

# 8 Understanding the Portable HTML template

---

## What's in this chapter

- ❖ Browser requirements
  - ❖ Why use Portable HTML?
  - ❖ Modifying the template
  - ❖ Nifty navigation techniques
  - ❖ Setting up a frameset
  - ❖ Understanding graphics
- 

WebWorks Publisher's Portable HTML template is the simplest of the templates provided with the product. It doesn't have a lot of complicated features or bells and whistles. But many of the more complex templates are based on the Portable HTML template, so it makes sense to spend some time understanding how this template works.

## Browser requirements

The Portable HTML template outputs HTML 3.2, a standard set by the World Wide Web Consortium (W3C). You can find details about the HTML 3.2 specification at [www.w3c.org](http://www.w3c.org).

In practice, creating HTML 3.2 means that your files can be read by Netscape Navigator version 2 or higher or Internet Explorer version 3 or higher. Most other browsers (such as Lynx) also support HTML 3.2.

The browser requirements for Portable HTML are very basic; all of the other HTML templates require higher-level browsers.

## Why use Portable HTML?

Portable HTML is an excellent choice as a starting point if you need to accommodate some of the older browser versions or if you are concerned about cross-platform or cross-browser compatibility. Although the Dynamic HTML template gives you better formatting control, some of the technology used in that template works inconsistently in different browsers.

Portable HTML also produces relatively small files. The WebWorks Help template, for example, includes two support files that are each about one megabyte. Portable HTML does not have any large supporting files, so the final set of output files is quite compact.

## Modifying the template

The Portable HTML template is the basis for all the other HTML templates, so many of the customizations you make for Portable HTML work for other templates. This section describes several different customization techniques, but keep in mind that WebWorks Publisher is almost infinitely customizable. You can expand or modify these ideas, or create your own styles.

## Changing the background

The background used by an HTML file can be either a color or a graphic.

To change the background color, you use the bgcolor attribute in the <body > tag. To change the background image, you use the background attribute.

---

**WARNING:** If you specify both a background image and a background color, the image overrides the color, and the color is ignored. Therefore, you should only include one or the other. The default Portable HTML template specifies both a color and an image, so only the image is used. If you want to use a background color instead, you must remove the background image reference.

---

To change the background for the HTML pages, follow these steps:

- 1 In the Support Files folder, double-click the page template (ASP) file you want to open and modify. For the table of contents, open TOC.asp; for the index, open Index.asp; for other files, open Normal.asp.

- 2 Locate the `<body >` tag in the file, which looks like this:

```
<body link="#3366CC" vlink="#9999CC" text="#000000"
alink="#0000CC" bgcolor="#FFFFFF" background="images/
backgrnd.gif">
```

- 3 Modify the attributes to get the effect you want. To use a background image, change the background reference (highlighted in the preceding code) to reference your graphic name:

```
background="images/mybackground.gif"
```

To specify a background color, remove the background attribute and modify the bgcolor reference to use the hex code:

Example 4

```
<body link="#3366CC" vlink="#9999CC" text="#000000"
alink="#0000CC" bgcolor="#111111">
```

**NOTE:** For a list of hex codes, see Table 20 on page 107.

- 4 Save the template file.
- 5 If you specified a graphic in step 3, copy the graphic to the Support directory's images folder using your file system.
- 6 Save and regenerate the project.

The background color or image you've specified now appears in your output files.

**NOTE:** Your background color may not match the background color of your graphics. See page 366 for information on how to solve this problem.

## Changing the header and footer

HTML pages don't have real headers and footers, but you can put repeated information at the top and bottom of each page.

To modify the headers and footers, follow these steps:

- 1 In the Support Files folder, double-click the page template (ASP) file you want to open and modify. For the table of contents, open TOC.asp; for the index, open Index.asp; for other files, open Normal.asp.
- 2 Modify the HTML code as necessary to insert the information you want. For example, to remove the Quadralay contact information from the bottom of the Normal.asp file, locate the code that inserts this information. It looks like this:

```
<table align="right" border="0" cellspacing="0"
cellpadding="0">
<tr><td align="right"><font size="1">
<a href="http://www.webworks.com">
```

```
</a><br>  
Quadralay Corporation<br>  
http://www.webworks.com<br>  
Voice: (512) 719-3399<br>  
Fax: (512) 719-3606<br>  
<a href="mailto:sales@webworks.com">sales@webworks.com  
</a><br></font></td>  
</tr>  
</table>
```

You can remove this code (remove everything from `<table >` to `</table >`) and replace it with your code, or just change the contact information as appropriate.

- 3 Save the template file.
- 4 Save and regenerate the project.

## Creating notes

The basic styles provided with the template cover most of the formats that you need to get started, but one item that's missing is a note format. You'll want to create a note format that, at a minimum, makes the word "NOTE:" bold at the beginning of the paragraph and then outputs the text of the paragraph as regular text. An even better-looking solution would be to create a note with a hanging indent. For that, you'll need an invisible table (an HTML table with no border).

These procedures assume that the FrameMaker tag for notes uses autonumbering to insert "NOTE:."; that is, the word is not typed as part of the text.

### Basic Note paragraph style

To create a basic Note paragraph style, follow these steps:

- 1 Open the Style Designer. The **Paragraphs** tab is displayed by default.
- 2 From the Name drop-down list, select **Body** to display the Body style definition.
- 3 Select the **Duplicate** button to create a copy of the Body style. "Duplicate" is automatically appended to the copied style's name (BodyDuplicate).
- 4 In the Name field, select BodyDuplicate and type in Note.

You have now created a new Note style that has the same definition as Body. The Note definition looks like this:

```
$BPDivOpen;  
<p>  
  <a name="$AUTOTAG;" > </a>\
```

```
<font $BPFontFamilyOverride;>\
$BPNumberAndData;\
</font>
</p>
$BPDivClose;
```

Example 5

- 5 Replace **\$BPNumberAndData** (shown in bold in the preceding code) with the following code:  

```
<b>NOTE:</b> $DATA;\
```
- 6 Select **OK** to close the Style Designer and save the new style.
- 7 Map your FrameMaker document's Note paragraph tag to your new Note style. (If the names match exactly, WebWorks Publisher will automap them, but it's always a good idea to check.)
- 8 Save and regenerate your project.

Your notes now display "NOTE:" in boldface at the beginning of the paragraph as shown in Figure 66.

**NOTE:** Welcome to Scriptorium Publishing!

**Figure 66:** A bold note

### Using a table to create a hanging indent for the note

To create a hanging indent for your note, follow these steps:

- 1 Open the Style Designer.
- 2 Select the **New** button to create a new, blank paragraph style.  
**NOTE:** The new style isn't much like any of the existing styles, so it's easier to create it from scratch.
- 3 In the Name field, select the name of the new style (NewStyle1) and type in Note.

You have now created a new, blank style named Note.

- 4 Insert the following code for the note:

Example 6

```
<a name="$AUTOTAG;" > </a>\
<table border="0">
<tr valign="top"><td><font $BPFontFamilyOverride;>
<b>NOTE:</b>
</font></td>
<td><font $BPFontFamilyOverride;>
$DATA;
</font></td>
</tr>
</table>
```

**NOTE:** In HTML, line breaks are not important, so you can put them anywhere you want.

- 5 Select **OK** to close the Style Designer and save the new style.
- 6 Map your FrameMaker document's Note paragraph tag to your new Note style.
- 7 Save and regenerate your project.

Your notes now have a hanging indent in the output (Figure 67).

**NOTE:** You can use vacation as soon as you earn it. If you would like to take time off and do not have enough vacation hours accrued, check with your manager to find out what your options are.

*Figure 67: Hanging indent*

## Creating character styles

WebWorks Publisher provides a Default character style, which attempts to generate the correct formatting information for your HTML output by matching the formatting in the FrameMaker file. This approach is problematic because it means that your HTML output will contain lots of extraneous formatting code and also because it forces you to match the FrameMaker and HTML formatting. In some cases, you might want items that are bold in FrameMaker to be output as regular, red text in HTML. The Default style won't do this.

**NOTE:** A similar problem occurs in the Dynamic HTML template with both the Default character style and the Default paragraph style. See "What about the Default paragraph style?" on page 210 for more information.

Instead of using Default, we strongly recommend that you use separate styles for bold, italics, and other formatting.

To create a new character style, follow these steps:

- 1 Open the Style Designer.
- 2 Select the **Characters** tab.
- 3 Select the **New** button to create a new, blank style.
- 4 In the Name field, select the name of the new style (NewStyle1) and type in a new name.
- 5 In the text field, type the style definition. You can use one of the examples shown in Table 25, or you can create your own style.

**Table 25:** Character style examples

Style	Code
Bold	<code>&lt;b&gt;\${DATA}; &lt;/b&gt;</code>
Italics	<code>&lt;i&gt;\${DATA}; &lt;/i&gt;</code>
Bold italics	<code>&lt;b&gt;&lt;i&gt;\${DATA}; &lt;/i&gt;&lt;/b&gt;</code>
Underline	<code>&lt;u&gt;\${DATA}; &lt;/u&gt;</code>
Superscript	<code>&lt;sup&gt;\${DATA}; &lt;/sup&gt;</code>
Subscript	<code>&lt;sub&gt;\${DATA}; &lt;/sub&gt;</code>
Color	<code>&lt;font color="#111111"&gt;\${DATA}; &lt;/font&gt;</code> <sup>1</sup>
Bold color	<code>&lt;font color="#111111"&gt;&lt;b&gt;\${DATA}; &lt;/b&gt;&lt;/font&gt;</code>
Courier (monospaced text)	<code>&lt;code&gt;\${DATA}; &lt;/code&gt;</code>

1. See Table 20 on page 107 for color codes.

## Changing the default font

In the Portable HTML template, most paragraph styles include a reference to the `$BPFontFamilyOverride` building block, which calls the `$UMFont` user macro. If you want to change the default font for the page, just change `$UMFont`.

To change `$UMFont`, follow these steps:

- 1 Open the Style Designer.
- 2 Select the **User Macros** tab.
- 3 From the Name drop-down list, select **UMFont**. The default definition is:

```
Verdana, Arial, Helvetica, sans-serif
```

- 4 Modify the definition with the fonts you want. For example, you might remove Verdana:

Example 7

```
Arial, Helvetica, sans-serif
```

- 5 Select **OK** to close the Style Designer and save your changes.
- 6 Save and regenerate your project.

The new output files display the new font setting.

### Font woes

Specifying a font in your HTML pages does not guarantee that your readers will actually see that font. The font must be available on the end user's system. One way to work around this limitation is to provide a list of fonts, such as the default for \$UMFont (shown in Step 3 on page 167).

The result of this list is that if Verdana is installed on the end user's system, it is used. If not, the browser looks for Arial. If that is not available, it tries Helvetica. If that fails as well, the browser uses the default sans-serif font on that system. Verdana is available on most Windows systems, Arial is a standard Windows font, and Helvetica is a standard Macintosh and UNIX font, so the list provided by the default definition has a good chance of finding one of the top three fonts listed.

## Breaking up files into smaller chunks

By default, WebWorks Publisher outputs a single HTML file for each of your FrameMaker files. This can result in very long HTML files, which would force users to scroll through pages and pages of content to find what they want. To eliminate this problem, you can break up your output files by specifying that a particular paragraph tag should start a new HTML file.

To break up output into smaller files, follow these steps:

- 1 Identify the paragraph tag or tags in your FrameMaker source files that should start a new page.
- 2 In WebWorks Publisher, right-click the FrameMaker and Generated Files folder, then select **Properties** from the pop-up menu. The File Properties dialog box is displayed.
- 3 In the paragraph mappings (shown by default), select each paragraph tag that should start a new page and map it to NewHTMLPage (Figure 68).

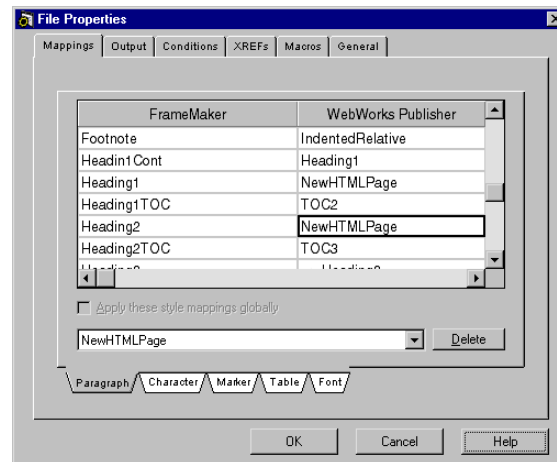
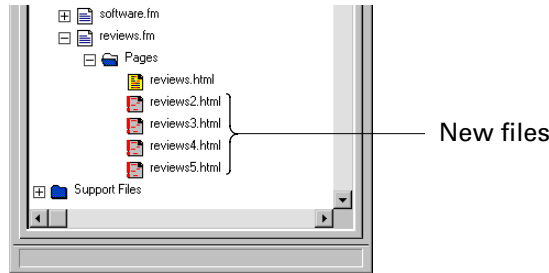


Figure 68: Mapping with the NewHTMLPage style

- 4 Select **OK** to close the File Properties window and save your new mappings.

WebWorks Publisher immediately scans your files to identify the new output files. You should now see additional output files in the Pages folder under each FrameMaker file (Figure 69). The new files are not created, however, until you regenerate the project.



**Figure 69:** New files in the project window

- 5 Save and regenerate your project.  
The new HTML files are created.

## Modifying the table of contents

The table of contents styles (TOC1–5) are set up to remove the page number from the end of a table of contents entry. If, for example, you have a fairly standard table of contents entry like this:

Chapter 1: Introduction.....4

WebWorks Publisher will remove the dot leader and the page number and output just this:

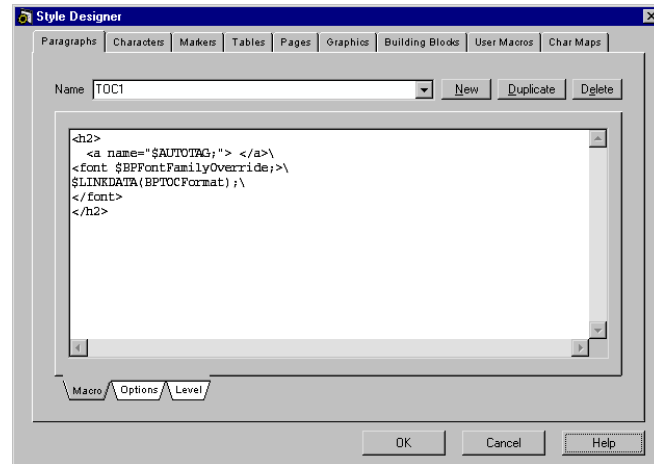
Chapter 1: Introduction

You might still want to remove the reference to the chapter number, though. To do so, you need to modify the \$UMTOCFilter user macro. \$UMTOCFilter contains the code that strips the page numbers; it looks like this:

```
$GET_GLOBAL(tocpara) ["[ ] [0-9xviXVI]+$", ""];
```

The tocpara global attribute contains the text of the table of contents paragraph that's currently being processed. The code in brackets is a *regular expression*; it's looking for a particular pattern of text and deleting it. For a detailed discussion of regular expressions, see Chapter 19.

**NOTE:** The TOC1–5 paragraph styles call \$BPTOCFormat, which sets the tocpara attribute and calls \$UMTOCFilter (Figure 70 on page 170).



**Figure 70:** TOC style

The following sections discuss how to modify \$UMTOCFilter to remove chapter references or remove page numbers that don't follow the pattern that \$UMTOCFilter is looking for.

### Removing page numbers

When you use regular expressions to search and replace, you use the following syntax:

```
[ "xxx" , "yyy" ]
```

The result is that every case of “xxx” is replaced with “yyy”.

You can also search for something and delete it—by replacing it with nothing:

```
[ "xxx" , " " ]
```

The regular expression in \$UMTOCFilter uses this search-and-replace technique:

```
[ "[ ][0-9xviXVI]+$" , " " ]
```

The expression searches for the pattern in the first set of quotation marks:

```
"[ ][0-9xviXVI]+$"
```

and replaces it with nothing:

```
" "
```

The search pattern breaks down like this:

[ ]	Match a space
[0-9xviXVI]+	Match one or more (+) of the following characters: 0,1,2,3,4,5,6,7,8,9,10,x,v,i,X,V,I
\$	Match the end of the line

Before writing the contents of the paragraph into `tocpara`, WebWorks Publisher has already replaced tabs, carriage returns, new lines, and break tags with regular spaces (this is done in `$BPTOCFormat`). So, the preceding pattern should pick up any numeric or Roman numerals at the end of the line that are preceded by spaces. There are, however, a number of problems that can occur with the provided pattern:

- *Roman numerals.* The pattern provides for matching roman numerals that contain x (10), v (5), and i (1). But the Roman numeral for 50 is L, so the pattern wouldn't match anything that uses that numeral. This means that anything after 39 (xlix) doesn't match. Similarly, if your numbers use C (100), D (500), or M (1000), you'll need to add those to the pattern. The modified pattern looks like this (additions are bold):

Example 8

```
$GET_GLOBAL(tocpara) ["[ ] [0-9xvilcdmXVILCDM]+$", " "];
```

- *Multipart page numbers.* If your page numbers use the chapter number (for example, 2-1, 2-2, 2-3, and so on), the provided pattern won't work. To allow for numbers as prefixes, add a hyphen to the pattern:

Example 9

```
$GET_GLOBAL(tocpara) ["[ ] [0-9xviXVI\\-]+$", " "];
```

**NOTE:** Because the hyphen is considered a special character inside brackets, you must precede it with a backslash. See Chapter 19, "Regular expressions," for more information.

If all of your numbers are multipart page numbers, including the appendices (A-1, A-2, A-3, and so on), try this new pattern:

Example 10

```
$GET_GLOBAL(tocpara) ["[ ] [A-Z0-9]+-[0-9]+$", " "];
```

Notice that this pattern no longer matches Roman numerals.

### Removing chapter references

If your table of contents entries contain a reference to the chapter, such as:

Chapter 2: Tempering chocolate

you may want to remove the reference and just output the text:

Tempering chocolate

To do this, you need to add another pattern to the original `$UMTOC-Filter` pattern. The original pattern is as follows:

```
["[ ] [0-9xviXVI]+$", " "]
```

You'll add a new pattern that looks for the word `chapter` followed by a space, followed by a number:

Example 11

```
$GET_GLOBAL(tocpara) ["Chapter [0-9]+: "," "]["[ ] [0-9xviXVI]+$", " "];
```

