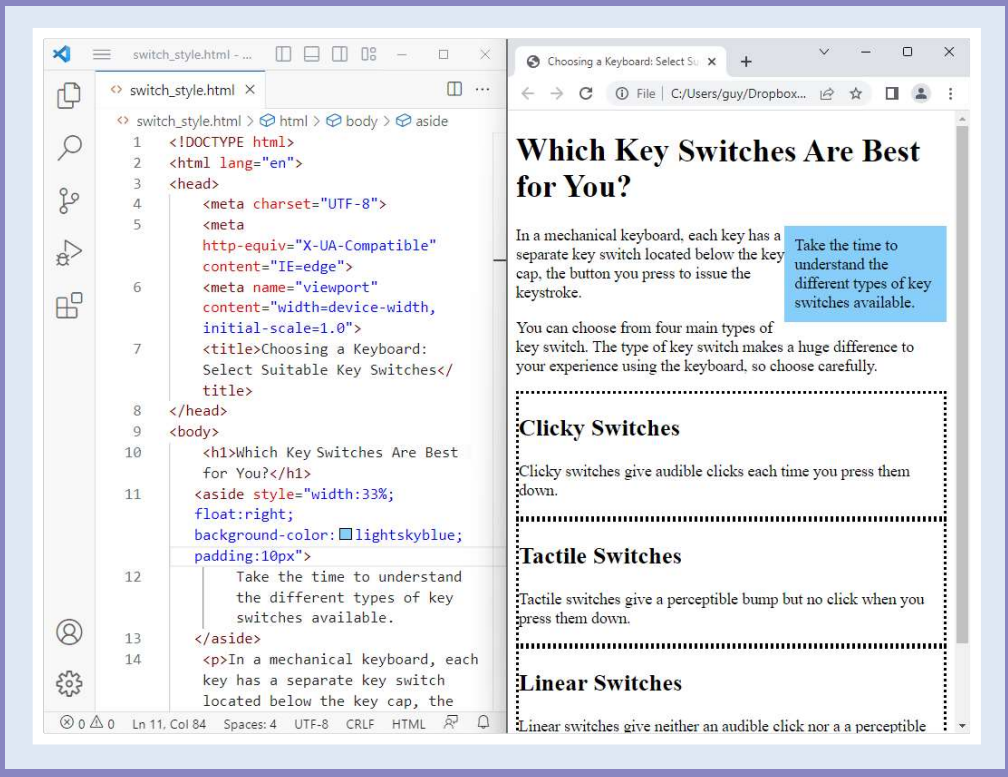
CHAPTER 3

Structuring a Web Page

In this chapter, you learn to structure a web page by using semantic elements, which are elements whose names explain their purpose, such as the `header` element and the `article` element. You also learn how to use the nonsemantic `span` and `div` elements to select parts of a page.



Meet the Elements for Structuring Web Pages

HTML enables you to use a wide variety of elements to structure your web pages. Some elements, such as the `header` element and the `footer` element, are *semantic*, which means their names clearly express their roles: The `header` element goes at the top of a web page or of another element, and the `footer` element goes at the bottom. Other elements are *nonsemantic*, meaning that their names do not clearly express their roles; for example, the `span` element specifies a short section of text, and the `div` element specifies a longer section, without expressing what part of the page those sections represent.

Grasp Semantic and Nonsemantic Elements

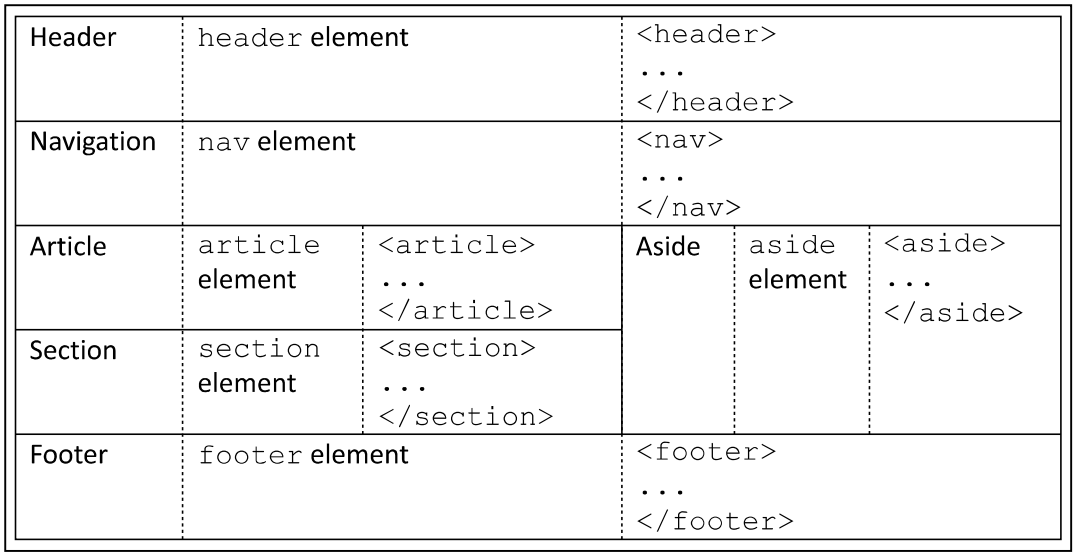
To structure your web pages, you will use semantic elements, such as the `header` element, the `figure` element, and the `section` element. To format your web pages, you will use both semantic elements and nonsemantic elements, such as the `span` element and the `div` element.

Table 3-1 explains the most useful nonsemantic elements and semantic elements.

Table 3-1: Nonsemantic Elements and Semantic Elements

Element	Explanation
	Nonsemantic Elements
`span`	Selects part of a paragraph or other short element.
`div`	Selects one or more paragraphs or other elements.
	Semantic Elements
`article`	Contains an "article," a self-standing part of the web page. For example, a web page may contain multiple `article` elements, each containing a separate topic.
`aside`	Contains a usually small amount of content that is indirectly related to the nearby content or page.
`details`	Contains extra information that the reader can expand to read or collapse to hide. For example, a `details` element can act as a widget that can show or hide the information it contains.
`figcaption`	Contains the caption for a `figure` element. This element can be nested as either the first child element or the last child element in the `figure` element.
`figure`	Contains an illustration, such as a photo, a diagram, or a code listing.
`footer`	Contains information to be displayed at the bottom of a web page or a particular element, such as copyright information, contact information, or when the page was last updated.
`header`	Contains information to be displayed at the top of a web page or a particular element, such as a heading and introduction or navigational links.

Semantic Elements	
`main`	Contains the main content of the page. An HTML file can contain only one `main` element. The `main` element can contain elements such as `article`, `aside`, `footer`, `header`, and `nav`, but it cannot be placed inside any of these elements.
`mark`	Contains marked or highlighted text. By default, browsers display the `mark` element as black text on a yellow background.
`nav`	Contains a set of navigation links. The link destinations can be either within the page or outside it.
`section`	Contains a section of a web page. For example, if a page covers several topics at the `h2` level, you might create a `section` element for each topic.
`summary`	Contains a visible heading within the `details` element. The viewer can click the heading to display the details.
`time`	Contains a time or a date and a time. The `time` element has a `datetime` attribute that supplies a machine-readable time that search engines and browsers can use.



The above illustration shows a breakdown of a web page structured with semantic elements.

Select Items with `span` and `div` Elements

HTML's `span` and `div` elements enable you to specify just the amount of text you need so that you can format it. You typically use a `span` element to identify text within a paragraph or another short element. For example, you might use a `span` element to identify text to which you want to apply particular font formatting. Similarly, you use a `div` element to specify a "division" or particular section of text, usually consisting of one or more paragraphs. You could then apply formatting, such as a border, to the entire division.

Select Items with `span` and `div` Elements

Select Text with the `span` Element

Note: To work through this section, you may want to turn off Visual Studio Code's HTML Auto Closing Tags feature temporarily. See the first tip.

1. In Visual Studio Code, open the file you want to use.
2. Open the file in a browser window.
3. In Visual Studio Code, click to place the insertion point where you want to start the span.
4. Type the opening `<span>` tag, including the `style` attribute and formatting to apply the `red` color to the text:

```
<span style="color:red">
```

5. Type the text contents of the span.

Note: To use existing text in the `span` element, click to place the insertion point at the end of that text.

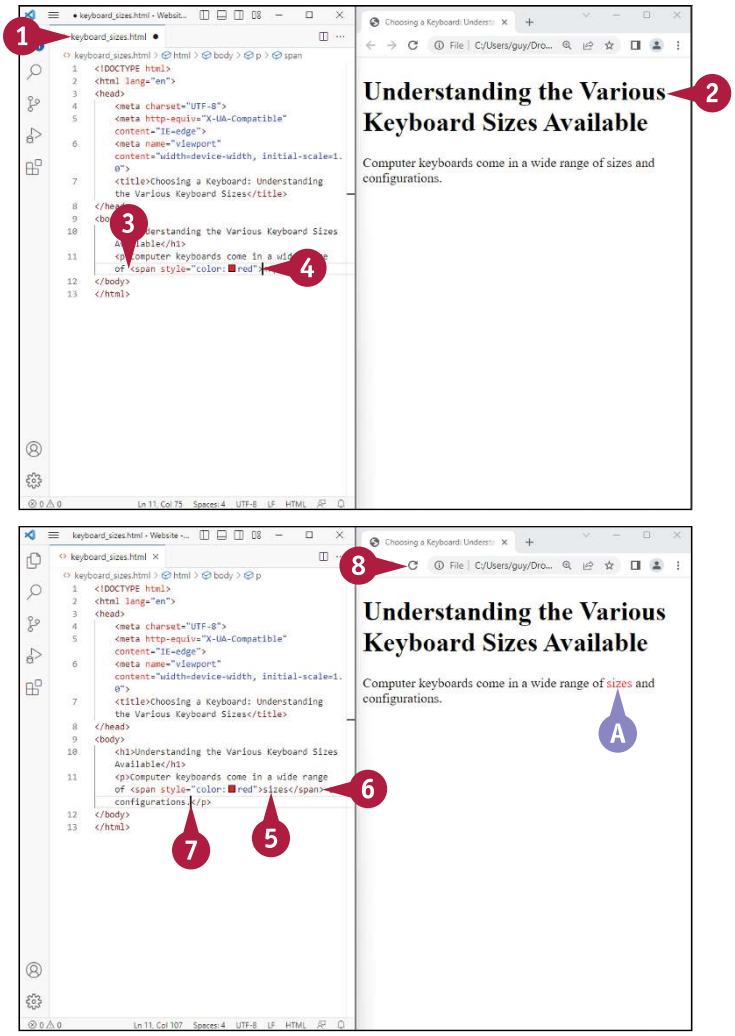
6. Type the closing `</span>` tag:

```
</span>
```

7. Type any text that should appear after the `span` element — for example, the rest of the paragraph.
8. Click **Refresh** (↻).

   The browser displays the updated web page.

A. The text in the `span` element appears in red.



Select Text with the `div` Element

1. Click where you want the `div` element to begin.
2. Type the opening `<div>` tag, including the `style` attribute and formatting to apply right alignment to the text:

```
<div style="text-align:right">
```

3. Press Enter twice, and then type the ending `</div>` tag:

```
</div>
```

4. Click to place the insertion point on the blank line.
5. Type an `h2` element and one or more `p` elements — for example:

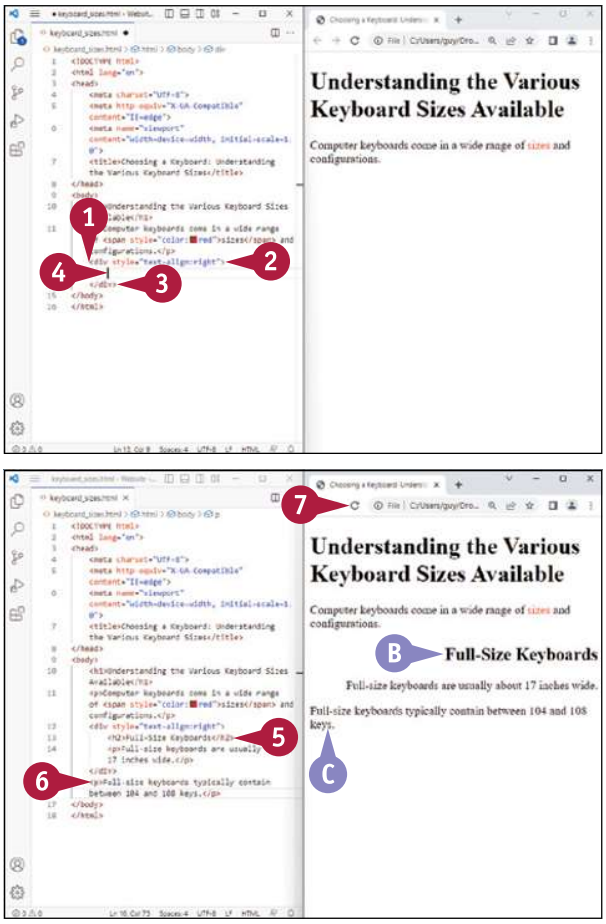
```
<h2>Full-Size Keyboards</h2>

<p>Full-size keyboards are usually
about 17 inches wide.</p>
```

6. Click after the closing `</div>` code, press Enter, and then type another `p` element — for example:

```
<p>Full-size keyboards typically
contain between 104 and 108 keys.</p>
```

7. Click **Refresh** (↻).

   The browser displays the updated web page.



B The text in the `div` element is aligned right.

C The text after the `div` element returns to left alignment, the default.

TIPS

How do I stop Visual Studio Code from inserting closing tags automatically?

Disable the Auto Closing Tags feature. Press Control+, on Windows or Linux, or press ⌘+, on the Mac, to display the Settings screen. Click **Search settings** and type **html closing**, and then deselect **Enable/disable autoclosing of HTML tags** (☐).

How else can I format my `span` elements and `div` elements?

Instead of applying style formatting inline, you can use external CSS to format your `span` elements and `div` elements, as explained in Chapter 8. This section formats the elements inline to help keep the example easy to follow. Using external CSS is faster, more efficient, and more flexible than using direct formatting.

Create `header` Elements and `footer` Elements

You can create a header for a web page by using the `header` element. A `header` element starts with the opening `<header>` tag and ends with the closing `</header>` tag; between them, you usually put one or more headings plus any introductory information the page needs. You might also use a header element to provide navigational links to different parts of a long web page.

Similarly, you can create a footer in HTML by using the `footer` element, which starts with the opening `<footer>` tag and ends with the closing `</footer>` tag.

Create `header` Elements and `footer` Elements

1. In Visual Studio Code, either create and save a new file, or open the existing file you want to use.
2. Open the file in a browser window so you can see the results of the changes you make.
3. In Visual Studio Code, click to place the insertion point where you want to start the header.
4. Type the opening `<header>` tag, including the `style` attribute to apply a dotted border:

```
<header style="border-style:
dotted">
```

5. Press Enter twice, and then type the closing `</header>` tag:

```
</header>
```

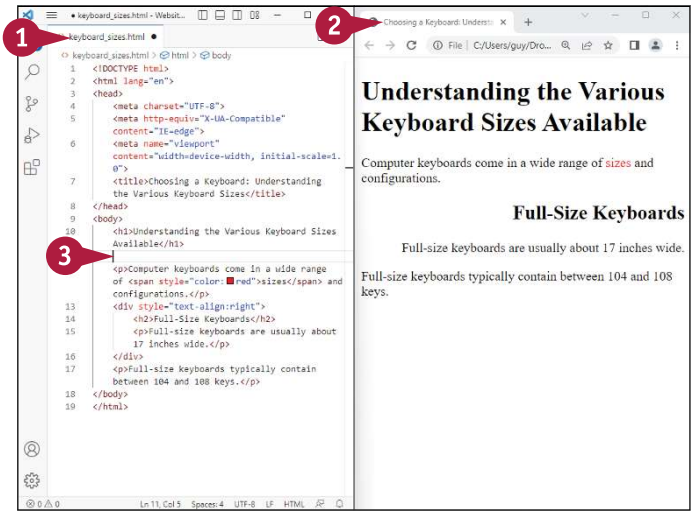
6. Click to place the insertion point on the blank line.
7. Type some content to display in the `header` element — for example:

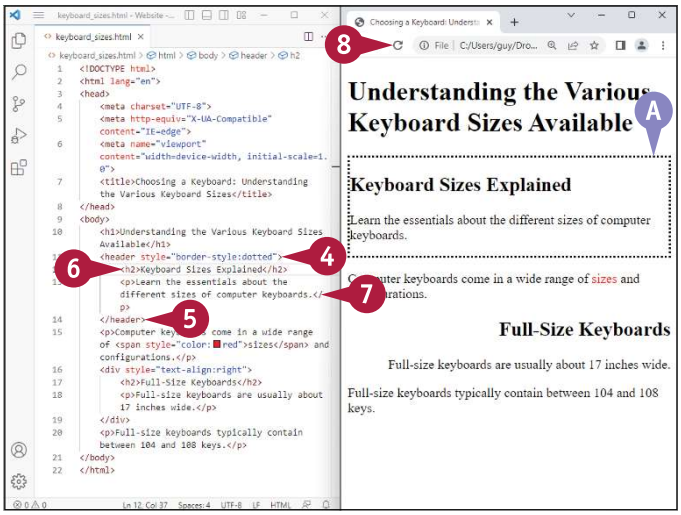
```
<h2>Keyboard Sizes Explained</h2>

<p>Learn the essentials about
the different sizes of computer
keyboards.</p>
```

8. Click **Refresh** ( ).

   The browser displays the updated page.

A The header appears with a dotted border.





9 Click where you want to start the footer.

Note: Normally, you would place the footer at the bottom of the web page.

10 Type the opening `<footer>` tag, including the `style` attribute to assign first the `background-color` property with the color `aqua` and then the `border-style` property with the type `solid`:

```
<footer style="background-color:aqua; border-style:solid">
```

Note: Separate the two properties with a semicolon.

11 Type some text for the `footer` element — for example:

```
<p>Copyright &copy; 2023 M. Jones Productions</p>
```

Note: `&copy;` is the HTML code for the copyright symbol, ©. See the section "Understanding HTML Entity Codes" in Chapter 8 for more information.

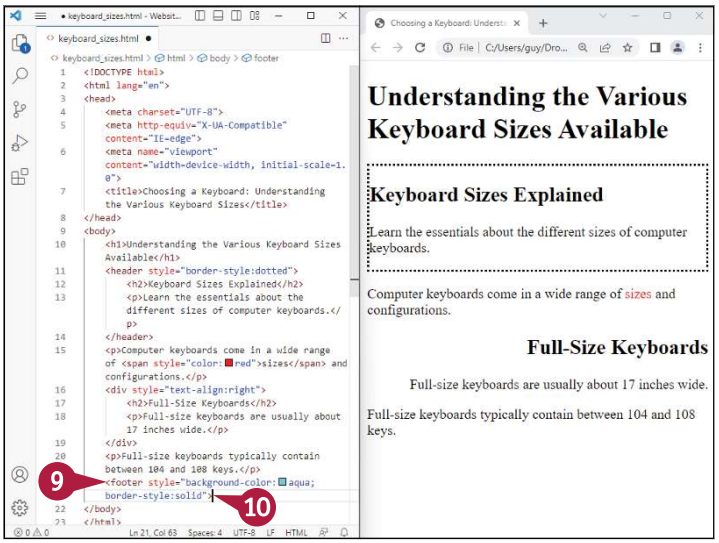
12 Type the closing `</footer>` tag:

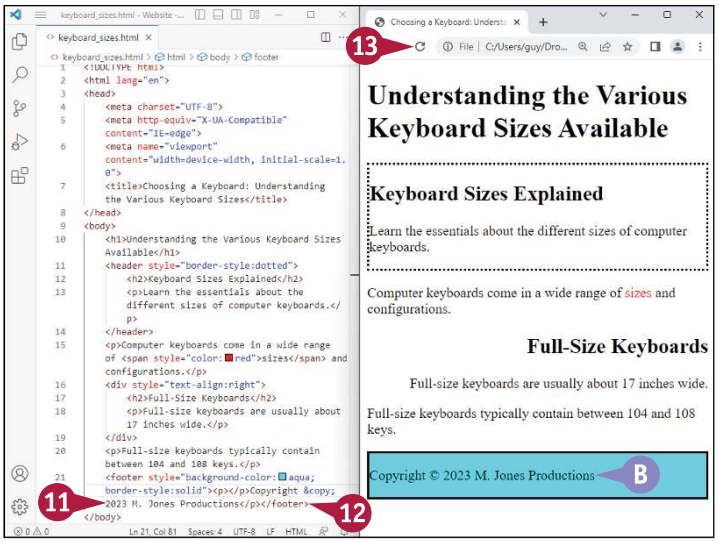
```
</footer>
```

13 Click **Refresh** ( ).

The browser displays the updated page.

B The footer appears with a solid border and an aqua background.





TIP

Can I create multiple `header` elements in a web page?

Yes — you can create as many `header` elements as you need. Each `header` element must be separate: You cannot nest one `header` element within another `header` element. You also cannot place a `header` element inside a `footer` element — as you would probably expect — or within an `address` element.

Similarly, you can create multiple `footer` elements, but you cannot place a `footer` element within a `footer` element, within a `header` element, or within an `address` element.

Add `article` Elements to a Page

When a page includes stand-alone content topics, you can use the `article` element to present those topics as logically separate articles.

An article can be whatever length and complexity the subject and coverage requires. The example articles in this section are very short because of the constraints of the book page.

Add `article` Elements to a Page

1. In Visual Studio Code, either create and save a new file, or open the existing file you want to use.
2. Open the file in a browser window so you can see the results of the changes you make.
3. In Visual Studio Code, click to place the insertion point where you want to start the first `article` element.
4. Type the opening `<article>` tag, including the `style` attribute with the value `border-style:dotted` to make the element's extent easy to see.

```
<article style="border-style:dotted">
```

5. Type the content for the article — for example:

```
<h2>Clicky Switches</h2>

<p>Clicky switches give audible clicks each time you press them down.</p>
```

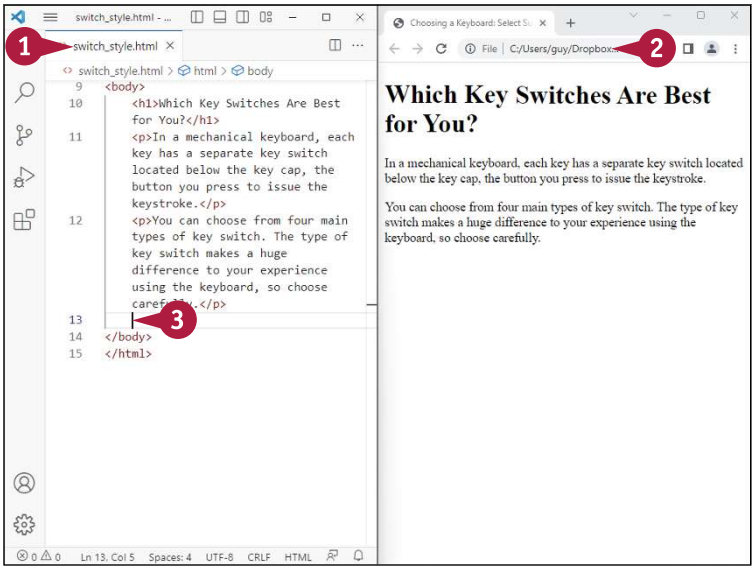
6. Type the closing `</article>` tag:

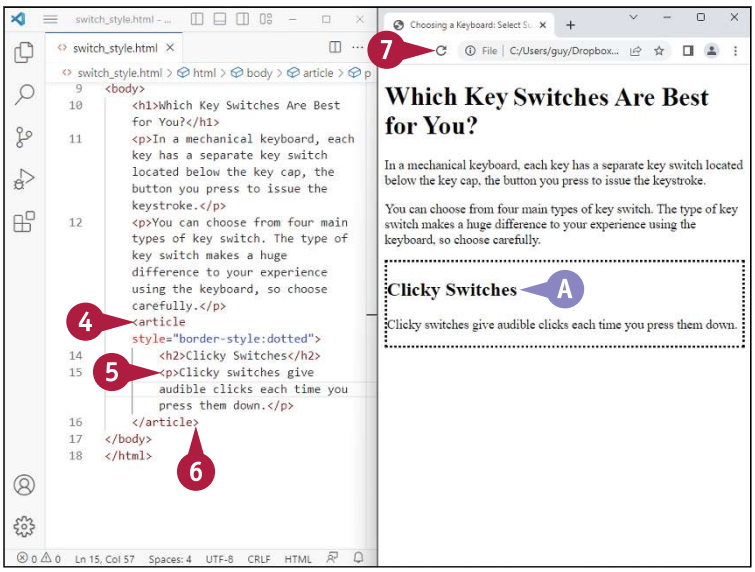
```
</article>
```

7. Click **Refresh** (↻).

   The browser displays the updated page.

A. The article appears with a dotted border.





8 Press Enter and type the opening `<article>` tag for another article, again specifying a dotted border:

```
<article style="border-style:
dotted">
```

9 Type the contents of the article — for example:

```
<h2>Tactile Switches</h2>

<p>Tactile switches give a
perceptible bump but no click when
you press them down.</p>
```

10 Type the closing `</article>` tag:

```
</article>
```

11 Click **Refresh** ( ).

B The second article appears.

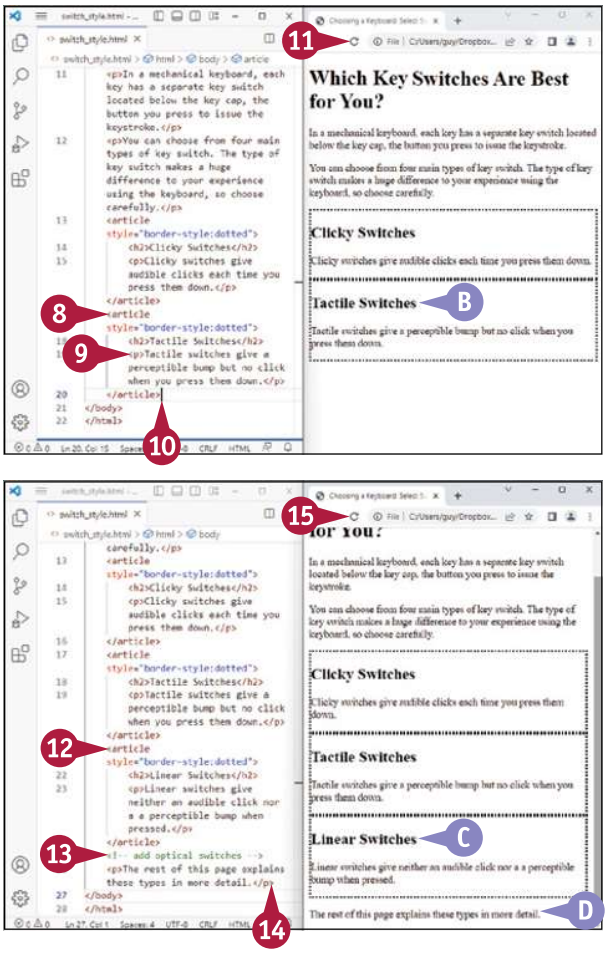
12 Repeat steps 8 to 10 to add a third article — for example:

```
<article
style="border-style:dotted">

<h2>Linear Switches</h2>

<p>Linear switches give neither an
audible click nor a perceptible
bump when pressed.</p>

</article>
```

13 Type a comment noting you need to add another article:

```
<!-- add optical switches -->
```

14 Type a body paragraph — for example:




```
<p>The rest of this page explains
these types in more detail.</p>
```

15 Click **Refresh** ( ).

C The third article appears.

D The body paragraph follows the third article.

TIP

How do I stop the border from touching the text in my `article` elements?

You can apply padding to the `article` element to put some space between the border and the element's contents. For this section, try changing `<article style="border-style:dotted">` to `<article style="border-style:dotted;padding:10px">`, which puts 10 pixels of padding top, bottom, left, and right. You can also apply different padding on the various sides. See the section "Specify Padding and Borders for an Element" in Chapter 9 for more details.

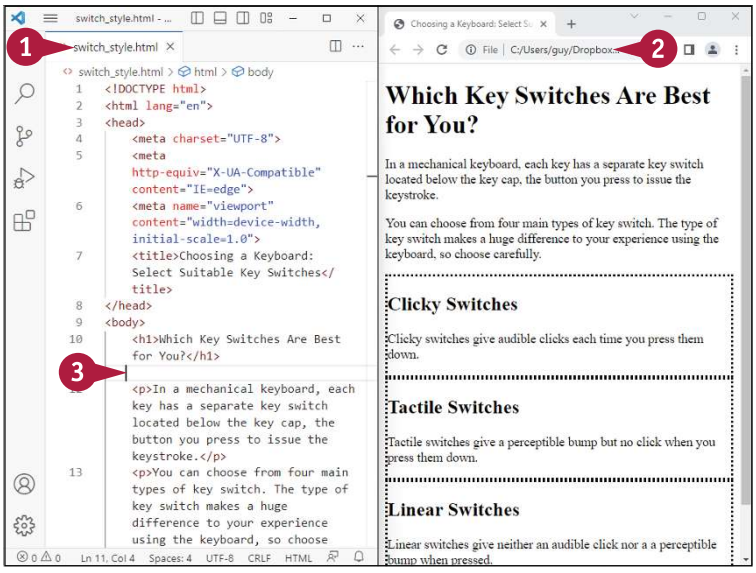
Create Pull Quotes with the `aside` Element

HTML's semantic elements include the `aside` element, which you use to separate some content from the content that surrounds it. An `aside` element can be a useful way to emphasize part of your web page or to draw the reader's attention to the element in which the aside is positioned.

The `aside` element has no specific positioning, but you can use the `style` attribute to position and format the `aside` element as needed to complement your web page.

Create Pull Quotes with the `aside` Element

1. In Visual Studio Code, either create and save a new file or open the existing file you want to use.
2. Open the file in a browser window so you can see the results of the changes you make.
3. In Visual Studio Code, click to place the insertion point where you want to position the `aside` element.



4. Type the opening `<aside>` tag:

```
<aside>
```

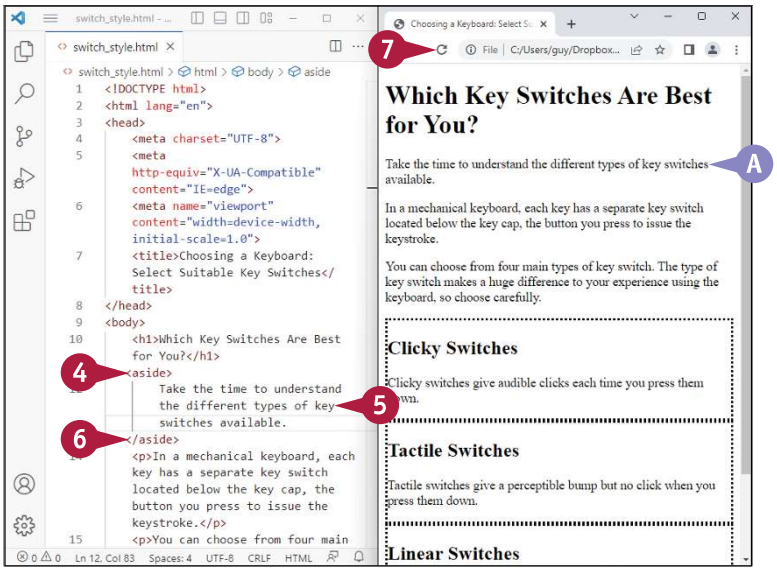
5. Type the contents you want to display in the `aside` element — for example:

```
Take the time to understand the different types of key switches available.
```

6. Type the closing `</aside>` tag:

```
</aside>
```

7. Click **Refresh** (↻).



A. The `aside` element appears in the page.

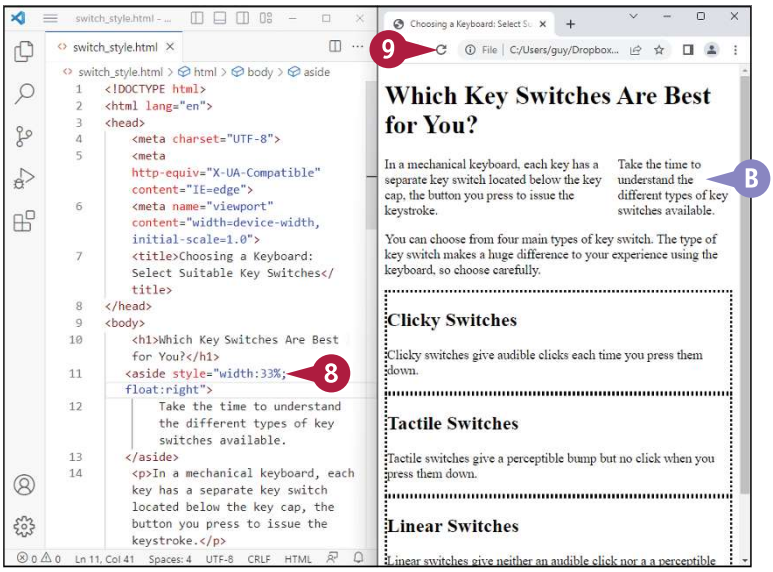
Because you have not specified any style formatting, the `aside` element appears like the other paragraphs.

8. Click before the closing > of the opening `<aside>` tag and type the `style` attribute, specifying `width:33%` and `float:right`, so the tag looks like this:

```
<aside style="width:33%;
float:right">
```

9. Click **Refresh** (↻).

B The `aside` element appears at one-third the page width and floating right.

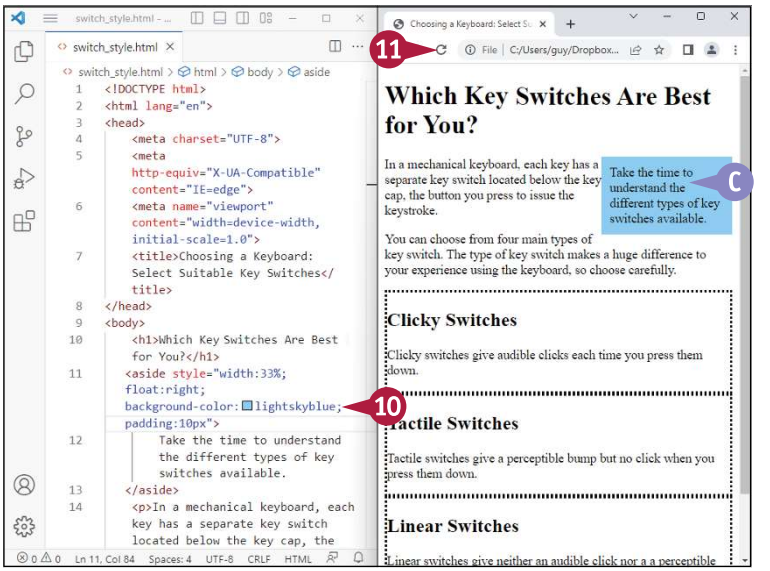


10. Click after `float:right` and before the double quotes and continue the `style` attribute formatting, adding `background-color:lightskyblue` and `padding:10px`. The complete tag looks like this:

```
<aside style="width:33%;
float:right;background-
color:lightskyblue;
padding:10px">
```

11. Click **Refresh** (↻).

C The `aside` element takes on a light blue background and 10 pixels of padding on each side.



TIP

What are other uses of the `aside` element?

Apart from creating pull quotes, as shown in this section, the `aside` element is widely used to create sidebars, to implement navigational elements, and for advertising.

Divide a Page Using `section` Elements

HTML's `section` element enables you to divide a web page into separate sections. Because `section` is a semantic element, the page's division into sections should be logical, but you can also use it for practical purposes. For example, you can apply formatting to all the child elements in a `section` element simultaneously by specifying the formatting for the `section` element.

Divide a Page Using `section` Elements

1. In Visual Studio Code, either create and save a new file or open the existing file you want to use.
2. Open the file in a browser window so you can see the results of the changes you make.
3. In Visual Studio Code, click to place the insertion point where you want to begin the first `section` element.
4. Type the opening `<section>` tag:

   ```
   <section>
   ```

5. Type the contents for the first section. The example includes an `h2` element and a `p` element:

   ```
   <h2>Introduction</h2>

   <p>The Anne Pro 2 from Obinslab is a 60%-size mechanical keyboard with built-in wireless connectivity.</p>
   ```

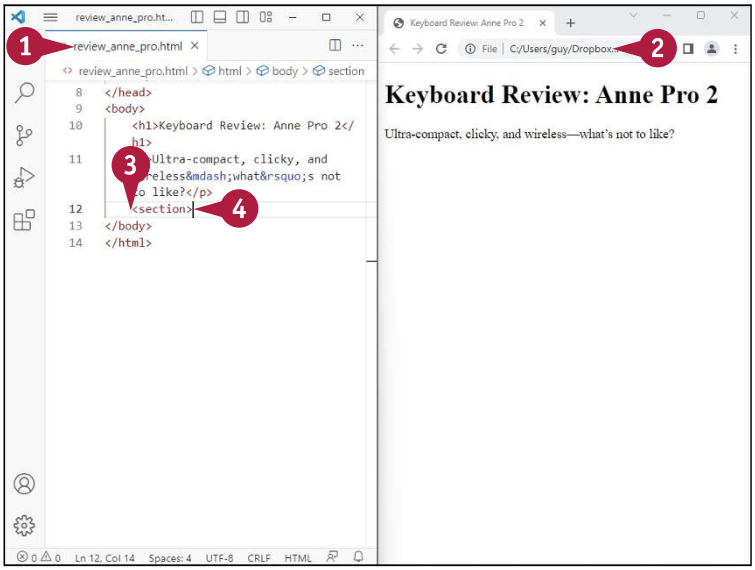
6. Type the closing `</section>` tag:

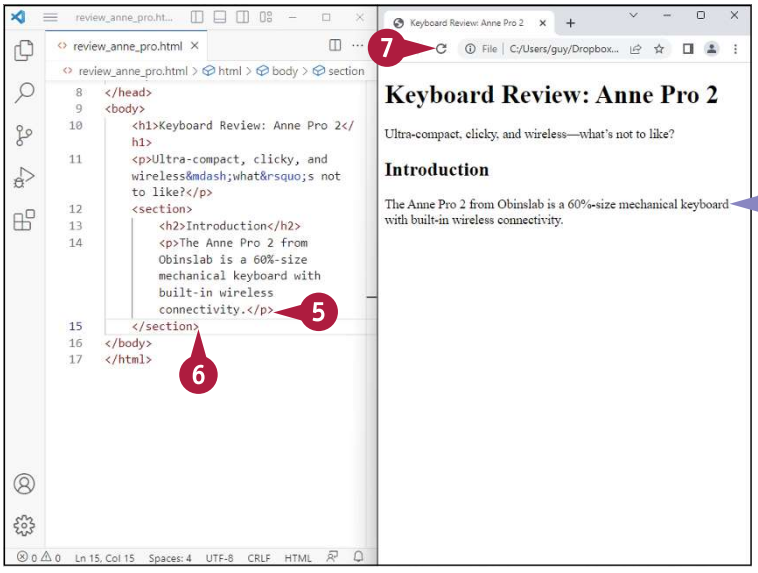
   ```
   </section>
   ```

7. Click **Refresh** (↻).

A. The section's contents appear in the web page.

There is no visible indication that the section exists.





8. Click in the Visual Studio Code window and type the opening `<section>` tag, this time including the `style` attribute and specifying `border-style:dashed`:

```
<section style="border-style:
dashed">
```

9. Type the contents for the second section. The example includes an `h2` element and a `p` element:

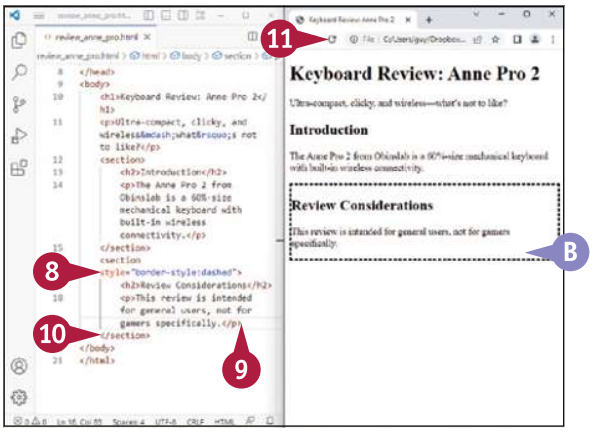
```
<h2>Review Considerations</h2>

<p>This review is intended for
general users, not for gamers
specifically.</p>
```

10. Type the closing `</section>` tag:

```
</section>
```

11. Click **Refresh** (↻).

B. The section's contents appear with a dashed outline.

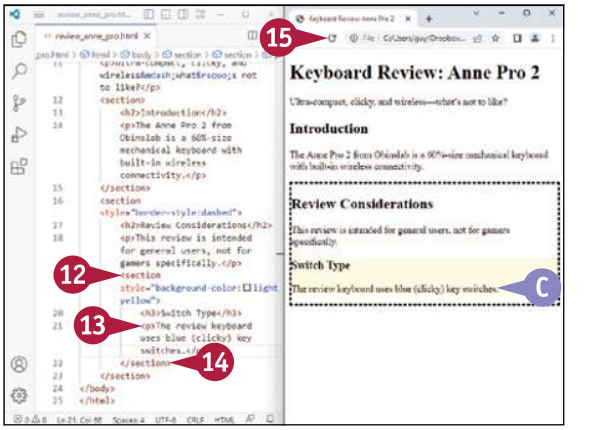


12. Click before the closing `</section>` tag and type the opening `<section>` tag for a subsection, including the `style` attribute and specifying `background-color:lightyellow`:

```
<section style="background-color:
lightyellow">
```

13. Enter an `h3` element and a `p` element. For example:

```
<h3>Switch Type</h3>

<p>The review keyboard uses blue
(clicky) key switches.</p>
```



14. Type the closing `</section>` tag for the subsection:

```
</section>
```

15. Click **Refresh** (↻).

C. The subsection appears in the page.

TIP

What is the difference between the `section` element and the `div` element?

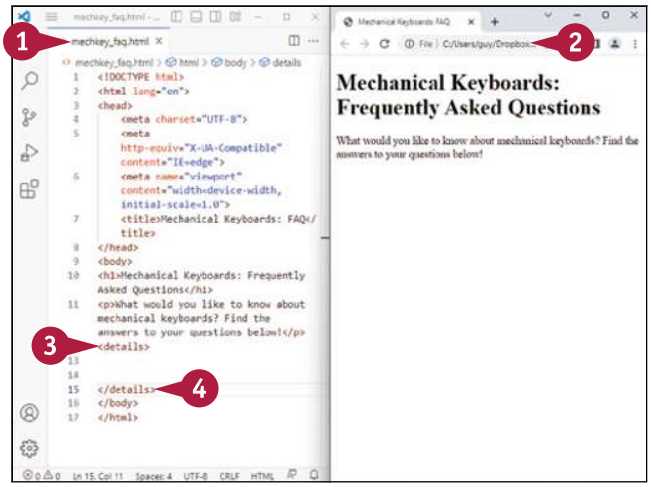
Both the `section` element and the `div` element enable you to group child elements and optionally apply formatting to them all at once; you will often see `section` and `div` used more or less interchangeably. However, `section` is a semantic element intended to suggest that all its contents relate to the same theme, whereas `div` is a nonsemantic element that carries no such implication.

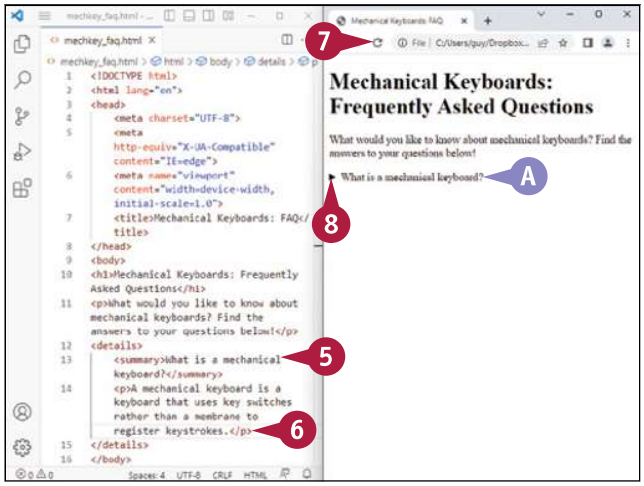
Best practice is to use semantic elements, such as the `section` element and the `article` element, to identify particular sections of a web page logically and to use the `div` element only when no semantic element is suitable.

Create Collapsible Sections

The `details` element enables you to create content sections that the user can expand and collapse as needed. For example, you might create a Frequently Asked Questions page for your website that appears at first as a list of questions whose answers are not visible. By clicking a question, the user can expand the content section to display its answer; after reading the answer, the user can click the question again to hide the answer once more. To create this effect, you use the `details` element for the expanding and collapsing and the `summary` element to display the text that is initially visible.

Create Collapsible Sections





1. In Visual Studio Code, either create and save a new file, or open the existing file you want to use.
2. Open the file in a browser window so you can see the results of the changes you make.
3. In Visual Studio Code, click to place the insertion point where you want to begin the first `details` element.
4. Type the opening `<details>` tag, press Enter twice, and then type the closing `</details>` tag:

```
<details>

</details>
```

5. Click between the `<details>` tag and the `</details>` tag and type the opening `<summary>` tag, its contents, and the closing `</summary>` tag — for example:

```
<summary>What is a mechanical keyboard?</summary>
```

6. Type the contents that the `details` element will display when clicked — for example:

```
<p>A mechanical keyboard is a keyboard that uses key switches rather than a membrane to register keystrokes.</p>
```

7. Click **Refresh** (↻).

A. The `details` element appears but is collapsed, so you see only the `summary` element.

8. Click **Expand** (▶ changes to ▼).

B The `details` element expands, revealing its contents.

C You can click **Collapse** (▼ changes to ▶) to collapse the `details` element again.

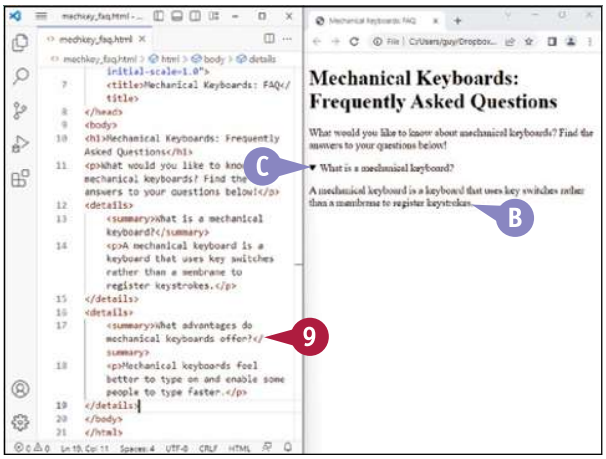
9 Click after the `details` element in Visual Studio Code and type another `details` element, including the `summary` element — for example:

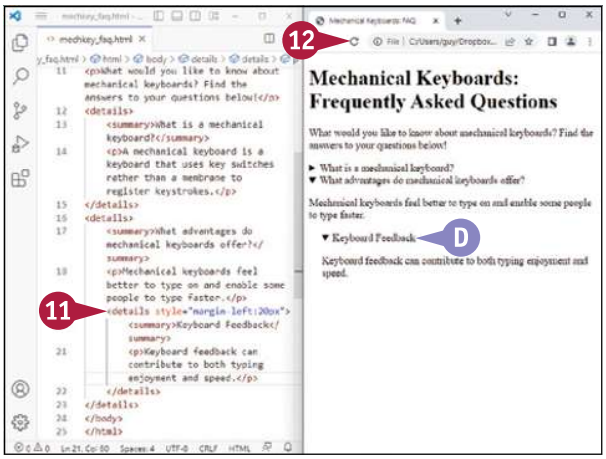
```
<details>
    <summary>What advantages
do mechanical keyboards offer?
</summary>
    <p>Mechanical keyboards feel
better to type on and enable some
people to type faster.</p>
</details>
```

10 Click before the closing `</details>` tag of the second `details` element (not shown).

11 Type a nested `details` element. Specify the `style` attribute with the value `margin-left:20px`, as in this example:

```
<details style="margin-left:20px">
    <summary>Keyboard Feedback
</summary>
        <p>Keyboard feedback can
contribute to both typing enjoyment
and speed.</p>
</details>
```





12 Click **Refresh** (↻).

The updated page appears.

D You can expand both the outer `details` element and the nested `details` element.

TIP

Must I include the `summary` element in each `details` element?

Normally, including the `summary` element is helpful, because it enables you to display suitable text to the right of the Expand arrow (▶) for the `details` element. However, you can omit the `summary` element without causing an error. If you do so, HTML displays the default text `Details` to the right of the Expand arrow (▶).