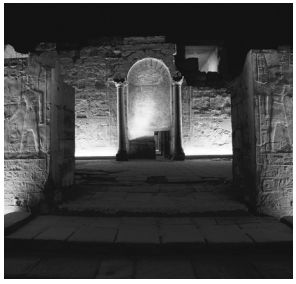

4

Converting HTML to XHTML

UNLESS YOU'VE NEVER CREATED A WEB PAGE BEFORE, you're probably interested in how to convert an existing HTML page to XHTML. The Web has expanded to consist of an overwhelming number of Web pages—most of which are created with HTML. Undoubtedly, many Web developers will make the switch to XHTML before long. If you're one of these developers, this chapter prepares you for the task.

Differences Between HTML and XHTML

To convert an HTML document to XHTML, you must first identify the differences between the two languages. When it comes to these differences, there are two fundamental truths that any Web developer must understand:

- HTML will not be developed any further. XHTML 1.0 is what you need to learn.
- XHTML is a reformulation of HTML 4 in XML 1.0.

This section focuses on the differences between HTML and XHTML, but as this discussion progresses, keep in mind that XHTML is just the next evolutionary step in the HTML family. The elements used to create an HTML Web page are the very same elements used to create an XHTML Web page. XHTML is just a restatement of HTML goals in the form of an open, extensible markup language based on XML.

Since late 1998, there has been a migration toward XML-related technologies. The XML hype is no surprise. After all, XML gives authors the capability to deliver standardized markup while separating display from structure, which makes maintenance and parsing much easier. In addition, XML allows you to define custom vocabularies (in other words, create your own element tags). Who wouldn't be chomping at the bit?

With the influx of portable Web-enabled browsers, there is another reason authors are ready to tackle XML: strict, clean, standardized code.

In the past, HTML did what was needed, but it's outlived its usefulness. Most HTML documents are bloated with unnecessary code, which makes it difficult to create pages compatible with nonconventional platforms, such as handheld devices. Who wants to force the user to mess with bloated—and yes, slow—pages? Bloated code is just one of the many current limitations of HTML.

Limitations of HTML

HTML was never designed to do what we expect of it today. You can look at HTML markup and the various iterations of the language itself, and see how it has evolved as a way to keep up with the Web's demands. What began as a simple structural language has evolved to include stylistic markup that affects display and not structure. As Netscape and Internet Explorer fought for market share, they too introduced their own HTML elements, which forced Web designers to learn different nonstandard elements. Why has HTML fallen behind in the rat race? Here are a few limitations to working with HTML today:

- Lack of large amounts of semantic markup leads to difficulty in describing and exchanging data.
- Proprietary and stylistic elements create sloppy, bloated code that is difficult to maintain—not to mention slow to download.
- The predefined element set (as well as the way developers use the element set) does not work well with nonconventional platforms, such as handheld devices.

These are some of the reasons HTML has seen its day. To learn more about why XHTML will make any Web designer grin from ear to ear, see Chapter 2, "All About Markup."

Viewing Web Pages

According to the Web gods, by 2002 over 75 percent of desktop users will view Web pages on platforms other than your average desktop machine, for example, mobile phones, refrigerators, and Palm Pilots.

Major Differences

The major differences between HTML and XHTML can be broken into two groups: syntactical and functional. Once browsers get up to speed, XHTML will not be as forgiving as HTML. We use the future tense here because right now, XHTML functions much like HTML in older browsers. To understand how XHTML can render in non-XML-compatible browsers, see the section titled "Compatibility Issues and Browser Requirements" later in this chapter. If you're familiar with the HTML-browser debate, you probably know two things:

- Not all browsers are alike: Different browsers support different elements and some proprietary elements of their own. Until recently, browser developers were not concerned with standards.
- Browsers let authors get away with murder, theoretically anyway.

If you want to test the previous theories, do a quick test. Create the following XHTML document in a text editor and save it on your hard drive with a .htm extension:

```
<html>
<body>
<h4>This is a heading 4</h4>
<p>Here is our paragraph</p>
</html>
```

Right off the bat, you *should* notice a couple mistakes. First, there's not a `head` element or a `title` element (which are both required for a valid HTML document). Second, the closing `body` tag is not present.

After you save the document on your hard drive, open it in your browser. To open an (X)HTML document in your browser, select File, Open, click the Browse button, select the document name, and then click Open. The document should render in your browser without a hitch, as shown in Figure 4.1.

With XHTML, all bets are off. When you create an XHTML document, you're instructing the browser to follow XML rules. If you make a mistake, such as omitting a closing tag or typing element names in uppercase, your page will not be well formed or valid, and the result will be an ugly error message. The terms *well formed* and *valid* are essential to XHTML. If you do not know what they mean, see Chapter 2, "All About Markup."

As mentioned earlier in this section, there are two categories of differences between XHTML and HTML: syntactical and functional. The syntactical differences have to do with the structure of the document (its syntax). As you become well versed in XML (and you will because XHTML is an XML application), syntax becomes increasingly important. When we mention XML rules, we're talking about XML syntax rules. One of the advantages of XML is its standardized—and yes, strict—markup. This means that the first significant difference between HTML and XHTML is syntax.

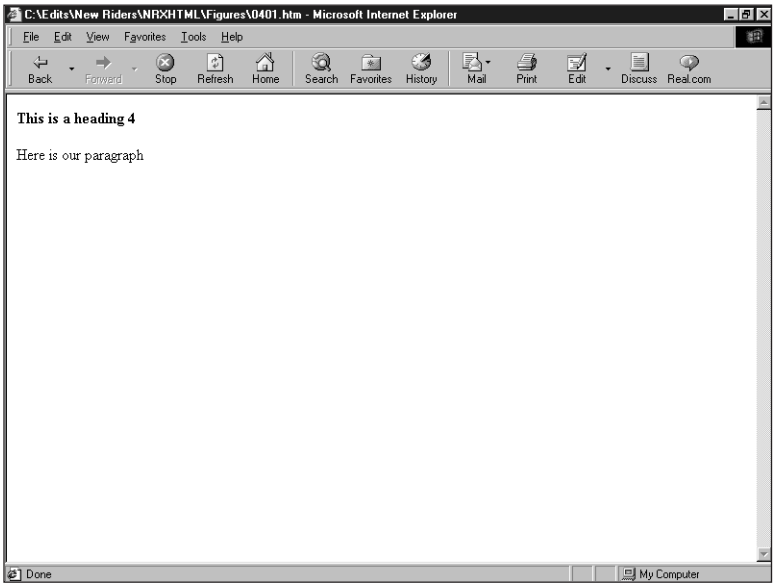

Figure 4.1 The HTML document is displayed, even with mistakes.

HTML follows its own syntax rules based on a predefined set of elements (most of which define style or structure), whereas XHTML must adhere to the syntax rules defined by the XML specification. (We identify each rule in detail later in this chapter in the "XML Syntax Rules" section.) With the capability to create your own elements in XML, minus the capability to manipulate style, XML allows only for elements that define the meaning and structure of their content. That alone doesn't do more than HTML. To really take advantage of the XHTML benefits, you should read Chapter 12, "The Benefits of Extensibility," and Chapter 13, "Where the Future Leads, XHTML Follows," and learn more about extending your markup.

The next set of major differences is driven by function. As discussed earlier, HTML has outlived its usefulness, and XHTML intends to pick up where HTML left off. Most advantages found in XHTML actually come from XML. Because XHTML is an application of XML (which means it follows the rules of XML), it reaps the rewards of extensibility and portability.

When someone from the XHTML community says *extensible*, he or she means that XHTML can be extended beyond its predefined core elements. With extensibility, other XML applications, such as Scalable Vector Graphics (SVG) and your very own XML creations, can be incorporated in XHTML documents.

The second XML advantage is *portability*. With XML, a single page can result in multiple outputs. After the markup and content is compiled into an XML (or in our case, an XHTML) document, various style sheets can be applied that control different outputs. And, who's to say that output is only for display. Maybe the data needs to be sent to a database at another company, with XML. This type of data transfer is now an option. The idea is to only write and modify the content once. (Now that is a media we can get behind!) However, remember that you can't just write an XHTML document and expect it to start calling up other companies and sharing information; there is some work involved. The key is that it's possible.

Compatibility Issues and Browser Requirements

Good news: XHTML documents will render in older browsers. There are a few glitches, but there are simple workarounds that fix the problems. Here they are, one by one:

- Empty elements should include a trailing space after the element name and before `/>` (for example, `<br />`).
- Do not minimize elements that are not defined as empty, even if they do not appear to contain content. For example, sometimes the `p` element is used alone with no content. If this occurs, be sure to write it as `<p></p>` and not `<p />`.
- Avoid using multiple spaces or line breaks within an element's tag(s). In XML, using line breaks is common practice. For example:

```
<book  type="paperback"
       isbn="99-9999-999-9"
              year="1972">
```

  In XHTML, do not include line breaks, for example:

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

Note that throughout this book, we insert spaces in markup for readability purposes only. When you write your markup, be sure to follow the spacing rules outlined in this section.

This is by no means an exhaustive list. For more information on compatibility, visit `www.w3.org/TR/xhtml1/#guidelines` or flip to Appendix A "XHTML 1.0: The Extensible HyperText Marlkup Language," which contains the entire XHTML Specification, including the HTML Compatibility Guidelines.

XML-Compatible Browsers

There are many XML-compatible browsers that render XHTML documents perfectly, such as Amaya, Mozilla, and Internet Explorer 5.

- Amaya is offered by the W3C and boasts support for most W3C standards, including heavy-duty CSS support. (We like it here at the office.) As far as popularity goes, however, Amaya is only used by a fraction of the Web-browsing community.

- Mozilla was created as an open-source venture, and is incorporated into the most recent version of Netscape 6. It too boasts support for the most common Web standards, such as HTML and XML (and therefore XHTML). Mozilla also includes heavy-duty CSS support.
- Internet Explorer 5 is Microsoft's premier browser. Although not perfect, Internet Explorer 5 does support XML. One word of warning, however: it does not support the full CSS specification, and CSS is important to XHTML display. The next version of Internet Explorer should include better CSS support.

For a complete listing of XML-compatible browsers, see `www.xmlsoftware.com/browsers/`.

Mechanical Translation from HTML to XHTML

Translation from HTML to XHTML is truly a breeze. Instead of thinking of it as a full-blown conversion process, think of it as cleaning up your HTML document. XHTML uses the same elements—the only difference is syntax and rules. We break them down for you in the following sections.

XML Syntax Rules

All XML documents have something in common: syntax. There are a few, easy rules to remember when you work with XML documents. Here they are, one by one:

- All elements must be contained by a root element (also called a document element). For XHTML, the `html` element is the root element: `<html>...</html>` contains all other elements
- If your document adheres to a Document Type Definition (DTD), the document must include a document type declaration. For XHTML, the document type declaration is formed as follows:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

  The `DOCTYPE` declaration is not an element; it's a declaration and has its own syntax. All `DOCTYPE` declarations begin with an exclamation point and uppercase keyword (for example, `<!DOCTYPE>`). This is not negotiable. If you're new to the term DTD, please read Chapter 2.
- All element and attribute names must be lowercase (for example, `<head>` not `<HEAD>`).
- All nonempty elements must have a closing tag (for example, `<p>...</p>`).

- All empty elements must use the following syntax: `<img />`. (For backward-compatibility reasons, although it's not required, be sure to include a space between the element name and `/>`.)
- Elements must be nested correctly (for example, `<p><b>This is correct</b></p>` and `<p><b>This is not correct</p></b>`).
- All attributes must have values and those values must be contained by single or double quotation marks. This means that the standalone attributes used in HTML (such as `<td nowrap>`) are no longer valid). The correct form is `<td nowrap="nowrap">`.

The main syntax rules end there. However, there are a few XHTML-specific rules you have to follow as you convert your documents.

XHTML-Specific Rules

To become familiar with the XHTML-specific rules, let's go through an HTML document from top to bottom and identify the necessary changes and additions to required elements.

The first item on an HTML page is normally the `html` element. This is no longer the case. There are a few pieces of markup that must come at the beginning of any XHTML document.

First, you must include a `DOCTYPE` declaration (also known as a document type declaration). You may also include XML version and encoding information (which is optional) in the XML declaration right before the `DOCTYPE` declaration. We strongly recommend that you include the XML declaration. For more on the XML declaration, see Chapter 2. The correct markup takes the form:

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC
      "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

The next item on the list is the `html` element. In XHTML, this element is called the root element. The *root element* contains all other elements. To read more on including a root element, see Chapter 3, "Overview of Element Structures." Not only is the `html` element required, but it must also contain a predefined attribute-value pair. This specific required attribute is called an *XML namespace*. XML namespaces are commonly used in XML documents. The specific namespace used for XHTML documents is often referred to as the *XHTML namespace* and is a way to uniquely identify the set of elements as XHTML elements. The correct XHTML namespace is as follows:

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

Imagine that you're in the world of XML and you want to combine your XHTML document with some elements you made from scratch. In the list of elements you named, you decided to use a `title` element type to represent the title of a book. This means that your combined document now has two `title` element types with two completely different meanings.

How do you tell the XML processor (a browser is one type of processor) that the XHTML `title` element is the title of the document and your `title` element represents the title of a book? You use a namespace to uniquely separate the two.

Namespaces are like surnames. Defining a namespace in the root element means that all elements contained by the `html` element belong to the XHTML element set. Any elements that are not contained by the `html` element or that have their own namespaces attached will belong to a different set of elements. The namespace debate is not yet settled at the W3C—for a while, there was an argument about how and when to use namespaces. To learn more about namespaces, see Chapter 2 or check out the W3C site at `www.w3.org/TR/REC-xml-names/`.

Next, you must include the `head`, `title`, and `body` structural elements, which are required in XHTML. The one exception is if you're creating a frameset document. In this case, you have to replace the `body` element with the `frameset` element.

Now that you know the differences between XHTML and HTML, it's time to convert an HTML document to an XHTML document.

Step-by-Step Examples

If you're following along at your computer, go to the Chapter 4 examples on the CD and look for Example 4.1. For this exercise, we're going to take this fairly unattractive code and clean it up by hand. The markup behind the page is shown in Example 4.1.

Example 4.1 **This Markup Needs to Be Cleaned Up**

```
<html>
<head>
<title>Picasso, Pablo</title>
</head>
<body>
<h1>Pablo Picasso</H1>

<ul>
<li>
<img src="http://www.artic.edu/aic/collections/modern/pictures/E31631.jpg"
alt="Picasso's The Old Guitarist">
<cite>Picasso's The Old Guitarist
</cite>
<br>
Oil on panel, 1903;
Helen Birch Bartlett Memorial Collection, 1926.253
```

```
<li>
<img src="http://www.nga.gov/image/a00022/a00022bd.jpg" alt="Picasso's The
Tragedy">
<cite>The Tragedy</cite>
<br>
Oil on panel, 1903; Chester Dale Collection, 1963.10.196
<li>
<img src="http://www.diamondial.org/limage/70.190.JPEG" alt="Picasso's Melancholy
Woman">
<cite>Melancholy Woman</cite>
<br>
Oil on canvas, 1902; Bequest of Robert H. Tannahill, 70.190
</ul>
</body>
</html>
```

If you want to make the changes as we do in this book, save the file from the CD in a folder on your hard drive, and open it in a text editor.

When converting a document by hand, first look for syntax differences. For example:

- Change all element and attribute names to lowercase.
- Add quotation marks to all attribute values.
- Add appropriate closing tags.
- Add trailing space and forward slash (/) to all empty elements.

After you make the necessary syntax changes, you need to add a few XHTML-specific pieces of information:

- **Add the XML declaration.** To do this, add the following markup as the first line of the document:

  ```
  <?xml version="1.0"?>
  ```

- **Add a `DOCTYPE` declaration.** To do this, you have three options to choose from (to learn more about these options, see Chapter 2). To add this markup correctly, insert the following markup as the second line of the document:

  ```
  <!DOCTYPE html PUBLIC
        "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
  ```

- **Add the XHTML namespace to the root element.** To do this, simply add the following attribute-value pair to the `html` element:

  ```
  <html xmlns="http://www.w3.org/1999/xhtml">
  ```

After you make the previous changes, the resulting code should look like Example 4.2.

Example 4.2 **This XHTML Document Is Syntactically Correct**

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC
      "-//W3C//DTD XHTML 1.0 Strict//EN"
      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Picasso, Pablo</title>
</head>
<body>
<h1>Pablo Picasso</h1>

<ul>
<li>
<img src="http://www.artic.edu/aic/collections/modern/pictures/E31631.jpg"
alt="Picasso's The Old Guitarist" />
<cite>Picasso's The Old Guitarist</cite>
</li>
<br />
Oil on panel, 1903;
Helen Birch Bartlett Memorial Collection, 1926.253
<li>
<img src="http://www.nga.gov/image/a00022/a00022bd.jpg" alt="Picasso's The
Tragedy" />
<cite>The Tragedy</cite>
</li>
<br />
Oil on panel, 1903; Chester Dale Collection, 1963.10.196
<li>
<img src="http://www.diamondial.org/limage/70.190.JPEG" alt="Picasso's Melancholy
Woman" />
<cite>Melancholy Woman</cite>
</li>
<br />
Oil on canvas, 1902; Bequest of Robert H. Tannahill, 70.190
</ul>
</body>
</html>
```

You may notice that we avoided using too much stylistic markup to format the page. Using stylistic markup is a big no-no in the XML world. Achieve style using style sheets (see Chapter 6, "Adding Style," or Chapter 7, "Adding Style with XSL"). It's almost impossible to omit all stylistic markup when using XHTML 1.0 as your markup language of choice because most XHTML elements are style oriented. However, keep in mind that many style elements will be omitted from the next generation of XHTML. Therefore, try to stick to the structural and semantic elements if you can—more on this in Chapter 6.

Working with HTML Tidy

We know the answer to all of your prayers is automation. If you have only a couple HTML pages to your credit, manually converting your documents is a snap. However, if you're a veteran of the Web-design field and went public with your pages years ago (or even months ago), you probably have many HTML documents to convert. There must be an easier way to tackle converting them, and there is.

If you're familiar with the W3C site, you might be familiar with Dave Raggett. He's the Tom Cruise of Web design—well, one of them anyway—and has been hard at work to make your life easier. For anyone searching for a good tool to convert those HTML pages to XHTML, Dave Raggett has created one with your interests in mind: HTML Tidy. You can choose from three different interfaces from which to work with HTML Tidy: a DOS-based interface, a Web-based one, and graphical user interface (GUI) form. It gets better—all three versions of this tool are free!

HTML Tidy from a Command Line

In the past, Windows users could use only HTML Tidy from the command line (and you still can). Therefore, if you used computers in the 80s and early 90s, you're familiar with the command line, and you really want to use it, read the this section.

Before Windows and the ease of drag and drop, PCs were run from a DOS prompt. From the DOS prompt, a PC user could do most of what you do today in a Windows environment. Except the user had to use commands to tell the computer what to do. Some diehards still work from the DOS prompt, but we make the assumption that most of you out there have adopted the wonderful world of Windows. So, to start with, download this version of HTML Tidy from `www.w3.org/People/Raggett/tidy/`.

Next, you need to see the DOS prompt in action. Go to Start, Programs, MS-DOS Prompt. A window with a black background opens with a prompt at the folder `C:\Windows`.

HTML Tidy for the MAC

If you use a Mac, the GUI version was available long ago. Mac users can find out more at `www.geocities.com/SiliconValley/1057/tidy.html`.

Now you can experiment with HTML Tidy. To get started, you do the same as demonstrated previously. Go to Start, Programs, MS-DOS Prompt. Find the folder that contains HTML Tidy. For this example, the folder is `C:\tidy`. To locate this folder, we typed the following at the command line:

```
cd ..\tidy
```

Use the `cd` command to change the folder and then use `..\` to navigate back to the root folder. The last code is `tidy` and it opens the tidy folder. Be sure to include a space between `cd` and `..\` but no space between `..\` and `tidy`.

After you do that, you're ready to use HTML Tidy. If you decided that you're brave enough to tackle HTML Tidy from the DOS prompt, the next step is to convert the nasty HTML document shown in Example 4.3 to clean XHTML.

Example 4.3 **This Document Needs to Be Converted to Clean XHTML, Which You Do with HTML Tidy**

```
<HTML>
<Title>Sloppy Code at Play</Title>
<H1>HTML Document with Mistakes<h1>
<P>This document does not adhere to XHTML rules.
<p>Take a second to see all the mistakes.
<ul>
<li>All element names are not lowercase.
<li>Many elements do not contain the required closing tags.
<li>The XHTML namespace is not used.
<li>The document is missing some required elements.
<li>The document lacks the required DOCTYPE declaration.
</UL>
</html>
```

On our machine, the previous HTML file is located in the following directory:

```
c:\XHTML\tidy.html
```

To convert this document on your computer, you need to do a few things first:

- Create a folder on your hard drive called XHTML. (The name is not important, but if you decided to choose your own filename, be sure to change the name in all the right places.)
- Find Example 4.3 on the CD in the Chapter 4 examples folder and save it in the XHTML folder on your hard drive with the name `tidy.html`.

Now, back to the example. To clean up this file with HTML Tidy, enter the following code at the command prompt:

```
tidy -asxml -m  c:\XHTML\tidy.htm
```

The spaces are important and so is every consonant. What does all that code mean? First, `tidy` identifies the program to use. `-asxml` instructs Tidy to convert the HTML document to XHTML. `-m` tells the program to modify the document in its current location, and `c:\XHTML\tidy.htm` is the location of the messy document to be converted.

After you enter this line and press Enter, the next time your open your document (tidy.html), you'll find an XHTML document instead. The resulting code is shown in Example 4.4.

Example 4.4 **The Result of Running the Code in Example 4.3 Through HTML Tidy**

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta name="generator" content="HTML Tidy, see www.w3.org" />
<title>Sloppy Code at Play</title>
</head>
<body>
<h1> HTML Document with Mistakes</h1>
<p>This document does not adhere to XHTML rules.</p>
<p>Take a second to see all the mistakes.</p>
<ul>
  <li>All element names are not lowercase.</li>
  <li>Many elements do not contain the required closing tags.</li>
  <li>The XHTML namespace is not used.</li>
  <li>The document is missing some required elements.</li>
  <li>The document lacks the required DOCTYPE declaration.</li>
</ul>

</body>
</html>
```

If you want to, save the XHTML document in a separate location to keep the HTML document intact.

You can also personalize the experience and affect only one conversion rule (for example, convert all element names to lowercase), leaving the other XHTML rules alone. Both of these cases, as well as many more, have been taken into consideration. All it takes to perform these tasks is a different code word or two. For a complete listing of commands to work with, see `www.w3.org/People/Raggett/tidy/#help`.

HTML Tidy Online

If you're like many people, you might want to avoid the whole DOS experience. For those of us who are addicted to the Web, WebReview has published a Web-friendly version of HTML Tidy that has an easy-to-use interface. All you have to do is type the URL for the HTML Web page that you want converted and click a button. The new XHTML page is displayed for you online. Keep in mind that when the new page is displayed, you have to select File, Save to save it to your computer. To view the code before saving it, you can always view the source code (View, Source). View the HTML Tidy, Web-friendly front end at `www.webreview.com/1999/07/16/feature/xhtml.cgi`.

TidyGUI

For those of us who don't enjoy a DOS-dominated interface and feel more comfortable doing the conversion on our own machines, André Blavier has created a Windows interface for HTML Tidy. TidyGUI provides all the options of HTML Tidy, with all the ease of a Windows interface that you can customize. For a snapshot of this new Tidy facelift, see Figure 4.2. Enter the desired file to be converted into the field titled, Source File, and then select the Tidy button. If you want to customize the conversion process, select the Configuration button and make any changes you like. It is that easy!

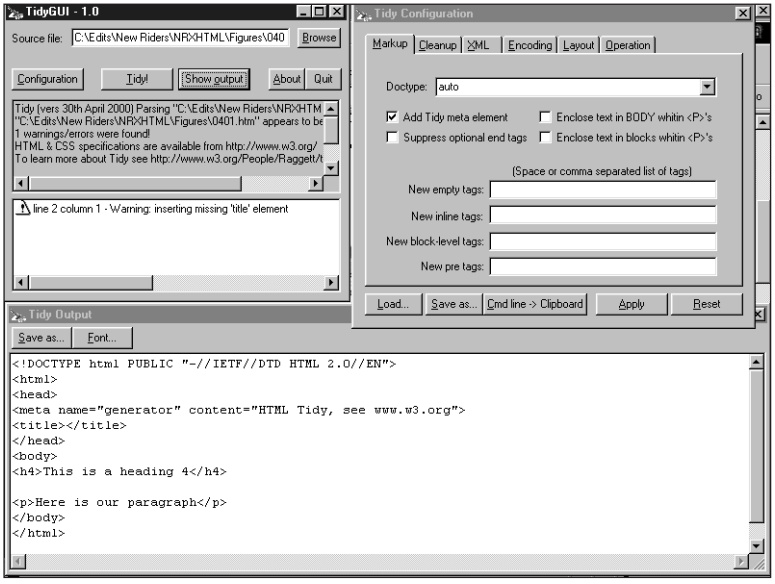

Figure 4.2 The three windows of the TidyGUI utility.

For More Information

By no means does the buck stop here. The W3C is already working the next generation of XHTML and there are some anticipated changes on the horizon (see Chapter 13, "Where the Future Leads, XHTML Follows," for a complete breakdown of the next version of XHTML). In addition, we expect that additional tools will be introduced as the demand for HTML to XHTML conversion increases. To keep up to date on both manual and automated conversion techniques, keep an eye on the following Web sites:

- `http://wdvl.com/Authoring/Languages/XML/XHTML/`
- `www.hwg.org/opcenter/gutenberg/markupXHTML.html`
- `www.xhtml.org/`
- `www.w3.org/MarkUp/Activity.html`
- `www.xmlsoftware.com`