3

How to use HTML to structure a web page

In chapter 2, you saw the basic structure of an HTML document, you learned the basic techniques for coding the elements that make up a document, and you learned that these elements specify the structure and content of a page when it's displayed in a browser. Now, in this chapter, you'll learn how to code the HTML elements that you'll use in most of the documents you create. Then, in the next three chapters, you'll learn how to use CSS to format those HTML elements.

How to code the head section

The head section of an HTML document contains elements that provide information about the web page rather than the content of the page. In chapter 1, for example, you learned that the title element sets the text that's displayed in the tab for the page in the web browser. Now, you'll learn more about this element as well as some other elements that you can code in the head section.

How to code the title element

The head section of every page should include a unique title element that describes the content of the page. In the HTML shown in figure 3-1, for example, you can see that this element gives the name of the organization followed by the keywords *speakers* and *luncheons*. This title is used by search engines for search engine optimization, and it appears in the results of a search to help the users decide whether they want to go to that page. That's why you should follow the SEO guidelines in this figure when you code this element.

The content of the title element is also displayed in the browser's tab for the page. As you can see in this figure, though, only the portion of the title that fits in the tab is displayed. But if you hover the mouse over the tab, the entire title will be displayed in a tooltip.

How to link to a favicon

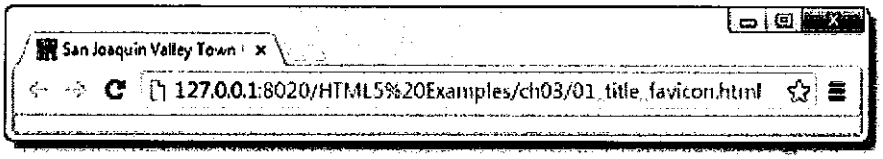
The link element is an optional element that you can use to specify a file that should be linked to the web page. In figure 3-1, the link element is used to link a custom icon, called a *favicon*, to the web page. This causes the icon to be displayed to the left of the title in the tab for the page. Note that you typically name this icon favicon.ico as shown in this figure.

When you code the link element, you typically include two attributes. The rel attribute indicates the relationship of the linked resource to the document. For a favicon, you use the value "shortcut icon".

The href attribute provides the URL of the resource. In this example, the favicon is in the same folder as the HTML file, so the URL is just the filename. If the file is in a different location, though, you can use an absolute or relative URL to identify that location. You'll learn how to code absolute and relative URLs later in this chapter.

Note that when you deploy a website to a web server, it isn't necessary to include a link element for the favicon. Instead, you can place the favicon in the root folder for the website and it will automatically appear in the tab for each page. In this book, though, we've included the link element so you can see the favicon as you develop a website on your own computer.

A browser that shows the title and favicon



A head section that specifies a title and links to a favicon

```
<head>
    <title>San Joaquin Valley Town Hall | speakers and luncheons</title>
    <link rel="shortcut icon" href="favicon.ico">
</head>
```

SEO guidelines for the title tag

- Always code a title tag in the head section.
- The title should accurately describe the page's content, and it should include the one or two keywords, called focus keywords, that you want to be used to rank the page.
- The title should be interesting enough to entice the reader to click on it when it's shown in the search results for a search engine.
- The title should be unique for each page in your website.
- Limit the length of your titles to around 65 characters because most search engines don't display more than that in their results.

Description

- The title element specifies the text that's displayed in the browser's tab for the web page.
- The title is also used as the name of a favorite or bookmark for the page.
- A custom icon, called a *favicon*, is typically named favicon.ico. A favicon typically appears to the left of the title in the browser's tab for the page. It may also appear to the left of the URL in the browser's address bar, and it may be used in a favorite or bookmark.
- To specify a favicon for a page, you can use a link tag exactly like the one shown above.
- When you deploy a website to a web server, you can omit the link element for the favicon if you place the favicon in the root folder for the website.
- To create an ico file, you can use an icon editor, a program that converts an image to an ico file, or a web-based converter. You may also be able to find an icon on the Internet by searching for "web icons". For more information, see chapter 11.

Figure 3-1 How to code the title element and link to a favicon

How to include metadata

The meta element is another optional element that you can code within the head element. You use it to specify *metadata*, which provides information about the content of the document.

The head element in figure 3-2 includes three meta elements. The first one specifies the character encoding used for the page, and UTF-8 is the encoding that's commonly used for the World Wide Web. Since this element is required for HTML5 validation, you should include it in the head section of every HTML document, and it should be one of the first elements in the head section.

The next two meta elements in this figure provide metadata that can be used by search engines to index the page. Here, the first element uses the name attribute with the value "description" to indicate that the content attribute that follows contains a description of the web page. This can be a longer description than the one in the title element, and it should also be unique for each page in your website.

The second meta element uses the name attribute with the value "keywords" to indicate that the content attribute contains a list of keywords related to the page. At one time, the keywords were an important factor for search engine optimization. However, Google and Bing no longer use the keywords, although other search engines like Yahoo may. This is partly due to the fact that some web masters used keywords to misrepresent what a page contained.

Because the algorithms that search engines use are changed frequently, it's hard to know what the best use of metadata is. There is also the danger that you might change the content of a page and forget to change the metadata. In that case, a search engine might index the page incorrectly. Nevertheless, it's still a good practice to provide at least the description metadata for the important pages of a website. Just make sure that your descriptions accurately represent the contents of your pages.

A head section that includes metadata

```
<head>
    <title>San Joaquin Valley Town Hall | speakers and luncheons</title>
    <meta charset="utf-8">
    <meta name="description" content="A yearly lecture series with speakers
        that present new information on a wide range of subjects">
    <meta name="keywords" content="san joaquin, town hall, speakers,
        lectures, luncheons">
</head>
```

Three attributes of the <meta> tag

Attribute	Description
charset	A required tag in HTML5 that specifies the type of character encoding to be used for the page. UTF-8 is the encoding that's commonly used for the World Wide Web.
name	Specifies the type of metadata being added to the document. The values "description" and "keywords" can be used to specify content that's used by some search engines.
content	Specifies the value to be used for the item specified by the name attribute.

SEO guidelines

- Code the description metadata for each page of your website. It should summarize the contents of the page, it should be unique for each page, and it can be longer than the title tag. When it is displayed in the search-engine results, it should encourage users to click on your link.
- Code the keywords metadata for each page of your website. It should consist of no more than 10 keywords or phrases, and it should be unique for each page.

Description

- The meta element provides information about the HTML document that's called *metadata*.
- The charset metadata is required for HTML5 validation.
- All or part of the description metadata may be displayed in the search results of some search engines.
- Google and Bing ignore the keywords metadata, but Yahoo may still use it.

Figure 3-2 How to include metadata

How to code text elements

Within the body of a document, you can code two types of elements: block elements and inline elements. In the topics that follow, you'll learn how to code a variety of block and inline elements that define the text for a document.

How to code headings and paragraphs

Headings and paragraphs are the most common content of a web page. These are defined by the HTML elements shown in figure 3-3. These elements are called *block elements*, and each one begins on a new line when it is displayed.

If you review the example in this figure, you shouldn't have any trouble understanding how these elements work. Here, the HTML uses the h1, h2, and <p> elements to generate the text that's shown. When these elements are displayed by a browser, each element has a default font and size that's determined by the base font of the browser. This base font is typically Times New Roman in 16 pixels.

When you use the h1 through h6 elements, you should use them to provide a logical structure for your document, not to format the text. That means that the most important heading on a page should always be an h1 element, and you should only code one h1 element on each page. That also means you should only go down one level at a time, not jump down two or more levels to indicate less importance. In other words, the first heading level after an h2 should be an h3, not an h4.

For example, the h1 element in this figure is used to mark the most important heading on the page, and the h2 elements are used to mark the next level of importance. Then, if the first h2 element required two subheadings below it, they would both be coded at the h3 level. This structure helps search engines index your site properly, and it makes your pages more accessible to devices like screen readers. Then, you can use CSS to size and format the text in these elements, as shown in the next chapter.

Common block elements for headings and paragraphs

Element	Description
h1	Creates a level-1 heading with content in bold at 200% of the base font size.
h2	Creates a level-2 heading with content in bold at 150% of the base font size.
h3	Creates a level-3 heading with content in bold at 117% of the base font size.
h4	Creates a level-4 heading with content in bold at 100% of the base font size.
h5	Creates a level-5 heading with content in bold at 83% of the base font size.
h6	Creates a level-6 heading with content in bold at 67% of the base font size.
p	Creates a paragraph of text at 100% of the base font size.

HTML that uses the block elements

```
<h1>San Joaquin Valley Town Hall Programs</h1>
<h2>Pre-lecture coffee at the Saroyan</h2>
<p>Join us for a complimentary coffee hour, 9:15 to 10:15 a.m. on the day
   of each lecture. The speakers usually attend this very special event.</p>

<h2>Post-lecture luncheon at the Saroyan</h2>
<p>Extend the excitement of Town Hall by purchasing tickets to the
   luncheons.</p>
```

The block elements in a web browser

San Joaquin Valley Town Hall Programs

Pre-lecture coffee at the Saroyan

Join us for a complimentary coffee hour, 9:15 to 10:15 a.m. on the day of each lecture. The speakers usually attend this very special event.

Post-lecture luncheon at the Saroyan

Extend the excitement of Town Hall by purchasing tickets to the luncheons.

SEO guidelines

- Use the heading tags to show the structure and importance of the content on a page. Always use the h1 tag to identify the most important information on the page, and only code a single h1 tag on each page. Then, decrease one level at a time to show subsequent levels of importance.
- Don't use heading levels as a way to size text. Instead, use CSS to size the headings.

Description

- *Block elements* are the main building blocks of a website. Each block element begins on a new line.
- The base font size and the spacing above and below headings and paragraphs are determined by the browser, but you can change these values by using CSS.

Figure 3-3 How to code headings and paragraphs

How to code special blocks of text

In addition to the elements for headings and paragraphs, HTML provides some elements that you can use to code special blocks of text. For instance, three of the most common elements are described in figure 3-4.

You typically use the pre element to display preformatted blocks of code. When you use this element, any whitespace or line breaks that appear in the content is maintained. In addition, the content is displayed in a monospaced font. In this figure, the pre element is used to display two lines of JavaScript code.

You typically use the blockquote element to display an actual quote like the one shown in this figure. Depending on the browser, this element may cause the content to be indented from the left side of the block element that contains it or from both the left and right sides. But note that this doesn't add quotation marks to the text.

You can use the address element to display contact information for the developer or owner of a website. In this figure, the address element is used to display a phone number and an email address. As you can see, the browser displays this information in italics.

Although these elements provide some default HTML formatting, you shouldn't think of these elements that way. Instead, you should use these elements to identify specific types of content. Then, you can use CSS to format these elements so they look the way you want them to.

Block elements for special types of text

Element	Description
`pre`	Used for portions of code that are formatted with line breaks and spaces. Creates a block of preformatted text that preserves whitespace and is displayed in a monospaced font.
`blockquote`	Used for quotations. Can be used with the cite and <q> elements of figure 3-5.
`address`	Used for contact information for the developer or owner of a website.

HTML that uses the block elements

```
<p>How to use JavaScript to display the year:</p>
<pre>
        var today = new Date();
        document.writeln( today.getFullYear() );
</pre>

<p>Ernest Hemingway wrote:</p>
<blockquote>Cowardice, as distinguished from panic, is almost always
    simply a lack of ability to suspend the functioning of the imagination.
</blockquote>

<p>How to contact Mike Murach &amp;amp; Associates:</p>
<address>1-800-221-5528<br>
    <a href="mailto:murachbooks@murach.com">murachbooks@murach.com</a>
</address>
```

The block elements in a web browser

How to use JavaScript to display the year:

```
var today = new Date();
document.writeln( today.getFullYear() );
```

Ernest Hemingway wrote:

> Cowardice, as distinguished from panic, is almost always simply a lack of ability to suspend the functioning of the imagination.

How to contact Mike Murach & Associates:

1-800-221-5528
murachbooks@murach.com

Description

- These block elements identify the type of content that they contain. That's consistent with the way the HTML5 semantic elements are used (see figures 3-8 and 3-9).

Figure 3-4 How to code special blocks of text

How to code inline elements for formatting and identifying text

In contrast to a block element, an *inline element* doesn't start on a new line. Instead, an inline element is coded within a block element. In this topic, you'll learn how to code some of the most common inline elements for identifying and formatting text.

The first table in figure 3-5 presents three elements for formatting text. For instance, the sub element is used to display text as a subscript, as shown in the fourth example in this figure. The sup element is similar, but it displays text as a superscript. With the exception of the <br> element, each element must have both an opening and a closing tag. Then, the appropriate formatting is applied to the content between these tags.

In contrast, the <br> element starts a new line of text. You can use this element to start a new line within another element, but you shouldn't use it to provide space between block elements. Instead, you should use CSS to control the space between block elements, which you'll learn how to do in chapter 5.

The second table in this figure presents elements that are used for identifying inline content. For instance, you can use the abbr element to identify an abbreviation, the <q> element to identify a quotation, and the cite element to identify the source of a block element like a quotation.

Unlike the blockquote element, which is a block element, the browser adds quotation marks to a <q> element. In the example in this figure, the quotation marks are straight, although some browsers display curly quotation marks. Here, you can see that a <q> element is used for a quotation within a paragraph, but this inline element could also be used within a blockquote element to add the quotation marks.

If you want to emphasize textual content, you can use the em or strong element as shown in the first two examples in this figure. As you can see, the contents of an em element is displayed in italics, and the contents of a strong element is displayed in bold. In the past, you could use the <i> and <b> elements to display contents in italics or bold without implying any special meaning. However, they should no long be used for this purpose, which is why they aren't presented here.

Inline elements for formatting text

Element	Description
`sub`	Displays the content as a subscript.
`sup`	Displays the content as a superscript.
`br`	An empty element that starts a new line of text.

Inline elements for identifying content

Element	Description
`abbr`	Used for abbreviations.
`cite`	Used to indicate a bibliographic citation like a book title.
`code`	Used for computer code, which is displayed in a monospaced font.
`dfn`	Used for special terms that can be defined elsewhere (definitions).
`em`	Indicates that the content should be emphasized, which is displayed in italics.
`kbd`	Used for keyboard entries, which is displayed in a monospaced font.
`q`	Used for quotations, which are displayed within quotation marks.
`samp`	Used to mark a sequence of characters (sample) that has no other meaning.
`small`	Used to display "fine print" such as footnotes in a smaller font.
`strong`	Indicates that the content should be strongly emphasized, which is displayed in bold.
`var`	Used for computer variables, which are displayed in a monospaced font.

HTML that uses some of the inline elements

```
<p>If you don't get 78% or more on your final, <em>you won't pass.</em></p>
<p>Save a bundle at our <strong>big yearend sale</strong>.</p>
<p>When the dialog box is displayed, enter <kbd>brock21</kbd>.</p>
<p>The chemical symbol for water is H<sub>2</sub>O.</p>
<p><q>To sleep, perchance to dream-ay, there's the rub.</q></p>
```

The inline elements in a web browser

If you don't get 78% or more on your final, *you won't pass.*

Save a bundle at our **big yearend sale**.

When the dialog box is displayed, enter `brock21`.

The chemical symbol for water is H_2O.

"To sleep, perchance to dream-ay, there's the rub."

Description

- An *inline element* is coded within a block element and doesn't begin on a new line.
- The formatting elements don't imply any special meaning.
- The content elements are used to convey meaning. Then, you can use CSS to format them.

Figure 3-5 How to code inline elements for formatting and identifying text

How to code character entities

Many of the web pages you develop will require special characters such as a copyright symbol and opening and closing "curly" quotes. To display these special characters, you use *character entities*. Figure 3-6 presents the most common of these entities.

As you can see, all character entities start with an ampersand (&) and end with a semicolon (;). Then, the rest of the entity identifies the character it represents. To insert the copyright symbol (©), for example, you use the © character entity.

Because the & character marks the start of each character entity, you shouldn't use this character within an HTML document to represent an ampersand. Instead, you should use the & entity. Similarly, because the left bracket (<) and right bracket (>) are used to identify HTML tags, you shouldn't use those characters within an HTML document to represent less-than and greater-than signs. Instead, you should use the < and > entities.

Besides the entities for characters, you may sometimes need to insert a non-breaking space to force a browser to display a space. To do that, you use the   character entity. In the third paragraph in this figure, for example, this character entity is used to indent the first line of the paragraph four spaces. However, because you can accomplish the same thing with CSS, you probably won't use the   character entity this way.

Common HTML character entities

Entity	Character	Entity	Character
`&`	&	`&deg;`	°
`<`	<	`&plusmn;`	±
`>`	>	`&lsquo;`	‘ (opening single quote)
`&copy;`	©	`&rsquo;`	’ (closing single quote or apostrophe)
`&reg;`	®	`&ldquo;`	“ (opening double quote)
`&trade;`	™	`&rdquo;`	” (closing double quote)
`&cent;`	¢	` `	non-breaking space

Examples of character entities

```
<p>It&rsquo;s time to start your Christmas shopping!</p>

<p>President John F. Kennedy said, &ldquo;And so, my fellow Americans,
ask not what your country can do for you; ask what you can do for
your country.&rdquo;</p>

<p>    Turning fear into hope, medical futurist
Dr. Alan J. Russell will discuss the science of regenerating damaged
or diseased human body parts, while offering real hope for the future
of human health.</p>

<p>&copy; 2018 Mike Murach &amp; Associates, Inc.</p>
```

The character entities in a web browser

It's time to start your Christmas shopping!

President John F. Kennedy said, "And so, my fellow Americans, ask not what your country can do for you; ask what you can do for your country."

Turning fear into hope, medical futurist Dr. Alan J. Russell will discuss the science of regenerating damaged or diseased human body parts, while offering real hope for the future of human health.

© 2018 Mike Murach & Associates, Inc.

Description

- *Character entities* can be used to display special characters in an HTML document.
- HTML5 provides a variety of character entities in addition to the ones above. For a complete list of character entities, see **https://www.w3.org/TR/html5/syntax.html#named-character-references**

Figure 3-6 How to code character entities

How to code the core attributes

Besides the attributes you've learned about so far, HTML provides some core attributes that you can use with most elements. In figure 3-7, you can see the core attributes that you're most likely to use.

You use the id attribute to uniquely identify an HTML element. Then, you can use CSS to work with the element.

The class attribute is similar, except it doesn't have to be unique. That lets you assign more than one element to the same class. Then, you can use CSS to apply the same formatting to all the elements in the class. When you use the class attribute, you can also assign more than one class to a single element.

In the example in this figure, you can see that the first input element has an id attribute with the value "email". As you'll see in the next chapter, CSS can be used to apply unique formatting to elements with id attributes.

In this example, you can also see that a class attribute is used to assign a class named "first" to the first <p> element. Then, another class attribute is used to assign two classes to the third <p> element. These classes are "first" and "field". Later, you can use CSS to apply formatting to the elements in each class.

You can use the title attribute to provide additional information for an element. In the example in this figure, this attribute is used to provide a tooltip for an input field that lets the user enter an email address (you'll learn more about input fields in chapter 13). Then, when the cursor is moved over this field in the browser, the tooltip is displayed. For some elements, though, a tooltip isn't displayed so the title attribute has no purpose.

The lang attribute lets you specify the language used by a document. For instance, we recommend that you code this attribute on the html element to specify the language for the entire document. For English-speaking countries, you typically code this attribute like this:

```
<html lang="en">
```

This can help a screen reader pronounce words correctly. However, you can also code this attribute for individual elements. If a sentence on a web page is written in French, for example, you can code this attribute for the element that contains the sentence.

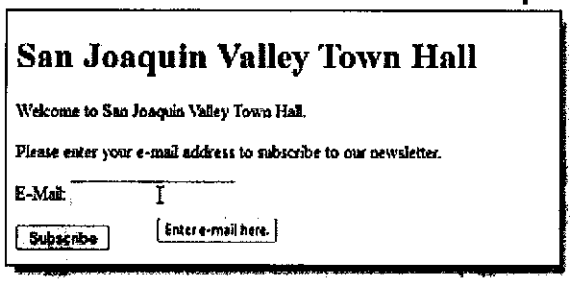
Core HTML attributes

Attribute	Description
`id`	Specifies a unique identifier for an element that can be referred to by CSS.
`class`	Specifies one or more class names that can be referred to by CSS, and the same name can be used for more than one element. To code more than one class name, separate the class names with spaces.
`title`	Specifies additional information about an element. For some elements, the title appears in a tooltip when the user hovers the mouse over the element.
`lang`	Identifies the language that the content of the element is written in.

HTML that uses these attributes

```
<html lang="en">
<body>
    <h1>San Joaquin Valley Town Hall</h1>
    <p class="first">Welcome to San Joaquin Valley Town Hall.</p>
    <form action="subscribe.php" method="post">
        <p>Please enter your e-mail address to subscribe to our
           newsletter.</p>
        <p class="first field">E-Mail:
        <input type="text" name="email" id="email"
               title="Enter e-mail here."></p>
        <p><input type="submit" value="Subscribe"></p>
    </form>
</body>
</html>
```

The HTML in a web browser with a tooltip displayed



Accessibility guideline

- Always code the lang attribute on the html element to identify the language for the page.

Description

- The core attributes can be coded for most HTML elements.
- ID and class names are case sensitive, should start with a letter, and can include letters, numbers, underscores, hyphens, colons, and periods.
- The lang attribute is typically used to assist screen readers to read content correctly and to provide for searches that are restricted by language.

Figure 3-7 How to code the core attributes

How to structure the content of a page

One of the key features of HTML5 is the addition of the new structural elements. After I present these elements, I'll show you how pages were traditionally structured before the release of HTML5.

How to use the primary HTML5 semantic elements

Figure 3-8 presents the HTML5 *semantic elements* that improve the way you can structure a page. For instance, the example in this figure shows how the header, main, and footer elements can be used to divide a document into three parts. This makes it easy to see the structure of the page by looking at the HTML tags. This also makes it easy to code the selectors that you need for formatting these elements.

All of the HTML5 semantic elements are block elements. Unlike the block elements you saw earlier in this chapter for coding text, though, you can nest other block elements within the HTML5 semantic elements. In the HTML code in this figure, for example, you can see that an h1 element is nested within the header element, and a <p> element is nested within both the main and footer elements.

The use of these structural elements is often referred to as *HTML5 semantics*. The implication is that you do a better job of creating meaning when you use these elements. In the long run, using HTML5 semantics will mean that search engines will be able to do a better job of coming up with relevant pages. For that reason, you should start using the semantic elements right away.

Although this figure only illustrates the use of three of the seven elements in the table, you'll see the nav element used in the web page at the end of this chapter. Then, in chapter 6, you'll see how the aside element is used for a sidebar within the main element, and you'll see how the section element is used for other content within the main element. In addition, you'll see how the article element is used for an article about a speaker.

If you review the HTML5 semantic elements in this figure, you'll see that the section element is the only one that doesn't indicate the specific type of content it contains. Because of that, we'll use section elements in this book whenever there isn't another appropriate HTML5 element for given content. The main point of HTML5 semantics, though, is to use the other HTML5 elements whenever they provide the appropriate meaning. That way, you'll get the benefits of HTML5.

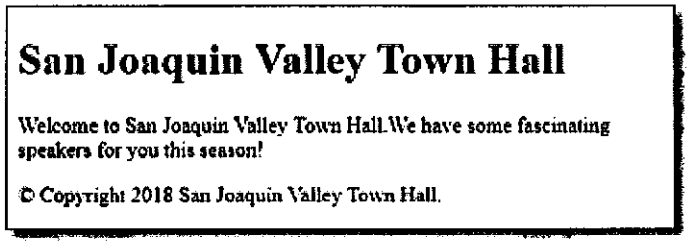
The primary HTML5 semantic elements

Element	Contents
`header`	The header for a page.
`main`	The main content for a page.
`section`	A generic section of a document that doesn't indicate the type of content.
`article`	A composition like an article in the paper.
`nav`	A section of a page that contains links to other pages or placeholders.
`aside`	A section of a page like a sidebar that is related to the content that's near it.
`footer`	The footer for a page.

A page that's structured with header, main, and footer elements

```
<body>
    <header>
        <h1>San Joaquin Valley Town Hall</h1>
    </header>
    <main>
        <p>Welcome to San Joaquin Valley Town Hall. We have some
           fascinating speakers for you this season!</p>
    </main>
    <footer>
        <p>&copy; Copyright 2018 San Joaquin Valley Town Hall.</p>
    </footer>
</body>
```

The page displayed in a web browser



Accessibility and SEO guideline

- Use the HTML5 semantic elements to indicate the structure of your pages.

Description

- HTML5 provides *semantic elements* that you should use to structure the contents of a web page. The use of these elements can be referred to as *HTML5 semantics*.
- All of the HTML5 semantic elements are block elements that can contain other block elements as well as inline elements.
- All of the HTML5 elements in this figure are supported by the modern browsers.

Figure 3-8 How to use the primary HTML5 semantic elements

How to use some of the other HTML5 semantic elements

Figure 3-9 presents some of the other HTML5 semantic elements that are currently supported by modern browsers. For instance, the time element provides a date or date and time in a standard format that can be parsed by a browser. This is illustrated by the first example. Here, the <p> element says that next year's conference will be on May 31st. But the time element makes it clear that this is May 31st of 2018. If the time element is used within an article, you can also use the pubdate attribute to make it clear that the date is the publication date for the article.

The second and third semantic elements in this figure apply to the use of figures within web pages. A figure can be a block of text, an image, a diagram, or anything that is referred to from the text outside of the figure. Within the figure element that contains the figure, the figcaption element can be used to provide a caption for the figure. This is illustrated by the second example, which treats two lines of JavaScript code as a figure. Then, in chapters 11 and 12, you can see how these elements can be used to treat images and tables as figures.

Here again, these elements are designed to provide more meaning. That way, search engines can do a better job of ranking your pages and screen readers can do a better job of reading your pages. In addition, using these elements makes it easier for you to code and format your pages.

Other HTML5 semantic elements

Element	Contents
`time`	A date or date and time that can be parsed by a browser.
`figure`	An illustration, diagram, photo, code listing or the like that is referred to from the main content of the document.
`figcaption`	The caption that identifies a figure.

The attributes of the time element

Attribute	Description
`datetime`	A date and time in a standard format that can be parsed by a browser.
`pubdate`	A Boolean attribute that indicates that the date is the publication date for the article that contains the time element.

A time element

```
<p>Next year's conference will be on
    <time datetime="2018-05-31">May 31st</time>.</p>
```

The figure and figcaption elements

```
<figure>
    <code>
        var today = new Date();<br>
        document.writeln( today.getFullYear() );<br><br>
    </code>
    <figcaption>
        JavaScript code for getting the year
    </figcaption>
</figure>
```

The code displayed in a browser

```
var today = new Date();
document.writeln( today.getFullYear() );
```

JavaScript code for getting the year

Accessibility and SEO guideline

- Use the HTML5 semantic elements to indicate the structure of your pages.

Description

- Although there are other HTML5 semantic elements, these are the most useful ones that are currently supported by modern browsers.

Figure 3-9 How to use some of the other HTML5 semantic elements

When and how to code div and span elements

Figure 3-10 shows how to code the div and span elements that have traditionally been used to structure a page and to format portions of inline content. You need to know how these elements are used because you're sure to see them in the HTML for pages that haven't yet been converted to HTML5. For new pages, though, you should use the HTML5 semantic elements.

The div element is a block element that you can use to divide an HTML document into divisions. In the example in this figure, you can see that the content of the document is divided into three divisions. The first division contains the header for the page, the second division contains the main content of the page, and the third division is for the footer. If you look at this web page as it's displayed in the browser, you can see that these div elements don't affect the appearance of the page.

For each div element, the id attribute is used to indicate the contents of the division. As you'll see in the next chapter, this id can be used as the selector that's used to apply CSS formatting to each division. If you refer back to figure 3-8, though, you'll see how this can be done more easily by using the HTML5 structural tags.

Does this mean that you shouldn't ever use div elements? The quick answer is, no. In general, you should only use div elements as a last resort, but there will be situations where it doesn't make sense to use a section element to define a grouping of elements. Just keep in mind that div elements should be used sparingly and only when an HTML5 semantic element doesn't apply.

The other element that's presented in this figure is the span element. This inline element has traditionally been used to identify content so CSS can be used to format it. For instance, the <p> element in the main division in this figure contains an inline span element. Here again, the span element doesn't affect the appearance of the page, but its id attribute can be used as the selector for CSS formatting.

For modern websites, span elements should only be used when more specific tags don't apply. Instead of span elements, you should use the block elements of figure 3-4 and the inline elements of figure 3-5 to identify the content types. By using these elements instead of span elements, you not only improve accessibility but also SEO.

A block element for structuring a web page

Element	Description
`div`	Lets you divide a page into divisions that can be formatted and positioned with CSS.

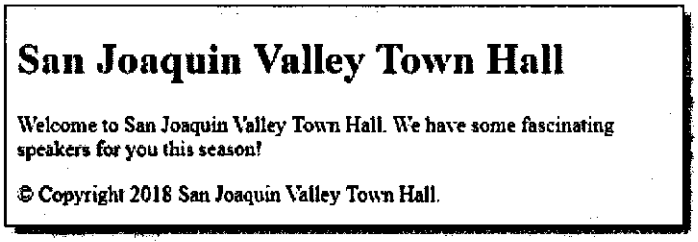
An inline element for formatting text

Element	Description
`span`	Lets you identify text that can be formatted with CSS.

A page that's structured with div and span elements

```
<body>
    <div id="header">
        <h1>San Joaquin Valley Town Hall</h1>
    </div>
    <div id="main">
        <p><span id="welcome">Welcome to San Joaquin Valley Town
            Hall.</span>
            We have some fascinating speakers for you this season!</p>
    </div>
    <div id="footer">
        <p>&copy; Copyright 2018 San Joaquin Valley Town Hall.</p>
    </div>
</body>
```

The page displayed in a web browser



Accessibility and SEO guidelines

- Use div tags only when the HTML5 semantic elements don't apply.
- Use span tags only when the tags for identifying content don't apply.

Description

- Before HTML5, div elements were used to define divisions within the body of a document. Now, the HTML5 semantic elements are replacing div elements.
- Before HTML5, span elements were used to identify portions of text that you could apply formatting to. Today, a better practice is to use the elements in figures 3-4 and 3-5 to identify content and to use CSS to format that content.

Figure 3-10 When and how to code the div and span elements

How to code links, lists, and images

Because you'll use links, lists, and images in most of the web pages that you develop, the topics that follow introduce you to these elements. But first, you need to know how to code absolute and relative URLs so you can use them with your links and images.

How to code absolute and relative URLs

Figure 3-11 presents some examples of absolute and relative URLs. To help you understand how these examples work, the diagram at the top of this figure shows the folder structure for the website used in the examples. As you can see, the folders for this website are organized into three levels. The root folder for the site contains five subfolders, including the folders that contain the images and styles for the site. Then, the books folder contains subfolders of its own.

In chapter 1, you learned about the basic components of a URL. The URL you saw in that chapter was an *absolute URL*, which includes the domain name of the website. This is illustrated by the first group of examples in this figure. Here, both URLs refer to pages at www.murach.com. The first URL points to the index.html file in the root folder of this website, and the second URL points to the toc.html file in the root/books/php folder.

When used within the code for the web pages of a site, an absolute URL is used to refer to a file in another website. In contrast, a *relative URL* is used to refer to a file within the same website. As this figure shows, there are two types of relative URLs.

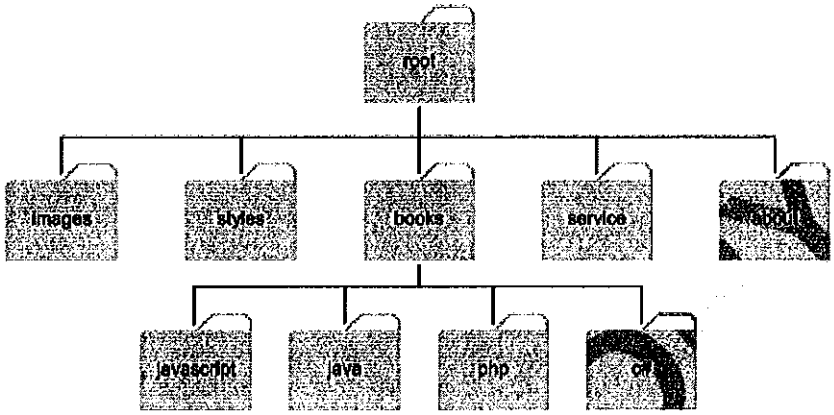
In a *root-relative path*, the path is relative to the root folder of the website. This is illustrated by the second group of examples. Here, the leading slash indicates the root folder for the site. As a result, the first path refers to the login.html file in the root folder, and the second path refers to the logo.gif file in the images folder.

In a *document-relative path*, the path is relative to the current document. This is illustrated by the third group of examples. Here, the assumption is that the paths are coded in a file that is in the root folder for a website. Then, the first path refers to a file in the images subfolder of the root folder, and the second path refers to a file in the php subfolder of the books subfolder. This illustrates paths that navigate down the levels of the folder structure.

But you can also navigate up the levels with a document-relative path. This is illustrated by the fourth group of examples. Here, the assumption is that the current document is in the root/books folder. Then, the first path goes up one level for the index.html file in the root folder. The second path also goes up one level to the root folder and then down one level for the logo.gif file in the images folder.

This shows that there's more than one way to code the path for a file. If, for example, you're coding an HTML file in the root/books folder, you can use either a root-relative path or a document-relative path to get to the images subfolder. If this is confusing right now, you'll quickly get used to it once you start coding your own pages.

A simple website folder structure



Examples of absolute and relative URLs

Absolute URLs

```
http://www.murach.com/index.html
http://www.murach.com/books/php/toc.html
```

Root-relative paths

```
/login.html                (refers to root/login.html)
/images/logo.gif           (refers to root/images/logo.gif)
```

Document-relative paths that navigate down from the root folder

```
images/logo.gif            (refers to root/images/logo.gif)
books/php/overview.html    (refers to root/books/php/overview.html)
```

Document-relative paths that navigate up from the root/books folder

```
../index.html              (refers to root/index.html)
../images/logo.gif         (refers to root/images/logo.gif)
```

Description

- When you code an *absolute URL*, you code the complete URL including the domain name for the site. Absolute URLs let you display pages at other websites.
- When you code a *relative URL*, you base it on the current folder, which is the folder that contains the current page.
- A *root-relative path* is relative to the root folder of the website. It always starts with a slash. Then, to go down one subfolder, you code the subfolder name and a slash. To go down two subfolders, you code a second subfolder name and another slash. And so on.
- A *document-relative path* is relative to the folder the current document is in. Then, to go down one subfolder, you code the subfolder name followed by a slash. To go down two subfolders, you code a second subfolder name followed by another slash. And so on.
- You can also go up in a document-relative path. To go up one level from the current folder, you code two periods and a slash. To go up two levels, you code two periods and a slash followed by two more periods and a slash. And so on.

Figure 3-11 How to code absolute and relative URLs

How to code links

Most web pages contain *links* that go to other web pages or web resources. To code a link, you use the <a> element (or anchor element) as shown in figure 3-12. Because this element is an inline element, you usually code it within a block element like a <p> element.

In most cases, you'll code only the href attribute for the <a> element. This attribute specifies the URL for the resource you want to link to. The examples in this figure illustrate how this works.

The first example uses a relative URL to link to a page in the same folder as the current page. The second example uses a relative URL to link to a page in a subfolder of the parent folder. The third example uses a relative URL to link to a page based on the root folder. And the last example uses an absolute URL to link to a page at another website.

By default, links are underlined when they're displayed in a browser to indicate that they're clickable. As a result, most web users have been conditioned to associate underlined text with links. Because of that, you should avoid underlining any other text.

When a link is displayed, it has a default color depending on its state. For instance, a link that hasn't been visited is displayed in blue, and a link that has been visited is displayed in purple. Note, however, that you can use CSS as described in the next chapter to change these settings.

When you create a link that contains text, the text should clearly indicate the function of the link. For example, you shouldn't use text like "click here" because it doesn't indicate what the link does. Instead, you should use text like that in the examples in this figure. In short, if you can't tell what a link does by reading its text, you should rewrite the text. This improves the accessibility of your site, and it helps search engines index your site.

Basic attribute of the <a> element

Attribute	Description
`href`	Specifies a relative or absolute URL for a link.

A link to a web page in the same folder

```
<p>Go view our <a href="products.html">product list</a>.</p>
```

A link to a web page in a subfolder of the parent folder

```
<p>Read about the <a href="../company/services.html">services we
provide</a>.</p>
```

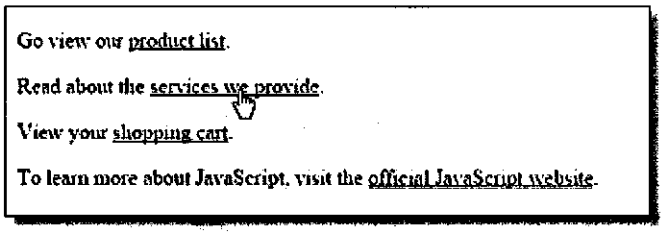
A link to a web page based on the root folder

```
<p>View your <a href="/orders/cart.html">shopping cart</a>.</p>
```

A link to a web page at another website

```
<p>To learn more about JavaScript, visit the
<a href="http://www.javascript.com/">official JavaScript website</a>.</p>
```

The links in a web browser



SEO and accessibility guideline

- The content of a link should be text that clearly indicates where the link is going.

Description

- The <a> element is an inline element that creates a *link* that loads another web page. The href attribute of this element identifies the page to be loaded.
- The text content of a link is underlined by default to indicate that it's clickable.
- If a link hasn't been visited, it's displayed in blue. If it has been visited, it's displayed in purple. You can change these values using CSS.
- If the mouse hovers over a link, the cursor is changed to a hand with the finger pointed as shown above.
- See chapters 4 and 7 for more information on formatting and coding the <a> element.

Figure 3-12 How to code links

How to code lists

Figure 3-13 shows how to code the two basic types of lists: ordered lists and unordered lists. To create an *unordered list*, you use the ul element. Then, within this element, you code one li (list item) element for each item in the list. The content of each li element is the text that's displayed in the list. By default, when a list is displayed in a browser, each item in an unordered list is preceded by a bullet. However, you can change that bullet with CSS.

To create an *ordered list*, you use the ol element, along with one li element for each item in the list. This works like the ul element, except that the items are preceded by numbers rather than bullets when they're displayed in a browser. In this case, you can change the type of numbers that are used with CSS.

The two lists shown in this figure illustrate how this works. Here, the first list simply displays the names of several programming languages, so these items don't need to reflect any order. In contrast, the second list identifies three steps for completing an order. Because these steps must be completed in a prescribed sequence, they're displayed in an ordered list.

When you work with the li element, you should be aware that it can contain text, inline elements, or block elements. For example, an li element can contain an <a> element that defines a link. In fact, it's a best practice to code a series of links within an unordered list. You'll see an example of that later in this chapter, and you'll learn all about this in chapter 7.

Elements that create ordered and unordered lists

Element	Description
`<ul>`	Creates an unordered list.
`<ol>`	Creates an ordered list.
`<li>`	Creates a list item for an unordered or ordered list.

HTML that creates two lists

```
<p>We have books on a variety of languages, including</p>
<ul>
    <li>JavaScript</li>
    <li>PHP</li>
    <li>Java</li>
    <li>C#</li>
</ul>

<p>You will need to complete the following steps:</p>
<ol>
    <li>Enter your billing information.</li>
    <li>Enter your shipping information.</li>
    <li>Confirm your order.</li>
</ol>
```

The lists in a web browser

We have books on a variety of languages, including

- JavaScript
- PHP
- Java
- C#

You will need to complete the following steps:

1. Enter your billing information.
2. Enter your shipping information.
3. Confirm your order.

Description

- The two basic types of lists are *unordered lists* and *ordered lists*. By default, an unordered list is displayed as a bulleted list and an ordered list is displayed as a numbered list.
- See chapters 4 and 7 for more information on formatting and coding lists.

Figure 3-13 How to code lists

How to include images

Images are an important part of most web pages. To display an image, you use the img element shown in figure 3-14. This is an inline element that's coded as an empty tag. In the example in this figure, this tag is coded before an h1 element. The h1 element is displayed below it, though, because it's a block element that starts on a new line.

The src (source) attribute of an img element specifies the URL of the image you want to display, and it is required. For instance, the src attribute for the image in the example indicates that the image named murachlogo.gif can be found in the images subfolder of the current folder.

The alt attribute should also be coded for img elements. You typically use this attribute to provide information about the image in case it can't be displayed or the page is being accessed using a screen reader. This is essential for visually-impaired users.

For instance, because the image in this figure is our company's logo, the value of the alt attribute is set to "Murach Logo". If an image doesn't provide any meaning, however, you can code the value of the alt attribute as an empty string (""). You should do that, for example, when an image is only used for decoration.

The images that you include in a web page need to be in one of the formats that modern web browsers support. Currently, most web browsers support the JPEG, GIF, and PNG formats. Typically, a web designer uses imaging software such as Adobe Photoshop to create and maintain these files for a website. In particular, you use imaging software to create images that are the right size for your web pages.

Then, you can use the height and width attributes of an img element to tell the browser what the size of an image is. That can help the browser lay out the page as the image is being loaded. Although you can also use the height and width attributes to render an image larger (known as "stretching") or smaller than the original image, it's better to use your image editor to make the image the right size. You'll learn more about working with images in chapter 11.

Attributes of the <img> element

Attribute	Description
`src`	Specifies the relative or absolute URL of the image to display. It is a required attribute.
`alt`	Specifies alternate text to display in place of the image. This text is read aloud by screen readers for users with disabilities. It is required.
`height`	Specifies the height of the image in pixels.
`width`	Specifies the width of the image in pixels.

An img element

```
<img src="images/murachlogo.gif" alt="Murach Logo" height="75">
<h1>Mike Murach &amp; Associates, Inc.</h1>
```

The image in a web browser

The image formats that are supported by most browsers

- JPEG (Joint Photographic Experts Group)
- GIF (Graphic Interchange Format)
- PNG (Portable Network Graphics)

Accessibility guidelines

- For images with useful content, always code an alt attribute that describes the image.
- For images that are used for decoration, code the alt attribute with no value ("").

Description

- The img element is an inline element that is used to display an image that's identified by the src attribute.
- The height and width attributes can be used to indicate the size of an image so the browser can allocate the correct amount of space on the page. These attributes can also be used to size an image, but it's usually better to use an image editor to do that.
- JPEG files commonly use the JPG extension and are typically used for photographs and scans. GIF files are typically used for small illustrations and logos. And PNG files combine aspects of JPEG and GIF files.
- See chapters 4 and 11 for more information on formatting and coding img elements.

Figure 3-14 How to include images

A structured web page

Now that you've seen the HTML elements for structuring a web page, you're ready to see a simple web page that uses these elements.

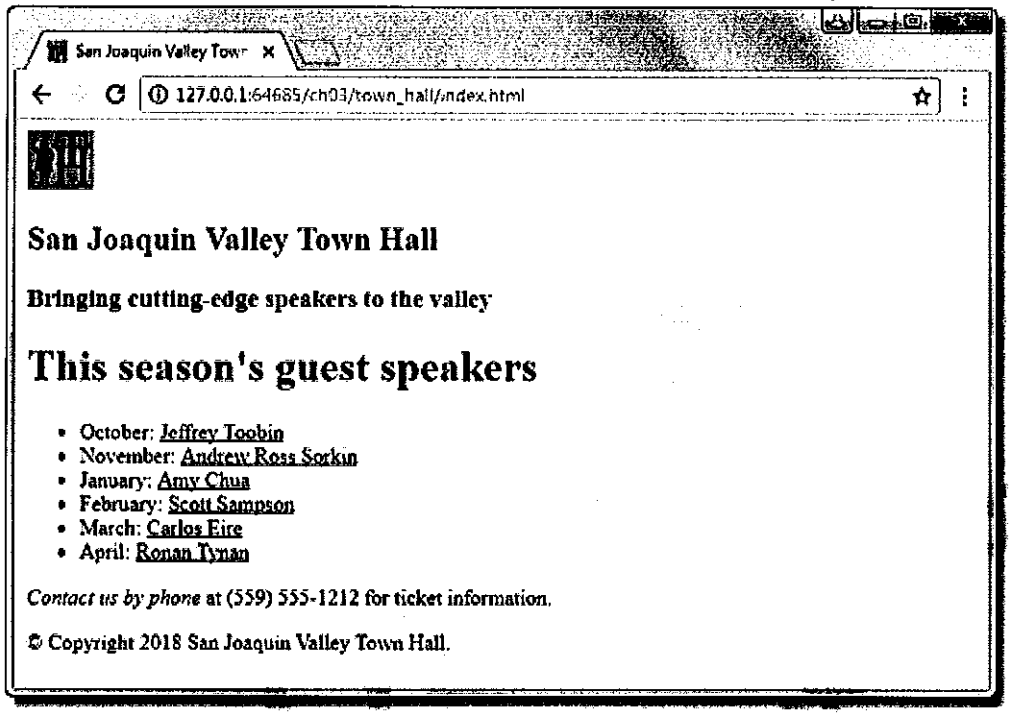
The page layout

The web page shown in figure 3-15 uses many of the HTML elements you learned about in this chapter. The purpose of this web page is to display information about a series of lectures being presented by a non-profit organization.

This figure shows the default formatting for the HTML that's used for this page. In other words, CSS hasn't been used to improve the formatting. In the next chapter, though, you'll learn how to use CSS to improve the formatting of this page.

When you review the HTML code for this page in the next figure, you'll see that the first heading in the header of this page is an h2 element and the second heading is an h3 element. Then, the heading in the main content of the page is an h1 element. That's because this heading identifies the most important information on the page, and this is important for SEO. Then, you can use CSS to format all three headings the way you want them to look in the browser.

A web page that uses some of the HTML presented in this chapter



Description

- This web page provides information for a non-profit organization that arranges lectures by renowned speakers.
- This web page contains HTML5 semantic elements that define a header, the main content, and a footer.
- The header element contains an image and h2 and h3 elements.
- The main element contains an h1 element, a nav element that contains an unordered list that contains <a> elements, and a <p> element.
- The footer element contains a <p> element that uses a character entity to include the copyright symbol.
- This page includes a favicon that's displayed to the left of the page title in the browser's tab.
- This illustrates the default formatting for the HTML elements. In the next chapter, you'll learn how to use CSS to format the HTML so it looks more appealing.

Figure 3-15 The page layout for a structured web page

The HTML file

Figure 3-16 presents the HTML file for the web page. To start, the DOCTYPE declaration, the html element, and head element illustrate the way these items should be coded in every document that you create.

In the html element, you can see the use of the lang attribute. In the head element, you can see the coding for the charset meta element and the title element. Although the link element for the favicon is optional, most websites use one. Beyond that, you should include a meta element for at least the description in most web pages, although it isn't used in the examples in this book.

What's most important, though, is the use of the HTML5 semantic elements. Here, the header element contains one h2 and one h3 element. Then, the main element contains one h1 element, a nav element, and a <p> element. Last, the footer element contains one <p> element.

Within the nav element is an unordered list that contains six li elements. Then, each li element contains an <a> element. This is consistent with HTML5 semantics because a nav element should contain a series of links. It is also a best practice to code the <a> elements within an unordered list.

You can also see the use of an element that italicizes the first four words in the second last paragraph and a character entity that inserts the copyright symbol into the last paragraph. But above all, the new HTML5 semantic elements make the structure of this page obvious.

The HTML file for the web page

```
<!DOCTYPE html>

<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>San Joaquin Valley Town Hall</title>
        <link rel="shortcut icon" href="images/favicon.ico">
    </head>

    <body>
        <header>
            <img src="images/logo.jpg" alt="Town Hall Logo" width="50">
            <h2>San Joaquin Valley Town Hall</h2>
            <h3>Bringing cutting-edge speakers to the valley</h3>
        </header>

        <main>
            <h1>This season's guest speakers</h1>
            <nav>
                <ul>
                    <li>October: <a href="speakers/toobin.html">
                        Jeffrey Toobin</a></li>
                    <li>November: <a href="speakers/sorkin.html">
                        Andrew Ross Sorkin</a></li>
                    <li>January: <a href="speakers/chua.html">
                        Amy Chua</a></li>
                    <li>February: <a href="speakers/sampson.html">
                        Scott Sampson</a></li>
                    <li>March: <a href="speakers/eire.html">
                        Carlos Eire</a></li>
                    <li>April: <a href="speakers/tynan.html">
                        Ronan Tynan</a></li>
                </ul>
            </nav>

            <p><em>Contact us by phone</em> at (559) 555-1212 for ticket
               information.</p>
        </main>

        <footer>
            <p>&copy; Copyright 2018 San Joaquin Valley Town Hall.</p>
        </footer>
    </body>
</html>
```

Figure 3-16 The HTML file for a structured web page

Perspective

This chapter has presented many of the HTML elements that you need for developing web pages. In particular, it has presented the main block and inline elements that you need for creating and identifying the content for a page. It has also presented the HTML5 elements that you need for structuring a page.

With these skills, you can create web pages with the default formatting of the browser. As I've mentioned throughout this chapter, though, the right way to format and lay out your web pages is to use CSS. That's why the next three chapters show you how to do that.

Terms

focus keyword
favicon
metadata
block element
inline element
character entity
semantic elements
HTML5 semantics
absolute URL
relative URL
root-relative path
document-relative path
link
ordered list
unordered list
JPEG (Joint Photographic Experts Group)
GIF (Graphic Interchange Format)
PNG (Portable Network Graphics)

Summary

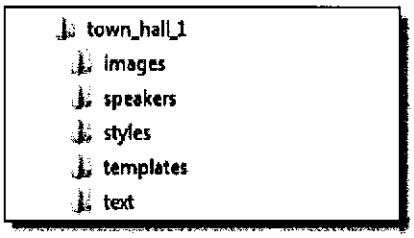
- In the head section of an HTML document, the title element provides the text that's displayed in the browser's title bar. This is an important element for search engine optimization.
- One common use of the link element in the head section is to identify a custom icon called a *favicon* that appears in the browser's tab for the page. This icon may also appear in the browser's address bar or as part of a bookmark.
- The meta elements in the head section provide *metadata* that is related to the page. Here, the charset metadata is required for HTML5 validation, and the description metadata should be coded because it can affect search engine optimization. The keywords metadata can also affect search engine optimization in some browsers.

- *Block elements* are the primary content elements of a website, and each block element starts on a new line when it is rendered by a browser. Headings and paragraphs are common block elements.
- *Inline elements* are coded within block elements, and they don't start on new lines when they are rendered. Some common inline elements like <i> (for italics) and b (for bold) can be used to format text without implying any meaning. Whenever possible, though, you should use the inline elements that imply meaning.
- *Character entities* are used to display special characters like the ampersand and copyright symbols in an HTML document. In code, character entities start with an ampersand and end with a semicolon as in   (a non-breaking space).
- The core attributes that are commonly used for HTML elements are the id, class, and title attributes. The id attribute uniquely identifies one element. The class attribute can be used to identify one or more elements. And the title attribute can provide other information about an element like its tooltip text.
- The HTML5 *semantic elements* are block elements that provide a newer way to structure the content within an HTML document. This makes it easy to code and format elements. In the long run, it will also improve the way search engines rank your pages.
- Historically, the div element has been used to divide the code for an HTML document into divisions, and the span element has been used to identify portions of text so formatting can be applied to them.
- When you code an *absolute URL*, you code the complete URL including the domain name. When you code a *relative URL*, you can use a *root-relative path* to start the path from the root folder for the website or a *document-relative path* to start the path from the current document.
- The <a> element (or anchor element) is an inline element that creates a link that usually loads another page. By default, the text of an <a> element is underlined. Also, an unvisited link is displayed in blue and a visited link in purple.
- Lists are block elements that can be used to display both *unordered lists* and *ordered lists*. By default, these lists are indented with bullets before the items in an unordered list and numbers before the items in an ordered list.
- The img element is used to display an image file. The three common formats for images are *JPEG* (for photographs and scans), *GIF* (for small illustrations and logos), and *PNG*, which combines aspects of JPEG and GIF.

About the exercises

In the exercises for chapters 3 through 8, you'll develop a new version of the Town Hall website. This version will be like the one in the text, but it will have different content, different formatting, and different page layouts. Developing this site will give you plenty of practice, and it will also show you how similar content can be presented in two different ways.

As you develop this site, you will use this folder structure:



This is a realistic structure with images in the images folder, speaker HTML pages in the speakers folder, CSS files in the styles folder, and template files in the templates folder. In addition, the text folder contains text files that will provide all of the content you need for your pages. That too is realistic because a web developer often works with text that has been written by someone else.

Exercise 3-1 Enter the HTML for the home page

In this exercise, you'll code the HTML for the home page. When you're through, the page should look like the one on the facing page, but with three speakers.

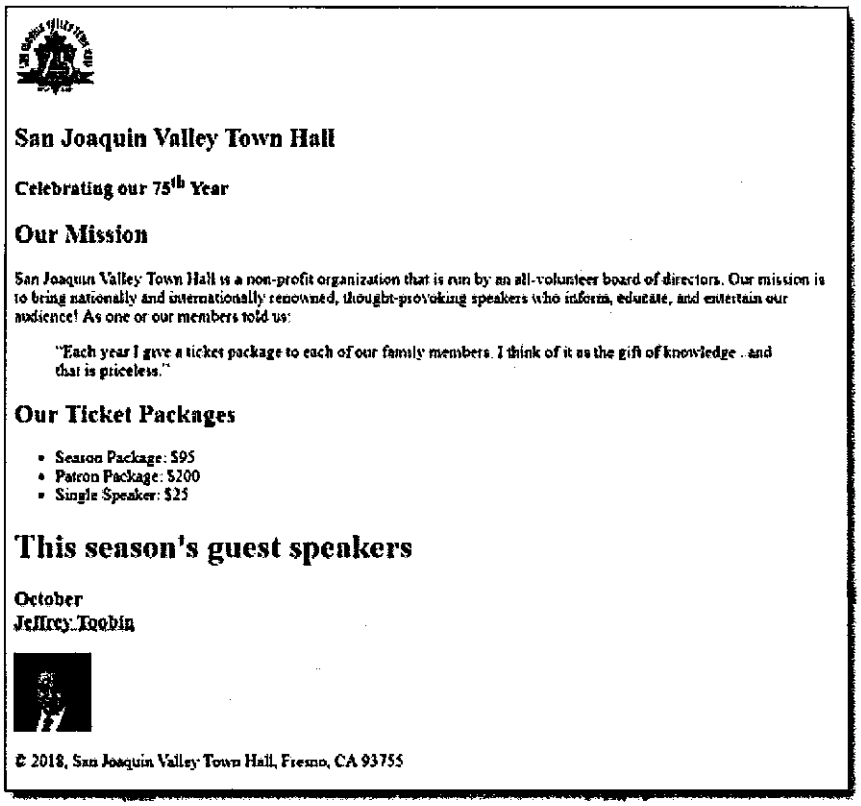
Open the starting page and get the contents for it

1. Use your text editor to open this HTML file:
   `c:\html5_css3_4\exercises\town_hall_1\index.html`
   Note that it contains the head section for this web page as well as a body section that contains header, main, and footer tags.
2. Use your text editor to open this text file:
   `c:\html5_css3_4\exercises\town_hall_1\text\c3_content.txt`
   Note that it includes all of the text that you need for this web page.

Enter the header

3. Code the img element that gets the image at the top of the page from the images directory. To locate the image file, use this document-relative path: images/town_hall_logo.gif. Be sure to include the alt attribute, and set the height attribute of the image to 80.
4. Copy the text for the first two headings from the txt file in the text folder into the header of the HTML file. Next, apply the h2 and h3 elements.
5. Test this page in Chrome. If necessary, correct the HTML and test again.

What the home page should look like



Enter the content for the main element

6. Copy all of the content for the main element from the txt file into the HTML file. Then, add an h1 tag to the heading "This season's guest speakers", and add h2 tags to these headings "Our Mission" and "Our Ticket Packages".
7. Add <p> tags to the first block of text after the "Our Mission" heading, and add blockquote tags to the second block of text as shown above.
8. Add the ul and li tags that are needed for the three items after the "Our Ticket Packages" heading. Then, test these changes and make any adjustments.
9. Format the name and month for the first speaker after the "This season's guest speakers" heading as one h3 element with a <br> in the middle that rolls the speaker's name over to a second line. Then, test and adjust.
10. When that works, do the same for the next two speakers.
11. Enclose the name for each speaker in an <a> tag. The href attribute for each tag should refer to a file in the speakers subfolder that has the speaker's last name as the filename and html as the file extension. In other words, the reference for the first speaker should be:

```
speakers/toobin.html
```

12. After the h3 element for each speaker, code an img element that displays the image for the speaker, and be sure to include the alt attribute. The images are in the images subfolder, and the filename for each is the speaker's last name, followed by 75 (to indicate the image size), with jpg as the extension. So to refer to the first speaker's file, you need to use a document-relative path like this: images/toobin75.jpg. Now, test and adjust.

Enter the footer

13. Copy the last paragraph in the txt file into the footer of the HTML file. Then, enclose the text in a <p> element.

Add character entities and formatting tags

14. Use character entities to add the quotation marks at the start and end of the text in the blockquote element.
15. Use a character entity to add the copyright symbol to the start of the footer.
16. Add the sup tags that you need for raising the *th* in the second line of the header (as in 75th). Then, test these enhancements.

Test the links, validate the HTML, and test in various browsers

17. Click on the link for the first speaker page. This should display a page that gives the speaker's name and says "This page is under construction". If this doesn't work, fix the href attribute in the link and test again. To return to the first page, you can click the browser's Back button.
18. Open the toobin.html file that's in the speakers subfolder. Then, add a link within a <p> element that says "Return to index page." To refer to the index.html file, you'll have to go up one level in the folder structure with a document-relative path like this: ../index.html. Now, test this link.
19. Validate the HTML for the index page as shown in figure 2-14 of chapter 2. This should indicate no warnings or errors. If any errors are detected, fix them and validate the HTML again.
20. Test the index page in Chrome. If necessary, fix any problems and test again.
21. If you have IE or Edge, test the index page in that browser. If necessary, fix any problems and test again in both browsers.