**296** *Section 1 A crash course in HTML and CSS*

**How to use CSS3 media queries**

In the topics that follow, you'll learn how to use media queries to control the appearance of a page in various screen sizes. You'll also learn how to build responsive menus using a jQuery plugin called SlickNay. But first, you'll learn how to control the mobile viewport so it works in conjunction with media queries.

**How to control the mobile viewport**

When you develop a website that uses Responsive Web Design to provide for various screen sizes, you'll want to be sure it configures the *viewport* appro­priately for mobile devices. To do that, you can use the meta element that's presented in figure 8-8.

To start, you should know that the viewport on a mobile browser works differently from the viewport on a desktop browser. On a desktop browser, the viewport is the visible area of the web page. The user can change the size of the viewport by changing the size of the browser window.

In contrast, the viewport on a mobile device can be larger or smaller than the visible area. In this figure, for example, you can see that the first web page is displayed so the entire width of the page is visible. Some mobile browsers reduce a page like this automatically if no meta element is included. In contrast, other mobile browsers don't reduce the page at all, so it extends beyond the visible area of the screen as shown in the second web page in this figure.

When you use media queries, you want to be sure that the web page isn't reduced, since the media queries will adjust the appearance of the page based on the screen size. In other words, you want the page to look like the second one in this figure. To do that, you add a meta element like the one shown in this figure.

Here, the name attribute is set to "viewport" to indicate that the element applies to the viewport. Then, the content attribute specifies the properties for the viewport. The first property, width, indicates that the width of the viewport should be set to the width of the screen in CSS pixels at a zoom factor of 100%. This makes sense if you realize that a CSS pixel isn't always the same as a device pixel. On mobile devices that have high resolutions, for example, one CSS pixel can be equal to two or more device pixels. In addition, if a user zooms into a page, the number of CSS pixels per device pixel increases. At a zoom factor of 200%, for example, each CSS pixel is two device pixels wide and high.

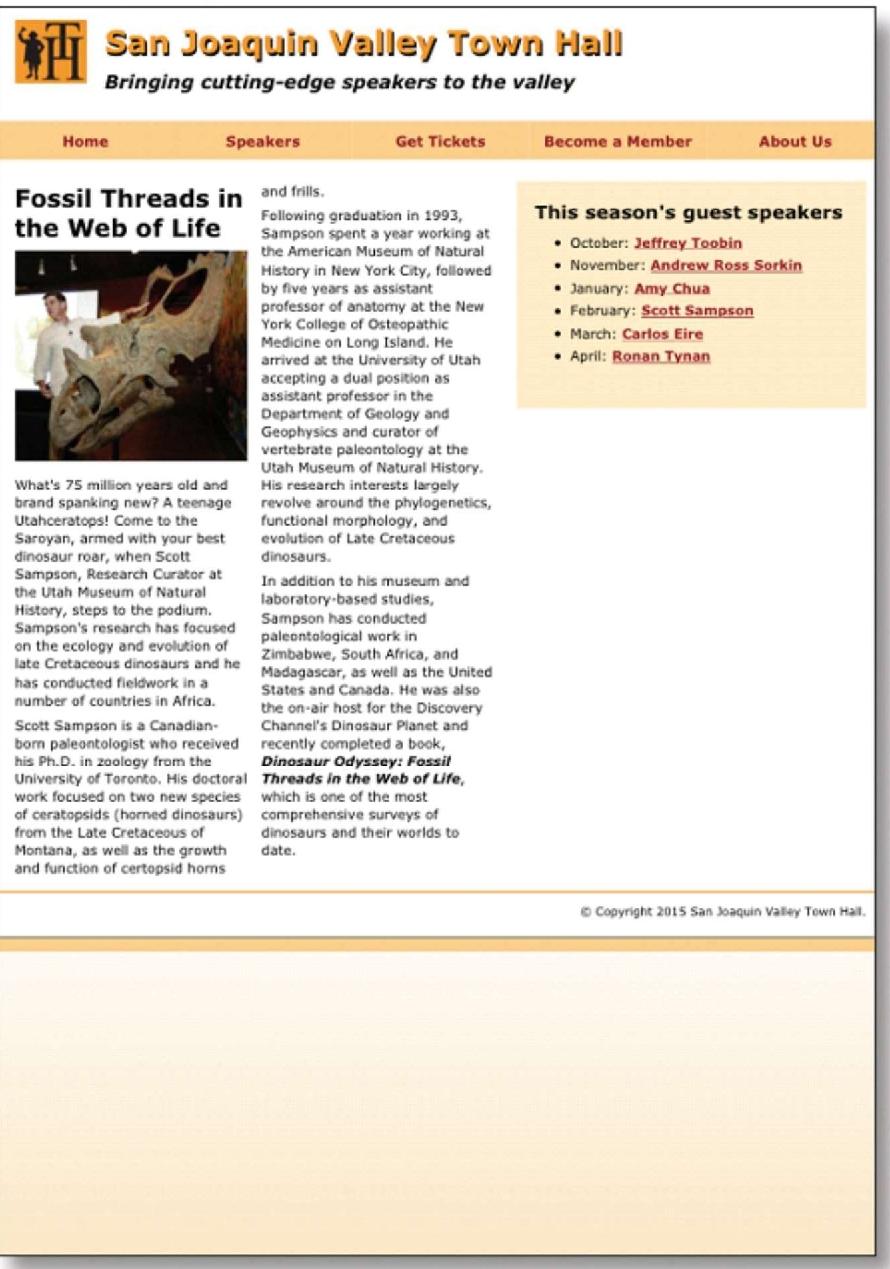
The second property, initial-scale, determines the initial zoom factor, or *scale,* for the viewport. In this case, this scale is set to 1, which represents the default width for the viewport. This is what keeps the browser from scaling the page automatically.

In addition to the width and initial-scale properties, you may want to set some of the other properties that are presented in this figure. Specifically, when you use RWD, you may want to set the user-scalable property to "no" so the user can't zoom in or out of the display. Or, if you want to let the user zoom in or

out, you can set the minimum-scale and maximum-scale properties to limit how much the user can zoom.

*Chapter 8 How to use Responsive Web Design* **297**

**A web page on a mobile device without and with a meta viewport element**



Moms

Speekere

Betrickets

Become a member

cljg Szin luzitjuin Hain

*Bringing cutting-edge speakers to the valley*

Research 0:aser

an m of raMeral s.

emes ftew to ...MM.' Samos. Ms conducted
  
Sampim's niW. has focused

17:e fr.b■Vr="[IndlNeed

was1115.

1i

ecreere

***Tar\* :11r. VI:of We,***

MN,

This season's guest speakers

,McPaCsoloefim

* Apra: Ronsm\_Teneo



***Slain JOIMPElL***

***Bringing cutting-***

**Home**

**Speak(**

**Fossil Threads in the Web of Life**

anc

Fol Sar the His by pro Yor Me

a rr
  
a cc
  
ass

**Content properties for viewport metadata**

Figure 8-8 How to control the mobile viewport

**A meta element that sets viewport properties**

**Property Description**

The logical width of the viewport specified in pixels. You can also use the device-width keyword to indicate that the viewport should be the width of the screen in CSS pixels at a scale of 100%.

The logical height of the viewport specified in pixels. You can also use the device-height keyword to indicate that the viewport should be the height of the screen in CSS pixels at a scale of 100%.

A number that indicates the initial zoom factor that's used to display the page. A number that indicates the minimum zoom factor for the page.

A number that indicates the maximum zoom factor for the page.

Indicates whether the user can zoom in and out of the viewport. Possible values are yes and no.

**width**

**height**

**initial-scale minimum-scale maximum-scale user-scalable**

**<meta name="viewport" content="width=device-width, initial-scale=l">**

**Description**

* The *viewport* on a mobile device determines the content that's displayed on the
    
  screen. It can be larger or smaller than the actual visible area of the screen.
* You use a meta element to control the viewport settings for a device. You add this element within the head element of a page.
* When you use media queries, you should set the width property of the content attribute to "device-width", and you should set the initial-scale property to 1. You can also prevent or limit scaling with the user-scalable, minimum-scale, and maximum-scale properties.

**298** *Section 1 A crash course in HTML and CSS*

**How to code media queries**

*Media queries* are a CSS3 feature that allows you to write conditional expressions directly within your CSS code. These conditional expressions can be used to query various properties of a device, such as the screen size. Figure 8-9 shows how to code media queries for a responsive design.

At the top of this figure, you can see the basic syntax of a media query. It starts with an *@media selector,* followed by a media type. When you're devel­oping a responsive design, you'll set the media type to "screen" so the media query will only be used if the page is displayed on a screen. You can also precede the media type with the only keyword. Then, if a browser doesn't support media queries, it will only check the media type.

After the media type, you code one or more conditional expressions, where each expression specifies the value of a property. The table in this figure lists some of the common properties for the screen media type. Note that all of these properties can also be prefixed with min- or max-, as shown in the media queries in this figure.

In the first media query, for example, the conditional expression uses the max-width property to check that the width of the viewport is 767 pixels or less. The second media query in this figure includes two conditional expressions. The first one checks that the width of the viewport is 480 pixels or more, and the second one checks that the width of the viewport is 767 pixels or less.

Within each media query, you code the CSS that adjusts the appearance of the web page so it's appropriate for the screen size that's specified by the query. For example, I could use a media query to change the font sizes that are used on a page or to change the layout of a page. You'll see the CSS for a web page that uses media queries like this later in this chapter.

This figure also lists all the desktop browsers that support media queries. This includes all the current versions of all the major browsers. Because of that, you can use any of these browsers to test that the media queries used by a web page work correctly. To do that, you just change the width of the browser window to the width specified by a media query. Then, the styles defined by that media query will be applied. Even if you do that, though, keep in mind that you'll still want to thoroughly test your media queries by using one of the techniques that were presented earlier in this chapter.

Before I go on, you should realize that all the desktop browsers except
  
for Internet Explorer are updated automatically when a new version becomes

available. Because of that, you don't need to check what version you have before testing media queries. If you want to use Internet Explorer for testing, though, you need to be sure that you have at least version 9.

*Chapter 8 How to use Responsive Web Design* **299**

Figure 8-9 How to code media queries

**The basic syntax of a media query**

**@media [only]** media-type **[and** (expression-1)] **[and** (expression-2)]... { ...styles go here

}

**Common properties for the screen media type**

**Property Description**

|  |
| --- |
| **width** The width of the viewport.  **height** The height of the viewport.  **device-width** The width of the display area of the device. This is the same  as width if the viewport is set as shown in figure 8-8.  **device-height** The height of the display area of the device.  **orientation** Landscape or portrait. |

**A media query that checks that the viewport width is 767 pixels or less**

**@media only screen and (max-width: 767px) {...}**

**A media query that checks that the viewport width is between 480 and 767 pixels**

**@media only screen and (min-width: 480px) and (max-width: 767px) {...}**

**Desktop browsers that support media queries**

* Internet Explorer 9 and later
* Firefox 3.6 and later
* Safari 4 and later
* Opera 10 and later
* Chrome 5 and later

**Description**

* A *media query* is defined by a CSS3 *@media selector.* This selector specifies the media type for the query and, for the screen media type, one or more conditional expressions. If all of the conditions are true, the styles within the media query are applied to the page.
* Each conditional expression can check one of the properties listed above. These properties can also be prefixed with min- or max-.
* The screen size at which a media query is used to change the appearance of a page can be referred to as a *breakpoint.*
* Media queries are supported by all the current versions of all the major desktop browsers. That makes media queries easy to test in those browsers.
* Media queries are also supported by all the current versions of all mobile browsers.
* If you include the only keyword, older browsers that don't support media queries will check the media type but not the conditional expressions.

**300** *Section 1 A crash course in HTML and CSS*

**Common media queries for a responsive design**

Figure 8-10 presents some common media queries that are used to imple­ment a responsive design. Before I describe these media queries, you should know that there are two standard techniques for developing the queries you need. First, you can start with the desktop design and then develop media queries that will be applied successively to the next smallest screen size. Second, you can start with the design for the smallest screen size and then develop media queries that will be applied successively to the next largest screen size. This design technique is often referred to as "mobile first design".

If you're developing a new website, the technique you choose may depend on personal preferences. If you're used to designing pages for the desktop, for example, you may want to do that first. But even if you do, you'll want to keep the mobile design in mind from the start. That will make it easier to add the media queries you need for smaller devices later on.

Another consideration is whether your website will be accessed primarily by users on mobile phones or by users on larger devices like tablets and computers. If it will be used primarily by mobile phone users, it makes sense to develop the mobile design first. Then, you can write the media queries for implementing the site on larger devices later.

If you're developing media queries using a desktop down design, your media queries should look something like those shown in the first example in this figure. These media queries check the max-width property, so they apply the styles they contain if a device has a width that's less than or equal to the value that's specified. Here, the first media query is for full-size tablets in portrait orientation (768 to 959 pixels), the second media query is for mobile phones

in landscape orientation (480 to 767 pixels), and the third media query is for mobile phones in portrait orientation (479 pixels or less).

In addition to these media queries, you could include a media query for the desktop and tablets in landscape orientation (960 pixels or larger), but these styles are typically coded outside the media queries. You could also include a media query for all mobile phones. Then, that query would contain styles that apply to any mobile phone, and the other two queries for phones would contain styles specific to phones in landscape or portrait orientation.

The second example in this figure shows the media queries that are commonly used for a mobile up design. These queries check for smaller screen widths first by using the min-width property. Because of that, the styles in these media queries are applied if a device has a width that's greater than or equal to the value that's specified. Just as in the first example, you could also code the styles for the smallest device in a media query, but that's not necessary.

It's important to keep in mind that the media queries must be coded in the sequence shown here. That way, the styles in each media query can override the styles in the previous media query if the condition on that media query is satis­fied. If you're using desktop down design, for example, and the screen width is between 480 and 767 pixels, the desktop styles will be applied first, followed by the tablet portrait styles, and then the mobile landscape styles. This will make more sense when you see the example at the end of this chapter.

*Chapter 8 How to use Responsive Web Design* **301**

**Figure 8-10 Common media queries for a responsive design**

**Common media queries for a desktop down design**

**/\* tablet landscape and desktop layout (960px or more) \*/ ...tablet landscape and desktop styles go here**

**/\* tablet portrait layout (768px to 959px) \*/ @media only screen and (max-width: 959px) { ...tablet portrait styles go here**

**/\* mobile landscape layout (480px to 767px) \*/ @media only screen and (max-width: 767px) { ...mobile landscape styles go here**

}

**/\* mobile portrait layout (479px or less) \*/ @media only screen and (max-width: 479px) { ...mobile portrait styles go here**

**Common media queries for a mobile up design**

**/\* mobile portrait layout (479px or less) \*/ ...mobile portrait styles go here**

**/\* mobile landscape layout (480px to 767px) \*/ @media only screen and (min-width: 480px) { ...mobile landscape styles go here**

**/\* tablet portrait layout (768px to 959px) \*/ @media only screen and (min-width: 768px) ...tablet portrait styles go here**

**/\* tablet landscape and desktop layout (960px or more) \*/ @media only screen and (min-width: 960px) {**

**...tablet landscape and desktop styles go here**

}

**Description**

* One standard technique for developing the media queries for a responsive design is to start with the queries for larger devices and then work your way down to smaller devices.
* Another standard technique for developing media queries is to develop the design for the smallest mobile devices first and then work your way up to larger devices.
* When you use the techniques shown above, each media query inherits the styles that precede it in the style sheet, including styles that are coded outside of media queries. Then, the media queries can override the inherited styles.
* You can also code the starting set of styles in a media query. For a desktop down design, you would code the styles in a media query that specifies a minimum width of 960 pixels. For a mobile up design, you would code the styles in a media query that specifies a maximum width of 479 pixels.

**302** *Section 1 A crash course in HTML and CSS*

**How to build responsive menus with the SlickNav plugin**

When you develop a responsive design, you need to provide for menus that can be accessed easily on small mobile devices. One way to do that is to use a jQuery plugin called SlickNay. Figure 8-11 shows how this plugin works.

To use the SlickNav plugin, you start by downloading its Zip file from <http://slicknay.com> and then unzipping this file. Next, you copy the slicknay.css and jquery.slicknay.min.js files to your website. Then, you include a link element for the .css file, which includes the styles for the plugin, and you include a script element for the .js file, which includes the JavaScript for the plugin.

In addition to the link and script elements for the SlickNav plugin, you must include a script element for the jQuery core library, since this library is used

by the SlickNav plugin. Note that this element must be coded before the script element for the plugin. Also note that you can retrieve the jQuery library from the Content Delivery Network (CDN) on the jQuery website by coding the script element as shown here. That way, you don't have to download this library.

Finally, you code a script element that calls the slicknav method within the jQuery ready event handler. To do that, you use a jQuery selector that refers

to the id of the standard navigation menu, followed by the method name. Note that this menu must be coded using an unordered list as shown in the previous chapter. You can also code one or more parameters for the slicknav method. Here, the prependTo parameter is used to add the mobile menu to the nav element that has an id of "mobile\_menu". Without this parameter, the menu would be added at the beginning of the body element. For a complete list of parameters, see the SlickNav website.

When you use the SlickNav plugin, you must provide a way to display the mobile menu on smaller screens and hide it on larger screens. To do that, you use media queries. If you're using a desktop down design, for example, you'll want to include a rule set in the desktop styles that hides the mobile menu by setting its display property to "none". Then, when you want to display this menu, you set the display property for this menu to "block". In the example in this figure, this is done by using an id selector.

In addition to hiding and displaying the mobile menu, you can change its appearance using classes that are defined in the slicknay.css file. In the next figure, for example, you'll see a mobile menu with a different background color than the one shown here. To find out what classes are available, just display the slicknay.css file.

To use the SlickNav menu, you simply click on the MENU button to display the first level of the menu. Then, you can click an item in the menu to display another page. Or, if a menu item has a submenu, you can click on it to display that submenu. To hide any menus that are displayed, just click the MENU button again.

*Chapter 8 How to use Responsive Web Design* **303**

**A multi-tier menu that uses SlickNav**



**Code for the SlickNav plugin in the head element**

**<link rel="stylesheet" href="styles/slicknay.css">**

**<script src="**[**https://code.jquery.com/jquery-2.1.3.min.js**](https://code.jquery.com/jquery-2.1.3.min.js)**"></script> <script src="js/jquery.slicknay.min.js"></script>**

**<script type="text/javascript"›**

**$(document).ready(function(){ $(1#nav\_menu1).slicknav({prependTo:"#mobile\_menu"}); ));**

**</script>**

**HTML for the SlickNav menu and the navigation menu**

**<nav id="mobile menu"></nav>**

**<nav id="nav menufl>**

**<ul>**

**<li><a href="index.html">Home</a></li>**

**</ul> </nav>**

**CSS for the SlickNav menu**

**/\* hide the mobile menu \*/ #mobile**\_**menu { display: none;** }

**/\* display the mobile menu \*/ #mobile\_menu { display: block; }**

**Description**

* To use the SlickNav plugin, download the Zip file from <http://slicknay.com>. Then, unzip this file and add the slicknay.css and jquery.slicknay.min.js files to your website.
* Before you can use the SlickNav plugin on a web page, you must include a link element for the slicknay.css file and a script element for the jquery.slicknay.min.js file. You must also include a script element for the jQuery core library, and this element must be coded before the script element for the jquery.slicknay.min.js file.
* To use the SlickNav plugin, add a script element with the jQuery code shown above that refers to the element that contains the unordered list for the menu and the element for the mobile menu. Then, use the CSS display property to hide and display the mobile menu depending on the size of the screen.

**Figure 8-11 How to build responsive menus with the SlickNav plugin**

**304** *Section 1 A crash course in HTML and CSS*

**A web page that uses Responsive Web**

**Design**

To complete this chapter, the next three figures present a web page that uses a responsive design. This is the same web page you saw in figure 8-7 that used a fluid layout, but this page also uses media queries.

**The design of the web page**

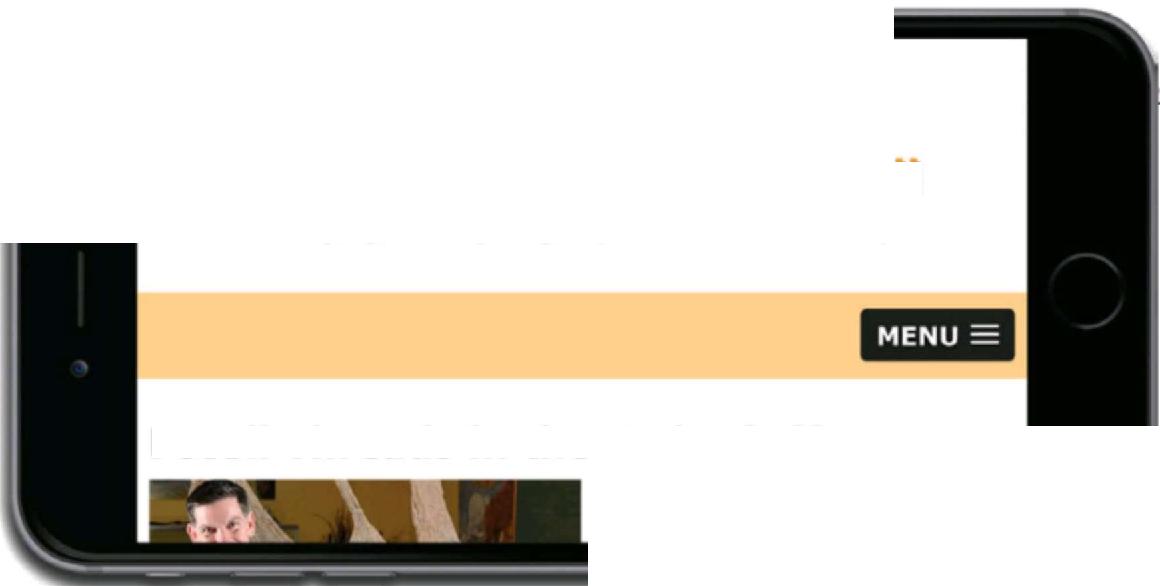
Figure 8-12 shows the design of this page on the desktop and in both landscape and portrait orientation on an iPhone 6. The design of this page in portrait orientation on a tablet is identical to the design for the desktop, except that some of the fonts are smaller.

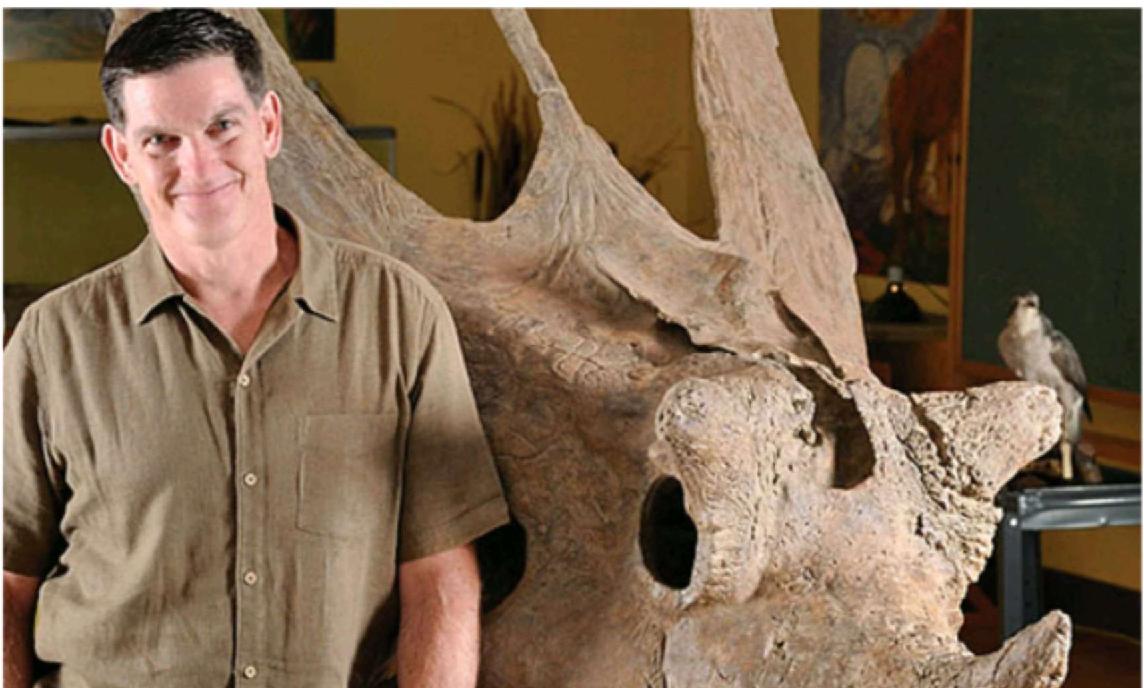
If you compare the appearance of this page in the desktop browser with its appearance in the iPhone in landscape orientation, you'll notice some major differences. To start, everything is smaller on the iPhone, which you would expect. Second, the margin and border have been removed from the page so it can use all of the available screen. Third, the logo is displayed above the text for the header and the logo and text are centered. If you scrolled down to the bottom of the page, you would see that the footer is centered too. Fourth, a SlickNav menu is displayed instead of a standard navigation menu. Fifth, the content is displayed in a single column with the article displayed before the sidebar. And sixth, the image in the article is floated to the left and it's sized so it takes up only a portion of the article width.

You'll also notice one main difference between the appearance of this page in landscape orientation and in portrait orientation on an iPhone. That is, the image in the article is changed back so it once again occupies the full width of the article.

*Chapter 8 How to use Responsive Web Design* **305**

**A speaker page in desktop and mobile layouts**

**111 San Joaquin** Valley Town x



**Fossil Threads in the Web of Life**



ATM aG 10 13 AM

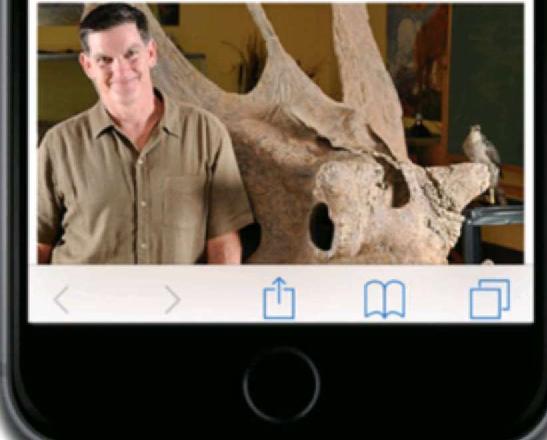
[murach.com](http://murach.com)

Lz.in juzgluir, Tr.wn

*Bringing cutting-edge speakers to the
  
valley*

**MENU**

Fossil Threads in the Web of Life



C 127.0.0.1:8020/HTML5%20Book%20Apps/ch08/ch08\_media\_queries\_fluid/town\_hall/sampson.html

Figure 8-12 The design of the web page

|  |
| --- |
| **rim Salm VEllay Town r J** J J  ***Bringing cutting-edge speakers to the valley*** |

**Home Speakers Get Tickets Become a Member About Us**

**This season's guest speakers**

* October: Jeffrey -roobin
* November: **Andrew Ross Sorkin**
* **January: Amy Chua**
* **February: Scott Sampson**
* **March: Carlos Eire**
* **April: Ronan Tynan**

What's 75 million years old and brand spanking new? A teenage Utahceratops! Come to the Saroyan, armed with your best dinosaur roar, when Scott Sampson, Research Curator at the Utah Museum of Natural History, steps to the podium. Sampson's research has focused on the ecology and evol

fieldwork **in a** r*r*

sari jr.)D:JUifl IzdJ7z/Tr.pm, /11111

***Bringing cutting-edge speakers to the valley***

**Fossil Threads in the Web of Life**

What's 75 million years old and
  
brand spanking new? A teenage

**Description**

* This web page uses a fluid layout and scalable images like the one in figure 8-7.
* This web page also uses media queries to change the appearance of the page depending on the size of the screen where it's displayed.
* The tablet portrait layout for this page is identical to the desktop and tablet landscape layout except that the font sizes are reduced.