7

More on Links, Layout, and Mobile

Chapter Objectives

In this chapter, you will learn how to . . .

- Code relative hyperlinks to web pages in folders within a website
- Configure a hyperlink to a named fragment internal to a web page
- Provide for accessibility by configuring ARIA landmark roles for structural HTML elements
- Configure images with CSS sprites
- Configure a three-column page layout using CSS
- Configure CSS for printing
- Describe mobile design best practices
- Configure web pages for mobile display using the viewport meta tag
- Apply responsive web design techniques with CSS3 media queries
- Apply responsive image techniques including the new HTML5 picture element
- Apply the new CSS3 Flexbox layout model

Now that you've had some experience coding HTML and CSS, you're ready to explore a variety of techniques in this chapter, including relative hyperlinks and named fragment hyperlinks, CSS sprites, three-column page layout, styling for print, styling for mobile browsers, configuring CSS3 media queries to target mobile devices, configuring responsive images, and exploring the new CSS3 Flexbox layout model.

7.1 Another Look at Hyperlinks

Hyperlinks make the Web a "web" of interconnected information. In this section, you'll revisit the topic of hyperlinks and explore coding relative links, using the target attribute to open web pages in a new browser window, coding hyperlinks that are internal to a web page, supporting accessibility with landmark roles, configuring block anchors, and configuring telephone number hyperlinks for use on smartphones.

More on Relative Linking

As discussed in Chapter 2, a relative hyperlink is used to link to web pages within your site. You've been coding relative links to display web pages that are all inside the same folder. There are times when you need to link to files that are located in other folders on your website. Let's consider a website for a bed and breakfast that features rooms and events. The folder and file listing is shown in Figure 7.1. The main folder for this website is called casita, and the web developer has created separate subfolders—named images, rooms, and events—to organize the site.

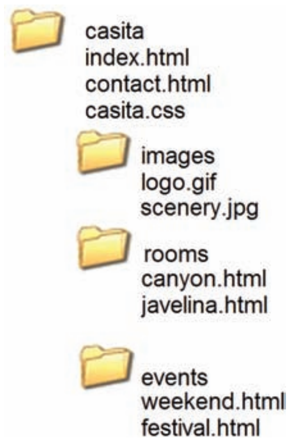


Figure 7.1 The web page files are organized in folders

Relative Link Examples

Recall that when linking to a file located in the same folder or directory, the value of the href attribute is the name of the file. For example, to link from the home page (index.html) to the contact.html page, code the anchor element as follows:

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

When linking to a file that is inside a folder within the current directory, use both the folder name and the file name in the relative link. For example, to link from the home page (index.html) to the canyon.html page (located in the rooms folder), code the anchor element as follows:

```
<a href="rooms/canyon.html">Canyon</a>
```

As shown in Figure 7.1, the canyon.html page is located in the rooms subfolder of the casita folder. The home page for the site (index.html) is located in the casita folder. When linking to a file that is up one directory level from the current page, use the "`../`" notation. To link to the home page for the site from the canyon.html page, code the anchor element as follows:

```
<a href="../index.html">Home</a>
```

When linking to a file that is in a folder on the same level as the current folder, the href value will use the "`../`" notation to indicate moving up one level; then specify the desired folder. For example, to link to the weekend.html page in the events folder from the canyon.html page in the rooms folder, code the anchor element as follows:

```
<a href="../events/weekend.html">Weekend Events</a>
```

Don't worry if the use of "`../`" notation and linking to files in different folders seems new and different. In most of the exercises in this book, you will code either absolute links to other websites or relative links to files in the same folder. You can explore the example of the bed and breakfast website located in the student files (see chapter7/CasitaExample) to become more familiar with coding references to files in different folders.

Hands-On Practice 7.1

This hands-on practice provides an opportunity to practice coding hyperlinks to files in different folders. The website you'll be working with has pages in prototype form: the navigation and layout of the pages are configured, but the specific content has not yet been added. You'll focus on the navigation area in this Hands-On Practice. Figure 7.2 shows a partial screen shot of the bed and breakfast's prototype home page with a navigation area on the left side of the page.

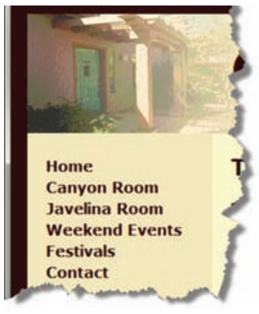


Figure 7.2 The navigation area

Examine Figure 7.3 and notice the new juniper.html file listed within the rooms folder. You will create a new web page (Juniper Room) named juniper.html and save it in the rooms folder. Then, you will update the navigation area on each existing web page to link to the new Juniper Room page.

Let's get started.

1. Copy the CasitaExample folder (chapter7/CasitaExample) from the student files. Rename the folder casita.
2. Display the index.html file in a browser and click through the navigation links. View the source code of the pages and notice how the href values of the anchor tags are configured to link to and from files within different folders.
3. Launch a text editor and open the canyon.html file. You'll use this file as a starting point for your new Juniper Room page. Save the file as juniper.html in the rooms folder.
   a. Edit the page title and h2 text. Change "Canyon" to "Juniper".
   b. Add a new li element in the navigation area that contains a hyperlink to the juniper.html file.

   ```
   <li><a href="juniper.html">Juniper Room</a></li>
   ```

   Place this hyperlink between the Javelina Room and Weekend Events navigation hyperlinks as shown in Figure 7.4. Save the file.

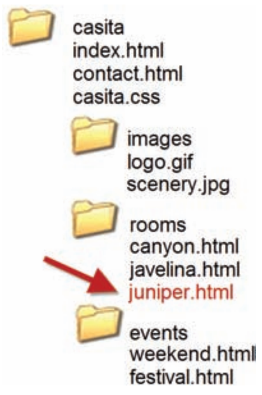


Figure 7.3 New juniper.html file is in the rooms folder

4. Use the coding for the Canyon and Javelina hyperlinks as a guide as you add the Juniper Room link to the navigation area on each of the following pages:

   index.html

   contact.html

   rooms/canyon.html

   rooms/javalina.html

   events/weekend.html

   events/festival.html

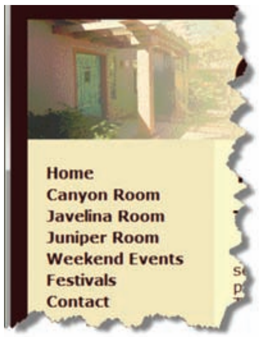


Figure 7.4 The new navigation area

Save all the .html files and test your pages in a browser. The navigation hyperlink to the new Juniper Room page should work from every other page. The hyperlinks on the new Juniper Room page should function well and open other pages as expected. A solution is in the student files (chapter7/7.1 folder).

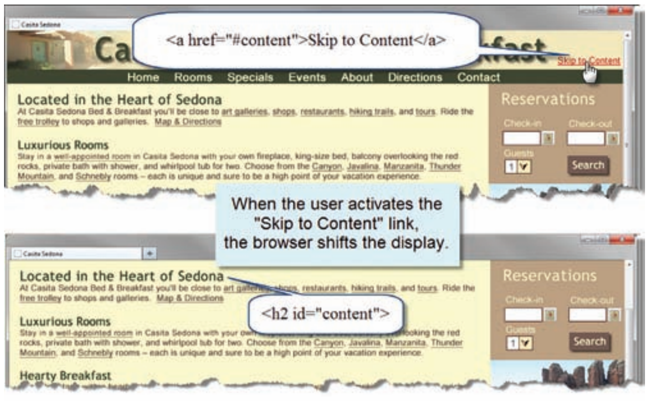
Fragment Identifiers

VideoNote
Linking to a Named Fragment

Browsers begin the display of a web page at the top of the document. However, there are times when you need to provide the capability to link to a specific portion of a web page instead of the top. You can accomplish this by coding a hyperlink to a **fragment identifier** (sometimes called a named fragment or fragment id), which is simply an HTML element assigned to an id attribute.

There are two components to your coding when using fragment identifiers:

1. The tag that identifies the **named fragment** of a web page: The tag must be assigned to an id. For example, `<div id="content">`
2. The anchor tag that links to the named fragment on a web page.



```
<a href="#content">Skip to Content</a>
```

When the user activates the "Skip to Content" link, the browser shifts the display.

```
<h2 id="content">
```

Figure 7.5 The "Skip to Content" link in action

Lists of frequently asked questions (FAQs) often use fragment identifiers to jump to a specific part of the page and display the answer to a question. Linking to a named fragment is often seen on long web pages. You might see a "Back to top" hyperlink that a visitor could select to cause the browser to quickly scroll the page back to the top of the page for easy site navigation. Another use of fragment identifiers helps to provide for accessibility. Web pages may have a fragment identifier to indicate the beginning of the actual page content. When the visitor clicks on the "Skip to content" hyperlink, the browser links to the fragment identifier and shifts focus to the content area of the page. This "Skip to content" or "Skip navigation" link provides a way for screen reader users to skip repetitive navigation links (see Figure 7.5).

Focus on Accessibility

This is accomplished in two steps:

1. **Establish the Target.** Create the "Skip to content" fragment identifier by configuring an element that begins the page content with an id, for example, `<div id="content">`
2. **Reference the Target.** At the point of the page where you want to place a hyperlink to the content, code an anchor element. Use the `href` attribute and place a `#` symbol (called a hash mark) before the name of the fragment identifier. The code for a hyperlink to the named fragment "content" is

```
<a href="#content">Skip to Content></a>
```

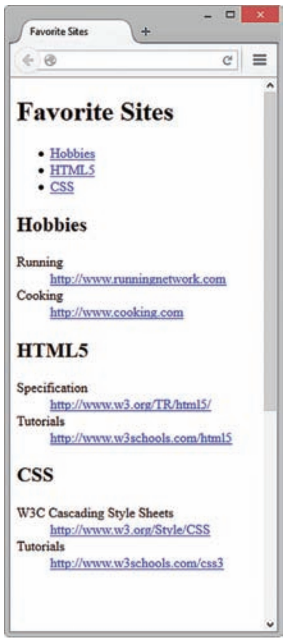
The hash mark indicates that the browser should search for an id on the same page. If you forget to type the hash mark, the browser will not look on the same web page; it will look for an external file.

Legacy Alert. Older web pages may use the name attribute and refer to named anchors rather than fragment identifiers. This coding technique is obsolete and not valid in HTML5. Named anchors use the name attribute to identify or name the fragment. For example,

```
<a name="content" id="content"></a>
```

Hands-On Practice 7.2

You will work with fragment identifiers in this Hands-On Practice. Locate chapter7/starter1.html in the student files. Save the file as favorites.html. Figure 7.6 shows a screenshot of this web page. Examine the source code and notice that the top portion of the page contains an unordered list with categories of interest (such as Hobbies, HTML5, and CSS) that correspond to the text displayed in the h2 elements below. Each h2 element is followed by a description list of topics and URLs related to that category. It might be helpful to web page visitors if they can click a category item and immediately jump to the page area that has information related to that item. This could be a useful application of linking to fragment identifiers!



Favorite Sites

- Hobbies
- HTML5
- CSS

Hobbies

Running
    http://www.runningnetwork.com
Cooking
    http://www.cooking.com

HTML5

Specification
    http://www.w3.org/TR/html5/
Tutorials
    http://www.w3schools.com/html5

CSS

W3C Cascading Style Sheets
    http://www.w3.org/Style/CSS
Tutorials
    http://www.w3schools.com/css3

Figure 7.6 You will add hyperlinks to fragment identifiers

Modify the page as follows:

1. Code a named fragment for each h2 element in the definition list. For example,

```
<h2 id="hobbies">Hobbies</h2>
```

2. Add hyperlinks to the items in the unordered list so that each entry will link to its corresponding heading.
3. Add a named fragment near the top of the page.
4. Near the bottom of the favorites.html page, add a hyperlink to return to the top of the page.

Save the file and test it in a browser. Compare your work with the sample found in the student files (chapter7/7.2/favorites.html).

There may be times when you need to link to a named fragment on another web page. To accomplish this, place a "#" followed by the fragment identifier id value after the file name in the anchor tag. So, to link to the "Hobbies" (given that it is a named fragment called "hobbies") from any other page on the same website, you could use the following HTML:

```
<a href="favorites.html#hobbies">Hobbies</a>
```

FAQ Why don't some of my hyperlinks to fragment identifiers work?

The web browser fills the browser (viewport) with the web page and will scroll to display the named fragment at the top of the viewport. However, if there is not enough "page" left below the named fragment, the content where the named fragment is located will not be displayed at the top of the browser viewport. The browser tries to do the best it can while still filling the viewport with the web page content. Try adding some blank lines (use the `<br>` tag) or padding to the lower portion of the web page. Save your work and test your hyperlinks.

Landmark Roles with ARIA

Focus on Accessibility

You've seen how easy it is to code hyperlinks to sections of web pages using fragment identifiers. A skip to content hyperlink can be helpful to a person who is using assistive technology such as a screen reader to listen to a web page. The W3C's Web Accessibility Initiative (WAI) has developed a standard to provide for additional accessibility, called **Accessible Rich Internet Applications (ARIA)**. ARIA provides methods intended to increase the accessibility of web pages and web applications (http://www.w3.org/WAI/intro/aria). We'll focus on ARIA landmark roles in this section. A **landmark** on a web page is a major section such as a banner, navigation, main content, and so on. ARIA landmark roles allow web developers to configure semantic descriptions of HTML elements using the **`role` attribute** to indicate landmarks on the web page. For example, to indicate the landmark role of `main` on an element containing the main content of a web page document, code `role="main"` on the opening tag.

People visiting a web page with a screen reader or other assistive technology can access the landmark roles to quickly skip to specific areas on a web page (watch the video at http://www.youtube.com/watch?v=IhWMou12_Vk for a demonstration). Visit http://www.w3.org/TR/wai-aria/roles#landmark_roles for a complete list of ARIA landmark roles. Commonly used ARIA landmark roles include:

- `banner` (a header/logo area)
- `navigation` (a collection of navigation elements)
- `main` (the main content of a document)
- `complementary` (a supporting part of the web page document, designed to be complementary to the main content)
- `contentinfo` (an area that contains information about the content such as copyright)
- `form` (an area that contains a form)
- `search` (an area of a web page that provides search functionality)

Access chapter7/roles/index.html in the student files for a sample web page with the `banner`, `navigation`, `main`, and `contentinfo` roles configured. Notice that while the role attribute does not change the way the web page displays, it offers additional information about the document that can be used by assistive technologies.

The Target Attribute

You may have noticed as you have coded hyperlinks that when a visitor clicks on a hyperlink, the new web page will automatically open in the same browser window. You can configure the **target attribute** on an anchor tag with `target="_blank"` to open a hyperlink in a new browser window or browser tab. For example,

`<a href="http://yahoo.com" target="_blank">Yahoo!</a>` will open Yahoo! home page in a new browser window or tab.

Note that you cannot control whether the web page opens in a new window or opens in a new tab; this is dependent upon your visitor's browser configuration. Why not create a test page and try it? The target attribute with the value `"_blank"` configures the web page to open in a new browser window or tab.

Hands-On Practice 7.3

You will work with the target attribute in this Hands-On Practice. Locate chapter7/starter1.html in the student files. Save the file as target.html. Launch a text editor and open the file. Let's practice using the target attribute. Choose one of the external hyperlinks to modify. Configure `target="_blank"` so that the hyperlink opens in a new browser window or tab. An example is shown below:

```
<a href="http://www.cooking.com"
target="_blank">http://www.cooking.com"</a>
```

Save the file. Launch a browser and test the file. When you click on the hyperlink that you modified, the new page will display in a new browser window or tab. You can compare your work to the solution in the student files (chapter7/7.3/target.html).

Block Anchor

It's typical to use anchor tags to configure phrases or even just a single word as a hyperlink. HTML5 provides a new function for the anchor tag—the block anchor. A block anchor can configure one or more elements (even those that display as a block, such as a div, h1, or paragraph) as a hyperlink. See an example in the student files (chapter7/block.html).

Telephone and Text Message Hyperlinks

If a web page displays a phone number, wouldn't it be handy for a person using a smartphone to be able to tap on the phone number and place a call or send an SMS (Short Message Service) text message? It's very easy to configure a telephone hyperlink or SMS hyperlink for use by smartphones.

According to RFC 3966, you can configure a telephone hyperlink by using a telephone scheme: Begin the href value with `tel:` followed by the phone number. For example, to configure a telephone hyperlink on a web page for use by mobile browsers, code as follows: `<a href="tel:888-555-5555">Call 888-555-5555</a>`

RFC 5724 indicates that an SMS scheme hyperlink intended to send a text message can be configured by beginning the href value with `sms:` followed by the phone number, as shown in the following code:

```
<a href="sms:888-555-5555">Text 888-555-5555</a>
```

Not all mobile browsers and devices support telephone and text hyperlinks, but expect increased use of this technology in the future. You'll get a chance to practice using the `tel:` scheme in the Chapter 7 case study.

7.2 CSS Sprites

When browsers display web pages, they must make a separate http request for every file used by the page, including .css files and image files such as .gif, .jpg, and .png files. Each http request takes time and resources. As mentioned in Chapter 4, a **sprite** is an image file that contains multiple small graphics. Using CSS to configure the small graphics combined in the sprite as background images for various web page elements is called **CSS sprites**, a technique made popular by David Shea (http://www.alistapart.com/articles/sprites).

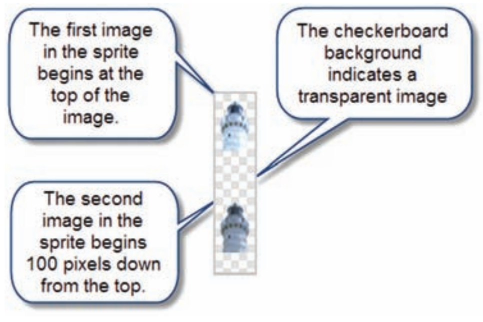
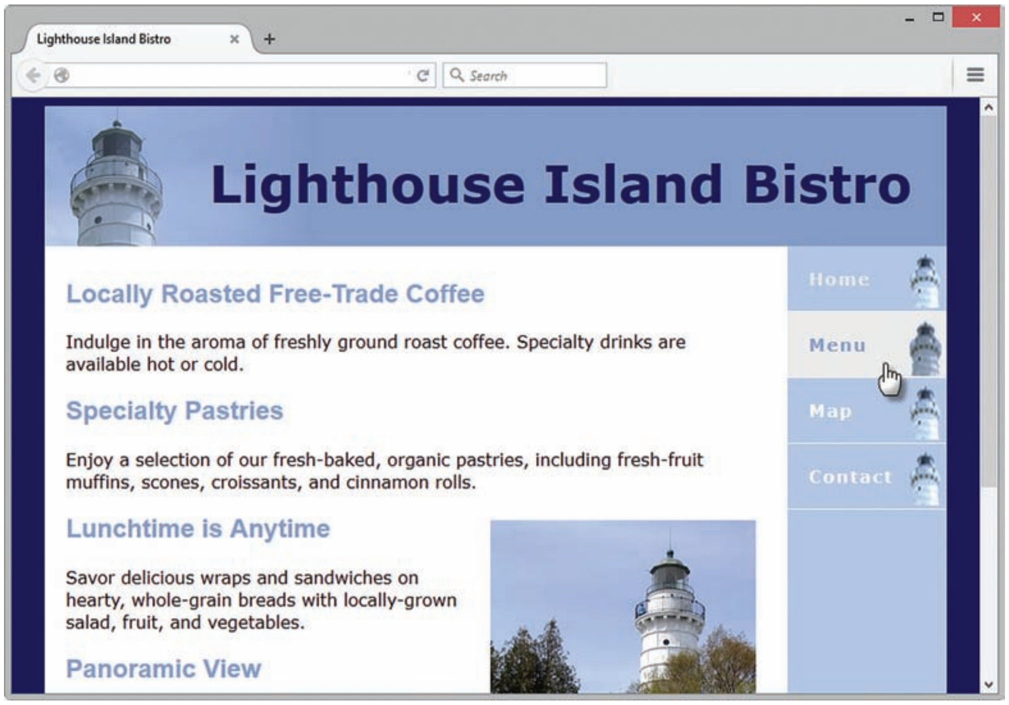

The first image in the sprite begins at the top of the image.
The checkerboard background indicates a transparent image
The second image in the sprite begins 100 pixels down from the top.

Figure 7.7 The sprite consists of two images in a single graphics file

The CSS sprites technique uses the CSS `background-image`, `background-repeat`, and `background-position` properties to manipulate the placement of the background image. The single graphics file saves download time because the browser only needs to make one http request for the combined image instead of many requests for the individual smaller images. Figure 7.7 shows a sprite with two lighthouse images on a transparent background. These images are configured as background images for the navigation hyperlinks with CSS as shown in Figure 7.8. You'll see this in action as you complete the next Hands-On Practice.

Figure 7.8 Sprites in action



Lighthouse Island Bistro

Locally Roasted Free-Trade Coffee

Indulge in the aroma of freshly ground roast coffee. Specialty drinks are available hot or cold.

Specialty Pastries

Enjoy a selection of our fresh-baked, organic pastries, including fresh-fruit muffins, scones, croissants, and cinnamon rolls.

Lunchtime is Anytime

Savor delicious wraps and sandwiches on hearty, whole-grain breads with locally-grown salad, fruit, and vegetables.

Panoramic View

Hands-On Practice 7.4

You will work with CSS sprites in this Hands-On Practice as you create the web page shown in Figure 7.8. Create a new folder named sprites. Locate chapter7/starter2.html in the student files. Copy starter2.html into your sprites folder. Copy the following files from chapter7/starters into your sprites folder: lighthouseisland.jpg, lighthouselogo.jpg, and sprites.gif. The sprites.gif, shown in Figure 7.7, contains two lighthouse images. The first lighthouse image starts at the top of the graphics file. The second lighthouse image begins 100 pixels down from the top of the graphics file. We'll use the value 100 when we configure the display of the second image.

Launch a text editor and open starter2.html. Save the file as index.html. You will edit the embedded styles to configure background images for the navigation hyperlinks.

1. Configure the background image for navigation hyperlinks. Add the following styles to the `nav a` selector to set the background image to the sprites.gif with no repeat. The value `right` in the `background-position` property configures the lighthouse image to display at the right of the navigation element. The value 0 in the `background-position` property configures the display at offset 0 from the top (at the very top) so the first lighthouse image displays.

```
nav a { text-decoration: none;
        display: block;
        padding: 20px;
        background-color: #b3c7e6;
        border-bottom: 1px solid #ffffff;
        background-image: url(sprites.gif);
        background-repeat: no-repeat;
        background-position: right 0; }
```

2. Configure the second lighthouse image to display when the mouse pointer passes over the hyperlink. Add the following styles to the `nav a:hover` selector to display the second lighthouse image. The value `right` in the `background-position` property configures the lighthouse image to display at the right of the navigation element. The value `-100px` in the `background-position` property configures the display at an offset of 100 pixels down from the top so the second lighthouse image appears.

```
nav a:hover { background-color: #eaeaea;
              color: #869dc7;
              background-position: right -100px;   }
```

Save the file and test it in a browser. Your page should look similar to Figure 7.8. Move your mouse pointer over the navigation hyperlinks to see the background images change. Compare your work with the sample found in the student files (chapter7/7.4/index.html).

FAQ How can I create my own sprite graphics file?

Most web developers use a graphics application such as Adobe Photoshop, Adobe Fireworks, or GIMP to edit images and save them in a single graphics file for use as a sprite. Or, you could use a web-based sprite generator such as the ones listed below:

- CSS Sprites Generator: http://csssprites.com
- CSS Sprite Generator: http://spritegen.website-performance.org
- SpriteMe: http://spriteme.org

If you already have a sprite graphic, check out the online tool at Sprite Cow (http://www.spritecow.com) that can generate pixel-perfect `background-position` property values for a sprite.

Checkpoint 7.1

1. Why should you organize the files in a website using folders and subfolders?
2. Which attribute configures a hyperlink to open the file in a new browser window or tab?
3. State an advantage of using CSS sprites in a website.

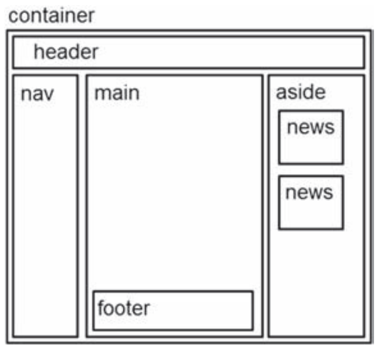
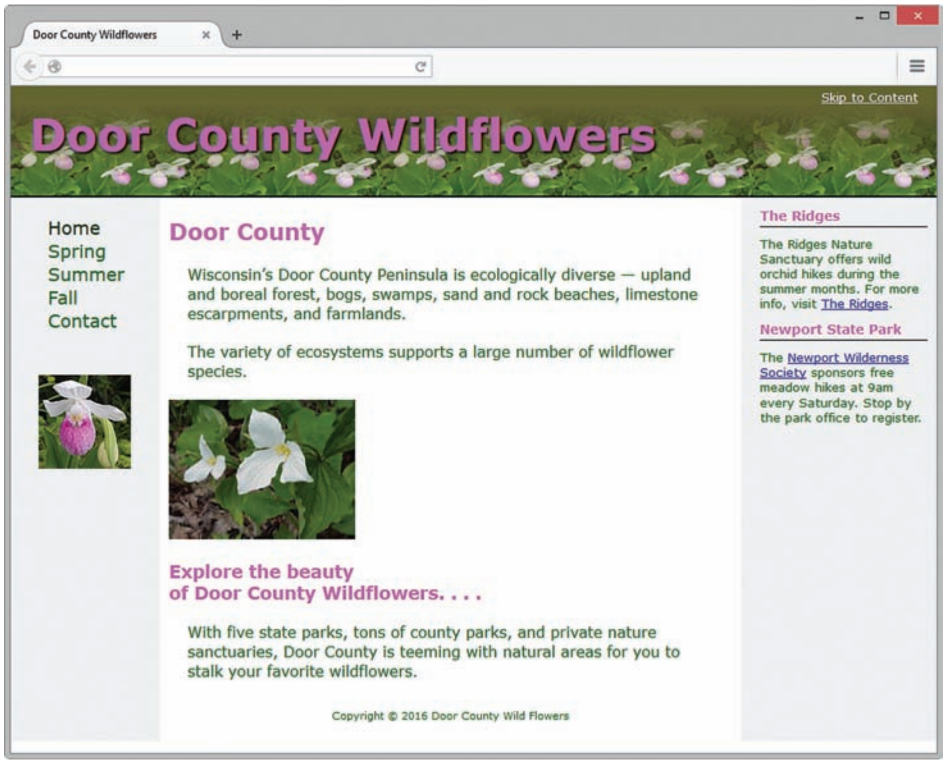


Figure 7.9 Wireframe for a three-column page layout

7.3 Three-Column CSS Page Layout

Often a web page layout will consist of a header across the top of the page with three columns below: navigation, main content, and sidebar. If you are thinking about this layout as a series of boxes—you're thinking correctly for configuring pages using CSS! Figure 7.9 shows a wireframe of this page layout design. Figure 7.10 shows a web page configured using this design. You will create this page in the next Hands-On Practice.



Skip to Content

Door County Wildflowers

Home
Spring
Summer
Fall
Contact

Door County

Wisconsin's Door County Peninsula is ecologically diverse — upland and boreal forest, bogs, swamps, sand and rock beaches, limestone escarpments, and farmlands.

The variety of ecosystems supports a large number of wildflower species.

Explore the beauty
of Door County Wildflowers. . . .

With five state parks, tons of county parks, and private nature sanctuaries, Door County is teeming with natural areas for you to stalk your favorite wildflowers.

Copyright © 2016 Door County Wild Flowers

The Ridges

The Ridges Nature Sanctuary offers wild orchid hikes during the summer months. For more info, visit The Ridges.

Newport State Park

The Newport Wilderness Society sponsors free meadow hikes at 9am every Saturday. Stop by the park office to register.

Figure 7.10 This three-column page layout is designed using CSS

Hands-On Practice 7.5

In this Hands-On Practice, you will develop your first three-column web page using CSS. The same techniques that you used to configure the two-column page will apply here. Think of the page as a series of elements or boxes. Using a wireframe as a guide, configure the basic page structure with HTML. Then code CSS to configure page areas; use ids or classes when appropriate. Recall that a key technique for creating a two-column web page with left-column navigation is to design the left column to float to the left. A key technique for our three-column page is to code the left column with `float:left` and the right column with `float:right`. The center column occupies the middle of the browser window. Refer to Figures 7.9 and 7.10 as you complete this Hands-On Practice.

Getting Started

Locate the showybg.jpg, plsthumb.jpg, and trillium.jpg files in the chapter7/starters folder in the student files. Create a new folder called wildflowers3. Copy the files to the folder.

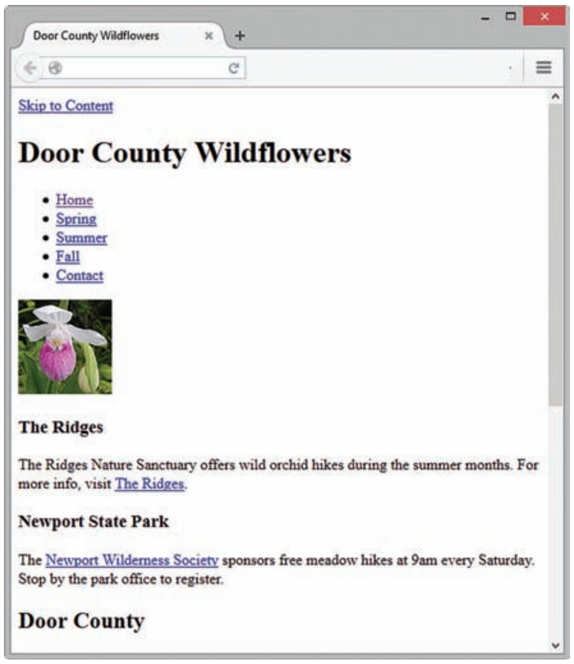
Part 1: Code the HTML

Review Figures 7.9 and 7.10. Notice the page elements: a header area with a background image; a left column with a navigation area and an image; a center column with paragraphs of text, headings, and an image that floats to the right; a right column with two news items; and a footer. You will code CSS to configure the layout and other styles. The navigation menu hyperlinks will be configured using an unordered list. As you code the HTML document, you will place the elements on the page and assign id and class values that correspond to the wireframe areas in Figure 7.9. ARIA landmark roles will also be assigned. Launch a text editor and type in the following HTML:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Door County Wildflowers</title>
<meta charset="utf-8">
</head>
<body>
<div id="container">
  <header role="banner">
     <span><a href="#content">Skip to Content</a></span>
     <h1>Door County Wildflowers</h1>
  </header>
  <nav role="navigation">
     <ul>
     <li><a href="index.html">Home</a></li>
     <li><a href="spring.html">Spring</a></li>
     <li><a href="summer.html">Summer</a></li>
     <li><a href="fall.html">Fall</a></li>
     <li><a href="contact.html">Contact</a></li>
   </ul>
   <img src="plsthumb.jpg" width="100" height="100" alt="Showy Lady
Slipper">
</nav>
<aside role="complementary">
     <h3>The Ridges</h3>
     <p class="news">The Ridges Nature Sanctuary offers wild orchid
hikes during the summer months. For more info, visit
<a href="http://ridgessanctuary.org">The Ridges</a>.</p>
   <h3>Newport State Park</h3>
   <p class="news">The <a href="http://newportwildernesssociety.org">
Newport Wilderness Society</a> sponsors free meadow hikes at 9am every
Saturday. Stop by the park office to register.</p>
</aside>
<main role="main" id="content">
    <h2>Door County</h2>
    <p>Wisconsin &#39;s Door County Peninsula is ecologically diverse
&mdash; upland and boreal forest, bogs, swamps, sand and rock beaches,
limestone escarpments, and farmlands.</p>
   <p>The variety of ecosystems supports a large number of wildflower
species.</p>
```

```
   <img src="trillium.jpg" width="200" height="150" alt="Trillium"
class="floatright">
     <h3>Explore the beauty <br>of Door County Wildflowers....</h3>
   <p>With five state parks, tons of county parks, and private nature
sanctuaries, Door County is teeming with natural areas for you to
stalk your favorite wildflowers.</p>
    <footer role="contentinfo"> Copyright &copy; 2016 Door County Wild
Flowers<br>
   </footer>
  </main>
</div>
</body>
</html>
```

Also configure the HTML5 shim (see Chapter 6) in the head section of the web page to provide for support of older versions of Internet Explorer. Save your page as index.html in your wildflowers3 folder. Test the page in a browser. Your display will not look like Figure 7.10 because you have not yet configured the CSS. The top of your page should look similar to the page shown in Figure 7.11.


Door County Wildflowers

Skip to Content

Door County Wildflowers

- Home
- Spring
- Summer
- Fall
- Contact

The Ridges

The Ridges Nature Sanctuary offers wild orchid hikes during the summer months. For more info, visit The Ridges.

Newport State Park

The Newport Wilderness Society sponsors free meadow hikes at 9am every Saturday. Stop by the park office to register.

Door County

Figure 7.11 The three-column page before CSS is applied

Part 2: Code the Basic CSS

For ease of editing, in this Hands-On Practice, you will code the CSS as embedded styles in the header section of the web page. However, if you were creating an entire website, you would most likely use an external style sheet. Launch a text editor and open index.html. Let's take a moment to consider the areas on the page shown in Figure 7.10: header, left-column navigation area, right-column sidebar area, center column, and footer. The left column will contain a navigation area and a small image. The center column will contain paragraphs, a heading, and a right-floating image. The right column will contain a series of headings and news items. Locate these areas on the sketch in Figure 7.9. Notice also that the same font is used throughout the page and the page begins immediately at the browser margin.

With your file open in a text editor, modify the head section of your document and code a `<style>` tag. Now let's consider the CSS configuration. Type the CSS in your document as indicated below:

1. Support HTML5 elements in older browsers:

   ```
   header, nav, main, footer, aside { display: block; }
   ```

2. Configure box-sizing for all HTML elements:

   ```
   * { box-sizing: border-box;}
   ```

3. **Body Element Selector.** Set the margin to 0. Configure the background color to `#ffffff`.

   ```
   body { margin:0;
          background-color: #ffffff; }
   ```

4. **Container.** Configure this area with background (`#eeeeee`) and text (#006600) colors, a minimum width of 960 pixels, and a font family of Verdana, Arial, or sans-serif.

   ```
   #container { background-color: #eeeeee;
                color: #006600;
                min-width: 960px;
                font-family: Verdana, Arial, sans-serif; }
   ```

5. **Header.** Configure a background color (#636631) and a background image (position showybg.jpg at the bottom of the element to repeat horizontally). Set the height to 120 pixels, text color to `#cc66cc`, text alignment to right, no top padding, no bottom padding, 20 pixels left padding, and 20 pixels right padding. Configure a 2 pixel solid black border across the bottom of this area.

   ```
   header { background-color: #636631;
            background-image: url(showybg.jpg);
            background-position: bottom;
            background-repeat: repeat-x;
            height: 120px;
            color: #cc66cc;
            text-align: right;
            padding: 0 20px;
            border-bottom: 2px solid #000000; }
   ```

6. **Left Column.** One of the keys to this three-column page layout is that the left navigation column is designed to float to the left of the browser window. Configure a width of 150 pixels.

   ```
   nav { float:  left;
         width: 150px; }
   ```

7. **Right Column.** One of the keys to this three-column page layout is that the right sidebar column is designed to float to the right of the browser window. Configure a width of 200 pixels.

   ```
   aside { float:  right;
           width: 200px; }
   ```

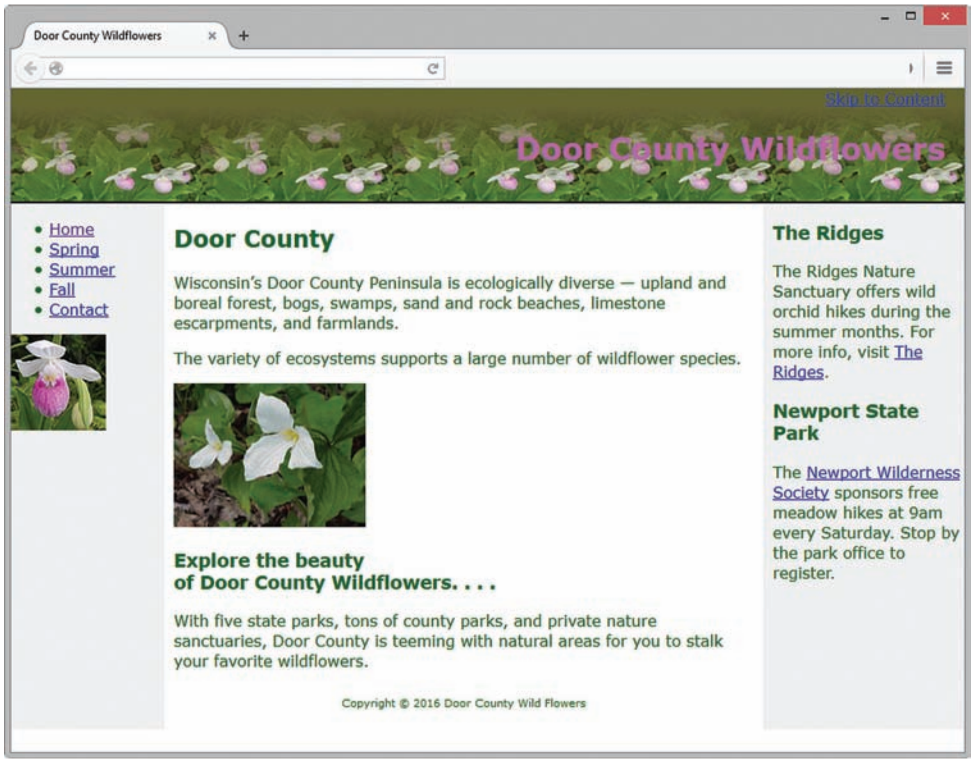
8. **Center.** The main content will take up all of the room that is available after the left and right columns float. The content area has a special need for margins because the left and right columns are floating on either side. Set the left margin to 160 pixels, the right margin to 210 pixels, and the remaining margins to 0. Also configure the padding for this area. Set the background (#ffffff) and text (#006600) colors.

```
main { margin: 0 210px 0 160px;
       padding: 1px 10px 20px 10px;
       background-color: #ffffff;
       color: #006600; }
```

9. **Footer.** Configure the page footer with very small text that is centered. Configure the background (`#ffffff`) and text (`#006600`) colors for this area. Set the top padding to 10 pixels. Clear the floated image in the center content area.

```
footer { font-size: .70em;
         text-align: center;
         color: #006600;
         background-color: #ffffff;
         padding-top: 10px;
         clear: both; }
```

At this point, you have configured the main elements of the three-column page layout. Code the closing HTML style tag with `</style>`. Save index.html in the wildflowers3 folder. It's a good idea to open your page in a browser to make sure you are on the right track. It should look similar to the one shown in Figure 7.12. Note that there is still some detail work to do, but you are well on your way!



- Home
- Spring
- Summer
- Fall
- Contact

Door County

Wisconsin's Door County Peninsula is ecologically diverse — upland and boreal forest, bogs, swamps, sand and rock beaches, limestone escarpments, and farmlands.

The variety of ecosystems supports a large number of wildflower species.

Explore the beauty
of Door County Wildflowers. . . .

With five state parks, tons of county parks, and private nature sanctuaries, Door County is teeming with natural areas for you to stalk your favorite wildflowers.

Copyright © 2016 Door County Wild Flowers

The Ridges

The Ridges Nature Sanctuary offers wild orchid hikes during the summer months. For more info, visit The Ridges.

Newport State Park

The Newport Wilderness Society sponsors free meadow hikes at 9am every Saturday. Stop by the park office to register.

Figure 7.12 The CSS for the basic elements of the three-column layout has been completed

Part 3: Continue Coding CSS

Now you are ready to continue with your styles. Open index.html in a text editor and position your cursor on a blank line above the closing style tag.

1. **Header Area.**

   a. **The h1 Element Selector.** Notice the extra space above the "Door County Wildflowers" text, which is contained within the `<h1>` tag in the header element. You can reduce this extra space by setting a 0 top margin for the h1 element selector. Also configure left alignment, text shadow, and a font size of 3em for the h1 selector.

   ```
   h1 { margin-top: 0;
       font-size: 3em;
       text-align: left;
       text-shadow: 2px 2px 2px #000000; }
   ```

   b. **Skip to Content.** Configure the "Skip to Content" hyperlink in the header area with a 0.80em font size. Also configure pseudo-classes for `:link`, `:visited`, `:hover`, `:active`, and `:focus` with text color as shown below.

   ```
   header a {font-size: 0.80em; }
   header a:link, header a:visited { color: #ffffff; }
   header a:focus, header a:hover { color: #eeeeee; }
   ```

2. **Left Navigation Column.**

   a. **Navigation Menu.** Configure the unordered list in the `nav` element selector to provide for a 20 pixel top margin and not to display any list markers.

   ```
   nav ul { margin-top: 20px;
            list-style-type: none; }
   ```

   The navigation links should have no underline (`text-decoration: none`) and a 1.2em font size. Configure pseudo-classes for `:link`, `:visited`, `:hover`, `:active`, and `:focus` with text color as shown below.

   ```
   nav a { text-decoration: none;
           font-size: 1.2em; }
   nav a:link { color:#006600;}
   nav a:visited { color: #003300; }
   nav a:focus, nav a:hover { color: #cc66cc; }
   nav a:active { color: #000000;}
   ```

   b. **Left Column Image.** Configure images in the `nav` element with a margin of 30 pixels.

   ```
   nav img { margin: 30px;}
   ```

3. **Main Content.**

   a. **Paragraphs.** Configure the main content's paragraph element selector to display with a margin of 20 pixels.

   ```
   main p { margin: 20px; }
   ```

b. **Headings.** Configure the h2 and h3 element selectors in the main content with the same text color as the logo header text and the same background color as the main body of the page.

```
main h2, main h3 { color: #cc66cc;
                   background-color: #ffffff; }
```

c. **Image Floating at the Right.** Create a `floatright` class to use a 10 pixel margin and float to the right.

```
.floatright { margin: 10px;
              float: right; }
```

4. **Right Sidebar Column. This column is contained within the aside element.**

a. **Headings.** Configure the h3 element selector in this area with a 1 pixel black solid bottom border, 2 pixels of padding at the bottom, a 10 pixel margin, 0.90em font size, and the same text color as the logo header text.

```
aside h3 { padding-bottom: 2px;
           border-bottom: 1px solid #000000;
           margin: 10px;
           font-size: 0.90em;
           color: #cc66cc; }
```

b. **News Items.** Configure a class called `news` that uses a small font and has a 10 pixel margin.

```
.news { font-size: 0.80em;
        margin: 10px; }
```

Save index.html in the wildflowers3 folder.

Part 4: Test the Page

Now that your styles are coded, test the index.html page again. Your page should look similar to the screenshot shown in Figure 7.10. Recall that Internet Explorer version 9 and below does not support the `text-shadow` property. If there are other differences, verify the id and class values in your HTML. Also check the syntax of your CSS. You may find the W3C CSS Validation Service at http://jigsaw.w3.org/css-validator to be helpful when verifying CSS syntax. The student files contain a copy of this page in the chapter7/7.5 folder.

7.4 CSS Styling for Print

Even though the "paperless society" has been talked about for decades, the fact is that many people still love paper and you can expect your web pages to be printed. CSS offers you some control over what gets printed and how the printouts are configured. This is easy to do using external style sheets. Create one external style sheet with the configurations for browser display and a second external style sheet with the special printing configurations.

Associate both of the external style sheets to the web page using two link elements. The link elements will utilize the **`media` attribute**, which indicates the **media type** for which the styles are intended, such as screen display or print display. See a list of media attribute values in Table 7.1.

Table 7.1 The link element's media attribute

Value	Purpose
`screen`	Default value; Indicates the style sheet that configures the typical browser viewport display on a color computer screen
`print`	Indicates the style sheet that configures the printed formatting
`handheld`	Although this value is intended by the W3C to indicate the style sheet that configures display on handheld mobile devices, in practice, the attribute value is not reliably supported. The next section will describe other methods for configuring the design of mobile web pages.

Modern browsers will use the correct screen or print style sheet, depending on whether they are rendering a screen display or preparing to print a document. Configure the link element for your browser display with `media="screen"`. Configure the link element for your printout with `media="print"`. An example of the HTML follows:

```
<link rel="stylesheet" href="wildflower.css" media="screen">
<link rel="stylesheet" href="wildflowerprint.css" media="print">
```

Print Styling Best Practices

You might be wondering how a print style sheet should differ from the CSS used to display the web page in a browser. Commonly used techniques for styling print are listed below.

Hide Non-Essential Content

It's common practice to prevent banner ads, navigation, or other extraneous areas from appearing on the printout. Use the `display: none;` style declaration to hide content that is not needed on a printout of the web page.

Configure Font Size and Color for Printing

Another common practice is to configure the font sizes on the print style sheet to use pt units. This will better control the text on the printout. You might also consider configuring the text color to black (#`000000`) and background color to white (#FFFFFF) if you envision the need for visitors to print your pages often. The default setting on most browsers prevent background colors and background images from printing, but you can prevent background image display in your print style sheet by setting the `background-image` property to the value none.

Control Page Breaks

Use the CSS **`page-break-before`** or **`page-break-after` properties** to control page breaks when printing the web page. Well-supported values for these properties are `always` (the page break will always occur as designated), `avoid` (if possible, the page break will occur before or after, as designated), and `auto` (default). For example, to configure a page break at a specific point in the document (in this case, right before an element assigned to the class named `newpage`), configure the CSS as shown below:

```
.newpage { page-break-before: always; }
```

Print URLs for Hyperlinks

Consider whether a person would find it useful to see the actual URL for a resource when reading the printout of your web page. You can use CSS to display the value of the href attribute right on the page using two CSS coding techniques: a CSS pseudo-element and the CSS `content` property. The purpose of a CSS **pseudo-element** is to apply some type of effect to its selector. Table 7.2 lists pseudo-elements and their purpose.

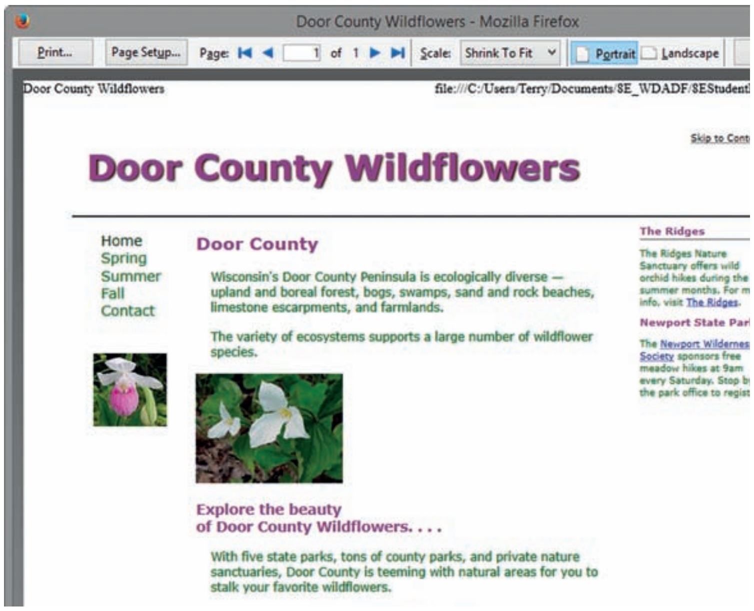
Table 7.2 CSS 2.1 pseudo-elements

Pseudo-element	Purpose
`:after`	Inserts generated content after the selector: Configure the generated content with the content property
`:before`	Inserts generated content before the selector: Configure the generated content with the content property
`:first-letter`	Applies styles to the first letter of text
`:first-line`	Applies styles to the first line of text

Use the CSS **`content` property** along with the `:after` and `:before` pseudo-elements to generate content. A useful feature of the content property is the `attr(X)` function, which returns the value of the HTML attribute provided. You can use this to print the URL of a hyperlink (the value of the `href` attribute). Use quotation marks to contain additional text or characters, such as parentheses. The CSS below will display the URL for each hyperlink in the aside element within parentheses after the text hyperlink.

```
aside a:after { content: " (" attr(href) ") "; }
```

Figure 7.13 shows the print preview of the content page you created in Hands-On Practice 7.5 (see Figure 7.10). Notice that the print preview includes the navigation area.



Door County Wildflowers file:///C:/Users/Terry/Documents/8E_WDADF/8EStudent

Skip to Cont

Door County Wildflowers

Home
Spring
Summer
Fall
Contact

Door County

Wisconsin's Door County Peninsula is ecologically diverse — upland and boreal forest, bogs, swamps, sand and rock beaches, limestone escarpments, and farmlands.

The variety of ecosystems supports a large number of wildflower species.

Explore the beauty
of Door County Wildflowers. . . .

With five state parks, tons of county parks, and private nature sanctuaries, Door County is teeming with natural areas for you to stalk your favorite wildflowers.

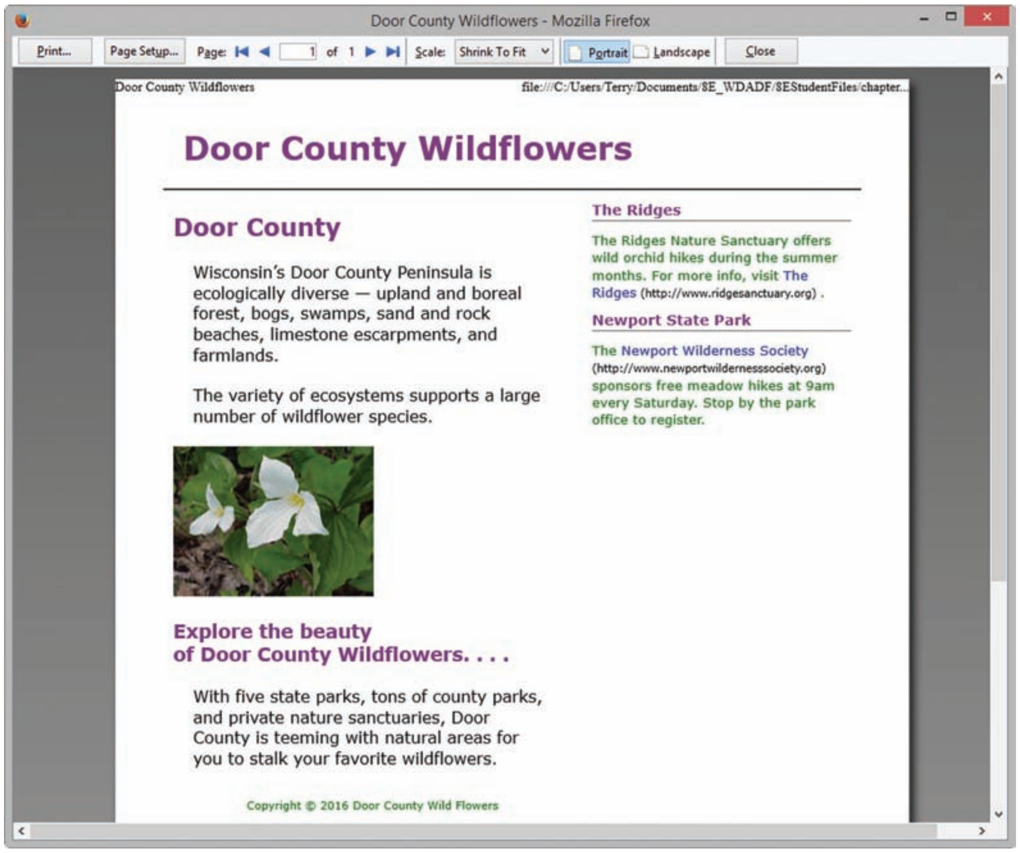
The Ridges

The Ridges Nature Sanctuary offers wild orchid hikes during the summer months. For m info, visit The Ridges.

Newport State Par

The Newport Wildernes Society sponsors free meadow hikes at 9am every Saturday. Stop b the park office to regist

Figure 7.13 Print preview of the page displayed in Figure 7.10.
Screenshots of Mozilla Firefox. Courtesy of Mozilla Foundation



Door County Wildflowers file:///C:/Users/Terry/Documents/8E_WDADF/8EStudentFiles/chapter...

Door County Wildflowers

Door County

Wisconsin's Door County Peninsula is ecologically diverse — upland and boreal forest, bogs, swamps, sand and rock beaches, limestone escarpments, and farmlands.

The variety of ecosystems supports a large number of wildflower species.

Explore the beauty of Door County Wildflowers. . . .

With five state parks, tons of county parks, and private nature sanctuaries, Door County is teeming with natural areas for you to stalk your favorite wildflowers.

Copyright © 2016 Door County Wild Flowers

The Ridges

The Ridges Nature Sanctuary offers wild orchid hikes during the summer months. For more info, visit The Ridges (http://www.ridgesanctuary.org) .

Newport State Park

The Newport Wilderness Society (http://www.newportwildernesssociety.org) sponsors free meadow hikes at 9am every Saturday. Stop by the park office to register.

Figure 7.14 Print preview using CSS to configure for printing. Screenshots of Mozilla Firefox. Courtesy of Mozilla Foundation.

Figure 7.14 displays an alternate version of the page that uses CSS to prevent the display of the navigation area, configures pt units for font sizes, and prints the URL for the hyperlinks in the sidebar area. You will explore these techniques in the next Hands-On Practice.

Hands-On Practice 7.6

In this Hands-On Practice, you will code special styles for use when printing a web page. We will use the Door County Wildflowers index.html page from Hands-On Practice 7.5 as a starting point. Figure 7.10 shows a browser display of the index.html page. You will create a new version of index.html that is associated with two external style sheets: one for the screen display and the other for printing. When you are finished, your printed page should resemble Figure 7.14.

Getting Started

Create a new folder named wildflowersPrint. Copy index.html, plsthumb.jpg, showybg.jpg, and trillium.jpg from the student files chapter7/7.5 folder into your wildflowersPrint folder.

Part 1: Create a Style Sheet for Screen Display

Launch a text editor and open index.html. Copy all of the style rules between the opening and closing style tags. Use your text editor to create a new file and save it as wildflower.css in your wildflowersPrint folder. Paste the style rules into wildflower.css and save the file.

Part 2: Edit the HTML

Edit index.html and delete the embedded CSS and style tags. Code a link element to associate the web page with the external style sheet that you just created (wildflower.css). Add the `media` attribute with the value of `"screen"` to the link element for wildflower.css. Code a second link element to invoke an external style sheet called wildflowerprint.css for printing (`media="print"`). The HTML is as follows:

```
<link rel="stylesheet" href="wildflower.css" media="screen">
<link rel="stylesheet" href="wildflowerprint.css" media="print">
```

Save index.html in the wildflowersPrint folder.

Part 3: Code the New Style Sheet for Print Display

Launch a text editor and open wildflower.css. Because you want to keep most of the styles for printing, you will modify a copy of the screen display external style sheet. Save wildflower.css as wildflowerprint.css in the wildflowersPrint folder. You will modify several areas on this style sheet, including the `container` id, h1 element selector, header element selector, nav element selector, main element selector, and the hyperlinks located in the header and aside elements.

1. Modify the h1 element selector so that the printer will use 24 point font size. Remove the `text-align` property.

```
h1 { margin-top: 0;
     font-size: 24pt;
     text-align: left }
```

2. Modify the header element selector style rules. Configure a white background color (`#ffffff`). Remove the background image declarations. The height property is also not needed because there is no longer a background image. Also remove the text alignment property.

```
header { color: #cc66cc;
         background-color: #ffffff;
         border-bottom: 2px solid #000000;
         padding: 0 20px; }
```

3. Configure the “Skip to Content” hyperlink in the header to not display. Replace all style rules associated with `header a` with the following:

```
header a { display: none; }
```

4. Configure the navigation area to not display. Remove all style rules associated with nav element selector and add the following:

```
nav { display: none; }
```

5. Remove the styles for the `header a:link`, `header a:visited`, `header a:focus`, and `header a:hover` selectors.

6. Configure the main content area. Modify the style rules for main element selector. Set the left margin to 0, the right margin to 40%, text color to black (#000000), and the font size to 12pt.

```
main { margin: 0  40%  0  0;
       padding: 1px 10px 20px 10px;
       font-size: 12pt;
       background-color: #ffffff;
       color: #000000;  }
```

7. Configure the width of the sidebar area to 40%.

```
aside { float: right;
        width: 40%; }
```

8. Configure the `news` class. Modify the style rule to configure the font size to 10pt.

```
.news {  font-size: 10pt;
         margin: 10px; }
```

9. Configure the hyperlinks in the sidebar area (the area within the aside element) so that the text is not underlined and the URLs are printed in 8pt black font. Add the following CSS:

```
aside a { text-decoration: none; }
aside a:after { content: " (" attr(href) ") ";
                font-size: 8pt;
                color: #000000; }
```

10. Look through the style rules and configure the `background-color` for each h2 and h3 element selector to be white (#ffffff).

11. Locate the `#container` selector and remove the `min-width` style declaration.

Save your file in the wildflowersPrint folder.

Part 4: Test Your Work

Test index.html in a browser. Select File > Print > Preview. Your display should look similar to the page shown in Figure 7.14. The font sizes have been configured and the navigation does not display. The URLs appear after the hyperlink text in the sidebar area. The student files contain a solution in the chapter7/print folder.

Note that there is a change in CSS3 syntax, which requires two colons in front of each pseudo-element (for example, CSS3 uses `::after` instead of `:after`). However, we'll stick with the CSS2 pseudo-elements and syntax for now because there is broader browser support.

7.5 Designing for the Mobile Web

Chapter 5 introduced you to three methods that can be used to provide access for website visitors who use mobile devices. One option is to design and publish a second website with a .mobi TLD. Another option is to design and publish a separate website within your own domain that is optimized for mobile use. This technique is utilized by the White House at http://www.whitehouse.gov. (desktop) and http://m.whitehouse.gov (mobile).

The third option is to configure one website with separate styles for desktop browser (Figure 7.15) and mobile (Figure 7.16) display. Before we focus on coding, let's consider design techniques for the mobile web.

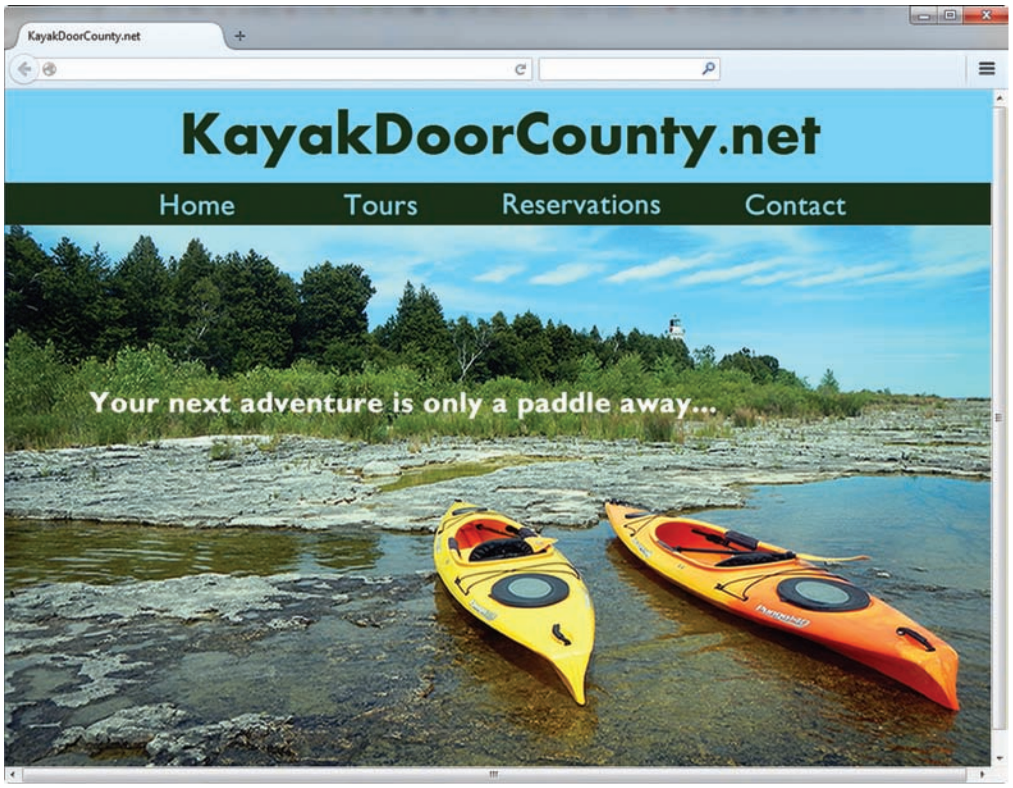


Figure 7.15 The desktop browser display

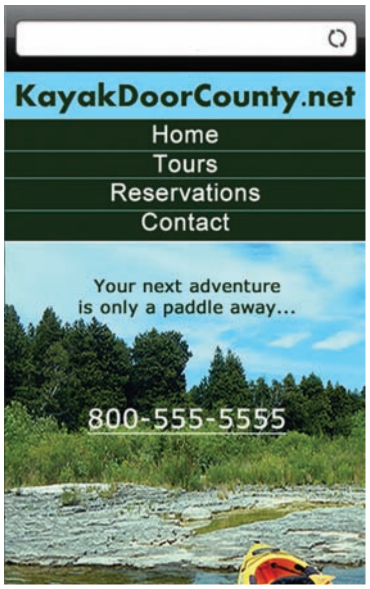


Figure 7.16 The mobile display

Mobile Web Design Best Practices

Mobile web users are typically on-the-go, need information quickly, and may be easily distracted. A web page that is optimized for mobile access should try to serve these needs. Take a moment to review Figures 7.15 and 7.16 and observe how the design of the mobile website addresses the design considerations discussed in Chapter 5:

- **Small screen size.** The size of the header area is reduced to accommodate a small screen display.
- **Low bandwidth (slow connection speed).** Note that a smaller version of the large image visible in Figure 7.15 is displayed on the mobile version (Figure 7.16).
- **Font, color, and media issues.** Common font typefaces are utilized. There is also good contrast between text and background color.
- **Awkward controls, and limited processor and memory.** The mobile website uses a single-column page layout that facilitates keyboard tabbing and will be easy to control by touch. The page is mostly text, which will be quickly rendered by a mobile browser.
- **Functionality.** Hyperlinks to popular site features display directly under the header. A phone number is prominently provided.

Let's build on this base of design considerations and expand them.

Optimize Layout for Mobile Use

A single-column page layout (Figure 7.17) with a small header, key navigation links, content, and page footer works well for a mobile device display. Mobile screen resolutions vary greatly (for example, 320×240, 360×640, 480×800, 640×690, and 640×1136). W3C recommendations include the following:

- Limit scrolling to one direction.
- Use heading elements.
- Use lists to organize information (such as unordered lists, ordered lists, and definition lists).
- Avoid using tables (see Chapter 8) because they typically force both horizontal and vertical scrolling on mobile devices.
- Provide labels for form controls (see Chapter 9).
- Avoid using pixel units in style sheets.
- Avoid absolute positioning in style sheets.
- Hide content that is not essential for mobile use.

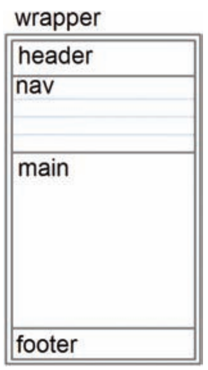


Figure 7.17 Wireframe for a typical single-column mobile page layout

Optimize Navigation for Mobile Use

Easy-to-use navigation is crucial on a mobile device. The W3C recommends the following:

- Provide minimal navigation near the top of the page.
- Provide consistent navigation.
- Avoid hyperlinks that open files in new windows or pop-up windows.
- Try to balance both the number of hyperlinks on a page and the number of levels of links needed to access information.

Optimize Graphics for Mobile Use

Graphics can help to engage visitors, but be aware of the following W3C recommendations for mobile use:

- Avoid displaying images that are wider than the screen width (assume a 320 pixel screen width on a smartphone display).
- Configure alternate small, optimized background images.
- Some mobile browsers will downsize all images, so images with text may become difficult to read.
- Avoid the use of large graphic images.
- Specify the size of images.
- Provide alternate text for graphics and other non-text elements.

Optimize Text for Mobile Use

It can be difficult to read text on a small mobile device. The following W3C recommendations will aid your mobile visitors.

- Configure good contrast between text and background colors.
- Use common font typefaces.
- Configure font size with em units or percentages.
- Use a short, descriptive page title.

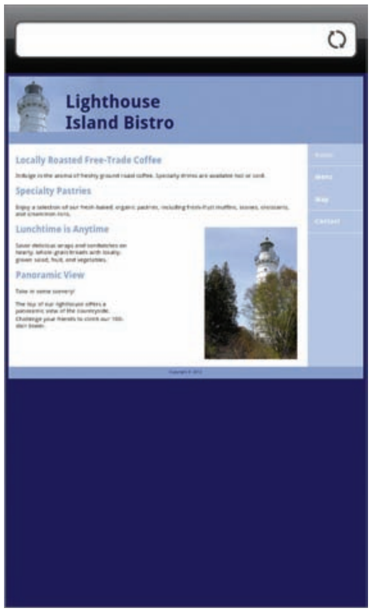


Figure 7.18 Mobile display of a web page without the viewport meta tag

The W3C has published Mobile Web Best Practices 1.0, a list of 60 mobile web design best practices, at http://www.w3.org/TR/mobile-bp. Visit http://www.w3.org/2007/02/mwbp_flip_cards.html for flipcards that summarize the Mobile Web Best Practices 1.0 document.

Design for One Web

The W3C mission of building "**One Web**" refers to the concept of providing a single resource that is configured for optimal display on multiple types of devices. This is more efficient than creating multiple versions of a web document. With the "One Web" in mind, we'll next explore using the viewport meta tag and CSS media queries to target and deliver style sheets that are optimized for mobile display to mobile devices.

7.6 Viewport Meta Tag

There are multiple uses for meta tags. You've used the meta tag since Chapter 2 to configure the character encoding on a web page. Now, we'll explore the new **viewport meta tag**, which was created as an Apple extension that helps with displays on mobile devices such as iPhones and Android smartphones by setting the width and scale of the viewport. Figure 7.18 displays a screen shot of a web page displayed on an Android smartphone without the viewport meta tag. Examine Figure 7.18 and notice that the mobile device zoomed out to display the entire web page on the tiny screen. The text on the web page is difficult to read.

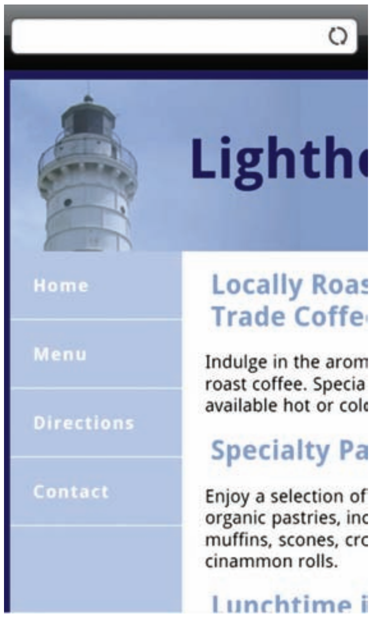


Figure 7.19 The viewport meta tag helps with mobile displays

Figure 7.19 shows the same web page after the viewport meta tag was added to the head section of the document. The code is shown below:

```
<meta name="viewport"
content="width=device-width, initial-scale=1.0">
```

Code the viewport meta tag with the HTML `name="viewport"` and `content` attributes. The value of the HTML `content` attribute can be one or more **directives** (also referred to as properties by Apple), such as the `device-width` directive and directives that control zooming and scale. Table 7.3 lists viewport meta tag directives and their values.

Table 7.3 Viewport meta tag directives

Directive	Values	Purpose
`width`	Numeric value or `device-width` which indicates actual width of the device screen	The width of the viewport in pixels
`height`	Numeric value or `device-height` which indicates actual height of the device screen	The height of the viewport in pixels
`initial-scale`	Numeric multiplier; Set to 1 for 100% initial scale	Initial scale of the viewport
`minimum-scale`	Numeric multiplier; Mobile Safari default is 0.25	Minimum scale of the viewport
`maximum-scale`	Numeric multiplier; Mobile Safari default is 1.6	Maximum scale of the viewport
`user-scalable`	`yes` allows scaling, `no` disables scaling	Determines whether a user can zoom in or out

Now that you've scaled the page to be readable, what about styling it for optimal mobile use? That's where CSS comes into play. You'll explore CSS Media Queries in the next section.

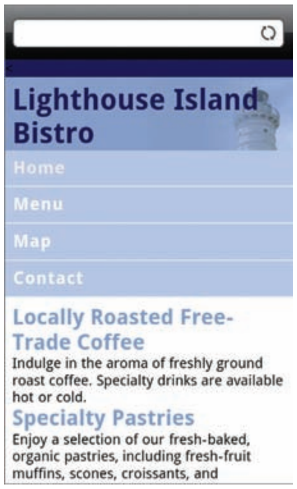
7.7 CSS3 Media Queries

Recall from Chapter 5 that the term **responsive web design** refers to progressively enhancing a web page for different viewing contexts (such as smartphones and tablets) through the use of coding techniques including fluid layouts, flexible images, and media queries.

For examples of the power of responsive web design techniques, review Figures 5.43, 5.44, and 5.45, which are actually the same .html web page file that was configured with CSS to display differently, depending on the viewport size detected by media queries. Also visit the Media Queries website at http://mediaqueri.es to view a gallery of sites that demonstrate responsive web design. The screen captures in the gallery show web pages displayed with the following browser viewport widths: 320px (smartphone display), 768px (tablet portrait display), 1024px (netbook display and tablet landscape display), and 1600px (large desktop display).

What's a Media Query?

According to the W3C (http://www.w3.org/TR/css3-mediaqueries) a **media query** is made up of a media type (such as screen) and a logical expression that determines the capability of the device that the browser is running on, such as screen resolution and orientation (portrait or landscape). When the media query evaluates as true, the media query directs browsers to CSS you have coded and configured specifically for those capabilities. Media queries are supported by current versions of major browsers, including Internet Explorer (version 9 and later).



Locally Roasted Free-Trade Coffee
Indulge in the aroma of freshly ground roast coffee. Specialty drinks are available hot or cold.
Specialty Pastries
Enjoy a selection of our fresh-baked, organic pastries, including fresh-fruit muffins, scones, croissants, and

Figure 7.20 CSS media queries help to configure the page for mobile display

Media Query Example Using a Link Element

Figure 7.20 shows the same web page as Figure 7.18, but it looks quite different because of a link element that includes a media query and is associated with a style sheet configured for optimal mobile display on a popular smartphone. The HTML is shown below:

```
<link href="lighthousemobile.css" rel="stylesheet"
        media="only screen and (max-width: 480px)">
```

The code sample above will direct browsers to an external stylesheet that has been configured for optimal display on the most popular smartphones. The media type value `only` is a keyword that will hide the media query from outdated browsers. The media type value `screen` targets devices with screens. Commonly used media types and keywords are listed in Table 7.4.

Table 7.4 Media types

Media Type	Value Purpose
`all`	All devices
`screen`	Computer screen display of web page
`only`	Causes older nonsupporting browsers to ignore the media query
`print`	Printout of web page

In the link element shown above, the `max-width` **media feature** is set to 480px. While there are many different screen sizes for smartphones these days, a maximum width of 480px will target the display size of many popular models. A maximum width of 768px will target many modern large smartphones and small tablets. A media query may test for both minimum and maximum values. For example,

```
<link href="lighthousetablet.css" rel="stylesheet"
      media="only screen and (min-width: 768px) and (max-width: 1024px)">
```

Media Query Example Using an `@media` Rule

A second method of using media queries is to code them directly in your CSS using an **`@media` rule**. Begin by coding `@media` followed by the media type and logical expression. Then enclose the desired CSS selector(s) and declaration(s) within a pair of braces. The sample code below configures a different background image specifically for mobile devices with small screens. Table 7.5 lists commonly used media query features.

```
@media only screen and (max-width: 480px) {
  header { background-image: url(mobile.gif);
  }
}
```

Table 7.5 Commonly used media query features

Features	Values	Criteria
`max-device-height`	Numeric value	The height of the screen size of the output device in pixels is smaller than or equal to the value
`max-device-width`	Numeric value	The width of the screen size of the output device in pixels is smaller than or equal to the value
`min-device-height`	Numeric value	The height of the screen size of the output device in pixels is greater than or equal to the value
`min-device-width`	Numeric value	The width of the screen size of the output device in pixels is greater than or equal to the value
`max-height`	Numeric value	The height of the viewport in pixels is smaller than or equal to the value; (reevaluated when screen is resized)
`min-height`	Numeric value	The height of the viewport in pixels is greater than or equal to the value; (reevaluated when screen is resized)
`max-width`	Numeric value	The width of the viewport in pixels is smaller than or equal to the value; (reevaluated when screen is resized)
`min-width`	Numeric value	The width of the viewport in pixels is greater than or equal to the value; (reevaluated when screen is resized)
`orientation`	Portrait or landscape	The orientation of the device

FAQ What values should I use in my media queries?

Web developers often check the `max-width` feature to determine the type of device being used to view a page: smartphone, tablet, and desktop. When choosing values, be aware of the way the navigation and content display as the viewport narrows and widens. Choose values for your media queries with the web page display in mind.

You'll see many media queries that target smartphones with a maximum width of 480 pixels, but modern smartphones have even higher screen resolutions. Here is a typical media query to target smartphone display which checks for a max-width value of 768 pixels:

```
@media only all and (max-width: 768px) {
}
```

It's now common practice to configure a tablet display when the width is between 769 pixels and 1024 pixels. For example,

```
@media only all and (min-width: 769px) and (max-width: 1024px) {
}
```

Another approach focuses on responsive display of the content instead of focusing on pixel units and configures the media queries as needed for the content to reflow on a variety of screen sizes with em units. Michael Barrett (http://blog.abouthalf.com/development/ems-in-css-media-queries/) suggests targeting smartphone display with `max-width:37.5em`, tablet display with `min-width:37.5em` and `max-width:64em`, and desktop display with `min-width:64em`. Using this approach, a media query to target a typical smartphone would check for a `max-width` feature value of 37.5em units.

For example,

```
@media only all and (max-width: 37.5em) {
}
```

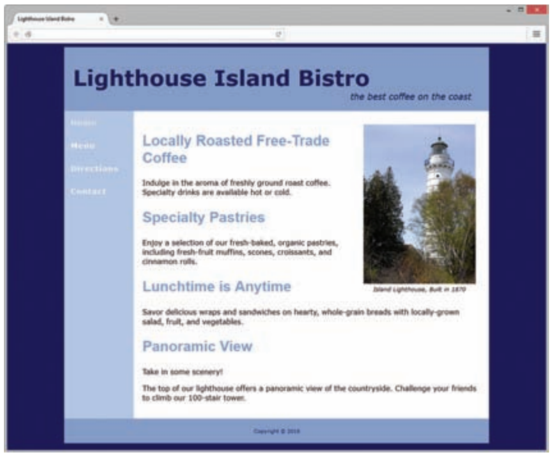
Testing on a variety of devices is very useful when configuring media queries. Be flexible and prepared to adjust media query pixel and em values to best display the content of your website on smartphones, tablets, and desktops.

Visit the following resources for more information about media query breakpoints and viewport sizes:

- http://css-tricks.com/snippets/css/media-queries-for-standard-devices
- http://reports.quickleft.com/css
- http://viewportsizes.com
- https://css-tricks.com/snippets/css/retina-display-media-query/
- https://developers.google.com/web/fundamentals/layouts/rwd-fundamentals/how-to-choose-breakpoints

Hands-On Practice 7.7

In this Hands-On Practice you'll rework a version of the two-column Lighthouse Island Bistro home page (Figure 7.21) to display a single-column page when the viewport size is a maximum of 1024 pixels (a typical tablet display) and display a page further optimized for smartphone display when the viewport size is 768 pixels or smaller. Create a folder named query7. Copy the starter3.html file from the chapter7 folder into the query7 folder and rename it index.html. Copy the lighthouseisland.jpg file from the student files chapter7/starters folder into the query7 folder. Launch a browser and view index.html as shown in Figure 7.21. Open index.html in a text editor. Review the embedded CSS and note that the two-column layout is fluid with an 80% width. The two-column look is accomplished by configuring a nav element that floats to the left.



Lighthouse Island Bistro

the best coffee on the coast

Locally Roasted Free-Trade Coffee

Indulge in the aroma of freshly ground roast coffee. Specialty drinks are available hot or cold.

Specialty Pastries

Enjoy a selection of our fresh-baked, organic pastries, including fresh-fruit muffins, scones, croissants, and cinnamon rolls.

Lunchtime is Anytime

Savor delicious wraps and sandwiches on hearty, whole-grain breads with locally-grown salad, fruit, and vegetables.

Panoramic View

Take in some scenery!

The top of our lighthouse offers a panoramic view of the countryside. Challenge your friends to climb our 100-stair tower.

Figure 7.21 The two-column desktop display

1. Edit the embedded CSS. Add the following `@media` rule before the ending style tag. The `@media` rule will configure styles that take effect when the viewport size is 1024 pixels or smaller: eliminate the left margin on the main element and change the float and width properties configured for the nav element selector.

```
@media only screen and (max-width: 1024px) {
 main { margin-left: 0; }
 nav { float: none; width: auto; }
}
```

2. Save the index.html file. Test your file in a desktop browser. When the browser is maximized, the page should look similar to Figure 7.21. When you resize the browser to be smaller (width less than or equal to 1024 pixels), the page should look similar to Figure 7.22 with a single column layout. As you can see, we still have some work to do.

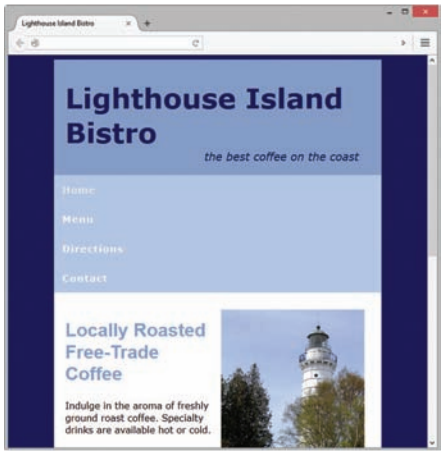

Lighthouse Island Bistro
Lighthouse Island Bistro
the best coffee on the coast
Home
Menu
Directions
Contact
Locally Roasted Free-Trade Coffee
Indulge in the aroma of freshly ground roast coffee. Specialty drinks are available hot or cold.

Figure 7.22 The media query has been applied

3. Edit the embedded CSS and add style rules within the media query that remove the margin on the body element selector, and expand the wrapper id. Also configure the nav area li elements with inline-block display and padding, the nav area ul elements with centered text, the nav area anchor elements with no border, the h1 and h2 elements with 120% font size, and the p elements with 90% font size.

```
@media only screen and (max-width: 1024px) {
  body { margin: 0; }
  #wrapper { width: auto; }
  main { margin-left: 0; }
  nav { float: none; width: auto; }
  nav li { display: inline-block; padding: 0.5em; }
  nav ul { text-align: center; }
  nav a { border-style: none; }
  h1, h2 { font-size: 120%; }
  p { font-size: 90%; }
}
```

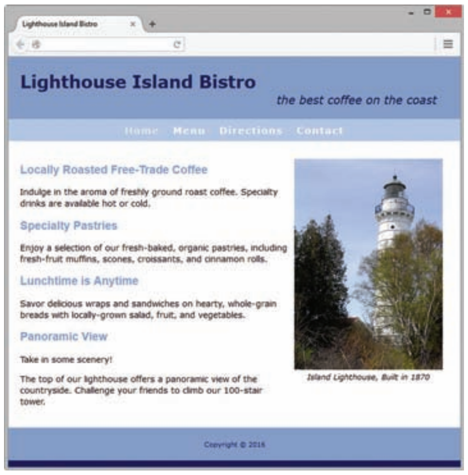
4. Save the file. Test your file in a desktop browser. When the browser is maximized, the page should look similar to Figure 7.21. When you resize the browser to be smaller (a width less than or equal to 1024 pixels), the page should look similar to Figure 7.23. Continue to resize the web page and notice that the hyperlinks will shift and are not well-aligned. We still have more work to do to optimize the page for display on small mobile devices.

5. Edit the embedded CSS. Add the following `@media` rule before the ending style tag. The `@media` rule will further configure the following styles: font-size for h1, h2, and p elements, no display of the figure element (and its child image element), no padding for the nav element and its child ul and li elements, block display for the nav li selector, and block display with a border and padding for the anchor elements in the navigation area. The code follows:

```
@media only screen and (max-width: 768px) {
  h1, h2 { font-size: 100%; }
  p { font-size: 90%; }
  figure { display: none;}
  nav, nav ul, nav li { padding: 0; }
  nav li { display: block; }
  nav a { display: block; padding: 0.5em 0;
          border-bottom: 2px ridge #00005D; }
 }
```

6. Save the file. Test your file in a desktop browser. When the browser is maximized, the page should look similar to Figure 7.21. When you resize the browser to be

smaller (width equal to or less than 1024 pixels and greater than 768 pixels), the page should look similar to Figure 7.23 with a single column layout. When you resize the browser to be even smaller (width equal to or less than 768 pixels) the page should look similar to Figure 7.24. The web page you've created is an example of applying responsive web design techniques. The student files contain a suggested solution in the chapter7/7.7 folder.



Lighthouse Island Bistro

the best coffee on the coast

Home Menu Directions Contact

Locally Roasted Free-Trade Coffee

Indulge in the aroma of freshly ground roast coffee. Specialty drinks are available hot or cold.

Specialty Pastries

Enjoy a selection of our fresh-baked, organic pastries, including fresh-fruit muffins, scones, croissants, and cinnamon rolls.

Lunchtime is Anytime

Savor delicious wraps and sandwiches on hearty, whole-grain breads with locally-grown salad, fruit, and vegetables.

Panoramic View

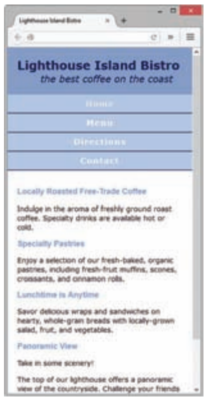
Take in some scenery!

The top of our lighthouse offers a panoramic view of the countryside. Challenge your friends to climb our 100-stair tower.

Island Lighthouse, Built in 1870

Copyright © 2016

Figure 7.23 The web page is configured for the width of a typical tablet



Lighthouse Island Bistro

the best coffee on the coast

Home

Menu

Directions

Contact

Locally Roasted Free-Trade Coffee

Indulge in the aroma of freshly ground roast coffee. Specialty drinks are available hot or cold.

Specialty Pastries

Enjoy a selection of our fresh-baked, organic pastries, including fresh-fruit muffins, scones, croissants, and cinnamon rolls.

Lunchtime is Anytime

Savor delicious wraps and sandwiches on hearty, whole-grain breads with locally-grown salad, fruit, and vegetables.

Panoramic View

Take in some scenery!

The top of our lighthouse offers a panoramic view of the countryside. Challenge your friends

Figure 7.24 The web page is configured for the width of a typical smartphone

7.8 Responsive Images

In his book, **Responsive Web Design**, Ethan Marcotte described a **flexible image** as a fluid image that will not break the page layout as the browser viewport is resized. Flexible images (often referred to as responsive images), along with fluid layouts and media queries, are the components of responsive web design. You will be introduced to several different coding techniques to configure a responsive image in this section.

Flexible Images with CSS

The most widely supported technique to configure an image as flexible requires a change to the HTML and additional CSS to style the flexible image.

1. Edit the img elements in the HTML. Remove the height and width attributes.
2. Configure the `max-width: 100%;` style declaration in the CSS. If the width of the image is less than the width of the container element, the image will display with its actual dimensions. If the width of the image is greater than the width of the container element, the image will be resized by the browser to fit in the container (instead of hanging out over the margin).
3. To keep the dimensions of the image in proportion and maintain the aspect ratio of the image, Bruce Lawson suggests to also set the `height: auto;` style declaration in the CSS (see http://brucelawson.co.uk/2012/responsive-web-design-preserving-images-aspect-ratio).

Background images can also be configured for a more fluid display at various viewport sizes. Although it's common to code a height property when configuring a background image with CSS, the result is a somewhat non-responsive background image. Explore configuring other CSS properties for the container such as font-size, line-height, and padding in percentage values. The background-size: cover; property can also be useful. You'll typically see a more pleasing display of the background image in various-sized viewports. Another option is to configure different image files to use for backgrounds and use media queries to determine which background image is displayed. A disadvantage to this option is that multiple files are downloaded although only one file is displayed. You'll apply flexible image techniques in the next Hands-On Practice.

Hands-On Practice 7.8

In this Hands-On Practice you'll work with a web page that demonstrates responsive web design. Figure 7.25 depicts the three-column desktop browser display and shows the effects of media queries which are configured to display a two-column page when the viewport size is 1024 pixels or smaller (a tablet display) and display a single-column page optimized for smartphone display when the viewport size is 768 pixels or smaller. You will edit the CSS to configure flexible images.

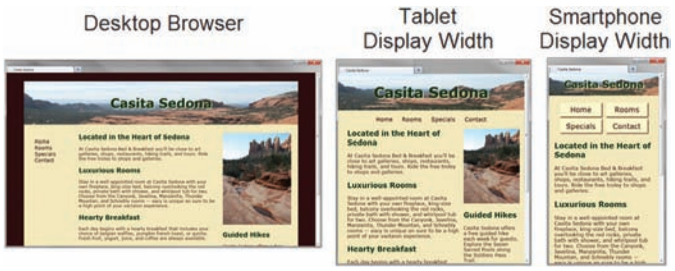


Figure 7.25 The web page demonstrates responsive web design techniques

Create a folder named flexible7. Copy the starter4.html file from the chapter7 folder into the flexible7 folder and rename it index.html. Copy the following images from the student files chapter7/starters folder into the flexible7 folder: header.jpg and pools.jpg. Launch a browser and view index.html as shown in Figure 7.26. View the code in a text editor and notice that the `height` and `width` attributes have already been removed from the HTML. View the CSS and notice that the web page uses a fluid layout with percentage values for widths. Edit the embedded CSS.

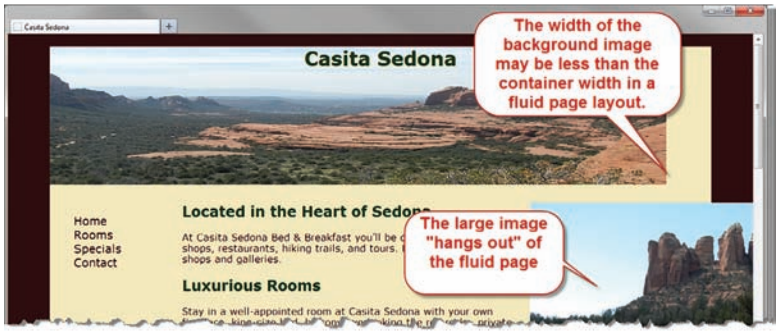

Casita Sedona
The width of the background image may be less than the container width in a fluid page layout.
Home
Rooms
Specials
Contact
Located in the Heart of Sedona
At Casita Sedona Bed & Breakfast you'll be close to all galleries, shops, restaurants, hiking trails, and tours. shops and galleries.
The large image "hangs out" of the fluid page
Luxurious Rooms
Stay in a well-appointed room at Casita Sedona with your own

Figure 7.26 The web page before the images are configured to be flexible

1. Locate the h1 element selector. Remove the height style declaration. Add declarations to set the font size to 300%, top padding 5%, bottom padding 5%, 0 left padding, and 0 right padding. The CSS follows:

```
h1 { text-align: center;
     font-size: 300%;
     padding: 5% 0;
     text-shadow: 3px 3px 3px #F4E8BC; }
```

2. Locate the header element selector. Add the `background-size: cover;` declaration to cause the browser to scale the background image to fill the container. The CSS follows:

```
header { background-image: url(header.jpg);
         background-repeat: no-repeat;
         background-size: cover; }
```

3. Add a style rule for the img element selector that sets maximum width to 100% and height to the value auto. The CSS follows:

```
img { max-width: 100%;
      height: auto; }
```

4. Save the index.html file. Test your index.html file in a desktop browser. As you resize the browser window, you'll see your page respond and look similar to the screen captures in Figure 7.25. The web page demonstrates responsive web design with the following techniques: fluid layout, media queries, and flexible images. A suggested solution is in the student files chapter7/7.8 folder.

You just applied basic techniques for configuring flexible, fluid images. These techniques should work well on popular browsers. Next, you'll explore two new HTML5.1 responsive image techniques which offer even more options when designing web pages with responsive images.

HTML5.1 Picture Element

New to HTML5.1 (http://www.w3.org/TR/html51) and in working draft status at the time this was written, the purpose of the **picture element** is to provide a method for a browser to display different images depending on specific criteria indicated by the web developer. At the current time the picture element is only supported by recent versions of Firefox, Chrome, and Opera. Check http://caniuse.com/picture for the current level of browser support. The picture element begins with the `<picture>` tag and ends with the `</picture>` tag. The picture element is a container element that is coded along with source elements and a fallback img element to provide multiple image files that can be chosen for display by the browser.

Source Element

The **source element** is a self-contained, or void, tag that is used together with a container element. The picture element is one of several elements (see the video and audio elements in Chapter 11) that can contain one or more source elements. When used with a picture element, multiple source elements are typically configured to specify different images. Code the source elements between the opening and closing picture tags. Table 7.6 lists attributes of the source element when coded within a picture element container.

Table 7.6 Attributes of the source element

Attribute	Value
`srcset`	Required. Provides image choices for the browser in a comma-separated list. Each item can contain the image URL (required), optional maximum viewport dimension, and optional pixel density for high resolution devices.
`media`	Optional. Media query to specify conditions for browser display.
`sizes`	Optional. Numeric or percentage value to specify the dimensions of the image display. May be further configured with a media query.

There are many potential ways to configure responsive images with the picture and source elements. We will focus on a basic technique that uses the media attribute to specify conditions for display.

Hands-On Practice 7.9

In this Hands-On Practice you will configure responsive images with the picture, source, and img elements as you create the page shown in Figure 7.27.

Create a new folder named ch7picture. Copy the large.jpg, medium.jpg, small.jpg, and fallback.jpg files from the chapter7/starters folder into your ch7picture folder. Launch a text editor and open the template file located at chapter2/template.html in the student files. Save the file as index.html in your ch7picture folder. Modify the file to configure a web page as indicated:

1. Configure the text, Picture Element, within an h1 element and within the title element.
2. Code the following in the body of the web page:

```
<picture>
    <source media="(min-width: 1200px)" srcset="large.jpg">
    <source media="(min-width: 800px)" srcset="medium.jpg">
    <source media="(min-width: 320px)" srcset="small.jpg">
    <img src="fallback.jpg" alt="waterwheel">
</picture>
```

Save your file and test your page in a current version of Firefox or Chrome. Notice how a different image is displayed depending on the width of the browser viewport. If the

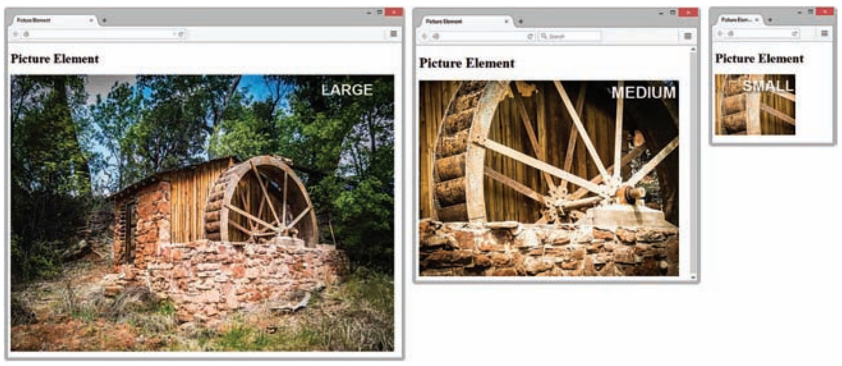


Figure 7.27 *Responsive image with the picture element*

viewport's minimum width is 1200px or greater, the large.jpg image is shown. If the viewport's minimum width is 800px or greater but less than 1200px, the medium.jpg image is displayed. If the viewport's minimum width is 320px greater but less than 800px, the small.jpg image is shown. If none of these criteria are met, the fallback.jpg image should be displayed. As you test, try resizing and refreshing the browser display. You may need to resize the browser, close it, and launch it again to test for display of the different images. Browsers that do not support the new picture element will process the img tag and display the fallback.jpg image. A suggested solution is in the student files chapter7/7.9 folder.

This Hands-On Practice provided a very basic example of responsive images with the picture element. The picture and element responsive image technique is intended to eliminate multiple image downloads that can occur with CSS flexible image techniques. The browser downloads only the image it chose to display based on the criteria provided.

HTML5.1 Responsive Img Element Attributes

New to HTML5.1 (http://www.w3.org/TR/html51) and in working draft status at the time this was written, the new `srcset` and `sizes` attributes have been created for the img element. At the current time the new attributes are only supported by recent versions of Firefox, Chrome, and Opera. Check http://caniuse.com/srcset for the current level of browser support.

The `sizes` Attribute

The purpose of the img element's **`sizes` attribute** is to inform the browser as it processes the srcset attribute about how much of the viewport should be used to display the image. The default value of the sizes attribute is `100vw`, which indicates 100% of the viewport width is available to display the image. The value of the sizes attribute can be a percentage of the viewport width or a specific pixel width (such as 400px). The sizes attribute can also contain one or more media queries along with the width for each condition.

The `srcset` Attribute

The purpose of the img element's **`srcset` attribute** is to provide a method for a browser to display different images depending on specific criteria indicated by the web developer. The value of the srcset attribute provides image choices for the browser in a comma-separated list. Each list item can contain the image URL (required), optional maximum viewport dimension, and optional pixel density for high resolution devices.

There are many potential ways to configure responsive images with the img element, sizes, attribute, and srcset attribute. We will focus on a basic technique that uses the browser viewport dimension to specify conditions for display.

Hands-On Practice 7.10

In this Hands-On Practice you will configure responsive images with the picture, source, and img elements as you create the page shown in Figure 7.28.

Create a new folder named ch7image. Copy the large.jpg, medium.jpg, small.jpg, and fallback.jpg files from the chapter7/starters folder into your ch7image folder. Launch a

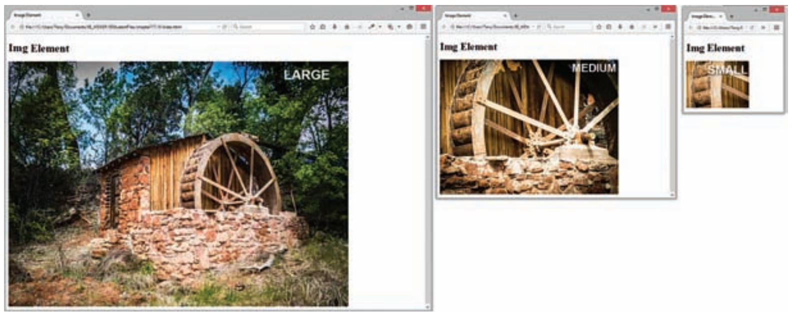


Figure 7.28 Responsive image with the image element's srcset attribute

text editor and open the template file located at chapter2/template.html in the student files. Save the file as index.html in your ch7image folder. Modify the file to configure a web page as indicated:

1. Configure the text, Img Element, within an h1 element and within the title element.
2. Code the following in the body of the web page:

```
<img src="fallback.jpg"
  sizes="100vw"
  srcset="large.jpg 1200w, medium.jpg 800w, small.jpg 320w"
  alt="waterwheel">
```

Save your file and test your page in a current version of Firefox or Chrome. Notice how a different image is displayed depending on the width of the browser viewport. If the viewport's minimum width is 1200px or greater, the large.jpg image is shown. If the viewport's minimum width is 800px or greater but less than 1200px, the medium.jpg image is displayed. If the viewport's minimum width is 320px greater but less than 800px, the small.jpg image is shown. If none of these criteria are met, the fallback.jpg image should be displayed. As you test, try resizing and refreshing the browser display. You may need to resize the browser, close it, and launch it again to test for display of the different images. Browsers that do not support the image element's new sizes and srcset attributes will ignore these attributes and display the fallback.jpg image. A suggested solution is in the student files chapter7/7.10 folder.

This Hands-On Practice provided a very basic example of responsive images with the img element and new sizes and srcset attributes which (like the picture element responsive image technique) is intended to eliminate multiple image downloads that can occur with CSS flexible image techniques. The browser downloads only the image it chose to display based on the criteria provided.

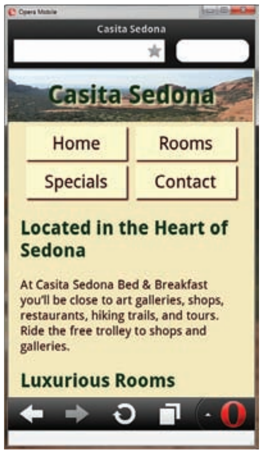
Explore Responsive Images

There is so much to learn about responsive image techniques! Visit the following resources to explore the topic of responsive images:

- http://responsiveimages.org
- http://html5hub.com/html5-picture-element
- http://www.sitepoint.com/improving-responsive-images-picture-element
- https://longhandpixels.net/blog/2014/02/complete-guide-picture-element
- http://blog.cloudfour.com/responsive-images-101-part-5-sizes

7.9 Testing Mobile Display

The best way to test the mobile display of a web page is to publish it to the Web and access it from a mobile device. (See the Appendix for an introduction to publishing a website with FTP.) However, not everyone has access to a smartphone. Several options for emulating a mobile display are listed below:



Casita Sedona

Home Rooms
Specials Contact

Located in the Heart of Sedona

At Casita Sedona Bed & Breakfast you'll be close to art galleries, shops, restaurants, hiking trails, and tours. Ride the free trolley to shops and galleries.

Luxurious Rooms

Figure 7.29 Testing a web page with the Opera Mobile Emulator. Terry Felke-Morris

- **Opera Mobile Emulator** (shown in Figure 7.29)

  Windows only download; Supports media queries
  http://www.opera.com/developer/mobile-emulator
- **Mobile Phone Emulator**

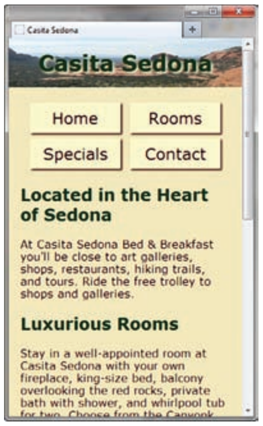
  Runs in a browser window; Supports media queries
  http://www.mobilephoneemulator.com
- **iPhone Emulator**

  Runs in a browser window; Supports media queries
  http://www.testiphone.com
- **iPadPeek**

  Runs in a browser window; Supports media queries
  http://ipadpeek.com

Testing with a Desktop Browser

If you don't have a smartphone and/or are unable to publish your files to the Web-no worries-as you've seen in this chapter (also see Figure 7.30) you can approximate the mobile display of your web page using a desktop browser. Verify the placement of your media queries.



Casita Sedona

Home Rooms
Specials Contact

Located in the Heart of Sedona

At Casita Sedona Bed & Breakfast you'll be close to art galleries, shops, restaurants, hiking trails, and tours. Ride the free trolley to shops and galleries.

Luxurious Rooms

Stay in a well-appointed room at Casita Sedona with your own fireplace, king-size bed, balcony overlooking the red rocks, private bath with shower, and whirlpool tub for two. Choose from the Canyon

Figure 7.30 Approximating the mobile display with a desktop browser. Terry Felke-Morris

- If you have coded media queries within your CSS, display your page in a desktop browser and then reduce the width and height of the viewport until it approximates a mobile screen size (such as 320×480).
- If you have coded media queries within a link tag, edit the web page and temporarily modify the link tag to point to your mobile CSS style sheet. Then, display your page in a desktop browser and reduce the width and height of the viewport until it approximates a mobile screen size (such as 320×480).

  While you can guess at the size of your browser viewport, the following tools can be helpful:
- **Chris Pederick's Web Developer Extension**

  Available for Firefox and Chrome

  http://chrispederick.com/work/web-developer

  Select Resize > Display Window Size
- **Internet Explorer Developer Tools**

  Select Tools > F12 Developer Tools > Tools > Resize

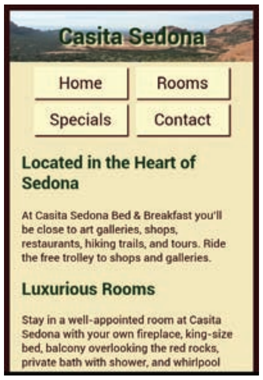
Another option for testing your responsive web pages is to use one of the following online tools that provide instant views of your web page in a variety of screen sizes and devices:

- Am I Responsive http://ami.responsivedesign.is
- DevicePonsive http://deviceponsive.com
- Responsive Test http://responsivetest.net
- Screenfly http://quirktools.com/screenfly

It's fun to view your responsive website on these browser tools. The true test, however, is to view your web pages on a variety of physical mobile devices.

For Serious Developers Only

If you are a software developer or information systems major, you may want to explore the SDKs (Software Developer Kits) for the iOS and Android platforms. Each SDK includes a mobile device emulator. Figure 7.31 shows an example screen capture. Visit http://developer.android.com/sdk/index.html for information about the Android SDK.



Located in the Heart of Sedona

At Casita Sedona Bed & Breakfast you'll be close to art galleries, shops, restaurants, hiking trails, and tours. Ride the free trolley to shops and galleries.

Luxurious Rooms

Stay in a well-appointed room at Casita Sedona with your own fireplace, king-size bed, balcony overlooking the red rocks, private bath with shower, and whirlpool

Figure 7.31 Testing the web page with a smartphone.

Media Queries and Internet Explorer

Keep in mind that Internet Explorer prior to version 9 does not support media queries. The Google code repository offers a JavaScript workaround for this issue. Add the following code in the head section of your web page to make the script available for Internet Explorer version 8 and lower.

```
<!--[if lt IE 9]>
<script src=
"http://css3-mediaqueries-js.googlecode.com/svn/trunk/css3-mediaqueries.js">
</script>
<![endif]-->
```

Mobile First

This section provided an introduction to mobile web design. The styles for desktop browser viewing were coded and the media queries were constructed to adapt the layout for mobile devices. This is a typical workflow when you need to rework an existing website for mobile device display.

However, if you are designing a new website, there is an alternate approach that was first proposed by Luke Wroblewski. Design the mobile style sheet first and then develop alternate styles for tablet and/or desktop browsers that progressively enhance the design with multiple columns and larger images. You can find out more about this “Mobile First” approach at the following resources:

- http://www.lukew.com/ff/entry.asp?933
- http://www.lukew.com/ff/entry.asp?1137
- http://www.techradar.com/news/internet/mobile-web-design-tips-mobile-should-come-first-719677

7.10 CSS3 Flexible Box Layout

You have used the CSS float property to create the look of two-column and three-column web pages. While the float layout technique is still a common practice, there is a new emerging flexbox layout technique that uses the CSS3 Flexible Box Layout Module (http://www.w3.org/TR/css3-flexbox/) which is currently in W3C Working Draft status. The purpose of **flexbox** is to provide for a flexible layout—elements contained within a flex container can be configured either horizontally or vertically in a flexible manner with flexible sizing. In addition to changing the horizontal or vertical organization of elements, flexbox can also be used to change the order of display of the elements. Due to its flexibility, flexbox is ideally suited for responsive web design!

Although you can expect increasing levels of browser support with each new version, flexbox is not yet well supported by browsers. Check http://caniuse.com/flexbox for the current level of browser support. At the time this was written, Firefox, Chrome, Opera, Internet Explorer (version 11), Microsoft Edge, and Safari (version 9) supported flexbox.

Configure a Flexible Container

To configure an area on a web page that uses flexbox layout, you need to indicate the **flex container**, which is the element that will contain the flexible area. Use the CSS `display` property to configure a flex container. The value `flex` indicates a flexible block container. The value `inline-flex` indicates a flexible inline-display container. Elements contained within a flex container are referred to as **flex items**.

The `flex-direction` Property

Configure the direction of the flex items with the **`flex-direction` property**. The value `row` configures a horizontal direction. The value `column` configures a vertical direction. For example, to configure an id named demo as a flexible block container with a horizontal flexible flow, code the following CSS:

```
#demo { display: flex;
        flex-direction: row; }
```

The `flex-wrap` Property

The **`flex-wrap` property** configures whether flex items are displayed on multiple lines. The default value is `nowrap`, which configures single-line display. The value `wrap` will allow the flex items to display on multiple lines, which could be useful for navigation or for an image gallery.

The `justify-content` Property

The **`justify-content` property** configures how the browser should display any extra space that may exist in the flex container. A list of values for the `justify-content` property is available at http://www.w3.org/TR/css3- flexbox/#justify-content-property. The value `center` will center the flex items with equal amounts of empty space before and after. The value `space-between` evenly distributes the flex items and allocates empty space between them.

Configure the Flex Items

By default, all elements contained within a flex container are flexible in size, are allocated the same amount of display area in the flex container, and display in the same order they are coded in. You can modify the default display with CSS, including the new flex and order properties.

The `flex` Property

Use the **`flex` property** to customize the size of each flex item. The value of the flex property does not correspond to a unit of measure but is proportionate to the whole. Values for the flex property include both keywords and positive numbers. See http://www.w3.org/TR/css3-flexbox/#flex-common for a list of values commonly used with the flex property. We'll focus on the numeric values in this section. The default value is 1. If you configure an element with `flex: 2;` then it will take up twice as much space within the container element as the others. Since the values work in proportion to the whole, use flex values for companion flex items that add up to 10. Examine the three-column layout in Figure 7.32 and notice how the `nav`, `main`, and `aside` elements are within another element that will serve as a flex container. The CSS to configure the proportion of the flexible area allocated to each column could be as follows:

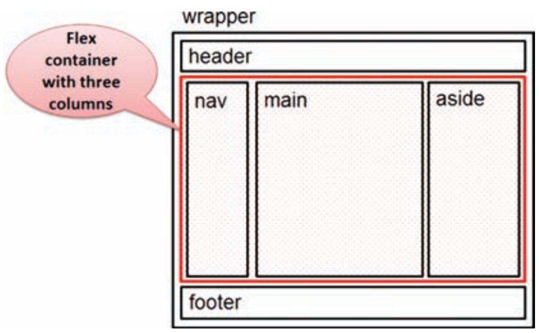


Figure 7.32 Three column page layout with the flex container indicated

```
nav   { flex: 1; }
main  { flex: 7; }
aside { flex: 2;}
```

The `order` Property

Use the **`order` property** to display the flex items in a different order than they are coded. The `order` property accepts numeric values. The default value of the order property is 0. To configure an element to display *before* the other flex items within the flex container, set its order to –1. Similarly, to configure an element to display *after* the other flex items within the flex container, set its order to 1. You'll practice using the `order` property in the next Hands-On Practice.

This section introduced several new CSS properties used with Flexible Box Layout, but there is much more to explore about this powerful new layout technique. Visit the following resources to delve deeper into flexbox:

- http://css-tricks.com/snippets/css/a-guide-to-flexbox
- http://demo.agektmr.com/flexbox
- https://developer.mozilla.org/en-US/docs/Web/Guide/CSS/Flexible_boxes

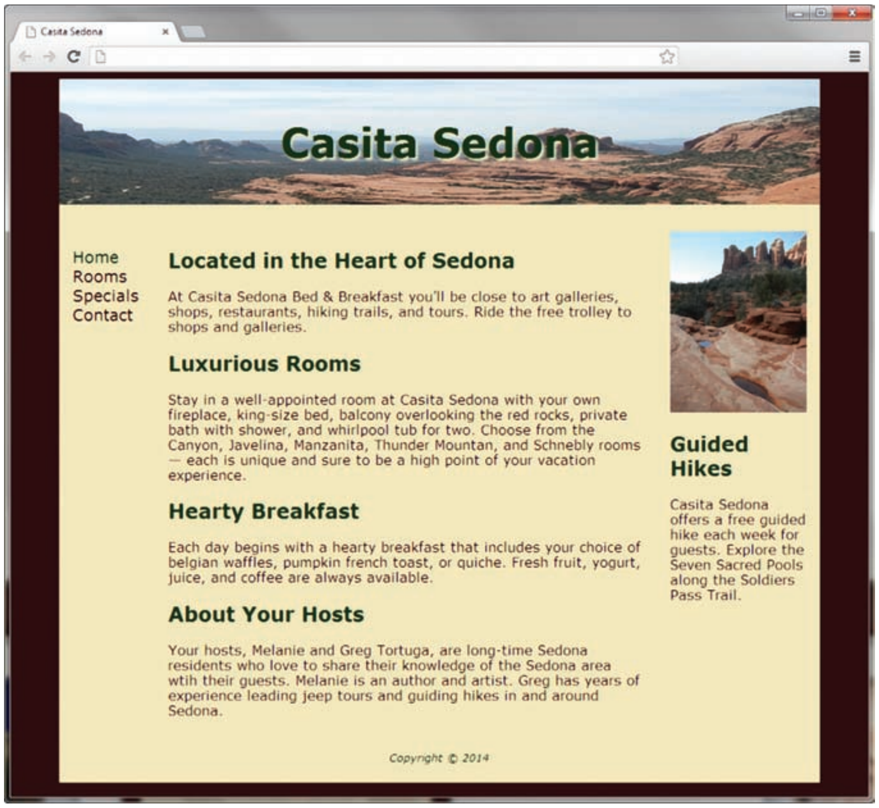
Hands-On Practice 7.11

In this Hands-On Practice you'll use flexbox layout to configure a three-column layout (Figure 7.32) on a web page and also apply media queries to modify the display for mobile devices. Figure 7.33 shows the three-column desktop browser display.

You'll configure the desktop browser display first and then modify the media queries to configure a single-column tablet display with horizontal navigation (Figure 7.34) and a single-column smartphone display with navigation reordered to appear above the footer (Figure 7.35).

Create a folder named flexbox7. Copy the starter5.html file from the chapter7 folder into the flexbox7 folder and rename it as index.html. Copy the following images from the student files chapter7/starters folder into the flexbox7 folder: header.jpg and pools.jpg. When you launch a browser to view the index.html file, it will look similar to Figure 7.36 with a single-column layout.

1. View the code in a text editor and locate the opening div tag coded above the nav element—this is the beginning of the flexible container. Edit the HTML and assign this div to an id named demo. Notice that three elements are contained within the `#demo` div: the `nav` element, the `main` element, and the `aside` element. These three elements are flex items.



Casita Sedona

Home
Rooms
Specials
Contact

Located in the Heart of Sedona

At Casita Sedona Bed & Breakfast you'll be close to art galleries, shops, restaurants, hiking trails, and tours. Ride the free trolley to shops and galleries.

Luxurious Rooms

Stay in a well-appointed room at Casita Sedona with your own fireplace, king-size bed, balcony overlooking the red rocks, private bath with shower, and whirlpool tub for two. Choose from the Canyon, Javelina, Manzanita, Thunder Mountan, and Schnebly rooms — each is unique and sure to be a high point of your vacation experience.

Hearty Breakfast

Each day begins with a hearty breakfast that includes your choice of belgian waffles, pumpkin french toast, or quiche. Fresh fruit, yogurt, juice, and coffee are always available.

About Your Hosts

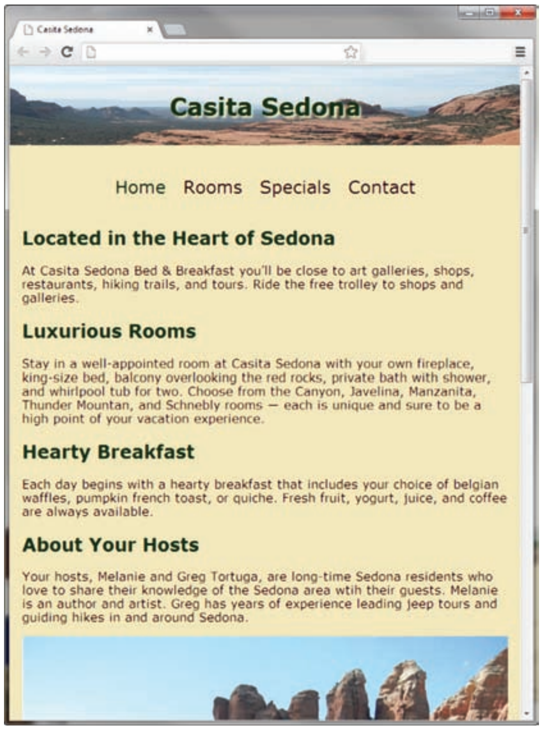
Your hosts, Melanie and Greg Tortuga, are long-time Sedona residents who love to share their knowledge of the Sedona area wtih their guests. Melanie is an author and artist. Greg has years of experience leading jeep tours and guiding hikes in and around Sedona.

Guided Hikes

Casita Sedona offers a free guided hike each week for guests. Explore the Seven Sacred Pools along the Soldiers Pass Trail.

Copyright © 2014

Figure 7.33 Three-column layout using flexbox



Casita Sedona

Home Rooms Specials Contact

Located in the Heart of Sedona

At Casita Sedona Bed & Breakfast you'll be close to art galleries, shops, restaurants, hiking trails, and tours. Ride the free trolley to shops and galleries.

Luxurious Rooms

Stay in a well-appointed room at Casita Sedona with your own fireplace, king-size bed, balcony overlooking the red rocks, private bath with shower, and whirlpool tub for two. Choose from the Canyon, Javelina, Manzanita, Thunder Mountan, and Schnebly rooms — each is unique and sure to be a high point of your vacation experience.

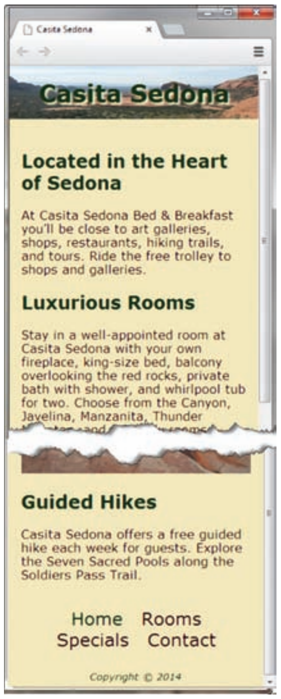
Hearty Breakfast

Each day begins with a hearty breakfast that includes your choice of belgian waffles, pumpkin french toast, or quiche. Fresh fruit, yogurt, juice, and coffee are always available.

About Your Hosts

Your hosts, Melanie and Greg Tortuga, are long-time Sedona residents who love to share their knowledge of the Sedona area wtih their guests. Melanie is an author and artist. Greg has years of experience leading jeep tours and guiding hikes in and around Sedona.

Figure 7.34 Single-column tablet display



Casita Sedona

Located in the Heart of Sedona

At Casita Sedona Bed & Breakfast you'll be close to art galleries, shops, restaurants, hiking trails, and tours. Ride the free trolley to shops and galleries.

Luxurious Rooms

Stay in a well-appointed room at Casita Sedona with your own fireplace, king-size bed, balcony overlooking the red rocks, private bath with shower, and whirlpool tub for two. Choose from the Canyon, Javelina, Manzanita, Thunder

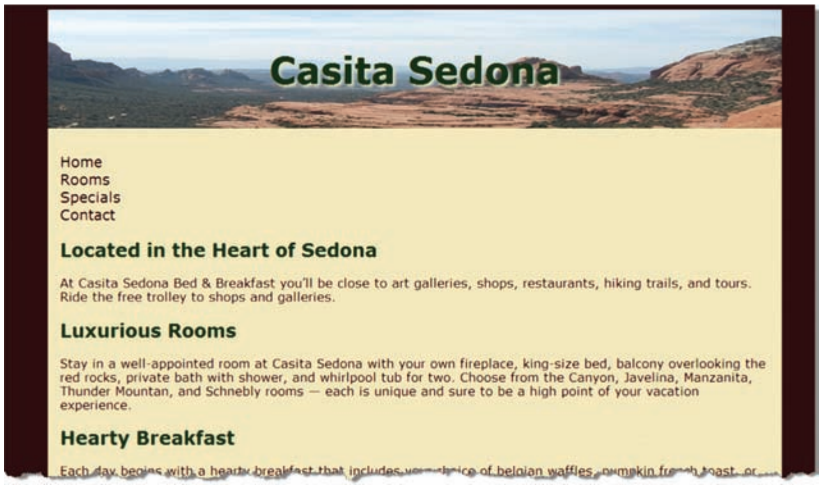
Guided Hikes

Casita Sedona offers a free guided hike each week for guests. Explore the Seven Sacred Pools along the Soldiers Pass Trail.

Home Rooms
Specials Contact

Copyright © 2014

Figure 7.35 Smartphone display



Casita Sedona

Home
Rooms
Specials
Contact

Located in the Heart of Sedona

At Casita Sedona Bed & Breakfast you'll be close to art galleries, shops, restaurants, hiking trails, and tours. Ride the free trolley to shops and galleries.

Luxurious Rooms

Stay in a well-appointed room at Casita Sedona with your own fireplace, king-size bed, balcony overlooking the red rocks, private bath with shower, and whirlpool tub for two. Choose from the Canyon, Javelina, Manzanita, Thunder Mountain, and Schnebly rooms — each is unique and sure to be a high point of your vacation experience.

Hearty Breakfast

Each day begins with a hearty breakfast that includes your choice of belgian waffles, pumpkin french toast, or

Figure 7.36 The web page before the flexbox layout is applied

2. Edit the embedded CSS above the media queries and add styles to configure the flexible layout:

a. To create a flexible container that displays its flex items in a row, configure an id selector named `demo` with `display` set to `flex` and `flex-direction` set to `row`.

```
#demo { display: flex;
        flex-direction: row; }
```

b. By default, the browser will display the flex items as the same size. The wireframe in Figure 7.32 shows the flex items with different widths. Configure CSS for the `nav` element, `main` element, and `aside` element selectors that set the `flex` property as follows:

```
nav { flex: 1;}
main { flex: 7; }
aside { flex: 2;}
```

3. Save the index.html file and display it in a browser that supports flexbox, such as Firefox, Chrome or Opera. Your display should be similar to Figure 7.33. What an easy way to configure a three column layout! However, if you resize your browser to be smaller, you'll find that we need to make some adjustments for typical tablet and smartphone display sizes.

4. Open the index.html file in a text editor and locate the first media query, which is for typical tablet display.

a. Configure the single-column display shown in Figure 7.34 by setting the `flex-direction` property for the demo id to the value `column`. Add the following CSS within the first media query:

```
#demo { flex-direction: column; }
```

b. Review Figure 7.34 and notice that the navigation hyperlinks display in a horizontal manner even though they are structured within an unordered list. To accomplish the horizontal display, configure a `nav ul` selector with `display` set to `flex`, `flex-direction` set to `row`, and `justify-content` set to `center`. Add the following CSS within the first media query:

```
nav ul { display: flex;
         flex-direction: row;
         justify-content: center; }
```

5. Locate the second media query, which is typical for smartphone display. You will add style rules to this media query to further configure the display for small mobile devices.

a. Review Figure 7.35 and note that the navigation now appears above the footer. Recall that the default flex order value is 0. Set the nav element selector's `order` property to the value 1 to cause the navigation to display after the other flex items in the container. Add the following CSS within the second media query:

```
nav { order: 1; }
```

b. Notice also in Figure 7.35 that the navigation is displayed on more than one line. Configure the `nav ul` flex container to use multiple lines by setting the `flex-wrap` property to `wrap`. Add the following CSS within the second media query:

```
nav ul { flex-wrap: wrap; }
```

6. Save the index.html file and display it in a browser that supports flexbox, such as Chrome or Opera. Your desktop display should be similar to Figure 7.33. When you resize your browser to be smaller, the display should be similar to Figure 7.34 or Figure 7.35. A suggested solution is in the student files chapter7/7.11 folder.

FAQ What happens when browsers that don't support flexbox display the web page?

Browsers that don't support flexbox will ignore the new properties and values. Flexbox is an intriguing layout technique that promises to simplify page layout for web developers. However, until browser support increases, it is best used for experimental pages. Check http://caniuse.com/flexbox for the current level of browser support.

Checkpoint 7.2

1. State an advantage of using CSS to style for print.
2. Describe a design consideration when configuring a web page for mobile display.
3. Describe coding techniques that will configure an image with a flexible display.

Chapter Summary

This chapter explored a variety of web development topics, including relative hyperlinks and linking to fragment identifiers, CSS sprites, three-column page layout, styling for print, and styling for mobile devices. Visit the textbook website at http://www.webdevfoundations.net for examples, the links listed in this chapter, and updated information.

Key Terms

`:after`
`:before`
`:first-letter`
`:first-line`
`@media` rule
Accessible Rich Internet Applications (ARIA)
`content` property
CSS sprite
directive
flex container
flex item
`flex` property
`flex-direction` property
`flex-wrap` property
flexbox
flexible image
fragment identifier
`justify-content` property
landmark
media attribute
media feature
media query
media type
named fragment
One Web
`order` property
`page-break-after` property
`page-break-before` property
picture element
pseudo-element
responsive web design
role attribute
sizes attribute
source element
sprite
srcset attribute
target attribute
viewport meta tag

Review Questions

Multiple Choice

1. Which meta tag is used to configure display for mobile devices?
a. viewport
b. handheld
c. mobile
d. screen

2. How would you link to the named fragment `#jobs` on the page employ.html from the home page of the site?
a. `<a href="employ.html#jobs">Jobs</a>`
b. `<a name="employ.html#jobs">Jobs</a>`
c. `<a link="employ.html#jobs">Jobs</a>`
d. `<a href="#jobs">Jobs</a>`

3. Which of the following is a container element used to configure responsive images?
a. display
b. flex
c. picture
d. link

4. Which attribute below can be applied to an anchor tag to open a link in a new browser window?
a. window
b. target
c. rel
d. media

5. Which of the following is the attribute used to indicate whether the style sheet is for printing or screen display?
a. rel
b. type
c. media
d. content

6. Which of the following attributes define a fragment identifier on a page?
a. id
b. href
c. fragment
d. bookmark

7. Which pseudo-element can be used to generate content that precedes an element?
   a. `:after`
   b. `:before`
   c. `:content`
   d. `:first-line`
8. Which of the following is a mobile web design best practice?
   a. Configure a multiple-column page layout.
   b. Avoid using lists to organize information.
   c. Configure a single-column page layout.
   d. Embed text in images wherever possible.
9. Which of the following font units is recommended for mobile display?
   a. pt unit
   b. px unit
   c. em unit
   d. cm unit
10. Which of the following is an image file that contains multiple small graphics?
    a. viewport
    b. sprite
    c. background-image
    d. media

Fill in the Blank

11. ____________ is an emerging page layout technique for configuring multi-column layouts.
12. _______________ determine the capability of the mobile device, such as browser viewport dimensions and resolution.
13. The concept of _______________ relates to providing a single resource that is configured for optimal display on multiple types of devices.
14. Provide _______________ navigation near the top of the page when optimizing for mobile display.
15. When using CSS media queries, code the _______________ keyword to hide the query from older nonsupporting browsers.

Apply Your Knowledge

1. **Predict the Result.** Draw and write a brief description of the web page that will be created with the following HTML code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Predict the Result</title>
<meta charset="utf-8">
<style>
body { background-color: #eaeaea;
       color: #000;
       font-family: Verdana, Arial, sans-serif; }
#wrapper { width: 80%;
           margin: auto;
           overflow: auto;
           min-width: 960px;
           background-color: #d5edb3;}
nav { float: left;
      width: 150px; }
```

```
main { margin-left: 160px;
       margin-right: 160px;
       background-color: #ffffff;
       padding: 10px; }
aside { float: right;
        width: 150px;
        color: #000000; }
</style>
</head>
<body>
<div id="wrapper">
<header role="banner">
    <h1>Trillium Media Design</h1>
</header>
<nav role="navigation">
<ul>
  <li><a href="index.html">Home</a></li>
  <li><a href="products.html">Products</a></li>
  <li><a href="services.html">Services</a></li>
  <li><a href="clients.html">Clients</a></li>
  <li><a href="contact.html">Contact</a></li>
</ul>
</nav>
<aside role="complementary">
<h2>Newsletter</h2>
<p>Get monthly updates and free offers. Contact <a
 href="mailto:me@trilliummediadesign.com">Trillium</a>
to sign up for our newsletter.</p>
</aside>
<main role="main">
<p>Our professional staff takes pride in its working relationship
with our clients by offering personalized services that listen
to their needs, develop their target areas, and incorporate these
items into a website that works. </p>
</main>
</div>
</body>
</html>
```

2. **Fill in the Missing Code.** This web page should be configured so that the left navigation column has a light green background color and floats on the left side of the browser window. Instead, the navigation displays with a white background color. CSS properties and values, indicated by "_" (underscore), are missing. Fill in the missing code to correct the error.

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Fill in the Missing Code</title>
<meta charset="utf-8">
```

```
<style>
  body { background-color: #d5edb3;
         color: #000066;
         font-family: Verdana, Arial, sans-serif; }
  nav { float: left;
      width: 120px; }
  main { "_":  "_";
      background-color: #ffffff;
      color: #000000;
      padding: 20px; }
</style>
</head>
<body>
<header role="banner">
   <h1>Trillium Media Design</h1>
</header>
<nav role="navigation">
<ul>
  <li><a href="index.html">Home</a></li>
  <li><a href="products.html">Products</a></li>
  <li><a href="services.html">Services</a></li>
  <li><a href="clients.html">Clients</a></li>
  <li><a href="contact.html">Contact</a></li>
</ul>
</nav>
<main role="main">
<p>Our professional staff takes pride in its working relationship
with our clients by offering personalized services that listen
to their needs, develop their target areas, and incorporate these
items into a website that works. </p>
</main>
</body>
</html>
```

3. **Find the Error.** The page below is intended for the navigation area to display on the right side of the browser window. What needs to be changed to make this happen?

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Find the Error</title>
<meta charset="utf-8">
<style>
   body { background-color: #d5edb3;
          color: #000066;
          font-family: Verdana, Arial, sans-serif; }
   nav { float: left;
         width: 120px; }
   main { padding: 20px 150px 20px 20px;
          background-color: #ffffff;
          color: #000000; }
</style>
```

```
</head>
<body>
<header role="banner">
   <h1>Trillium Media Design</h1>
</header>
<nav role="navigation">
  <ul>
     <li><a href="index.html">Home</a></li>
     <li><a href="services.html">Services</a></li>
     <li><a href="contact.html">Contact</a></li>
  </ul>
</nav>
<main role="main">
   <p>Our professional staff takes pride in its working
relationship with our clients by offering personalized services
that listen to their needs, develop their target areas, and
incorporate these items into a website that works.</p>
  </main>
</body>
</html>
```

Hands-On Exercises

1. Write the HTML to create a fragment identifier at the beginning of a web page designated by “top”.
2. Write the HTML to create a hyperlink to the named fragment designated by “top”.
3. Write the HTML to assign the header element to an appropriate ARIA landmark role.
4. Write the HTML to associate a web page with an external style sheet named myprint.css to configure a printout.
5. Write the @media rule to target a typical smartphone device and configure the nav element selector with width set to auto.
6. Write the CSS to configure a graphic named mysprite.gif to display as a background image on the left side of a hyperlink. Note that mysprite.gif contains two different images. Configure the image that is located 67 pixels from the top of the mysprite.gif graphic to display.
7. **Configure Printing for Hands-On Practice 7.4.** Configure special printing for the Lighthouse Island Bistro index.html file created in Hands-On Practice 7.4. This file is in the chapter7/7.4 folder in the student files. Modify the web page so that it is associated with an external style sheet called lighthouse.css instead of using embedded styles. Save and test your page. Create a new style sheet called myprint.css, which will prevent the navigation from displaying when the page is printed. Modify the index.html page to be associated with this file. Review the use of the media attribute on the link element. Save all files and test your page. Select File > Print > Preview to test the print styles.
8. **Extending Hands-On Practice 7.5.** In Hands-On Practice 7.5, you created the home page for the Door County Wildflowers website. This file is also available in the chapter7/7.5 folder in the student files. In this exercise, you will create two additional content pages

for the Door County Wildflowers site (spring.html and summer.html). Make sure that all CSS is placed in an external style sheet (mywildflower.css). Modify index.html to use this style sheet. The following is some content to include on the new pages:

Spring Page (spring.html):

- Use the trillium.jpg image (see the chapter7/starters folder in the student files).
- Trillium facts: 8–18 inches tall; perennial; native plant; grows in rich, moist deciduous woodlands; white flowers turn pink with age; fruit is a single red berry; protected flower species

Summer Page (summer.html):

- Use the yls.jpg image (see the chapter7/starters folder in the student files).
- Yellow Lady's Slipper facts: 4–24 inches tall; perennial; native plant; grows in wet, shaded deciduous woods, swamps, and bogs; an orchid; official flower of Door County

9. **Modify the Design of Hands-On Practice 7.5.** Locate the index.html page you created in Hands-On Practice 7.5. This file is in the chapter7/7.5 folder in the student files. Recall from Chapter 5 that a web page using fluid layout can have content in the center of the web page with empty margins on either side. Configure the style rules for index.html to display the page in this manner. Choose an appropriate background color to display in the margin area.
10. **Create a Mobile Design for Hands-On Practice 7.5.** Locate the index.html page you created in Hands-On Practice 7.5. This file is in the chapter7/7.5 folder in the student files. Modify the web page so that it links to an external style sheet (flowers.css) instead of using embedded styles. Modify the index.html page to be associated with this file. Configure the viewport meta tag and CSS media queries for a single-column smartphone mobile display. Save all files and test your page with both a desktop browser and a mobile device or emulator.
11. **Practice with Flexbox.** Locate the index.html page you created in Hands-On Practice 7.5. This file is in the chapter7/7.5 folder in the student files. Modify the HTML and CSS to use the Flexible Box Layout (flexbox) technique to configure the three columns on the web page. Save and test your page in browsers that support flexbox, such as Chrome or Opera.

Web Research

As you read about mobile web design best practices in this chapter, you may have noticed some overlap with techniques that provide for accessibility, such as alternate text and use of headings. Explore the Web Content Accessibility and Mobile Web document at http://www.w3.org/WAI/mobile. Explore related links that interest you. Write a one-page, double-spaced summary that describes areas of overlap and how web developers can support both accessibility and mobile devices.

Focus on Web Design

Take a few moments and visit the CSS Zen Garden at http://www.csszengarden.com. Explore the site and note the widely different designs. What thought processes and decisions are needed as a person creates a new design for this site? Visit Sheriar Designs (http://manisheriar.com/blog/anatomy-of-a-design-process) or Behind the Scenes of Garden Party (http://www.bobbyvandersluis.com/articles/garden-party/) for a behind-the-scenes look at how web developers have approached this task. Reflect on their stories and suggestions. Write a one-page, double-spaced essay that describes ideas about the design process that you'll be able to use as you begin to design websites for personal or professional use. Be sure to include the URL of the resources that you used.

WEBSITE CASE STUDY

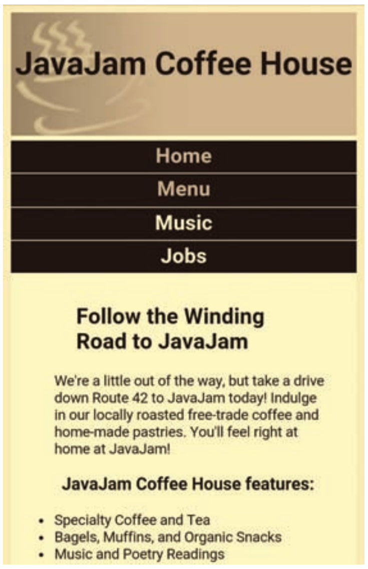
Styling for the Mobile Web

Each of the following case studies continues throughout most of the text. This chapter configures the website for display on mobile devices.

JavaJam Coffee House

See Chapter 2 for an introduction to the JavaJam Coffee House case study. Figure 2.30 shows a site map for the JavaJam website. In this case study, you will configure the website to display in mobile devices using the single-column layout shown in Figure 7.17. You will code media queries for mobile styles; modify the current desktop styles; and update the Home, Menu, and Music pages. Use the Chapter 6 JavaJam website as a starting point for this case study. When you are finished, the website will look the same in desktop browsers (see Figure 6.50). The mobile display should be similar to Figure 7.37 or 7.38. You have seven tasks in this case study:

1. Create a new folder for this JavaJam case study.
2. Modify the Home page to include a viewport meta tag.
3. Modify the Menu page to be consistent with the Home page.
4. Modify the Music page to be consistent with the Home page.
5. Modify the desktop styles in javajam.css.
6. Modify javajam.css and code a media query for tablet display.
7. Modify javajam.css and code a media query for typical smartphone display.



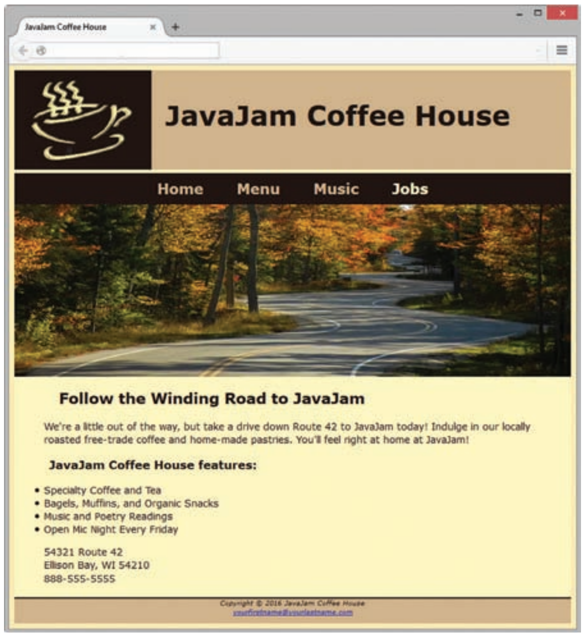
Follow the Winding Road to JavaJam

We're a little out of the way, but take a drive down Route 42 to JavaJam today! Indulge in our locally roasted free-trade coffee and home-made pastries. You'll feel right at home at JavaJam!

JavaJam Coffee House features:

- Specialty Coffee and Tea
- Bagels, Muffins, and Organic Snacks
- Music and Poetry Readings

Figure 7.37 The smartphone display



Follow the Winding Road to JavaJam

We're a little out of the way, but take a drive down Route 42 to JavaJam today! Indulge in our locally roasted free-trade coffee and home-made pastries. You'll feel right at home at JavaJam!

JavaJam Coffee House features:

- Specialty Coffee and Tea
- Bagels, Muffins, and Organic Snacks
- Music and Poetry Readings
- Open Mic Night Every Friday

54321 Route 42
Ellison Bay, WI 54210
888-555-5555

Copyright © 2016 JavaJam Coffee House
yourfirstname@yourlastname.com

Figure 7.38 A test of the new tablet (or high-resolution smartphone) display in a desktop browser

Hands-On Practice Case Study

Task 1: Create a Folder. Create a folder called javajam7. Copy all the files from your Chapter 6 javajam6 folder into the javajam7 folder. Copy the javajammini.jpg file from the chapter7/starters folder in the student files.

Task 2: Modify the Home Page. Open index.html in a text editor. Edit the code as follows:

1. Configure a viewport meta tag that configures the width to the `device-width` and sets the `initial-scale` to `1.0`.
2. The home page displays a phone number in the contact information area. Wouldn't it be handy if a person using a smartphone could click on the phone number to call the coffee house? You can make that happen by using `tel:` in a hyperlink. Configure a hyperlink assigned to an id named `mobile` that contains the phone number as shown below:

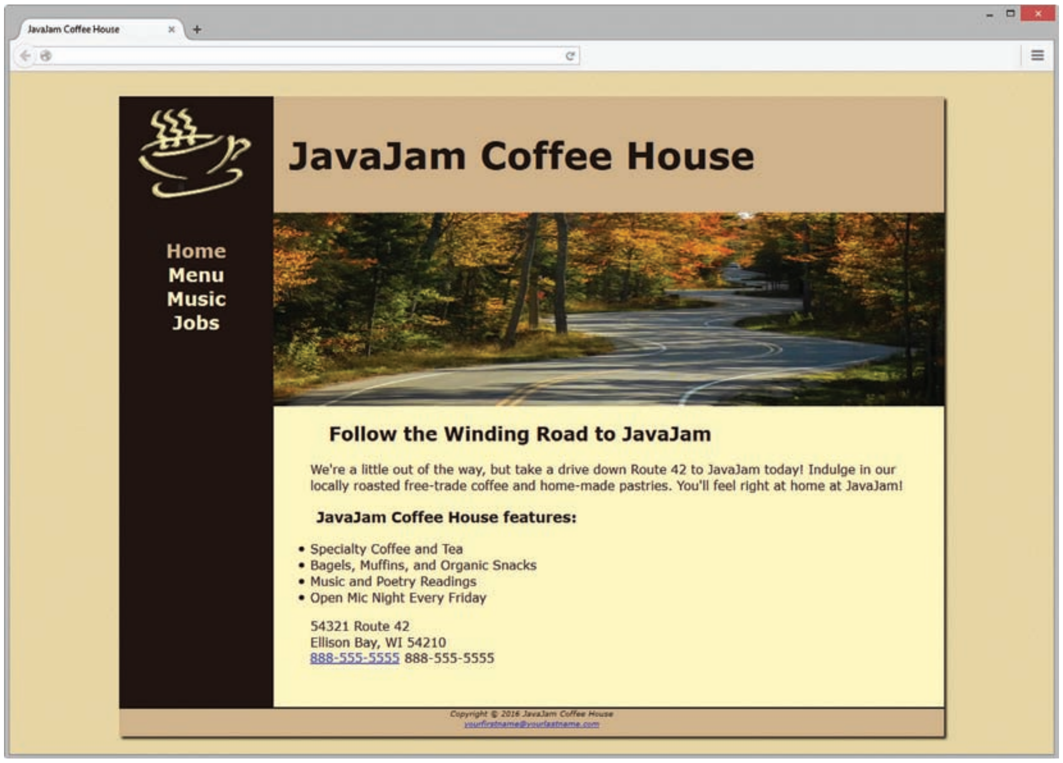
```
<a id="mobile" href="tel:888-555-5555">888-555-5555</a>
```

But wait a minute, a telephone link could confuse those visiting the site with a desktop browser. Code another phone number directly after the hyperlink. Code a span element assigned to an id named `desktop` around the phone number as shown below:

```
<span id="desktop">888-555-5555</span>
```

Don't worry about the two phone numbers that are now on the page. You'll configure CSS in Tasks 5 and 7 to show the appropriate phone number (with or without the telephone link) to your website visitors.

Save the index.html file. It should look similar to the web page shown in Figure 7.39 when displayed in a desktop browser. Remember that validating your HTML can help you find syntax errors. Test and correct this page before you continue.



JavaJam Coffee House

Follow the Winding Road to JavaJam

We're a little out of the way, but take a drive down Route 42 to JavaJam today! Indulge in our locally roasted free-trade coffee and home-made pastries. You'll feel right at home at JavaJam!

JavaJam Coffee House features:

- Specialty Coffee and Tea
- Bagels, Muffins, and Organic Snacks
- Music and Poetry Readings
- Open Mic Night Every Friday

54321 Route 42
Ellison Bay, WI 54210
888-555-5555 888-555-5555

Copyright © 2016 JavaJam Coffee House
yourfirstname@yourlastname.com

Figure 7.39 Temporary desktop browser display before new styles are configured

Task 3: Modify the Menu Page. Open menu.html in a text editor. Add the viewport meta tag in a manner consistent with the home page. Save and test your new menu.html page in a browser. Use the HTML validator to help you find syntax errors.

Task 4: Modify the Music Page. Open music.html in a text editor. Add the viewport meta tag in a manner consistent with the home page. Save and test your new music.html page in a browser. Use the HTML validator to help you find syntax errors.

Task 5: Modify the Desktop CSS. Open javajam.css in a text editor. Remember the telephone number hyperlink you created in Task 2? Configure the CSS for the phone number display as shown below:

```
#mobile { display: none; }
#desktop { display: inline; }
```

Save the javajam.css file. Use the CSS validator to help you find syntax errors.

Task 6: Configure the Tablet CSS. Open javajam.css in a text editor. Edit the style rules as follows:

1. Code a media query to select for typical tablet device viewport size, such as

```
@media only screen and (max-width: 1024px) {
}
```

2. Code the following new styles within the media query:
   - **a.** Configure a body element selector with margin set to 0. Set the `background-image` property to none.
   - **b.** Configure a `wrapper` id selector. Set the width to auto, `min-width` to 0, margin to 0, padding to 0, and `box-shadow` to none.
   - **c.** Configure the header element selector. Configure a 5px solid #FEF6C2 bottom border.
   - **d.** Configure the h1 element selector. Set top margin to 0, bottom margin to 1em, top padding to 1em, bottom padding to 1em, and 2.5em font size.
   - **e.** Configure the nav element selector. The mobile layout uses a single column. Set the float to none, auto width, 0 top padding, 10px margin, and 1.3em font size.
   - **f.** Configure the `nav li` selector. Set display to `inline-block`.
   - **g.** Configure the `nav a` selector. Set padding to 1em, width to 8em, font weight to bold, and `border-style` to none.
   - **h.** Configure the `nav ul` selector with 0 padding and margin.
   - **i.** Configure the `#heroroad`, `#heromugs`, and `#heroguitar` selectors. Configure 0 margin and 0 padding.
   - **j.** Configure the main element selector. Set padding to 0, margin to 0, and font size to 90%.

Save the javajam.css file. Use the CSS validator to help you find syntax errors.

Task 7: Configure the Smartphone CSS. Open javajam.css in a text editor. Note that any device with a screen max-width of 1024 pixels or less will apply the styles you coded in

Task 6. In this task you will code additional styles needed for smaller devices. Edit the style rules as follows:

1. Code a media query to select for typical smartphone device viewport size, such as

```
@media only screen and (max-width: 768px) {
}
```

2. Code the following new styles within the media query:
   - **a.** Configure the header element selector to display an image designed for small mobile devices. Set the background image to javajammini.jpg. Set the height to 128px.
   - **b.** Configure the h1 element selector. Set 2em font size, centered text, and 0 left padding.
   - **c.** Configure the nav element selector. Set the margin to 0.
   - **d.** Configure the anchor tags in the navigation area. Code a style rule for the `nav a` selector. Set the display to `block`, padding to 0.2em, and width to auto. Also configure a 1 pixel bottom border (use #FEF6C2 for the border color).
   - **e.** Configure the `nav li` selector. Set the display to block.
   - **f.** Configure the main element selector. Set top padding to 1px.
   - **g.** Configure the h2 element selector. Set 0.5em top padding, 0 right padding, 0 bottom padding, and 0.5em left padding. Set the right margin to 0.5em.
   - **h.** Configure the `details` class selector. Set left and right padding to 0.
   - **i.** Configure the `#heroroad, #heromugs,` and `#heroguitar` selectors. Set the background image to none. Set the height to auto.
   - **j.** Configure the `floatleft` class selector. Set left and right padding to 0.5em.
   - **k.** Remember the telephone number hyperlink you created in Task 2? Configure the CSS for the phone number display as shown below:

```
#mobile { display: inline; }
#desktop { display: none; }
```

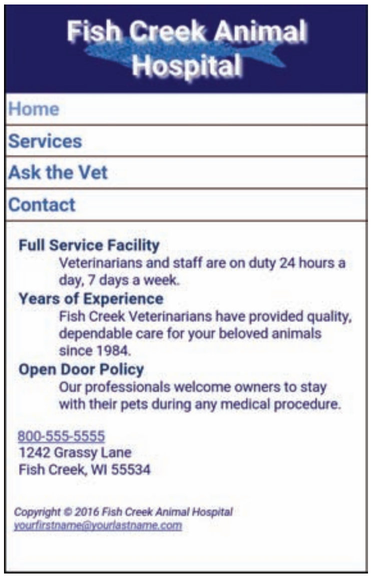
Save the javajam.css file. Use the CSS validator to help you find syntax errors.

Display your pages in a desktop browser. The pages should look the same as they did before you started this case study (see Figure 6.50). Next, test the mobile display. Display your page and reduce the width of the browser. Your mobile display should be similar to Figure 7.37 or 7.38. Select the hyperlinks to view the Menu and Music pages. They should be similar to the home page. JavaJam is mobile!

Fish Creek Animal Hospital

See Chapter 2 for an introduction to the Fish Creek Animal Hospital case study. Figure 2.34 shows a site map for the Fish Creek website. In this case study, you configure the website to display in mobile devices using the single-column layout shown in Figure 7.17. You will code media queries for mobile styles; modify the current desktop styles; and update the Home, Services, and Ask the Vet pages. Use the Chapter 6 Fish Creek website as a starting point for this case study. When you are finished, the website will look the same in desktop browsers (see Figure 6.54). The mobile display should be similar to Figure 7.40 or 7.41. You have seven tasks in this case study:

1. Create a new folder for this Fish Creek case study.
2. Modify the Home page to include a viewport meta tag and an updated header area.
3. Modify the Services page to be consistent with the Home page.
4. Modify the Ask the Vet page to be consistent with the Home page.
5. Modify the desktop styles in fishcreek.css.
6. Modify fishcreek.css and code a media query for tablet device display.
7. Modify fishcreek.css and code a media query for smartphone device display.



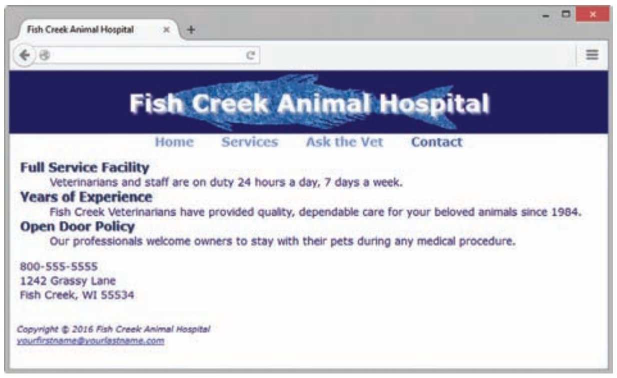
Fish Creek Animal Hospital

Full Service Facility
Veterinarians and staff are on duty 24 hours a day, 7 days a week.
Years of Experience
Fish Creek Veterinarians have provided quality, dependable care for your beloved animals since 1984.
Open Door Policy
Our professionals welcome owners to stay with their pets during any medical procedure.

800-555-5555
1242 Grassy Lane
Fish Creek, WI 55534

Copyright © 2016 Fish Creek Animal Hospital
yourfirstname@yourlastname.com

Figure 7.40 The smartphone display



Fish Creek Animal Hospital

Full Service Facility
Veterinarians and staff are on duty 24 hours a day, 7 days a week.
Years of Experience
Fish Creek Veterinarians have provided quality, dependable care for your beloved animals since 1984.
Open Door Policy
Our professionals welcome owners to stay with their pets during any medical procedure.

800-555-5555
1242 Grassy Lane
Fish Creek, WI 55534

Copyright © 2016 Fish Creek Animal Hospital
yourfirstname@yourlastname.com

Figure 7.41 A test of the new tablet (or high-resolution smartphone) display in a desktop browser

Hands-On Practice Case Study

Task 1: Create a Folder. Create a folder called fishcreek7. Copy all of the files from your Chapter 6 fishcreek6 folder into the fishcreek7 folder. Copy the lilfish.gif file from the chapter7/starters folder in the student files.

Task 2: Modify the Home Page. Open index.html in a text editor. Edit the code as follows:

1. Configure a viewport meta tag that configures the width to the `device-width` and sets the `initial-scale` to `1.0`.
2. The home page displays a phone number in the contact information area. Wouldn't it be handy if a person using a smartphone could click on the phone number to call the animal hospital? You can make that happen by using `tel:` in a hyperlink. Configure a hyperlink assigned to an id named `mobile` that contains the phone number as shown below:

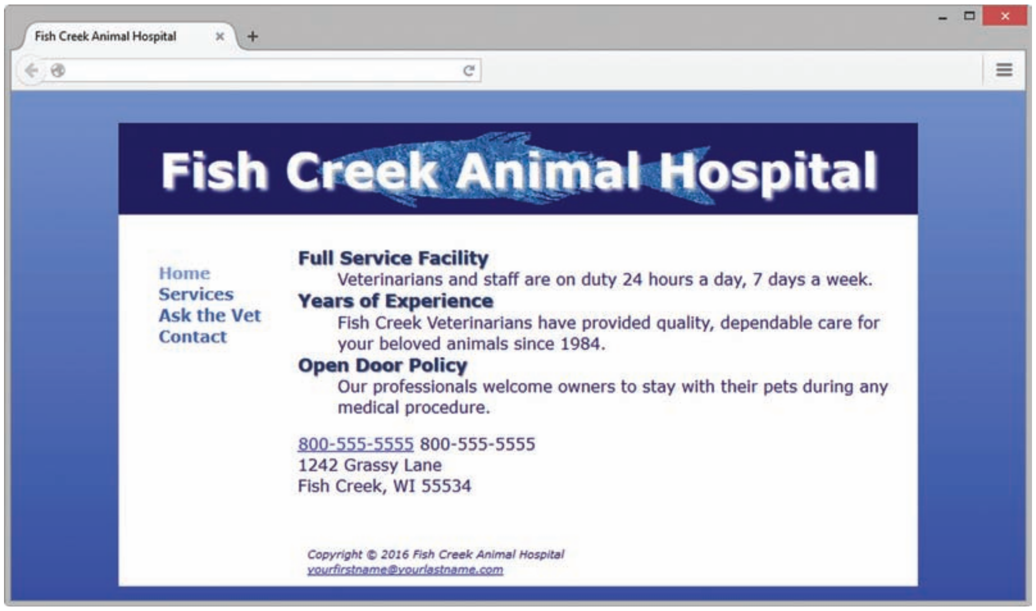
```
<a id="mobile" href="tel:800-555-5555">800-555-5555</a>
```

But wait a minute, a telephone link could confuse those visiting the site with a desktop browser. Code another phone number directly after the hyperlink. Code a span element assigned to an id named `desktop` around the phone number as shown below:

```
<span id="desktop">800-555-5555</span>
```

Don't worry about the two phone numbers that are now on the page. You'll configure CSS in Tasks 5 and 7 to show the appropriate phone number (with or without the telephone link) to your website visitors.

Save the index.html file. It should look similar to the web page shown in Figure 7.42 when displayed in a desktop browser. Remember that validating your HTML can help you find syntax errors. Test and correct this page before you continue.



Full Service Facility
Veterinarians and staff are on duty 24 hours a day, 7 days a week.
Years of Experience
Fish Creek Veterinarians have provided quality, dependable care for your beloved animals since 1984.
Open Door Policy
Our professionals welcome owners to stay with their pets during any medical procedure.

800-555-5555 800-555-5555
1242 Grassy Lane
Fish Creek, WI 55534

Copyright © 2016 Fish Creek Animal Hospital
yourfirstname@yourlastname.com

Figure 7.42 Temporary desktop browser display before new styles are configured

Task 3: Modify the Services Page. Open services.html in a text editor. Add the viewport meta tag in a manner consistent with the home page. Save and test your new services.html page in a browser. Use the HTML validator to help you find syntax errors.

Task 4: Modify the Ask the Vet Page. Open askvet.html in a text editor. Add the viewport meta tag in a manner consistent with the home page. Save and test your new askvet.html page in a browser. Use the HTML validator to help you find syntax errors.

Task 5: Modify the Desktop CSS. Open fishcreek.css in a text editor. Remember the telephone number hyperlink you created in Task 2? Configure the CSS for the phone number display as shown below:

```
#mobile { display: none; }
#desktop { display: inline; }
```

Save the fishcreek.css file. Use the CSS validator to help you find syntax errors.

Task 6: Configure the Tablet CSS. Open fishcreek.css in a text editor. Edit the style rules as follows:

1. Code a media query to select for typical tablet device viewport size, such as

```
@media only screen and (max-width: 1024px) {
}
```

2. Code the following new styles within the media query:
   - **a.** Configure a body element selector with margin and padding set to 0. Set the background color to white and the `background-image` property to none.
   - **b.** Configure a `wrapper` id selector. Set the width to auto, `min-width` to 0 and margin to 0.
   - **c.** Configure the h1 element selector. Set margin to 0, font size to 1.8em, and line height to 200%.
   - **d.** Configure the nav element selector. The mobile layout uses a single column. Set the float to none, and width to auto.
   - **e.** Configure the `nav li` selector. Set display to `inline-block`.
   - **f.** Configure the `nav a` selector. Set padding to 1em and font size to 1.2em.
   - **g.** Configure the `nav ul` element selector with centered text, 0 padding, and 0 margin.
   - **h.** Configure the main element selector. Set font size to 90%, margin to 0, and left padding to 2em.
   - **i.** Configure the footer element selector with 0 margin.

Save the fishcreek.css file. Use the CSS validator to help you find syntax errors.

Task 7: Configure the Smartphone CSS. Open fishcreek.css in a text editor. Note that any device with a screen max-width of 1024 pixels or less will apply the styles you coded in Task 6. In this task you will code additional styles needed for smaller devices. Edit the style rules as follows:

1. Code a media query to select for typical smartphone device viewport size, such as

```
@media only screen and (max-width: 768px) {
}
```

2. Code the following new styles within the media query:
   - **a.** Configure a header element selector. The bigfish.gif image is too wide to display well on a mobile device. Your design strategy is to configure a smaller image (lilfish.gif) as the header background for mobile display. Write style declarations to set the small fish logo (lilfish.gif) as a background image that does not repeat and has center `background-position`.
   - **b.** Configure the h1 element selector. Set font-size to 1.5em and `line-height` to 120%.
   - **c.** Configure the anchor tags in the navigation area. Code a style rule for the `nav a` selector. Set display to `block`, padding to 0.2em, and font size to 1.3em. Also configure a 1 pixel bottom border (use #330000 for the border color).
   - **d.** Configure the `nav li` selector. Set display to `block`.
   - **e.** Configure the `nav ul` selector. Set `text-align` to left.
   - **f.** Configure a main element selector. Set 1em left padding.
   - **g.** Configure a `category` class selector. While the `text-shadow` property can work well in the logo header area, it can make content text difficult to read on a mobile device. Set `text-shadow` to `none`.
   - **h.** Remember the telephone number hyperlink you created in Task 2? Configure the CSS for the phone number display as shown below:

```
#mobile { display: inline; }
#desktop { display: none; }
```

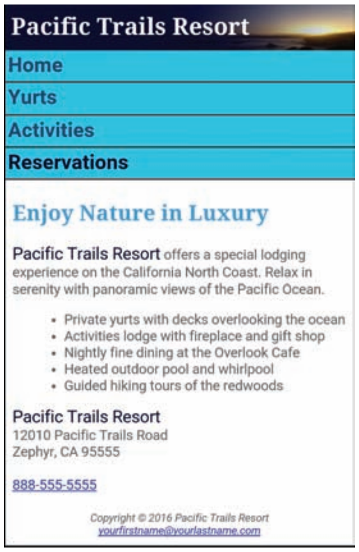
Save the fishcreek.css file. Use the CSS validator to help you find syntax errors.

Display your pages in a desktop browser. The home page should look similar to Figure 6.54. Next, test the mobile display. Display your page and reduce the width of the browser. Your mobile display should be similar to Figure 7.40 or 7.41. Select the hyperlinks to view the Services and Ask the Vet pages. They should be similar to the home page. Fish Creek is mobile!

Pacific Trails Resort

See Chapter 2 for an introduction to the Pacific Trails Resort case study. Figure 2.38 shows a site map for the Pacific Trails website. In this case study, you will configure the website to display on mobile devices using the single-column layout shown in Figure 7.17. You will code media queries for mobile styles; modify the current desktop styles; and update the Home, Yurts, and Activities pages. Use the Chapter 6 Pacific Trails website as a starting point for this case study. When you have finished, the website will look the same in desktop browsers (see Figure 6.56). The mobile display should be similar to Figure 7.43 or 7.44. You have seven tasks in this case study:

1. Create a new folder for this Pacific Trails case study.
2. Modify the Home page to include a viewport meta tag.
3. Modify the Yurts page to be consistent with the Home page.
4. Modify the Activities page to be consistent with the Home page.
5. Modify the desktop styles in pacific.css.
6. Modify pacific.css and code a media query for tablet device display.
7. Modify pacific.css and code a media query for smartphone device display.



Pacific Trails Resort

Home
Yurts
Activities
Reservations

Enjoy Nature in Luxury

Pacific Trails Resort offers a special lodging experience on the California North Coast. Relax in serenity with panoramic views of the Pacific Ocean.

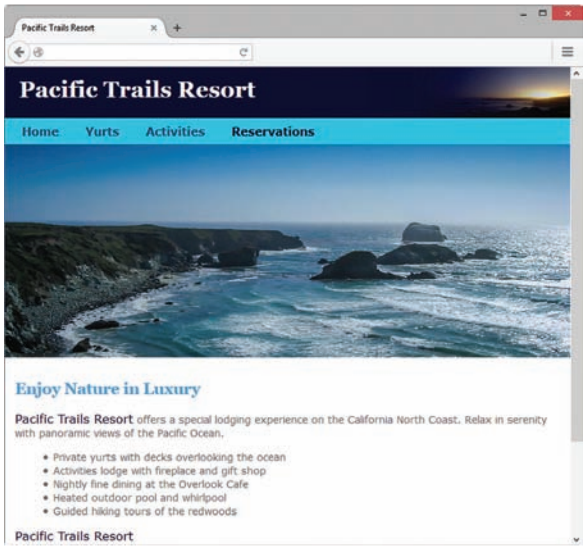
- Private yurts with decks overlooking the ocean
- Activities lodge with fireplace and gift shop
- Nightly fine dining at the Overlook Cafe
- Heated outdoor pool and whirlpool
- Guided hiking tours of the redwoods

Pacific Trails Resort
12010 Pacific Trails Road
Zephyr, CA 95555

888-555-5555

Copyright © 2016 Pacific Trails Resort
yourfirstname@yourlastname.com

Figure 7.43 Smartphone display



Pacific Trails Resort

Home Yurts Activities Reservations

Enjoy Nature in Luxury

Pacific Trails Resort offers a special lodging experience on the California North Coast. Relax in serenity with panoramic views of the Pacific Ocean.

- Private yurts with decks overlooking the ocean
- Activities lodge with fireplace and gift shop
- Nightly fine dining at the Overlook Cafe
- Heated outdoor pool and whirlpool
- Guided hiking tours of the redwoods

Pacific Trails Resort

Figure 7.44 A test of the new tablet (or high-resolution smartphone) display in a desktop browser

Hands-On Practice Case Study

Task 1: Create a Folder. Create a folder called pacific7. Copy all of the files from your Chapter 6 pacific6 folder into the pacific7 folder.

Task 2: Modify the Home Page. Open index.html in a text editor. Edit the code as follows:

1. Configure a viewport meta tag that configures the width to the `device-width` and sets the `initial-scale` to `1.0`.
2. The home page displays a phone number in the contact information area. Wouldn't it be handy if a person using a smartphone could click on the phone number to call the resort? You can make that happen by using `tel:` in a hyperlink. Configure a hyperlink assigned to an id named `mobile` that contains the phone number as shown below:

   ```
   <a id="mobile" href="tel:888-555-5555">888-555-5555</a>
   ```

   But wait a minute, a telephone link could confuse those visiting the site with a desktop browser. Code another phone number directly after the hyperlink. Code a span element assigned to an id named `desktop` around the phone number as shown here:

   ```
   <span id="desktop">888-555-5555</span>
   ```

   Don't worry about the two phone numbers that are now on the page. You'll configure CSS in Tasks 5 and 7 to show the appropriate phone number (with or without the telephone link) to your website visitors.

Save the index.html file. It should look similar to the web page shown in Figure 6.56 (except for temporarily displaying two phone numbers) when displayed in a desktop browser. Remember that validating your HTML can help you find syntax errors. Test and correct this page before you continue.

Task 3: Modify the Yurts Page. Open yurts.html in a text editor. Add the viewport meta tag in a manner consistent with the home page. Save and test your new yurts.html page in a browser. Use the HTML validator to help you find syntax errors.

Task 4: Modify the Activities Page. Open activities.html in a text editor. Add the viewport meta tag in a manner consistent with the home page. Save and test your new activities.html page in a browser. Use the HTML validator to help you find syntax errors.

Task 5: Modify the Desktop CSS. Open pacific.css in a text editor. Remember the telephone number hyperlink you created in Task 2? Configure the CSS for the phone number display as shown below:

```
#mobile { display: none; }
#desktop { display: inline; }
```

Save the pacific.css file. Use the CSS validator to help you find syntax errors.

Task 6: Configure the Tablet CSS. Open pacific.css in a text editor. Edit the style rules as follows:

1. Code a media query to select for typical tablet device viewport size, such as

   ```
   @media only screen and (max-width: 1024px) {
   }
   ```

2. Code the following new styles within the media query:
   a. Configure a body element selector with margin and padding set to 0. Set the `background-image` property to none.
   b. Configure a `wrapper` id selector. Set the width to auto, `min-width` to 0, margin to 0, and `box-shadow` to none.
   c. Configure the h1 element selector. Set margin to 0.
   d. Configure the nav element selector. The mobile layout uses a single column. Set the float to none, width to auto, and padding to 0.5em.
   e. Configure the `nav li` selector. Set display to `inline-block`.
   f. Configure the `nav a` selector. Set padding to 1em.
   g. Configure the main element selector. Set padding to 1em, left margin to 0, and 90% font size.
   h. Configure the footer element selector with 0 margin.
   i. Configure the `#homehero, #yurthero, and #trailhero` selectors. Set the left margin to 0.

Save the pacific.css file. Use the CSS validator to help you find syntax errors.

Task 7: Configure the Smartphone CSS. Create the mobile style sheet based on the desktop style sheet. Open pacific.css in a text editor. Note that any device with a screen max-width of 1024 pixels or less will apply the styles you coded in Task 6. In this task you will code additional styles needed for smaller devices. Edit the style rules as follows:

1. Code a media query to select for typical smartphone device viewport size, such as

```
@media only all and (max-width: 768px) {
}
```

2. Code the following new styles within the media query:
   a. Configure the h1 element selector. Set height to 100%, font size to 1.5em, and left padding to 0.3em.
   b. Configure the nav element selector. Set padding to 0.
   c. Configure the anchor tags in the navigation area. Code a style rule for the `nav a` selector. Set display to `block`, padding to 0.2em, and font size to 1.3em. Also configure a 1 pixel bottom border (use #330000 for the border color).
   d. Configure the `nav ul` selector and set margin and padding to 0.
   e. Configure the `nav li` selector with block display. Set margin and padding to 0.
   f. Configure the main element selector. Set 0.1em top padding, 0.6em right padding, 0.1em bottom padding, and 0.4em left padding.
   g. Configure the `#homehero, #yurthero, and #trailhero` selectors to not display. Set display to `none`.
   h. Configure the footer element selector. Set padding to 0.
   i. Remember the telephone number hyperlink you created in Task 2? Configure the CSS for the phone number display as shown below:

```
#mobile { display: inline; }
#desktop { display: none; }
```

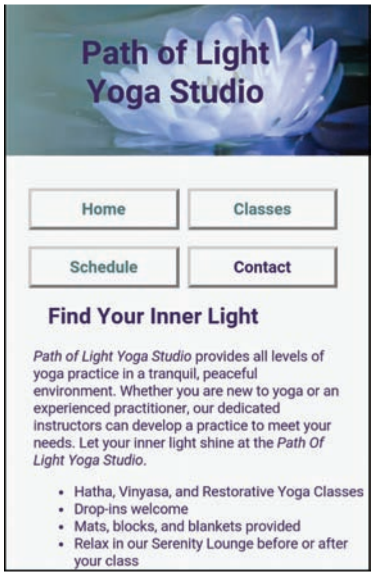
Save the pacific.css file. Use the CSS validator to help you find syntax errors.

Display your pages in a desktop browser. The pages should look the same as they did before you started this case study (see Figure 6.56). Next, test the mobile display. Display your page and reduce the width of the browser. Your mobile display should be similar to Figure 7.43 or 7.44. Select the hyperlinks to view the Yurts and Activities pages. They should be similar to the home page. Pacific Trails is mobile!

Path of Light Yoga Studio

See Chapter 2 for an introduction to the Path of Light Yoga Studio case study. Figure 2.42 shows a site map for the Path of Light Yoga Studio website. In this case study, you will configure the website to display on mobile devices using the single-column layout shown in Figure 7.17. You will code media queries for mobile styles; modify the current desktop styles, and update the Home, Classes, and Schedule pages. Use the Chapter 6 Path of Light Yoga Studio website as a starting point for this case study. When you are finished, the website will look the same (see Figure 6.58) in desktop browsers. The mobile display should be similar to Figure 7.45 or 7.46. You have seven tasks in this case study:

1. Create a new folder for this Path of Light Yoga Studio case study.
2. Modify the Home page to include a viewport meta tag and an updated header area.
3. Modify the Classes page to be consistent with the Home page.
4. Modify the Schedule page to be consistent with the Home page.
5. Modify the desktop styles in yoga.css.
6. Modify yoga.css and code a media query for tablet device display.
7. Modify yoga.css and code a media query for smartphone device display.

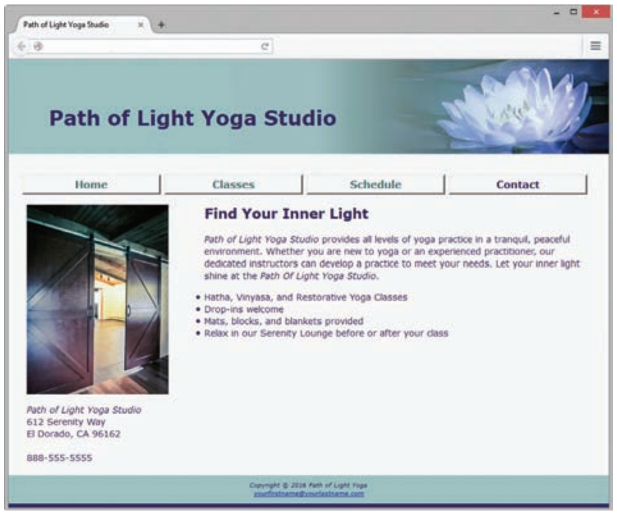


Find Your Inner Light

Path of Light Yoga Studio provides all levels of yoga practice in a tranquil, peaceful environment. Whether you are new to yoga or an experienced practitioner, our dedicated instructors can develop a practice to meet your needs. Let your inner light shine at the *Path Of Light Yoga Studio*.

- Hatha, Vinyasa, and Restorative Yoga Classes
- Drop-ins welcome
- Mats, blocks, and blankets provided
- Relax in our Serenity Lounge before or after your class

Figure 7.45 The smartphone display



Find Your Inner Light

Path of Light Yoga Studio provides all levels of yoga practice in a tranquil, peaceful environment. Whether you are new to yoga or an experienced practitioner, our dedicated instructors can develop a practice to meet your needs. Let your inner light shine at the *Path Of Light Yoga Studio*.

- Hatha, Vinyasa, and Restorative Yoga Classes
- Drop-ins welcome
- Mats, blocks, and blankets provided
- Relax in our Serenity Lounge before or after your class

Path of Light Yoga Studio
612 Serenity Way
El Dorado, CA 96162

888-555-5555

Copyright © 2016 Path of Light Yoga
yourfirstname@yourlastname.com

Figure 7.46 A test of the new tablet (or high-resolution smartphone) display in a desktop browser

Hands-On Practice Case Study

Task 1: Create a Folder. Create a folder called yoga7. Copy all of the files from your Chapter 6 yoga6 folder into the yoga7 folder.

Task 2: Modify the Home Page. Open index.html in a text editor. Edit the code as follows:

1. Configure a viewport meta tag that configures the width to the `device-width` and sets the `initial-scale` to `1.0`.
2. The home page displays a phone number in the contact information area. Wouldn't it be handy if a person using a smartphone could click on the phone number to call the company? You can make that happen by using `tel:` in a hyperlink. Configure a hyperlink assigned to an id named `mobile` that contains the phone number as shown below:

   ```
   <a id="mobile" href="tel:888-555-5555">888-555-5555</a>
   ```

   But wait a minute, a telephone link could confuse those visiting the site with a desktop browser. Code another phone number directly after the hyperlink. Code a span element assigned to an id named `desktop` around the phone number as shown below:

   ```
   <span id="desktop">888-555-5555</span>
   ```

   Don't worry about the two phone numbers that are now on the page. You'll configure CSS in Tasks 5 and 7 to show the appropriate phone number (with or without the telephone link) to your website visitors.

Save the index.html file. It should look similar to the web page shown in Figure 6.58 (except for temporarily displaying two phone numbers) when displayed in a desktop browser. Remember that validating your HTML can help you find syntax errors. Test and correct this page before you continue.

Task 3: Modify the Classes Page. Open classes.html in a text editor. Add the viewport meta tag in a manner consistent with the home page. Remove the height and width attributes from the img tag. Save and test your new classes.html page in a browser. Use the HTML validator to help you find syntax errors.

Task 4: Modify the Schedule Page. Open schedule.html in a text editor. Add the viewport meta tag in a manner consistent with the home page. Remove the height and width from the img tag. Save and test your new schedule.html page in a browser. Use the HTML validator to help you find syntax errors.

Task 5: Modify the Desktop CSS. Open yoga.css in a text editor. Remember the telephone number hyperlink you created in Task 2? Configure the CSS for the phone number display as shown below:

```
#mobile { display: none; }
#desktop { display: inline; }
```

Save the yoga.css file. Use the CSS validator to help you find syntax errors.

Task 6: Configure the Tablet CSS. Open yoga.css in a text editor. Edit the style rules as follows:

1. Code a media query to select for typical tablet device viewport size. such as

   ```
   @media only screen and (max-width: 1024px) {
   }
   ```

2. Code the following new styles within the media query:
   a. Configure a body element selector with margin and padding set to 0.
   b. Configure a `wrapper` id selector. Set the width to 100%, `min-width` to 0, margin to 0, and padding to 0.
   c. Configure the header element selector. Set top padding to 1px.
   d. Configure the nav element selector. The mobile layout uses a single column. Set the float to none, width to auto, and left padding to 2em.
   e. Configure the `nav a` selector. Set padding to 0.2em, left margin to 0.3em, float to left, and width to 23%.
   f. Configure the main element selector. Set top and bottom padding to 2.5em, left and right padding to 1em, margin to 0, 90% font size, and clear all floats.
   g. Configure the `#hero img` selector. Code a style declarations to set the width to 100% and the height to auto.
   h. Configure the h2, h3, p, and dl element selectors. Set the left and right padding to 2em.
   i. Configure the `main ul` selector. Set left margin to 2em.
   j. Configure the `floatleft` class with 2em left margin and 1em bottom margin.
   k. Configure the `clear` class with 2em left padding.

Save the yoga.css file. Use the CSS validator to help you find syntax errors.

Task 7: Configure the Mobile CSS. Open yoga.css in a text editor. Note that any device with a screen max-width of 1024 pixels or less will apply the styles you coded in Task 6. In this task you will code additional styles needed for smaller devices. Edit the style rules as follows:

1. Code a media query to select for typical smartphone device viewport size, such as

```
@media only all and (max-width: 768px) {
}
```

2. Code the following new styles within the media query:
   a. Configure an h1 element selector. Set the font size to 2em, top padding to 0.25em, left padding to 1.5em and width to 85%. Also configure centered text.
   b. Configure the anchor tags in the navigation area. Configure a style rule for the `nav a` selector. Set padding to 0.5em, width to 45%, float to left, minimum width to 6em, and left margin to 0.5em.
   c. Configure the main element selector. Set top padding to 0.
   d. Configure the `floatleft` class selector. Set the float property to `none`. Set display to none.
   e. Configure the `#hero` selector. Set display to none.
   f. Configure the footer element selector. Set padding to 0.5em and margin to 0.
   g. Remember the telephone number hyperlink you created in Task 2? Configure the CSS for the phone number display as shown below:

```
#mobile { display: inline; }
#desktop { display: none; }
```

Save the yoga.css file. Use the CSS validator to help you find syntax errors.

Display your pages in a desktop browser. The pages should look the same as they did before you started this case study (see Figure 6.58). Next, test the mobile display. Display your page and reduce the width of the browser. Your mobile display should be similar to Figure 7.45 or 7.46. Select the hyperlinks to view the Classes and Schedule pages. They should be similar to the home page. Path of Light Yoga Studio is mobile!

Web Project

See Chapters 5 and 6 for an introduction to the Web Project case study. In this case study, you configure the website to display in mobile devices using the single-column layout shown in Figure 7.17. You will configure a media query for mobile device display and modify the web pages as needed.

Hands-On Practice Case Study

1. Modify the style sheet and configure a media query with styles for mobile device display.
2. Modify each web page and configure a viewport meta tag.
3. Display your pages in a desktop browser. The pages should look the same as they did before you started this case study. Next, display your pages in a mobile device or emulator. Your pages should be optimized for mobile display.