Excerpt from "Sams Teach Yourself HTML5 Mobile Application Development" by Jennifer Kyrnin (Sams, 2011)

HOUR 9

Adding Meaning with HTML5 Sectioning and Semantic Elements

What You'll Learn in This Hour:

- How to define sections of HTML5 documents
- The four primary sectioning elements
- Understanding elements like `<figure>` and `<details>`
- Using the many semantic elements
- Providing meaning with semantic elements

Many of the new elements in HTML5 relate to how the content is organized on the web page. These are called sectioning and heading elements. This hour you will learn more about these elements as well as some new semantic tags that help you define your web pages more precisely.

You will learn the value of marking up your pages semantically and about some of the new and existing semantic elements that you may not be using.

What Are Sectioning Elements?

Sectioning elements were added to HTML5 to help web designers organize content. These elements create logical sections of the document, each of which would start with a heading element, and often end with a footer element.

The four sectioning elements in HTML5 are

- `<article>`
- `<aside>`
- `<nav>`
- `<section>`

Two new elements, though not explicitly sectioning elements, are sectioning roots:

- `<details>`
- `<figure>`

A few other new elements could be considered sectioning elements as well. These new elements work with the sectioning elements and sectioning roots:

- `<header>`
- `<footer>`
- `<hgroup>`

One thing to keep in mind is that sectioning and semantic elements do not, by themselves, make any visible changes to the contents of a web page. They are used to provide structure to an HTML document that a computer can read. This allows the computer to create outlines as described next or otherwise interact programmatically with the document. If you want these elements to have a visual style in your pages, you need to attach CSS styles to them.

Using the New Sectioning Elements

One thing you should keep in mind is that the sectioning elements `<article>`, `<aside>`, `<nav>`, and `<section>` each work with the header and footer elements to create a section of your document. Following each of these elements in your document with a headline tag such as `<h1>` should be possible. And with those headline tags, your HTML5 document will have an outline.

Creating Outlines with Sectioning Elements

HTML5 introduces a new concept into the HTML specification: outlines. HTML5 outlines look at how the document is structured, using sectioning elements and header elements to label the outline sections. These outlines are similar to ones that a word processing program might create.

The purpose of sectioning elements is to create outlines with headers and footers. Specifically, the outlines can be created with algorithms. Outlines are important because they make the pages more accessible. Accessible user agents, such as screen readers, can use the outline to determine what parts are the most important to present to the user.

An HTML5 document might look something like this:

```
<section>
    <h1>Part 1: Building Web Pages</h1>
    <section>
        <h1>Chapter 1: HTML</h1>
        <p>In this section you will learn how to write HTML
    </section>
    <section>
        <h1>Chapter 2: CSS</h1>
        <p>And this section will teach you CSS
        <section>
            <h1>CSS1</h1>
            <p>...
        </section>
    </section>
</section>
<section>
    <h1>Part 2: Publishing Web pages</h1>
    ...
</section>
```

This creates an outline that looks like this:

```
I. Part 1: Building Web Pages
    I.1 Chapter 1: HTML
    I.2 Chapter 2: CSS
        I.2.i CSS1
II. Part 2: Publishing Web pages
```

HTML 4 had no sectioning elements, and so web designers typically used the `<div>` tag to create artificial sections within their documents. Although these worked, they were inconsistent across sites, so assistive devices could never use them to help determine content priority.

By the Way

Testing for the Correct Use of Sectioning Elements

One way you can test your HTML5 document to see whether you are using the sectioning elements correctly is to test your page in the HTML5 Outliner (http://gsnedders.html5.org/outliner/). This shows you how your outline looks, and you'll be able to see what headers you may be missing.

The best thing about the new sectioning elements is how easy they make the HTML to read. For example, when you see the <article> tag, you know that the contents are an article or some other type of syndicatable content. The HTML5 outlines are a representation of that structure.

The <article> Element

The <article> element represents a section of content that can stand on its own. If you were to syndicate a web page, you would syndicate the article portions—they are the part of the page that defines the page. An article might be

- A magazine or newspaper article
- A blog post
- A forum post
- A blog comment
- Any independent item of content

You can put a <header> and a <footer> element inside an <article>. Most articles also have some type of headline. You can also put <section> tags inside articles, or you can put an article inside a section.

The easiest way to think about what type of content should be in an article is to ask yourself whether it is content that can stand alone in syndication—such as in an RSS feed.

Here is how a simple <article> might look (assume it has a lot more content):

```
<article>
    <h1>My Pets</h1>
    <p>I have always loved animals, and I've always had a lot of pets....
</article>
```

The <aside> Element

The <aside> element is defined as content that is tangentially or loosely related to the main content of the document. However, it can also be a sidebar on a web page, providing information that is related to the site as a whole, rather than the main content of the page.

The way you determine which type of <aside> you're using depends upon where that <aside> is found. If the <aside> is found inside an <article> tag, then it

defines content that is related to the article itself, such as a glossary or list of related articles. If the <aside> is found outside of an <article> tag, then it defines content that is related to the website as a whole, but not the article(s) on the page, necessarily. Examples would be sidebar elements such as a blogroll or advertising (that is related to the page content).

By the Way

Sectioning Elements Do Not Have Page Locations Defined

It is easy to think that an <aside> element will appear on the side of a web page and a <footer> element will appear at the bottom. But the placement of these elements on a web page is defined only by CSS, not by the tags. You can place <footer> content at the very top of your page, if you feel that semantically that content is information usually found in a footer, or content in an <aside> element can appear in the main column of the page. These elements are semantic only, meaning that they define what the content *is* rather than where it should display or what it should do.

Here is an example of the two types of <aside> elements. This is an <aside> inside the <article>, which indicates that it is related specifically to the content on the page:

```
<article>
    <h1>My Pets</h1>
    <p>I have always loved animals, and I've always had a lot of pets....
    <aside>
        <h1>Photos of my Pets</h1>
        <ul>
            <li><a href="">Shasta</a></li>
            <li><a href="">Suni</a></li>
            ...
        </ul>
    </aside>
</article>
```

This <aside> is outside the <article>, so it is related to the content of the site as a whole but not to the article specifically:

```
<aside>
    <h1>My Favorite Sites</h1>
    <ul>
        <li><a href="">Dogs</a></li>
        <li><a href="">Cats</a></li>
        <li><a href="">Horses</a></li>
    </ul>
</aside>
```

Did you Know?

What's with All the <h1> Elements?

One thing you may have noticed is that every section has a headline using the <h1> element. In HTML 4 the <h1> through <h6> tags were the only outlining option designers had, so the recommendation was to use only one <h1> tag—for the page title. However, HTML5 added all the additional sectioning elements, and that the recommendation is now to use <h1> as the first headline for every section. This indicates that the <h1> headline is the most important headline *of that section*, leaving the <h2> through <h6> headlines as subheads there. You may find that older browsers may respond awkwardly to this convention, especially with CSS disabled, and if so, you can continue to use <h2> for second-level headlines and so on, but the W3C recommends using <h1> as the first headline in any new section.

The <nav> Element

The <nav> element defines a section of the content that links to other pages or areas of the site. You don't have to enclose every list of links you put on a page in a <nav> element. It is only for the major navigation of a page.

Most web designers are already using a form of the <nav> element when they put in a or <ul> with an ID of "nav" or "navigation", such as:

```
<ul id="nav">
    <li><a href="">Home</a>
    <li><a href="">Products</a>
    <li><a href="">Services</a>
    <li><a href="">About Us</a>
</ul>
```

If you have a section on your web page like that, all you need to do is add the <nav> element around it.

```
<nav>
<h1>Navigation</h1>
<ul id="nav">
    <li><a href="">Home</a>
    <li><a href="">Products</a>
    <li><a href="">Services</a>
    <li><a href="">About Us</a>
</ul>
</nav>
```

The best way to use the <nav> element is to use it for major navigation, because that is typically the only navigation most users care about. However, what major navigation is depends on the site and the site's developers. Some sites have only one major

navigation section, with other navigation elements just being links within the site. Other sites might have three or four <nav> elements. For example, my site on About.com (http://webdesign.about.com/) has several forms of navigation:

- A bread crumb trail at the top
- Four tabs below the site name
- The "Must Reads" section
- The "Browse Topic" links
- And even the "Explore Web Design / HTML" section at the bottom of the page

Most of these could be considered major navigation, but if I were to convert this page to HTML5, I would add <nav> elements around the four tabs and the "Browse Topic" links, as those provide the best navigation through the site. Until the specification is clear on this topic, either solution, marking all navigation as <nav> or just the major navigation, is okay.

Some places you might use the <nav> element, beyond the primary navigation bar on your site include:

- Table of contents
- Previous and next links
- Breadcrumb trail

You should be careful not to confuse the <nav> element with the <menu> element. The <menu> element is a list of commands. It can be a navigation list where each item points to a different location—like a goto list. If the menu is a list of major navigation elements, then you can put it inside of a <nav> element, but it shouldn't replace the <nav> element.

The <section> Element

The <section> element is possibly the most confusing of the new sectioning elements. Most designers who use it tend to use it incorrectly. The most common way it's used is to set up page divisions for styles, which is incorrect usage. You should be using the element for styling your layout.

Don't feel bad if you have been using the <section> element incorrectly. Many people have. In fact, you can find it used incorrectly on many sites teaching HTML5.

According to the W3C, "The section element is not a generic container element. When an element is needed for styling purposes or as a convenience for scripting, authors are encouraged to use the div element instead."[1]

Here are a few ways you can use to determine whether you should use a <section> element:

- Does the section have a natural headline? If it doesn't, you shouldn't use a <section> element.
- Would the section be a natural part of a page outline? If it isn't, then it shouldn't be a <section>.
- Is there more purpose to the section than just the style? If you're using the <section> tag just as hooks for styles, you should use the element instead.
- Does the section content meet the criteria of an <article>, <aside>, or <nav>? If syndicating the content makes sense then you should use an <article>. If the content is related to the site or to the article then <aside> is appropriate. Of course, if it's navigation, then the <nav> element is best.

One place you might put a <section> element is in an <aside> for your blog. Most blogs have an <aside> that includes sections for a Blogroll, latest posts, categories, and so on. Each of those could be a <section>. For example:

```
<aside>
    <section>
        <h1>Blogroll</h1>
        <ul>
            <li><a href="http://diveintohtml5.org">Dive into HTML5</a></li>
            <li><a href="http://html5gallery.com/">HTML5 Gallery</a></li>
            <li><a href="http://dev.w3.org/html5/spec/Overview.html">HTML5
➥Specification (W3C)</a></li>
        </ul>
    </section>
    <section>
        <h1>Categories</h1>
        <ul>
            <li>...</li>
        </ul>
    </section>
</aside>
```

[1] "The Section Element." HTML5 Working Draft. www.w3.org/TR/html5/sections.html#the-section-element. Referenced May 12, 2011.

**When to Use **

The element has been the standby for designs and layout for standards-based designers for several years now, but now with HTML5, that use should lessen. In fact, you should really only use the element when no other appropriate element exists. For now, you may want to continue to use it in conjunction with the sectioning tags (that is, <article>), but you should only do that if you are having trouble getting your pages to display correctly.

By the Way

Sectioning Root Elements

The sectioning root elements are elements in HTML5 that can have their own outlines, but they don't contribute to the outlines of their ancestors. These elements are:

- <blockquote>
- <body>
- <details>
- <fieldset>
- <figure>
- <td>

You should already be familiar with the <body>, <blockquote>, <fieldset>, and <td> elements. The new elements in HTML5 are <details> and <figure>.

You use the <details> element to hide and show additional information about the content. It has a related element that is also new—<summary>. The <summary> element is an optional element inside of the <details> element that represents a caption or summary of the <details> element.

Use these elements to add information to your documents that isn't critical to the content. For example, you can add information about a form field in a <details> element that people can open if they need more help:

```
<input id="phone-number" type="phone">
<details>
<summary>Format</summary>
<p>(xxx) xxx-xxxx
</details>
```

You use the <figure> element to define self-contained units of content. Most commonly a figure is an image, but it can be any type of content that can stand on its own. The related element <figcaption> provides a legend or caption for a figure.

Some common figures include:

- Images or groups of images
- Blocks of code
- Poetry or quotations
- Charts and graphs

When you are deciding whether to put content into a `<figure>` you need to determine whether the content is essential to the content of the page. If it is essential, and the exact placement in the content is not critical, then a `<figure>` element is a good choice.

You can write a simple `<figure>` like this:

```
<figure>
<img src="shasta.jpg" alt="Shasta">
<figcaption>
<p>A photo of my dog Shasta.
</figcaption>
</figure>
```

Remember that you don't use the `<figure>` element around every image on your page. Banners and advertisements are not related to the page contents and so are not figures.

Heading, Header, and Footer Elements

The heading, header, and footer elements are not technically sectioning elements because they do not contribute to the site outline, but most web designers think of them as sectioning elements because they are used to define the semantic layout of your web document.

Heading elements include `<hgroup>` and all the heading tags: `<h1>` through `<h6>`. Use the `<hgroup>` element to group a set of one or more heading elements so that they are collected as one element in the document outline. For example:

```
<h1>This is a Headline</h1>
<h2>This is a Sub-Head</h2>
<section><h1>And this is a Sub-section</h1></section>
```

This has the following outline:

```
1. This is a Headline
    1. This is a Sub-Head
    2. And this is a Sub-section
```

Suppose you group the <h1> and <h2> together in an <hgroup>:

```
<hgroup>
<h1>This is a Headline</h1>
<h2>This is a Sub-Head</h2>
</hgroup>
<section><h1>And this is a Sub-section</h1></section>
```

Then you get the following outline:

```
1. This is a Headline
    1. And this is a Sub-section
```

You can use the <header> and <footer> elements the same way you might have used and in your HTML. However, don't use the <hgroup> element unless you really are grouping together more than one heading tag. Writing the following is redundant and incorrect:

```
<hgroup><h1>My Headline</h1></hgroup>
```

Also, although the <hgroup> element is valid at the time of this writing, some controversy about whether it should remain in the HTML5 specification still exists, and it may be left out of the final HTML5 specification.

The <header> can contain a section's headline as well as additional information that introduces the section, such as a table of contents, dateline, or relevant icons. What is crucial to understand is that though most of the time you will use the <header> element at the top of your document, you can also include a <header> element at the beginning of any sectioning element, such as an <article>, <aside>, or <nav>.

The <footer> element acts just like the <header> element except that it comes at the end of a section. A footer contains information about the section such as who wrote it, copyright data, or even related links. Just like the <header> element, you can include a <footer> in any sectioning element, and it does not create a new section.

By the Way

Don't Get Hung Up on the Name "Footer"

Even though the most common location for a footer is at the bottom or foot of the page, you don't have to place it there in your design. For example, in a blog post, information such as the date the post was written and the author's name could be considered footer information, but you can put that footer at the top of the post if that's where you want it to appear in your design.

If you have contact information in your footer for the author of a post or the page, you should wrap it in the <address> element. You can then put the <address> in the <footer>.

There is a lot to learn when starting to build correctly sectioned HTML5 documents. But with practice you will start to understand the purpose of each of the sectioning elements so that you can build a correctly sectioned document yourself.

▼ Try It Yourself

Building a Blog Using HTML5 Sectioning Elements

Blogs typically have standard page elements such as posts, comments, navigation, and so on. By following these steps you can create a blog page that contains two posts, with space for comments, site navigation, a sidebar, and a site header and footer.

1. Start your HTML5 document as you normally would:

```
<!DOCTYPE HTML>
<html lang="en-us">
<head>
    <meta charset="UTF-8">
    <title>My Blog</title>
    <script src="modernizr-1.7.min"></script>
</head>
<body class="no-js">
</body>
</html>
```

2. Add a container inside the <body> tag so that you have a hook for your CSS styles:

```
<body class="no-js" >
    <div id="container">
    </div>
</body>
```

3. Create a <header> element inside the to contain your blog title, tagline, and navigation:

```
<div id="container">
    <header>
        <hgroup>
            <h1>My Blog</h1>
            <h2>Where I Talk About What Interests Me</h2>
        </hgroup>
        <nav>
            <h1>Navigation</h1>
            <ul>
```

```
                <li><a href="">Home</a></li>
                <li><a href="">About Me</a></li>
                <li><a href="">Photos</a></li>
            </ul>
        </nav>
    </header>
```

4. After the <header> add an <article> with a <header> that contains the post title:

```
</header>
<article>
    <header>
        <h1>Post #1</h1>
    </header>
</article>
```

5. Place your post contents after the article <header>:

```
<article>
    <header>
        <h1>Post #1</h1>
    </header>
    <p>This is my first post. It includes a photo of my dog.
    <figure>
        <img src="shasta.jpg" alt="Shasta">
        <figcaption>My Dog Shasta</figcaption>
    </figure>
</article>
```

6. Put a <footer> including your name and the date inside the <article>:

```
<footer>
    <p>By: <address><a
href="mailto:enn@html5in24hours.com">Jennifer</a></address>
    <p>Date: <time datetime="2011-05-12">May 12, 2011</time>
</footer>
</article>
```

7. Put the comments in a second <article> inside the post's <article>:

```
<article>
    <h1>Comments</h1>
    <p><a href="">Post</a> your comments or <a href="">read</a>
other comments
    </article>
</article>
```

8. Outside the post's <article> but still inside the put an <aside> to include a site sidebar:

```
<aside>
    <h1>Learn More</h1>
    <section>
```

```
        <h1>Recent Comments</h1>
        <p>None
    </section>
    <section>
        <h1>Stay in Contact</h1>
        <p><a href="">RSS</a> or <a href="">Newsletter</a>
    </section>
</aside>
</div>
```

9. Finally, add a `<footer>` with your copyright information at the bottom of the document:

```
</aside>
<footer>
    <p>Copyright &copy; 2011 Jennifer Kyrnin
</footer>
</div>
```

You will want to add styles to make the page look nicer, and of course, you will want to add more content. But if you check, the HTML will outline correctly with no untitled sections.

The complete HTML is listed at www.html5in24hours.com/examples/sectioning-example-html.html.

▲

Marking Up HTML Semantically

Marking up HTML5 documents semantically involves more than just using sectioning elements. Semantic elements describe what the content is. HTML5 has only a couple new semantic tags, but a number of HTML 4 elements have also been changed in HTML5 to be more semantic.

The benefit of using semantic HTML tags is that you provide more information to the browsers, but if the browsers don't support the tags, they just ignore them. Using them can only improve your web pages.

Mobile support by iOS and Android is really good for the sectioning and semantic elements. Android has supported them since 2.1, and iOS had partial support in 3.2 and full support in 4.0 and later.

In fact, the majority of difficulty you will have with these elements is with Internet Explorer before version 9. If you want to support these browsers, you can use the scripts mentioned in Hour 8, "Converting Web Apps to Mobile."

Semantic HTML 4 Elements

A bunch of elements that are part of the HTML 4 specification are semantic:

- **<abbr> –** Abbreviations

By the Way

The <acronym> Element Is Obsolete

The <acronym> tag is a part of HTML 4 and defines acronyms—abbreviations that form words themselves, such as NASA, FAQ, or SCUBA. But because of the confusion between abbreviations and acronyms, and the fact that acronyms are also abbreviations, this element was made obsolete in HTML5.

- **<code>**—Code samples
- **<del>**—Deleted text
- **<dfn>**—Defining instance of a term
- ****—Emphatic stress
- **<ins>**—Inserted stress
- **<kbd>**—Keyboard input
- **<samp>**—Sample output
- ****—Strong importance
- **<var>**—Variable or placeholder text

Newly Semantic HTML 4 Elements

Several HTML elements were not semantic in HTML 4, but had their focus changed in HTML5 to become semantic:

- **<b>**—This used to mean just bold text, but now it represents text that would normally be displayed in bold, but doesn't have extra emphasis.
- **<hr>**—This used to represent a horizontal line, but now it represents a thematic break in the content.
- **<i>**—This used to mean italic text, but now it represents text that would normally be displayed in italics, but doesn't have any extra emphasis.
- **<s>**—This used to mean text that had a line through it (strikeout), but now represents content that is no longer accurate and has been "struck" from the document.

- **<small>**—This used to be text that was printed smaller than the surrounding text, but now it represents "small print" such as legalese.
- **<u>**—This used to be text that was underlined, but now it represents text that would normally be displayed with an underline.

By the Way

Emphasis Refers to Contents Read Aloud

When referring to text that is emphasized or not to be emphasized, this generally refers to how the text might be read in a screen reader. A book title would be italicized, but when read out loud, no special distinction would be made regarding the italics, as it is just a book title. In written form, the text may seem to stand out more, but with elements like <i> and <b> they would not be called out when read aloud.

New Semantic Elements in HTML5

A few new semantic elements in HTML5 enable you to further define the meaning of your content: <mark>, <meter>, <progress>, and <time>.

You use the <mark> element to indicate text that should be highlighted or stand out for reference purposes. You see text that is marked every time you use a search engine. The words that you searched for are highlighted (usually in bold) in the results text, as you can see in Figure 9.1.

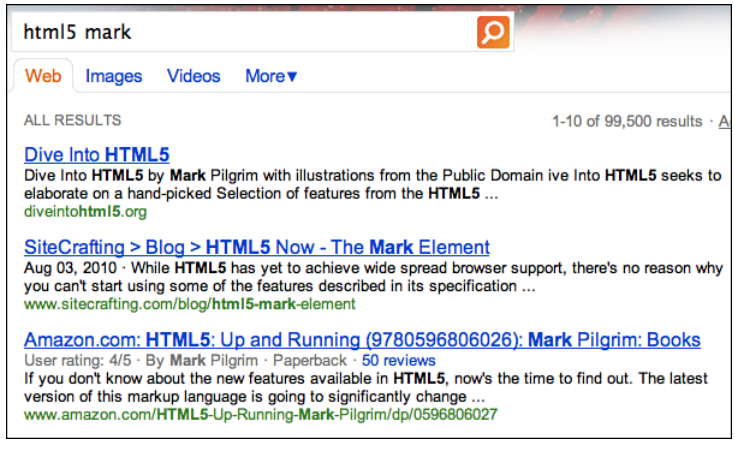

FIGURE 9.1 Searching Bing for "HTML5 mark" with the results in bold.

You should use the <mark> element whenever you want to draw attention to sections of the text in a document, such as with the search terms in Figure 9.1 or to otherwise

highlight a portion of text, that wouldn't otherwise be highlighted. It is different from the and elements because <mark> adds no extra emphasis or importance to the text, it is just highlighted.

You use the <meter> element to indicate a scalar measurement within a range, such as a star rating on a review, a letter grade on an exam, a percentage, or a fraction.

The <meter> element has six attributes:

- **value**—The measured amount shown by the meter. For example, a grade on a paper might be written <meter value="91">A-</meter>. This attribute is required, but can be what is written in the element. For example, <meter>3 out of 15</meter>. The value is "3" and the max is "15," because that is what is written in the element.
- **min**—The minimum for the range.
- **max**—The maximum for the range. For example, <meter value="91" min="0" max="100">A-</meter>.
- **low**—The point that marks the upper boundary of the "low" portion of the meter.
- **high**—The point that marks the lower boundary of the "high" portion of the meter. Along with low, high lets you divide your meter range into three parts, for example <meter value="5" low="3" high="7">1 to 10</meter>.
- **optimum**—The point that marks the optimal point on the range.

You can also use the title attribute to specify the unit, such as inches, days, or dollars.

Use the <meter> element to define ranges and provide extra information about that range. The contents of the <meter> element can be text that is then defined by the value attribute, such as:

```
Keep the dough <meter min="80" max="90" title="degrees">warm</meter>.
```

Watch Out!

Meter Is Not for Measurements

Although putting a <meter> element around things such as weights and heights is tempting, if there is not a definitive range, you should leave the text alone. Using it on measurements is incorrect unless there is a known maximum value.

The <progress> element is a form element that you use to indicate the progress of something that might take time to do, such as:

- Something being downloaded or uploaded
- A computation performing
- A game loading

Following is an example <progress> element:

```
I am <progress max="24" value="12" title="chapters">half way</progress>
through the book.
```

The `<progress>` element has limited browser support right now, working only in Opera and Chrome. However, you can use a polyfill (such as this one at http://blogupstairs.com/html5-polyfill-progress-element/) or a detection script and a jQuery plug-in for non-compliant browsers. You detect `<progress>` support like this:

```
return 'value' in document.createElement('progress');
```

The `<time>` element indicates a date or time. You use it to mark up times and dates in your document. You can add the `datetime` attribute to give the precise date and time, numerically. You can also add the attribute `pubdate` to indicate that the time is the publication date and time of the nearest ancestor `<article>` element or of the document as a whole.

The `datetime` attribute can contain a date, a time, or a date and time. It is written according to RFC3339 (http://tools.ietf.org/html/rfc3339) in the following formats:

- **date**—`YYYY-MM-DD`.
- **time**—`HH:MM:SS`. Use a 24-hour clock; seconds are optional.
- **time with timezone**—`HH:MM:SS+Zone`. Timezones range from –12:00 to +14:00.
- **time at UTC**—`HH:MM:SSZ` Z is the same as Universal Coordinated Time (UTC), or +00:00.
- **time and date**—`YYYY-MM-DDTHH:MM:SS+Zone`. The T in the middle is just a T, to separate the date from the time.

The text in your <time> elements doesn't have to display a full date or time. Here are some examples:

```
<time datetime="1940-10-04">October 4, 1940</time>
<time datetime="1940-10-04">Oct 4</time>
<time datetime="00:00:00-08:00">Midnight</time>
```

One thing to keep in mind is that the <time> element cannot set specific datetime values for pre-AD dates. You also cannot encode imprecise dates such as "March 2008." To indicate dates of this nature, you need to use microformats, which Hour 15, "Microformats and Microdata," discusses in more detail.

Summary

This hour taught you all about the new semantic elements in HTML5. You learned how to create semantic markup that can be outlined using sectioning elements as well as the semantic elements added and changed in HTML5.

Outlines may not seem important to some designers, but anyone who designs with accessibility in mind knows that outlines help screen readers use the documents more easily. Well-marked-up documents with sectioning elements create outlines that are easy to read.

Semantic elements may also not seem important, but providing extra information by indicating the semantics of text doesn't hurt either. Plus, you might be surprised at how much it does help.

Q&A

Q. ***Do I have to use every sectioning element in every document I create?***

A. No, of course not. Like all other semantic elements, you should use the sectioning elements that make sense in your document. If you don't have tangential text, then you shouldn't use the <aside>. Likewise, if you have no sections that can't be defined by other sectioning elements, then the <section> element is not needed.

Q. ***Why do you refer to some elements as "obsolete?" I thought the term was "deprecated."***

A. HTML 4 was a standard written primarily for browser makers. The term *deprecated* was intended to tell them that they no longer needed to support that item in the future. HTML5 was written more for web authors and so *obsolete* was considered a clearer term.

Q. ***What if I used the*** `<meter>` ***element with a value outside the minimum or maximum?***

A. Although the possibility exists that a future browser might display some type of error message if you did this, the chances are very slim. As with all semantic elements, you should try to use the elements correctly.

Workshop

The workshop contains quiz questions to help you process what you've learned in this chapter. Try to answer all the questions before you read the answers. See Appendix A, "Answers to Quizzes," for answers.

Quiz

1. What are the four new sectioning elements in HTML5?
2. Which of the following are valid within a `<figure>`?
   a. An image
   b. A block of text
   c. Both
3. True or False. An `<aside>` should be used to mark up a site sidebar.
4. True or False. A `<section>` should be used as just a hook for styles.
5. True or False. A sectioning root contributes its contents to the entire site outline.
6. True or False. A `<footer>` is not a sectioning element.
7. What are the new, non-sectioning, semantic elements in HTML5?

Exercises

1. Write an HTML document with correct sectioning elements. Check your outline in the HTML5 Outliner (http://gsnedders.html5.org/outliner/). If you have any untitled sections, check to make sure that you need that section, and if you do, add a headline with an `<h1>` element.

2. Examine some existing web pages, either your own or online. See whether you can find places where the `<mark>`, `<meter>`, `<progress>`, and `<time>` elements would be appropriate. If you are examining your own pages, add in the elements so that your pages are more semantic.