

# So How Does It Work?

## Process & Product Overview

Creating HTML Help combines Web technologies (HTML, CSS, DHTML) with technical writing skills (audience analysis, information acquisition, organization, writing, grammar), and graphic design talents (layout, conceptualization, color management). Making the pieces fit can feel like putting together a particularly rough-edged jigsaw puzzle! Windows help procedures seem flat and lifeless compared to the power of HTML; color schemes that work beautifully in print dither into mush on Web browsers. You switch from application to application, using different tools for writing, HTML editing, graphic design, interactive scripting....

So how can you get a handle on it all? One of the best ways is by walking through the process yourself. This chapter presents a high-level view of an entire HTML Help project, from start to finish. The opening design sections describe the types of choices you must make before beginning your project, while subsequent divisions detail the steps required to create an HTML Help system.

At each step of the walkthrough, you will find both case study information gleaned from the first HTML Help project I ever created and the actual



If you don't want to key in the tree planting project elements, you can download them at the companion Web site:

[www.jmek.net/html\\_help](http://www.jmek.net/html_help).

steps you'll need to take to create a sample project that guides a home gardener through the process of planting a tree, from the conceptual task of choosing an appropriate species to step-by-step directions on planting the particular specimen. While the topics of the tree-planting project are accurate, they are not extensive. (For example, the choice of species is limited to two!) The focus is on creating HTML Help rather than on actually planting a tree.

The case study describes the help for ManageX, a Windows NT-based system administration application from Hewlett-Packard's OpenView software division. Developed at a time when very few non-Microsoft applications had been released with HTML Help systems, this foray into the new technology presented both challenges and opportunities. The challenges stemmed from the need for a rapid transition to the new system and from the trials of working with a "bleeding edge" development tool (Microsoft's HTML Help Workshop), including some of its quirkiest features (notably, merged Help files). The opportunities brought in the wonderful (and bleeding edge) technologies of the World Wide Web, including dynamic HTML (DHTML) and cascading style sheets (CSS). Both the challenges and the opportunities should be familiar to anyone who has worked with HTML Help. Usability studies conducted on the Hewlett-Packard product show that our audiences felt the challenges were met successfully. Following this chapter's walkthrough should help you meet the challenges as well.

**NOTE:** In order to follow along with the procedures in this chapter, you must have Microsoft's HTML Help Workshop installed on your system. The Workshop is available as a free download at <http://msdn.microsoft.com/library/tools/htmlhelp/>. You may also need an HTML editor; in a pinch, you can write and edit HTML pages in the Windows Notepad or any other editor that produces plain text. Do not use a full-fledged word processor unless you know how to make it save your documents in a plain text format with an `.htm` extension.

## What the process looks like

One of the confusing aspects of the HTML Help process is the multitude of different parts, each using its own file type. Basically, the process involves the following steps:

### 1. Planning system design:

Plan the overall Help system, addressing both traditional online help decisions (audience analysis, high-level organization, navigation model) and HTML-specific questions (compiled or uncompiled projects, style sheet design, interactive elements). This is a good time to set up your directory structure.

### 2. Crafting reusable elements & content:

Use an HTML editor or Notepad to write individual topic files (`*.htm`). Identify and create required reusable elements including graphics, style sheets, and DHTML snippets. Combine these elements into HTML files that can be used as templates.

### 3. Creating & testing the HTML Help project:

Use the HTML Help Workshop or a third-party product to define the HTML Help project (`*.hhp`), combining the individual topic files, specifying project parameters, and building the navigation system (table of contents, `*.hhc`, and index, `*.hhk`). Finally, compile the project into the distributable file (`*.chm`), and troubleshoot the results.

### 4. Merging modular projects (optional):

If you will combine several help projects into a single master project (for example, if your documented application has components that are only available in certain packages), create a master project file (`*.hhp`) that specifies the constituent `.chm` files as "Merged Files" in the `.hhp` and with "INCLUDE" statements in the `.hhc`.

## Notes on file types & extensions

As the previous list shows, HTML Help involves many different file types, each with its own extension and special requirements. The following list and table (table 2.1) summarize the different file types, their extensions (and the extension of the Windows help counterparts), their uses, and their requirements.



`.hhp`, `.hhc`, and `.hhk` files are plain text files. You can create them in the HTML Help Workshop, a third-party editor or, if you're brave, in Notepad.

- Reusable elements can include cascading style sheet files (`.css`) to specify formatting; boilerplate text (in `.txt` files for unformatted language or in `.htm` files for formatted text); dynamic HTMLcode snippets (in standard `.txt` format for cut-and-paste or in separate JavaScript `.js` files). These can be combined into `*.htm` files that function as templates.
- Individual topics are contained in standard HTML files (`.htm`).
- The project file (`.hhp`) specifies such things as window definitions, compiler options, included files, and the default page for the help file.
- The contents file (`.hhc`) creates the expanding table of contents. It lists each displayed contents entry with its associated HTML file and icon. The file is similar to an unordered list `<ul>` in HTML.
- The index file (`.hhk`) defines the index. It lists each displayed index entry with its associated HTML title and HTML file. It, too, is similar to an undented list `<ul>` in HTML.
- The distributable help file then consists of a single file with the `.chm` extension. (One exception: If you're creating a modular system, you will have multiple `.chm` files to distribute, one for each module.) You create this file using the HTML Help Compiler (in the Workshop) to combine the `.htm`, `.hhp`, `.hhc`, and `.hhk` files.

TABLE 2.1 FILE TYPES & EXTENSIONS FOR HTML HELP COMPONENTS

EXTENSION (.HLP COUNTERPART)	FILE TYPE	CREATED IN	PURPOSE & USE
.chm (.hlp)	compiled help file	HTML Help Workshop	contains all elements of the Help system (text, graphics, styles, scripts, contents, index, search) in a single compressed file distributed to the end user
.css	cascading style sheet	HTML editor, specialized CSS tool, or text editor (Notepad)	contains format definitions (font, size, color, etc.). The .css extension identifies an external file that HTML files link to.
.hnc (.cnt)	help contents file	HTML Help Workshop or text editor (Notepad)	contains table of contents for the project; in modular projects, may contain references to external .hnc files
.nhk	help index file	HTML Help Workshop or text editor (Notepad)	contains a specific list of indexed terms and associated topics. Does <i>not</i> include either Alinks or Klinks, though Klinks may be merged into the index during compilation.
.hnp (.hjp)	help project file	HTML Help Workshop or text editor (Notepad)	contains information about the project, including window definitions, included files, compiler & compatibility options, default page
.htm, .html (.rtf)	Web page	HTML or text editor, or Workshop	contains actual text and graphics of the project's Help topic pages
.js	JavaScript or JScript	scripting tool or text editor (Notepad)	scripts for dynamic presentations or interactivity The .js extension identifies an external file that HTML files link to.

## Planning the system design

Before diving into the project, plan as much as possible. Planning is an essential part of any documentation project, but it's even more important when working with a new format that brings new expectations and new possibilities. HTML Help brings into play traditional online Help decisions (based on audience and needs analysis as well as choice of tools, organization, navigation, and presentation models) as well as the newer demands of a Web-savvy audience for style and interactivity.

In most software development projects, the Help developer has some "dead time" at the beginning of the cycle when the programmers are wrestling with the backend code. Planning before the development code is ready can save enormous amounts of time during the inevitable "crunch" at the end of the production cycle. Pre-designing individual elements (such as outlines, graphics, and scripts) for the ManageX help system allowed me to take advantage of HTML Help's opportunities without destroying the final production schedule. Our sample tree-planting project doesn't have the same schedule constraints, but planning will still help to insure a useful and well-designed help system.

### Traditional online help decisions

#### Audience/needs analysis

As much as possible, interview actual users of your product (or similar products) to determine their needs, desires, and expectations. If you can't talk to users directly, gather data from your company's technical support group (or usability lab, if you have one). Connecting with this team can provide valuable insight into what users are actually saying about the product. And don't let your information sources just feed back "received wisdom." Prod users and support groups for ideas about what they might like if they could wish for the moon; show them new ideas and see what

they say. You may be surprised. Remember: in the early days of the Web, black type on a gray background was on the outer limits of “cool.”

When starting the new ManageX HTML Help system, I looked back at the old system and the responses it had received. I was fortunate to have access to user comments. ManageX has a very active technical support and consulting community which frequently passes user comments back to the development team.

What the tech support professionals had to say convinced me that you can never assume you know what users want. ManageX users are network engineers, IT professionals who often come from a UNIX background that deals with “man pages” of white text on a black background, accessible only if you already know how to spell the proper command. I expected their navigational and aesthetic requirements to be limited, with more focus on code and commands. But once these users moved to an NT environment, they wanted full navigation capabilities (contents, index, full-text search) plus pictures, pictures, and more pictures (figure 2.1)!



*Figure 2.1 Never assume. While you might think that IT professionals would prefer a plain Help system, the primary request from our users was for more and more pictures! A powerful navigation system was also a “must have.”*

Within my development team, the mandate was to be on the “cutting edge” of Windows-based technologies. Because of this, management supported the move to a new system (even if it did require installation of Internet Explorer; see *Compiled or not?* later in this chapter), and the developers were willing to learn new ways to hook help into the application. The developers and managers had also been bitten by the Web bug and agreed with users that graphic appeal and easy navigation are essential, not optional, requirements.

These comments helped guide me toward the compiled form of HTML Help, which automatically provides full-text search and makes it easy to add contents and indexing. In addition, I designed a simple but attractive banner graphic combining logo elements with a “signpost” identifying the component of the application covered by the topic. The users responded enthusiastically to these and other easy-to-implement enhancements such as colored headline text and a lightly textured background (contrary to most usability recommendations). Usability tests conducted on the released product included many positive comments about the “snazzy” help!

For our sample tree-planting project, talk to friends and neighbors, particularly ones who may have been landscaping recently. Among my sources, the requests included:

- **Simple terminology.** When botanical terms are necessary, definitions for the layperson should be immediately available. The tree-planting project can meet this need by using a simple dynamic HTML script to show and hide definitions with a click of the mouse.
- **Simple, attractive design.** Gardeners are interested in aesthetics (or why would they plant trees and flowers?). So the project uses a cascading style sheet to create a simple design that pleases the eye without distracting from the pictures required by the desire for...
- **Pictures to identify specific trees.** This one is fun and (if you have good clip art or a digital camera) easy to implement. We'll also add small map graphics showing appropriate planting areas for each tree.

- **Quick access.** From e-mails, suggestions, Santa Cruz redwood, users want a picture and plenty of requirements *right now*. HTML Help's full-text search capability, indexing, and "favorites" tab will meet these requirements.

### Organization: reference vs. task

Most online help use one of two organizations: a command-reference orientation that follows the menu structure of the application or a task orientation that walks users sequentially through the steps required to perform tasks. Neither one is necessarily better; the choice depends on a combination of user analysis, application design, and the other available materials.

In the case of the ManageX system, the technical focus of our users combined with the task-based printed manuals allowed me to use a command-reference orientation for the online help. In addition, the organization of the application itself pointed in this direction. ManageX works as a "snap-in" to the Microsoft Management Console, but ManageX itself is composed of several "extension snap-ins." A user may add or remove extensions at will; in addition, different versions of ManageX may include different sets of extensions. For these reasons, it made sense to organize the help system around the individual extensions. In addition, I made the help system modular. The help system for each extension is composed of a single `.chm` that merges into the master help file. Thanks to the ability to merge help files at run time, I can keep up with the different versions of ManageX by adding and removing individual `.chm` files from the list of installed files.

From a user's point of view, this modular, reference-based system presents only those items that are available in the currently installed product. From the developers' point of view, they could "easily" add context-sensitivity at the extension level. (Unfortunately, this "ease" proved elusive and was one of the main places we noticed the "bleeding" edge of the HTML Help API.)

The tree-planting project involves both a task orientation (to walk users through the steps involved in choosing a tree and then in planting it) and a reference orientation (for the “encyclopedia” of specific trees). We can plan to expand our project (and future sales) by setting up for run-time merging of separately purchased reference modules for specific geographical areas. (Don't forget that the company's sales and marketing departments are an important secondary audience!)

## Navigation model

Once you decide on the basic organization, consider how users will navigate through it. Compiled HTML Help's tripane format uses the left pane for navigation tools (figure 2.2).



*Figure 2.2 Basic HTML Help Navigation Pane. Compiled HTML Help uses a “tripane” format with the left pane devoted to navigation tabs. The three standard tabs are the Contents, Index, and Search tabs. A Favorites tab is also available for storing users’ most frequently referenced pages.*

The **Contents** tab provides an expanding/collapsing table of contents that "auto-syncs" to the user's current location in the Help system. The Help author specifies the hierarchy, the page titles, destination for the links, and identifying icons.

The **Index** tab provides access to author-defined keywords. The Help author can code the keywords into the individual HTML files or create a separate index ( \*.hhk ) file created within the HTML Help workshop. The keywords need not actually appear in the Help file, making this navigation aid useful for providing synonyms (e.g., creating new topics, adding new topics, drafting new topics, etc.)

The **Search** pane provides automatic full-text search capabilities. When the user types a word and clicks List Topics, HTML Help searches through all words in the Help project. The Help author does not have to specify any keywords, but the word that the user types in must actually appear in the text of the Help topic.

The **Favorites** tab (not pictured) allows users to save their most frequently used topics in an easily accessed location. (The ManageX system was created under HTML Help 1.1; Favorites were not available until version 1.2, so it does not appear in the sample pictures taken from this project.)

All of these navigation aids are part of the HTML Help setup. You may want to include additional cues for users. For example, since the ManageX program was made up of multiple modules, I wanted a quick way to notify users of their current location. HTML Help's "Auto-sync" in the table of contents provides users one clue, but I decided to also use graphic "headers" that varied according to the extension. Because they depended only on high-level groupings, these headers were one of the elements I could design before the product was ready for more detailed documentation. The sample tree-planting project could also use zone maps for additional cues.



The HTML Help Workshop is a free download. Beginning with version 1.3, the download is at `msdn.microsoft.com/library/tools/htmlhelp/`.

## Tools

While analyzing audience and application needs, don't forget to consider your needs as well; specifically, what tools will you use? At a minimum, decide whether to use Microsoft's HTML Help Workshop directly or to access it through one of the available third-party tools. Also choose an HTML editor and, possibly, specialized tools for creating CSS style sheets and interactive DHTML elements.

Microsoft's HTML Help Workshop may prove to be the best choice for creating your HTML Help projects. This new technology is moving so quickly that the third-party tools have difficulty keeping up with the newest features. On the other hand, the third-party tools may provide "add-ons" that make it easier to keep up with the technology. So it's usually best to try the demos and see for yourself. Three leading products are:

- ForeHelp™ from ForeFront™, [www.ff.com](http://www.ff.com)
- Doc-to-Help™ from Wextech™, [www.wextech.com](http://www.wextech.com)
- RoboHelp™ from eHelp™ (formerly Blue Sky™), [www.blue-sky.com](http://www.blue-sky.com)

As Blue Sky, eHelp's zealous marketing department won RoboHelp a large market share based on corporate purchases. However, ForeHelp carries the votes of many actual users based on both product functionality and service after the sale. (At the leading Help author conferences in 1999, ForeHelp won Author's Choice from the WinWriters Conference as well as Best Overall Windows help Authoring Tool and Best HTML Help Output from Help University). In addition, Doc-to-Help has long been the favorite for authors who single-source online and printed content.

For the ManageX HTML Help project, I decided to work directly in the HTML Help Workshop. When writing Windows help, I used third-party tools and never got "under the hood" to deal directly with the Help compiler. As a result, I never felt I understood the product as thoroughly as I would like. At this point, I want to really understand the technologies; so when I ultimately settled on HTML Help, I decided to put the project together and compile it directly in Microsoft's HTML Help Workshop. I deliberately did not use any third-party help authoring tools; in fact, I often find myself

editing the project files with Notepad. This is not to denigrate third-party tools, but sometimes you can learn more working in a hands-on mode.

As for the HTML pages themselves, the goal is to find an editor that writes clean code and moves quickly and transparently from raw code editing to a more polished graphical user interface (GUI). Ideally, it should also have sophisticated tools for creating more complex elements. It is important that the editor not require the distribution of special additions (such as the Microsoft Front Page® extensions), inhibit you from moving among tools, or add extraneous and/or proprietary code that might conflict with HTML standards (or the HTML Help Workshop requirements). For example, a common problem involves the superfluous use of `<DIV>` tags. Some HTML editors separate HTML pages into "divisions" using `<DIV>` tags before applying other tags, even when there is absolutely no need to add the `<DIV>` tags. Since `<DIV>` has other, special uses in HTML, this just adds clutter and the potential for misinterpreted HTML code. Once you've located an editor that writes clean code, look for one that also supports "assets" or "libraries" for reusable text, graphic, or code elements for drag-and-drop insertion and enhanced productivity. Other things to look for are internal tools for creating image maps, cascading style sheets, and dynamic HTML, as well as site management tools that make sense to you.

My personal favorite for working with HTML is HoTMetaL Pro® from SoftQuad® ([www.softquad.com](http://www.softquad.com)), which writes beautifully clean code and provides asset and site management, image map tools, a CSS editor, and a great links checker. For dynamic HTML, I use Dreamweaver™ from Macromedia™ ([www.macromedia.com](http://www.macromedia.com)), which includes a number of extremely useful "pre-canned" DHTML effects. In particular, the timeline feature (which lets you show, hide, and move elements at predefined time intervals to create a low-overhead online "movie") is worth the cost of the program. Dreamweaver has good site and asset management tools, but its basic code is somewhat cluttered.

Many HTML editors are available, so choose your favorite. You may still find yourself using Notepad for quick edits. Instructions in this book require only Microsoft's HTML Help Workshop and a basic text or HTML editor.

## HTML aspects

### Compiled or not?

At the current time, the main decision point in adopting compiled HTML Help will be users' reactions to the requirement of Microsoft Internet Explorer (IE). Without IE, you can only use the uncompiled version of HTML Help, which requires an entire "web" of HTML files and lacks features such as full text search and associative links. Compiled HTML Help provides built-in contents, index, and searching facilities (although these facilities have their own "bugs"). It also supports most of the special capabilities available to HTML, such as cascading style sheets and dynamic HTML. Since HTML Help uses the Internet Explorer engine, it does away with the difficulties of testing and coding for various browsers. Finally, compiled HTML Help allows you to distribute Help in a single file (`.chm`).

However, compiled HTML Help works only if the user has IE 3.02 or higher installed. When Windows 2000 becomes the dominant Windows platform, this will no longer be an issue (since IE 5 is built into the system). Under current versions of Windows 9x or NT, full HTML Help functionality requires IE 4.0 or higher. To provide this functionality without requiring users to upgrade to IE 4.0, Microsoft's MSDN Web site (<http://msdn.microsoft.com/library/tools/htmlhelp/wkshp/download.htm>) provides a freely distributable file (`hhupd.exe`) that updates the user's system without installing the full Internet Explorer 4.0 componentry. You can include this file as part of your application's installation, providing virtually transparent installation. This is another place where it pays to plan ahead, letting your installation engineer know requirements in advance.

The sample tree-planting project's requirements for full-text search and a favorites tab dictates using compiled HTML Help as well. (Adding full-text search to an uncompiled HTML Help project requires adding a third-party search engine, which is beyond the scope of this book.)

## Graphical look and feel (CSS)

Cascading style sheets—like styles in Microsoft Word®, Adobe™ PageMaker® or FrameMaker®—define the visual appearance of the page. Style sheets can determine such things as fonts, point sizes, colors, background graphics, and bullet styles. Specific style “classes” can specify consistent formatting for code examples or important notes. Most of this planning and design work can be done at the outset of the project, before the application is completely ready for documentation.

One of the first decisions is the type of cascading style sheet to use:

- inline style sheets, located on each page, directly within the code, near the element they describe;
- embedded style sheets, located on at the beginning of each page;
- linked style sheets, located in an external file (`.css`), but referenced at the beginning of each individual HTML page.

The reusability and centralization of linked style sheets enhances productivity. All the HTML Help files in a project can link to one or two style sheets stored in separate text files. Then, if the requirements used in the early planning and design change and you need to modify the basic project font from Verdana to Arial Narrow, changing a few lines in the style sheets will flow the new design to literally hundreds of HTML pages.

For the ManageX project, I chose a background graphic that mimics lined pads, providing a familiar environment. In addition, using lined paper rather than “bond paper” look provides a more casual feel. I also experimented with different colors of lined paper, hoping to provide an affordance or subtle visual clue signaling the type of information: conceptual topics had a pale gray background, while procedural topics had pale yellow. While the overall look tested well for usability (the pattern is subtle enough to add interest without making the text difficult to read), the difference between colors proved too subtle. Learning from this experience, the tree-planting project uses a single background—white with a leafy border. This sample project also adopts the linked style sheet model.

## Action & interactivity (DHTML)

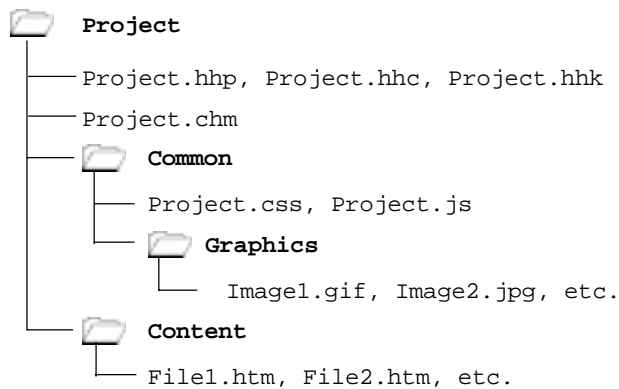
Dynamic HTML takes Web pages another step, adding easy interactivity. Dynamic HTML combines CSS with scripting (usually JavaScript or JScript) to layer information, change colors, display movement, and otherwise provide interactivity and movement. For example, one of the simplest (but most useful) tricks is a few lines of code to show or hide extra information whenever the user clicks on a graphic or piece of text. The sample tree-planting project uses such a “show/hide” script to provide the immediate definitions that users requested. (We could use HTML Help’s built-in pop-up facility, but it is limited to plain text and an unreliably colored background.)

Like the cascading style sheets, DHTML comes in various types, including embedded scripts defined in each HTML page’s header and linked script files. Again, the embedded approach provides the most rapid development path. When I first began designing the ManageX project, I created reusable snippets of code and stored them in a simple text file, cutting and pasting the code into HTML pages as needed. Even with the external .js files, you may want to keep a text file of the code for calling the script (since the actual call must be made within the individual HTML file at the text location). You may also want to investigate Dreamweaver and other HTML editors that provide pre-canned scripts and special storage approaches.

For the tree-planting project we’ll use a single script that displays and hides botanical definitions when the user’s mouse pauses over the scripted section of text. (We could also use the HTML Help Workshop’s pop-up functionality, but then you’re limited to text only. The JScript allows the use of pictures as well.)

## Directory structure

HTML Help has some specific preferences on directory structure. In particular, it prefers that the project and output files (`*.hhp`, `*.hhc`, `*.hhk`, and `*.chm`) be on a directory level *above* the project's component parts. For this reason, I recommend the following directory structure.



### Tree Project 1: Set up the directory structure

On your hard drive, create a directory named `Tree`, with subdirectories named `Common` and `Content`. The `Common` directory should have its own subdirectory, named `Graphics`.

## Crafting reusable elements & templates

Once the planning is done, begin to create the HTML Help system's reusable elements—even if the application is not yet ready for documentation. You may be able to begin writing some topics and use these to identify the necessary reusable components. Then you can begin creating style sheets, DHTML, and graphics, even before the application is far enough along to be fully documented. Although it may seem odd to start working on stylistic elements before writing the text itself, this “plan ahead” approach and the use of templates can prove invaluable both in terms of speed and user satisfaction.

Templates, in the sense used here, are sample HTML files that combine the reusable elements (graphics, CSS, DHTML) with boilerplate text for common tasks. Rapid development can occur once the templates are complete and the application is ready for documentation: Simply open the appropriate template, save it under another name, and make edits as necessary to reflect the individual application concepts and procedures.

The question of speed always arises at deadline time. Experience leads most technical writers to expect a very small window between the moment when the developers stop changing the application and the day when the documentation is due. (In fact, I have even had the distinctly uncomfortable experience of having a documentation deadline fall *prior to* code freeze.) Designing the reusable elements and assembling them into templates in advance allows time to focus on the writing during the inevitable end-of-cycle “crunch.”

In addition, the extra time put into designing the “style” elements pays off with users. While it's essential that the content be solid, the style first catches the eye. Style should not take precedence over substance, but why not have both?

## Create the reusable elements (graphics, CSS, and DHTML)

The audience analysis and other planning steps should provide a good list of elements that will be required by the HTML Help system. In the ManageX project, I knew I needed graphics for the headers to identify each ManageX module as well as designs for the background; a cascading style sheet that would cover common elements for concepts and procedures; and DHTML snippets for expanding lists and show/hide definitions.

For the sample tree-planting project, the previous audience analysis and planning sections provide the following list of reusable elements that can be created before writing the topic:

- **Graphics:** a pale green textured background and a “zone map” identifying planting areas for each tree
- **CSS:** a simple but attractive design to appeal to the aesthetics of gardeners
- **DHTML:** a show/hide snippet for definitions of botanical terms

### The graphics

A course in creating graphics for the Web is beyond the scope of this book. However, you should at least keep these two points in mind:

- **File format:** HTML Help requires Web-style graphics. In general, the file format should be `.jpg` (best for images with continuous tones, such as gradient fills photographic styles) or `.gif` (best for images with sharp differentiation between color areas, such as small icons or items with text that needs to be readable).
- **Color palette:** Anything that will be displayed in a Web browser window (including HTML Help) should be confined to the 216 colors of the browser-safe palette.

For the ManageX project, the graphics included small headers for each project module; icons denoting cautions and hints; and two background papers. This book's companion Web site allows you to download the required graphics for the sample project, including

- the page background (`green.jpg`)
- pictures of different trees (`tree1.jpg`, `tree2.jpg`, etc.)
- zone maps (`zone1.gif`, `zone2.gif`, etc.)

### The style sheets (CSS)

The style sheets for the ManageX project were created in the HoTMetal style sheet editor and refined in Notepad (don't forget that style sheets are just plain text files). These style sheets defined the general elements expected in the project (`BODY`, `P`, `FONT`, `HTML`, `H1`, `H2`, and `TABLE` with its associated headers, data, and rows; ordered and unordered lists), specifying for each the font family, font size, line spacing, and color.

#### ManageX CSS (excerpt 1)

```
BODY { font-style: normal; font-weight: normal;
      font-size: 9pt; line-height: 150%; font-
      family: "Verdana"; color: black;
      background: transparent
      url(..\GRAPHICS\paper.JPG); }
P {font-size: 9pt; line-height: 150%; font-
  family: "Verdana";}
FONT {font-size: 9pt; line-height: 150%; font-
  family: "Verdana";}
HTML { font-family: "Verdana"; }
H1 { font-weight: bold; font-size: 12pt; line-
  height: 140%; font-family: "Verdana";
  color: #CC0000;}
H2 { font-style: normal; font-weight: bold;
  font-size: 10pt; margin-top: 10pt; font-
  family: "Verdana"; color: black; }
```

```

TABLE {font-size: 9pt; line-height: 50%; font-
      family: "Verdana"; }
TH     {font-size: 9pt; line-height: 150%; font-
      family: "Verdana"; }
TD     {font-size: 9pt; line-height: 150%; font-
      family: "Verdana" }
TR     {font-size: 9pt; line-height: 150%; font-
      family: "Verdana"; }

```

In Excerpt 1, notice that this style sheet specifies the font ("Verdana") in every case. Under the style sheet's rules of inheritance, specifying the font for the BODY or HTML should carry through to the other element's font; however, experience showed that such meager specification produces erratic results, especially in tables. It's easiest to simply specify the font for every element in the style sheet—and it's *still* shorter than specifying the font for every occurrence of every element in every file!



For more details on CSS, see Chapter 4, "Doing It In Style: CSS (Cascading Style Sheets)."

### ManageX CSS (excerpt 2)

```

OL     {font-size: 9pt; line-height: 150%; font-
      family: "Verdana";}
UL     {font-size: 9pt; line-height: 150%; margin-
      top: 0pt; font-family: "Verdana"; list-
      style-type: square;}
LI     {font-style: normal; font-weight: normal;
      font-size: 9pt; line-height: 150%; margin-
      top: 6pt; font-family: "Verdana"; color:
      black;}
.INFO  {font-size: 9pt; line-height: 150%; font-
      family: "Verdana"; border-style: ridge;
      border-color: #CCECFE; padding-left: 16px;
      padding-top: 6px; padding-bottom: 6px;
      padding-right: 2 px;}

```

Excerpt 2 presents the portion of the ManageX style sheet defining bulleted and unbulleted lists. Style sheets provide a great deal of control over list items, specifying the bullet type, margins, etc. Note that you can specify

items such as line spacing (line-height) in absolute terms (for example, points) or in relative terms (here, a percentage of the font size). The 150% gives more "air" to the look (standard line spacing is approximately 120% of the font size).

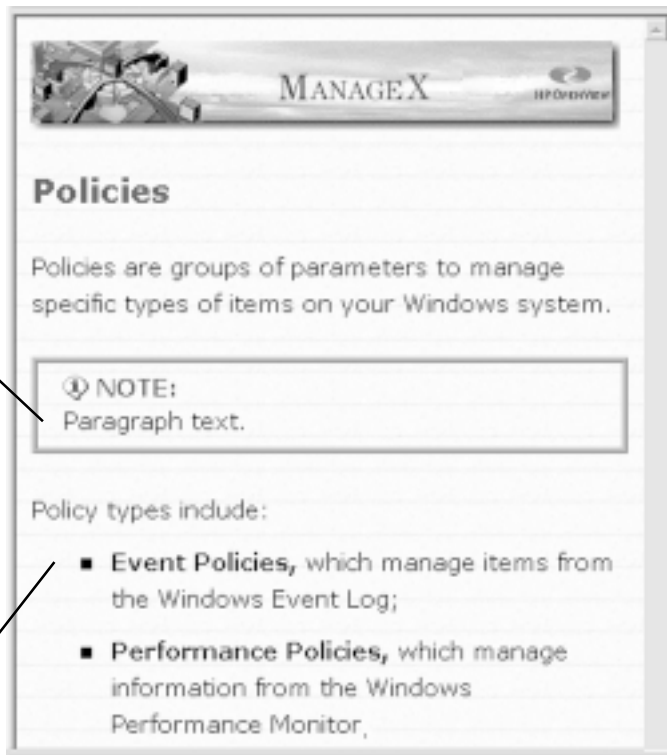
In addition, the special class for `INFO` paragraphs specifies a colored border and extra padding (space between text and border), providing a consistent way to highlight important information. Once this class is defined, prefacing any paragraph in the HTML files with `<P CLASS="INFO">` automatically adds a ridged blue border and extra space around it.

In figure 2.3, the bordered paragraph prefaced with "Note" uses the code shown in the adjacent text box. Note the efficiency of using a class tag

```
<P CLASS="INFO">
<IMG
SRC="Info.gif"
WIDTH="16"
HEIGHT="16"
ALIGN="BOTTOM">
<B><FONT
COLOR="#006699">
NOTE:</FONT></B>
<BR>Paragraph
text</P>
```

```
<UL>
<LI><B>Item 1,</B>
description1;
</LI>

<LI><B>Item 2,
</B>
description2;
</LI>
```



*Figure 2.3 Style sheets save time. This sample HTML Help topic page shows how a few lines of CSS code can save time and avoid repetitive coding in the topics. Callouts show HTML coding used to create the topic elements.*

`<P CLASS="INFO">` in the HTML page. Without the CSS, each "Note" would require the following additional lines of code:

```
.INFO { font-size: 9pt; line-height: 150%; font-
      family: "Verdana"; border-style: ridge;
      border-color: #00699; padding-left: 16px;
      padding-top: 6px; padding-bottom: 6px;
      padding-right: 2 px}
```

### Tree-planting style sheet

The style sheet for the sample project borrows from the ManageX style sheet, but is adjusted to suit the previous audience analysis. For example, the ManageX style sheet uses colors and graphics determined by the company—the signature red of the product (ManageX) and the signature blue of the corporation (Hewlett-Packard). The tree-planting project reflects the look of the garden with green and brown colors as well as leaf and tree graphics. Also, we'll suit the hobbyist audience with a larger, more casual font to enhance readability and approachability. Finally, we'll adapt the INFO class as a DEF class to display easily identifiable pop-up definitions.

#### Tree Project 2: Create the style sheet

Create the tree-planting style sheet by typing the text below into a Notepad file named `tree.css` or downloading it from the companion Web site: [www.jmek.net/html\\_help](http://www.jmek.net/html_help). Be sure to store the file in the `Tree\Common` directory.

```
BODY { font-style: normal; font-weight: normal;
      font-size: 11pt; line-height: 160%; font-
      family: Tahoma, Arial; color: black;
      background-image:
      url(..\GRAPHICS\LeafBkgrd.gif);background-
      color: white; margin-left: 40px; }
P { font-style: normal; font-weight: normal;
   font-size: 11pt; line-height: 160%; font-
   family: Tahoma, Arial; color: black; }
```



The style sheet continues on the next page.

```

HTML { font-style: normal; font-weight: normal;
      font-size: 11pt; line-height: 160%; font-
      family: Tahoma, Arial; color: black; }
LI   { font-style: normal; font-weight: normal;
      font-size: 11pt; line-height: 160%; font-
      family: Tahoma, Arial; color: black; }
H1   { font-weight: bold; font-size: 15pt; line-
      height: 200%; font-family: Tahoma, Arial;
      color: green;}
H2   {font-weight: bold; font-size: 11pt; margin-
      top: 10pt; font-family: Tahoma, Arial; color:
      black; }
OL   {font-size: 11pt; line-height: 150%; font-
      family: Tahoma, Arial;}
UL   {font-size: 11pt; line-height: 150%; margin-
      top: 0pt; font-family: Tahoma, Arial; list-
      style-image: url(TreeBullet.gif); }
LI   {font-weight: normal; font-size: 11pt; line-
      height: 150%; margin-top: 6pt; font-family:
      Tahoma, Arial; color: black }
.DEF {font-size: 11pt; line-height: 150%; font-
      family: Tahoma, Arial; border-style: inset;
      border-color: rgb(51,204,153); padding-left:
      16px; padding-top: 6px; padding-bottom: 6px;
      padding-right: 2 px }
.DEFTERM {font-weight: bold; color:
      rgb(51,204,153);}

```

### The dynamic HTML (DHTML)

The ManageX project used snippets of DHTML to create collapsing lists and other bits of interactivity. One of the most useful of these tricks involves a few lines of code that show or hide extra information whenever the user clicks on a graphic or piece of text. This simple trick requires no external

or embedded styles, just some code in the text to control display of the hidden text. The tree-planting project incorporates this code to create pop-up definitions and/or pictures when the user's mouse moves over specialized terms.

```

Previous text: Whatever lines of text might
                precede the coded section.
<SPAN ONMOUSEOVER="IDx.style.display=''"
        ONMOUSEOUT="IDx.style.display='none'">
  <SPAN CLASS="DEFterm"> clickableText </
  SPAN> </SPAN>
  with a definition. </P>
<P CLASS="DEF" ID="IDx" STYLE="display:none"> This
  is the definition. </P><P CLASS="DEF"
  ID="IDx" STYLE="display:none"> popupText </P>
Following text: Whatever lines of text might
                follow the coded section.

```

The code sample shows a reusable snippet that displays the `popupText` whenever the user's mouse passes over the `clickableText`. When the user's mouse moves past the `clickableText`, the `popupText` disappears.

Certain elements need to be customized for each use of this DHTML:

- `IDx` is the placeholder for the identification code given each instance of show/hide text, with *x* reminding us that each identifier must be unique within a given HTML file. If a single file contains multiple terms requiring pop-up definitions, each term and its definition must be given a unique ID (at a minimum, ID1, ID2, etc.).
- `clickableText` is the placeholder for the trigger point, the text that triggers the display or hiding of the extra information.
- The `previous text` and `following text` paragraphs show how the coded text fits within the larger HTML file.

For the tree-planting project, store these lines in a simple text file or as part of a template. Then cut, paste, and modify whenever needed (you could even add a picture to aid with the definition, if desired). For example, to



This onmouseover snippet is one of the easiest ways to write show/hide code. More elegant approaches using scripts and an external .js file are explored in Chapter 5, "Making Waves: DHTML (Dynamic HTML)".



Making the placeholder text a single word (`IDx`, `clickableText`) makes it easy to select with a double-click when it's time to customize the code.

create a definition of *deciduous*, use the following customization of the code:

You may choose a tree that is

```
<SPAN
  ONMOUSEOVER="ID1.style.display=''"
  ONMOUSEOUT="ID1.style.display='none'">
<SPAN STYLE="DEFterm">deciduous</SPAN>
</SPAN>
```

or one that is

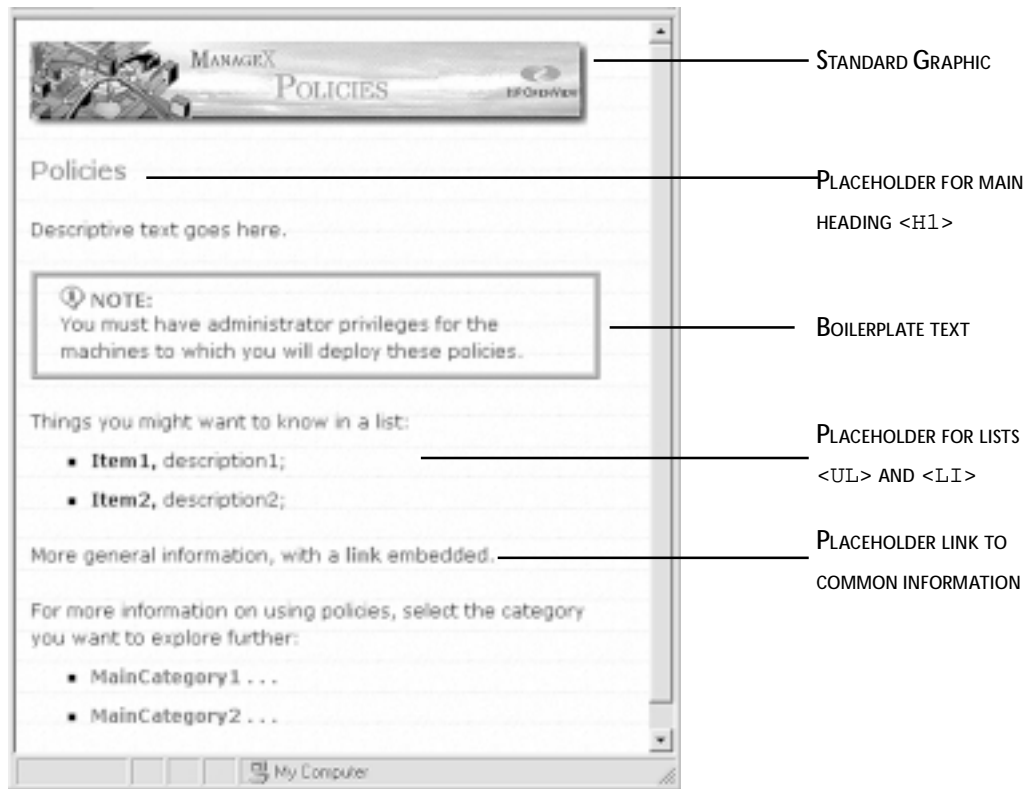
```
<SPAN
  ONMOUSEOVER="ID2.style.display=''"
  ONMOUSEOUT="ID2.style.display='none'">
<SPAN STYLE="DEFterm">evergreen</SPAN></SPAN>
<P CLASS="DEF" ID="ID1" STYLE="display:none">
  Losing its leaves in winter. </P>
<P CLASS="DEF" ID="ID2" STYLE="display:none">
  Retaining most of its leaves all year. </P>
```

## Assemble the templates

For the ManageX project, I began by creating templates for the three general purposes: overviews of each extension (figure 2.4), explanations of concepts, and directions for carrying out procedures. As the project continued, patterns of application behavior emerged, allowing the creation of additional templates for common procedures and tasks.

For rapid, efficient project development, templates should include:

- **HTML page title:** The title appears in your Help system's title bar and as the default identifier elsewhere in the project. Adding



*Figure 2.4 Template elements. This topic page was designed as a template for overview pages within the ManageX project. It provides for a general description, a frequently used warning, and two types of lists. The bottom list uses DHTML to allow each Main Category to expand and display its subcategories.*

`<TITLE>My Application Help </TITLE>` within the `<HEAD>` tags insures that you'll never have a blank title. It also provides a consistent search term when you want to create a more meaningful title.

- **CSS and JS links:** External stylesheet (`*.css`) and script (`*.js`) files provide the most efficient way to make global format changes or add new bits of DHTML.
- **Standard graphics:** For example, you may want a page header. Where appropriate, also include icons for tips and cautions, etc.
- **Layout placeholders:** Easily identifiable text to hold the space for standard layouts elements, including such things as level 1 headers, descriptive paragraphs, and lists of procedural steps.
- **Boilerplate text:** A timesaver for commonly repeated warnings or procedures. Since online Help is nonlinear, information must be repeated in multiple locations. Sometimes you'll want a link to another file, but short or important information should be repeated with boilerplate text.

### Tree Project 3: Create the template (optional)

The tree-planting project uses a single basic template. If you plan to write the topic files (`*.htm`) yourself, create this template by typing the text below into a Notepad file named `Template.htm` within the `Tree\Content` directory. You can also download the template from the Web site:

[www.jmek.net/html\\_help](http://www.jmek.net/html_help).

You can skip this step entirely if you plan to use the prewritten topic files from the Web site.

The following HTML lines create `ProcedureTemplate.htm`:

```

<HTML>
<HEAD>
<TITLE>Tree-Planting Help</TITLE>
<LINK REL="STYLESHEET"
      HREF="..\Common\tree.css">
<!-- The following lines are for projects with
      external script files.
<SCRIPT LANGUAGE="javascript"
      SRC="..\Common\tree.js">
</SCRIPT>
--!>
</HEAD>
<BODY>
<H1>Procedure Title</H1>
<P>Introductory descriptive text about the
      procedure, including a
<SPAN ONMOUSEOVER="ID1.style.display=''"
      ONMOUSEOUT="ID1.style.display='none'">
  <SPAN CLASS="DEFterm"> technicalTerm
  </SPAN>
</SPAN> with a definition. </P>
<P CLASS="DEF" ID="ID1" STYLE="display:none"> This
      is the definition. </P>
<H2>Performing the task (sub-title)</H2>
<OL>
<LI>Step1</LI>
<LI>Step2</LI>
<LI>Step3</LI>
<LI>Step4</LI>
</OL>
</BODY>
</HTML>

```

This code produces the HTML template shown in figure 2.5.



The template provides for an external JavaScript file, which is not used in the chapter's sample project files. These files use only a single bit of JavaScript, which is included in topic files as needed.

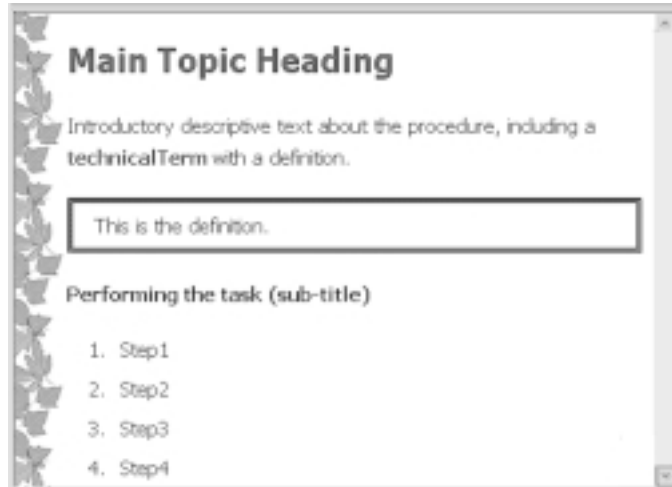


Figure 2.5 The results. The HTML in step 3 creates this procedural template page.

## Write the individual pages

Once the templates are complete, you're ready to begin documenting the application as soon as it's available! For each page, open the template, save it under a more descriptive name, and edit away.

### Tree Project 4: Create the topic files

For the tree-planting project, you won't need to create the topic and graphic files; just download the following from the book's companion Web site. Store topic (\*.htm) files in the `Tree\Content` directory. Graphic files (\*.jpg, \*.gif) go in `Tree\Common\Graphics`.

Main.htm	LeafBkgrd.gif
Choose.htm	Ch_Site.htm
Ch_Zones.htm	ZoneMap.gif
Ch_Species.htm	SantaCruz.htm
SantaCruz.jpg	Oak.htm
Oak.jpg	Prepare.htm
Prep_Site.htm	Prep_Tree.htm
Plant.htm	Planting.gif
Care.htm	

## Creating the HTML Help project

The HTML Help project file specifies and manages the parameters and files of your help system. It brings the required elements (topics, graphics, style sheets, script files, contents, indexing) together with the project parameters (window definitions, compilation parameters, etc.), ultimately compiling all the pieces into a single distributable HTML Help file (`*.chm`).

While the project file can be created directly in Notepad or another text editor, Microsoft's HTML Help Workshop (and third-party HTML Help editors) provides a graphical interface and helps insure that the project file's structure and syntax are correct. Later chapters will describe times and places when you should edit the project file directly; for this overview, we'll use the HTML Help Workshop.

You can download ready-made project files from the companion Web site, but you'll learn more by following through the rest of this chapter and creating your own.

### Project definition

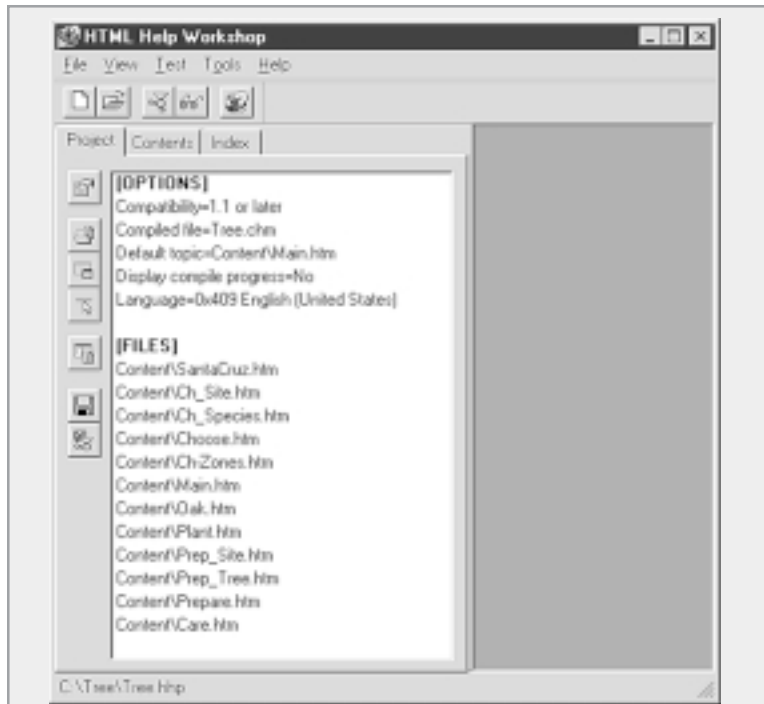
#### Create the project & add its topic files

Creating the new project is as simple as selecting **FILE>NEW** from the HTML Help Workshop's menus and then following the **NEW PROJECT** Wizard. The HTML Help Workshop automatically creates a file with the bare minimum required settings, including the file list, default topic and compiled filenames, language and compatibility settings, and compile settings.

Be sure to immediately add as many topic files to the project as possible. The HTML Help Workshop makes it difficult to specify project parameters until you define the set of working files.

### Tree Project 5: Create & populate the project file

1. In the Workshop, go to the **FILE** menu and select **NEW**. When asked to **SPECIFY WHAT TO CREATE**, choose **PROJECT** and click **OK**.
2. In the **NEW PROJECT** wizard, select **NEXT** to bypass the first page (asking if you want to convert Windows help project).
3. On the Wizard's **NEW PROJECT – DESTINATION** screen, use the **BROWSE** button to locate the top level Tree directory you created earlier for this sample project. Once within the directory, type **Tree** as the project file name. Click **OPEN** to verify the selections, then click **NEXT** to continue.
4. On the Wizard's **NEW PROJECT – EXISTING FILES** screen, select only the **HTML FILES (\*.HTM)** option so that you can quickly add the downloaded topic and graphic files. Click **NEXT** to continue.
5. On the Wizard's **NEW PROJECT – HTML FILES** screen, click **ADD** and then use the standard file **OPEN** box to navigate to the `tree\Content` subdirectory. First select the `main.htm` file and click **OPEN**. (The first file selected is both added to the project and set as the project's default topic.)
6. Back on the Wizard's screen, **NEW PROJECT – HTML FILES** you'll see `Content\Main.htm` has been added to the list. Click **ADD** again and use **CTRL-click** and **SHIFT-click** to select the rest of the topic files. (You could also add the graphic and style sheet files now, but we'll do that outside the Wizard.) Click **NEXT** to continue.
7. Click **FINISH** to have the Wizard create the basic project file (`*.hnp`) and display the result as shown on the next page.  
  
If you opened `tree.hnp` in Notepad, it would look exactly like the left pane of the Workshop shown here, with the



addition of a line saying [ INFOTYPES ] . Obviously enough, the [ FILES ] section reflects your choices in the **New Project Wizard**. In the [ OPTIONS ] section, the compiled file name is the one specified on the first page of the **New Project Wizard**, while the default topic is the first .htm file chosen in the Wizard. The other options are the Workshop defaults (more later).

8. In theory, the HTML Help compiler automatically adds all files referenced in your topics. In practice, this seldom happens, so it is necessary to manually add graphic, style sheet, and script files.



To add the graphic files to the project, click the **Add/Remove Topic Files** button (second from the top on the left side). In the resulting dialog, click **Add** and navigate to the **Tree\Common\Graphics** folder. Force the Workshop to display the graphics files, by typing **\*\*** in the **FILE NAME** text

box and pressing **ENTER**. Select all the graphics files and click **OPEN** to add them to the project. Repeat this procedure to add `Tree\Common\Tree.css` to your project. Click **OK** to return to the main part of the Workshop.

9. From the **FILE** menu, choose **SAVE ALL FILES**.

### Specify parameters

The HTML Help's New Project Wizard defines the following default parameters:

- **Compatibility=1.1 or later:** This setting allows your HTML Help system to be backwardly compatible. (Version 1.0 was available for only a short time and did not offer all the functionality of 1.1 and later versions.)
- **Compiled file:** Specified by you from within the Wizard
- **Default topic:** The first topic (.htm) file selected in the Wizard's **New Files** page.
- **Display compile progress=No:** Setting this parameter to "Yes" causes the compiler to display the file currently being compiled in the right window of the HTML Help Workshop. This option can be useful for troubleshooting a compiled project.
- **Language=0x409 English (United States):** By default, this is the language set as your machine's default locale (in the Regional Settings of the Windows Control Panel). Be aware that language settings other than English may experience difficulties in using the full-text search capability.

For the sample tree-planting project, we'll accept the default options and add two more:

- **Title:** Determines what displays in the title bar at the top of the compiled HTML Help project.
- **Compile full-text search information:** Allows the user to search for any word that appears anywhere in the text of the topics.

#### Tree Project 6: Specify additional project options


1. In the left window of the HTML Help Workshop, double-click [OPTIONS].
2. On the GENERAL tab of the OPTIONS dialog, type *Tree-Planting* in the text box labeled TITLE.
3. Click the COMPILER tab. Select the COMPILE FULL-TEXT SEARCH INFORMATION option.
4. From the FILE menu, choose SAVE ALL FILES.

Now we'll go on to the slightly more complicated option of defining the default window.

#### Window definition: size & styles

Window types are one of the tricky parts of the HTML Help Workshop. Although the New Project Wizard does not specify a default window type, erratic behavior is common in projects without one. In addition, HTML Help seems happiest if the default window type is named "main," especially in merged projects. You can modify this window in the WINDOW TYPES dialog.

### Tree Project 7: Define the default window

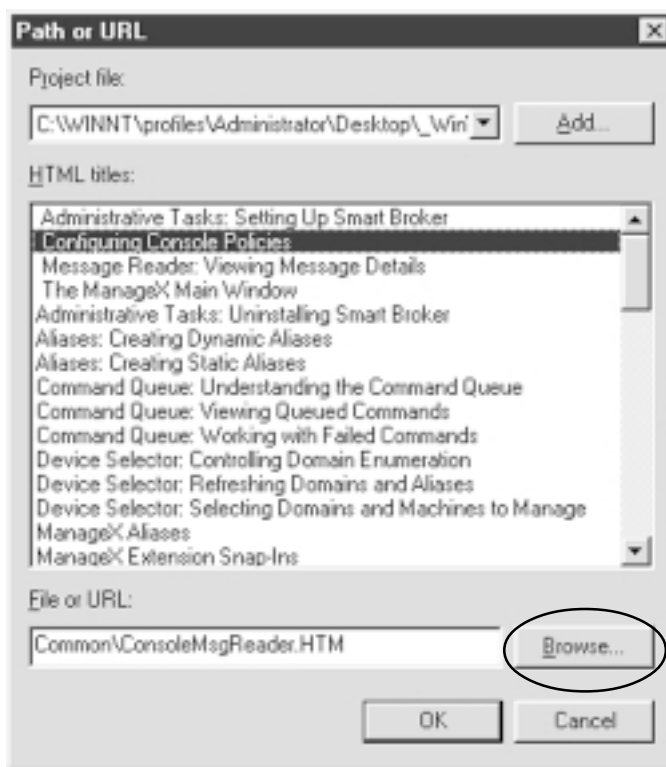
1. In the HTML Help Workshop, click the **ADD/MODIFY WINDOWS DEFINITIONS** button (third from the top on the left) 
2. In the **ADD A NEW WINDOW TYPE** dialog, type **Main** for the window type name.
3. On the **GENERAL** tab of the **WINDOWS TYPES** dialog, type **Tree-Planting** as the **TITLE BAR TEXT**.
4. Click the **NAVIGATION PANE** tab. If you've been experimenting and have created any other types, make sure that the **WINDOW TYPE** text box shows **Main**.
  - Set the **NAVIGATION PANE WIDTH** to 200.
  - Select the **AUTO SYNC** option to make sure that the topic in the navigation (left) pane of your HTML Help system always places the highlight on the topic pertaining to the currently open **MAIN** window.
  - Choose **Contents** as the **DEFAULT TAB**.
5. Click the **POSITION** tab and make sure that **Main** is the currently selected **WINDOW TYPE**. Then click the **AUTOSIZER** button. Drag the model **AUTOSIZER** window to the upper right corner of your screen, then resize the window to a comfortable size. This sets the size and location in which your HTML Help system opens by default. Click **OK** and you'll see your choices reflected in the **WINDOW SIZE AND POSITION** fields (I recommend a width of about 550 and a height of about 400).
6. Back in the **POSITION** tab, select **SAVE USER DEFINED WINDOW POSITION AFTER FIRST USE** to allow your users the courtesy of sizing and positioning the window to their liking. Click **OK** to save all choices.
7. From the **FILE** menu, choose **SAVE ALL FILES**.

## Building the navigation system

### Contents

HTML Help's Table of Contents feature is right on the *bleeding* edge. Things change without warning and you're often not sure what you did to fix it! In version 1.0 of the HTML Help compiler, for example, books often lost their settings and became pages, while the indent level changed apparently at random. Most of this seemed fixed in 1.1, but such behavior is still reported occasionally. At least three bugs persist through version 1.3.

- Adding a contents entry using the list of HTML files in the **ADD PATH OR URL** dialog is apt to add the wrong file. In figure 1.6, taken from the ManageX system, the HTML title ("Configuring Console Policies") is




*Figure 1.6 Bug alert. The Add Path or URL dialog loses the synchronization between HTML titles and File or URL paths. Choose the correct file by using the Browse button.*

the title of a file named `CP-configuring.htm`. The filename shown as the **File or URL** (`Common\ConsoleMsgReader.HTM`) actually refers to the third item in the HTML files list ("Message Reader: Viewing Message Details"). The workaround? Use the **Browse** button to access a standard Windows browse dialog where you can choose your topics by their filename.

- There's also a video refresh bug: If your table of contents grows longer than the window can show, the display does not refresh when you close the **TABLE OF CONTENTS ENTRY** tab after adding a new entry. You must force a refresh by going to one of the other tabs and back again.
- It takes two tries to assign a non-standard icon to a contents entry. If you add an entry (book or page) on the **GENERAL** tab and go directly to the **ADVANCED** tab, the **IMAGE NUMBER** spin control will not work. You must click **OK** to exit the dialog, then select the new entry in the contents list and click on the **EDIT** icon (the pencil). Now when you go to the **ADVANCED** tab, you will be able to change the icon image.

### Tree Project 8: Create the table of contents

1. To create a new, blank contents file, select the **CONTENTS** tab of the HTML Help Workshop. When a dialog labeled **TABLE OF CONTENTS NOT SPECIFIED** appears, choose **CREATE A NEW CONTENTS FILE** and click **OK**. In the **SAVE AS** dialog that appears, navigate to the **TREE** directory and type **Tree** as the **FILE NAME**, then click **SAVE**.
2. To tie the contents to your default window, click the  **CONTENTS PROPERTIES** button (top one on the left). Type **main** as the **DEFAULT WINDOW**, then click **OK**.
3. Right-click in the blank window on the left of the HTML Help Workshop and select **INSERT HEADING** from the shortcut menu. On the **GENERAL** tab, type *Introduction* as the **ENTRY TITLE**, then click **ADD**. Use the **BROWSE** button to locate and select `Main.htm`. Click **OK** twice to exit.

4. Back in the main workshop window, right-click in the left pane and select **INSERT HEADING** from the shortcut menu again. Answer **No** when asked if you want to insert the entry at the beginning of the table of contents. This time, type **Choosing a Tree** as your **ENTRY TITLE**, and use the **ADD** and **BROWSE** buttons to select **Choose.htm**. Click **OK** twice to exit.
5. To use a book icon to set off the heading topics, click on your new **Introduction** entry, then click the **EDIT SELECTION** button (fourth from the top on the left). Choose its **ADVANCED** tab, then use the **INDEX IMAGE** spin controls to change to 1 (a book). Click **OK** to save your changes.
6. Right-click again in left pane of the workshop window, but this time choose **INSERT TOPIC** from the shortcut menu. Add the following entry titles and associated files:
  - **Identifying your planting zone** (**Ch\_zone.htm**)
  - **Selecting and assessing your site** (**Ch\_site.htm**)
  - **Choosing appropriate species** (**Ch\_species.htm**)
  - **Oak** (**Oak.htm**)
  - **Santa Cruz redwood** (**SantaCruz.htm**)
7. When you have added the above files, right-click on the entry for **Oak** and choose **MOVE RIGHT** from the shortcut menu. Answer **Yes** to the prompt asking if you're sure you want to do this. Both the **Oak** and the **Santa Cruz redwood** entry beneath it should move to the right, appearing as subtopics of the **Choosing appropriate species** entry.
8. With the **Santa Cruz redwood** entry selected, right-click and select **INSERT HEADING** from the shortcut menu. Add a heading titled **Preparing to Plant the Tree** that points to



the `prepare.htm` file. Select the new heading, click the **EDIT SELECTION** button, and use the **ADVANCED** tab to change its image to a book (1). Right-click this new entry and choose **MOVE LEFT** from the shortcut menu. Move the entry to the left one more time, so that it appears as a top-level heading.

9. Add the following files as sub-topics of the **Preparing to Plant the Tree** heading.
  - **Preparing the site** (`prep_site.htm`)
  - **Preparing the tree** (`prep_tree.htm`)
10. Add the following files as top-level headings with the book icon.
  - **Planting the tree** (`plant.htm`)
  - **Caring for the new tree** (`care.htm`)
11. From the **FILE** menu, choose **SAVE ALL FILES**.

## Index

Indexing is another complicated topic in the HTML Help Workshop. Beyond the normal indexing questions of choosing terms and organizing synonyms, you must also choose whether to insert the index entries in the individual HTML files or directly in an index (`*.hbk`) file and whether or not to use a binary index. There are also questions of **ALINKS** and **KLINKS**. The more complex issues are covered in a later chapter. For the tree-planting project, we'll simply add a few sample index entries to an index file created within the HTML Help Workshop.

### Tree Project 9: Create the index

1. To create a new, blank index file, select the **INDEX** tab of the HTML Help Workshop. When a dialog labeled **INDEX NOT SPECIFIED** appears, choose **CREATE A NEW INDEX FILE** and click **OK**. In the **SAVE AS** dialog that appears, navigate to the **Tree** directory and type **Tree** as the **FILE NAME**, then click **SAVE**.
2. To tie the contents to your default window, click the **INDEX PROPERTIES** button (top one on the left).  Type **main** as the **DEFAULT WINDOW**, then click **OK**.
3. Right-click in the blank window on the left of the HTML Help Workshop and select **INSERT KEYWORD** from the shortcut menu. On the **GENERAL** tab, type **pine** as the **KEYWORD**, then click **ADD**. Use the **BROWSE** button to locate and select **santaCruz.htm**. Click **OK** twice to exit.
4. Repeat the previous step to add the following keywords and their associated files. You may add as many additional entries as you wish.
  - **redwood** (**santaCruz.htm**)
  - **evergreen** (**santaCruz.htm**)
  - **deciduous** (**oak.htm**)
  - **soil** (**ch\_site.htm** and **prep\_site.htm**). You can add multiple topic files to a single keyword by repeatedly choosing the **ADD** button in the **INDEX ENTRY** dialog.
5. When you finish adding entries, click the **SORT KEYWORDS ALPHABETICALLY** button (third from the bottom on the left).  The HTML Help Workshop automatically alphabetizes your index.

## Compiling & testing the project

In general, compiling is the most automatic part of the process. And, if it works when you test it, you're done! (Troubleshooting will be covered in a later chapter.)

### Tree Project 10: Compile & test

1. Click the **PROJECT** tab of the HTML Help Workshop. In the project file pane on the left, verify that the project contains the following two lines:

**Contents file=Tree.hhc**

**Index file=Tree.hhk**

If you don't see these lines, click the **CHANGE PROJECT OPTIONS** button (top on the left) and choose its **FILES** tab. Type `Tree.hhc` as the **CONTENTS FILE** and `Tree.hhk` as the **INDEX FILE**. Click **OK** to save your changes.

2. Click the **ADD/MODIFY WINDOW DEFINITIONS** (third button on the left) and click on its **FILES** tab. Use the drop-down lists to set `Tree.hhc` as the **TOC** and `Tree.hhk` as the **INDEX**. Click **OK** to save your changes.



3. Select the **COMPILE HTML FILE** button (third from the left on the top). Select both the **SAVE ALL FILES BEFORE COMPILING** and **AUTOMATICALLY DISPLAY COMPILED HELP FILE WHEN DONE** options.



4. Click **COMPILE** and watch the compiler's progress in the right pane of the workshop.

5. If the compiled file does not display automatically, click the **VIEW COMPILED FILE** button (fourth from the left on the top). Verify that the correct file is listed and click **VIEW**.



Enjoy your first compiled HTML Help file!

## Merging modular projects

Merging HTML Help modules allows you to customize online Help systems by quickly adding and removing a “chapter” of information from the project. When merged modules are present on the user’s machine, they appear seamlessly in the table of contents, index, and full-text search; when the modules are deleted, they disappear just as seamlessly.

For example, the ManageX project was designed to support the addition of Smart Plug-Ins (SPIs)—special modules that add extra functionality in key areas. Not all SPIs were available when the main ManageX application was released and not all users would buy all SPIs. So I created the project with “stubs” for any SPI Help modules that would be created and/or purchased later on. The tree-planting project could use the same approach to allow for the creation of specialized zone planting guides.

Note that the actual merging process occurs when the user opens the HTML Help file, *not* when the Help author compiles the files. Because of this, merging occurs only for those files that are actually present on the user’s system. This procedure lets you plan for expansion since the merged files do not need to exist when the master file is compiled.

### Efficiency

Planning and creating templates, style sheets, and script files saves a great deal of time during the production of the Help system’s topic files (`*.htm`). If you’re merging multiple modules, additional time savings accrue from the creation of a template for the project files (`*.hhp`, `*.hhc`, `*.hhk`) used in each module. When the first ManageX module became available, I used my HTML templates to create the documentation topics. Then I assembled the HTML files into a project in the HTML Help workshop. This project file became a template for the other modules. In fact, I discovered that the easiest way to create new modules was to copy the first module’s project files, remove all the topic files and TOC/index entries, and then add the topic files and contents/index entries for the new module.

## Merging requirements

The actual merging requires you to perform the following tasks:

1. Set up project options:
  - Insure that compatibility is set for version 1.1 or later.
  - Insure that a binary index is used.
  - Specify merge files in the **Project Options**.
  - Define a consistent default window for all modules.
2. Create cross-references among modules:
  - Prepare for merged tables of contents.
  - Use proper syntax for cross-module links.

All tasks must be performed in the master module (in this case, `Tree.chm`). The merged modules (here, `zone01.chm`, `zone02.chm`, etc.) only need to use the consistent default window and the appropriate syntax for cross-module links.

## Setup options

### Ensuring compatibility & binary index options

These options are defined in the HTML Help Workshop's **PROJECT** tab. To set them up, go into the **PROJECT OPTIONS** and select the **COMPILER** tab. Set the **COMPATIBILITY** to **1.1 OR LATER** and select the **CREATE A BINARY INDEX** option.

### Specifying merge files

If you're creating a modular merged project, you must specify the files to be merged in the **PROJECT** tab's **OPTIONS > MERGE FILES** dialog. You can specify dummy merge files (`zone01.chm`, `zone02.chm`, etc.) to take care of future additions. Just don't forget to use these same names in your table of contents and as the name of the final merged files.

## Window definitions

If you're merging modules, make sure you specify the same window name (for example, "main") as the default window for the project, contents, and index. (Each tab has a **PROPERTIES** page which allows you to specify its default window.)

## Merging references

### Preparing merged tables of contents

Under 1.1 and earlier versions of the HTML Help Workshop, merging files in the table of contents was definitely "bleeding edge." Sometimes they'd show up; most times they wouldn't. The pages were in the `.chm`; you could find them through the index or search facilities, but they wouldn't show up in the contents. Since the ManageX project wanted to preserve the appearance of merged modules, I used a complicated workaround with duplicates of the introductory pages included as part of the master file with links to the rest of the file. Unfortunately, this had unexpected side effects, including a situation where users who updated an individual module lost all the rest of the HTML Help system.

Fortunately, versions 1.2 and later of the HTML Help Workshop seem to have fixed most of the problems. When the merged file is present, it appears in the table of contents; when the merged file is not available, no entry appears.

This procedure adds an "Include" statement for the contents of your merged file. When the specified merge files are present on the user's system, they will appear in the contents; if the files are not present, no entry appears.



This syntax works with all versions of Internet Explorer. If you know users have IE 4.0 or later installed, you can replace `mk:@MSITStore` with `ms-its`.

## Using proper cross-file link syntax

HTML Help requires a special protocol for linking to pages within a compiled file. Links between pages in a single compiled HTML Help file (`*.chm`) use the standard format, with a relative path:

```
<A HREF="directory\page.htm">
```

In order to link to a page in another compiled HTML file, use this format:

```
<A HREF="ms-its:CompiledFile.chm::\page.htm">
```

where `page.htm` is the name of the specific topic you want to link to and `CompiledFile.chm` is the name of the merged compiled help file containing it.

## Tree Project Appendix: Creating a merged project

The following steps summarize the procedures necessary to set up a Master module for a merged Help project. We'll perform these steps on the Tree-Planting project; however, since we do not have the additional Zone modules, we will not be able to test the results.

1. In the left window of the HTML Help Workshop's **PROJECT** tab, double-click [ **OPTIONS** ] .
2. Click the **COMPILER** tab. Use the drop-down list to set the **COMPATIBILITY** option to **1.1 or later** and click to select the **CREATE A BINARY INDEX** option.
3. Click the **MERGE FILES** tab. Click the **ADD** button. In the **ADD MERGE FILE** dialog that appears, type `zone01.chm` in the text box titled **SPECIFY THE NAME OF THE FILE TO MERGE**. Merged projects work best if all merged modules are in the same directory, so you do not need to specify a path . Click **OK**. (In a real merged project, you would repeat this step for each merged module.)

4. Click **OK** again to close the **OPTIONS** dialog.
5. Since the tree-planting project already uses "Main" as the default window everywhere, you need make no more changes in this file. However, as you would create the "Zone" modules, you would need to make sure that all the window definitions were specified as **main** and used the same parameters as this master file.
6. To prepare for the merging of modules into the table of contents, go to the HTML Help Workshop's **CONTENTS** tab, right-click the entry named **Choosing appropriate species**, and choose **INSERT FILE** from the shortcut menu.
7. In the resulting dialog, type `zone01.chm::\Zone01.hhc` as the **FILE TO INCLUDE** and answer **Yes** to the prompt asking if you want to include the file even though it can't be found. (In a real merged project, you would repeat this step and the preceding one for each additional merged file.)
8. On the **PROJECT** tab, select the **COMPILE HTML FILE** button (third from the left on the top). Select the **SAVE ALL FILES BEFORE COMPILING** option, then click **COMPILE**.

