required in order to be accepted into Apple's iBookstore—is called *EpubCheck* and is available at <http://code.google.com/p/epubcheck/>. Once you download the program, you can put it in its own folder within your *EPUB* folder. I find it useful to simplify both the name of the folder and the name of the program so that they don't include the version numbers. That way, you don't have to type the version name every time you use the program.



The epubcheck.jar program within the epubcheck folder, within my ePub folder (where I keep all my EPUB related documents).

1

To begin, drag your latest EPUB file into the *epubcheck* folder.



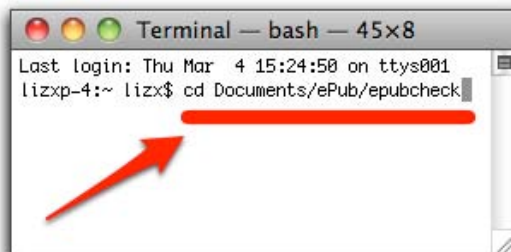
We'll put the Walden.epub file into the epubcheck folder temporarily so that we don't have to type a lot of paths in the Terminal window.



Tip

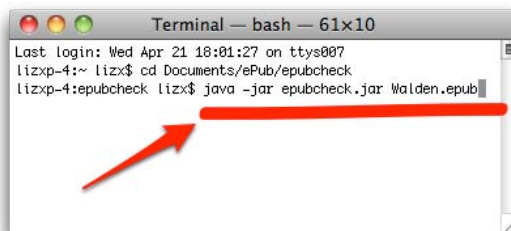
You can also validate your EPUB file by uploading your file to Threepress Consulting's validation service at <http://threepress.org/document/epub-validate/>.

2 Next open the Terminal program, open a new window, and navigate to the *epubcheck* folder.

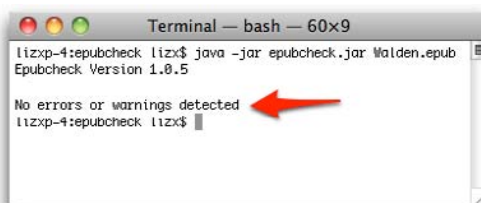


My epubcheck folder is in the ePub folder within my Documents folder.

3 Now we can perform the check. Type `java -jar epubcheck.jar Walden.epub`. You don't have to type the path to either the *epubcheck.jar* program or the *Walden.epub* file because they're both in the current directory to which we just navigated. The EpubCheck program lists any errors and gives the line number and file name where they occur to facilitate corrections.



By simplifying the name of the directory and program, and moving the pertinent files within it, working from the command line gets a little less cumbersome.



At first, epubcheck finds a typo in the media type on line 12 of the content.opf file. Once corrected, epubcheck tells us that the file has no errors (right).

Converting EPUB to Kindle's Mobi

Unfortunately, Amazon's Kindle does not support the EPUB format. Amazon has, however, published a tool for converting EPUB files to Kindle's Mobi format, called KindleGen. It's not a very friendly tool, and we'll have to return to the command line to make it work.

EPUB is a more robust format than Mobi, though they both are based on HTML, so for best results, keep your files simple, with no text wrap or background images. For more details on formatting for Kindle, I recommend Joshua Tallent's book, *Kindle Formatting*, published by eBook Architects.

- 1 Create a validated EPUB file as described in the rest of this book. (You can also generate Mobi format files directly from HTML.)
- 2 Download both KindleGen and the Kindle Previewer from Amazon's site (<http://www.amazon.com/gp/feature.html?ie=UTF8&docId=1000234621>).
- 3 Copy the KindleGen program to your home directory. (I couldn't get it to work anywhere else.) On a Mac, that's the directory with your username. On Windows, that's `c:\`.
- 4 Copy your completed, validated EPUB file to the KindleGen directory you just created.
- 5 Open Terminal on the Mac and type `~/KindleGen/kindlegen ~/KindleGen/Walden.epub`, (that is, the path and file name of the KindleGen program followed by the path and file name of the EPUB you want to convert). In Windows, open a Command window (by typing `cmd` in the Run window), and then type `c:\KindleGen\kindlegen c:\KindleGen\Walden.epub`. KindleGen will create a file with the same name as your EPUB file but with the `.mobi` extension.
- 6 Open the Kindle Previewer and test your new Kindle document.

Advanced EPUB Formatting

4

InDesign, not to mention Word, doesn't even begin to scratch the surface of the design capabilities available in EPUB. I haven't found an EPUB editor that does. Regardless of what you start with—InDesign, Word or handcoded XHTML—you can always add additional effects with a text editor in order to take advantage of all that EPUB supports. In this chapter, you'll learn about:

- ▶ Ensuring ereaders use your CSS
- ▶ How InDesign writes XHTML and CSS and what to do about it
- ▶ How to choose and specify fonts for your ebook
- ▶ How to keep elements together, create drop caps and small caps, control spacing and indents, deal with short lines, and insert borders and backgrounds
- ▶ What to do about hyphenation
- ▶ How to insert images and have text wrap around them
- ▶ How to add links, tables, and even video to your ebook
- ▶ And how to convert InDesign inheritance into CSS inheritance

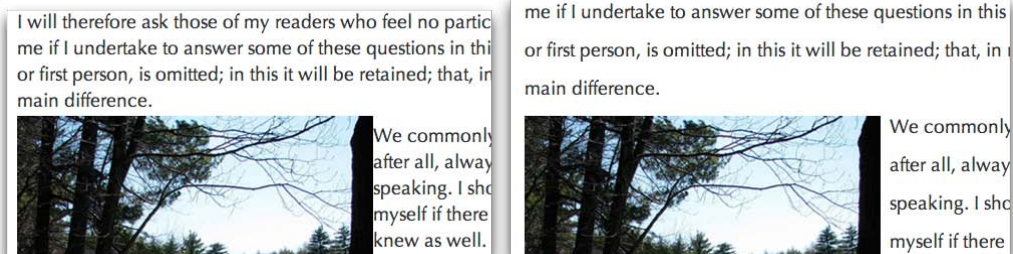
All of the techniques in this chapter are based on XHTML and CSS that work in iBooks on the iPad and other ereaders, but that aren't generated by Word or InDesign. For a more detailed explanation of how to write XHTML, see my bestselling *HTML, XHTML, and CSS: Visual QuickStart Guide, 6th Edition*, also published by Peachpit Press.

Ensuring ereaders use your CSS

Because older ereaders weren't as good at supporting CSS, newer ereaders got into the extremely dubious habit of automatically overriding designers' CSS in order to compensate. In particular, properties like `margin`, `padding`, `line-height`, `width`, `font-size`, and `font-family` are routinely ignored for one's own good! Gee, thanks! Luckily, some ereaders will listen to you if you override their override by using `!important`.

1 To insist that your CSS rule take precedence, add `!important` to the end of it (between the property/value pair and the semicolon).

```
margin: 0 .5em 0 0 !important;
```



The declarations in the CSS for these two examples are the same: a line-height of 2 em, and a margin of .5em to the right of the photo. But on the right, I added `!important` to each declaration so the ereader would not ignore my CSS.

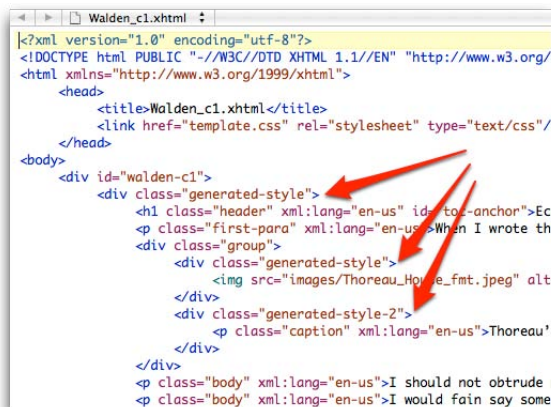
Because this overuse of `!important` can cause some rules to override other rules in the same style sheet, it will be wonderful when these ereaders stop requiring its use. In the meantime, be aware that an `!important` declaration takes precedent over one that is not so important, even among your own rules in your own CSS file.

Cleaning up InDesign EPUB files

I think InDesign is a really powerful tool for creating EPUB files, but it's nowhere near perfect—at least not yet. In this section, I'll show you how to clean up the EPUB files that InDesign produces. Even if you haven't used InDesign, you may find some useful tips. To edit InDesign's files, you'll have to unzip the EPUB as described in "Unzipping an EPUB" in the previous chapter.

How InDesign writes XHTML

When InDesign exports a document, it generates style information in a very specific way. First, it creates one large `div` element whose `id` is the title of your InDesign file. It then creates another `div`, with a `class` of `generated-style` for each frame, in order, from left-top to bottom-right. Images are exported into `divs` of their own with a `class` of `generated-style-2` while grouped items, unsurprisingly, get `divs` with a `class` of `group`. Lines and shapes are ignored outright. (If a document begins with an image, the `img` elements are enclosed in `generated-style` classed `divs` while the text frames are enclosed in `generated-style-2` `divs`.) All `generated-style` classes may be deleted.



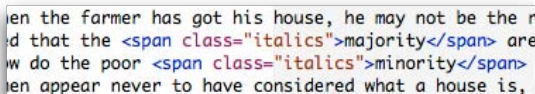
InDesign exports paragraphs from the TOC as headers and everything else as `p` elements. Everything is enclosed in larger `div` elements with generic classes.

In CS5, the way the actual text is exported depends directly on whether or not you've defined a TOC style (see "Creating a Table of Contents Style" in Chapter 2). If you have, and if you've selected that TOC style upon export (see "Export EPUB from InDesign" in Chapter 2), then the InDesign paragraph styles selected to generate the TOC will be exported as headers (`h1`, `h2`, and so on), while every other para-

graph—even Bulleted and Numbered lists—will be exported as a `p` element. (If you have not defined a TOC style or don't choose it when exporting, or if you're using CS4, *everything* is exported as a `p` element.) Regardless, all header and `p` elements get a `class` equal to the name of the paragraph style in InDesign.

InDesign also inserts an `xml:lang` attribute into every single element to specify the element's language. You may search and delete these.

Character styles are exported as `span` elements with the InDesign character style name as a `class` attribute.



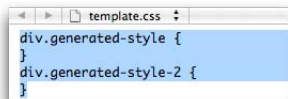
Text that was styled with a character style becomes enclosed in a `span` element with that character style's name as its `class`.

Local formatting (which is ignored in CS4), is exported as a `span` with a `class` of no-style-override. Sometimes, InDesign creates generated-style classes in `span` elements as well. These can be safely deleted.

How InDesign writes CSS

The CSS files that InDesign generates will depend directly on the choices you make when you do the export. You can always control the process completely by choosing Style Names Only or Use Existing CSS File upon export. The first option will output a CSS file with each InDesign style mapped to a class with the same name, attached to a `p` or header element. You can then define the style rules manually. The second option lets you choose a predefined CSS style sheet that will be used in the EPUB. If you often use the same style sheet, this may be the most effective option as long as you make sure the CSS classes match the style names in the InDesign document. In this example, nevertheless, we'll assume that you're using InDesign to generate the style information in the CSS style sheet.

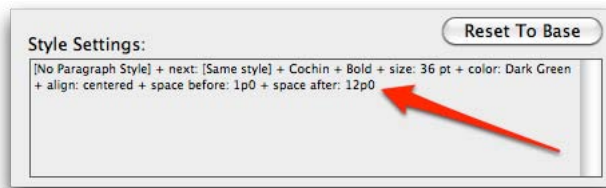
1 Just as InDesign creates extra generated-style classes in the XHTML, it creates empty style rules for these styles as well. I think it's best to get rid of them.



InDesign creates these selectors with no style definitions. They're pretty useless.

Next, for each paragraph style, InDesign creates a set of style rules that defines each of the following properties, whether you've defined a value for that property in InDesign or not: **font-family**, **font-weight**, **font-style**, **font-size**, **line-height**, **text-decoration**, **font-variant**, **text-indent**, **text-align**, **color**, and **margin**. Indeed, it *only* creates values for those 11 properties, and all other aspects of the style—even features that CSS and InDesign support in common—are ignored.

For example, imagine the header style in the *Walden* document, defined to have green, 36 point, bold text set in Cochin, centered, with 1 pica of space before and 12 picas after.



Note that the text is supposed to be 36pt, the space before is 1 pica and the space after is 12p. No indent, underline, small-caps, or italics are defined.

When InDesign exports this style information to CSS, it gets much of it right, but unfortunately, it erroneously offers *default values* for properties that weren't explicitly defined. In this example, that's **line-height**, **text-decoration**, **font-variant**, **font-style**, and **text-indent**.

```
h1.header {
  font-family: "Cochin";
  font-weight: bold;
  font-style: normal;
  font-size: 3em;
  line-height: 1.20em;
  text-decoration: none;
  font-variant: normal;
  text-indent: 0em;
  text-align: center;
  color: #1c910a;
  margin: 0.33 0 4em 0;
}
```

2 Unfortunately, because of CSS inheritance, a default value is not neutral. Instead of not defining, say the `text-decoration`, this CSS specifically says “make sure there is *no* text decoration”. I recommend eliminating these extraneous and sometimes harmful declarations. After the changes, the rule will look like this:

```
h1.header {  
    font-family: "Cochin";  
    font-weight: bold;  
    font-size: 3em;  
    line-height: 1.20em;  
    text-align: center;  
    color: #1c910a;  
    margin: 0.33 0 4em 0;  
}
```

Remember that an *em* is a curious measurement. When used to specify a font size, as the 3em in the previous code, it means 3 times the size of the parent element’s font size. The parent element in this case is a `div` which gets its font size from the `body`, which in iBooks and in many websites, is 16 pixels by default. So the `h1` element here would end up 48 pixels high (3 x 16 pixels).

When you use ems for line height, text indents, and margins, however, one em equals the *font size of the current element*. So, the specified 4 em for the `margin-bottom`, since the font size is 48 pixels, will be 192 pixels.

InDesign doesn’t always use the same font names that the iBooks app supports. For example, if you choose the *Bradley Hand* font for a style in InDesign, the exported CSS uses “Bradley Hand ITC TT” but iBooks will only understand “Bradley Hand” or `BradleyHandITCTT` (with no spaces). You can find the full list of official names in “Fonts in your ebook” later in this chapter.

3 Make sure the value for the `font-family` property is the official name for that font, as given in the list in “Fonts in your ebook”.

4 Repeat these steps for each set of rules in the CSS document.

Maintaining InDesign inheritance in the CSS

InDesign lets you base multiple styles on a single foundational style. Then when you change the foundation style, all of the styles that depend on that foundation are also updated. It's a very useful feature. For example, in this book that you are reading, I have formatted all of the code examples with variations on a single InDesign style called *code*. The base code style includes font information and basic indenting. The code-first style includes everything in *code*, and adds a rule above. The code-last includes everything in *code*, but adds a rule below. And so on. You get the idea.

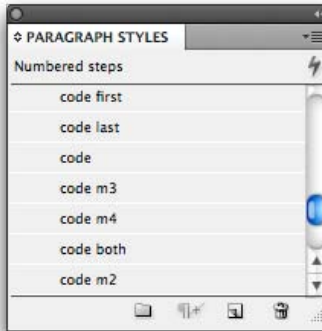
CSS has a similar functionality. You might assign a number of style rules to the *p* element, and then add additional style rules to only those *p* elements of class *code-last*, for example. In addition, CSS applies *inheritable* styles automatically to elements contained within other elements. For example, if there is an *em* element within a *p* element, the contents of the *em* element may inherit styles applied to the *p* element. Whether or not a style rule is inherited is determined by the property's characteristics (and sometimes on whether or not you choose to make it inheritable).

Unfortunately, and as you might expect, InDesign doesn't take advantage of CSS's inheritance features. As we've just seen, InDesign creates separate, independent sets of style rules for each and every style in your InDesign document. Not only that, but it sets the default values for several properties, even though these are not necessarily defined in InDesign. Further, it does not reflect the inheritance of one style on another in the exported declarations. This is a shame and can be remedied.

I have developed a technique for maintaining the inheritance in InDesign styles in the CSS styles that InDesign exports.

1 In InDesign, create the base style and title it with a single lowercase word. My base style is called *code*.

- 2 Still in InDesign, create the styles that are based on the main style and title them with the same lowercase word used for the base style, plus a space and a single word modifier. For example, *code first*, *code last*, *code both*, etc.



Each of these “child styles” is based on the base code style.

- 3 Style the text in your document with the InDesign styles and export to EPUB as usual. InDesign converts the space in the style names to a hyphen.

```
<p class="code-first">h1.header {</p>
  <p class="code-m2">font-family: "Cochin";</p>
  <p class="code-m2">font-weight: bold;</p>
```

...

- 4 In the generated CSS file, eliminate any default values from the base style that you do not want to declare explicitly.

```
p.code {
  font-family: "AmericanTypewriter";
  font-size:.8em;
  line-height: 1.2em;
  text-align:left;
  margin:0 5% 0 5% !important;
  background: #f8dddd;
  width: 90% !important;
}
```

5 Still in the CSS, eliminate the base style name from the child style rules. (In this example, *code-first* becomes just *first*; *code-m2* becomes just *m2*.) Then, simplify the child rules so that they contain only those declarations that are above and beyond what is defined in the base style.

```
.first {  
    border-top: 2px dashed red;  
    padding-left: 1%;  
    padding-top: .2em;  
    padding-bottom: .2em;  
}  
.m2 {  
    text-indent: -5%;  
    padding: .2em 0 .2em 11% !important;  
}
```

6 In the XHTML file, use search and replace to convert the hyphen that separates the base name from the child name into a space. Now, each of the two words are independent classes and *both* will be applied to the paragraph. In this way, both the base style and the child style will be applied to the affected paragraphs.

```
<p class="code first">h1.header {</p>  
    <p class="code m2">font-family: "Cochin";</p>  
    <p class="code m2">font-weight: bold;</p>  
...
```

Now when you change the rules for the base style in the CSS, all of the paragraphs that depend on that rule will be affected. At the same time, each paragraph will continue to be individually affected by the child rule that applies to it.

Fonts in your ebook

The EPUB specification allows you to designate particular fonts for your book so that ereaders can use your font of choice instead of the defaults. Unfortunately, not all ereaders support such a feature.

Many ereaders offer a very limited set of fonts for display. The Barnes & Noble Nook, for example, currently offers only Amasis, Helvetica Neue, and Light Classic. The iBooks app on the iPad has more than 30 fonts (with multiple styles of each) pre-installed, but only allows the reader to choose one of six: Palatino (the default), Baskerville, Cochin, Georgia, New Times Roman, and Verdana.

Choosing fonts

There are those that would say that ebook designers should always leave the choice of fonts up to the reader, who, if they so desire, should be able to change the font and text size of a book via the ereader's settings. While it is certainly true that ereaders can give human readers much more control over how their books should appear—and I will admit to loving making the text just a wee bit larger to appease my older eyes—I think it's important to offer an initial beautiful design even if the reader later overrides your choices.

I think well-designed books are a pleasure to read and their very design can be a major factor in attracting readers and drawing them into the content of the book. Conversely, a poorly designed book can be downright difficult to read, and can distract a reader from even well-written prose.

Specifying fonts by name

Let's look more carefully at the actual code required for specifying a font.

1 In the CSS, add `font-family: "fontname";` to your list of rules for a given selector. Multiple word names must be enclosed in parentheses. For example, to specify the American Typewriter font for all `samp` elements, you'd use this CSS:

```
samp {  
    font-family: "American Typewriter";  
}
```

You can find the correct name for each font in the list “Fonts available for ebooks on the iPad” further ahead in this section. Note that InDesign does not always use the name that iBooks supports; you may have to edit the InDesign CS5 generated code to match what is shown here.

Also keep in mind, however, that while these fonts are always available to any program on the iPad—including both Safari and iBooks—they are not all available on Mac, Windows, and Linux machines. It’s a good idea to specify alternate fonts should the desired one not be available. (For more details, see “Specifying alternate fonts” later in this section.)

Finally, be sure to read the section on “iBooks and fonts” if you’re having trouble making the fonts appear.

Specifying fonts by style

CSS also lets you specify a generic style of font, that depends on the system on which the content is being displayed. For example, you could specify that a paragraph be shown in a sans-serif font and on some systems that would mean Helvetica and on others it might mean Arial. By giving guidelines instead of specific font choices, you have a chance of getting part of what you want, instead of going for all or nothing.

The iPad has designated five particular fonts to be matched to CSS generic font names: Times for **serif**, Helvetica for **sans-serif**, Courier for **monospace**, Snell Roundhand for **cursive**, and Papyrus for **fantasy**.

1 In the CSS, add **font-family: generic;** to your list of rules for a given selector, where *generic* is one of **serif**, **sans-serif**, **monospace**, **cursive**, or **fantasy**. For example, to specify a sans-serif font for all **samp** elements, you’d use this code:

```
samp {  
    font-family: sans-serif;  
}
```

Generic font names are never enclosed in quotes.

Finally, be sure to read the section on “iBooks and fonts” if you’re having trouble making the fonts appear.

Specifying alternate fonts

Because not every operating system has the same fonts installed, CSS offers the possibility of specifying multiple alternate choices of fonts. If the first is not available, it will try the second. If the second is not available, it will try the third, and so on.

1 In the CSS, in the `font-family` rule, add multiple font selections, each one separated by a comma, in order of preference. Enclose multiple-word name-`font-family` in parentheses. Specify a generic font style as the last choice.

```
samp {
  font-family: "Optima", "Verdana", sans-serif;
}
```

Generic style names are never enclosed in double quotes. If a font name has only one word, quotation marks are optional. Separate each choice with a comma.

iBooks and fonts

Apple inexplicably crippled font support in iBooks 1.1. While iBooks is perfectly capable of displaying the more than 30 font faces that are pre-installed on the iPad, Apple has for some reason kept iBooks from recognizing those fonts when applied to certain elements, namely: `p`, `div`, `body`, `html`, and `span`. On the other hand, iBooks has no problem understanding fonts applied to non-body text elements like headers (`h1`, `h2`, and so on), list items (`li`), definition lists (`dl`, `dd`, and `dt`), highlighted text (`em`, `strong`, `b`, `i`, `code`, and so on).

One way around this is to enclose body text in an additional non body-text element. For example, you could use a rarely used element like `samp`:

```
<p><samp>Even a criminal is entitled to fair play; and certainly when a man
  who has done no harm has been unjustly treated, he is privileged to do
  his best to right himself.</samp></p>
```

Then apply the desired font to the `samp` selector in the CSS:

```
samp {font-family: "American Typewriter";}
```

You'll find updates on this issue on my website.

Fonts available for ebooks on the iPad

The iPad has 33 font families for English (and similar) alphabets, with some repeats, and a number of variations of each. These are: Academy Engraved LET, American Typewriter, AppleGothic, Arial, Arial Rounded MT Bold, Baskerville, Bodoni 72, Bodoni 72 Oldstyle, Bodoni 72 Smallcaps, Bradley Hand, Chalkduster, Cochin, Copperplate, Courier, Courier New, DBLCDTempBlack, Didot, Futura, Georgia, Gill Sans, Helvetica, Helvetica Neue, Hoefler Text, Marker Felt, Optima, Palatino, Papyrus, Party LET, Snell Roundhand, Times New Roman, Trebuchet MS, Verdana, and Zapfino. (For information on non-English fonts, see “Using non-English fonts” further ahead in this chapter.)

I hope this isn’t overkill, but I’m going to go through each iPad font one-by-one both to show you what each font looks like and to give you the required name for using it. For other ereaders, you’ll have to either embed the font or offer alternates (or both!).

For *Academy Engraved LET*, you can either use that name, in quotes, or use `AcademyEngravedLETPlain` with no spaces, and the quotes are optional. There are no additional styles.

Academy Engraved LET Plain: Most men, even in this comparatively free country,

For *American Typewriter*, you can either use that name, in quotes, or use `AmericanTypewriter` with no spaces, and optional quotes. For bold, either use `AmericanTypewriter-Bold`, or add `font-weight: bold` to a rule that already contains the `font-family` declaration.

American Typewriter: Some of you, we all know, are poor, find it hard to live

For *AppleGothic*, use that name, with optional quotes. There are no additional styles. Note that there is never a space.

AppleGothic: I sometimes wonder that we can be so frivolous,

For *Arial*, use the single word font name, and optional quotes. For bold, italic, and bold-italic, respectively, use *Arial-BoldMT*, *Arial-ItalicMT*, or *Arial-BoldItalicMT* respectively, or add *font-weight* and/or *font-style* definitions to the rule containing the *font-family* declaration.

Arial: The mass of men lead lives of quiet desperation.

For *Arial Rounded MT Bold*, use *ArialRoundedMTBold*, with no spaces. No other font styles may be applied.

ArialRoundedMTBold: One farmer says to me, “You cannot live on vegetable food solely,

For *Baskerville*, use the single word name, quotes optional. For bold, italic, and bold-italic, use *Baskerville-Bold*, *Baskerville-Italic*, or *Baskerville-BoldItalic*, respectively, quotes optional. Or add *font-style* and/or *font-weight* definitions to the same rule that contains the *font-family* declaration.

Baskerville: One farmer says to me, “You cannot live on vegetable food solely,

For *Bodoni 72*, use that name in quotes, or *BodoniSvtyTwoITCTT-Book* with optional quotes. For bold or italics, use *BodoniSvtyTwoITCTT-Bold* or *BodoniSvtyTwoITCTT-BookIta* respectively, or add *font-weight* or *font-style* definitions to the same rule that contains the *font-family* definition. There is no bold-italic.

Bodoni 72: The whole ground of human life seems to some to have been gone over by their predecessors,

For *Bodoni 72 Oldstyle*, use that name in quotes, or `BodoniSvtyTwoOSITCTT-Book` with optional quotes. For bold or italic, use `BodoniSvtyTwoOSITCTT-Bold` or `BodoniSvtyTwoOSITCTT-BookIt`, respectively, or add `font-weight` or `font-style` to the rule containing `font-family` declaration.

Bodoni 72 Oldstyle: The whole ground of human life seems to some to have been gone over by their predecessors,

For *Bodoni 72 Smallcaps*, use that name in quotes, or `BodoniSvtyTwoSCITCTT-Book` with optional quotes. If you apply `font-variant: small-caps`, you'll get the computer generated version, which is not nearly as nice as this special version. No additional styles are available.

BODONI SMALLCAPS: WE MIGHT TRY OUR LIVES BY A THOUSAND SIMPLE TESTS

For *Bradley Hand*, use that name in quotes, or `BradleyHandITCTT-Bold` with optional quotes. Despite the name, there is neither an additional bold nor an additional normal style available.

Bradley Hand: I think that we may safely trust a good deal more than we do.

For *Chalkduster*, use that name with optional quotes. No additional styles are available.

Chalkduster: Let us consider for a moment what most of the trouble and anxiety which I have referred to is about,

For *Cochin*, use that name with optional quotes. For bold, italic, or bold-italic, use `Cochin-Bold`, `Cochin-Italic`, or `Cochin-BoldItalic`, or add the desired `font-weight`, `font-style` or both to the same rule that contains the `font-family` definition.

Cochin: Let us consider for a moment what most of the trouble and anxiety which I have referred to is about,

For *Copperplate*, use that name with optional quotes. For bold, use **Copperplate-Bold** or add `font-weight: bold` to the rule that contains the `font-family` declaration.

COPPERPLATE: BY THE WORDS, NECESSARY OF LIFE, I MEAN
WHATEVER, OF ALL THAT MAN OBTAINS BY HIS OWN
EXERTIONS,

For *Courier*, use that name with optional quotes. You can also use the generic name, `monospace` (without quotes). For bold, oblique, or bold-oblique, use **Courier-Bold**, **Courier-Oblique**, or **Courier-BoldOblique** respectively, or add `font-weight: bold`, `font-style: oblique`, or both to the declaration that contains the `font-family` declaration. Note that oblique is slightly darker than italic.

Courier: By the words, necessary of life, I
mean whatever, of all that man obtains by his
own exertions,

For *Courier New*, use that name in quotes, or **CourierNewPSMT** with optional quotes. For bold, italic, and bold-italic, use **CourierNewPS-BoldMT**, **CourierNewPS-ItalicMT**, or **CourierNewPS-BoldItalicMT**, respectively, or add the `font-weight` or `font-style`, or both definitions to the rule that contains the `font-family` declaration.

New Courier: By the words, necessary of life, I
mean whatever, of all that man obtains by his
own exertions,

Personally, I have a hard time telling Courier and New Courier apart.

For *DB LCD Temp*, use **DBLCDTempBlack** with optional quotes.

DBLCDTempBlack: ThE grAnd NEcEssITy. ThEn. For our
bodIEs. Is To kEEp wArm. To kEEp ThE vITAl hEAT IN us.

For *Didot*, use that name with optional quotes. For bold or italic, use **Didot-Bold** or *Didot-Italic*, or add **font-weight** or **font-style** definitions to the rule that contains the **font-family** declaration. There doesn't appear to be a bold-italic variation.

Didot: Most of the luxuries, and many of the so-called comforts of life, are not only not indispensable,

For *Futura*, use that name or **Futura-Medium** with optional quotation marks. For *Futura Medium Italic*, use **Futura-MediumItalic** with optional quotation marks, or add **font-style: italic** to the rule containing the Futura **font-family** declaration. For *Futura Condensed Extra Bold*, use **Futura-CondensedExtraBold** with optional quotes. No other styles appear to be available.

Futura-Medium: When a man is warmed by the several modes which I have described, what does he want next?

Futura-CondensedExtraBold: I do not mean to prescribe rules to strong and valiant natures, who will mind their own affairs whether in heaven or hell,

For *Georgia*, use that name with optional quotes. For bold, italic, or bold-italic, use **Georgia-Bold**, *Georgia-Italic*, or **Georgia-BoldItalic** or add **font-weight**, **font-style**, or both to the rule that contains the **font-family** declaration.

Georgia: In any weather, at any hour of the day or night, I have been anxious to improve the nick of time,

For *Gill Sans*, use that name with quotes, or **GillSans** with optional quotes. For bold, italic, or bold-italic, use **GillSans-Bold**, *GillSans-Italic*, or **GillSans-BoldItalic** or add the **font-weight** or **font-style** or both to the rule that contains the **font-family** declaration.

Gill Sans: I long ago lost a hound, a bay horse, and a turtle dove, and am still on their trail.

For *Helvetica*, use that name with optional quotes or use the generic name, **sans-serif** (without quotes). For bold, italic, or bold-italic, use **Helvetica-Bold**, **Helvetica-Oblique**, or **Helvetica-BoldOblique**, or add the **font-weight**, **font-style** or both to the rule that contains the **font-family** declaration.

Helvetica: I have looked after the wild stock of the town, which give a faithful herdsman a good deal of trouble by leaping fences;

For *Helvetica Neue*, use that name with quotes or **HelveticaNeue** with optional quotes. For bold, italic, or bold-italic, use **HelveticaNeue-Bold**, **HelveticaNeue-Italic**, or **HelveticaNeue-BoldItalic** or add the **font-weight**, **font-style** or both to the rule containing the **font-family** definition.

Helvetica Neue: I have looked after the wild stock of the town, which give a faithful herdsman a good deal of trouble by leaping fences;

For *Hoefler Text*, use that name with quotes or **HoeflerText-Regular** with optional quotes. For bold, italic, and bold-italic, use **HoeflerText-Black**, **HoeflerText-Italic**, and **HoeflerText-BlackItalic**, respectively, or add **font-weight: bold** (not black) and/or **font-style: italic** to the rule that contains the **font-family** declaration.

HoeflerText-Regular: Finding that my fellow-citizens were not likely to offer me any room in the court house, or any curacy or living anywhere else,

For *Marker Felt*, use that name in quotes, or **MarkerFelt-Thin** with optional quotes. For bold, use **MarkerFelt-Wide** with optional quotes or add **font-weight: bold** to the rule that contains the **font-family** declaration.

Marker Felt: I have always endeavored to acquire strict business habits; they are indispensable to every man.

For *Optima*, use that name or **Optima-Regular** with optional quotes. For bold, italic, or bold-italic, use **Optima-Bold**, **Optima-Italic**, or **Optima-BoldItalic** or add **font-weight**, **font-style**, or both to the rule containing the **font-family** declaration.

Optima Regular: I have thought that Walden Pond would be a good place for business, not solely on account of the railroad and the ice trade;

For *Palatino*, one option is to create a style definition that doesn't declare or inherit any font family. To declare it explicitly, use **Palatino** or **Palatino-Roman** with optional quotes. For bold, italic, or bold-italic, use **Palatino-Bold**, **Palatino-Italic**, or **Palatino-BoldItalic** or add **font-weight**, **font-style**, or both to the rule containing the **font-family** declaration.

Palatino: As this business was to be entered into without the usual capital, it may not be easy to conjecture where those means, that will still be indispensable to every such undertaking, were to be obtained.

For *Papyrus*, use that name with optional quotes. You can also use the generic name, **fantasy** (without quotes). No other styles are available.

Papyrus: As this business was to be entered into without the usual capital, it may not be easy to conjecture where those means, that will still be indispensable to every such undertaking, were to be obtained.

For *Party LET*, use that name in quotes, or **PartyLetPlain** with optional quotes. No other style variations are available.

Party LET: As this business was to be entered into without the usual capital, it may not be easy to conjecture where those means, that will still be indispensable to every such undertaking, were to be obtained.

For *Snell Roundhand*, use that name in quotes, `SnellRoundhand` with optional quotes, or the generic name `cursive` without quotes. For bold, use `SnellRoundhand-Bold` or add `font-weight: bold` to the rule that contains the `font-family` declaration. No other styles are available.

Snell Roundhand: A man who has at length found something to do will not need to get a new suit to do it in; for him the old will do,

For *Times New Roman*, use that name in quotes, `TimesNewRomanPSMT` with optional quotes, or the generic name `serif` without quotes. For bold, italic, or bold-italic, use `TimesNewRomanPS-BoldMT`, `TimesNewRomanPS-ItalicMT`, or `TimesNewRomanPS-BoldItalicMT`, or add `font-weight`, `font-style` or both to the rule that contains the `font-family` declaration.

Times New Roman: I have always endeavored to acquire strict business habits; they are indispensable to every man.

For *Trebuchet MS*, use that name in quotes, or `TrebuchetMS` with optional quotes. For bold, italic, or bold-italic, use `TrebuchetMS-Bold`, `TrebuchetMS-Italic`, or `TrebuchetMS-BoldItalic` or add `font-weight`, `font-style`, or both to the rule that contains the `font-family` declaration.

Trebuchet MS: When I ask for a garment of a particular form, my tailoress tells me gravely, "They do not make them so now,"

For *Verdana*, use that name with optional quotes. For bold, italic, or bold-italic, use `Verdana-Bold`, `Verdana-Italic`, or `Verdana-BoldItalic`, or add `font-weight` and/or `font-style` definitions to the rule that contains the `font-family` declaration.

Verdana: When I ask for a garment of a particular form, my tailoress tells me gravely, "They do not make them so now,"

For *Zapfino*, use that name with optional quotes. No other style variations are available.

*Zapfino: The childish and savage taste of men
and women for new patterns keeps how many
shaking and squinting through kaleidoscopes that
they may discover the particular figure which this
generation requires today.*

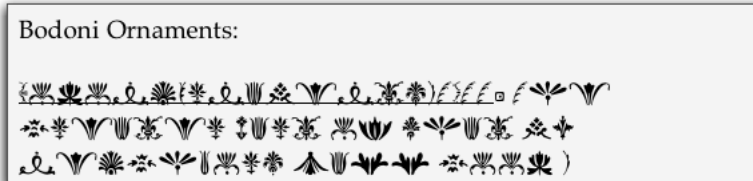
» Tips «

At press time, there was a bug in the iBooks app that caused the font family to be forgotten if you apply a bold, italic, or bold-italic style to the same text in a separate rule. The workaround is to declare both the font family and the style in the same rule.

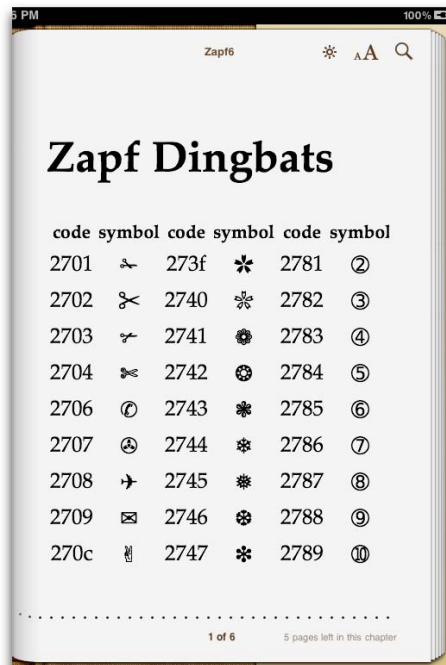
Ornaments, dingbats, and symbols

There are two fonts available on the iPad for ornaments and dingbats, Bodoni Ornaments and Zapf Dingbats, respectively.

The first is applied just like any other font. That is, for Bodoni Ornaments, use that name in quotes or `BodoniOrnamentsITCTT` with optional quotes. No additional styles are available.



Zapf Dingbats works differently. Each dingbat has its own special Unicode encoding, which you can consult at <http://en.wikipedia.org/wiki/Dingbat>. For example, to insert an airplane symbol in your ebook, you would type `&#x`, followed by the symbol's Unicode encoding (in this case 2708), followed by a semi-colon: `✈`



To get a dingbat, use the proper Unicode code. You don't need to choose the font itself.

Indeed, you are not limited to the symbols in the Zapf Dingbats. The iPad can display most of the character entities supported by HTML. That includes all of the characters in most Western-European languages, like à, ç, and ñ, as well as mathematical symbols, punctuation, and shapes and arrows. You can find a full table of these entities at <http://www.elizabethcastro.com/html/extras/entities.html>.

The screenshot shows an iPad screen with a table titled "Entities for mathematical and technical characters (including Greek)". The table has four columns: Entity, Number, Symbol, and Description. It lists various mathematical symbols and their corresponding HTML entities and decimal numbers.

Entity	Number	Symbol	Description
°	°	°	degree sign
÷	÷	÷	division sign
½	½	½	vulgar fraction one half = fraction one half
¼	¼	¼	vulgar fraction one quarter = fraction one quarter
¾	¾	¾	vulgar fraction three quarters = fraction three quarters
≥	≥	≥	greater-than or equal to
≤	≤	≤	less-than or equal to
−	−	-	minus sign
²	²	²	superscript two = superscript digit two = squared
³	³	³	superscript three = superscript digit three = cubed
×	×	×	multiplication sign
ℵ	ℵ	ℵ	alef symbol = first transfinite cardinal (note)
∧	∧	∧	logical and = wedge
∠	∠	∠	angle
≈	≈	≈	almost equal to = asymptotic to
∩	∩	∩	intersection = cap
≅	≅	≅	approximately equal to
∪	∪	∪	union = cup
∅	∅	∅	empty set = null set = diameter
≡	≡	≡	identical to

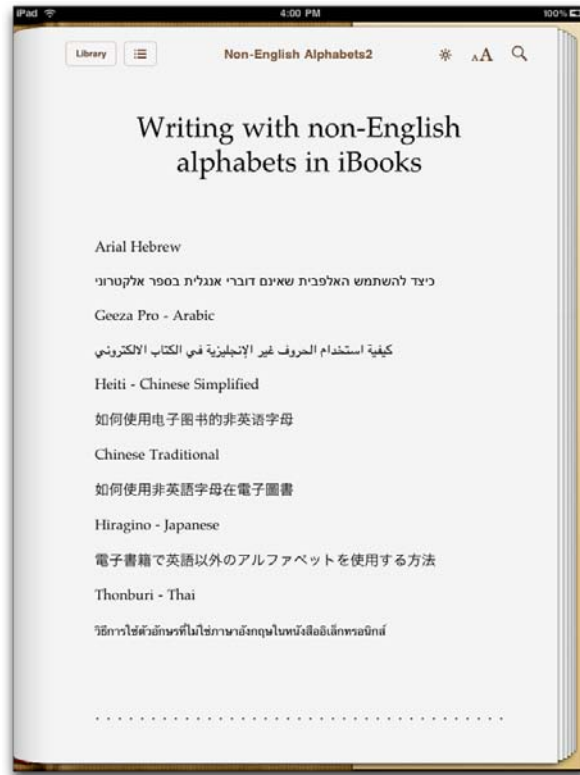
Character entities allow you to enter characters you might not have on your keyboard. They work just fine in EPUB.

using non-English fonts

The iPad also has several non-English fonts that you can use in ebooks to be displayed in iBooks. These are Arial Hebrew, Geeza Pro for Arabic, Heiti J, Heiti K, Heiti SC and Heiti TC for Chinese Simplified and Traditional, Hiragino Kaku Gothic ProN and Hiragino Mincho ProN for Japanese, and Thonburi for Thai.

There's nothing special you need to do in order to use these fonts, indeed, explicitly specifying them as you might a decorative font won't magically transliterate your text.

Instead, enter the text as you normally would, making sure to save the XHTML document in UTF-8 format. Then when you create your EPUB document, the characters will appear properly.



Hopefully I haven't written anything terrible here: I just used Google Translate to translate the phrase "How to use non-English alphabets in an electronic book".

Embedding fonts

It's important to remember that a program can only display characters in a given font if the system on which the ebook is running has access to the font. This is as true for websites as it is for ebooks. So, it doesn't matter what fonts you, the designer, have access to on your own computer; if your reader doesn't have the font, it won't display properly on their system.

The fonts that we've talked about so far in this chapter are all included with the iPad, but that doesn't mean that they're available on Windows, Mac, or other mobile devices.

CSS has long had a system for embedding font information in websites (and now ebooks), but it wasn't until 2009 that there was some agreement on a format, and the practice began to become more commonplace. In fact, as ever, there are

several formats for fonts to be embedded in a website, but the most common are OpenType (OTF), Web Open Font Face (WOFF), and Scalable Vector Graphics (SVG). The EPUB specification recommends using OpenType fonts. You can use a program like FontForge to convert TrueType fonts to OpenType format (with the .otf extension).

Unfortunately, support for font embedding is still not the norm. iPad only supports font embedding of SVG fonts, and iBooks currently doesn't even support that. At press time there was still no way to embed a font in an ebook for iBooks. I'll post any updates to my website.

Getting fonts

There are more and more web font services to be found online. Kernest.com offers a number of free and licensed fonts. Other popular paid and free services include Typekit and Typotheque. Note that font licenses don't always permit use in EPUB files. Check first to be sure. Once you have the font files, create a folder for them, named *fonts*, within the OEBPS folder. InDesign can embed fonts for you upon export to EPUB.

Importing embedded fonts into a style sheet

Once the files are in your EPUB directory, call them into your CSS style sheet by placing code like the following at the top of the style sheet.

1 At the top of your style sheet, before you've used any fonts in any CSS rules, add an `@font-face` for each font you want to embed.

```
@font-face {  
    font-family: 'Travelogue';  
    src: url('fonts/travelogue.otf');  
}
```

Note that in this example, I'm importing the font named "Travelogue" and the corresponding OpenType font file *travelogue.otf* is in the folder named *fonts* within the OEBPS folder.

Then call the embedded font just like any other, as described in "Specifying fonts by name", earlier in this chapter.

Controlling text alignment

Print fiction has historically been displayed with justified text, that is, all the lines of text are adjusted with extra spaces so that they end evenly on both sides. Unfortunately, to do this in a way that looks half decent, you need a robust hyphenation dictionary. Otherwise, particularly long words can cause unsightly rivers of white space that flow through your pages.

Some ereaders let the user control the alignment of an ebook's text. The iBooks app displays justified text by default. Users can go to Settings, click iBooks and choose OFF next to Full Justification to use the settings specified by the book designer. Unfortunately, that setting is a bit hidden. Even worse, iBooks has a bug that keeps it from following the `text-align` property in the CSS.

Luckily, there is a workaround. It turns out that iBooks 1.1 will pay attention to the CSS `text-align` property if and only if there is a second element, within the main `p` element, that contains the text.

1 Contain the body text within every single `p` element in another element. If you're not specifying the font, use `span`. If you do want to specify the font, use `samp`, as described in "iBooks and fonts". You can use the GREP expression, `<p>(.)</p>` to find all of your `p` elements (add the `class` to that expression if necessary), and then replace it with `<p><samp>\1</samp></p>` or `<p><span>\1</span></p>`, depending on whether you're applying font family info or not.

```
<p><span>Here is the text.</span></p>
```

2 Specify the alignment for the `p` elements in the CSS.

```
p {text-align: left}
```

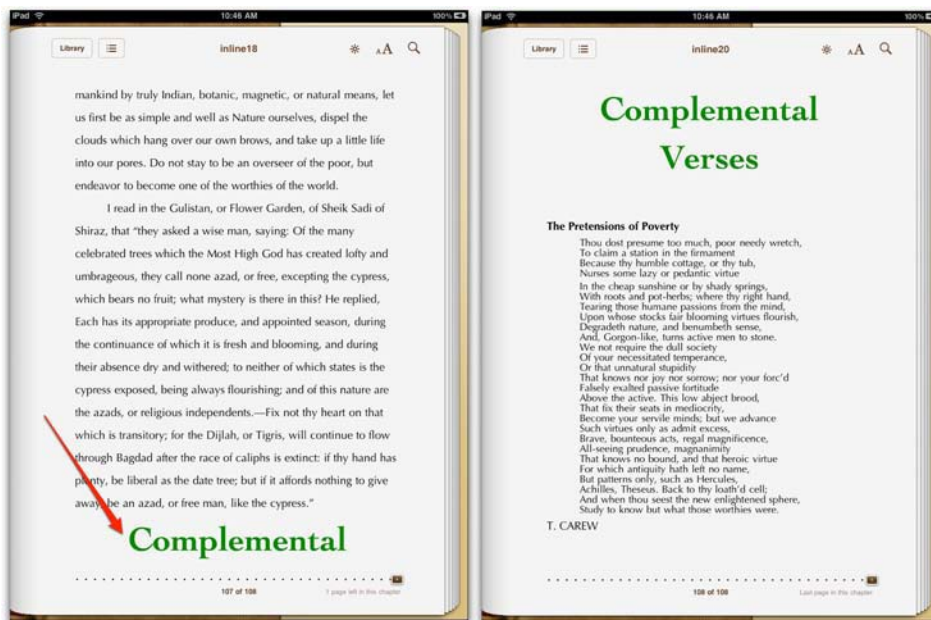
If you do use `samp`, be sure to set the font in the CSS as well. If you use `span`, no extra CSS is required for the `span` selector.

When Apple fixes this bug in iBooks, the workaround will no longer be necessary, but leaving it in shouldn't create problems in future, less-buggy ereaders.

Keeping elements together

The iBooks app doesn't currently support the standard CSS page element controls, like `page-break-before`, `page-break-after`, and `page-break-inside`. Neither does it support the standard CSS `orphans` and `widows` properties that can keep individual lines from appearing at the top and bottom of pages. Some other readers, including Barnes & Noble's Nook, do support the `orphans` and `widows` properties and even have default values set. How then can you best control the flow of text to keep headers from showing up alone at the bottom of pages, captions from being separated from their illustrations, and the last line of a paragraph from starting a new page?

Until iBooks supports the real page-break controls, you can exert a fair amount of force with one special value of the `display` property: `inline-block`. This value forces iBooks to view the affected content much as if it were one discrete item, like an image. So, if you apply `display: inline-block` to a `div`, all of the contents of that `div` are treated as a unit, and are kept together on a page—as much as possible—often starting a new page in the process. It is a roundabout way of creating a page break, but it's definitely better than nothing.



On the left, without `inline-block`, the header starts at the bottom of the page.
Ew! On the right, thanks to `inline-block`, the header begins a new page.

Controlling a header's position

You don't really ever want a header to begin at the bottom of the page. At the very least, it should be followed by one paragraph, and sometimes, perhaps more than that.

- 1 Insert an opening `div` element with an appropriate `class` before the header and a closing `div` element after the paragraph that should be kept together with the header.

```
<div class="keep">
<h1>Complemental Verses</h1>
<p class="quote">The Pretensions of Poverty
    Thou dost presume too much, poor needy wretch,
    To claim a station in the firmament...</p></div>
```

- 2 Create a rule in the CSS that applies to the `div` and `class` you created in the previous step, and set the `display` property to `inline-block`.

```
div.keep {
    display: inline-block;
}
```

Keeping captions with their images

Another common set of elements that you might want to keep together are captions with their images.

- 1 Insert an opening `div` element with an appropriate `class` before the `img` element, and a closing `div` element after the paragraph element that contains the caption.

```
<div class="keep">
    
    <p class="caption">This is a reproduction of Thoreau's House at Walden
        Pond.</p>
</div>
```

2 Create a rule in the CSS that applies to the `div` and `class` you created in the previous step, and set the display property to `inline-block`. (Of course, if you already created the rule in the previous section, you don't have to repeat this step.)

```
div.keep {  
    display: inline-block;  
}
```

Setting widows and orphans properties

Although the iBooks app does not currently support the CSS `widows` and `orphans` properties, other ereaders, like Barnes & Noble's Nook and Adobe Digital Editions, do. It's probably a good idea to include the `widows` and `orphans` properties in your ebooks so that they look as good as possible in those ereaders that support them.

1 To insist that at least two lines of a paragraph appear together at the bottom of a page, type `orphans: 2;` for the selector that controls the text in your ebook.

```
p {  
    orphans: 2;  
}
```

2 To insist that at least two lines of a paragraph appear together at the top of a page, type `widows: 2;` for the selector that controls the text in your ebook.

```
p {  
    widows: 2;  
}
```

Of course, you can insist on any number of minimum number of lines at the end or beginning of a page.

Setting page break options

The CSS controls for page breaks are much more powerful than the `display: inline-block` workaround discussed earlier. Unfortunately, iBooks does not yet support them. However, since other ereaders do, it's probably a good idea to start to use them so that your EPUB documents will view properly—and at their best—as soon as possible.

- 1 Use `page-break-before: always` to have an element begin on a new page. This might be the case of a chapter title or a major sub-title.
- 2 Use `page-break-before: avoid` to keep an element from beginning on a new page, unless absolutely necessary. This might be the case of a caption that should stay with the image that precedes it.
- 3 Use `page-break-after: always` to have the element be the last item on the page, even if it means a lot of empty white space.
- 4 And use `page-break-after: avoid` to discourage a page break after an item. This might be a good idea for images or minor sub-heads that should not appear alone at the bottom of a page.
- 5 Use `page-break-after: auto` and `page-break-before: auto` to let the ereader determine the best place to break a page. (These are the defaults.)
- 6 Finally, use `page-break-inside: avoid` to keep an element from being divided by a page break. You might use this rule on a table, list, or some other element that should not be divided across various pages. (This is the CSS property that `display: inline-block` roughly approximates.)

Remember that any time you create manual page breaks, you're liable to create extra blank space in your ebook.

Further, remember that page break properties are always dependent on the content in the book. If a reader's page can only hold 18 lines and your book has a 20 line paragraph, the reader will be forced to break the paragraph onto two pages, no matter what you've chosen for page breaks. I have seen ereaders try other alternatives (like refusing to display the additional content), and that is far worse.

Drop caps and small caps

It's a common practice for the first word in a new chapter to have a drop cap—an initial letter that is bigger than the rest of the line—and for the rest of the word, and sometimes a few additional words, to be in small caps. There are two techniques for designating what should be a drop cap and what should be set in small caps: you can either let CSS do the work for you, or you can explicitly tag what should be affected in the XHTML code (or have InDesign do it for you). The disadvantage of the second technique is that the tags that you must insert in order to mark the first letter and the rest of the line make it difficult to search for the text involved. On the other hand, although the iBooks app can understand the standard, more search-friendly CSS code, not all ereaders do. Nevertheless, especially for ebooks for the iPad, I recommend letting CSS do the heavy lifting. I'll show you both methods.

Having CSS mark the first letter and line

In our example, we want the first paragraph of each chapter to have a drop cap and that the entire first line be formatted in small caps. Instead of hard coding those selections by enclosing the desired text in tags in the XHTML document, we can tell CSS to use the first letter of a given styled paragraph, whatever it may be, or the first line of a paragraph, whatever it may be. This saves time and also keeps the text completely searchable.

- 1 Make sure that the first paragraph of each chapter is tagged in the XHTML document. In our example, this was already done by InDesign.

```
<p class="first-para">When I wrote the following pages, or rather the bulk  
of them, I lived alone, in the woods,...</p>
```

- 2 In the CSS, begin the special rule by appending `:first-letter` to the selector for the first paragraph.

```
p.first-para:first-letter
```

- 3** Specify the formatting that should be applied to the first letter of the first-para paragraphs. The formatting that is particularly important for the drop cap is highlighted below, and includes floating to the left, a larger font size, and some margin values to position the letter properly.

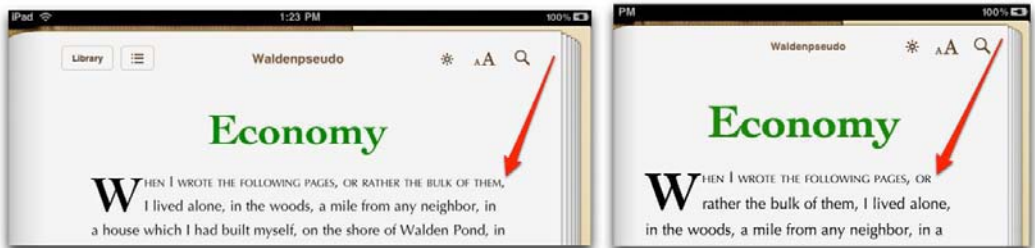
```
p.first-para:first-letter {
  font-family: "Cochin";
  font-weight: bold;
  float: left;
  font-size: 3.5em;
  margin-top: 0.25em;
  margin-right:.05em;
}
```

- 4** Create the special rule for the first line of text by appending `:first-line` to the selector for the first paragraph.

```
p.first-para:first-line
```

- 5** Specify the small caps formatting that should be applied to the entire first line (however long it is, depending on the display).

```
p.first-para:first-line {
  font-variant: small-caps;
}
```



The beauty of letting CSS apply formatting to the first line is that this might change depending on the display, or the orientation of same, as shown here. Further, the text remains completely searchable.

Tagging the first letter and first line explicitly

Because the CSS `first-line` and `first-letter` selectors are not universally supported (currently Adobe Digital Editions, for one, does not support them), you may prefer to tag the drop cap letter and the first few words explicitly in the XHTML.

- 1 Enclose the first letter of the chapter in `span` tags with a special class that indicates that it should be a drop cap.

```
<p class="first-para"><span class="drop-cap">W</span>hen I wrote the  
following pages, or rather the bulk of them, I lived alone, in the woods,...
```

- 2 In the CSS, create a rule for the `.drop-cap` class that includes a larger font-size, left float, and margin adjustments to keep it properly aligned, where necessary. The actual formatting instructions may be identical to those in the previous section, with special attention to the `float`, `font-size`, and `margin` properties.

```
.drop-cap {  
  font-family: "Cochin";  
  font-weight: bold;  
  float: left;  
  font-size: 3.5em;  
  margin-top: 0.25em;  
  margin-right:.05em;  
}
```

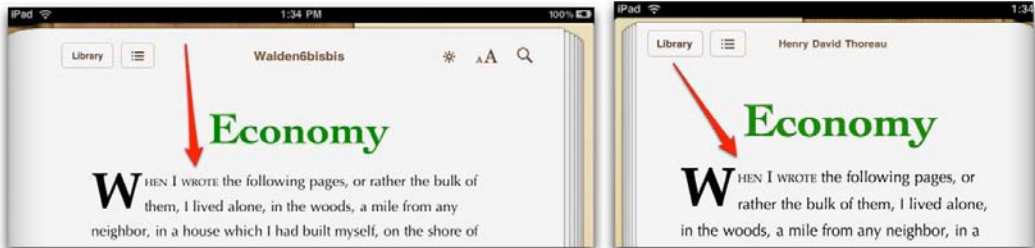
- 3 Enclose the rest of the drop-capped word, and if desired, the next few words, in `span` tags with a special class that indicates that they should be in small caps. Since you can't know what the first line will be, it's better just to choose a few words.

```
<p class="first-para"><span class="drop-cap">W</span><span class="small-  
caps">hen I wrote</span> the following pages, or rather the bulk of them, I  
lived alone, in the woods,...
```

4 In the CSS, create a rule for the `small-caps` class with the `font-variant` property.

```
span.small-caps {
    font-variant: small-caps;
}
```

5 Rezip and test the EPUB file to make sure you like the way the small caps are displayed.



When you mark text explicitly, you have to choose ahead of time what will be marked, and thus it can't vary with the line length.

Tip

Just for the record, Adobe Digital Editions doesn't currently support `font-variant: small-caps`.

Controlling spacing

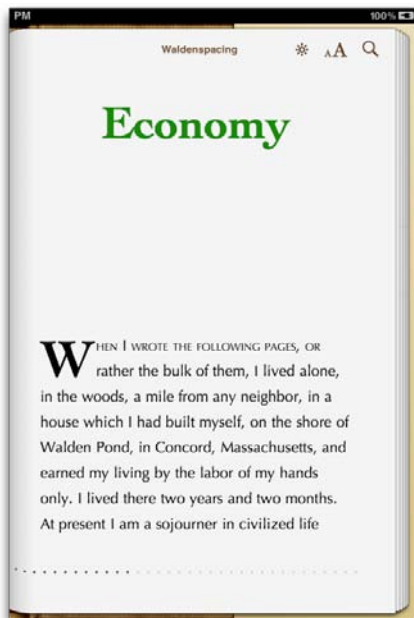
Now that we've got the drop cap and small caps in the first line, it seems like the Chapter title should be set off with more space. Although we defined 2 inches of space after the title in InDesign, the exported value in the CSS is only .33em. This is a curious bug that occurs only in headers and only when you export to EPUB using a TOC style (which is necessary in order to create a navigational TOC). Luckily, it is easily remedied.

1 Open the CSS document for the book. Although we specified 1 pica of space before and 12 picas after, the `h1.header` rule generated by InDesign has a margin property that looks like this:

```
margin: 0.03em 0em 0.33em 0em;
```

2 Change the margin-bottom value to make there be more space between the header and the first paragraph. I also like to simplify the values.

```
margin: 0 0 4em 0;
```



The first page of the chapter looks much better now.

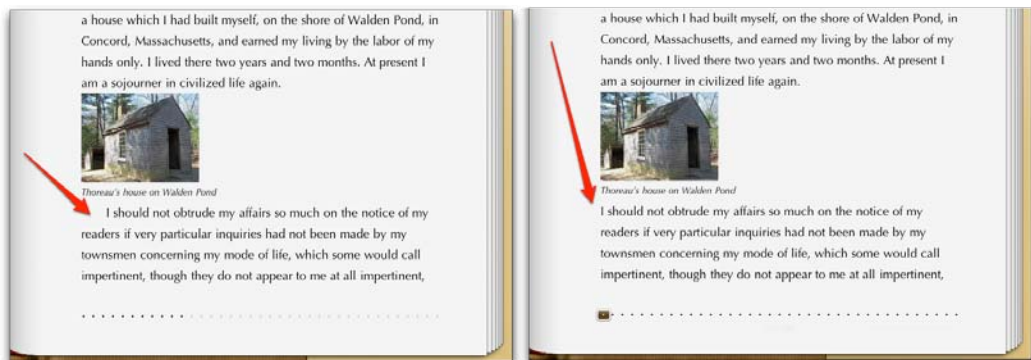
Controlling indents

It's common practice for most but not all paragraphs in books to be indented. In particular, paragraphs that follow illustrations or quotes should generally not be indented. Happily, CSS is quite adept—perhaps even more so than InDesign—at applying indents in the proper places. It does so by looking at which elements *follow each other*, or in CSS parlance, are *siblings*. For example, if a regular paragraph follows another regular paragraph, there should be an indent, but if a regular paragraph follows an illustration, there should not.

In our example, the illustration is an `img` element that is contained in a `div`. And indeed there are no situations in which a `div` should be followed by an indented paragraph. Therefore, we can specify in the CSS that every `body` classed `p` that follows a `div` should not have any text indent.

- 1 Type the selector for the element that directly precedes the element for which you want to remove the text indent, in this case, `div`.
- 2 Then type a plus sign (+) followed by the name of the element that you wish to format, in this case `p.body`.
- 3 Finally, specify the `text-indent` property that should only be applied to `body` classed `p` elements that follow a `div`.

```
div + p.body {text-indent: 0em !important}
```



Before (on left), text is indented after an image. After (on right): no text indent.
 The cool thing about CSS is that you don't have to create or apply a special style. It knows when to apply it!

Formatting short lines

One of the tricky things about ereaders is their notoriously narrow widths. Try reading poetry on an iPhone and you'll quickly understand what I mean. So, it's important to be able to format short lines—that don't automatically wrap—in an attractive fashion. My solution is hanging indents and specialized spacing.

Keep in mind when working with margins and short lines that ems might not be the most appropriate measurement. While 1 or 2 ems on an iPad screen is perfectly reasonable, it's much too much on an iPhone. A better solution is to use percentages so that the indent and margins depend on the screen width. Although this works less well for very wide desktop computer windows, probably anyone reading an ebook on a computer would keep the window narrow enough so that the line length was comfortable, and thus percentage values would still work.

Our quote paragraph style in InDesign had an original margin of 2 picas and 0 text indent. Because the font size was 11pt, upon export, the margin was converted to 2.18em, while the text indent stayed 0:

```
p.quote {  
  text-indent: 0em;  
  margin: 0em 0em 0em 2.18em;  
  ...  
}
```

The text indent for the body paragraph rule was 2 picas, and InDesign converted that to 2em (since the text is set at 12pt):

```
p.body {  
  text-indent: 2em;  
  ...  
}
```

1 First, make sure your short lines are tagged in a consistent way in the XHTML file. The ones in the example text are marked with the quote class. Each line should be tagged separately with no `<br/>` elements.

```
<p class="body">But men labor under a mistake...</p>  
  <p class="quote">Inde genus durum sumus, experiensque laborum,</p>  
  <p class="quote">Et documenta damus qua simus origine nati.</p>
```

- 2 Change the `text-indent` property of the body classed paragraphs to 10% so that it adjusts to the screen in which it is displayed. Be sure to add the `!important` modifier so that ereaders will pay attention.

```
p.body {
    text-indent: 10% !important;
```

- 3 Change the left margin for the quote class to 20%, and give it a negative text indent of -10%. We want the quote line to begin 10% from the left (the same as the text indent of body paragraphs).

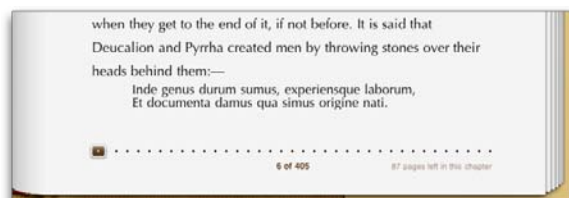
```
p.quote {
    text-indent: -10% !important;
    margin: 0 0 0 20% !important;
```

If the screen is wide enough, the entire line will be displayed on a single line. But if the line wraps, the content on the second and successive lines will get tucked under the first line at 20% from the left.

- 4 I like my quotes to have less leading (space between lines), so I also changed the `line-height` property to 100%, again, adding the `!important` rule to make sure ereaders use it.

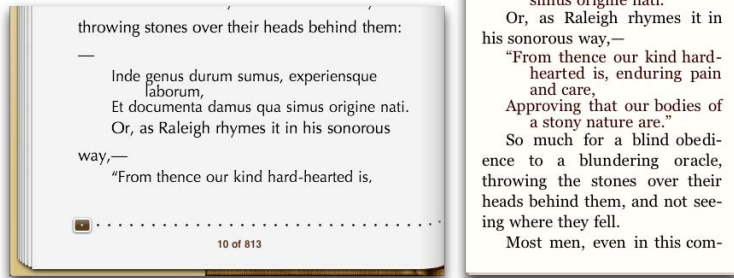
```
p.quote {
    text-indent: -10% !important;
    margin: 0 0 0 20% !important;
    line-height: 100% !important;
```

On the iPad held vertically with a single, large page, the lines are not broken:



Even on the largest iBooks page, the 10% indent is not exorbitant.

On the iPad held horizontally, the lines break, but the hanging indents keep the text coherent. The same happens with Stanza on the iPhone (below right).

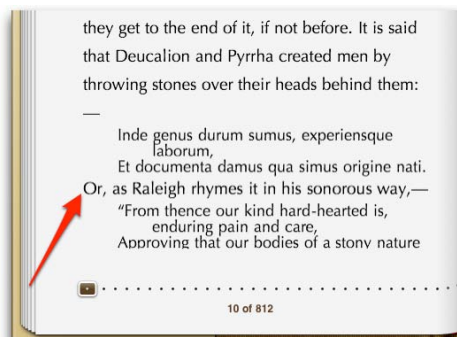


As the screen gets narrower, the indent gets slightly smaller, always taking up 10% and looking proportional in all three venues.

The only problem is that the regular body paragraphs should not have a text indent when they directly follow a quote. We already saw how to handle this in “Controlling indents”: we’ll remove the text indent for body paragraphs that follow a quote paragraph.

5 Add a new rule for body paragraphs that follow a quote paragraph so that there is no text indent.

```
p.quote + p.body {
  text-indent: 0 !important;
}
```



The regular body paragraphs between the quoted ones stand out better now that they’re not indented.

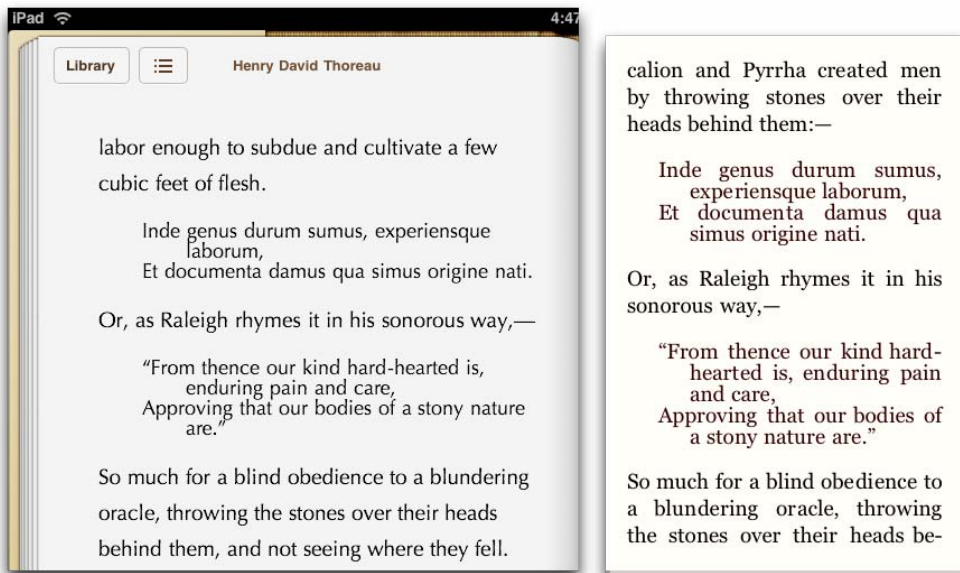
Since the body paragraphs have 1.67em line-height and the quote paragraphs only have 100%, and neither style has space before or after, the quotes get lost in the body of the text. But since each quote is on its own line, we can't simply add space before and after. Indeed, we only want space *before the first quote* and *after the last one*. Once again, we'll take advantage of CSS's powerful sibling selectors.

6 Create a style rule that adds space before a quote only when it follows a body paragraph.

```
p.body + p.quote {margin-top:1em !important}
```

7 Finally, add the same margin rule to our earlier rule that selects body paragraphs when they're preceded by a quote paragraph:

```
p.quote + p.body {
  text-indent: 0 !important;
  margin-top: 1em !important;
}
```



Now the short-lined quotes look great no matter how wide the screen is.

Perhaps the only other modification you might make is reducing the size of the quote text slightly.

Borders and backgrounds

Borders and backgrounds are sort of pseudo-images, as they are created on the fly, generally around your text. They can make portions of your text really stand out. The iBooks app on the iPad supports most of the same CSS that Safari does, that is, most of CSS2 and a chunk of CSS3, including Webkit extensions. Other readers like Barnes & Noble Nook, Ibis Reader, and Stanza support many of these features too. There's quite a lot you can do.

Creating a sidebar

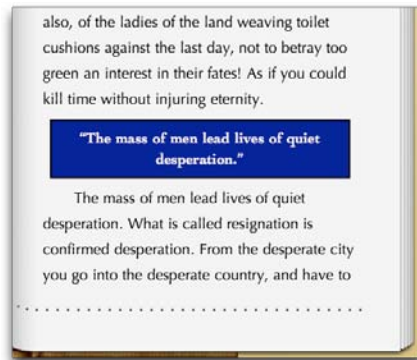
One of the easiest and most useful things you can do with borders and backgrounds is to create a sidebar (or *central bar* as the case may be) to offset a piece of your text and thereby call attention to it.

- 1 Add a `class` to the desired `p` element in your XHTML document. I like to use the word *sidebar*.

```
<p class="sidebar">"The mass of men lead lives of quiet desperation."</p>
```

- 2 The CSS should include margin values to offset the sidebar from the surrounding text, padding values to offset the contents from the border, possibly a percentage width to have it take up just a portion of the screen, dependent on the screen's size, a background color, text color, and border. Remember to add `!important`—particularly to the spacing properties—as well.

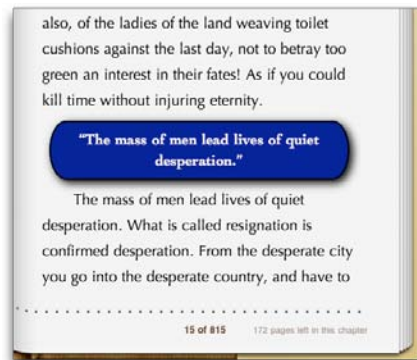
```
p.sidebar {  
  margin: .5em !important;  
  padding: .5em !important;  
  width:30% !important;  
  text-align:center !important;  
  background: #0627a2;  
  border: 2px solid black;  
  color:white;  
  font-family: Cochin, serif;  
  font-weight: bold;  
  text-indent:0 !important;  
}
```



This is just a proof of concept (not a proof of lack of design sense!).

3 If you like, you may wish to explore the Webkit and Mozilla extensions, many of which work in the iBooks app and elsewhere. You can find a complete list of Webkit and Mozilla extensions on my website. For example, you may wish to add rounded corners and a shadow to your sidebar.

```
-webkit-border-radius: 3em 5em;
-webkit-box-shadow: 4px 4px 1em black;
-moz-border-radius: 3em 5em;
-moz-box-shadow: 4px 4px 1em black;
```



There are many more effects you can create, but that would require an entire new book on CSS3 and Webkit extensions.

You can also wrap text around a sidebar, using the technique explained later in this chapter in “Working with images”.

Hyphenation

You only have to open a full-justified ebook on a small screen to see how urgent the topic of hyphenation has become. Unfortunately, perfect solutions are still not at hand. Thankfully, some options are available.

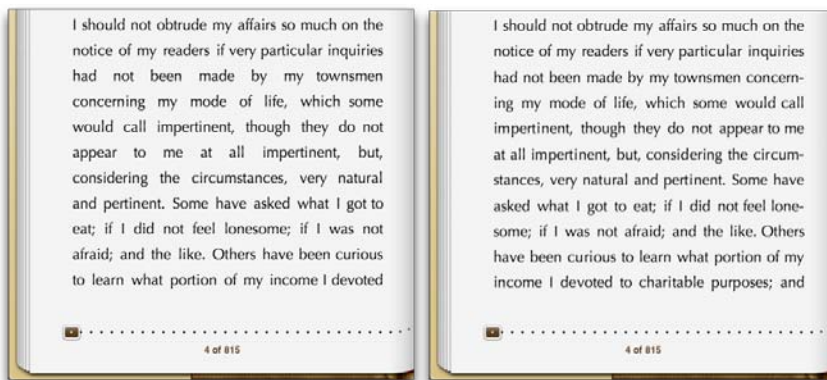
Adding soft hyphens

A soft hyphen is only displayed if the word is broken between two lines, and not if the word occurs in the middle of the line. It seems ideal at first glance. Unfortunately, soft hyphens have two possibly deal-breaking disadvantages. First, they must be hardcoded into the text, either by hand or with a script, and second, because they divide up the actual text, they render it unsearchable: If you're looking for *hyphenation* you won't find *hy­phe­nation*.

1 Add `­` to indicate where a soft hyphen should go. You can also use `­`.

```
<p class="body">I should not ob&shy;trude my affairs so much
on the notice of my readers if very par&shy;tic&shy;u&shy;lar
in&shy;qui&shy;ries had not been made by my towns&shy;men
con&shy;cern&shy;ing my mode of life, which some would call
im&shy;pert&shy;i&shy;nent, though they do not appear to me at all
im&shy;pert&shy;i&shy;nent, but, con&shy;sid&shy;er&shy;ing the
```

You can see how this gets quickly out of hand.



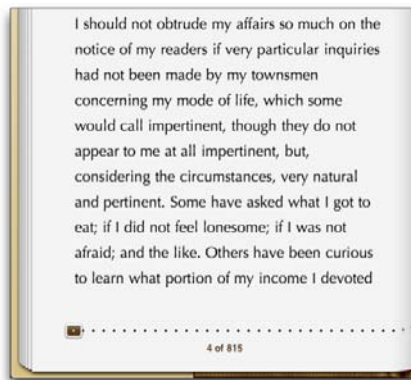
The text does look better when some words are hyphenated (right) on the small iBooks page. On a smaller device, the difference would be even greater.

using left-aligned text

Personally, my inclination is to insist on left-aligned text. While this doesn't solve all hyphenation problems, it is a huge improvement over full-justified and saves the trouble of adding soft hyphens.

1 Make sure that your main body text is left aligned and not fully justified. You can try using `!important` to override any ereader presets (but read tip below).

`p {text-align: left !important}`



This is by no means perfect, but I think it's much better than justified and much simpler than adding hyphens manually.

At some point in the very near future, ereaders will come with build-in, robust hyphenation dictionaries, and this will all be a moot point. Indeed, some, like Stanza, already do. Hurrah!

Tip

Apple's iBooks 1.1 has an important anti-feature. Full justification has been applied to all books by default on the iPad (but not the iPhone) and cannot be overridden by the book designer. To turn off Full Justification, the user has to go to the general iPad Settings. Unless the user has turned off Full Justification, the book designer can choose right or centered alignment, but not left-alignment. It's terrible. I explain how to work around this anti-feature in "Controlling text alignment" earlier in this chapter.

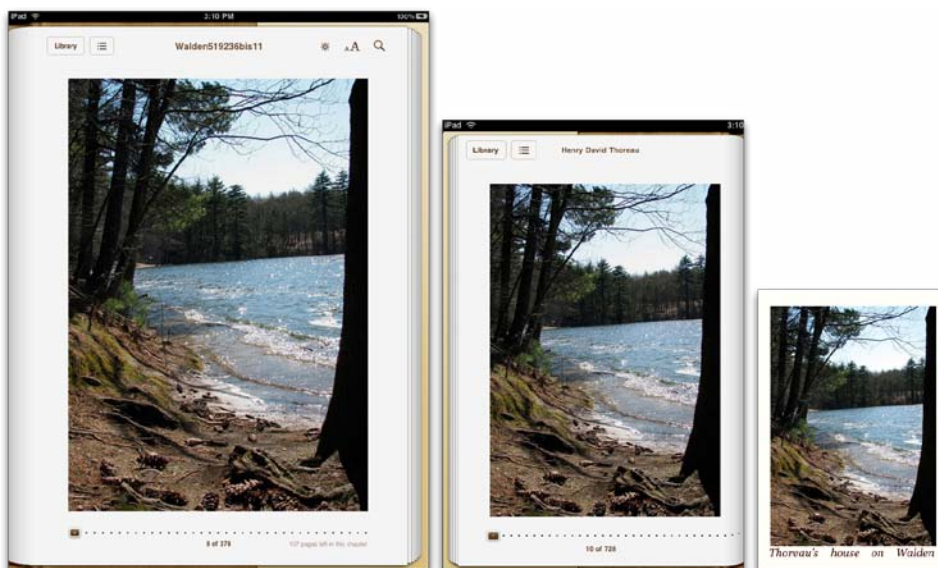
Working with images

While you have already seen how to insert images into an EPUB document with InDesign, there is much more you can do by editing the code directly.

Size

In XHTML and CSS you may be used to sizing an image explicitly by specifying the width and height in pixels. Because ereaders come in many different sizes, this solution is less satisfactory for EPUB documents and can result in an image that is too small in some ereaders and split over multiple pages in others. In addition, many ereaders, and iBooks on the iPad in particular, resize images automatically to fit on a page. However, even on these ereaders and without explicit sizes, particularly tall images are sometimes divided between multiple pages. The effect is not pretty.

If you want an image to fill an iBooks page, the trick is to use images that measure exactly 600 wide by 860 pixels tall. (I call images of this size *magic images*.) Because 600 x 860 is the size of the usable space on a large iBooks page, magic images fill that page, effectively creating a page break. Most ereaders will resize automatically for smaller pages. You could use larger images with the same proportion, but that would be a waste of bandwidth.



The iBooks app automatically resizes images to fit both larger and smaller pages. Even Stanza on the iPhone (right) resizes the image to fit.

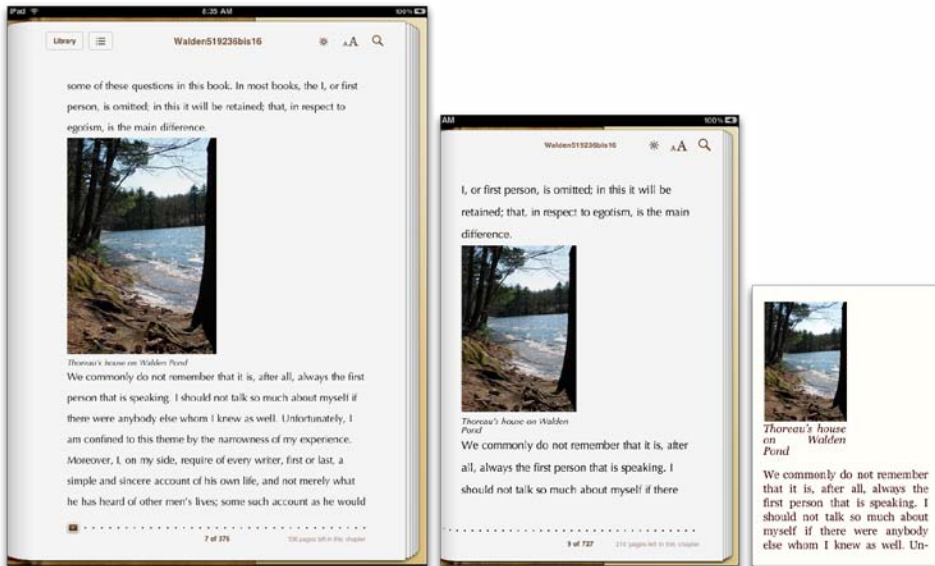
If you want images *smaller* than a full page, use percentage values for the **width** property in CSS to limit the width of the image to a portion of whatever size screen it displays in. When you do this, you have to keep in mind that whatever percentage you use is calculated with respect to the item's *parent element*, not its own original size. Further, in my tests, I've noticed that it doesn't work to apply the **width** directly to the **img** element. But I have found a workaround.

- 1 Create a **div** with a **class** that contains the image that you want to reduce in size. In my example that was exported from InDesign, I had a group that contained the image and the caption, which InDesign exported as a **div** with a **class** of **illustration**.

```
<div class="illustration">
<p class="caption">Thoreau's house on Walden Pond</p></div>
```

- 2 In the CSS, apply the **width** property with the desired percentage to the **div**. (I've used 50%.)

```
div.illustration {width: 50% !important;}
```



Now the image only takes up half the width of the screen, no matter how big or small that screen is. (From left to right: iPad vertical, one page of iPad horizontal, and Stanza on iPhone).

You *should* be able to apply the `width` property directly to the `img` element, according to the CSS specifications, but that doesn't yet work in iBooks.

Note that the iPad measures 1024 by 768 pixels, and since the usable area in iBooks is only about 600 by 860 pixels, it doesn't make any sense to use images that are any bigger than that. Although they'll be resized to physically fit, the actual file itself will be the same size, and will take longer to download and to view. If you use the `width` property to habitually reduce your images to 50% of the page size, make sure the images themselves don't measure any more than 300 pixels wide.

Wrapping text around images

Once you can reduce the size of images to a given percentage of the width of the ereader screen, then you have space to wrap text around those images.

- 1 Make sure the image is enclosed in a `div` element with a `class`. In my example, exported from InDesign, I have a group that contains the image and the caption, that was exported as a `div`.

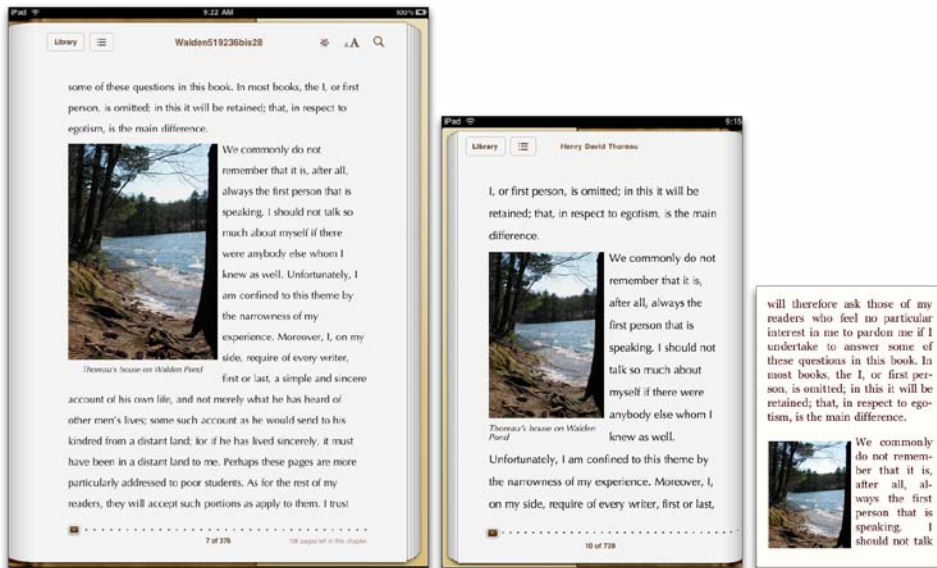
```
<div class="illustration">
<p class="caption">Thoreau's house on Walden Pond</p></div>
```

- 2 Create a rule for the `div` and `class` with a `width` of the desired percentage, the `float` property to the desired direction (left or right), and the `margin` property that inserts a small amount of space between the image and the surrounding text. Remember that the values for `margin` go in the order of top, right, bottom, left. (Because the surrounding text has a line-height of 2em, I have applied that here as well, to keep the image aligned with the text.)

```
div.illustration {
  width: 50% !important;
  float: left;
  margin: 0 .5em 0 0 !important;
  line-height: 2em !important;
}
```

3 Finally, apply a width of 100% to the `img` element. Otherwise, some ereaders will have the image inherit the 50% width, creating an image that is only 25% of the ereader width, leaving a big empty white space (since the `div` is still only 50% of the width). I have also added the `vertical-align: text-top` property to keep the image aligned with the top of the text and a `margin-bottom` of .5em to separate the image from the caption.

```
img {
  width: 100% !important;
  vertical-align: text-top
  margin-bottom: .5em !important;
}
```



The image takes up 50% of the screen, and is nicely padded from the text to the right and the caption underneath. Stanza's division of the image into two pages (right) is not ideal, but is perhaps acceptable, given the extreme size restraints.

Note that the `img` tag is an inline element and must be enclosed in a `div` (or some other block-level) element in order to be valid in XHTML. I have also added a caption, though this is optional.

Wrapping text around sidebars

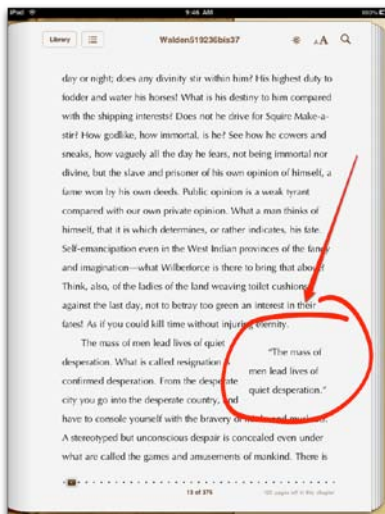
Of course you can float pretty much anything you want, including images, or even other bits of text. In fact, a sidebar is nothing more than one block of text floated next to another.

- 1 Create a class for your sidebar and apply it to the desired paragraph in your XHTML document.

```
<p class="sidebar">“The mass of men lead lives of quiet desperation.”</p>
```

- 2 In the CSS file, apply the `float` and `width` properties to the sidebar. The first can be set to right or left, the latter determines how much of your screen the sidebar will occupy. I also recommend specifying the `margin`, so that there's space between the sidebar and the surrounding text, and the `padding`, so that there's space between the contents of the sidebar and its border.

```
p.sidebar {
  float: right !important;
  width: 30% !important;
  margin: 0 0 0 .5em !important;
  padding: .5em !important;
}
```

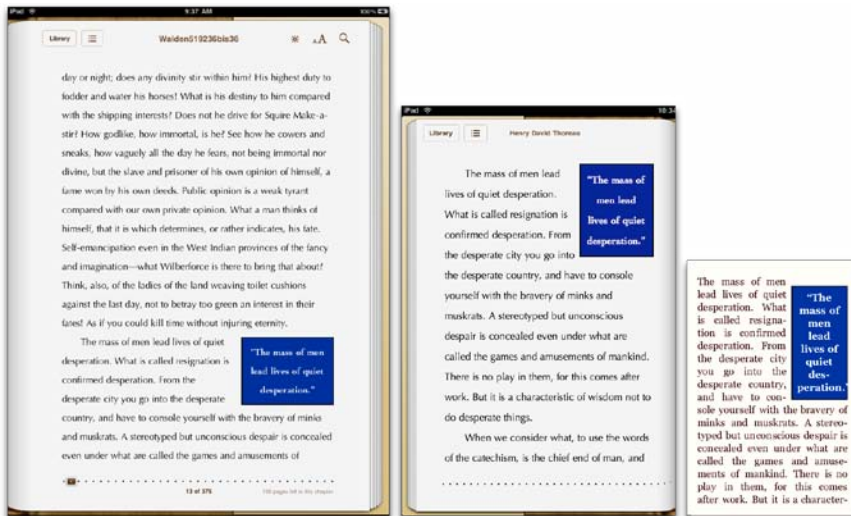


The sidebar is floating but needs some other formatting to help it stand out.

3 If desired, add additional formatting to make the sidebar stand out. In this example, I've added a background color, contrasting text color, and border, and changed the font, boldness, text indent, and text alignment.

```
p.sidebar {
    margin: 0 0 0 .5em !important;
    padding: .5em !important;
    float:right !important;
    width:30% !important;
    text-align:center !important;
    background: #0627a2;
    border: 2px solid black;
    color:white;
    font-family: Cochin;
    font-weight: bold;
    text-indent:0 !important;
}
```

This technique works in all the ereaders I tested it in. You have to be a bit careful about small displays, but as long as you use a *percentage* value for the **width** and keep the text size small, it works pretty well.



The sidebar floats to the right, next to the paragraph which it precedes. (From left to right: vertical iPad, one page of horizontal iPad and Stanza on iPhone).

Creating links

One of the great benefits of electronic books is that you can quickly jump from one section to another via a link, just like on any web page.

When you create a link to a web page, you use the destination web page's location, or URL, in the link code to indicate where a link should bring the visitor. Sometimes, the URL references pages on a different server, and sometimes they are pages on the same site. So it is with EPUB links: you can link to any XHTML file contained within the EPUB and you can also link to external pages on any site on the Internet.

Links in a book, however, tend to be somewhat more finely grained than links to an entire web page. It's more likely that you want to link directly to a particular passage. That is also possible, but it requires creating an *anchor* at the destination to which you can point.

For example, suppose we wanted to have a list of quotations from Thoreau's *Walden*, and then link them to their actual location within the text. I've created a separate XHTML document with the quotes themselves.

To link each quote to its location in the body of *Walden*, we need to create an anchor at the original location.

1 Find the destination that you want the link to bring the reader to.

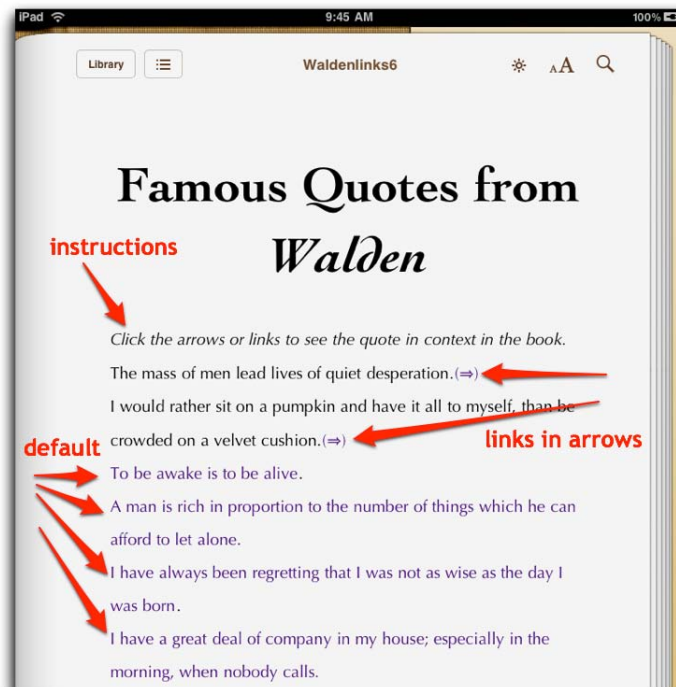
2 Enclose the destination in an `a` element with the `id` attribute equal to a keyword (with no spaces) that identifies the destination. In this example, I've used *pumpkin* as the keyword.

of the Celestial Empire, which Jonathan should be ashamed to know the names of. `<a id="pumpkin">`I would rather sit on a pumpkin and have it all to myself than be crowded on a velvet cushion.`</a>` I would rather ride on earth in an ox cart, with a free circulation, than go to heaven in the fancy car of an excursion train and breathe a malaria all the way.`</p>`

3 Open the document where you want to place the link and add an `a` element that links to the destination you’ve just created. Be sure to reference the document that contains the destination (*Walden_c1.xhtml* in this example) followed by a `#` followed by the keyword you used in the previous step (*pumpkin*).

```
<p><a href="Walden_c1.xhtml#pumpkin">I would rather sit on a pumpkin  
and have it all to myself, than be crowded on a velvet cushion.</a></p>
```

You may also wish to style the links in the CSS to indicate to the reader that they exist. The default in many ereaders is the classic website blue underline. iBooks, however, displays links in a light purple with no text decoration, by default.

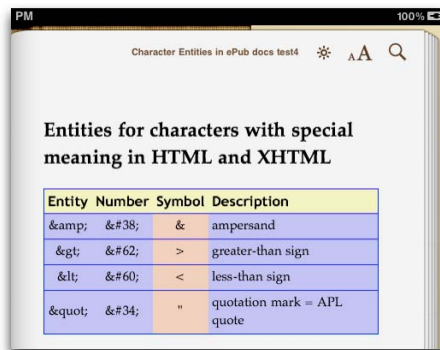


I’m not sure this is obvious enough, and frankly, the blue underline is so ugly, that it is not my preferred choice either. Unfortunately, iBooks doesn’t support hover, focus, or active states. I have experimented with using Zapf Dingbat arrows as a visual clue (first two lines above) without having to resort to underline or highlighting. You might need to use words like “click here” or add instructions in the text until iBooks recognizes other methods of making links obvious but not glaring.

Creating tables

Tables can be the perfect solution for displaying particular kinds of information, and thankfully, XHTML and CSS have generally good support for them. iBooks has a few quirks that we'll have to work around. The principal difficulty with tables, of course, is that they try to pack a lot of information into a small amount of space. The small page on the iPad is only about 530 pixels wide, which is quickly eaten up by multiple columns.

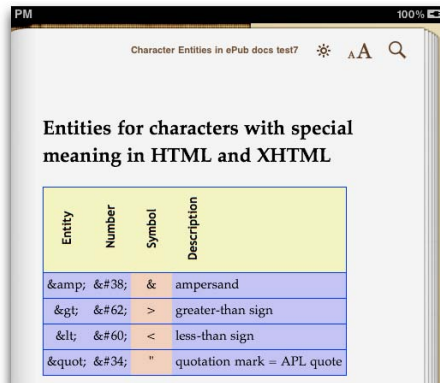
Since iBooks doesn't have a hyphenation dictionary, it tends to choke on words that are too long to fit in a column. Its principal strategy is to widen the column to fit, regardless of the rest of the contents of the column.



Entity	Number	Symbol	Description
&	&	&	ampersand
>	>	>	greater-than sign
<	<	<	less-than sign
"	"	"	quotation mark = APL quote

Notice how the Symbol column is much too big for its contents, except for the title, which is squished in a little too tightly.

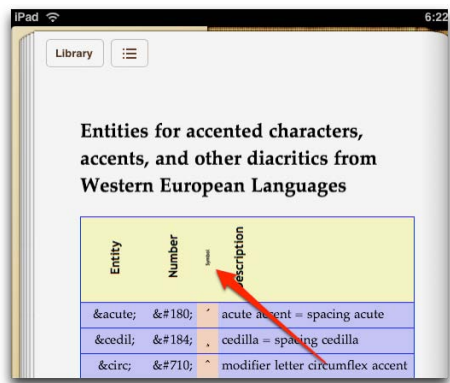
To solve this problem, either use very short headers for your columns, use smaller type, or, as I did in this case, use images placed on their side.



Entity	Number	Symbol	Description
&	&	&	ampersand
>	>	>	greater-than sign
<	<	<	less-than sign
"	"	"	quotation mark = APL quote

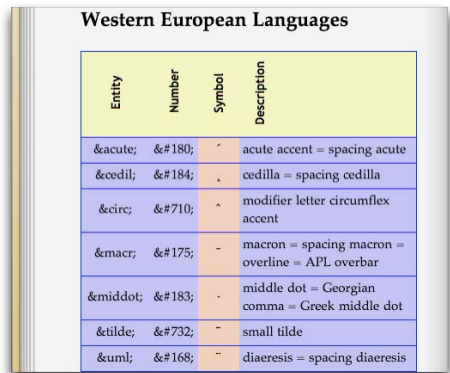
One option is to use images rotated 90°.

The size of each column of a table is automatically determined according to the contents of its largest cell. If there is a particularly large column (like Description, in this example), other columns may shrink to accommodate them. In the second part of my example table, the descriptions are longer and so the column that holds the symbols was reduced in size. In iBooks, the *Symbol* image in the header row was actually shrunk to fit. This is a bug and should not happen (and doesn't in web browsers).



The solution in this case, is to use the `min-width` property to keep each column from collapsing past a certain point. This keeps the *Symbol* column wide enough to support the image (which it should have done on its own).

```
td {min-width: 40px}
```



The `min-width` property makes the *Symbol* column wide enough for the image not to be shrunk.

video in your ebook

Video is in a huge state of flux all over the Internet. You may have noticed the building conflict between Apple and Adobe over the use of Flash versus HTML5 for displaying video files. Apple doesn't want to use Flash because it's a proprietary Adobe format (which is true). Adobe says HTML5 isn't fully baked yet (which is also true).

And then there is also the issue of formats. Unfortunately, the most popular video file formats are all patented, and although nobody is charging anyone to use them yet, they still could at some point. For that reason, Firefox, which is open source itself, has refused to support the H.264 compression, which is Apple's preferred format and used in MP4 and M4V files, and Apple doesn't support Ogg Theora, which is the open source, but purportedly slower, solution.

Since this book concentrates mostly on how to create EPUBs for iBooks, I'll also focus on the kind of video that works with iBooks. Unfortunately, video doesn't work with iBooks in a standard way. While I've been telling you all along that EPUB needs conforming XHTML 1.1 files to be valid—and that continues to be true—it's also true that iBooks can read EPUB documents created from HTML5 documents. And HTML5 documents support video in a format that iBooks recognizes. Unfortunately, I haven't found any other ereaders that support this technique, so you should be wary of using it for books intended for any ereader besides iBooks on the iPad. In addition, even in iBooks, it's something less than rock solid, as you will see, and it won't be accepted for the iBookstore because using HTML5 breaks EpubCheck validation. I offer it to you more as a "look what's coming" rather than a "you can use this right now!" sort of technique.

Finally, it's quite probable that the EPUB format will eventually support and encompass HTML5, and that iBooks support will become more robust, at which time this technique will be perfectly valid. I'll post updates to my website as needed.

Creating your video

The iPad only supports video compressed with H.264 and saved in MP4/M4V format. Luckily, this is relatively easy to achieve with the free QuickTime Player from Apple.

- 1 Open your video with QuickTime Player. You can download the free program from <http://www.apple.com/quicktime/download/>. It can open .avi files (created by many digital cameras), .mov files (older QuickTime files), and .wmv files (a common Windows video format), and others.
- 2 Choose File > Export for Web. Choose one or more of the format options. The iPhone setting will create medium-sized M4V files, the iPhone (cellular) option creates very small files in .3gp format for viewing on cellphones, and the Desktop option creates a larger M4V file. I recommend the first iPhone option.
- 3 The QuickTime Player creates a whole folder full of files for you, only one of which you will need for your EPUB document. Drag the M4V file into the folder that contains your EPUB documents.

Adding code for the video

Again, be aware that this technique currently only works in the iBooks app for the iPad. Nevertheless, there are some techniques you can follow to help it degrade gracefully in ereaders that don't support it.

- 1 Instead of the XHTML document opening that I recommended back in "XHTML and CSS files" in Chapter 3, use the following one instead. There's no point in including the DOCTYPE or XML declaration if you're writing the code in HTML5.

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

- 2 Where you want your video to appear, begin the `video` element, by typing `<video>`. (Note that I haven't been able to make a video appear anywhere except on the first page of an ebook. This is a strange bug that I imagine will get fixed soon.)
- 3 The most important attribute is `src`, which gives the location of your video. The previous section explains which formats are supported. In this example, I used `src="Walden.m4v"`.
- 4 Unless you plan to build the controls for your Video player, you must add `controls="true"` so that your reader can play, rewind, change the volume, and view the video full screen.

5 The default dimensions for a video object in iBooks are 304 x 154 pixels—pretty close to, but not exactly, widescreen proportions. It's probably better to specify the dimensions of your video, which you can find by selecting your video, and choosing File > Get Info in the Finder.

Note that iBooks won't calculate one dimension if you specify just the other. For example, if you specify `width="400"`, you'll get a 400 pixel wide frame, but the height will remain the same, resulting in black bars on either side of the video (unless it was 400 pixels wide to begin with).

Also note that iBooks won't scale your video to fit if you make it too big. When you view a video on the small page (for example, with the iPad held horizontally), the maximum display area is 400 pixels. If you make your player any bigger than that, it may display properly on the single large page, but it will get cut off on the smaller one.

6 Finally, there seems to be a little bug in the autoplay function. It seems like the default behavior is that movies do not play automatically, which is good. However, I've found that after viewing a book and then adding a new book, the old book's video starts playing on its own, even if you close it!! To avoid that, type `autoplay="false"`.

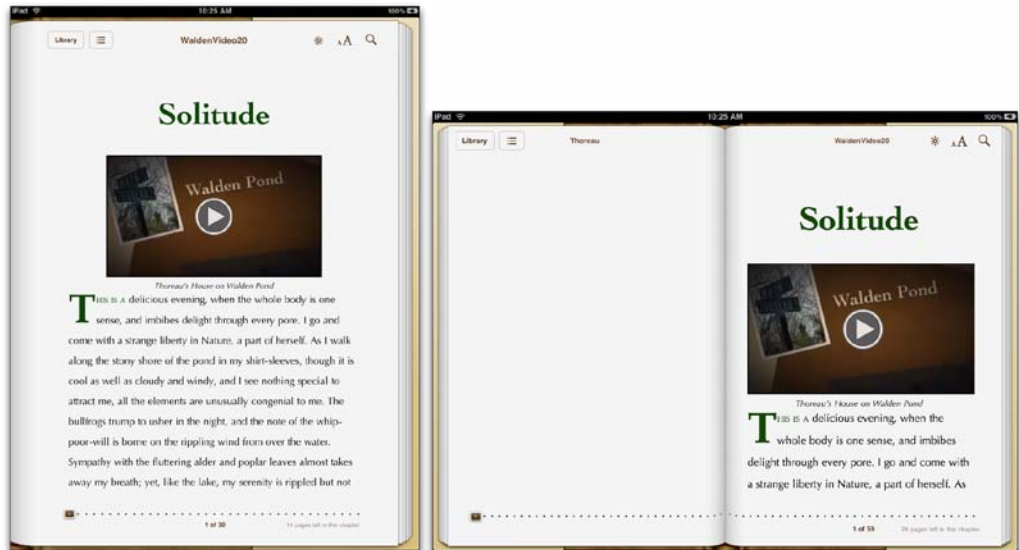
7 Finish the opening video element with a greater than sign: `>`.

8 Next, type what should appear if the video does not. In this example, I've created a link with an image. Most of the ereaders that don't support video *do* support a link to an external site that contains the video.

```
<a href="http://www.elizabethcastro.com/pgw_extras/Walden.m4v"></a>
```

9 Finally, add the closing `</video>` tag. When you're done, the whole thing should look like this:

```
<video src="Walden.m4v" width="320" height="240" autoplay="false">
  <a href="http://www.elizabethcastro.com/pgw_extras/Walden.m4v">Click here to see the video.</a>
</video>
```



If you choose a width of 400 pixels, it fills the page when held horizontally (right), and still looks good on the single larger page.



You can include an image and/or a link for ereaders like Adobe Digital Editions that don't yet support video. (This is just a screenshot of the video with a link to an external version of the video—not the video itself.)

Index

A

- accented characters 151
- Adobe Digital Editions 14
 - cover icon in 114, 115, 117
 - metadata display 108
 - navigational TOC in 98
 - title display in 105
- advanced formatting 129–186
 - adding video 183–186
 - alignment 172
 - backgrounds 169–170
 - borders 169–170
 - drop caps 159–162
 - fonts 138–153
 - hyphenation 171–172
 - images 173–178
 - wrapping text around 175–176
 - indents 164
 - inheritance 135–186
 - insisting on CSS 130
 - keeping elements together 155–158
 - links 179–180
 - orphans and widows 155–158
 - page breaks 155–158
 - short lines 165–168
 - sidebars
 - creating 169–170
 - wrapping text around 177–178
 - small caps 159–162
 - spacing 163
 - tables 181–182
- a element, for links 179
- alignment 154, 172
- alternate fonts 140
- Arabic font 151

- arrows 151
- author, entering in InDesign 81

B

- backgrounds 169–170
- Barebones Software 31
- Barnes & Noble Nook. *See* Nook
- BBEEdit 31, 94
- borders 169–170
- bulleted lists 49, 132

C

- caching on iPad, and title 120
- captions, keeping with images 156–157
- chapter breaks. *See also* page breaks
 - automatic in InDesign 89
- character set. *See* language
- Chinese font 151
- closing slash, add to Word files 43
- CMYK images in EPUB files 82
- content.opf* file 103–119
 - book title 104
 - contents of book 109
 - cover of ebook 115
 - date 106
 - graphic 118
 - guide section 111–112
 - language of book 106
 - manifest section 109–110
 - metadata section 104–119
 - order of book's elements 110
 - spine section 110–111
 - unique id 105

copyright
 adding metadata in InDesign 81
 specifying in *content.opf* file 107
 cover of ebook 113–118
 adding with iTunes 113
 size 113
 cross-references, in InDesign 73–75
 CSS
 and EPUB 14
 `display: inline-block` 155–158
 ensuring ereaders use yours 130
 `first-letter` pseudo selector 159–160
 `first-line` pseudo selector 160–186
 `float` property 175, 177
 for borders and backgrounds 169–170
 generated by InDesign 132–137
 importing embedded fonts 153–186
 inheritance 135–186
 inside EPUB file 96–97
 `min-width` property 182
 moving to separate file in Word 33
 orphans 155–158, 157
 page break properties 155–158, 158
 sibling selectors to control indents 164
 small caps 162–186
 `text-indent` property 165–168
 widows 155–158, 157
 `width` property 174, 176, 178

D

dashes. *See* hyphenation
 date
 in *content.opf* file 106
 InDesign bug 81
`dc:date` element, in *content.opf* file 106
`dc:identifier` element, in *content.opf* file 105
`dc:language` element, in *content.opf* file 106
`dc:title` element, in *content.opf* file 104
 dingbats 150–151
 DRM (Digital Rights Management) 95
 drop caps 159–162
 in InDesign 57–60
`dtb:uid` meta element, in *content.opf* file 105
 Dublin Core Metadata
 elements 106–107
 specification 104

E

ebook. *See* EPUB, iBooks
 sales 7
 vs print vs website 8–13
 embedding fonts 152–155
 in InDesign 84
 in Word 34
 encryption.xml, file 95

EpubCheck, for validation 126–128
 EPUB file 91–128
 advanced formatting 129–186
 cleaning up from InDesign 131–137
 content.opf file 103–119
 copying to iPad 122–123
 description 14
 DRM 95
 editing 124–125
 encryption.xml 95
 files that make up 94–112
 Kindle, converting for 128
 OEBPS folder (contents) 95–96
 organizing contents 119–120
 reziping after edits 120–121
 unzipping 92–93
 validating 126–128
 writing from scratch 91–128
 XHTML and CSS 96–97
 zipping and testing 119–128
 ereaders. *See also* iBooks, Stanza, Adobe Digital Editions
 description 14
 Export Book for Digital Editions, option 83
 Export Book to EPUB, option 87
 exporting to EPUB
 from InDesign CS4 83–86
 from InDesign CS5 87–90
 from Word (roughly) 28–44

F

Filtered HTML, option in Word 29
`first-letter` pseudo selector 159–160
`first-line` pseudo selector 160
 Flash vs HTML5 183
`float` property, in CSS 175, 177
`@font-face` rule 153–154
 fonts in ebooks 138–153
 available for iPad 141–149
 choosing 138–140
 crippled in iBooks 140
 workaround 140
 embedding 152–155
 generic 139
 non-English 151–152
 specify alternate 140
 specify by name 138–139
 specify by style 139

G

GREP 31, 41, 94
 guide, in *content.opf* file 111–112
 and cover of ebook 116



H.264 compression 183
 headers, controlling position 156
 Hebrew font 151
 HTML. *See also* XHTML
 and EPUB 14
 cleaning up Word files 31
 generating with Word 28–30
 in ebooks vs websites 8
 HTML5 vs Flash 183
HTML, XHTML, and CSS: Visual QuickStart Guide, 6th Edition 129
 hyperlinks 71–73
 creating 72–73
 destination, creating 71–72
 hyphenation 171–172
 and Stanza 172
 and tables 181



Ibis Reader 14
 iBooks
 alignment 172
 anatomy of page 16–17
 and iBookstore 13
 borders and backgrounds 169
 bugs
 autoplay video 185
 images shrunk in columns 182
 styles in different rules 149
 video 184
 caching of EPUB files 120
 cover icon 116
 default font size 134
 dictionary 106
 dingbats 150
 DRM 13
 first page shown 111
 fonts
 crippled support for 140
 embedded, no support for 17, 153
 names for 134
 non-English 151
 workaround 140
 guide in 116
 hyphenation 171
 justification 172
 keeping elements together 155–157
 link display 180
 metadata display 108
 opening book in 123
 ornaments 150–152
 page breaks 155–157
 searching in 11
 specifying author in 107
 symbols 150
 tables in 181–182
 titles in 105
 validating EPUB for 126
 video in 183–185
 width property in CSS 175
 iBookstore 13
 IDPF 7, 14
 images 173–178
 adding to InDesign 61–69
 and captions 67–71
 and file names 62
 keeping caption with 156–157
 placing in InDesign 62–64
 wrapping text around 64–70, 175–176
 Images options
 in InDesign CS4 85
 in InDesign CS5 88
 !important, in CSS 130, 164, 166, 169
 indents, controlling 164
 InDesign
 adding images 61–69
 adding links 70–75
 automatic chapter breaks 89
 bugs
 broken links 70, 82
 date missing 81
 erroneous em conversion 163
 cleaning up EPUB files 131–137
 creating EPUBs with 45–90
 about 46–47
 cross-references 73
 CSS generated by 132–137
 exporting EPUB 82–90
 from CS4 83–86
 from CS5 87–90
 inheritance 135–186
 metadata 80–81
 one file or many 47
 styles 48
 XHTML generated by 131–132
 inheritance, InDesign vs CSS 135–137
 initial page of ebook 111
 iPad. *See also* iBooks
 page, anatomy of 16
 resolution 17
 ISBN
 in *content.opf* file 105
 in *toc.ncx* file 99
 iTunes, copying EPUB files to iPad 122

J

Japanese font 151
 justification forced by Apple in iBooks 1.1 172
 workaround 154

K

keeping elements together 155–158
 Kernest.com 153
 Kindle 14
 converting from EPUB 128

L

language
 declared by InDesign 132
 declaring in *content.opf* file 106
 declaring in Word file 41
 left text alignment 154
 linked styles, in Word 36
 links
 creating by hand 179–180
 in InDesign 70–75
 styles for 70–71
 lists 49, 132
 local formatting
 description 53
 replacing in InDesign 53–57

M

manifest 109–110
 and cover of ebook 115
 mathematical symbols 151
max-width property 114
 metadata
 displayed by ereaders 108
 in *content.opf* file 104–119
 in InDesign 80–81
 MIME types in manifest 109
min-width property, in CSS 182
 Mozilla extensions 170

N

navigational TOC
 creating InDesign style for 77–82
 generated with InDesign 76–79
 preparing InDesign book for 76–77
 subentries 102
 toc.ncx file for 98–103
navLabel element, in *toc.ncx* file 100
navMap element, in *toc.ncx* file 100
navPoint element, in *toc.ncx* file 100
 nested styles, in InDesign 57
 non-English fonts 151–152

Nook

 and borders and backgrounds 169
 and widows and orphans 155, 157
 fonts offered 138

Notepad 94

Notepad++ 31, 94

numbered lists 49, 132

O

OEBPS folder 95–96
 Open Packaging Format specification 104
 OpenType fonts 153
 ornaments 150–151
 orphans. See keeping elements together

P

page breaks. See keeping elements together
 page, initial 111
 poetry, formatting 165–168
 points, in Word files 33
 print book vs ebook vs website 8–13
 publisher, specifying in metadata 107

Q

QuickTime Player 184
 quotation marks, in Word files 41
 quotes, formatting 165–168

R

relative units, in Word files 38
 RGB images in EPUB files 82

S

shapes 151
 short lines, formatting 165–168
 sidebars
 creating 169–171
 wrapping text around 177–183
 size
 of iBooks page 16
 small caps 159–162
 soft hyphens 171
 spacing, controlling 163–168
 spine, in *content.opf* file 110–128
 Springy 93
 Stanza 14
 borders and backgrounds 169
 hyphenation dictionaries 172
 images 173
 line breaks 167
 sidebars 178
 text wrap 176

styles. *See also* CSS

in InDesign 48

applying main 50–51

applying special 52–53

creating 49–50

drop caps and nested styles 57

for links 70–71

replacing local formatting 53–57

in Word

adding new 23–24

and CSS 27

applying 26–27

cleaning up 35, 37, 39–40

consolidating 36–37

loading from document 25–26

moving to separate file 33

relative units 38

setting up 21–26

using shortcuts 39

style source document 77

SVG (Scalable Vector Graphics) format 153

symbols 150–151

T

table of contents. *See also* navigational TOC
with cross references in InDesign 73

Table of Contents Style 77–82

tables, creating 181–182

text alignment 154

text-align property 154

TextEdit 94

text editor. *See* BBEdit, TextWrangler, Notepad++

for editing EPUB files 94

vs word processor 31

text-indent property 165–168

TextMate 31, 94

TextWrangler 31

text wrap

around images 175–176

around sidebars 177–178

for drop caps 159–162

in InDesign 64–70

Thai font 151

title of ebook

adding in InDesign 81

and caching 120

in *content.opf* file 104

language of 104

toc.ncx file 98–103

typefaces. *See* fonts in ebooks

Typekit 153

Typotheque 153

U

universally unique id. *See* user id

unzipping EPUB 92–93

URLs, and links 179

user id

in *content.opf* file 105

in *toc.ncx* file 99

utf-8 format

and non-English fonts 151

declaring in Word 32–33

in XHTML files 96

V

validating EPUB file 126–128

video, adding 183–186

W

Webkit extensions 170

website

for this book 18

vs print vs ebook 8–13

widows. *See* keeping elements together

width property, in CSS 174, 176, 178

for video 185

WOFF (Web Open Font Face) format 153

Word, Microsoft 19

adding closing slash 43

adding quotation marks 41

cleaning up 44

declaring language 41

saving files as HTML 28–30

styling your document 20–27

wrapping text

around images 175–176

around sidebars 177–183

X

XHTML

and EPUB 14

declaration 31, 96

generated by InDesign 131–132

inside EPUB file 96–97

XML, and *content.opf* file 103

xml:lang attribute

adding to Word 41

generated by InDesign 132

Z

Zapf Dingbats 150–151

zipping EPUB file 124–128

again after edits 120–121

commands for 121