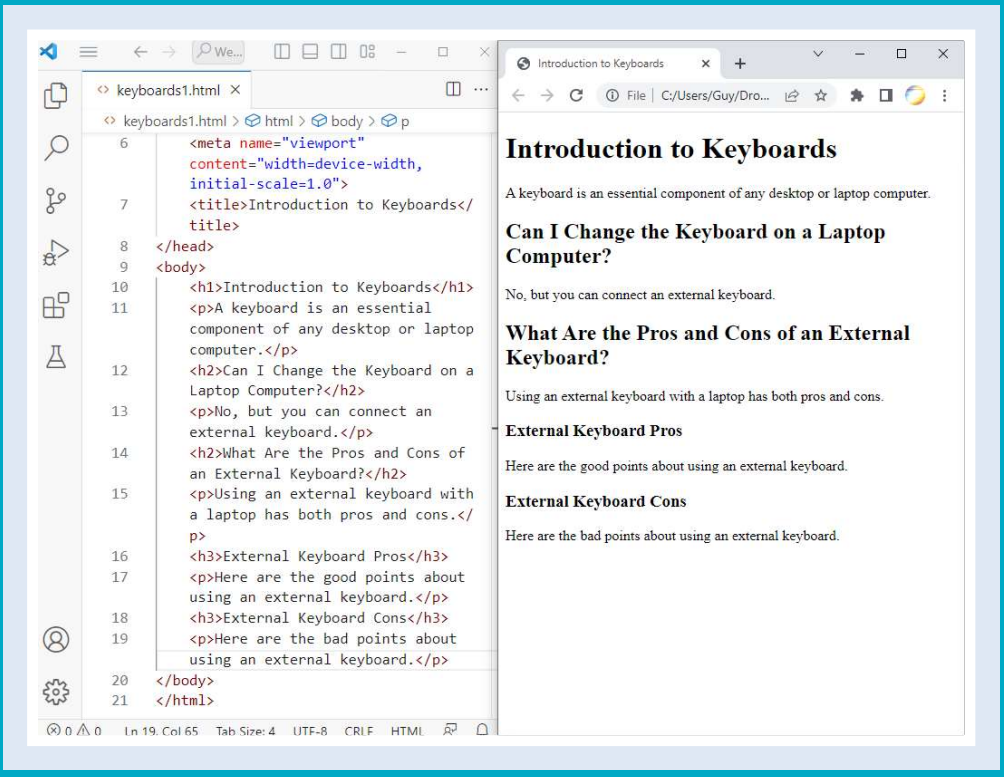
CHAPTER 2

Creating Your First Web Pages

In this chapter, you briefly study the anatomy of a web page before launching Visual Studio Code, configuring it, and using it to create your first web pages. You learn to add headings, text, and comments to web pages; view a page's source code and validate HTML; and create hyperlinks between web pages.



Study the Anatomy of a Web Page

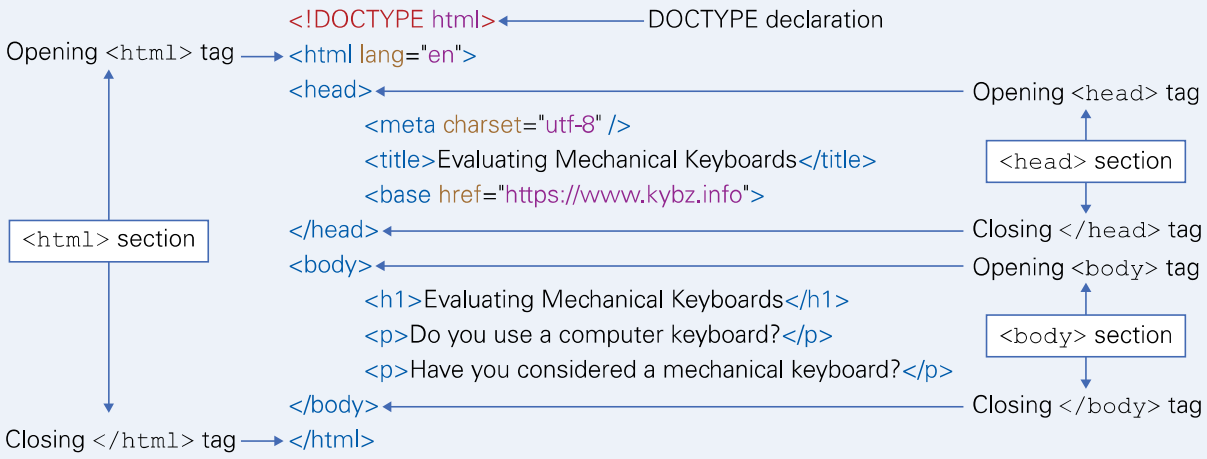
In HTML, each web page has the same basic structure, no matter how simple or complex the page is.

An HTML web page starts with a `DOCTYPE` definition that specifies the document's encoding type. Next comes a statement of the language used, such as `lang="en"` to indicate English. After that, the web page consists of a `head` element and a `body` element. The `head` element contains information about the document, such as the page title and the base URL for links. The `body` element holds the remaining content of the web page, such as headings, text, and linked media files.

Identify the Four Key Elements of a Web Page

Each valid HTML page must contain four key elements, as illustrated in the nearby code:

- **`DOCTYPE` declaration.** This declaration tells the browser the document type of the HTML page. The browser needs this information to interpret the document's codes correctly.
- **`<html>` tags.** The whole of the web page appears between the opening `<html>` tag and the closing `</html>` tag. These tags show you the standard format for two-part tags: The closing tag consists of a forward slash, `/`, and the same text as the opening tag. For example, you use the `<p>` opening tag to tell HTML to start a paragraph and the `</p>` closing tag to end the paragraph.
- **`head` element.** This element contains the metadata for the document, including the character set used for encoding and the title, which most web browsers display in the title bar.
- **`body` element.** This element contains the content of the web page, such as the text of headings, paragraphs, and lists, and links to external content, such as images.



Understanding the DOCTYPE Declaration

Each HTML page begins with a declaration of the document type using the DOCTYPE keyword. The DOCTYPE declaration for an HTML5 document is straightforward and short:

```
<!DOCTYPE html>
```

Given that HTML5 has been in use for 15 years now, this is the DOCTYPE declaration you are likely to see most often. But if you work with legacy web pages, you are likely to see other DOCTYPE declarations, so it is helpful to be able to recognize them.

HTML version 4 and Extensible Hypertext Markup Language, XHTML, use more complex DOCTYPE declarations that include the details of the document type definition, DTD, the page uses. For example, the HTML 4 Strict standard uses the following DOCTYPE declaration:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

Similarly, the XHTML 1.1 standard uses the following DOCTYPE declaration:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

The opening <html> tag for an XHTML doctype includes the xmlns attribute, which provides details of the XHTML namespace the document uses:

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

A *namespace* is a particular class of elements in which each element has a unique name. For example, XHTML uses a different namespace than HTML 4, but some names in the separate namespaces are the same.

Understanding HTML Validity and Validation

An HTML document must be valid in order to display properly and consistently in all browsers. *Valid* means the document contains all the essential elements in an acceptable order and that all the formatting tags are correct, complete, and in the right places.

You can perform an informal validity check by opening an HTML document in several browsers and seeing if it displays correctly. But because browsers encounter many pages that contain errors, the browsers are built to tolerate errors and display pages as well as they can, so opening an HTML document in a browser is not a strict test of the HTML's validity. For a strict check, you can use validity checkers built into many web development tools or online checkers such as the W3C Markup Validation Service checker at https://validator.w3.org.

Tell Visual Studio Code Which Folder to Use

Before creating HTML files, tell Visual Studio Code which folder to store them in. This can be either an existing folder or a folder you create now from within Visual Studio Code. After identifying the folder, you instruct Visual Studio Code whether to trust the folder's parent folder — the folder that contains the folder.

Visual Studio Code's primary sidebar includes an Explorer pane that enables you to create, manage, and open folders and files. Working in the Explorer pane in Visual Studio Code is easier and quicker than working in a File Explorer window.

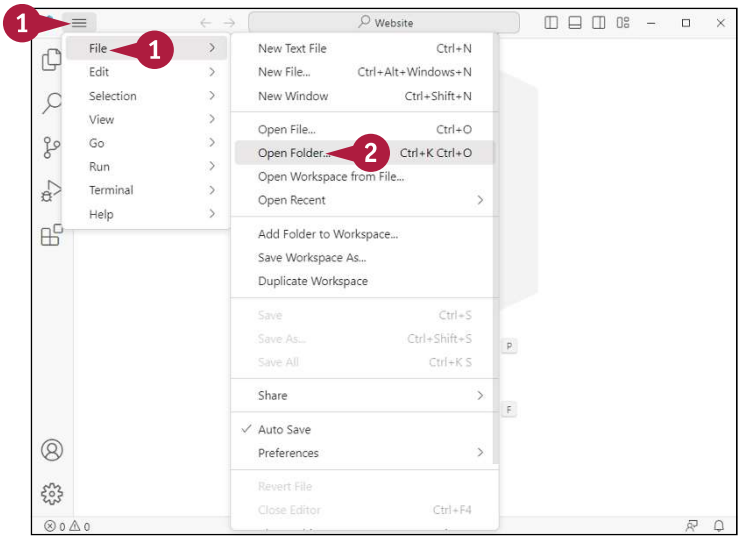
Create a Folder for Your Website

1. In Visual Studio Code on Windows, click **Menu** (☰), and then click **File**.

   On macOS and Linux, click **File**.

   The File menu opens.

2. Click **Open Folder**.



   The Open Folder dialog box appears.

3. Navigate to the folder in which you want to store the folder containing your website.

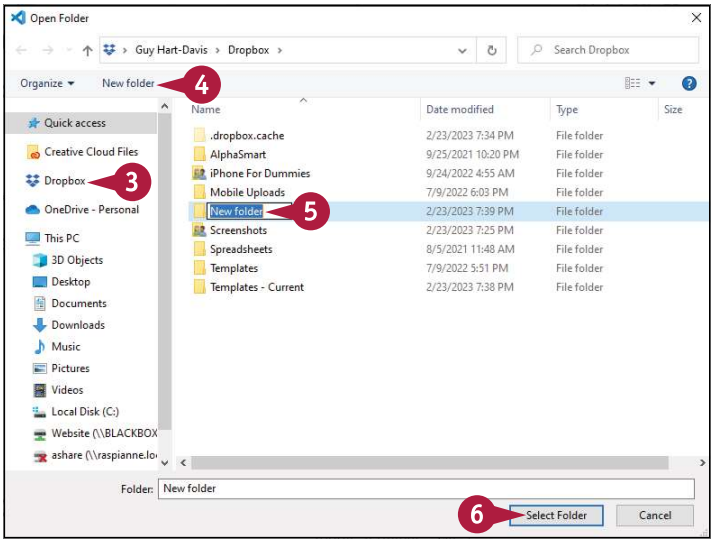
Note: If you want to use an existing folder, navigate to that folder and select it.
Go to step 6.

4. Click **New Folder**.

   Visual Studio Code creates a new folder in the folder, gives it the default name *New folder*, and selects the name.

5. Type the name for the folder, and then press Enter to apply it.

6. Click **Select Folder**.



The Open Folder dialog box closes.

The Do You Trust the Authors of the Files in This Folder? dialog box opens.

A If you want to trust the folder that contains this folder, select **Trust the authors of all files in the parent folder** (☑).

7 Click **Yes, I trust the authors**.

Note: If you do not trust the authors of the files, click **No, I don't trust the authors**.

The Do You Trust the Authors of the Files in This Folder? dialog box closes.

B Visual Studio Code opens the primary sidebar with the Explorer pane displayed.

C The folder you selected appears.

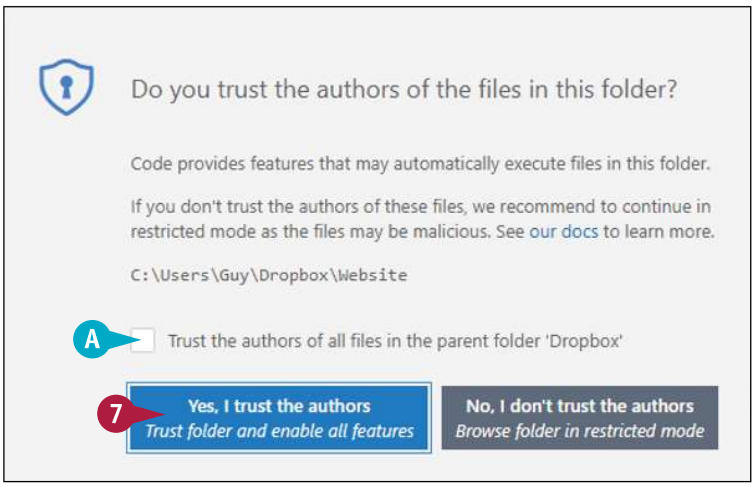
D You can click **Collapse** (⌄) to collapse a section of the Explorer pane.

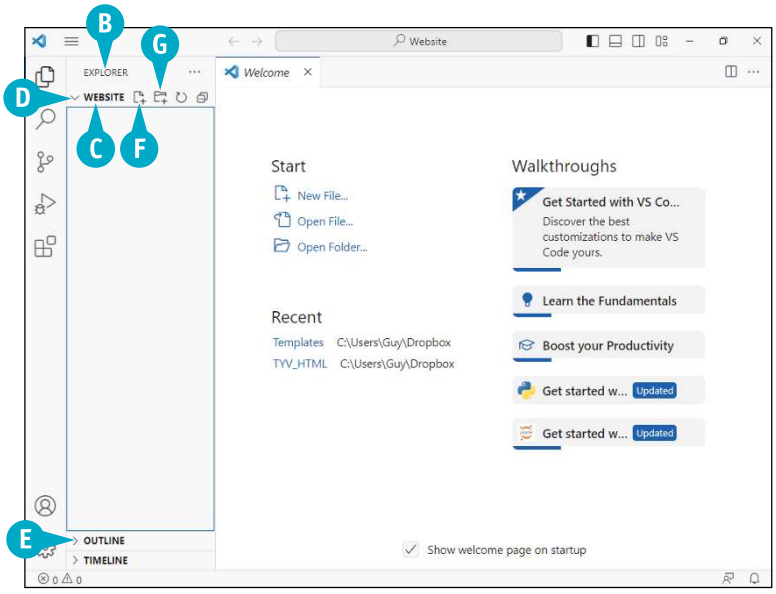
E You can click **Expand** (›) to expand a section of the Explorer pane.

Note: Move the pointer over the folder section of the Explorer bar to display the New File button and New Folder button.

F You can click **New File** (📄) to create a new file in the folder.

G You can click **New Folder** (📁) to create a new folder in the folder.





TIP

How do I stop the folder name from appearing in the Search box in the Visual Studio Code title bar?

After you open a folder, its name appears in the Search box in the title bar of the Visual Studio Code app. To remove the name, close the folder by clicking **File** — on Windows, click **Menu** (☰) and then click **File** — and then clicking **Close Folder** on the menu.

Create Your First Web Page

After setting Visual Studio Code to use your website's folder, you can create your first web page. In this section, you begin the page, adding a title to the page header and inserting some placeholder text in the page's body. If you configured the AutoSave feature in Settings, Visual Studio Code saves your work automatically. If not, press Control+S on Windows or Linux or ⌘+S on the Mac when you want to save the file.

Complete this section before the following several sections, in which you view the page and then add headings, text, and other elements.

Create Your First Web Page

1. In Visual Studio Code, click **Explorer** ().

   The primary sidebar opens, showing the Explorer pane.

Note: If Visual Studio Code does not automatically expand the folder section of the Explorer pane, click **Expand** () to expand it.

2. Click **New File** ().

   An edit box appears.

3. Type the filename, including the `.html` file extension, and then press Enter.

   The example uses the filename `keyboards1.html`.

   Visual Studio Code creates the file and displays it in the main pane.

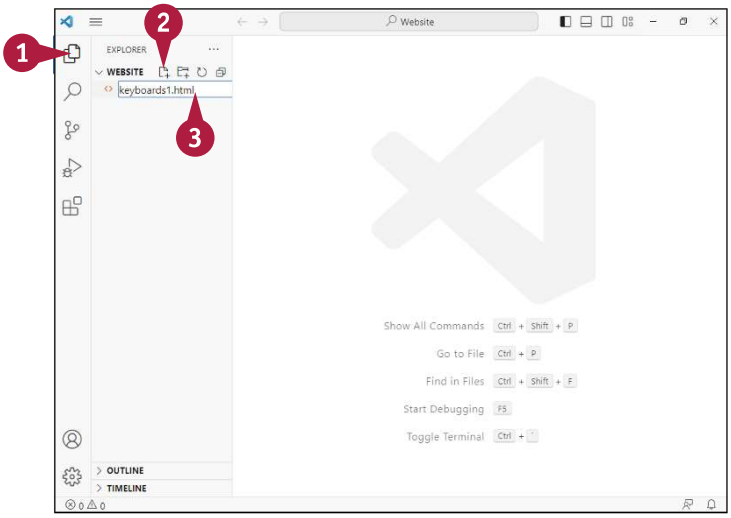
   The insertion point appears in the first line of the file, which is numbered 1.

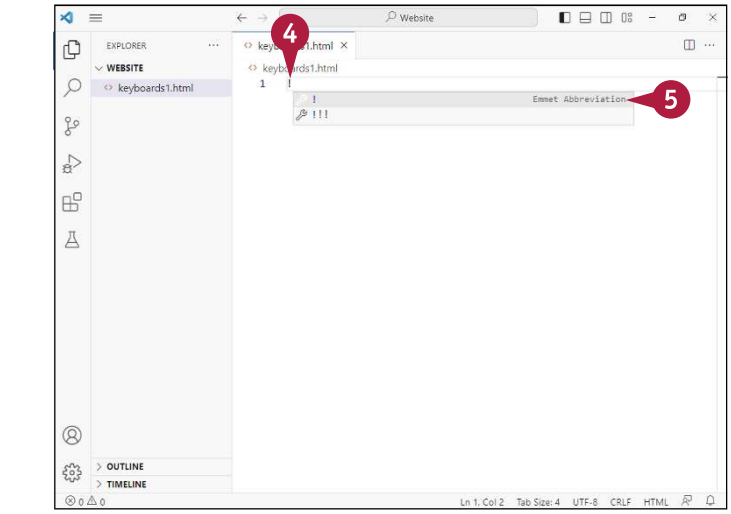
4. Type `!`.

   The AutoComplete list opens, showing available AutoComplete entries that start with the character !.

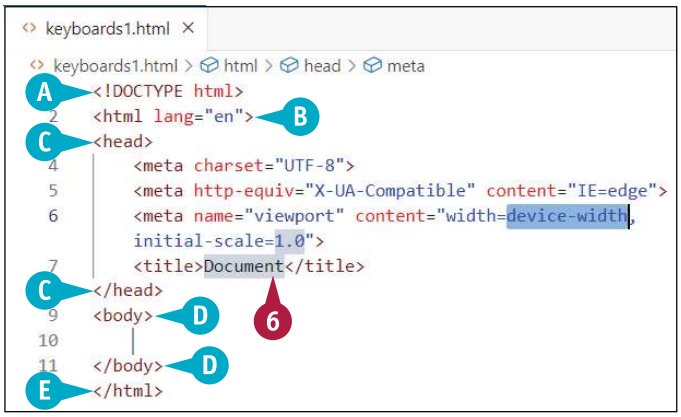
5. Click the first item, which is identified as an Emmet abbreviation.

Note: You can also expand the Emmet abbreviation by pressing Tab with the abbreviation highlighted.





Visual Studio Code expands the abbreviation, entering the skeleton code of a web page in place of the exclamation point.



A The `DOCTYPE` declaration specifies `html`, making this an HTML5 document.

B The opening `<html>` tag specifies `lang="en"`, setting the language to English.

C The opening `<head>` tag and closing `</head>` tag delimit the `head` section, which contains meta information and the page's title.

D The opening `<body>` tag and closing `</body>` tag delimit the `body` section, which is empty.

E The closing `</html>` tag ends the web page.

6 Double-click **Document** on line 7.

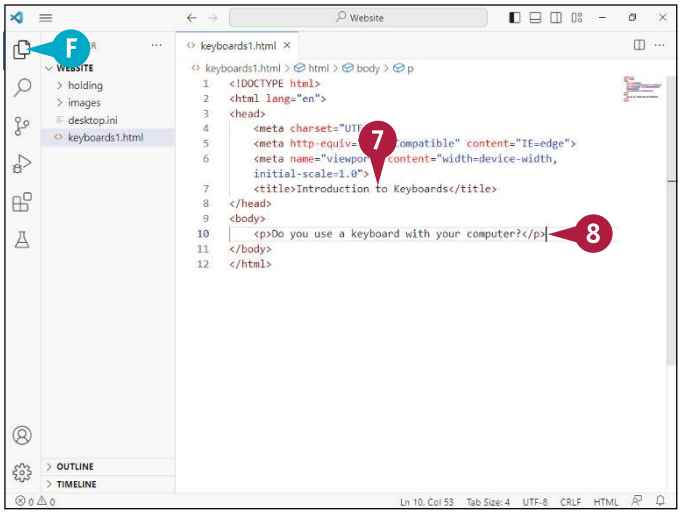
Visual Studio Code selects the word.

7 Type the title you want to give the web page.

Note: The web page's title appears in the browser's title bar. If the web page is on a browser tab, the title appears on the tab.

8 Click in line 10 and type the opening `<p>` tag, text of your choosing, and the closing `</p>` tag — for example:

```
<p>Do you use a keyboard with
your computer?</p>
```



F You can click **Explorer** (⧉) to close the primary sidebar, giving yourself more room to work on the web page.

TIP

What is the tiny pane of illegible text on the right of the Visual Studio Code window?

This pane contains the Minimap, a visual feature for navigating quickly through your code. The Minimap provides essentially a thumbnail view of the code in the main pane. Move the pointer over the Minimap to display a gray highlight showing you one screen's worth of the code, scroll up and down to the code you want to view, and then click the display that screen's worth of code in the main pane. If you do not find the Minimap useful, click **View** on the menu bar and then click **Minimap** to turn the Minimap off.

Open the Web Page in a Browser

Visual Studio Code does not have a built-in browser for viewing web pages, but you can quickly open a web page in a browser to view the page. Displaying the page makes it easy for you to see the effects of the changes you make in the HTML code. You can use whichever browser you prefer; this section uses Google Chrome for the example.

This section assumes that you have created a web page in Visual Studio Code, as explained in the previous section, "Create Your First Web Page."

Open the Web Page in a Browser

1. In Visual Studio Code, with your web page open, right-click the web page's tab.

   The contextual menu opens.

2. Click **Reveal in File Explorer**.

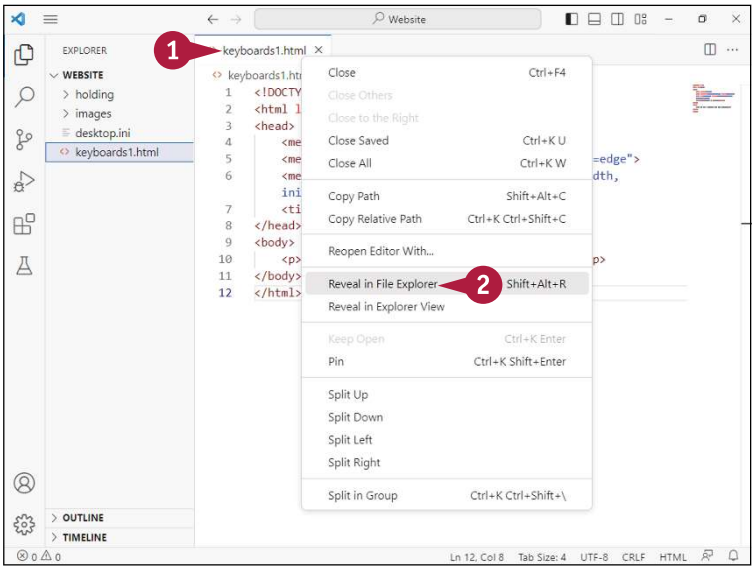
Note: If the primary sidebar is displayed and showing the Explorer pane, you can also right-click the web page file and then click **Reveal in File Explorer** on the contextual menu.

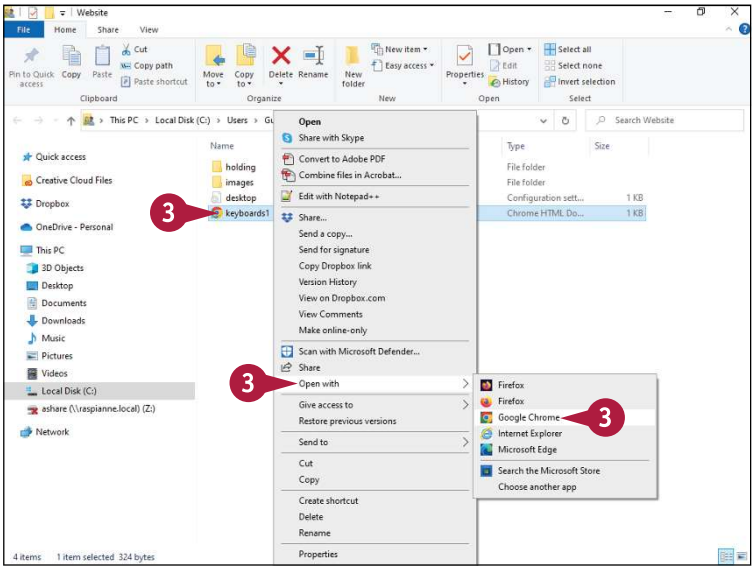
   A File Explorer window opens to the web page's folder.

Note: If you want to open the web page in your default web browser, simply double-click the file.

3. Right-click the web page file, click or highlight **Open with** on the contextual menu, and then click the appropriate browser — for example, **Google Chrome**.

Note: If your computer has a large monitor, splitting the screen between your Visual Studio Code window and a browser window can help you work efficiently. If you have two monitors, you may want to place Visual Studio Code on one monitor and the browser on the other.





A browser window or tab opens showing the web page.

You can now arrange the browser window and the Visual Studio Code window so that you can see both. The following steps show one easy way to do this using the Windows Snap feature.

4. Click the title bar of the browser window and drag left or right until the pointer hits the edge of the window.

A. Windows Snap resizes and positions the window to fit that half of the screen.

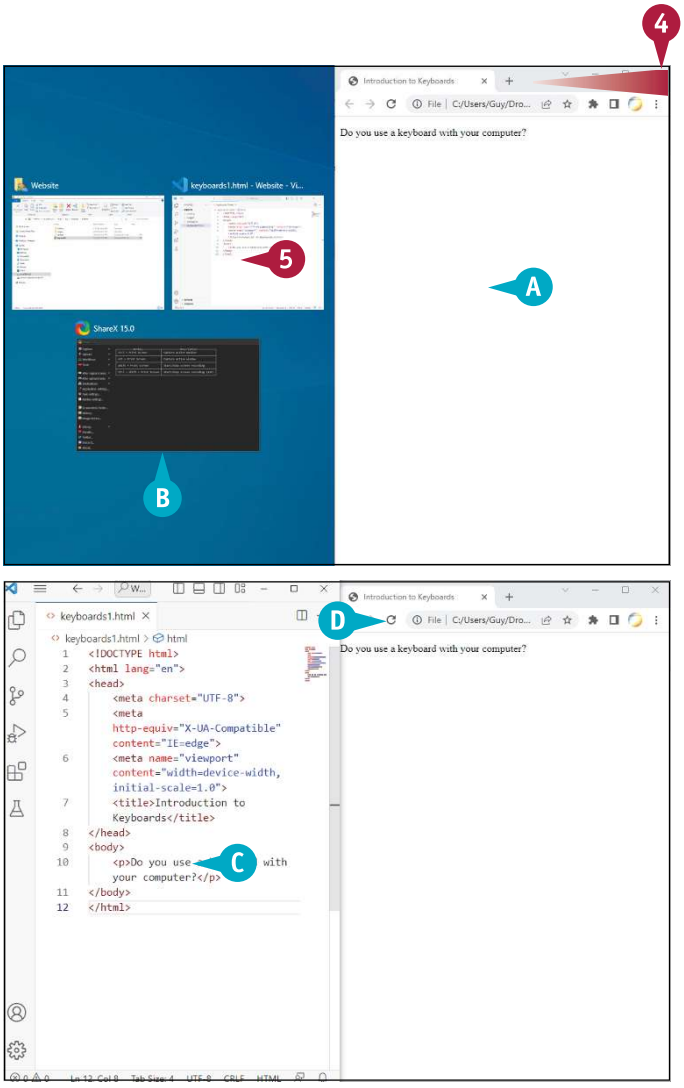
B. Windows Snap displays a thumbnail for each other open window you can place in the other half of the screen.

5. Click the Visual Studio Code window.

C. Windows Snap positions that window in the other half of the screen.

You can now make changes in the Visual Studio Code window.

D. Click **Refresh** (such as ↻ in Google Chrome) to refresh the web page to make it show your latest changes.



TIP

Which is the best browser to use for viewing pages as I work?

Any of the major browsers — Google Chrome, Microsoft Edge, Mozilla Firefox, Apple's Safari — will work fine. To ensure that your pages work consistently, rotate through a variety of browsers to view the pages. The easiest way to switch among browsers is to right-click the file in a File Explorer window, click or highlight **Open With**, and then click the browser you want to use this time.

Add Headings and Text

Most web pages benefit from having a structure that uses different heading levels. HTML provides six levels of headings, with `h1` being the highest level of heading and `h6` being the lowest level. Two or three levels of headings are often enough.

In this section, you add three levels of headings to your web page, with paragraphs of regular text separating the headings. The screens in this section show the Visual Studio Code window and the Google Chrome browser window tiled vertically using Windows Snap. You can either follow this arrangement or arrange the windows however best suits you.

Add Headings and Text

1. In Visual Studio Code, with your web page open, select the whole of line 10, the one paragraph in the `body` element.
2. Type **h1** to start creating an opening `<h1>` tag using an Emmet abbreviation:

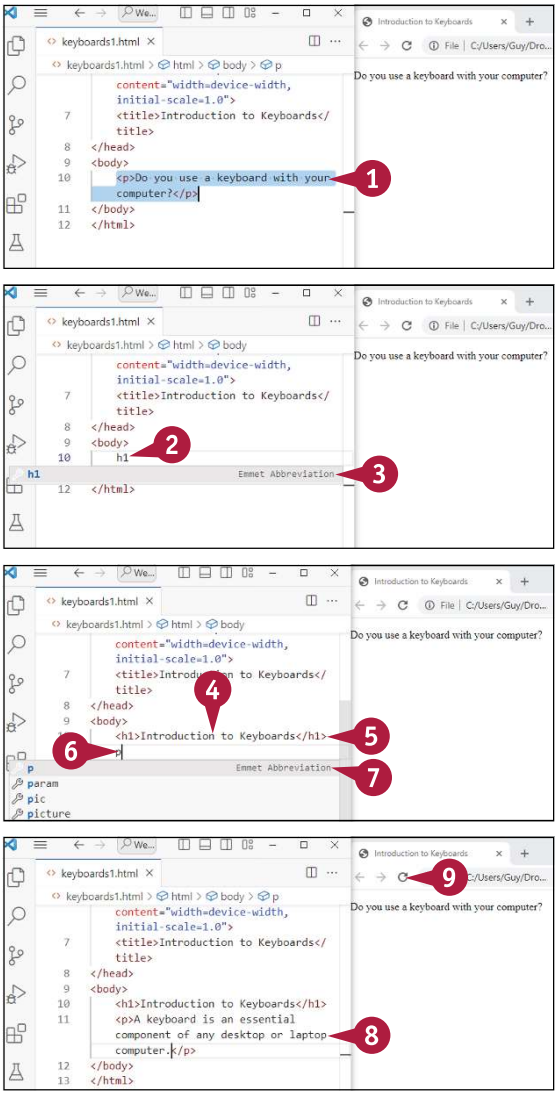
   `h1`

   The pop-up menu opens, with the h1 Emmet abbreviation item selected by default.
3. Click **h1**, or simply press Tab.

   Visual Studio Code expands the abbreviation to the full tag pair, the opening `<h1>` tag and the `</h1>` tag, placing the insertion point between the two.
4. Type the text of the top-level heading.
5. Click after the closing `</h1>` tag, and then press Enter to create a new line.
6. Type **p** to start creating an opening `<p>` tag using an Emmet abbreviation.

   The pop-up menu opens, with the p Emmet abbreviation item selected by default.
7. Click **p** or press Tab.

   Visual Studio Code expands the abbreviation to the full tag pair.
8. Type a paragraph of body text.
9. Click **Refresh** (↻).



Google Chrome displays the changes made to the web page.

A The first-level heading stands out clearly.

10 Click in the beginning of line 12.

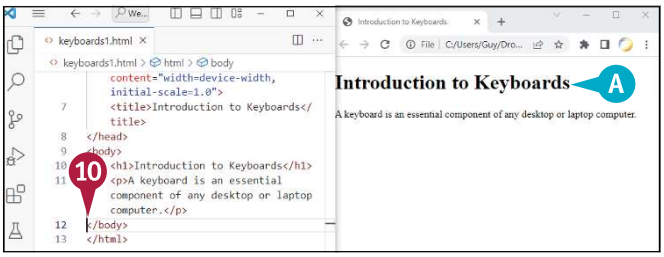
11 Type some text that includes second-level headings, between `<h2>` and `</h2>` tags, and body paragraphs, between `<p>` and `</p>` tags — for example:

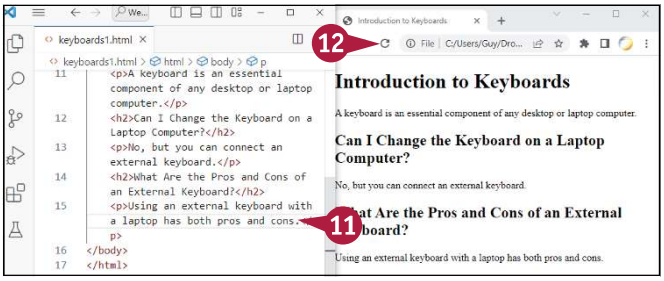
```
<h2>Can I Change the Keyboard
on a Laptop Computer?</h2>
<p>No, but you can connect an
external keyboard.</p>
<h2>What Are the Pros and Cons
of an External Keyboard?</h2>
<p>Using an external keyboard
with a laptop has both pros and
cons.</p>
```

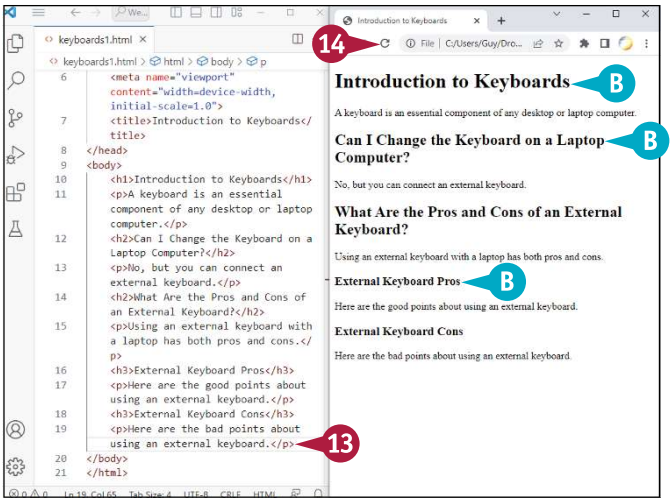
12 Click **Refresh** (↻) to refresh the browser.

13 Continue the page by typing some text that includes third-level headings, between `<h3>` and `</h3>` tags, and further body paragraphs — for example:

```
<h3>External Keyboard Pros</h3>
<p>Here are the good points
about using an external
keyboard.</p>
<h3>External Keyboard Cons</h3>
<p>Here are the bad points about
using an external keyboard.</p>
```







14 Click **Refresh** (↻) to refresh the browser.

B You can clearly see the three levels of headings.

TIP

How do I select text in Visual Studio Code?

Visual Studio Code enables you to use most standard means of selection. For example, you can hold down Shift and press ←, →, ↑, or ↓ to select text with the keyboard. You can click and drag with the pointer; or you can click at the beginning of what you want to select and then press Shift+click at the end. You can double-click to select a word or triple-click to select a whole paragraph.

Nest One Element Within Another Element

As you have seen already in this book, HTML enables you to place an element within another element. For example, apart from the `DOCTYPE` declaration at the beginning, the `html` element contains the whole of a web page: Within the `html` element are the `head` element and the `body` element, each of which contains other elements.

Placing one element inside another element is called *nesting* an element. When you nest an element inside another element, it is important that you close the nested element before closing the element in which it is nested.

Grasp How Nesting Works

To nest one element within another element, first enter the opening tag and closing tag for the containing element. For example, the `body` element of a web page uses the opening `<body>` tag and the closing `</body>` tag:

```
<body></body>
```

You can then nest an element between the opening and closing tags. For example, you might nest a paragraph, as shown here.

```
<body><p>Here is a paragraph.</p>
</body>
```

If you are going to nest many elements, you will probably want to place the outer element's tags on separate lines so that you can easily see where the element starts and ends. Indenting the nested elements helps distinguish them visually without changing the semantic meaning of the HTML, because browsers ignore the indentation. For example:

```
<body>
    <h1>Here Is a First-Level Heading</h1>
    <p>Here is a paragraph.</p>
</body>
```

The key to nesting elements is that you must close a nested element before you close the element that contains it. If you close the containing element before the nested element, errors occur, and the HTML is not valid. The following example closes the `body` element before closing the paragraph:

```
<body>
    <h1>Here Is a First-Level Heading</h1>
    <p>Here is a paragraph.
</body>
</p>
```

This example causes an error such as *Saw an end tag after* `body` *had been closed*.

CHAPTER 2

Add Comments

HTML enables you to include comments. A *comment* is text that appears in the HTML source code but that a browser does not display as part of the web page. Adding comments can be helpful, both as reminders while creating a web page and when you want to document it for others or yourself.

You can also use comments to hide existing elements of a web page, preventing them from being displayed — for example, if something isn't working. This is typically a temporary fix. You would normally remove comments before making a web page available online.

Add Comments

1. Open your web page in Visual Studio Code and in your browser.
2. Click where you want to insert the comment.
3. Type **<!--** to begin a comment tag.

```
<!--
```

A Visual Studio Code automatically inserts `-->` to close the tag:

```
<!--  -->
```

4. Type the comment text — for example:

```
<!-- Here is a comment. -->
```

B The comment appears in a different color in Visual Studio Code.

5. Click **Refresh** (↻).

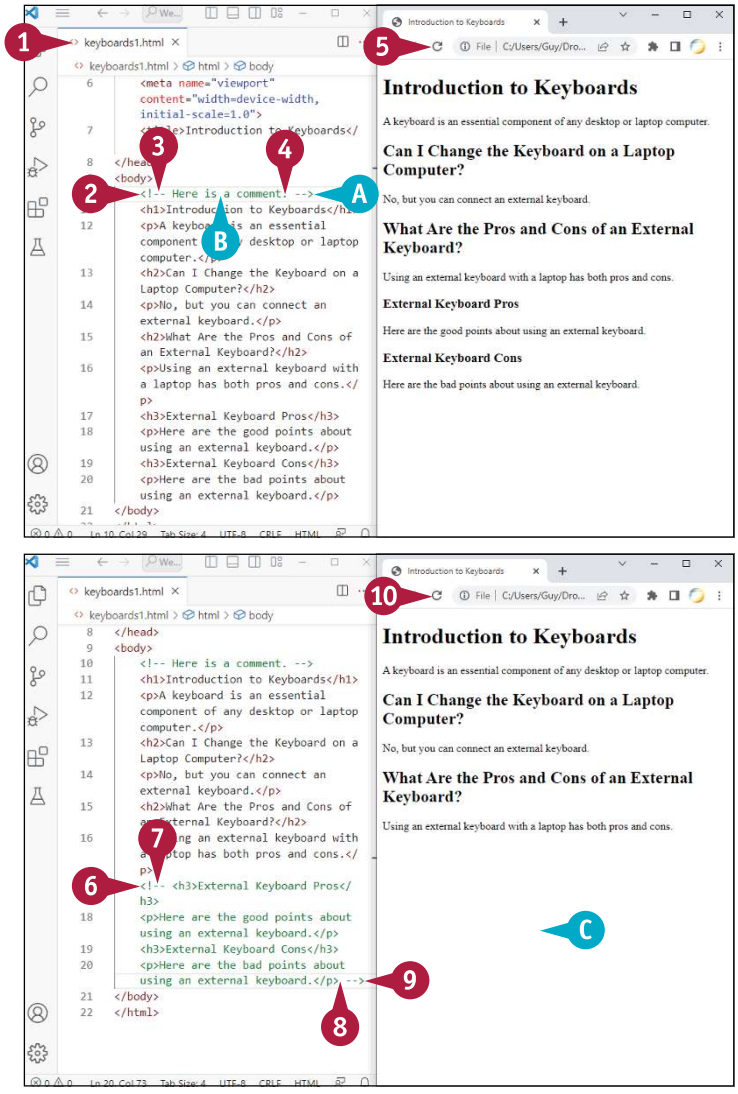
   The comment does not appear in the browser.

6. Click before the first element you want to hide.
7. Type **<!--** to begin a comment tag.
8. Click at the end of the last element you want to hide.
9. Type **-->** to close the comment tag.

Note: A comment can span multiple lines.

10. Click **Refresh** (↻).

C The elements enclosed by the comment no longer appear.



Apply Direct Formatting

HTML gives you various ways to format the text and objects in your web pages. Direct formatting is the most straightforward way of changing the appearance of text or an object. For example, you can apply attributes such as boldface, italic, or underline to specific parts of your text by putting the appropriate tags around them.

Direct formatting is easy and effective, but it is not efficient. Formatting via CSS is much more efficient and allows easier updating. However, you should recognize and understand direct formatting, because you will likely encounter it on many web pages.

Apply Direct Formatting

1. Open your web page in Visual Studio Code and in your browser.
2. Click before a word to which you want to apply boldface.
3. Type the opening `<b>` tag:

   ```
   <b>
   ```

4. Click after the word and type the closing `</b>` tag, so the tags enclose the word like this:

   ```
   <b>keyboard</b>
   ```

5. Click **Refresh** (↻).

   A. The word appears in boldface in the browser.

6. Click before some text to which you want to apply italic and underline.
7. Type the opening `<i>` tag and the opening `<u>` tag:

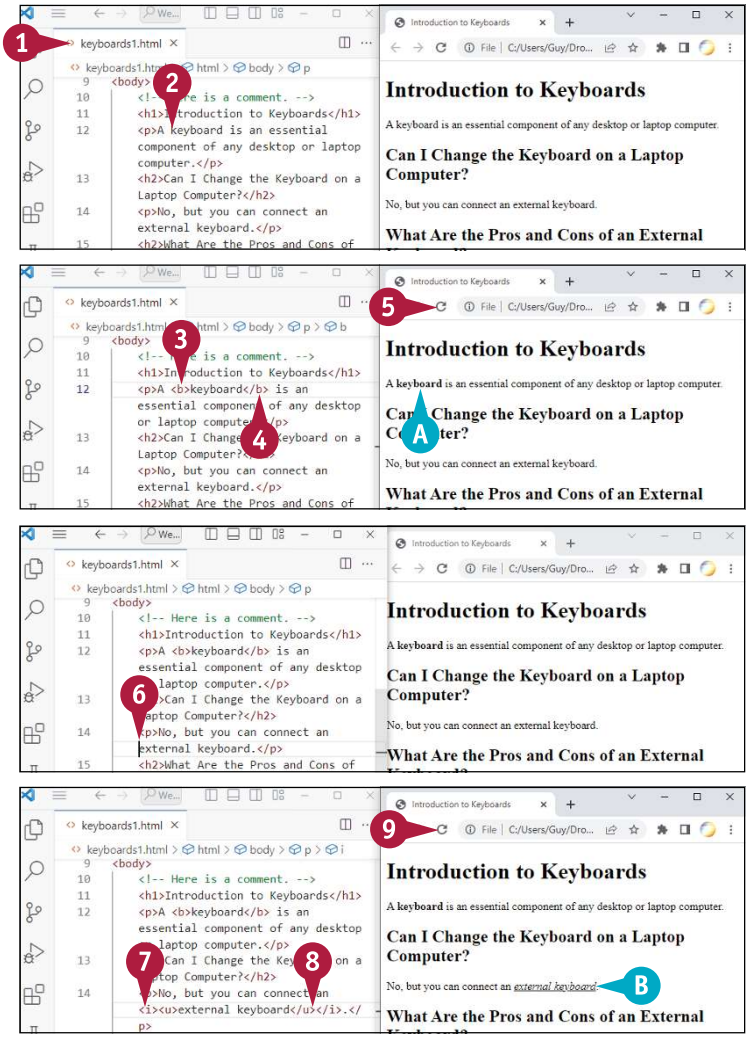
   ```
   <i><u>
   ```

8. Click after the text and type the closing `</u>` tag and the closing `</i>` tag.

Note: Close the nested `<u>` tag before closing the `<i>` tag that contains it.

9. Click **Refresh** (↻).

   B. The text appears in underlined italics in the browser.



CHAPTER 2

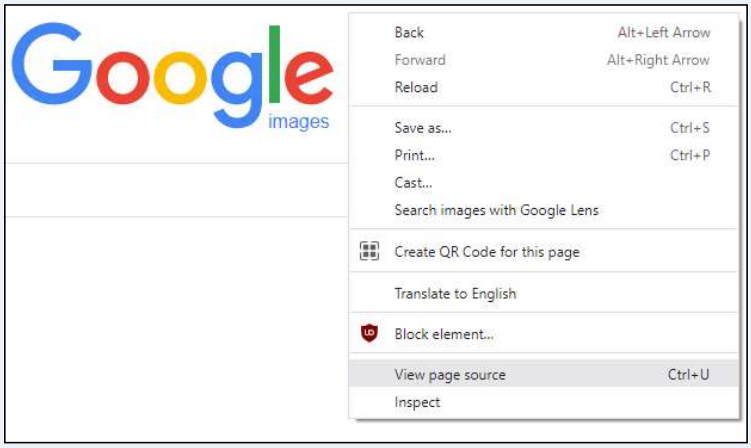
View a Page's Source Code

Normally, when you open a web page in a browser, the app displays the results of the HTML — for example, a bright, graphical web page. But sometimes you may want to view a page's source code instead. Viewing the source code can help you understand how a particular element is implemented in HTML or let you identify a problem that prevents a page from rendering correctly.

The techniques for viewing a page's source code vary from browser to browser. This section shows you how to view source code in Google Chrome, Firefox, Microsoft Edge, and Safari.

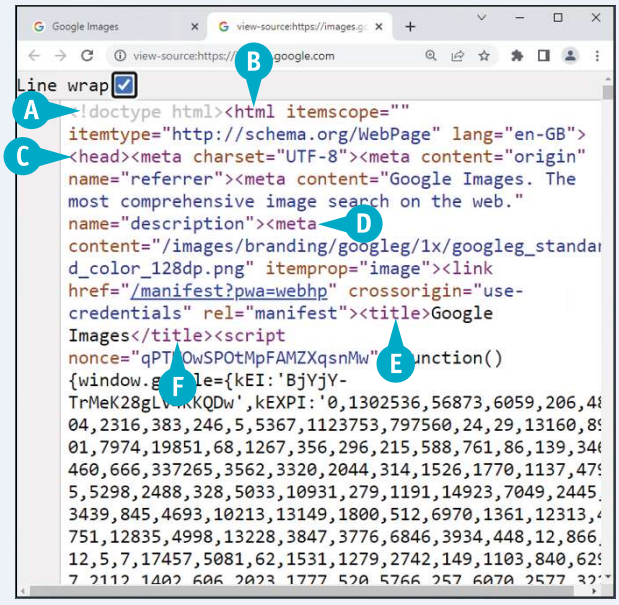
Display a Page's Source Code in a Browser

In Google Chrome, Mozilla Firefox, and Microsoft Edge, you can display a page's source code by right-clicking the page and then clicking **View page source** on the contextual menu.



On macOS, Safari makes life more difficult. First, you must add the Develop menu to the menu bar. To do so, click **Safari** and **Preferences** to open the Safari Preferences window; click **Advanced** to display the Advanced tab; and then select **Show Develop menu in menu bar** (☑). You can then click **Develop** on the menu bar and click **Show Page Source**; alternatively, Control-click or right-click the page and click **Show Page Source** on the contextual menu.

The source code for a complex page may seem overwhelming, but if you look at the beginning, you can recognize standard elements such as the `DOCTYPE` declaration (A), the opening `<html>` tag (B), the opening `<head>` tag (C), assorted `<meta>` tags (D), the opening `<title>` tag (E), and the closing `</title>` tag (F).



Validate a Web Page

To make sure HTML is correct, you can validate it. Various methods of validating HTML are available. For example, some code editors, integrated development environments, and web-authoring tools have built-in validation features.

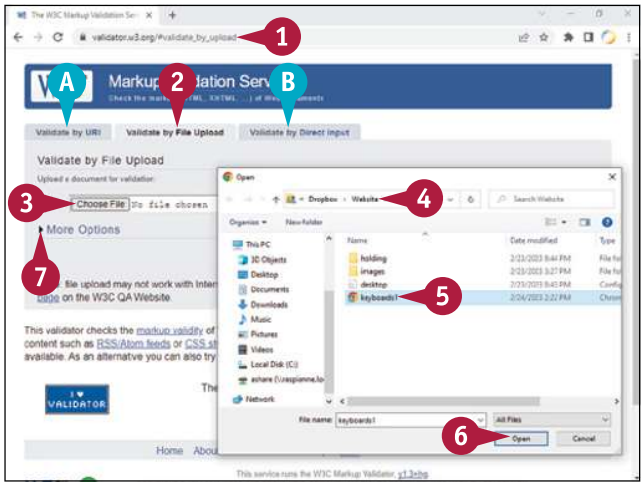
This section shows you how to validate HTML using the W3C Markup Validation Service, an online resource provided by the World Wide Web Consortium, W3C, the main international standards organization for the World Wide Web. You can validate by uploading a file, as shown here; by providing the URI of the web page to check; or by simply pasting or typing code into a box.

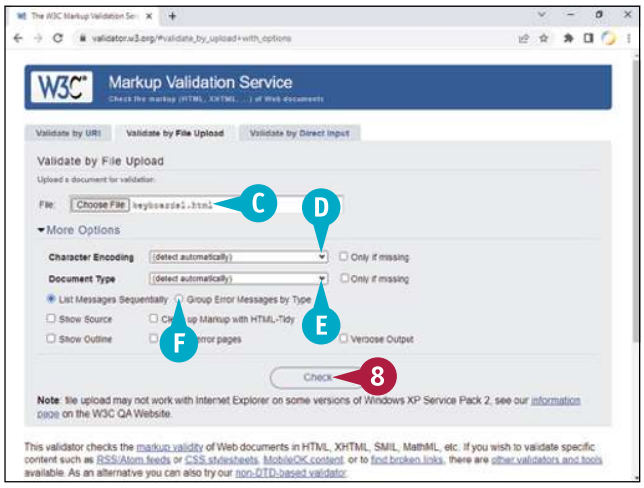
Validate a Web Page

1. Open a browser window to `https://validator.w3.org`.
2. Click **Validate by File Upload** to follow this example.
   - A You can click **Validate by URI** to validate a web page that is already online.
   - B You can click **Validate by Direct Input** to validate code you paste — or type — into a box.
3. Click **Choose File**.

   The Open dialog box appears.
4. Navigate to the appropriate folder.
5. Click the file.
6. Click **Open**.
7. Click **Expand** (▶ changes to ▼) next to More Options.

   The More Options section expands.
   - C The filename appears in the File box.
   - D You can click **Character Encoding** (⌄) and specify a particular character encoding instead of "(detect automatically)."
   - E You can click **Document Type** (⌄) and specify a particular document type instead of "(detect automatically)."
   - F You can click **Group Error Messages by Type** (◯ changes to ◉) to group error messages. The List Messages Sequentially setting is the default.





8. Click **Check**.

The Markup Validation Service checks the HTML in the file.

G The error list identifies errors found in the HTML.

H The file contains the tag `<bdy>`, a misspelling of the `<body>` tag that encloses the `body` element of an HTML document.

I Because the `<body>` tag is missing, the closing `</body>` tag also causes an error.

J Another error occurs because there is no `</bdy>` closing tag to close the `<bdy>` tag.

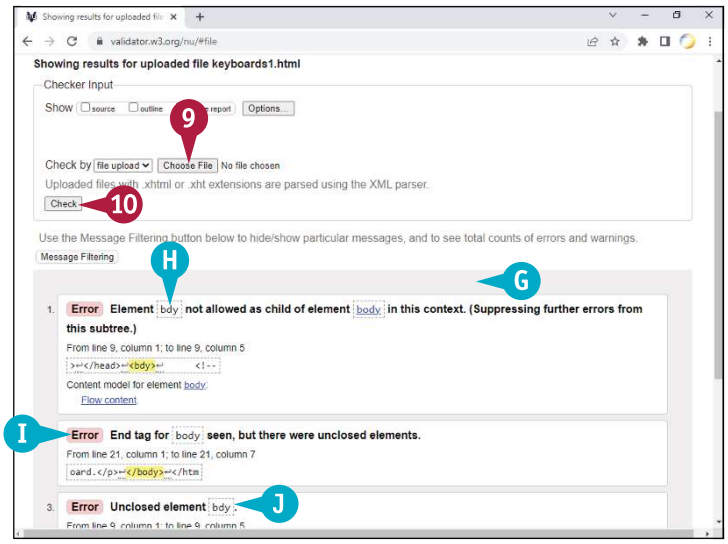
9 Looking at the errors, correct the HTML, click **Choose File**, and upload the corrected file.

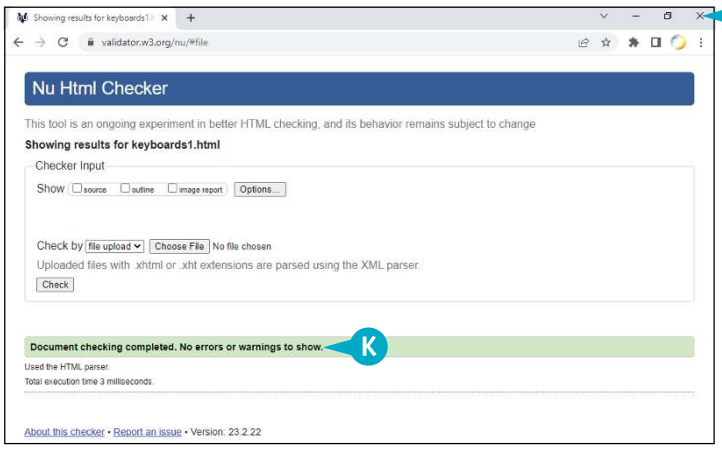
10 Click **Check**.

The Markup Validation Service checks the HTML in the file.

K The *Document checking completed. No errors or warnings to show.* message indicates that the file has been validated.

L You can click **Close** ( ) to close the browser window.





TIP

Why does the Markup Validation Service show errors for an HTML file that displays correctly in a browser?

The Web contains massive amounts of faulty HTML, so browsers are built to be tolerant of errors and to display web pages as well as they can. This means that viewing a web page in a browser is not an effective check of whether the page's HTML is valid. To ensure that your web pages display correctly in all browsers, be sure to validate every page's HTML and remove any errors.

Create Another Web Page

In this section, you create a second web page in Visual Studio Code, using the same techniques as for the first page you created earlier in this chapter. You create this page to provide a destination for a hyperlink you create later in the chapter. The page does not need to be complex; it just needs to be present so that you can establish the link to it from the previous page.

Create Another Web Page

1. In Visual Studio Code, click **Explorer** ().

   The primary sidebar opens, showing the Explorer pane.

2. Move the pointer over the folder's heading and click **New File** ().

   An edit box appears.

3. Type the filename, including the `.html` file extension, and then press Enter.

   The example uses the filename `keyboard_types.html`.

   Visual Studio Code creates the file and displays it in the main pane.

4. Type `!`.

   The AutoComplete list opens, showing available AutoComplete entries that start with the character !.

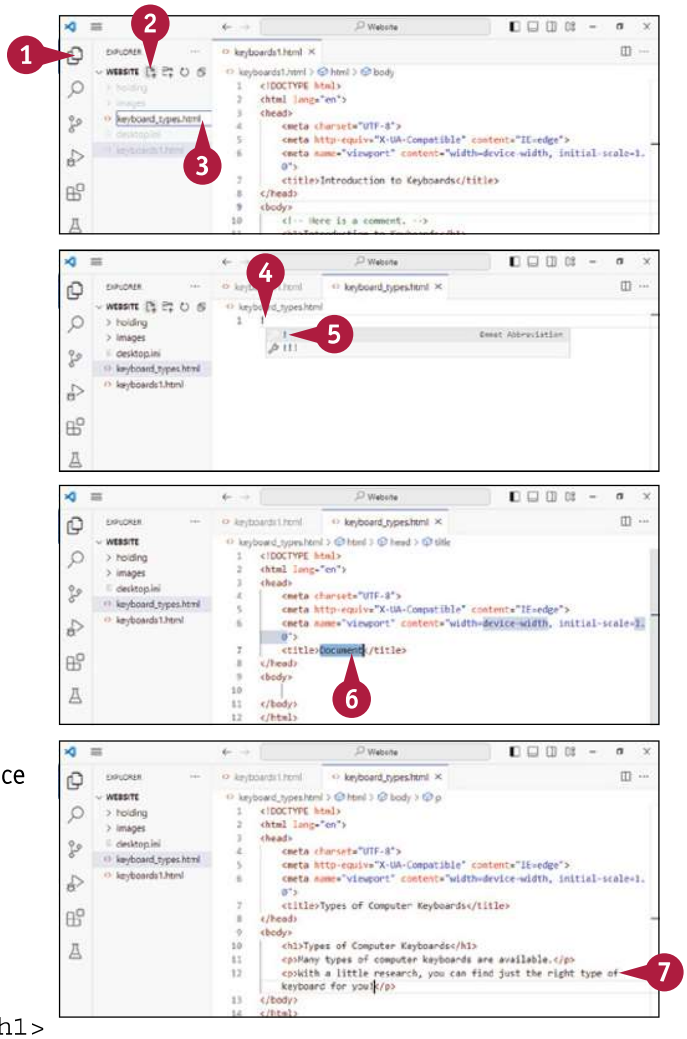
5. Click the Emmet abbreviation item.

   Visual Studio Code expands the abbreviation, entering the skeleton code of a web page in place of the exclamation point.

6. With the default title text, *Document*, selected, type the web page's title.

7. Click before the `</body>` tag and enter some straightforward HTML. Here is an example:

```
<h1>Types of Computer Keyboards</h1>
<p>Many types of computer
keyboards are available.</p>
<p>With a little research, you
can find just the right type of
keyboard for you!</p>
```



CHAPTER 2

Understanding the Essentials of Hyperlinks

The Web uses *hyperlinks*, often called simply *links*, to create connections between different web pages or parts of pages. Clicking a link on one page causes the browser to display the linked page, either in the same browser window or tab or in a new browser window or tab.

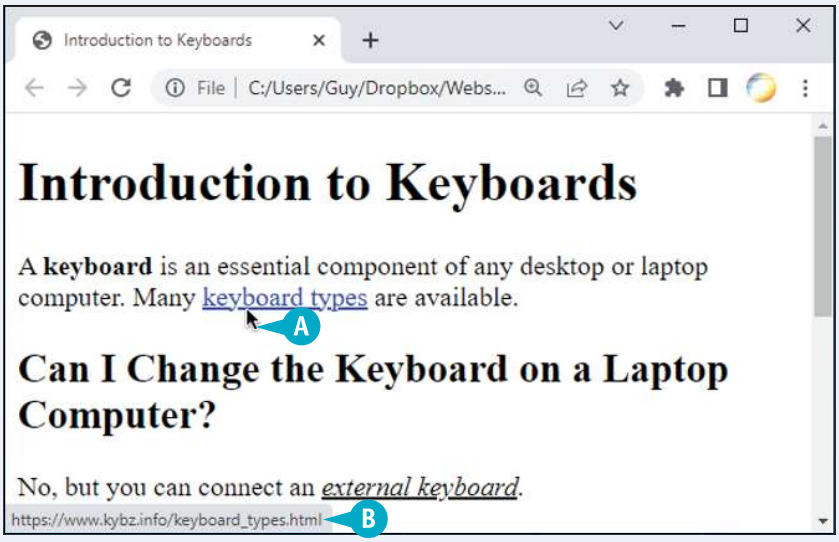
In this section, you learn the essentials of hyperlinks and identify the HTML elements that make up a hyperlink. In the following sections, you create first straightforward hyperlinks and then hyperlinks that redirect the browser to a different destination.

Identify the Components of a Hyperlink

In HTML, a hyperlink uses an `<a>` tag, where the letter *a* stands for *anchor*. The `<a>` tag for a hyperlink uses the following format:

```
<a href="address">link text</a>
```

Here, the `href` — hyperlink reference — attribute specifies that the anchor tag contains a hyperlink. The link's address, entered in double quotation marks, specifies the destination for the hyperlink. The link text, which appears between the opening `<a>` tag and the closing `</a>` tag, provides the text to display on the web page to indicate the link.



For example, the following hyperlink displays the text *keyboard types* and links to the page named `keyboard_types.html` at the site `www.kybz.info`.

```
<a href="https://www.kybz.info/keyboard_types.html">keyboard types</a>
```

The nearby illustration shows the linked word in context. When you hold the pointer over a link (A), most browsers display the link destination. In this example, Google Chrome displays the link destination in the lower-left corner (B) of the window.

Create a Hyperlink Between Your Web Pages

Now that you have created two web pages, you can create hyperlinks between them. To create a hyperlink, you insert an anchor tag at the appropriate place in the web page, specify the destination for the link, and enter the text to display on the page to represent the link.

The example in this section creates a hyperlink between files stored on your computer rather than files stored on a web server.

Create a Hyperlink Between Your Web Pages

1. In Visual Studio Code, click **Explorer** ( ).

   The primary sidebar opens, showing the Explorer pane.

Note: If the folder for your website is collapsed, click **Expand** ( ) to expand it.

2. Click the web page in which you want to create the hyperlink.

   The web page appears in the main part of the Visual Studio Code window.

3. Click to place the insertion point where you want the link.

4. Type **a**.

   The AutoComplete list opens, showing available AutoComplete entries that start with the character *a*.

5. Click the first item, which is identified as an Emmet abbreviation.

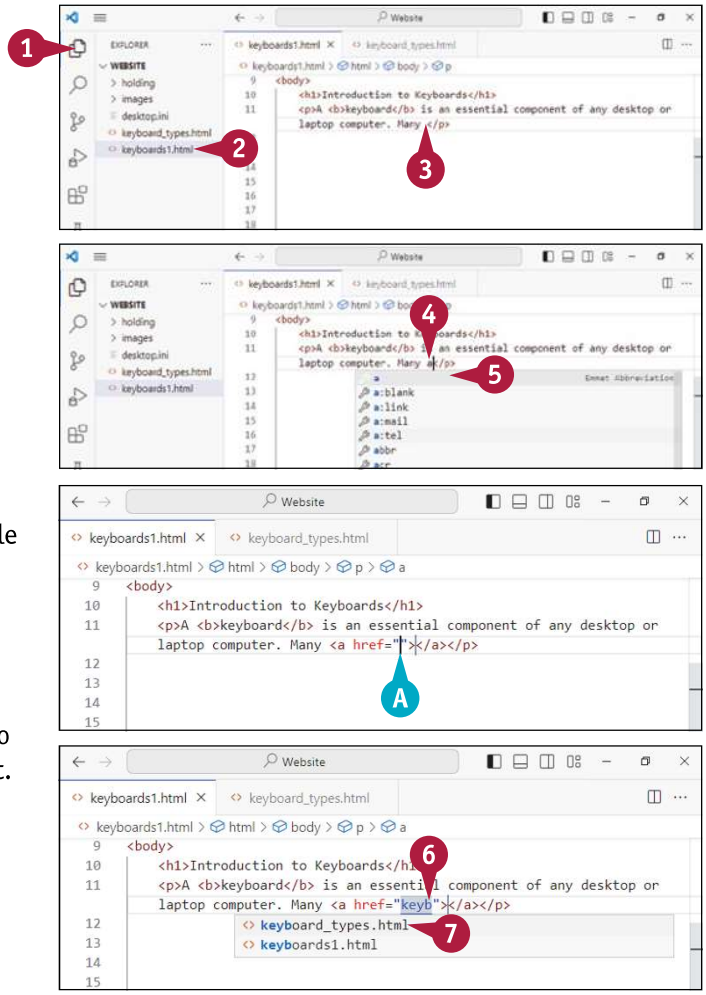
   Visual Studio Code expands the abbreviation to the anchor tags for a hyperlink, as shown next.

```
<a href=""></a>
```

A. Visual Studio Code places the insertion point between the double quotation marks.

6. Start typing the destination for the link between the double quotation marks. The example uses the page `keyboard_types.html`, so you would start typing **keyb**.

   The AutoComplete list opens, showing possible matches from your folders.



7. If a match is correct, click it.

   Otherwise, finish typing the destination.

B The destination appears in your code as blue underlined text to indicate it is a link.

8 Click between the > that closes the <a> tag and the < that opens the tag.

9 Type the text you want the page to display for the link. This example uses the words *keyboard types*.

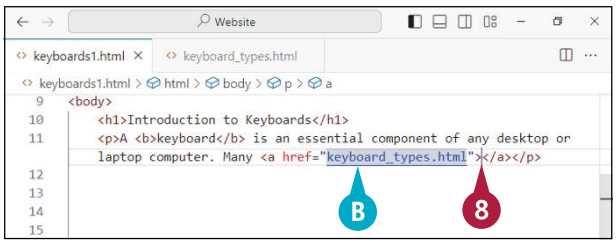
10 Type any text needed to complete the sentence.

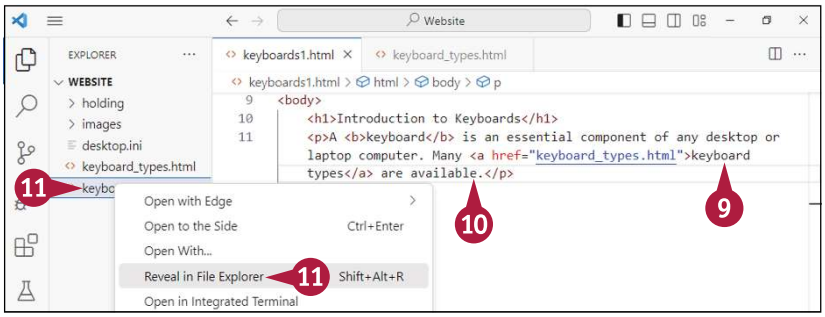
11 Right-click the page name in the Explorer pane and click **Reveal in File Explorer** to display the file in a File Explorer window. Then right-click the file, click or highlight **Open With**, and click the browser.

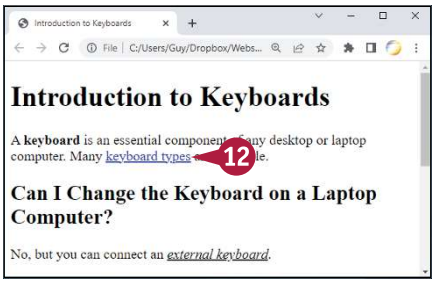
The web page opens in your chosen browser.

12 Click the link.

The linked page appears.







TIP

What other types of hyperlinks can I create using HTML?

Apart from the straightforward type of link shown in this section, you can create links that open new browser tabs or windows, links that redirect the browser to a different destination, and links that start email messages. See Chapter 5 for in-depth information on working with links.

Interpret HTTP Status Codes

HTTP uses five categories of status codes for tracking the interactions between web servers and clients. First, an *information response* occurs when the server receives the request and is working on fulfilling it. Second, a *successful response* indicates the server can fulfil the client's request. Third, a *redirection message* means that the server's response involves redirecting the client's request. Fourth, a *client error response* occurs when the server detects a problem with the client's request. Fifth, a *server error response* happens when the server encounters a problem fulfilling the request. You can use these codes for identifying and troubleshooting problems.

HTTP Status Codes and Their Meanings

Table 2-1 shows the HTTP status code responses you are most likely to encounter in your web browsing and development.

Table 2-1: HTTP Status Codes and Their Meanings

HTTP Code	Code Status	Explanation
		Information Responses
100	Continue	The client should continue the request if it has not been completed; if the request has been completed, the client should ignore this code.
101	Switching Protocols	The server is switching to the specified protocol following an Upgrade request from the client.
102	Processing	The server is processing a WebDAV request but has no response yet.
		Successful Responses
200	OK	The file request completed successfully.
201	Created	A POST request or PUT request succeeded, creating a new resource, such as a web page.
202	Accepted	The server has received the request but not yet acted on it.
204	No Content	The server has no content to send for the request but is returning the headers in case they are useful.
205	Reset Content	The server instructs the user agent to reset the document that sent the request.
206	Partial Content	The server is returning a response to a Range header that requests only part of a resource.
		Redirection Messages
300	Multiple Choices	The server has multiple resources for the request, and the client needs to pick one.
301	Moved Permanently	The requested URL has been permanently changed to the new URL the server is returning.

HTTP Code	Code Status	Explanation
302	Found	The server has found the requested resource at a different URL, but the client should continue to use the original URL because the change is supposedly temporary.
303	See Other	The server is directing the client to send a GET request to a different URL.
307	Temporary Redirect	Same as for 302, but the client must use the same HTTP method, such as POST, for the new request.
308	Permanent Redirect	Same as for 301, but the client must use the same HTTP method, such as POST, for the new request.
		Client Error Responses
400	Bad request	The client request is incorrectly formatted or contains deceptive routing.
401	Unauthorized	The client must authenticate itself to access the page.
403	Forbidden	The client does not have permission to access the page.
404	Not Found	The server cannot find the page the client requested. Some servers send a 404 error instead of a 403 error to obscure the fact that the page exists but the client is forbidden to access it.
426	Upgrade Required	The server refuses the request with the protocol the client used but may fulfil the request if the client upgrades to the specified protocol.
429	Too Many Requests	The server is limiting the client because the client has sent too many requests in a given time period.
451	Unavailable for Legal Reasons	The server refuses the request because it cannot legally provide the content; for example, because the content is *geofenced* — restricted — to a specific area that the client is outside.
		Server Error Responses
500	Internal Server Error	The server has suffered an error it cannot resolve.
501	Not Implemented	The server does not accept the request method used. Servers must accept GET requests and READ requests.
502	Bad Gateway	The server attempted to relay the request to another server but received an invalid response.
503	Service Unavailable	The server cannot fulfil the request because it is overloaded or the server from which it would get the information is down.
504	Gateway Timeout	The server is acting as a gateway, relaying the client's request to another server, but has not received a response from that server in the specified period.