2

How to code, test, and validate a web page

In this chapter, you'll learn how to create and edit HTML and CSS files. Then, you'll learn how to test those files to make sure they work correctly. Last, you'll learn how to validate the code in HTML and CSS files to make sure that it doesn't have any errors. When you're through with this chapter, you'll be ready to learn all the details of HTML and CSS coding.

The HTML syntax

When you code an HTML document, you need to adhere to the rules for creating the HTML elements. These rules are referred to as the syntax of the language. In the four topics that follow, you'll learn the HTML syntax.

The basic structure of an HTML document

Figure 2-1 presents the basic structure of an *HTML document*. As you can see, every HTML document consists of two parts: the DOCTYPE declaration and the document tree.

When you use HTML5, you code the *DOCTYPE declaration* exactly as it's shown in this figure. It will be the first line of code in every HTML document that you create, and it tells the browser that the document is using HTML5. If you've developed web pages with earlier versions of HTML or XHTML, you will be pleased to see how much this declaration has been simplified.

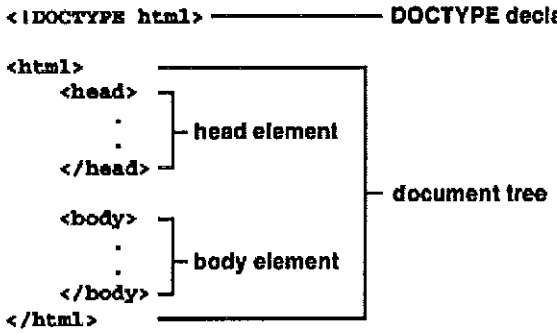
The *document tree* starts right after the DOCTYPE declaration. This tree consists of the *HTML elements* that define the content and structure of the web page. The first of these elements is the html element itself, which contains all of the other elements. This element can be referred to as the *root element* of the tree.

Within the html element, you should always code a head element and a body element. The head element contains elements that provide information about the page itself, while the body element contains the elements that provide the structure and content for the page. You'll learn how to code these elements in the next chapter.

You'll use the elements shown in this figure in every HTML document that you create. As a result, it's a good practice to start every HTML document from a template that contains this code or from another HTML document that's similar to the one you're going to create. Later in this chapter, you'll learn how you can use Brackets to do that.

When you use HTML5, you can code elements using lowercase, uppercase, or mixed case. For consistency, though, we recommend that you use lowercase unless uppercase is required. The one exception we make is in the DOCTYPE declaration because DOCTYPE has historically been capitalized (although lowercase works too). You will see this use of capitalization in all of the examples and applications in this book.

The basic structure of an HTML5 document



A simple HTML5 document

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <title>San Joaquin Valley Town Hall</title>
    </head>
    <body>
        <h1>San Joaquin Valley Town Hall</h1>
        <p>Welcome to San Joaquin Valley Town Hall.</p>
        <p>We have some amazing speakers in store for you this season!</p>
        <p><a href="speakers.html">Speaker information</a></p>
    </body>
</html>
```

General coding recommendation for HTML5

- Although you can code the HTML using lowercase, uppercase, or mixed case, we recommend that you do all coding in lowercase because it's easier to read.

Description

- An *HTML document* contains *HTML elements* that define the content and structure of a web page.
- Each HTML5 document consists of two parts: the DOCTYPE declaration and the document tree.
- The *DOCTYPE declaration* shown above indicates that the document is going to use HTML5. You'll code this declaration at the start of every HTML document.
- The *document tree* starts with the html element, which marks the beginning and end of the HTML code. This element can be referred to as the *root element* of the document.
- The html element always contains one head element that provides information about the document and one body element that provides the structure and content of the document.

Figure 2-1 The basic structure of an HTML document

How to code elements and tags

Figure 2-2 shows you how to code elements and tags. As you have already seen, most HTML elements start with an *opening tag* and end with a *closing tag* that is like the opening tag but has a slash within it. Thus, <h1> is the opening tag for a level-1 heading, and </h1> is the closing tag. Between those tags, you code the *content* of the element.

Some elements, however, have no content or closing tag. These tags are referred to as *empty tags*. For instance, the
 tag is an empty tag that starts a new line, and the <img> tag is an empty tag that identifies an image that should be displayed.

The third set of examples in this figure shows the right way and the wrong way to code tags when one element is *nested* within another. In short, the tags for one element shouldn't overlap with the tags for another element. That is, you can't close the outer element before you close the inner element.

From this point on in this book, we will refer to elements by the code used in the opening tag. For instance, we will refer to head elements, h1 elements, and img elements. To prevent misreading, though, we will enclose single-letter element names in brackets. As a result, we will refer to <a> elements and <p> elements. We will also use brackets wherever else we think they will help prevent misreading.

Two elements with opening and closing tags

```
<h1>San Joaquin Valley Town Hall</h1>
<p>Here is a list of links:</p>
```

Two empty tags

```
<br>
<img src="logo.gif" alt="Murach Logo">
```

Correct and incorrect nesting of tags

Correct nesting

```
<p>Order your copy <i>today!</i></p>
```

Incorrect nesting

```
<p>Order your copy <i>today!</p></i>
```

Description

- Most HTML elements have an opening tag, content, and a closing tag. Each tag is coded within a set of brackets (<>).
- An element's *opening tag* includes the tag name. The closing tag includes the tag name preceded by a slash. And the *content* includes everything that appears between the opening and closing tags.
- Some HTML elements have no content. For example, the
 element, which forces a line break, consists of just one tag. This type of tag is called an *empty tag*.
- HTML elements are commonly *nested*. To nest elements correctly, though, you must close an inner set of tags before closing the outer set of tags.

Figure 2-2 How to code elements and tags

How to code attributes

Figure 2-3 shows how to code the *attributes* for an HTML element. These attributes are coded within the opening tag of an element or within an empty tag. For each attribute, you code the attribute name, an equal sign, and the attribute value.

When you use HTML5, the attribute value doesn't have to be coded within quotation marks unless the value contains a space, but we recommend that you use quotation marks to enclose all values. Also, although you can use either double or single quotes, we recommend that you always use double quotes. That way, your code will have a consistent appearance that will help you avoid coding errors.

In the examples in this figure, you can see how one or more attributes can be coded. For instance, the second example is an opening tag with three attributes. In contrast, the third example is an empty img element that contains a src attribute that gives the name of the image file that should be displayed plus an alt attribute that gives the text that should be displayed if the image file can't be found.

The next example illustrates the use of a *Boolean attribute*. A Boolean attribute can have just two values, which represent either on or off. To turn a Boolean attribute on, you code just the name of the attribute. In this example, the checked attribute turns that attribute on, which causes the related check box to be checked when it is rendered by the browser. If you want the attribute to be off when the page is rendered, you don't code the attribute.

The next set of examples illustrates the use of two attributes that are commonly used to identify HTML elements. The id attribute is used to uniquely identify just one element, so each id attribute must have a unique value. In contrast, the class attribute can be used to mark one or more elements, so the same value can be used for more than one class attribute. You'll see these attributes in a complete example in figure 2-6.

How to code an opening tag with attributes

An opening tag with one attribute

```
<a href="contact.html">
```

An opening tag with three attributes

```
<a href="contact.html" title="Click to Contact Us" class="nav_link">
```

How to code an empty tag with attributes

```
<img src="logo.gif" alt="Murach Logo">
```

How to code a Boolean attribute

```
<input type="checkbox" name="mailList" checked>
```

Two common attributes for identifying HTML elements

An opening tag with an id attribute

```
<div id="page">
```

An opening tag with a class attribute

```
<a href="contact.html" title="Click to Contact Us" class="nav_link">
```

Coding rules

- An attribute consists of the attribute name, an equal sign (=), and the value for the attribute.
- Attribute values don't have to be enclosed in quotes if they don't contain spaces.
- Attribute values must be enclosed in single or double quotes if they contain one or more spaces, but you can't mix the type of quotation mark used for a single value.
- Boolean attributes can be coded as just the attribute name. They don't have to include the equal sign and a value that's the same as the attribute name.
- To code multiple attributes, separate each attribute with a space.

Our coding recommendation

- For consistency, enclose all attribute values in double quotes.

Description

- *Attributes* can be coded within opening or empty tags to supply optional values.
- A *Boolean attribute* represents either an on or off value.
- The id attribute is used to identify a single HTML element, so its value can be used for just one HTML element.
- A class attribute with the same value can be used for more than one HTML element.

Figure 2-3 How to code attributes

How to code comments and whitespace

Figure 2-4 shows you how to code *comments.* Here, the starting and ending characters for two comments are highlighted. Then, everything within those characters, also highlighted here, is ignored when the page is rendered.

One common use of comments is to describe or explain portions of code. That is illustrated by the first comment.

Another common use of comments is to *comment out* a portion of the code. This is illustrated by the second comment. This is useful when you're testing a web page and you want to temporarily disable a portion of code that you're having trouble with. Then, after you test the rest of the code, you can remove the comments and test that portion of the code.

This figure also illustrates the use of *whitespace*, which consists of characters like tab characters, return characters, and extra spaces. For instance, the return character after the opening body tag and all of the spaces between that tag and the next tag are whitespace.

Since whitespace is ignored when an HTML document is rendered, you can use the whitespace characters to format your HTML so it is easier to read. In this figure, for example, you can see how whitespace has been used to indent and align the HTML elements.

That of course is a good coding practice, and you'll see that in all of the examples in this book. Note, however, that the code will work the same if all of the whitespace is removed. In fact, you could code all of the HTML for a document in a single line.

Although whitespace doesn't affect the way an HTML document is rendered, it does take up space in the HTML file. As a result, you shouldn't overdo your use of it. Just use enough to make your code easy to read.

An HTML document with comments and whitespace

```
<!DOCTYPE html>
<!--
    This document displays the home page
    for the website.
-->

<html>
    <head>
        <title>San Joaquin Valley Town Hall</title>
    </head>

    <body>
        <h1>San Joaquin Valley Town Hall</h1>
        <h2>Bringing cutting-edge speakers to the valley</h2>
        <!-- This comments out all of the HTML code in the unordered list
            <ul>
                <li>October: Jeffrey Toobin</li>
                <li>November: Andrew Ross Sorkin</li>
                <li>January: Amy Chua</li>
                <li>February: Scott Sampson</li>
                <li>March: Carlos Eire</li>
                <li>April: Ronan Farrow</li>
            </ul>
        The code after the end of this comment is active -->
        <p>Contact us by phone at (559) 444-2180 for ticket information.</p>
    </body>
</html>
```

Our coding recommendations

- Use whitespace to indent lines of code and make them easier to read.
- Don't overdo your use of whitespace, because it does add to the size of the file.

Description

- An HTML *comment* is text that appears between the <!-- and --> characters. Since web browsers ignore comments, you can use them to describe or explain portions of your HTML code that might otherwise be confusing.
- You can also use comments to *comment out* elements that you don't want the browser to display. This can be useful when you're testing a web page.
- An HTML comment can be coded on a single line or it can span two or more lines.
- *Whitespace* consists of characters like tab characters, line return characters, and extra spaces.
- Since whitespace is ignored by browsers, you can use it to indent lines of code and separate elements from one another by putting them on separate lines. This is a good coding practice because it makes your code easier to read.

Figure 2-4 How to code comments and whitespace

The CSS syntax

Like HTML, CSS has a syntax that must be adhered to when you create a CSS file. This syntax is presented next.

How to code CSS style rules and comments

A CSS file consists of *style rules*. As the diagram in figure 2-5 shows, a style rule consists of a *selector* followed by a set of braces. Within the braces are one or more *declarations*, and each declaration consists of a *property* and a *value*. Note that the property is followed by a colon and the value is followed by a semicolon.

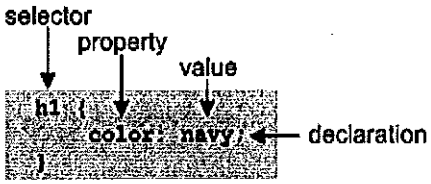
In this diagram, the selector is h1 so it applies to all h1 elements. Then, the style rule consists of a single property named color that is set to the color navy. The result is that the content of all h1 elements will be displayed in navy blue.

In the CSS code that follows, you can see four other style rules. Three of these contain only one declaration, but the third example is a style rule that consists of two declarations: one for the font-style property, and one for the border-bottom property.

Within a CSS file, you can also code comments that describe or explain what the CSS code is doing. For each comment, you start with /* and end with */, and anything between those characters is ignored. In the example in this figure, you can see how CSS comments can be coded on separate lines or after the lines that make up a style rule.

You can also use comments to comment out portions of code that you want disabled. This can be useful when you're testing your CSS code just as it is when you're testing your HTML code.

The parts of a CSS style rule



A simple CSS document with comments

```
/*************************************************************
* Description: Primary style sheet for valleytownhall.com
* Author:      Anne Boehm
*************************************************************/
/* Adjust the styles for the body */
body {
    background-color: #FACD8A;          /* This is a shade of orange. */
}

/* Adjust the styles for the headings */
h1 {
    color: #363636;
}
h2 {
    font-style: italic;
    border-bottom: 3px solid #EF9C00;  /* Adds a line below h2 headings */
}

/* Adjust the styles for the unordered list */
ul {
    list-style-type: square;            /* Changes the bullets to squares */
}
```

Description

- A CSS *style rule* consists of a selector and zero or more declarations enclosed in braces.
- A CSS *selector* consists of the identifiers that are coded at the beginning of the style rule.
- A CSS *declaration* consists of a *property*, a colon, a *value*, and a semicolon.
- To make your code easier to read, you can use spaces, indentation, and blank lines within a style rule.
- CSS *comments* begin with the characters /* and end with the characters */. A CSS comment can be coded on a single line, or it can span multiple lines.

Figure 2-5 How to code CSS style rules and comments

How to code basic selectors

The selector of a style rule identifies the HTML element or elements that the declarations should be applied to. To give you a better idea of how this works, figure 2-6 shows how to use the three basic selectors for CSS style rules. Then, in chapter 4, you'll learn how to code all types of selectors.

The first type of selector identifies HTML elements like body, h1, or <p> elements. For instance, the selectors in the first two examples apply to the body and h1 elements. These selectors are called *type selectors.*

The second type of selector starts with the hash character (#) and applies to the single HTML element that's identified by the id attribute. For instance, #copyright applies to the HTML element that has an id attribute with a value of copyright. As you can see, that's the last <p> element in the HTML code.

The third type of selector starts with a period (.) and applies to all of the HTML elements that are identified by the class attribute with the named value. For instance, .base_color applies to all elements with class attributes that have a value of base_color. In the HTML code, this includes the h1 element and the last <p> element.

Starting with chapter 4, you'll learn all of the coding details for style rules. But to give you an idea of what's going on in this example, here's a quick review of the code.

In the style rule for the body element, the font-family is set either to Arial (if the browser has access to that font) or the sans-serif type that is the default for the browser. This font is then used for all text that's displayed within the body element, unless it's overridden later on by some other style rule. So in this example, all of the text will be Arial or sans-serif, and you can see that font in the browser display.

The style rule for the body element also sets the font-size to 100% of the default size. Although this is the default, this selector is often coded for completeness. Next, the width of the body is set to 300 pixels, and the padding between the contents and the border is set to 1 em, which is the height of the default font.

In the style rule for the h1 element, the font-size is set to 180% of the default font size for the document (the size set by the selector for the body element). Then, in the style rule for the second <p> element (#copyright), the font size is set to 75% of the default font size, and the text is right-aligned. Here again, you can see how these style rules are applied in the browser display.

Last, in the style rule for the class named base_color, the color is set to blue. This means that both of the HTML elements that have that class name (the h1 element and the second <p> element) are displayed in blue.

This example shows how easy it is to identify the elements that you want to apply CSS formatting to. This also shows how the use of CSS separates the formatting from the content and structure that is defined by the HTML.

HTML elements that can be selected by element type, id, or class

```
<body>
    <h1 class="base_color">Student materials</h1>
    <p>Here are the links for the downloads:</p>
    <ul id="links">
        <li><a href="exercises.html">Exercises</a></li>
        <li><a href="solutions.html">Solutions</a></li>
    </ul>
    <p id="copyright" class="base_color">Copyright 2018</p>
</body>
```

CSS style rules that select by element type, id, and class

Type

```
body {
    font-family: Arial, sans-serif;
    font-size: 100%;
    width: 300px;
    padding: 1em;
}
h1 {
    font-size: 180%;
}
```

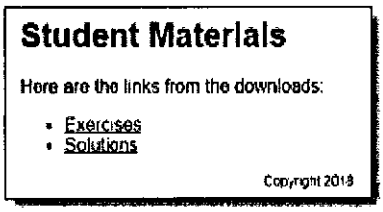
ID

```
#copyright {
    font-size: 75%;
    text-align: right;
}
```

Class

```
.base_color {
    color: blue;
}
```

The elements in a browser



Description

- To code a selector for an HTML element, you simply name the element. This is referred to as a *type selector.*
- If an element is coded with an id attribute, you can code a selector for that id by coding a pound sign (#) followed by the id value, as in #copyright.
- If an element is coded with a class attribute, you can code a selector for that class by coding a period followed by the class name, as in .base_color.

Figure 2-6 How to code basic selectors

How to use Brackets to work with HTML and CSS files

In chapter 1, you were introduced to Brackets, which is a text editor that was created by Adobe. This is the text editor that we recommend for this book because it's free; it runs on Windows, Mac OS, and Linux; and it has some excellent features. In the topics that follow, you'll learn how to use Brackets for building web pages.

If you're going to use a different editor as you work with the applications for this book, you may still want to browse these topics because they will give you a good idea of what an editor should be able to do. They may also encourage you to give Brackets a try.

How to open and close the folder for a website

To work with a web application in Brackets, you start by opening the folder that contains all of the subfolders and files for the website. To do that, you can use one of the two techniques in figure 2-7. After you open a folder for a website, Brackets considers it to be the current *project*, and its subfolders are displayed in a *file tree* in the left pane beneath the name of the project folder.

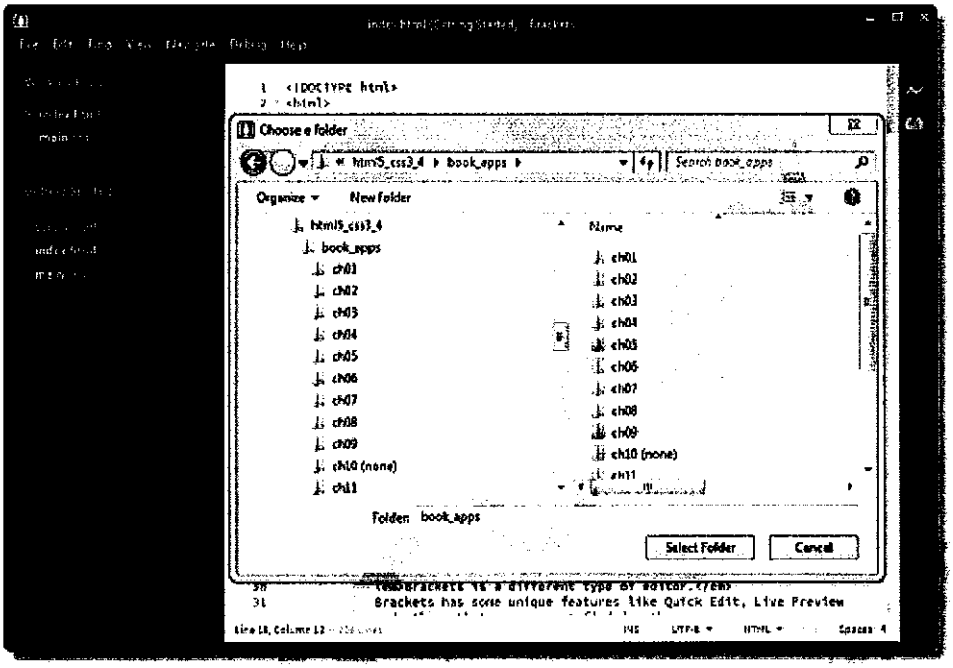
In this example, the current project is Getting Started, which introduces some of the features of Brackets. This project is displayed the first time you start Brackets. Beneath the project name, you can see a folder named screenshots, a file named index.html, and a file named main.css. After you learn how to preview an HTML file, you may want to preview the index.html page. You may also want to review the code in the HTML and CSS files.

When you click on the name of the current project, which in this example is Getting Started, a list of recently opened folders is displayed. Then, you can select the project that you want to work with from that list. Or, you can click on the Open Folder command at the top of the list to open another project. Either way, when another project is opened, it becomes the current project and the previous project is closed.

To make it easier to work with the applications for this book, we recommend that you use Brackets to open the folder that contains all of them. This is illustrated by the dialog box in this figure. After you do that, all of the chapter folders will be listed in the file tree at the left side of the window. Then, you'll be able to expand these folders to view and work with the subfolders and files that they contain. You'll get a chance to do this in the first exercise for this chapter.

Incidentally, notation like File→Open Folder means to drop-down the File menu and select the Open Folder. Similarly, View→Vertical Split means to run the Vertical Split command in the View menu. You will see this notation used throughout this book.

The dialog box for choosing a folder in Brackets



Two ways to open a project folder for the first time

- To open a project folder, use the File→Open Folder command to display the Choose a Folder dialog box. Then, choose the folder you want to open and click the Select Folder button.
- Click the name of the current project folder to display a list of recently opened folders. Then, select Open Folder at the top of the list to display the Choose a Folder dialog box.

How to re-open a project folder after it has been closed

- Click the name of the current project folder to display a list of recently opened folders. Then, select a folder from that list.

Description

- When you open a folder, it's displayed in a *file tree* at the left side of the window. Brackets considers this folder to be the current *project*.
- The first time you start Brackets, it displays a Getting Started project. You can preview its index.html file to learn about some of the features of Brackets.
- When you open another project folder, the previous project is closed.
- In general, each Brackets project should contain the folders and files for one web application. For the purposes of this book, however, you can treat all of the book applications as a single project and all of the exercises as another project.

Figure 2-7 How to open and close the folder for a website in Brackets

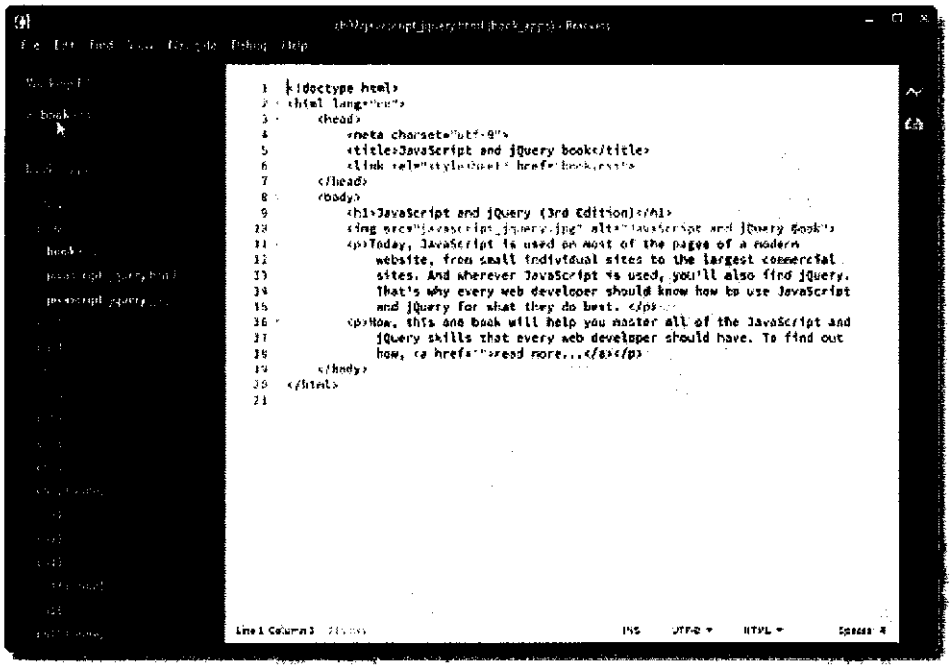
How to open, close, and display files

Figure 2-8 presents several techniques that you can use to open, close, and display files. Once you open the folder that contains the file, you can drill down to the file that you want to open by clicking on the ▶ symbols for the folders. In this example, the ch02 folder has been expanded so you can see the three files for the application for chapter 2. Then, to open a file, you just double-click on it.

You can also open a file that isn't part of the current project. To do that, you use the File→ Open File command to locate and select the file.

When you open a file in Brackets, it appears in the Working Files list that's above the file tree at the left side of the window. Then, you can switch between files by clicking on the file you want to display. You can also click on a file in the file tree to display a file without opening it. If you make a change to a file, though, it's opened and added to the Working Files list. To close a file in the Working Files list, you point to it and click the "X" to the left of the file name. Then, if you've made changes to the file and haven't saved them, Brackets will ask you if you want to save them.

Brackets with one open file and another file displayed



How to open a file

- To open a file within the open project, locate the file and then double-click on it.
- To open a file that isn't in the open project, use the File→Open command to locate and select the file.
- When you open a file, it appears in the Working Files list above the file tree.

How to close a file

- Point to the file in the Working Files list and then click the "X" to the left of the file name.

How to display a file

- To display a file without opening it, select it in the file tree. If you edit the file, it will be opened automatically.
- To display a file that's already open, select it in the Working Files list.

Description

- Unlike many text editors, Brackets doesn't use a tabbed interface. Instead, it lists open files in the Working Files list above the file tree for the project.

Figure 2-8 How to open, close, and display files in Brackets

How to start a new HTML file

All HTML documents contain the same DOCTYPE declaration and the same starting tags for the document tree. Because of that, it makes sense to add this code to each HTML file you create or to start each new HTML file from an HTML document that's similar to the one you're going to create. One way to add the starting code for an HTML document is to use an extension called Emmet.

Emmet is an invaluable web development tool that is available for nearly every text editor and IDE for web development. Because it is essential for increasing your productivity, you'll learn more about it in chapter 20. For now, you'll just learn how to use Emmet to add starting code to an HTML file.

To install Emmet or any other extension, you use the Extension Manager as shown in figure 2-9. To make it easier to find the extension you want to install, you can enter one or more keywords in the search box to filter the list of extensions in the Available tab. In this example, I entered "emmet". Then, to install an extension, you just click its Install button.

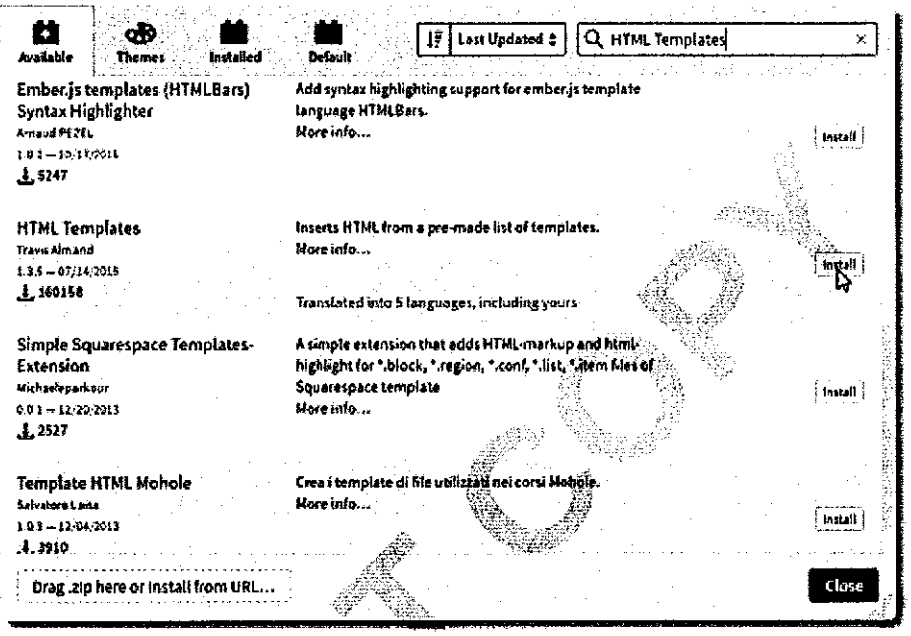
Note that you can run into problems if you try to install Emmet and you already have other extensions installed. In that case, you can disable any installed extensions from the Installed tab of the Extension Manager, install Emmet, and then re-enable the other extensions.

After you install the Emmet extension, you can use it to add the starting code for an HTML file. To do that, you create a new, empty file and save it with the .html extension. Then, you enter an exclamation mark (!) and press the Tab key. That adds the starting code for an HTML document to the file.

Another way to add starting code to an HTML file is to start a new HTML file from an existing HTML file. If, for example, you're going to create a second book page that's similar to an existing book page, it makes sense to start the new file from the old file. To do that, you just open the old file and save it with a new name. Then, you can delete the code in the old file that the new page doesn't need and add any new code that it does need.

As you become more familiar with HTML, you can create your own HTML files that contain the elements you need for different types of pages. Then, you can use those files as *templates* for creating other pages. If, for example, you're going to develop several pages that have the same general content, you can create an HTML template for those pages. Then, for each page you create, you can open the HTML file that contains the template and save it with a new name before you modify the file. That way, the original template file remains unchanged.

Brackets Extension Manager



How to add the Emmet extension

- Click the Extension Manager icon at the right side of the window or choose File→Extension Manager. From the Available tab of the Extension Manager, search for "emmet" and then click the Install button to the right of that extension.

How to start a new HTML file using Emmet

1. Select the File→New command to create a new, empty file.
2. Use the File→Save command to select the folder that the new file should be saved in, and enter a name for the new file including its .html extension.
3. Enter an exclamation mark (!) into the new file and press the Tab key to add the starting code for the HTML document.

How to start a new HTML file from another HTML file

- Open the file that you want to base the new file on. Then, use the File→Save As command to save the file with a new name.

Description

- By default, Brackets doesn't provide a way to add the starting elements for an HTML file. However, you can install an extension called Emmet that provides for that and many other timesaving tasks.
- If you're going to create a new file that's similar to an existing file, you can open the existing file and save it with a new name.

Figure 2-9 How to start a new HTML file in Brackets

How to edit an HTML file

Figure 2-10 shows how to edit an HTML file. When you open a file with an htm or html extension, Brackets knows what type of file you're working with so it can use color to highlight the syntax components. That makes it easier to spot errors in your code.

As you enter new code, the *code hints* feature presents lists of words that you can enter into your code. If, for example, you enter the left bracket (<) for a new element, a list of all of the elements is displayed. Then, you can select an element and press the Enter key to insert it into your code. You can also enter one or more letters after the starting bracket to filter the list so it only displays elements that start with those letters. In the example in this figure, I entered the letter *i* so only the words that start with that letter are displayed.

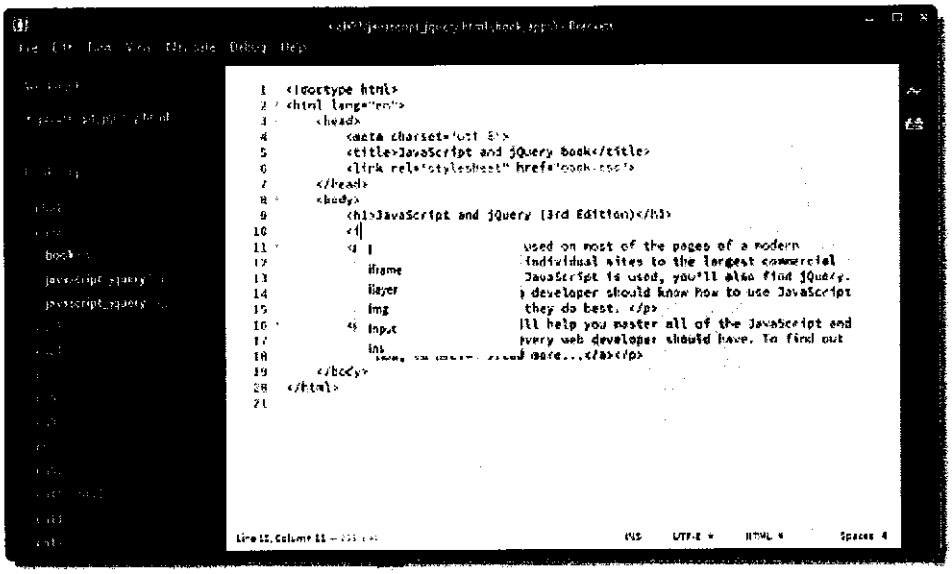
You can use a similar technique to enter attributes within an opening tag. To do that, enter a space and one or more letters after the element name. That displays a list of all the attributes that start with those letters. Then, you can select the attribute you want to select and press the Enter key. When you do, Brackets inserts the attribute along with an equal sign and double quotes, and you can enter the value of the attribute between the quotes.

After you've entered all of the attributes, you type the closing bracket (>) for the tag. Then, Brackets adds the closing tag so all you have to do is enter the content for the tag.

This figure also lists some common coding errors. Often, the color coding will help you spot the first three types of errors. If, for example, you misspell an element name or if you forget to code a closing quotation mark, the color coding will indicate that the code isn't correct.

On the other hand, Brackets has no way of knowing what the correct code should be for file references in link, img, or <a> elements. As a result, you must discover those errors when you test the web page. If, for example, the file reference for a style sheet in a link element is incorrect, the CSS won't be applied. If the file reference in an img element is incorrect, the image won't be displayed and the value of the alt attribute will be displayed. And if the file reference for an <a> element is incorrect, the browser won't access the correct page.

Brackets with code hints for an HTML element



Common coding errors

- An opening tag without a closing tag.
- Misspelled element or attribute names.
- Quotation marks that aren't paired.
- Incorrect file references in link, img, or <a> elements.

Description

- Brackets displays the different parts of a file in different colors so they're easy to recognize. This helps you spot some of the common errors. For this to work, the file must have the htm or html extension.
- The *code hints* feature displays a list of elements that start with what you've typed. To insert an element, click on it or use the arrow keys to highlight it and press the Enter key. You can also use this feature to insert attributes.
- When Brackets detects an error, it displays any code affected by the error in red.
- You can use the commands in the Edit and Find menus to do common editing tasks like commenting or uncommenting lines of code or finding and replacing code selections.
- You can also use standard editing practices like cutting, copying, and pasting.

Figure 2-10 How to edit an HTML file in Brackets

How to start and edit a CSS file

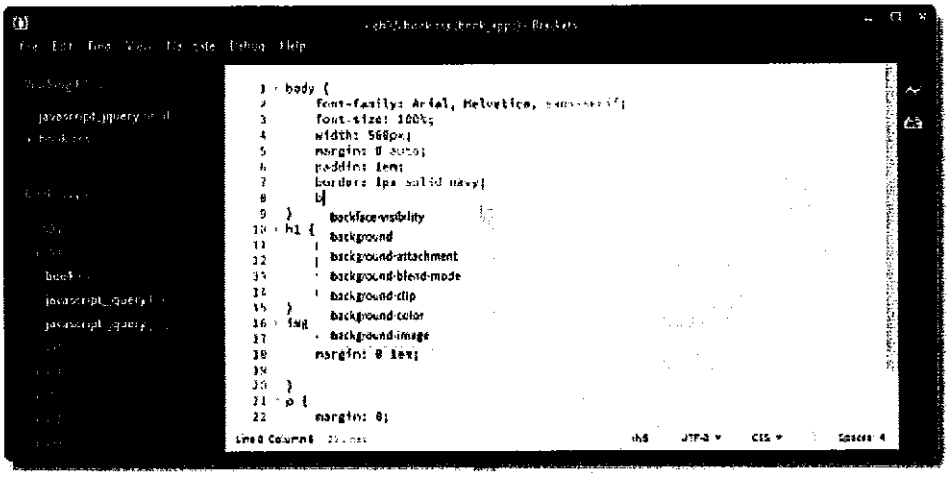
Figure 2-11 describes two options for starting a CSS file. First, you can use the File→New command to start a new, empty file. Then, before you start entering CSS code, you'll want to save the file with the .css extension so Brackets knows what kind of file it is.

Second, you can start a new CSS file from an existing CSS file. To do that, you just open the file in Brackets and save it with a new name. This makes sense when the new file will use many of the same styles as the old file. Then, you delete the styles that you don't need and add the styles that you do need.

This figure also shows how to edit a CSS file. When you open a file with the css extension, Brackets knows what type of file you're working with so it can use color to highlight the syntax components. When you type the first letter of a property, the code hints feature displays a list of properties that start with that letter. Then, you can select the property you want and press the Enter key to insert the name of the property and the colon that follows. In some cases, Brackets will display additional code hints that you can use to select the value of the property.

This figure also lists four common coding errors. Although the color coding can help you spot some of these errors, it can't help you spot all of them. For example, Brackets has no way of knowing if you code an id or class name incorrectly. Instead, you'll find that out during testing when you notice that the style rules that you've specified for the elements haven't been applied.

Brackets with code hints for a CSS property



Common coding errors

- Braces that aren't paired correctly.
- Misspelled property names.
- Missing semicolons.
- Id or class names that don't match the names used in the HTML.

Two ways to start a new CSS file

- To start a new CSS file from scratch, use the File→New command to create an empty file. Then, use the File→Save command to select the folder that the new file should be saved in, and enter a name for the new file including its .css extension.
- To start a new CSS file from another CSS file, open the file that you want to base the new file on and then use the File→Save As command to save the file with a new name.

How to edit a CSS file

- You can use the same techniques that you use to edit an HTML file, but the file must have the .css extension. Here again, the color coding will help you spot syntax errors.

Description

- The code hints feature displays a list of properties that start with what you've typed. Then, after you've selected one, Brackets automatically adds the colon after the property name. This code hints feature also works for property values.
- When you type the left brace after a selector, Brackets automatically adds the right brace. Then, you can enter the declarations between the braces.

Figure 2-11 How to start and edit a CSS file in Brackets

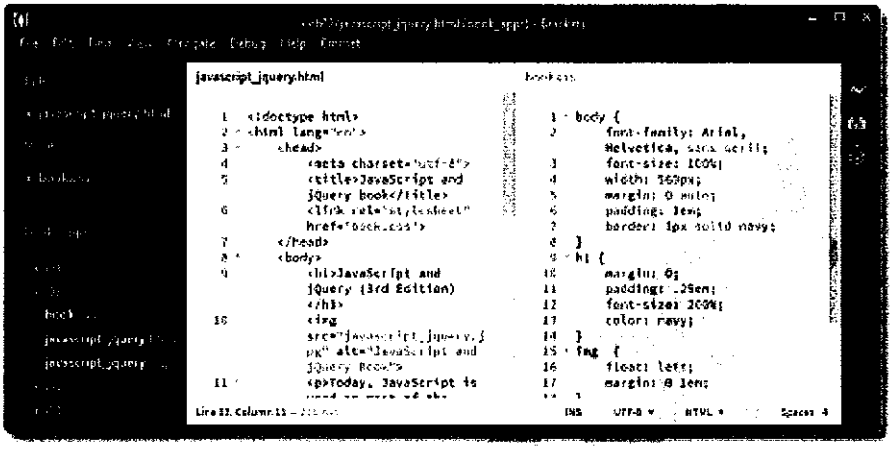
How to use split view and the Quick Edit feature

As you write the code for web pages, you often want to look at the HTML for a page and the related CSS at the same time. Unfortunately, most text editors don't let you do that. In contrast, Brackets provides two ways to do that, and they are summarized in figure 2-12.

The first way is to split the screen horizontally or vertically, and then put the HTML in one part of the split screen and the CSS in another. This is illustrated by the first example in this figure, which is split vertically. Here, the HTML is in the left pane and the CSS is in the right pane.

The other way to see the HTML and the CSS at the same time is to put the cursor in an HTML element name, an id attribute, or a class attribute and then press Ctrl+E (or Cmd+E for Mac OS X). When you do that, a Quick Edit window is opened that displays all of the style rules for that element, id, or class. This is illustrated by the second example in this figure. Better yet, you can edit the style rules that are shown or create new style rules.

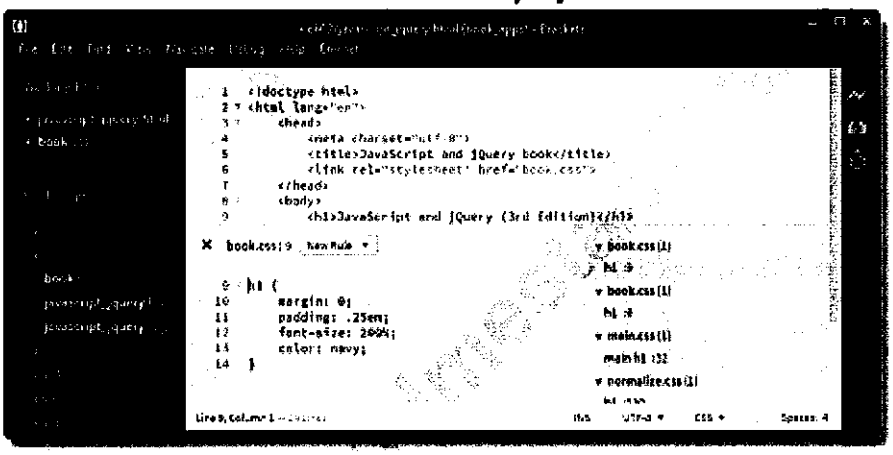
Brackets with a vertical split screen



How to display two files with a split screen

- Select View→Vertical Split or View→Horizontal Split to split the editor into two panes. Then, click in a pane and click on a file in the file tree to open it in that pane.
- To close split view, select View→No Split.

Brackets with the Quick Edit feature displayed



How to use the Quick Edit feature to display the CSS for an element

- Place the cursor in an element name or id or class attribute and press Ctrl+E for Windows or Cmd+E for Mac OS X. That will display the style rules for that element, id, or class.
- You can then edit the style rules or create new rules.

Figure 2-12 How to use split view and the Quick Edit feature in Brackets

How to preview an HTML file

Figure 2-13 shows how to preview an HTML file from Brackets. To do that, you display the HTML file and click the Live Preview icon. Then, the page is displayed in a separate copy of Chrome.

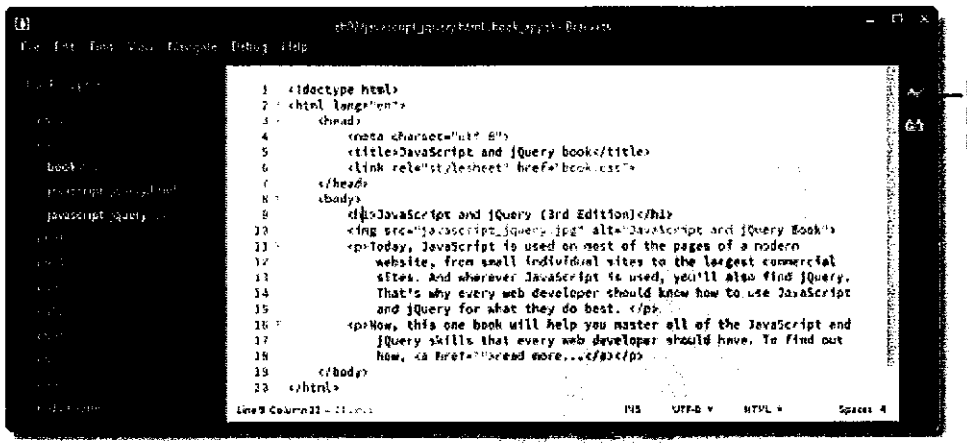
While a preview is displayed, you can make changes to the HTML and CSS code for the page and it will be reflected automatically in the preview. That makes it easy to test different content and formatting.

You can highlight an HTML element in the browser preview by placing the cursor in the opening tag of the element in the HTML file. And you can highlight all of the elements that a style rule is applied to in the browser preview by placing the cursor in that style rule in the CSS file. In this figure, for example, the cursor is in the h1 element, so that element is highlighted in the browser preview. Here, the dotted lines around this element indicate the padding for the element.

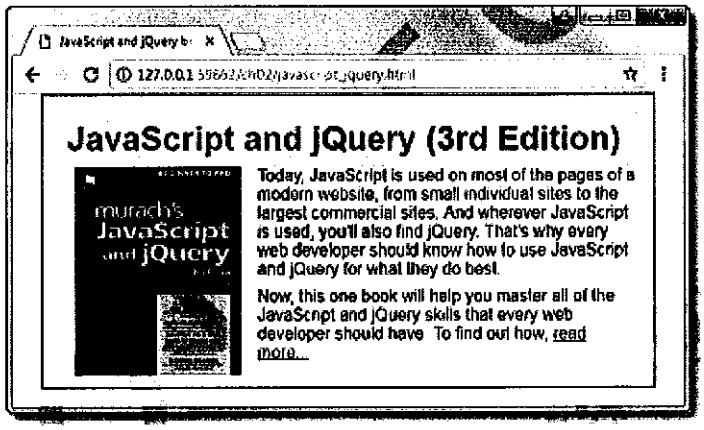
While Live Preview is connected to a browser window, the Live Preview icon is displayed in orange. To disconnect from the browser window, you can click this icon again. Note, however, that this doesn't close the browser window, so you'll want to do that when you're done previewing the page. You can also close the browser window while Live Preview is still connected. If you do, Brackets will display a message indicating that Live Preview was cancelled for an unknown reason.

Right now, Chrome is the only browser that you can use with Live Preview, and Brackets doesn't provide any other way to run an HTML file in other browsers. As a result, you need to use the techniques in the next figure to run an HTML file in other browsers. In the future, though, we expect Live Preview to work with other browsers.

The Live Preview icon



The Chrome browser with the selected h1 element highlighted



Description

- To preview an HTML file in Brackets, select it in the file tree or the Working Files list and then click the Live Preview icon at the right side of the window. The preview is displayed in a copy of Chrome that's separate from any other copies you may have open.
- If you make any changes to the HTML or CSS code for a page, the changes will be displayed automatically in the browser preview.
- If you place the cursor in an element in an HTML file, that element is highlighted in the browser preview.
- If you place the cursor in a style rule in a CSS file, all elements that the style rule is applied to are highlighted in the browser preview.

Figure 2-13 How to preview an HTML file from Brackets

How to test, debug, and validate HTML and CSS files

Now that you know how to create and edit HTML and CSS files, you're ready to learn how to test, debug, and validate those files.

How to test and debug a web page

When you *test* a web page, you run the file on all the browsers that are likely to access the file and try to identify all of the problems. When you *debug* a web page, you find the causes of the problems, fix them, and test again.

To run a web page, you can use one of the techniques in figure 2-14. If you're using Brackets, though, you can just click on the Live Preview icon. Once the page is displayed in the browser, you study it to make sure that it looks the way you want it to. You should also make sure that all of the links work.

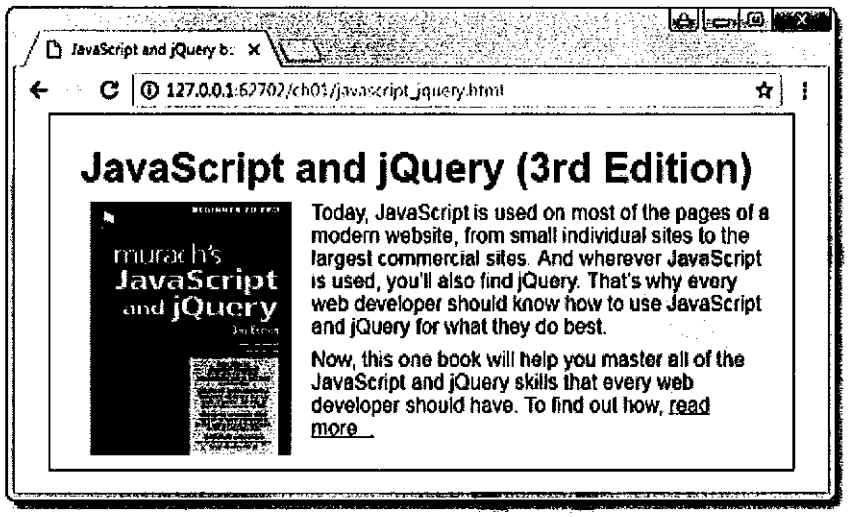
If you find errors, you need to change the HTML, the CSS, or both and test the page again. When you test a page for the second time, though, you don't need to run it again. Instead, you can click on the Reload or Refresh button in the browser's toolbar. Or, if you're using Brackets, the changes are applied automatically.

Often, the changes you make as you test a web page are just minor adjustments or improvements. But sometimes, the web page doesn't look at all the way you expected it to. Often, these errors are caused by trivial coding problems like missing tags, quotation marks, and braces, but finding these problems can be hard to do when your files consist of dozens of lines of code.

When you test a web page on more than one browser, you will sometimes find that the page works on one browser, but doesn't work on another. That's usually because one of the browsers makes some assumptions that the other browser doesn't. If, for example, you have a slight coding error in an HTML file, one browser might make an assumption that fixes the problem, while the other doesn't.

When you're using Brackets or an IDE, the color coding and error detection features should help you avoid problems caused by trivial coding errors. But sometimes, hard-to-detect errors still slip through. Then, it may help to validate an HTML or CSS file, which you'll learn how to do in the next two figures.

The HTML file displayed in the Chrome browser



Three ways to run a web page that's on an intranet or your own computer

- Start your browser, and use the File→Open or Open File command to open the file. Or, type the complete path and filename into the address bar, and press Enter.
- Use the file explorer on your system to find the HTML file, and double-click on it.
- If you're using Brackets, select the HTML file in the file tree and click the Live Preview icon to open the file in Chrome. If you're using another text editor or IDE, look for a similar link or a button or command for running the page.

How to rerun a web page from a browser after you change its source code

- Click the Reload or Refresh button in the browser.

How to test a web page

- Run the page in all of the browsers that are likely to access your site.
- Check the contents and appearance of the page to make sure it's correct in all browsers.
- Click on all of the links to make sure they work properly.

How to debug a web page

- Find the causes of the errors in the HTML or CSS code, make the corrections, and test again.

Description

- When you *test* a web page, you try to find all of the errors.
- When you *debug* a web page, you find the causes of the errors in the HTML or CSS code, correct them, and test the page again.

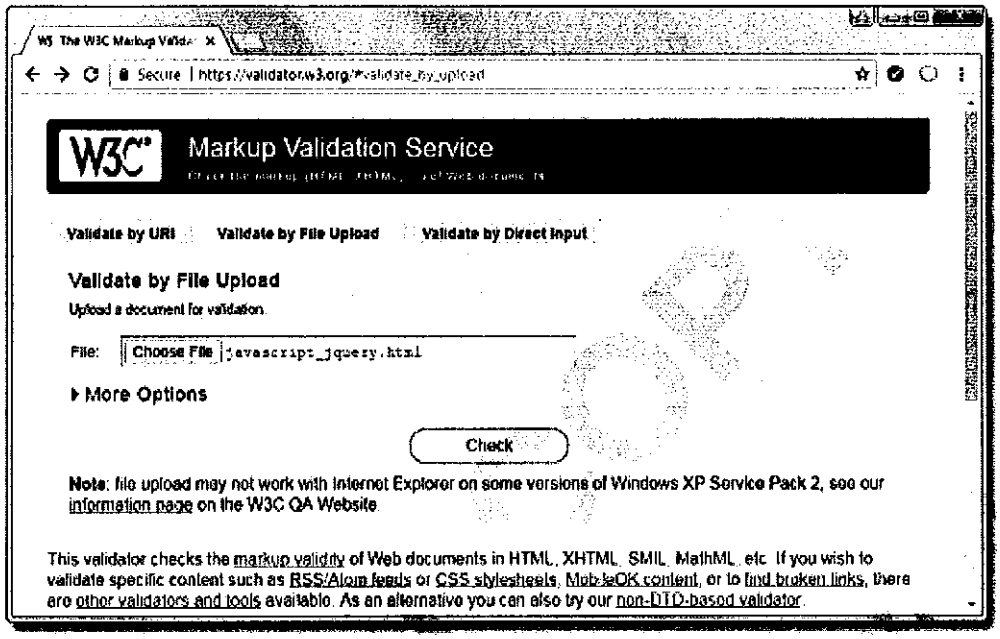
Figure 2-14 How to test and debug the HTML and CSS for a web page

How to validate an HTML file

To *validate* an HTML document, you use a program or website for that purpose. One of the most popular websites for validating HTML is the one for the W3C Markup Validation Service that's shown in figure 2-15. When you use this website, you can provide the HTML code that you want to validate in the three ways show in this figure.

In this example, the Validate by File Upload tab is shown. Then, you click the Browse button to find the file that you want to validate. Once that's done, you click the Check button to validate the document.

The home page for the W3C validator



How to use the W3C Markup Validation Service

- Go to the URL that follows, identify the file to be validated, and click the Check button:
  https://validator.w3.org/

Three ways to provide the code to be validated

- If the file you want to validate has already been uploaded to a web server, you can validate it by entering its URL on the Validate by URI tab.
- If the file hasn't been uploaded to a web server, you can validate it by locating it on the Validate by File Upload tab.
- You can also validate HTML by copying and pasting it into the Validate by Direct Input tab.

Description

- To *validate* the HTML for a page, you can use a program or website for that purpose. One of the most popular websites is the W3C Markup Validation Service.
- Validation insures that your code is correct, which may improve SEO. Validation can also help you find errors in your HTML that you aren't aware of.

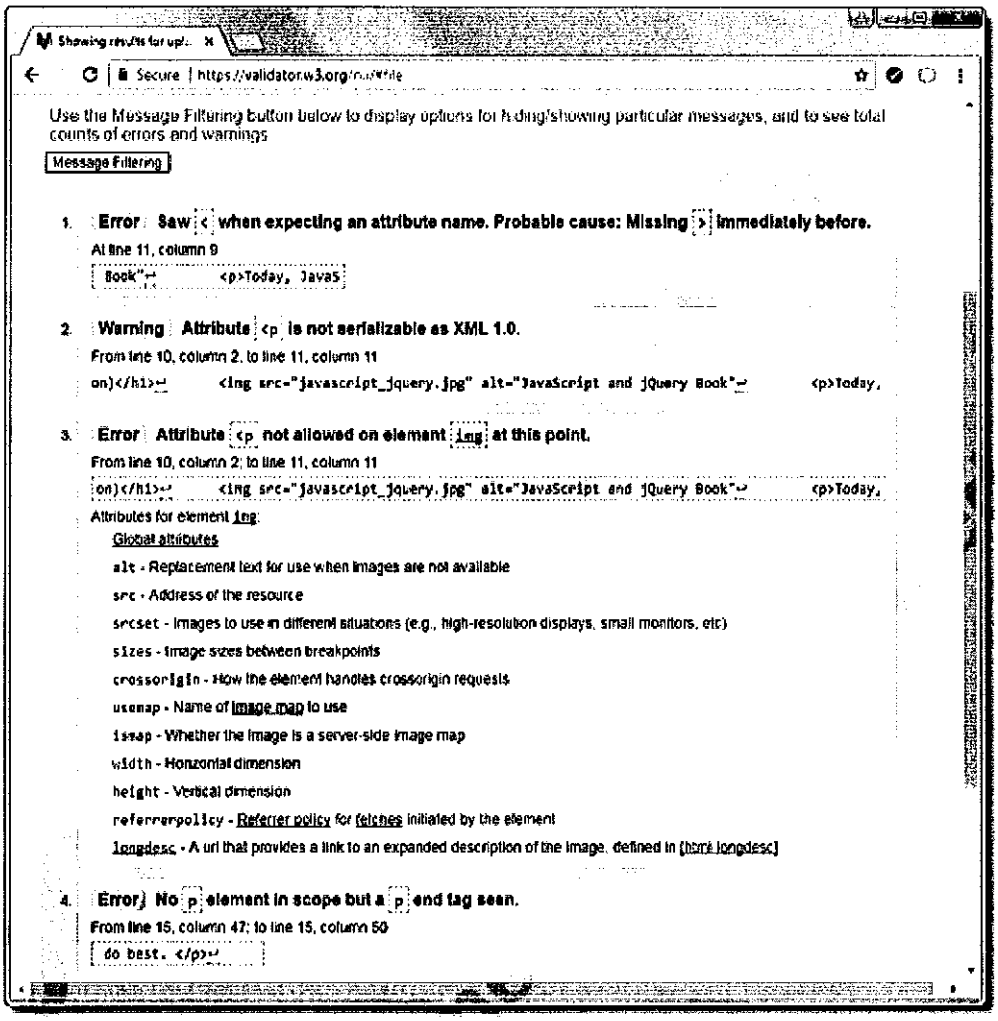
Figure 2-15 How to validate an HTML file (part 1 of 2)

If the HTML code is valid when a document is validated, the validator displays a message to that effect. However, it may still display one or more warning messages. On the other hand, if the code contains errors, the validator will display a list of the errors like the one in the second part of this figure. Here, the validation results are shown at the W3C Markup Validation Service website.

In this example, the error message for line 11, the warning and error messages for line 10, and the error message for line 15 were caused by a missing > at the end of the img element on line 10. In this case, Brackets would display the invalid code in red so this error shouldn't slip by you. And if it did, you should catch that error when you test the web page. But sometimes, a web page with an error or two will be displayed okay in one browser, even though it won't be displayed correctly in another browser.

Should all HTML documents be validated, even though it's estimated that 99% of all web pages aren't? We say, yes! As we see it, validation is a useful practice that will solve some testing problems, and IDEs like Dreamweaver make validation so easy that it's well worth doing. Besides that, validation may help your SEO results because clean code gets better results.

The errors and warning for an HTML file with a missing > at the end of the img element



Description

- If the HTML document is valid, the validator will indicate that the document passed the validation. However, one or more warnings may still be displayed.
- If the HTML document isn't valid, the validator will list and describe each error and warning.
- You can use the Message Filtering button above the list to display a count of the errors and warnings and to select which errors and warnings you want displayed.

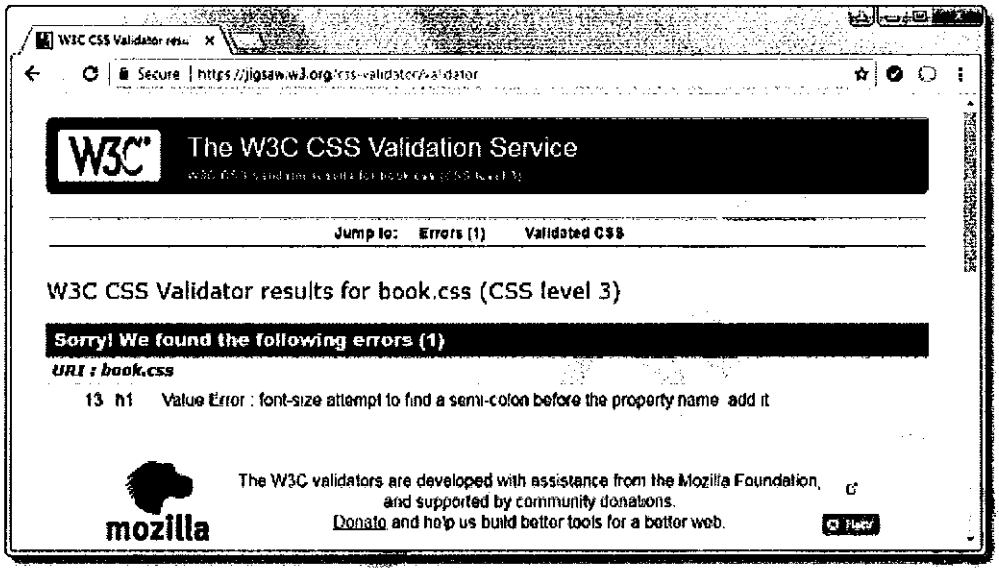
Figure 2-15 How to validate an HTML file (part 2 of 2)

How to validate a CSS file

You can validate a CSS file the same way you validate an HTML file, either with a validation program that may be part of an IDE or with a website service like the W3C CSS Validation Service. If you use the W3C Validation Service, the opening page looks like the one for HTML validation. Then, you can use any one of the three tabs to validate a CSS file.

If the file contains errors, the validation service will display a screen like the one in figure 2-16. Here, the error was caused by a missing semicolon at the end of a declaration. Although you would normally catch this type of error in Brackets due to the error markers and color coding, CSS validation is useful when the file is large, the CSS isn't working right, and you can't spot any errors. CSS validation may also mean that your CSS will work on an infrequently-used browser that isn't one of the browsers that you're using for testing.

The CSS Validation Service with errors displayed



How to use the W3C CSS Validation Service

- Go to the URL that follows, identify the file to be validated, and click the Check button:
  https://jigsaw.w3.org/css-validator/

Description

- To validate the CSS for a page, you can use a program or website for that purpose. One of the most popular websites is the W3C CSS Validation Service. This website works the same as the W3C Markup Validation Service.
- Validation not only insures that your code is correct, but it can also help you find errors in your CSS that you aren't aware of.

Figure 2-16 How to validate a CSS file

Perspective

Now that you've completed this chapter, you should be able to create and edit HTML and CSS files using Brackets or the editor of your choice. Then, you should be able to test those files by displaying their web pages in your web browser and in any other web browsers that your users might use. You should also be able to validate the HTML and CSS files for a web page.

At this point, you're ready to learn the coding details for HTML and CSS. So, in the next chapter, you'll learn the details for coding the HTML elements that define the structure and content for a web page. And in chapters 4, 5, and 6, you'll learn the details for coding the CSS style rules that format the HTML content.

Terms

syntax
HTML document
DOCTYPE declaration
document tree
HTML element
root element
opening tag
closing tag
content of an element
empty tag
nested elements
attribute
Boolean attribute
comment
comment out
whitespace
style rule
selector
declaration
property
value
type selector
Brackets project
file tree
template
code hint
testing
debugging
HTML validation
CSS validation

Summary

- An *HTML document* consists of a *DOCTYPE declaration* that indicates what version of HTML is being used and a *document tree* that contains the *HTML elements* that define the content and structure of a web page.
- The *root element* in a document tree is the html element, which always contains a head element and a body element. The head element provides information about the page, and the body element provides the structure and content for the page.
- Most HTML elements consist of an *opening tag* and a *closing tag* with *content* between these tags. When you *nest* elements with HTML, the inner set of tags must be closed before the outer set.
- *Attributes* can be coded in an opening tag to supply optional values. An attribute consists of the name of the attribute, an equal sign, and the attribute value. To code multiple attributes, you separate them with spaces.
- An *HTML comment* can be used to describe or explain a portion of code. Because comments are ignored, you can also use comments to *comment out* a portion of HTML code so it isn't rendered by the browser. This can be helpful when you're testing your HTML code.
- *Whitespace* consists of characters like tab characters, line return characters, and extra spaces that are ignored by browsers. As a result, you can use whitespace to indent and align your code.
- A *CSS style rule* consists of a selector and declarations. The *selector* identifies the HTML elements that are going to be formatted. Three of the common CSS selectors select by element (called a *type selector*), ID, and class.
- A *declaration* in a CSS style rule consists of a *property*, a colon, a *value*, and a semicolon that apply formatting.
- *CSS comments* work like HTML comments. However, CSS comments start with /* and end with */, and HTML comments start with <!-- and end with -->.
- Brackets is a text editor that can be used to edit HTML or CSS code. To help you read the code, Brackets displays the syntax components with different colors. It also provides *code hints* and error checking that detects common entry errors.
- When you start a new HTML or CSS file, it's best to start from a *template* or an old file that's similar to the new file that you're going to create.
- To *test* an HTML file, you run it on all of the browsers that your clients may use. Then, if you discover problems, you need to *debug* the code and test it again.
- To *validate* an HTML or CSS file, you can use a program or website for that purpose. Sometimes, this can help solve hard-to-detect debugging problems.

Before you do the exercises for this book...

If you haven't already done it, you should install the Chrome browsers and the applications, examples, and exercises for this book. If you're going to use Brackets as your text editor, you should also download and install that product. The procedures for doing all three are in appendix A.

Exercise 2-1 Get started right with Brackets

If you're going to be using Brackets as your text editor, this exercise will get you started right.

Set up the projects for this book

1. Start Brackets. Then, use the File→Open Folder command to open a project folder for the applications that are presented in this book:
   `C:\murach\html5_css3_4\book_apps`
   This should display a file tree with one folder for each chapter.
2. Use the same command to open a project folder for the book exercises:
   `C:\html5_css3_4\exercises`
   This should close the book_apps folder and replace it with a new file tree.
3. Click on the name of the exercises folder at the top of the tree. Then, select the Open Folder command and open this folder:
   `C:\murach\html5_css3_4\book_examples`
4. You now have easy access to all of the book applications, exercises, and examples for this book. To illustrate, click on the name of the book_examples folder at the top of the file tree. Then, select exercises from the drop-down list that's displayed to reopen that folder.

Edit the code in a book application and use Live Preview to test it

5. In the file tree, click on the ▶ symbol before ch02 to display the files in this folder. Then, double-click on the file named javascript_jquery.html to open and display that file. Note that it is now listed under Working Files.
6. Start a new h1 element right after the h1 element and note how Brackets provides code hints. Select h2 from the hints, type the closing bracket (>) for the opening tag, and note how Brackets adds the closing tag. Now, undo this change.
7. Click on the Live Preview icon shown in figure 2-13 to preview the file in Chrome. Then, click on an element in the HTML file to see that it is now highlighted in the browser preview.
8. Double-click on the file named book.css to open that file. Then, change the color property for the h1 element to red, and look at the browser preview to see that the heading is now displayed in red.

Use the Quick Edit and split screen features

9. Click on javascript_jquery.html in the Working files to display that file. Then, put the cursor in the h1 tag, and press Ctrl+E (or Cmd+E) to display the CSS for that element. Now, change the color back to navy, and note the change in Chrome. Last, click the X in the upper left of the Quick Edit feature to close it.
10. Use the View→Vertical Split command to split the screen. Then, put the CSS file in the right pane. To do that, you can drag the file from the Left group in Working Files to the Right group. Or, you can click the right pane to put the focus in it and click on the book.css file in the file tree (not in Working Files).
11. Change the float property for the img element from left to right, and note the change in Chrome.
12. Use the View→No Split command to end the split screen feature.

Start and edit new HTML and CSS files

13. Use the procedure in figure 2-9 to add the Emmet extension to Brackets.
14. Use the File→New command to start a new file, and use the File→Save As command to save it with the name testpage.html in the ch02 folder. Then, enter an exclamation mark into the new file and press the Tab key. This should insert the starting code for an HTML document into the file.
15. Select the book.css file in the ch02 folder. Then, save this file with the name testpage.css in the same folder. Now, delete the style rules after the body and h1 rules, and save the file.
16. In the testpage.html file, change the content for the title element to "Test Page". Then, add an h1 element within the body element that says "Test Page."
17. Display the javascript_jquery.html file, copy the link element, display the testpage.html file, and paste the link element right after the title element. Then, change the href attribute to "testpage.css".
18. Test this page by clicking the Live Preview icon. This illustrates how fast you can get started with a new web page. Then, click on the javascript_jquery.html file to display it, and see that Chrome switches to that file.

Experiment and clean up

19. If you want to experiment more, by all means do that. When you're through, continue with the next steps.
20. Use the File→Close All command to close all of the open files. Or, move the cursor to a file in the Working Files list and click the X on its left to close one file at a time. Either way, if a file hasn't been saved, a dialog box will be displayed so you can save it.
21. When all of the files have been closed, close the instance of Chrome that has been previewing the files. Then, close Brackets.

Exercise 2-2 Edit and test the book page

In this exercise, you'll edit and test the book page that is shown in figure 2-14. You should do this exercise whether you're using Brackets or another text editor.

Open and test the files for the book page

1. Start your text editor.
2. Open this HTML file in your text editor:
   `c:\html5_css3_4\exercises\ch02\javascript_jquery.html`
   For Brackets users, you can just open the exercises folder and locate this file.
3. Open the CSS file named book.css that's in the same folder.
4. Run the HTML file in Chrome.

Modify the HTML and CSS code and test again

5. Go to your text editor and display the book.css file. Then, change the float property for the img element from left to right. Now, test this change.
6. Go to the book.css file, and change the color for the h1 element to "red" and change the font-size to 180%. Now, test this change.
7. Go to the javascript_jquery.html file, and add a <p> element at the bottom of the page that has this content:
   `For customer service, call us from 8am to 5pm at 1-555-555-5555.`
   Then, test this change.
8. Go to the javascript_jquery.html file and add an id attribute with the value "service" to the <p> element that you just entered.
9. Go to the book.css file and enter a style rule for the element with "service" as its id. This rule should set the color property to red, and it should set the clear property to both. If you need help doing this, refer to figure 2-6. Now, test these changes.

Validate the HTML and CSS files

10. In the HTML file, delete the ending > for the img tag, and save the file. Then, go to the site in figure 2-15, and use the Validate by File Upload tab to validate the file. If you scroll down the page when the validation is done, you'll see 3 error messages, even though the file contains just one error.
11. In the CSS file, delete the semicolon for the color property in the h1 style rule, and save the file. Then, go to the site in figure 2-15, and use the Validate by File Upload tab to validate the file. This time, you'll see 1 error message.
12. Now, undo the errors in the files, save the files, and validate the HTML page again. This time, it should pass. Then, test the web page one last time.