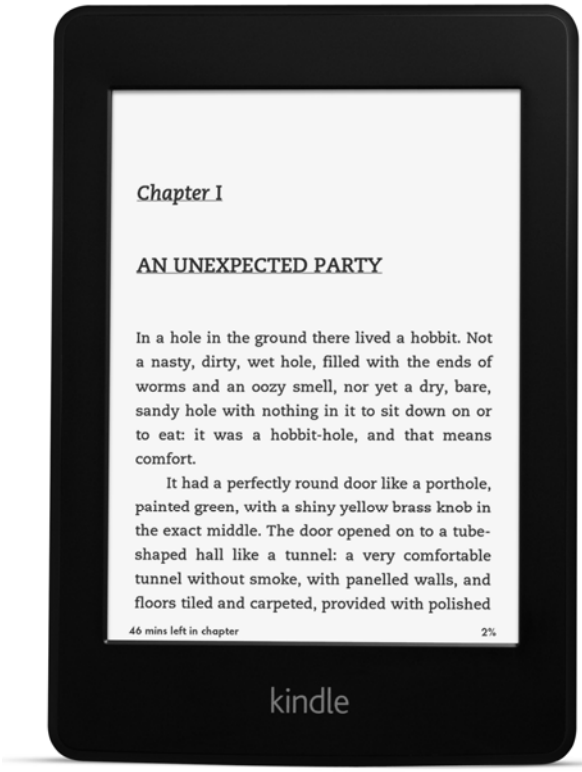

eBook Formatting

Dr. George W. Benthien

January 2013

E-mail: georgewb@gbenthien.net

Preface

Electronic books (eBooks) are rapidly surpassing printed books in sales volume. Besides being cheaper and easier to obtain, they are also simpler to publish. Most authors prepare their manuscript in a word processor such as Microsoft Word and then use a conversion program to put it in the desired electronic format. While this often works it usually doesn't produce nicely formatted eBooks. I will show you how, with a little effort, you can control the format of your eBook.

The two major eBook formats are the MOBI format (used by Amazon on most Kindles) and the EPUB format used by most other eBooks. Amazon has a new format (KF8) that is used on the Kindle Fire as well as the new Kindle and Kindle Touch, but has yet to be incorporated in older Kindle models. Until there is broader support for the new format it is better to stick with the older MOBI format that can be read on both the older and newer Kindles. Both the MOBI and EPUB formats are based on HTML, the language used by web pages. Therefore, it makes sense to submit your documents in HTML format instead of relying on some conversion program.

EPUB and KF8 have broader support for HTML than MOBI. Therefore, we will restrict ourselves to the limited subset of HTML supported by the older Kindles. This is not a serious limitation since most documents only require a few common HTML tags. it is also good for the beginner since there is much less to learn. For those familiar with HTML it will be necessary to learn which HTML tags are supported and how some of the defaults are different than those employed by the web. The MOBI format also supports a limited subset of CSS (Cascading Style Sheets). CSS styles give the user more control over the display of text and images. I will discuss the basics of HTML and CSS as used by the Kindle MOBI format (and also EPUB).

We will use the following approach to constructing an eBook:

1. Prepare your document(s) in Microsoft Word.
2. Copy the text into a text editor such as notepad++ (free), removing the Word formatting in the process.
3. Add necessary HTML tags to the text document(s).
4. Prepare a CSS style file to add desired visual effects.
5. Prepare organizational and navigation files required by EPUB.
6. Zip (compress) the EPUB files into an EPUB document.
7. Validate EPUB document.
8. Convert EPUB document into a MOBI document using Amazon's free Kindlegen program or Kindle Previewer.

If you are familiar with HTML, then you can skip the first three steps and write your document(s) directly in a text editor — adding HTML tags as you go.

Introduction

All the Kindle devices except the Kindle Fire and the new Kindle and Kindle Touch use the MOBI format. The newer Kindles can read the MOBI format in addition to the newer more powerful KF8 format. MOBI was developed by a French company that was bought by Amazon in 2005 in order to obtain rights to their format. Kindles can read unprotected MOBI files (with extensions .mobi or .prc) as well as their own files (with extension .amz) that differ only in the way digital rights are handled. The EPUB format is based on a standard developed by the International Digital Publishing Forum (IDPF). This format is used by almost all of the eBook publishers other than Amazon, e.g., Apple, Barnes & Noble, and Sony. An EPUB document consists of the HTML files containing the content along with several organizational files. These files are compressed into a single zip file having the extension epub, e.g., MyBook.epub.

The basic underlying language for both the EPUB and MOBI formats is HTML, the language of the web. This is not surprising since an eBook reader, like a web browser, is not page oriented. The text is free flowing. The user can modify how much is displayed on a page by changing the font size, line spacing, margins, or words per line. HTML stands for **H**yper **T**ext **M**arkup **L**anguage. It is a markup language not a programming language. It consists of tags placed in the document file that indicate how the document should be organized and displayed. The tags do not modify the content. The older Kindles only support a subset of the HTML language and many of the defaults are special. CSS (Cascading Style Sheets) can be used to control the visual appearance of the document. The older Kindles have limited support for CSS.

Since the basic underlying language is HTML, the obvious question is "How do I get my document into HTML?" Many authors, who are not familiar with HTML, construct their document in Microsoft Word and then save it as an HTML file. Word does the conversion to HTML. Unfortunately, the HTML produced contains a lot of extraneous code that often leads to unintended consequences. I prefer a different approach. The document can still be prepared in Microsoft Word if you like, but the conversion to HTML is handled differently. After some initial preparation, the text of the Word document is copied to the clipboard and then pasted into a text editor. This process removes the special formatting information that Microsoft Word adds. Word processors such as Microsoft Word add extra characters to the document file that are not visible on the screen. These extra hidden characters specify formatting information such as fonts, paragraphs, page breaks, and line spacing. These special characters need to be removed in order to have a valid HTML file.

After the text is copied into the text editor the remainder of the editing will be done there. Documents produced by Text editors contain only text and do not add extra formatting information. A good free text editor is notepad++. The next step is to add the HTML tags. The HTML tags could be added one by one, but this can be quite tedious. Instead we will make use of a powerful search and replace feature involving *regular expressions*. This will allow us to make multiple substitutions and deletions at the same time. The HTML tags will provide the required document structure and formatting information. Once the HTML has been added, we then construct the other informational files required by the EPUB standard. The collection of EPUB files are then zipped into a single compressed file. There are validation tools that can be used to insure that the EPUB document has been properly constructed. The validated EPUB document can then be converted to a MOBI document using the Kindlegen program or the Kindle Previewer, freely supplied by Amazon.

In the next section we will describe the basic structure of an HTML document and will define some of the HTML tags that are valid in both MOBI and EPUB documents.

Basic HTML

HTML stands for **H**ypertext **M**arkup **L**anguage. It consists of tags that are placed within a document to indicate how the document content should be organized and displayed. HTML tags are analogous to marks editors used to place in documents as instructions for the printer. The MOBI format used in older Kindles only supports a limited portion of HTML. For example, you can't prescribe the font family or the font color or the background. However, this means that there is less to learn. In fact you can write a nicely formatted Kindle or EPUB document using only a few HTML tags. HTML tags usually occur in pairs. There is an opening tag having the form <tag-name> and a closing tag having the form </tag-name>. For example, denotes the beginning of a paragraph and denotes the end of the paragraph. There are a few tags that are self closing. They have the form <tag-name />. We will see some examples of these later on. The general structure of an HTML document suitable for both EPUB and MOBI eBooks is shown below.

```
<?xml version="1.0" encoding ="UTF-8" ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DT

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">

<head>

<meta http-equiv="Content-Type" content="application/xhtml+xml; charset=utf-
8"/>

<!-- The head section contains information about the document that is not displayed. -->

</head>

<body>

<!-- The body section contains the document content. -->

</body>

</html>
```

The first few lines specify the version of HTML used in the document. The EPUB specification requires a strict form of HTML designated as XHTML. These initial lines do not change and can be pasted into the beginning of each document. The document starts with the tag <html …> and ends with the tag </html>. The content of the document is placed between the tags <body> and </body>. The head section begins with <head> and ends with </head>. It contains information about the document that is not displayed. We will see some items that can be placed there later on. We will now list some common HTML tags that can be used to format a MOBI or EPUB document.

** … **
: The included text forms a paragraph. Things such as first line indent and space between paragraphs are different, by default, in different eBook readers. I will show you later how to modify the default settings using CSS. With the help of CSS styles, paragraph tags can also be used to make chapter and section headings.

** … **
: The div tag is sort of a blank container to which we can add properties using CSS. The tag is often used to apply properties to a group of paragraphs.

<blockquote> … </blockquote>
: Indents ablock of text by providing larger margins. The default settings can be modified using CSS.

<h1> … </h1>
: Included text forms a top level heading. Lower order headings are denoted by h2, h3, h4, h5, and h6. The heading text is bold and heading sizes become smaller as the heading number increases. It has been my experience that forming headings using styled paragraph tags gives much more predictable results than the use of the above heading tags.

** ... **
Span is like a blank element that surrounds content within a line. By itself it doesn't do anything, but properties can be added to it using CSS.

** ... **
The text within is made bold. You can also use .

** ... **
The text within is made italic. You can also use .

<u> ... </u>
The text within is underlined. There are some places where the <u> tag is not allowed. Styled span tags can be used as a replacement.

****
Displays the image contained in the src file name. The text corresponding to alt should describe the image. The alt property is required for eBooks, but it can be left blank, i.e., alt="".

** ... **
Centers the contents horizontally.

<a id="*mark*" />
Establishes an internal bookmark to which you can link. In this case the bookmark has the name "mark". Older Kindles sometimes have problems with this type of bookmark.

** *Text to link to* **
Another way to establish an internal bookmark to which you can link. This method seems to work well in all readers.

Link Text
Link to a bookmark in the same file whose name follows #. Clicking on the link text takes you to the bookmark location.

Link Text
Link to a bookmark in an external file whose name follows #.

Link Text
Link to another HTML file.

**
**
Inserts a line break.

** ... **
Contains an unordered (bulleted) list. The individual list items are contained within <li> tags. To obtain an ordered (numbered) list replace ul by ol. Lists do not work properly in older Kindles if the default properties are modified using CSS.

_{...}
he text within is made smaller and lowered to form a subscript.

^{...}
The text within is made smaller and raised to form a superscript.

You can place comments in the HTML file that are not displayed as part of the document. The format is shown below

```
<!-- you can place a comment here -->
```

It should be noted that span tags with appropriate CSS styles can also be used to produce bold, italic, and underlined text.

In HTML all the formatting is determined by the HTML tags. Extra spaces or blank lines added by the document's creator are ignored. The default behavior of HTML tags can be modified by placing additional conditions within the opening tag. However, it is better to use CSS styles for this purpose. In the next section we will see how to modify HTML tags by using CSS (Cascading Style Sheets).

Basic CSS

CSS (Cascading Style Sheets) is a powerful way of adding properties to selected HTML elements. With CSS you can, for example, change the behavior of every <p> tag with one rule. There are three ways that CSS can be applied to an HTML file. The CSS rules can be contained in an external file that is loaded with the link command in the head section. For example,

<link href="*style.css***" rel="stylesheet" type="text/css" />**

would apply the rules in the file style.css to the present HTML file. The advantage of placing the CSS rules in a file is that the same rules could be applied to multiple HTML files. The rules can also be placed between the tags <style> and </style> in the head section. These rules would apply only to the present file. You can also place CSS rules within an opening tag of an individual HTML element using style="*CSS rules*". For example,

<p style="text-indent: 0;">body

In this case the rules would only apply to that particular element. The MOBI format only supports a limited number of CSS rules. The basic form of a CSS rule is

Selector {Attribute: value;}

A selector can be an HTML tag name such as p, div, or h1. It can also be a user determined class name preceded by a period, e.g., **.left**. The properties of a class are added to an element by placing a class="…" statement within the opening tag, e.g., <p class="left">. If you want to restrict a class selector to a particular HTML element, you can use a selector of the form **element.class**, e.g. p.left. More than one attribute can be assigned to a selector by separating the attributes by semicolons. For example,

p {text-indent: 0; text-align: left;}

More than one selector can be assigned the same attribute by separating them by commas. For example,

h1, h2, h3 {margin-bottom: 15;}

Many CSS properties involve units. Common units are pixels (px), percent (%), and em-units (em). An em is usually taken as the default font size. This is a very useful unit since it scales nicely when the user changes the font size. Unfortunately, the older Kindles do not handle this unit properly. For the older Kindles an em has a static size (it doesn't change with font size) and this size is not the same for all models.

Below are some examples of useful attributes

text-indent: 0;
: sets the amount of first line indent (0 in this case). Hanging indents can be obtained by using negative values. If no units are given, then the units are considered to be pixels.

text-align: left;
: text is aligned on the left. This produces a ragged right margin. Other values for text-align are right, center, and justify.

margin-top: 40px;
: sets a margin of 40 pixels above the element. margin-bottom would place a margin below the element.

padding-left: 1em;
: sets padding to 1em inside element on the left. The em unit is relative to the default font size

font-style: italic;
: sets the font style to italic. Other values are normal and oblique.

font-weight: bold;
: sets font weight to bold. Another possible value is normal.

font-size: 1.5em;

sets font size relative to the default size, in this case 1.5 times larger. Other possible property values are xx-small, x-small, small, medium, large, x-large, xx-large.

text-decoration: underline;
underlines text. Another possible value is line-through.

You can insert comments in a CSS file by placing them between /* and */.

The next section contains a sample CSS style file .

Example CSS File

```
/* Eliminate all default margin and padding settings */
html,body,div,h1,h2,h3,h4,h5,p,blockquote {margin: 0; padding: 0.01em;}

body {text-align: justify;}

/* Paragraph settings suitable for fiction */
p {text-indent: 1.25em; margin-top: 0; margin-bottom: 0;}

/* For non-fiction set text-indent to zero and margin-bottom to 1em. */

/* For chapter headings ( page break before recommended) */
p.chapter {text-indent : 0; font-size: 1.5em; font-weight: bold; margin-top: 1.0em;
margin-bottom: 1.5em; text-align: center; page-break-before: always;}

/* Styles for span elements */
span.i {font-style: italic;}

span.b {font-weight: bold;}

span.u {text-decoration: underline;}

/* Style for block quotes */
p.quote {margin-top: 1em; margin-bottom 1em; margin-left: 3em; margin-right: 3em;
font-style: italic;}
```

Example HTML File

<?xml version="1.0" encoding ="UTF-8" ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">

<!-- The head section contains information about the formatting of the document that is not part of the document content. This section begins with <head> and ends with </head> -->

<head>

<!-- The title is a required element of the head section, but it is not displayed as part of the document. You must add a heading in the body section to display the title. -->

<title>My Title**</title>**

<!-- CSS style properties can be added to the document by either linking to an external file using the link command or by placing the desired style rules between <style> tags. The style rules consist of a selector that determines the element to which the style will be applied followed by properties inside matching braces. The selector can be an HTML element like the heading h1 or can be a class definition like .left. The class can be applied to an element by adding the statement class="left" inside the opening tag, e.g., . -->

<link href="style.css" rel="stylesheet" type="text/css" />

<style>

h1 {font-size: 30; margin-bottom: 1.5 em;}

p {margin-bottom: 20px;}

.left {text-align: left;}

.noind {text-indent: 0;}

</style>

</head>

<!-- The body section contains the content of the document. HTML tags are added to organize the material and to display it properly. -->

<body>

<!-- HTML Content goes here. -->

<!-- A named div around a section heading allows you to link to that section from somewhere else in the document by means of a tag of the form Displayed Text (intro is the name of the bookmark). -->

<div id="intro"**><h1>**Introduction**</h1>**

** **This is a paragraph. ****This text will be bold.** **This text will be italic.****
<u>This text will be underlined**.</u> **

** ** This paragraph is aligned left which produces a ragged right margin.** **

** **This paragraph doesn't have any first line indent.** **

<blockquote>This paragraph is normally indented on the left and the right by giving it larger margins. For greater control you can assign CSS properties margin-left and margin-right to blockquote. **</blockquote>**

<!-- The following is an unordered (bulleted) list. To obtain an ordered (numbered) list replace ul by ol -->

```
<ul>
  <li> list item 1</li>
  <li> list item 2</li>
  <li> list item 3</li>
</ul>

</body>
</html>
```

Preparation of WORD Files

Most authors write their documents in the word processor Microsoft WORD or something equivalent. Word processors add additional formatting information to the document that is not visible on the screen. We will eventually need to remove all of this hidden formatting since it is not allowed in HTML. However, we need to make some initial preparations so that desired formatting information will be preserved in the HTML file. Here are the preliminary steps that I recommend:

1. You want to replace straight quotes with smart quotes and double hyphens (--) with em dashes. In WORD 2010 you can find these options as check boxes in the menu **File > Options > Proofing > AutoCorrect Options > AutoFormat** and in the menu **File > Options > Proofing > AutoCorrect Options > AutoFormat As You Type**. They should be checked in both places. In WORD 2003 they can be found in the menu **Tools > AutoCorrect Options > AutoFormat (AutoFormat As You Type)**
2. Using Find and Replace, replace ' by ' and replace " by ". This looks like it wouldn't do anything, but it activates the smart quote replacement. Similarly, replace -- by — (em dash) and - (space-hyphen-space) by – (en dash).
3. We want to mark the sections of text that are italic, bold, or underlined so that we will know where they are after the WORD formatting is removed. We will mark italic text by putting III at its beginning and end. Similarly, we will surround bold text with BBB and underlined text with UUU. We will later replace these markings with appropriate HTML tags. To mark the italic text we place **ctrl-i** in the Find box so it says Font:Italic and place III^&III in the Replace box (^& is Microsoft Word's symbol for the found text). Similarly replace **ctrl-b** by BBB^&BBB and **ctrl-u** by UUU^&UUU.
4. Copy the text to the clipboard and paste it into a text editor such as notepad++. This is the most important step. It removes all the hidden formatting information. Each paragraph is now a single line of text. Save this file with an html extension.

Adding HTML tags to the text file we have created involves a lot of searching and replacing of character strings. This process is much simpler if we make use of regular expressions. We will describe these regular expressions in the next section.

Regular Expressions

Now that we have the text from the WORD file in our text editor, we want to start adding HTML tags. To do this we will make use of The editor's find and replace capability along with regular expressions. Regular expressions allow us to find and replace multiple strings of characters in one operation. They are like a complex version of wildcards. You probably are aware that you can search for all jpeg files in a directory by using the wild card expression *.jpg in the Find Box. Regular expressions are an extension of this capability to character strings within a file. We will describe some of the regular expressions that can be used in notepad++, but other text editor's have similar capabilities.

.	(period) matches any character.
[…]	matches any of the characters within the brackets. For example, [a5g] matches any of the characters a, 5, or g. You can also use ranges of characters. For example, [0-9] matches any of the numbers zero through nine, and [a-z] matches any lowercase letter.
*****	matches the preceding item zero or more times. For example, A[0-9]* would match A, A5, A23, etc.
+	matches the preceding item one or more times.
[^…]	matches any character except those within the bracket.
^	anchors the next regular expression to the start of a line (when not included in brackets).
$	anchors the previous regular expression to the end of a line (doesn't include new line character).
\<	anchors the next regular expression to the start of a word.
\>	anchors the previous regular expression to the end of a word.
(…)	assigns a tag to the characters selected by the expressions inside the parentheses. The first set of tagged characters is assigned the tag \1, the second \2, and so on up to \9. These tags \1–\9 can be used as a substitute for the characters they represent in a replace operation
\n	matches a new line character (line feed)
\r	matches a carriage return
\t	matches a tab character
\s	matches a space or a tab character
\f	matches a page break

Sometimes * and + extend the match farther than we would like. For example, ^[.]*e applied to “The old man is dead” would select “The old man is de”, i.e., it extends the selection as far as possible. Adding ? after * or + limits the selection to the smallest matching string. In our example it would select “The”.

If you need to use a character that has a special meaning in regular expressions (for example a period) in its normal sense, then precede it by a backslash. Let's look at some simple examples.

Example 1 To remove blank lines we search (in extended mode) for

```
\n\r
```

and then replace with a blank entry.

Example 2 Paragraphs in the WORD file are now single lines of text. We want to put HTML paragraph tags around the text in each line. To do this we can search for

```
^(.+)$
```

and then replace with

```
<p>\1</p>
```

In the next section we will show how to use regular expressions to add html tags to our document.

Adding HTML Tags

We will now use regular expressions to clean up our text file and add HTML tags. The following are the steps we will follow:

1. To remove all tab characters we place \t in the find box and nothing in the replace box. Make sure that Regular expression is selected under Search Mode. Now hit the Replace All key.
2. To remove all blank lines we change to Extended mode. We then place \n\r in the find box and nothing in the replace box. Again we hit Replace All. Now change back to regular expressions.
3. We can remove white space from the beginning of each line by placing ^\s+ in the find box and nothing in the replace box. Again hit Replace All key.
4. We can remove white space from the end of each line by placing \s+$ in the find box and nothing in the replace box. Hit Replace All key.
5. Search for double spaces and replace by single spaces.
6. We now want to see if there are any special characters such as ï or è in the document. You can find any special characters by searching for [^<>A-Za-z0-9\.,'"?\\\^\|\-\[\]:!;()/$#@&%*_+{}=~\s“”’—–], i.e., any character that is not one of the standard keyboard characters. We need to either remove these or replace them by their HTML equivalent. You can find HTML equivalents at a number of sites on the internet such as www.w3schools.com/tags/ref_entities.asp. Use the HTML entity names and not the entity numbers.
7. There are some symbols that either have no keyboard representation or have a special meaning in XHTML. We will replace these by their HTML entity names. In particular
    - Replace & by &
    - Replace > by >
    - Replace < by <
    - Replace " by "
    - Replace “ by “
    - Replace ” by ”
    - Replace ‘ by ‘
    - Replace ’ by ’
    - Replace … by …
8. Since paragraphs are now lines, we want to wrap paragraph tags around each line. To do this we place ^(.+)$ in the find box and \1 in the replace box. Now hit Replace All.
9. You can cut and paste the text into the following HTML template:

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE html PUBLIC"-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">

<head>

<meta http-equiv="Content-Type" content="application/xhtml+xml; charset=utf-8" />

<title>*Your Title Here***</title>**

<link type="text/css" rel="stylesheet" href="*style.css***" />**

</head>

<body>

<!--Insert Content here-->

</body>

</html>

You can now add any links, headings, etc. you desire and create a CSS file to style the elements the way you want them. It is considered good style to have a page break before each chapter. You can do this by adding **page-break-before: always;** to the style for chapter headings. I would recommend using styled paragraphs for chapter and section headings instead of the h1, h2, … tags. Any heading that you want to link to from an HTML table of contents should be surrounded by div tags with an id, e.g., ** Chapter 1 **. In the next section we will show how to build an EPUB document.

Structure of EPUB Documents

It is convenient to place longer documents in multiple HTML files. For, example, you might place each chapter in its own HTML file. Web pages handle multiple HTML files by having internal links that allow the reader to jump from one HTML file to another. In eBooks multiple HTML files are handled by supplying a separate file that lists the reading order of the files. For EPUB and MOBI documents this function is performed by a package file having extension opf, e.g., content.opf. The EPUB specification also requires several other files used for navigational and organizational purposes. We will describe each of these required files. An EPUB document is merely a compressed zip file containing the HTML files and other required files making up the document. Below is the recommended directory structure for the EPUB files:

- mimetype
- META-INF/container.xml
- OEPBS/content.opf
- OEPBS/toc.ncx
- OEPBS/title.html
- OEPBS/content html files
- OEPBS/CSS style files
- OEPBS/image files

The **mimetype** file must have this name, must be in the root directory, must be exactly 20 bytes in size, and must contain the following text (no line feeds, carriage returns, or extra spaces)

application/epub+zip

This file is the same for all documents so it only needs to be created once.

The file **container.xml** must have that name and must be in the directory META-INF. An example container file is shown below.

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

The only thing in this file that is subject to change is the path to the opf file (underlined above). The directory does not have to be named OEPBS, but some readers have trouble with other names. The opf file name does not have to be content.opf, but again this is conventional. If we stick with the conventional directory and file name, then this file does not vary from document to document.

The file **content.opf** provides some basic information about the document, lists the files that make up the document, and specifies the reading order of the document's html files. The file doesn't have to have this name, but the name has to be that specified in container.xml. The next section contains an annotated example OPF file.

Example OPF File

<?xml version="1.0" ?>

<!-- The items below to be supplied by the author are underlined. -->

<!-- The package must have a unique identifier (BookId in this example) -->

<package version="2.0" xmlns="http://www.idpf.org/2007/opf" unique-identifier="BookId"**>**

<!-- Metadata: The required metadata element is used to provide information about the publication as a whole. -->

<metadata xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:opf="http://www.idpf.org/200

<!-- Title [mandatory]: The title of the publication. This is the title that will appear on the "Home" screen. -->

<dc:title>BookTitle**</dc:title>**

<!-- Language [mandatory]: the language of the publication. Some common language strings are:

"en" English
"en-us" English — USA
"en-gb" English — United Kingdom
"fr" French
"fr-ca" French — Canada
"de" German
"es" Spanish -->

<dc:language>en**</dc:language>**

<!-- Cover [mandatory]. The cover image must be specified below under <manifest> and referenced with the id given here under content. -->

<meta name="cover" content="BookCover" **/>**

<!-- Each book must have a unique uuid identifier. It can be generated at http://www.famkruithof.net/uuid/uuidgen. The id should be the package id specified above. -->

<dc:identifier id="BookId" **opf:scheme="uuid">urn:uuid:**efe839d0-4f4f-11e1-b86c-0800200c9a66**</dc:identifier>**

<!-- Creator [Mandatory]: The author of the book. For multiple authors, use multiple <dc:Creator> tags. Additional contributors whose contributions are secondary to those listed in creator elements should be named in contributor elements. -->

<dc:creator>Benthien, George**</dc:creator>**

<!-- Publisher [Mandatory]: Name of publisher or your name if self published -->

<dc:publisher>Benthien**</dc:publisher>**

<!-- Date [Mandatory]: Date of publication in YYYY-MM-DD format. (Days and month can be omitted).-->

<dc:date>2011-12-07**</dc:date>**

<!-- Subject [Optional]: A topic or key word. -->

<dc:subject>Reference**</dc:subject>**

<!-- Description [Optional]: A short description of the publication's content. -->

<dc:description>A short paragraph describing the publication … **</dc:description>**

<!-- Editor [Optional]

<dc:contributor opf:role="edt">Editor's Name**</dc:contributor>**

</metadata>

<!-- Manifest [Mandatory]: The manifest provides a list of the files that are part of the publication (e.g. Content Documents, NCX table of contents, image files, CSS style sheets). The manifest element must contain one or more item elements with the following media-type attributes:

application/xhtml+xml (HTML content files)
application/x-dtbncx+xml (NCX table of contents)
text/css (CSS style file)
image/jpeg (JPEG image)
image/gif (GIF image)
image/png (PNG image) -->

<manifest>

<!-- HTML content files [mandatory] -->

<item id="tc" **media-type= "application/xhtml+xml" href=**"toc.html"**></item>**

<item id="intro" **media-type= "application/xhtml+xml" href=**"introduction.html"**>**
</item>

<item id="c1" **media-type= "application/xhtml+xml" href=**"chapter1.html"**></item>**

<item id="c2" **media-type= "application/xhtml+xml" href=**"chapter2.html"**></item>**

<item id="c3" **media-type= "application/xhtml+xml" href=**"chapter3.html"**></item>**

<item id="c4" **media-type= "application/xhtml+xml" href=**"chapter4.html"**></item>**

<item id="c5" **media-type= "application/xhtml+xml" href=**"chapter5.html"**></item>**

<!-- Image and style files-->

<item id="fig1" **media-type= "image/gif" href=**"Figure1.gif"**></item>**

<item id="fig2" **media-type="image/jpeg" href=**"Figure2.jpg"**></item>**

<item id="MyStyle" **media-type= "text/css" href=**"style.css"**></item>**

<!-- NCX table of contents [mandatory] -->

<item id="MyNcx" **media-type= "application/x-dtbncx+xml" href=**"toc.ncx"**></item>**

<!-- Cover image [mandatory] -->

<item id="BookCover" **media-type= "image/jpg" href=**"cover.jpg"**></item>**

<!-- Cover page [optional] -->

<item id="cp" **media-type= "application/xhtml+xml" href=**"coverpg.html"**></item>**

</manifest>

<!-- Spine [Mandatory]: Following manifest, there must be one and only one spine element, which contains one or more itemref elements. Each itemref references the id of a document designated in the manifest. The order of the itemref elements organizes the associated content files into the linear reading order of the publication. The toc attribute refers to the id given to the NCX file specified in the manifest. -->

<spine toc="MyNcx"**>**

<!-- the spine defines the linear reading order of the book -->

<itemref idref="cp"**/>**

<itemref idref="tc"**/>**

<itemref idref="intro**"/>**

<itemref idref="c1**"/>**

<itemref idref="c2**"/>**

<itemref idref="c3**"/>**

<itemref idref="c4**"/>**

<itemref idref="c5**"/>**

</spine>

<!-- Guide [Mandatory]: Within the package there may be one guide element, containing one or more reference elements. The guide element identifies fundamental structural components of the publication, to enable Reading Systems to provide convenient access to them. The Kindle reading system supports two special guide items which are both mandatory.

type="toc" [mandatory]: a link to the HTML table of contents

type="text" [mandatory]: a link to where the content of the book starts (typically after the front matter)

Kindle reading platforms need both the guide items to provide a consistent user experience to the user. It is good practice to include both a logical table of contents (NCX) and an HTML table of contents (made of hyperlinks). The NCX enables various advanced navigation features, but the HTML table of contents can easily be discovered by the user by paging through the book. Both are useful. -->

<guide>

<reference type="cover" title="Cover**" href="**coverpg.html**"></reference>**

<reference type="toc" title="Table of Contents**" href="**toc.html**"></reference>**

<reference type="text" title="Beginning**" href="**intro.html**"></reference>**

</guide>

</package>

The NCX File

The NCX file specifies the entries in a navigational table of contents for EPUB eBooks. For older Kindles with keyboards the NCX file specifies a set of predefined bookmarks (usually chapters or major sections). The reader can jump between the preset bookmarks using the right or left arrows of the 5-way controller button. This is a very useful feature in the older Kindles. The bookmarks appear as small dots on the status bar. It appears that the Kindle Touch and the Kindle Fire do not use the NCX file. The status bar has been replaced by a slider bar that allows you to move continuously through the document. However, you can't make discrete jumps like you could with the keyboard Kindles. Below is an example NCX file

<!-- The items to be supplied by the author are underlined. -->

<!-- The first three statements are the standard header for an NCX file. -->

<?xml version="1.0" encoding="UTF-8"?>

**<!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"
xml:lang="en">**

<head>

<!-- The content of dtb:uid must be exactly the same as the uuid specified in the OPF file. -->

<meta name="dtb:uid" content="urn:uuid:efe839d0-4f4f-11e1-b86c-0800200c9a66"**/>**

<!-- The NCX specification allows for a hierarchal table of contents. It is recommended that the depth be set at 1 since that is all that the Kindle and Nook support. -->

<meta name="dtb:depth" content="1"/>

<!-- The total page count and the max page number are not needed. Leave them set to zero. -->

<meta name="dtb:totalPageCount" content="0"/>

<meta name="dtb:maxPageNumber" content="0"/>

</head>

<!-- For docTitle use the dc:title specified in the OPF file. -->

<docTitle><text>BookTitle**</text></docTitle>**

<!--For docAuthor use the dc:creator specified in the OPF file. The name should be given in the form LastName, FirstName. -->

<docAuthor><text>Benthien, George**</text></docAuthor>**

<!-- The navMap section is the heart of the NCX file. For EPUB eBooks this section specifies the entries in the navigational table of contents. For the older keyboard Kindles this section defines a set of predefined bookmarks. You can jump between the preset bookmarks by pressing the right or left arrows of the 5-way controller. It appears that the Kindle Touch and the Kindle Fire do not use the NCX file. The order of the entries is specified by the **playOrder** attribute. The location is specified by the **content src** attribute. This consists of an HTML file name such as "filename.html" or an HTML filename with a named anchor added such as "filename.html#mark". The preset bookmarks for the Kindle will appear as little dots on the progress bar at the bottom of the screen.-->

<navMap>

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

<navLabel><text>Introduction**</text></navLabel>**

<content src="introduction.html"/>

</navPoint>

<navPoint id="navpoint-2" playOrder="2">

<navLabel><text>Chapter 1</text></navLabel>

<content src="chapter1.html"/>

</navPoint>

<navPoint id="navpoint-3" playOrder="3">

<navLabel><text>Chapter 2</text></navLabel>

<content src="chapter2.html"/>

</navPoint>

<navPoint id="navpoint-4" playOrder="4">

<navLabel><text>Chapter 3</text></navLabel>

<content src="chapter3.html"/>

</navPoint>

<navPoint id="navpoint-5" playOrder="5">

<navLabel><text>Chapter 4</text></navLabel>

<content src="chapter4.html"/>

</navPoint>

<navPoint id="navpoint-6" playOrder="6">

<navLabel><text>Chapter 5</text></navLabel>

<content src="chapter5.html"/>

</navPoint>

</navMap>

</ncx>

Making an EPUB eBook

We have discussed most of the types of files that occur in an EPUB document. The main content is contained in the HTML files. The appearance can be modified by using CSS style files. The other files are organizational. It is useful to have both an NCX table of contents and an HTML table of contents. The HTML table of contents contains links to the various sections of the document and can be seen as you page through the document. An example HTML table of contents is shown below.

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<title>Table of Contents</title>
<style>
  p {margin-bottom: 1em; text-align: left}
  p.ind {margin-left: 2em;}
</style>
</head>
<body>
<p><a href="intro.html">Introduction</a></p>
<p><a href="chap1.html">Chapter 1</a></p>
<p><a href="chap2.html">Chapter 2</a></p>
<p class="ind"><a href="chap2.html#sect1">Section 2.1</a></p>
<p class="ind"><a href="chap2.html#sect2">Section 2.2</a></p>
<p><a href="chap3.html">Chapter 3</a></p>
<p><a href="chap4.html">Chapter 4</a></p>
<p><a href="chap5.html">Chapter 5</a></p>
<p><a href="refs.html">References</a></p>
</body>
</html>
```

This code will produce the following table of contents

You can also have an HTML cover page if you wish. This is a standard XHTML file with the cover image specified in a tag. The size requirements for covers change fairly often, so it is best to check with the publisher for the current requirements. The last I heard Amazon wanted a 1688x2700 pixel Jpeg image.

The final step in making an EPUB document is to zip the files into a file with extension .epub, e.g., MyBook.epub. There is one caveat. The file mimetype must be the first file added and it must be placed in the EPUB file uncompressed. Not all zip programs do this easily. One program that does this is the command line program **zip.exe** that can be downloaded from the site stahlworks.com/dev/index.php?tool=zipunzip. To add mimetype to the file MyBook.epub you would use the command

zip MyBook.epub -DX0 mimetype

To add the remaining files you would use

zip MyBook.epub -rDX9 META-INF OEPBS

At this point is a good idea to validate your EPUB file. There are a number of EPUB validators. You can validate online at validator.idpf.org or you can download a command line validator at

https://github.com/IDPF/epubcheck/releases/.

A good viewer for EPUB files is Adobe Digital Editions. It can be downloaded from

www.adobe.com/products/digitaleditions.

Conversion to MOBI Format

The conversion from EPUB to MOBI format can be accomplished using the Kindlegen program. This is a command line program that can be downloaded from

amazon.com/kindleformat/kindlegen.

I suggest that you download at the same time the Kindle Previewer from

amazon.com/kindleformat/kindlepreviewer.

This previewer allows you to see how a Kindle eBook will look for various settings. Since Kindlegen is a command line program, the user must enter the full path of the executable kindlegen.exe in order to use it. It is possible to develop an alias that includes all the path information and is set every time a command window is opened. The alias can be used to run Kindlegen in any folder. In this section I am assuming that we are running Kindlegen on a computer using a Windows operating system. We first create a bat file init.bat that we can use to set any command aliases we desire. If kindlegen.exe is stored in the directory c:\kindlegen, then we put the following command in init.bat

doskey kg=c:\kindlegen\kindlegen.exe -c1 -verbose $1

You can also put aliases for the zip commands (used in building EPUB file) in init.bat. If zip.exe is stored in the directory utilities, then we insert the following commands in init.bat

doskey zipnc=e:\utilities\zip.exe $1 -DX0 mimetype

doskey zipc=e:\utilities\zip.exe $1 -rDX9 META-INF OEPBS

If you dowloaded the epubcheck validator, you can put an alias for it in init.bat, i.e.,

doskey val=java -jar e:\utilities\epubcheck\epubcheck-3.0b4.jar $1

where you need to use the correct path for the jar file you downloaded.

Now running init.bat in the command window will replace the complete path to Kindlegen plus the options by the alias kg. Similarly, the zip commands will be replaced by zipnc and zipc, and the epubcheck command will be replaced by val. For all these aliases the argument that will replace $1 is the name of the EPUB file. We next modify the registry so that init.bat will run automatically every time a command window is opened. We run regedit from the start menu to open the registry editor. We then navigate down the tree to the key

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Command Processor

We then enter a new string value whose value name is *AutoRun* and whose value data is *c:\MyFolder\init.bat*. Of course, you need to substitute the path to init.bat for c:\MyFolder. It is necessary to restart the computer in order for these settings to be incorporated. Now you only need to type kg in the command window in order to run Kindlegen in any folder. To convert an EPUB file BookName.epub to MOBI format we execute the command

kg BookName.epub

The result will be a file called BookName.mobi. This MOBI file can be viewed with the Kindle Previewer at various font size settings in order to see approximately how it will look on a Kindle. If the displayed eBook doesn't meet with your approval, you can change the document files and repeat the process. The final test will be to view it on your Kindle. An alternate way to convert the EPUB file to a MOBI file is to view the EPUB file in the Kindle Previewer. The previewer will convert the file to MOBI format before displaying it.

To load the MOBI file on your Kindle connect the Kindle to your PC using the provided USB cable. The Kindle will appear as a drive on your PC. Under this drive there are several directories, one of which is named “documents”. You want to copy the MOBI file you created to this directory on the Kindle. You can then safely disconnect the Kindle as you would any connected device. When Windows says it is safe to disconnect the device, you can remove the cable from both the Kindle and the PC. The book you created will now appear on the home page of your Kindle.

Kindle Fire Format

The Kindle Fire introduced a new format called kf8. This format has much broader support of HTML and CSS standards than the older MOBI format. It supports most standard HTML and CSS elements as well as a good portion of the new HTML5 and CSS3 elements. At the present time the Kindle Fire is the only device that supports kf8. However, Amazon says that it plans to add this format to its top E ink devices. The new Kindlegen program and the new Kindle Previewer support both the kf8 and MOBI formats. There are, however, a few compatibility problems. For example, the Kindlegen program no longer supports negative values for the width property. Using negative values for width in a paragraph was a common way to produce hanging indents. Hanging indents are now produced by using negative values for the CSS text-indent property. However, the text-indent property is handled differently in the Kindle Fire and older Kindles. To handle cases such as this, Amazon provided two new media queries. Any CSS code inside of @media amzn-kf8{…} applies only to the Kindle Fire. Any CSS code inside of @media amzn-mobi{…} applies only to the older Kindles. Below is an example of how to use a class hang to produce a hanging indent

@media amzn-kf8{ p.hang {text-align: left; left-margin: 2em; text-indent: -2em;} }

@media amzn-mobi{ p.hang {text-align: left; text-indent: -2em;} }

The not modifier can also be used with the media queries, i.e.,

@media not amzn-kf8{…}

@media not amzn-mobi{…}.

Unfortunately, there is nothing corresponding to media queries for HTML code.

References

1. The BB eBooks web site http://bbebooksthailand.com/developers.html has a lot of good information and tools for formatting eBooks
2. The eBook **How to Format Your eBook for Kindle, NOOK, Smashwords, and Everything Else**, Paul Salvette (2011) is available from Amazon and Barnes&Noble.
3. Guido Henkel a nice series on eBook Publishing that is available from guidohenkel.com/2010/12/take-pride-in-your-ebook-formatting/.
4. The Amazon Kindle Publishing Guidelines (PDF) is available from kindlegen.s3.amazonaws.com/AmazonKindlePublishingGuidelines.pdf.
5. **Kindle Formatting: The Complete Guide**, Joshua Tallent (2009) is an excellent reference, but needs to be updated.