CHAPTER 2

Introducing HTML5

The HTML5 specification not only embraces the past, by supporting traditional HTML- and XHTML-style syntax, but also adds a wide range of new features. Although HTML5 moves forward from HTML 4, it also is somewhat of a retreat and an admission that trying to get every Web developer on the Internet to write their markup properly is a futile effort, particularly because few Web developers are actually formally trained in the technology. HTML5 tries to bring order to chaos by codifying common practices, embracing what is already implemented in browsers, and documenting how these user agents (browsers or other programs that consume Web pages) should deal with our imperfect markup.

HTML5's goals are grand. The specification is sprawling and often misunderstood. Given the confusion, the goals of this chapter are not only to summarize what is new about HTML5 and provide a roadmap to the element reference that follows, but to also expose some of the myths and misconceptions about this exciting new approach to markup.

NOTE *Perhaps just to be new, HTML5 omits the space found commonly between (X)HTML and its version number, as in HTML 4 or XHTML 1. We follow this style generally in the book, but note even the specification has not been stringent on this point.*

Hello HTML5

The syntax of HTML5 should be mostly familiar. As shown in the previous chapter, a simple HTML5 document looks like this:

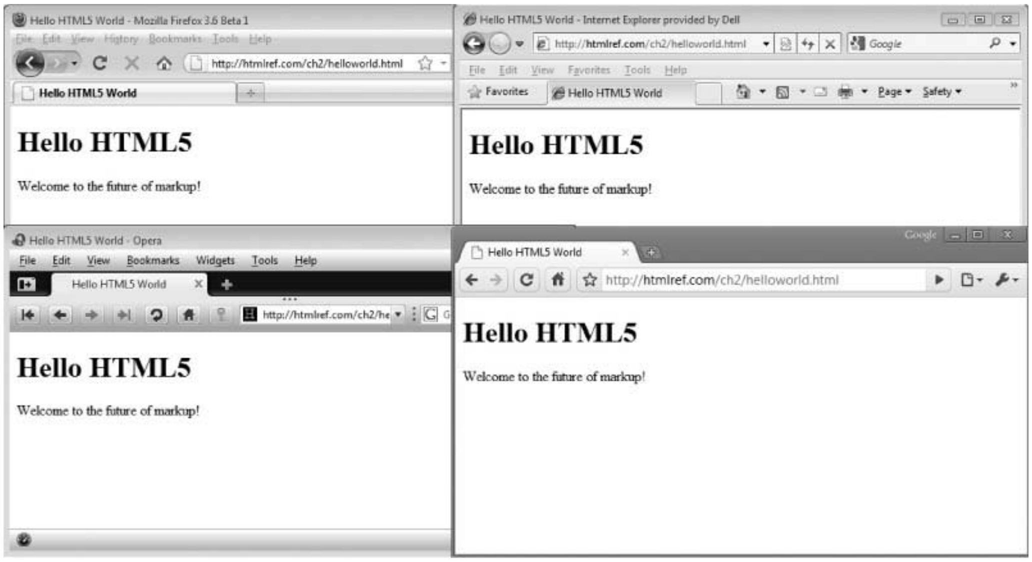
```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Hello HTML5 World</title>
</head>
<body>
<h1>Hello HTML5</h1>
<p>Welcome to the future of markup!</p>
</body>
</html>
```

ONLINE *http://htmlref.com/ch2/helloworld.html*

For all practical purposes, all that is different from standard HTML in this example is the **<!DOCTYPE>** statement. Given such minimal changes, of course, basic HTML5 will immediately render correctly in browsers, as demonstrated in Figure 2-1.

As indicated by its atypical **<!DOCTYPE>** statement, HTML5 is not defined as an SGML or XML application. Because of the non-SGML/XML basis for HTML, there is no concept of validation in HTML5; instead, an HTML5 document is checked for conformance to the specification, which provides the same practical value as validation. So the lack of a formal DTD is somewhat moot. As an example, consider the following flawed markup:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Hello Malformed HTML5 World</title>
</head>
<body>
<!-- note bad close tag below -->
<h1>Hello Malformed HTML5<h1>
<!-- unknown tag found here -->
<p>Welcome to the <danger>future</danger> of markup!</p>
<!-- missing </body>  -->
</html>
```


Hello HTML5 World - Mozilla Firefox 3.6 Beta 1
http://htmlref.com/ch2/helloworld.html
Hello HTML5 World - Internet Explorer provided by Dell
Hello HTML5 World - Opera
Hello HTML5 World

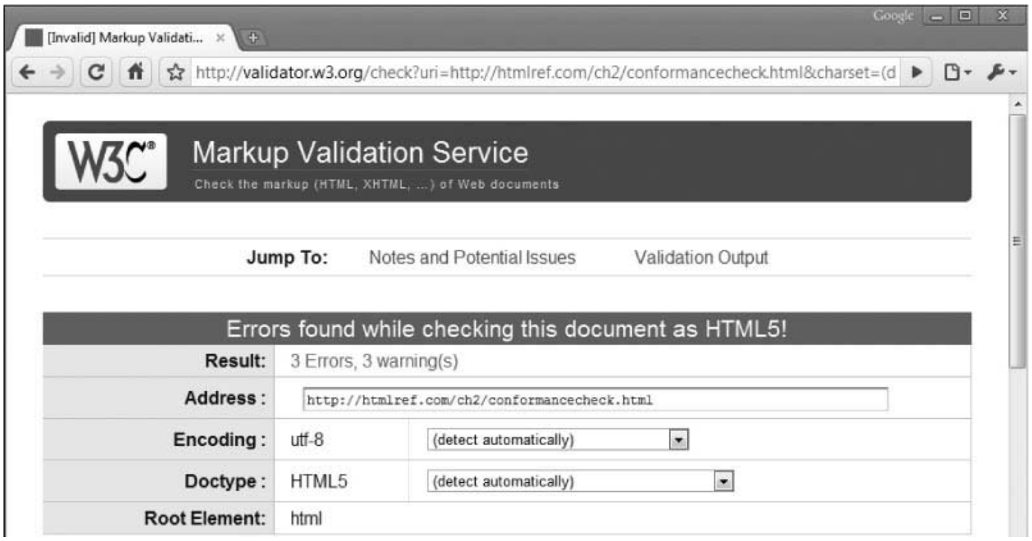
Hello HTML5

Welcome to the future of markup!

FIGURE 2-1 HTML5 is alive.

ONLINE *http://htmlref.com/ch2/conformancecheck.html*

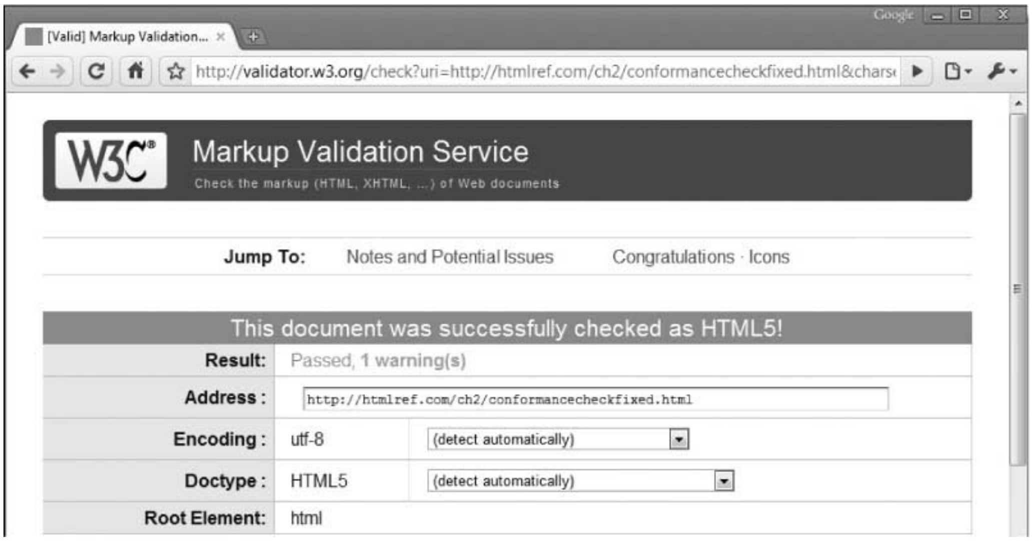
When checked with an HTML5 conformance checker, such as the W3C Markup Validation Service used in this chapter (available at http://validator.w3.org), you see the expected result:


[Invalid] Markup Validati...
http://validator.w3.org/check?uri=http://htmlref.com/ch2/conformancecheck.html&charset=(d
Markup Validation Service
Check the markup (HTML, XHTML, ...) of Web documents
Jump To: Notes and Potential Issues Validation Output

Errors found while checking this document as HTML5!

Result:	3 Errors, 3 warning(s)	
Address :	http://htmlref.com/ch2/conformancecheck.html	
Encoding :	utf-8	(detect automatically)
Doctype :	HTML5	(detect automatically)
Root Element:	html	

Later, with errors corrected, a clean check is possible:

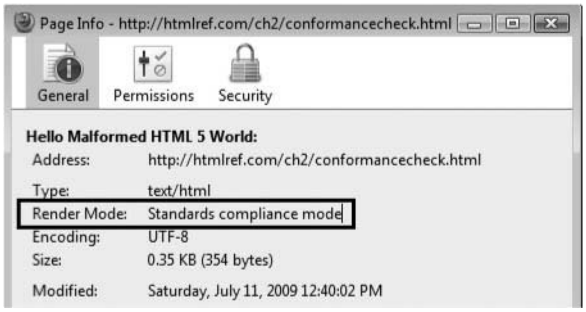

[Valid] Markup Validation...
http://validator.w3.org/check?uri=http://htmlref.com/ch2/conformancecheckfixed.html&chars
Markup Validation Service
Check the markup (HTML, XHTML, ...) of Web documents
Jump To: Notes and Potential Issues Congratulations · Icons

This document was successfully checked as HTML5!

Result:	Passed, 1 warning(s)	
Address :	http://htmlref.com/ch2/conformancecheckfixed.html	
Encoding :	utf-8	(detect automatically)
Doctype :	HTML5	(detect automatically)
Root Element:	html	

NOTE *Given the currently fluid nature of HTML5, developers are warned that, at least for now, HTML5 conformance may be a bit of a moving target.*

If you are wondering what mode the browser enters into because of the divergent **`<!DOCTYPE>`** used by HTML5, apparently it is the more standards-oriented mode:



Hello Malformed HTML 5 World:
Address: http://htmlref.com/ch2/conformancecheck.html
Type: text/html
Render Mode: Standards compliance mode
Encoding: UTF-8
Size: 0.35 KB (354 bytes)
Modified: Saturday, July 11, 2009 12:40:02 PM

Employing the more standards-oriented parsing mode might seem appropriate, but it is somewhat odd given the point of the next section.

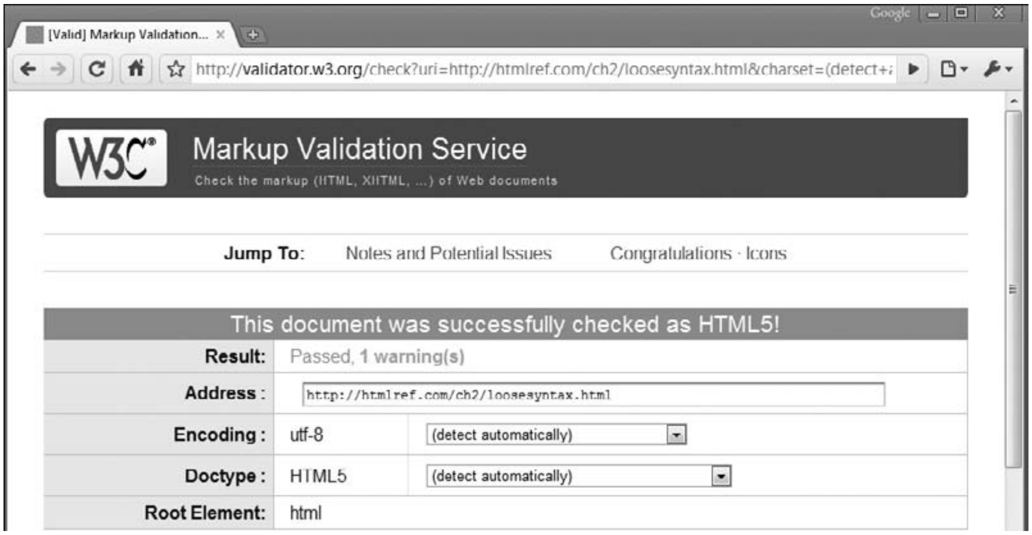
Loose Syntax Returns

An interesting aspect of HTML5 is the degree of syntax variability that it allows. Unlike its stricter markup cousin, XHTML, the traditional looseness of HTML is allowed. To demonstrate, in the following example, quotes are not always employed, major elements like **`html`**, **`head`**, and **`body`** are simply not included, the inference of close of tags like **`</p>`** and **`</li>`** is allowed, case is used variably, and even XML-style self-identifying close syntax is used at will:

```
<!DOCTYPE html>
<!-- I have no html, head, or body as they are actually optional -->
<meta http-equiv=Content-Type content="text/html; charset=utf-8">
<title>HTML5 Tag Soup Test</title>
<h1 title="more sloppy markup ahead!">HTML5</H1>
<p id=p1>Back to the future of loose markup!?
<p>Yes it looks that way
<ul>
  <li>optional elements
  <LI>case is no problem
  <li id=noquotes>quotes optional in many cases
  <li>inferred close tags
</UL>
<p>Oh my
<br>
<br />
<p>Intermixing markup styles!
<!-- ok that's enough let's stop now -->
```

ONLINE *http://htmlref.com/ch2/loosesyntax.html*

This example, at least currently, conforms to the HTML5 specification:


[Valid] Markup Validation...
http://validator.w3.org/check?uri=http://htmlref.com/ch2/loosesyntax.html&charset=(detect+
W3C
Markup Validation Service
Check the markup (HTML, XHTML, ...) of Web documents
Jump To: Notes and Potential Issues Congratulations · Icons

This document was successfully checked as HTML5!

Result:	Passed, 1 warning(s)	
Address :	http://htmlref.com/ch2/loosesyntax.html	
Encoding :	utf-8	(detect automatically)
Doctype :	HTML5	(detect automatically)
Root Element:	html	

Do not interpret the previous example to mean that HTML5 allows a markup free-for-all. Understand that these "mistakes" are actually allowed under traditional HTML and thus are allowed under HTML5. To ensure that you conform to the HTML5 specification, you should be concerned primarily about the following:

- Make sure to nest elements, not cross them; so

  ```
  <b><i>is in error as tags cross</b></i>
  ```

  whereas

  ```
  <b><i>is not since tags nest</i></b>.
  ```

- Quote attribute values when they are not ordinal values, particularly if they contain special characters, particularly spaces; so

  ```
  <p id=p1>Fine with no quotes</p>
  ```

  because it is a simple attribute value, whereas

  ```
  <p title=trouble here with no quotes>Not ok without quotes</p>
  ```

  is clearly messed up.

- Understand and follow the content model. Just because one browser may let you use a list item anywhere you like,

  ```
  <li>I should be in a list!</li>
  ```

  it isn't correct. Elements must respect their content model, so the example should read instead as

  ```
  <ul><li>All is well I am in a list!</li></ul>
  ```

  because it follows HTML5's content model.

- Do not use invented tags unless they are included via some other markup language:

  ```
  <p>I <danger>shouldn't</danger> conform unless I am defined in
  another specification and use a name space</p>
  ```

- Encode special characters, particularly those used in tags (`< >`), either as an entity of a named form, such as `&lt;`, or as a numeric value, such as `&#60;`. Appendix A covers this topic in some depth.

This brief list of what you should do might seem familiar; it is pretty much the list of recommendations for correct markup from the previous chapter returned to the traditional markup styles of HTML. What this means is that if you have been writing markup correctly in the past, HTML5 isn't going to present much of a change. In fact, in many cases, just by changing a valid document's doctype to the new simple HTML5 **`<!DOCTYPE html>`**, the result should be an HTML5–conforming document.

XHTML5

For those with a heavy investment in a strict XHTML syntax worldview, HTML5 might seem like a slap in the face. However, such a reaction is a bit premature; HTML5 neither makes the clean markup you write non-conforming nor suggests that you shouldn't author markup this way. If you want to pursue an "XMLish" approach to your document, HTML5 allows it. Consider, for example, a strict XHTML example that is now HTML5:

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Hello XHTML5 World</title>
<!-- Simple hello world in XHTML5 -->
</head>
<body>
<h1>Welcome to the World of XHTML5</h1>
<hr />
<p>XHTML5 <em>really</em> isn't so hard either!</p>
<p>HTML5 likes XML syntax too.</p>
<p>Make sure to serve it with the correct MIME type!</p>
<!-- IE users you will get a render error.
     Please read on to learn why. -->
</body>
</html>
```

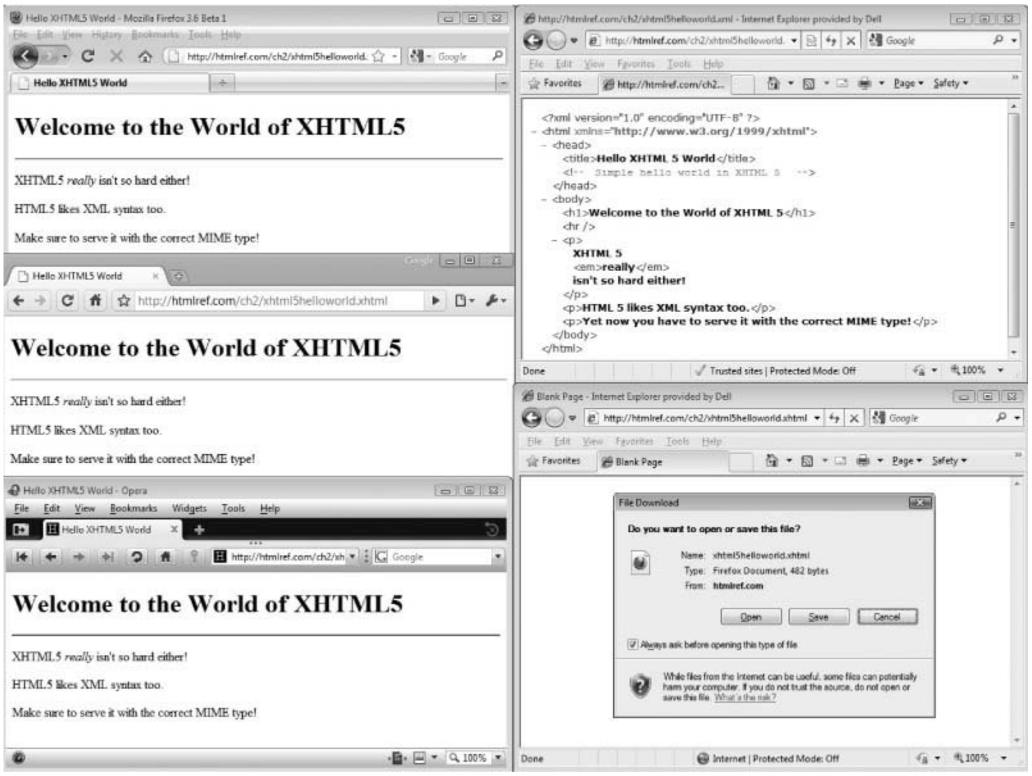
ONLINE *http://htmlref.com/ch2/xhtml5helloworld.xhtml*

NOTE *When using XML syntax with HTML5 according to HTML5 specification, this should be termed XHTML5.*

Notice that the previous example uses an `.xhtml` file extension. XHTML5 usage clearly indicates that an HTML5 document written to XML syntax must be served with the MIME type `application/xhtml+xml` or `application/xml`. The previous example was served with the former MIME type. You can find the same example served with latter XML MIME type at http://htmlref.com/ch2/xhtml5helloworld.xml.

Unfortunately, although HTML5 supports XML, the real value of XHTML—the true strictness of XML—has not been realized, at least so far, because of a lack of browser support. As of this edition's writing, Internet Explorer browsers (up to version 8) will not render XHTML served with the appropriate `application/xhtml+xml` MIME type and will take the raw XML form and render it as a parse tree. Other browsers, fortunately, don't do this (see Figure 2-2), which is little solace given Internet Explorer's widespread usage.

You can write XMLish markup and serve it as `text/html` but it won't provide the benefit of strict syntax conformance. In short, HTML5 certainly allows you to try to continue applying the intent of XHTML in the hopes that someday it becomes viable.


Hello XHTML5 World - Mozilla Firefox 3.6 Beta 1
http://htmlref.com/ch2/xhtml5helloworld.
Hello XHTML5 World
http://htmlref.com/ch2/xhtml5helloworld.xhtml
Hello XHTML5 World - Opera
http://htmlref.com/ch2/xhtml5helloworld.xml - Internet Explorer provided by Dell
Blank Page - Internet Explorer provided by Dell
http://htmlref.com/ch2/xhtml5helloworld.xhtml
Done
Trusted sites | Protected Mode: Off
Internet | Protected Mode: Off
100%

Welcome to the World of XHTML5

XHTML5 *really* isn't so hard either!

HTML5 likes XML syntax too.

Make sure to serve it with the correct MIME type!

```
<?xml version="1.0" encoding="UTF-8" ?>
- <html xmlns="http://www.w3.org/1999/xhtml">
  - <head>
      <title>Hello XHTML 5 World</title>
      <!--  Simple hello world in XHTML 5  -->
    </head>
  - <body>
      <h1>Welcome to the World of XHTML 5</h1>
      <hr />
    - <p>
        XHTML 5
        <em>really</em>
        isn't so hard either!
      </p>
      <p>HTML 5 likes XML syntax too.</p>
      <p>Yet now you have to serve it with the correct MIME type!</p>
    </body>
  </html>
```

File Download

Do you want to open or save this file?

Name: xhtml5helloworld.xhtml
Type: Firefox Document, 482 bytes
From: htmlref.com

Open | Save | Cancel

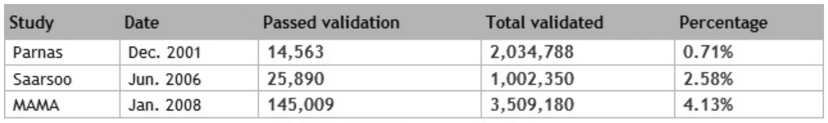
[✓] Always ask before opening this type of file

While files from the Internet can be useful, some files can potentially harm your computer. If you do not trust the source, do not open or save this file. What's the risk?

FIGURE 2-2 XHTML5 works, but Internet Explorer support lags.

HTML5: Embracing the Reality of Web Markup

Given the looseness HTML5 supports and its de-emphasis of the XML approach to markup, you might assume that HTML5 is a retreat from doing things in the right way and an acceptance of "tag soup" as legitimate markup. The harsh reality is that, indeed, valid markup is more the exception than the rule online. Numerous surveys have shown that in the grand scheme of things, few Web sites validate. For example, in a study of the Alexa Global Top 500 in January 2008, only 6.57 percent of the sites surveyed validated.[1] When sample sizes are increased and we begin to look at sites that are not as professional, things actually get worse. Some validation results from Opera's larger MAMA (Metadata Analysis and Mining Application) study are shown here[2]:



Study	Date	Passed validation	Total validated	Percentage
Parnas	Dec. 2001	14,563	2,034,788	0.71%
Saarsoo	Jun. 2006	25,890	1,002,350	2.58%
MAMA	Jan. 2008	145,009	3,509,180	4.13%

Fig 5-1: Validation pass rate studies

Interestingly, Google has even larger studies, and while they don't focus specifically on validation, what they indicate on tag usage indicates clearly that no matter the sample size, clean markup is more the exception than the rule.

Yet despite the markup madness, the Web continues to work. In fact, some might say the permissive nature of browsers that parse junk HTML actually helps the Web grow because it lowers the barrier to entry for new Web page authors. Certainly a shaky foundation to build upon, but the stark reality is that we must deal with malformed markup. To this end, HTML5 makes one very major contribution: it defines what to do in the presence of markup syntax problems.

The permissive nature of browsers is required for browsers to fix markup mistakes. HTML5 directly acknowledges this situation and aims to define how browsers should parse both well-formed and malformed markup, as indicated by this brief excerpt from the specification:

> This specification defines the parsing rules for HTML documents, whether they are syntactically correct or not. Certain points in the parsing algorithm are said to be *parse errors*. The error handling for parse errors is well-defined: user agents must either act as described below when encountering such problems, or must abort processing at the first error that they encounter for which they do not wish to apply the rules described below.

While a complete discussion of the implementation of an HTML5–compliant browser parser is of little interest to Web document authors, browser implementers now have a common specification to consult to determine what to do when tags are not nested, simply left open, or mangled in a variety of ways. This is the part of the HTML5 specification that

[1] Brian Wilson, "MAMA W3C Validator Research," subsection "Interesting Views of Validation Rates, part 2: Alexa Global Top 500," Dev.Opera, October 15, 2008, http://dev.opera.com/articles/view/mama-w3c-validator-research-2/?page=2#alexalist.

[2] Ibid., subsection "How Many Pages Validated?" http://dev.opera.com/articles/view/mama-w3c-validator-research-2/#validated.

will likely produce the most good, because obtaining consensus among browser vendors to handle markup problems in a consistent manner is a more likely path to an improved Web than defining some strict syntax and then attempting to educate document authors around the world en masse to write good markup.

HTML5's aim to bring order to the chaos of sloppy markup is but one of the grand aims of the specification. It also aims to replace traditional HTML, XHTML, and DOM specifications, and to do so in a backward-compatible fashion. In its attempt to do this, the specification is sprawling, addressing not just what elements exist but how they are used and scripted. HTML5 embraces the fact that the Web not only is composed of documents but also supports applications, thus markup must acknowledge and facilitate the building of such applications. More of the philosophy of HTML5 will be discussed later in the chapter when addressing some strong opinions, myths, and misconceptions surrounding the specification; for now, take a look at what markup features HTML5 actually changes.

Presentational Markup Removed and Redefined

HTML5 removes a number of elements and attributes. Many of the elements are removed because they are more presentational than semantic. Table 2-1 presents the elements currently scheduled for removal from HTML5.

NOTE *Although these elements are removed from the specification and should be avoided in favor of CSS, they likely will continue to be supported by browsers for some time to come. The specification even acknowledges this fact.*

Looking at Table 2-1, you might notice that some elements that apparently should be eliminated somehow live on. For example, `<small>` continues to be allowed, but `<big>` is obsolete. The idea here is to preserve elements but shift meaning. For example, `<small>` is no longer intended to correspond to text that is just reduced in size, similar to `<font size="-1">` or `<span style="font-size: smaller;">`, but instead is intended to represent the use of small text, such as appears in fine print or legal information. If you think this decision seems a bit preposterous, join the crowd. Some of the other changes to element meaning seem even a bit more preposterous, such as the claim that a `<b>` tag now represents inline text that is stylistically offset from standard text, typically using a different

Removed HTML Element	CSS Equivalent
`<basefont>`	`body {font-family: *family*; font-size: *size*;}`
`<big>`	`font-size: larger`
`<center>`	`text-align: center` or `margin: auto` depending on context
`<font>`	`font-family`, `font-size`, or `font`
`<s>, <strike>`	`text-decoration: strike`
`<tt>`	`font-family: monospace`
`<u>`	`text-decoration: underline`

TABLE 2-1 HTML 4 Elements Removed from HTML5

type treatment. So apparently `<b>` tags are not necessarily bold, but rather convey some sense that the text is "different" (which likely means bold). Unlikely to be thought of in such a manner by mere markup mortals, we simply say `<b>` tags live on, as do a number of other presentational elements. Table 2-2 presents the meaning-changed elements that stay put in HTML5 and their new meaning.

The meaning of some of these items might not be immediately clear, but don't worry about that now, because each will be demonstrated later in the chapter and a full reference presented in Chapter 3.

Like the strict variants of (X)HTML, HTML5 also removes numerous presentation-focused attributes. Table 2-3 summarizes these values and presents CSS alternatives.

Out with the Old Elements

A few elements are removed from the HTML5 specification simply because they are archaic, misunderstood, have usability concerns, or have a function that is equivalent to the function of other elements. Table 2-4 summarizes some of the elements that have been removed from the HTML5 specification.

NOTE *While frames are mostly removed from HTML5, inline frames live on. See the section "The Uncertain Future of Frames," later in the chapter, for more information.*

Table 2-4 is not a complete list of non-conforming elements, just the ones that are supported by recent HTML 4 and XHTML 1.x specifications. Discussing the fact that ancient tags like `<listing>` and `<plaintext>` continue not to be supported or that all the presentational tags

HTML Element	New Meaning in HTML5
`<b>`	Represents an inline run of text that is different stylistically from normal text, typically by being bold, but conveys no other meaning of importance.
`<dd>`	Used with HTML5's new **`details`** and **`figure`** elements to define the contained text. Was also used with a **`dialog`** element which was later removed from the HTML5 specification.
`<dt>`	Used with HTML5's new **`details`** and **`figure`** element to summarize the details. Was also used with a **`dialog`** element which was later removed from the HTML5 specification.
`<hr>`	Represents a thematic break rather than a horizontal rule, though that is the likely representation.
`<i>`	Represents an inline run of text in an alternative voice or tone that is supposed to be different from standard text but that is generally presented in italic type.
`<menu>`	Redefined to represent user interface menus, including context menus.
`<small>`	Represents small print, as in comments or legal fine print.
`<strong>`	Represents importance rather than strong emphasis.

TABLE 2-2 HTML 4 Elements Redefined in HTML5

Attribute Removed	Elements Effected	CSS Equivalent
align	**caption**, **col**, **colgroup**, **div**, **iframe**, **h1**, **h2**, **h3**, **h4**, **h5**, **h6**, **hr**, **img**, **input**, **legend**, **object**, **p**, **table**, **tbody**, **td**, **tfoot**, **th**, **thead**, **tr**	text-align or in some block element cases float
alink	**body**	body a:active {color: *color-value*;}
background	**body**	background-image or background
bgcolor	**body**, **table**, **td**, **th**, **tr**	background-color
border	**img**, **object**, **table**	border-width and/or border
cellpadding	**table**	padding
cellspacing	**table**	margin
char	**col**, **colgroup**, **table**, **tbody**, **td**, **tfoot**, **th**, **thead**, **tr**	N/A
charoff	**col**, **colgroup**, **table**, **tbody**, **td**, **tfoot**, **th**, **thead**, **tr**	N/A
clear	**br**	clear
compact	**dl**, **menu**, **ol**, **ul**	margin properties
frame	**table**	border properties
frameborder	**iframe**	border properties
height	**td**, **th**	height
hspace	**img**, **object**	margin properties
link	**body**	body a:link {color: *color-value*;}
marginheight	**iframe**	margin properties
marginwidth	**iframe**	margin properties
noshade	**hr**	border-style or border
nowrap	**td**, **th**	overflow
rules	**table**	border properties
scrolling	**iframe**	overflow
size	**hr**	width
text	**body**	body {color: *color-value*;}
type	**li**, **ol**, **ul**	list-style-type and list-style
valign	**col**, **colgroup**, **tbody**, **td**, **tfoot**, **th**, **thead**	vertical-align
vlink	**body**	body a:visited {color: *color-value*;}
width	**col**, **colgroup**, **hr**, **pre**, **table**, **td**, **th**	width

TABLE 2-3 HTML 4 Attributes Removed in HTML5

Removed Element	Reasoning	Alternatives
`acronym`	Misunderstood by many Web developers.	Use the **`abbr`** element.
`applet`	Obsolete syntax for Java applets.	Use the **`object`** element.
`dir`	Rarely used, and provides similar functionality to unordered lists.	Use the **`ul`** element.
`frame`	Usability concerns.	Use fixed-position elements with CSS and/or **`object`** elements with sourced documents.
`frameset`	Usability concerns.	Use fixed-position elements with CSS and/or **`object`** elements with sourced documents.
`isindex`	Archaic and can be simulated with typical form elements.	Use the **`input`** element to create text field and button and back up with appropriate server-side script.
`noframes`	Since frames are no longer supported, this contingency element is no longer required.	N/A

TABLE 2-4 Elements Removed by HTML5

like **`<font>`** and proprietary tags like **`<spacer>`**, **`<marquee>`**, and **`<blink>`** should be off limits is somewhat redundant and does not build on the specifications. However, the reference in Chapter 3 covers compliance points completely, so when in doubt check the appropriate element's entry.

In with the New Elements

For most Web page authors, the inclusion of new elements is the most interesting aspect of HTML5. Some of these elements are not yet supported, but already many browsers are implementing a few of the more interesting ones, such as **`audio`** and **`video`**, and others can easily be simulated even if they are not directly understood yet, as you will see later in the chapter. Table 2-5 summarizes the elements added by HTML5 at the time of this edition's writing, and the sections that follow illustrate their use. Again, Chapter 3 provides a complete element syntax discussion.

Sample of New Attributes for HTML5

One quite important aspect of HTML5 is the introduction of new attributes. There are quite a few attributes that are global and thus found on all elements. Table 2-6 provides a brief overview of these attributes. We'll take a look at many of these in upcoming sections and a complete reference for all is found in the next chapter.

The element reference in Chapter 3 provides the full syntax for the various HTML5 attributes that may have been added to specific elements. Some of them, such as **`reversed`** for use on ordered lists (**`<ol>`**), are a long time in coming, while others simply add polish, or address details that few page authors may notice.

New Element	Description
`article`	Encloses a subset of a document that forms an independent part of a document, such as a blog post, article, or self-continued unit of information.
`aside`	Encloses content that is tangentially related to the other content in an enclosing element such as `section`.
`audio`	Specifies sound to be used in a Web page.
`canvas`	Defines a region to be used for bitmap drawing using JavaScript.
`command`	Located within a `menu` element, defines a command that a user may invoke.
`datalist`	Indicates the data items that may be used as quick choices in an `input` element of `type="list"`.
`details`	Defines additional content that can be shown on demand.
`figure`	Defines a group of content that should be used as a figure and may be labeled by a `legend` element.
`footer`	Represents the footer of a `section` or the document and likely contains supplementary information about the related content.
`header`	Represents the header of a `section` or the document and contains a label or other heading information for the related content.
`hgroup`	Groups heading elements (`h1`–`h6`) for sectioning or subheading purposes.
`mark`	Indicates marked text and should be used in a similar fashion to show how a highlighter is used on printed text.
`meter`	Represents a scalar measurement in a known range similar to what may be represented by a gauge.
`nav`	Encloses a group of links to serve as document or site navigation.
`output`	Defines a region that will be used to hold output from some calculation or form activity.
`progress`	Indicates the progress of a task toward completion, such as displayed in a progress meter or loading bar.
`rp`	Defines parentheses around `ruby` text defined by an `rt` element.
`rt`	Defines text used as annotations or pronunciation guides. This element will be enclosed within a `ruby` element.
`ruby`	This is the primary element and may include `rt` and `rp` elements. A `ruby` element serves as a reading or pronunciation guide. It is commonly used in Asian languages, such as in Japanese to present information about Kanji characters.
`section`	Defines a generic section of a document and may contain its own `header` and `footer`.
`source`	Represents media resources for use by `audio` and `video` elements.
`time`	Encloses content that represents a date and/or time.
`video`	Includes a video (and potentially associated controls) in a Web page.

TABLE 2-5 Elements Added by HTML5

New Attribute	Description
accesskey	Defines the accelerator key to be used for keyboard access to an element.
contenteditable	When set to `true`, the browser should allow the user to edit the content of the element. Does not specify how the changed content is saved.
contextmenu	Defines the DOM **id** of the **menu** element to serve as a context menu for the element the attribute is defined on.
data-*X*	Specifies user-defined metadata that may be put on tags without concern of collision with current or future attributes. Use of this type of attribute avoids the common method of creating custom attributes or overloading the **class** attribute.
draggable	When specified, should allow the element and its content to be dragged.
hidden	Under HTML5, all elements may have **hidden** attribute which when placed indicates the element is not relevant and should not be rendered. This attribute is similar to the idea of using the CSS `display` property set to a value of `none`.
itemid	Sets a global identifier for a microdata item. This is an optional attribute, but if it is used, it must be placed in an element that sets both the **itemscope** and **itemtype** attributes. The value must be in the form of a URL.
itemprop	Adds a name/value pair to an item of microdata. Any child of a tag with an **itemscope** attribute can have an **itemprop** attribute set in order to add a property to that item.
itemref	Specifies a list of space-separated elements to traverse in order to find additional name/value pairs for a microdata item. By default, an item only searches the children of the element that contains the **itemscope** attribute. However, sometimes it does not make sense to have a single parent item if the data is intermingled. In this case, the **itemref** attribute can be set to indicate additional elements to search. The attribute is optional, but if it is used, it must be placed in an element that sets the **itemscope** attribute.
itemscope	Sets an element as an item of microdata (see "Microdata" later in the chapter).
itemtype	Defines a global type for a microdata item. This is an optional attribute, but if it is used, it must be placed in an element that sets the **itemscope** attribute. The value must be in the form of a URL.
spellcheck	Enables the spell checking of an element. The need for this attribute globally may not be clear until you consider that all elements may be editable at page view time with the **contenteditable** attribute.
tabindex	Defines the element-traversal order when the keyboard is used for navigation.

TABLE 2-6 Key Attributes Added by HTML5

HTML5 Document Structure Changes

As you have seen, the HTML5 document structure seems pretty much the same as in HTML 4 save a slightly different `<!DOCTYPE>` statement. However, if you look closer, there are a few important differences in HTML5 that show the document structure has in fact been expanded quite a bit.

HTML5 documents may contain a **header** element, which is used to set the header section of a document and thus often contains the standard **h1** to **h6** heading elements:

```
<header>
<h1>Welcome to the Future World of HTML5.</h1>
<h2>Don't be scared it isn't that hard!</h2>
</header>
```

Similarly, a **footer** element is provided for document authors to define the footer content of a document, which often contains navigation, legal, and contact information:

```
<footer>
 <p>Content of this example is not under copyright</p>
</footer>
```

The actual content to be placed in a `<footer>` tag is, of course, up to you and may be enclosed in **div**, **p**, or other block elements, as illustrated by this simple example:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>HTML5 header and footer example</title>
</head>
<body>
<header>
<h1>Welcome to the Future World of HTML5.</h1>
<h2>Don't be scared it isn't that hard!</h2>
</header>
<p>Some body content here.</p>
<p>Some more body content here.</p>

<footer>
 <p>Content of this example is not under copyright.</p>
</footer>
</body>
</html>
```

ONLINE *http://htmlref.com/ch2/headerfooter.html*

The HTML5 structural element with the most possible uses is the **section** element. A particular `<section>` tag can be used to group arbitrary content together and may contain further `<section>` tags to create the idea of subsections. Traditionally, we are familiar with sections; just as this book is broken into chapters and various main and secondary sections, so too could a Web document be structured in this way. The example here illustrates the basic use of HTML5 sections:

```
<section>
<h1>Chapter 2</h1>
 <p>New HTML5 elements.</p>
 <section>
  <h2>HTML5's section Element</h2>
  <p>These elements are useful to create outlines.</p>
  <section>
    <h3>Nest Away!</h3>
    <p>Nest your sections but as you nest you might want to indent.</p>
  </section>
 </section>
 <p>Ok that's enough of that.</p>
</section>
```

ONLINE *http://htmlref.com/ch2/section.html*

It may not be obvious but a **section** element may contain **header** and **footer** elements of its own:

```
<section>
 <header>
   <h1>I am Section Heading</h1>
 </header>
 <h2>I am outside the section header I'm just a plain headline.</h2>
 <p>Some more section content might go here.</p>
 <footer>
   <p>Hi from the footer of this section.</p>
 </footer>
</section>
```

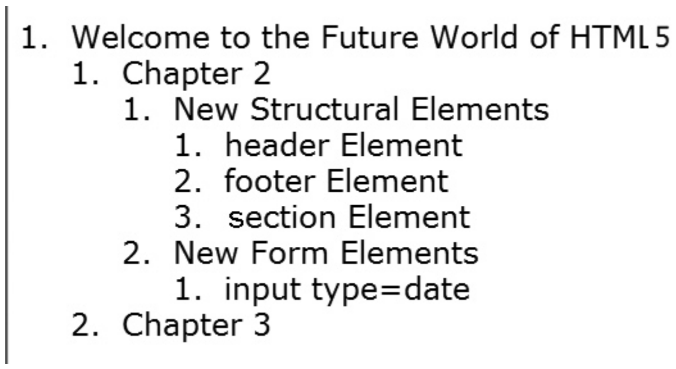
HTML5 uses headings and newly introduced elements like the **section** element for outlining purposes. For example, the expanded example here shows a number of sections with headers, footers, headlines, and content:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>HTML5 expanded section example</title>
</head>
<body>
<header>
 <h1>Welcome to the Future World of HTML5</h1>
 <h2>Don't be scared it isn't that hard!</h2>
</header>
```

```
<!-- assume chapter 1 before -->
<section id="chapter2">
 <header>
  <h1>Chapter 2</h1>
 </header>

 <p>Intro to chapter 2 here...</p>
 <section id="newStrucreElements">
  <header>
   <h2>New Structural Elements</h2>
  </header>
   <h3>header Element</h3>
   <p>Discussion of header element.</p>

   <h3>footer Element</h3>
   <p>Discussion of footer element.</p>

   <h3>section Element</h3>
   <p>Discussion of section element</p>
 </section>

 <section id="newFormElements">
   <header>
     <h2>New Form Elements</h2>
   </header>
   <h3>input type=date</h3>
   <p>Discussion here...</p>
   <footer>
     <p>These ideas are from WebForms specification.</p>
   </footer>
 </section>
</section>

<section id="chapter3">
 <header>
   <h2>Chapter 3</h2>
 </header>
 <p>Massive element reference...</p>
</section>
<footer>
 <p>Content of this example is not under copyright</p>
</footer>

</body>
</html>
```

ONLINE *http://htmlref.com/ch2/sectionoutline.html*

HTML5–compliant browsers should take this markup and define an outline based upon the use of headers, like so:



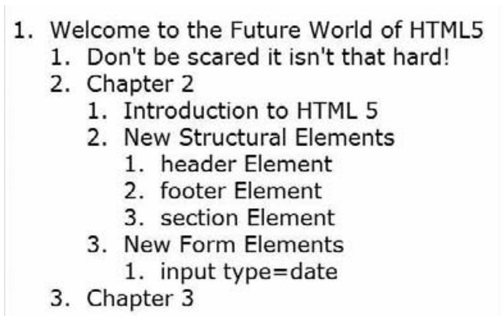
1. Welcome to the Future World of HTML5
    1. Chapter 2
        1. New Structural Elements
            1. header Element
            2. footer Element
            3. section Element
        2. New Form Elements
            1. input type=date
    2. Chapter 3

In theory, user agents could take the outlining semantics and derive meaning or even provide an alternative browser interface, although that is quite speculative at this point. It is clear, however, that if you introduce such outlining ideas, issues may arise. For example, the first header really was not two levels of sectioning but simply one with a subhead. To address this outlining, you would take this markup

```
<header>
 <h1>Welcome to the Future World of HTML5</h1>
 <h2>Don't be scared it isn't that hard!</h2>
</header>
```

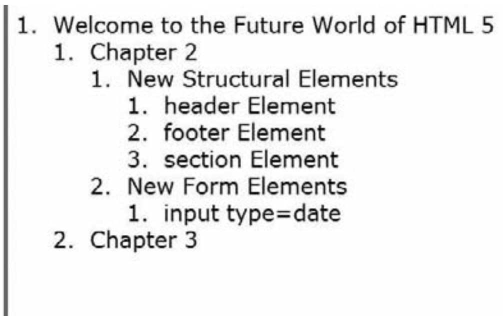
and then join the subhead to the headline with an **`hgroup`** element like so:

```
<header>
<hgroup>
 <h1>Welcome to the Future World of HTML5</h1>
 <h2>Don't be scared it isn't that hard!</h2>
</hgroup>
</header>
```



1. Welcome to the Future World of HTML5
    1. Don't be scared it isn't that hard!
    2. Chapter 2
        1. Introduction to HTML 5
        2. New Structural Elements
            1. header Element
            2. footer Element
            3. section Element
        3. New Form Elements
            1. input type=date
    3. Chapter 3

No **`hgroup`**
elements used



1. Welcome to the Future World of HTML 5
    1. Chapter 2
        1. New Structural Elements
            1. header Element
            2. footer Element
            3. section Element
        2. New Form Elements
            1. input type=date
    2. Chapter 3

`hgroup`
elements used

A complete example to explore can be found online, though you may find that a browser does not do anything of interest and that you need an outline simulator to see the difference between using `<hgroup>` tags or not.

ONLINE *http://htmlref.com/ch2/hgroup.html*

Given these semantics, it is clear that HTML5 sectioning elements are not just a formalization of `<div>` tags with appropriate `class` values. For example, you might consider

```
<div class="header">
  <!-- header here -->
</div>
<div class="section">
   <div class="header">
    <h2>Section Heading</h2>
   </div>
   <p>Content of section.</p>
</div>
<div class="footer">
  <!-- footer here -->
</div>
```

to be roughly the same as the previously introduced elements. To some degree this is true, but clearly the names of the `class` values aren't defined by a standard nor is any outlining algorithm defined.

Beyond sectioning, HTML5 introduces a number of other structural elements. For example, the `article` element is used to define a discrete unit of content such as a blog post, comment, article, and so on. For example, the following defines a few individual blog posts in a document:

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>HTML5 article example</title>
</head>
<body>

<header>
 <hgroup>
  <h1>Welcome to the Future World of HTML5 Blog</h1>
  <h2>Don't be scared it isn't that hard!</h2>
 </hgroup>
</header>
<section id="articleList">
 <h2>Latest Posts</h2>

 <article id="article3">
  <h2>HTML5 Here Today!</h2>
  <p>Article content here...</p>
 </article>
 <article id="article2">
  <h2>HTML5 Widely Misunderstood</h2>
  <p>Article content here...</p>
 </article>

<article id="article1">
  <h2>Discovering the article element</h2>
  <p>Article content here...</p>
 </article>
</section>

<footer>
 <p>This fake blog example is not real.</p>
</footer>

</body>
</html>
```

ONLINE *http://htmlref.com/ch2/article.html*

The idea of defining these discrete content units specially is that you might wish to extract them automatically, so again, having defined elements as opposed to some ad hoc use of **`class`** names on **`<div>`** tags is preferred.

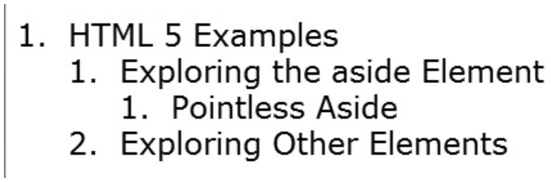
NOTE *Under early HTML5 drafts, the* **`article`** *element provided for* **`cite`** *and* **`pubdate`** *attributes, which may make the usage of the content more meaningful by outside sites; however, these were later dropped and use of* ***<time>*** *tags was encouraged.*

HTML5 also introduces an **`aside`** element, which may be used within content to represent material that is tangential or, as the element name suggests, an aside:

```
<p>Here we explore the various HTML5 elements.  I would write
   some real content here but you are busy reading the book anyway.
</p>

  <aside>
    <h2>Pointless Aside</h2>
    <p>Oh by the way did you know that the author lives in San Diego?
       It is completely irrelevant to the discussion but he seems
       to like the weather there as opposed to rainy New Zealand.</p>
  </aside>

<p>So as we continue to discuss the various HTML5 elements we must
    remember to stay focused as there is much to learn.
</p>
```

ONLINE *http://htmlref.com/ch2/aside.html*

You may have noted that an **`<h2>`** tag was used in the **`aside`**. While not required, it is useful as a reminder to readers that **`aside`** elements serve as outline sectioning elements, as shown here:



1. HTML 5 Examples
    1. Exploring the aside Element
        1. Pointless Aside
    2. Exploring Other Elements

NOTE *If a heading is not provided in an* **`aside`***, you may see an outline mechanism add "Untitled Section" or potentially even make up one based upon the start of the element content.*

Adding Semantics

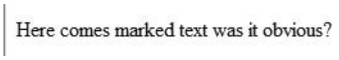
Many of the elements that HTML5 adds that can be used right away are semantic in nature. In this sense, HTML5 continues the appropriate goal of separating structure from style. In this section, you will see a number of repurposed elements as well as some that are all new. At first you won't see much value in using them other than to add semantics, but toward the end of the chapter we will explore how to make the elements understandable to most modern browsers and how to apply some simple styling for end users.

Marking Text

The new HTML5 element **`mark`** was introduced for highlighting content similarly to how a highlighter pen might be used on important text in a book. The following example wraps a few important words:

```
<p>Here comes <mark>marked text</mark> was it obvious?</p>
```

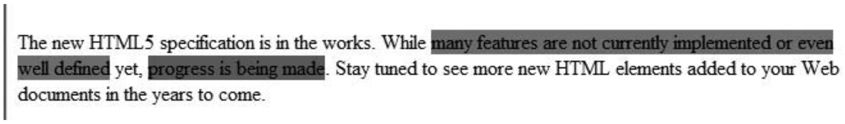
Unfortunately, you won't necessarily see anything with such an example:



Here comes marked text was it obvious?

You would need to apply a style. Here, inline styles are used just to show the idea:

```
<p>The new HTML5 specification is in the works.  While <mark
style="background-color: red;">many features are not currently
implemented or even well defined</mark> yet, <mark
style="background-color: green;">progress is being made</mark>.
Stay tuned to see more new HTML elements added to your Web documents
in the years to come.</p>
```



The new HTML5 specification is in the works. While many features are not currently implemented or even well defined yet, progress is being made. Stay tuned to see more new HTML elements added to your Web documents in the years to come.

ONLINE *http://htmlref.com/ch2/mark.html*