13 DESIGNING FOR MOBILE DEVICES

Lesson overview

In this lesson, you'll learn how to do the following:

- Access and configure Dreamweaver's tools and interface for mobile design
- Create a mobile-friendly page layout based on a web framework
- Insert and format new content and components into a Bootstrap-based layout
- Apply a mobile template to existing site pages

This lesson will take about 60 minutes to complete. If you have not already done so, please log in to your account on peachpit.com to download the project files for this lesson as described in the "Getting Started" section at the beginning of this book and follow the instructions under "Accessing the Lesson Files and Web Edition." Define a site based on the lesson13 folder.

Responsive design

The Internet was not designed for smartphones and tablets. In the '90s, the most difficult challenge a programmer or developer faced was accounting for the size and resolution differences between 13- and 15-inch computer monitors. For years, resolutions and screen sizes only got *larger*.

Today, the odds that some or all of your visitors are accessing your site with a smartphone or tablet are increasing daily. Statistics show that starting in 2014 more people used mobile devices to access the Internet than used desktop computers, and that number has been increasing steadily ever since.

Mobile-first design

As the number of people favoring phones and tablets over desktop computers continues to grow, the logical evolution is toward a philosophy known as *mobile-first* design. It assumes that sites will shed users and traffic if they aren't optimized for phones and tablets.

Mobile-first design starts with a design for mobile devices, such as smartphones, and then adds content and structure for larger devices and computers. In some cases, the site's content is actually minimized so that it loads and performs in an optimal fashion, and then additional content is *injected*, using JavaScript and online databases, for computers and more powerful devices.

How your site deals with smartphones and mobile devices depends on whether you're adapting an existing site or developing a new one from scratch. So what does all this mean to the website you built over the last 12 lessons? Let's take a look.

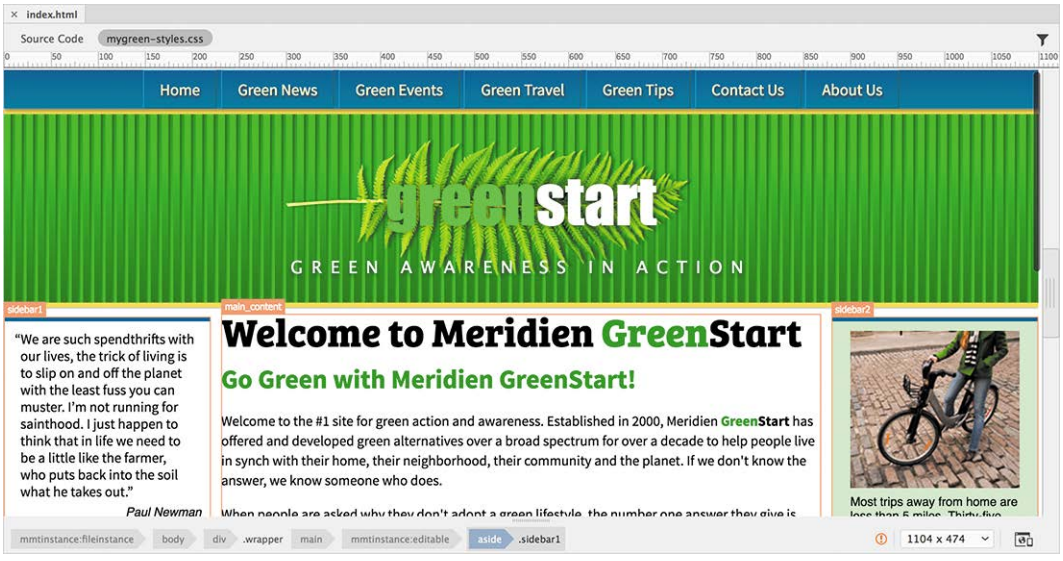
Testing responsiveness in Dreamweaver

You could upload the GreenStart site to the Internet, as you learned how to do in Lesson 11, "Publishing to the Web," and test the various pages on desktop computers, smartphones, and tablets. But most of the information you need can be found right inside Dreamweaver itself. You have learned that Live view was designed to preview the HTML, CSS, and JavaScript creating your page content exactly as it would appear on the web. This functionality also supports responsive design techniques.

1. Launch Dreamweaver CC (2018 release or later).
2. If necessary, select lesson13 from the site's drop-down menu in the Files panel.
3. Maximize the size of the program interface to fill the entire computer display.

   Many mobile designs are keyed into the display size and width of the browser window. It's essential to make sure the Dreamweaver document is as large as the computer display will allow.

4 Open **index.html** from the site root in Live view.



This page was created from the site template in Lesson 11. It represents the basic design of the various pages in the GreenStart website to this point. The Dreamweaver interface provides several features that allow you to test and control the responsiveness of a webpage.

Before Dreamweaver CC, you had to resize the whole document window to test the responsiveness of a particular webpage. The scrubber tool was added to Live view to enable you to interact with pages without changing the size of the program interface. It appears by default on the right edge of the Live view window.

▶ **Tip:** As you drag the scrubber, notice that the width and height of the document window is displayed on the right side of the scrubber.

5 Drag the scrubber to the left to set the width of the document window to 800 pixels.

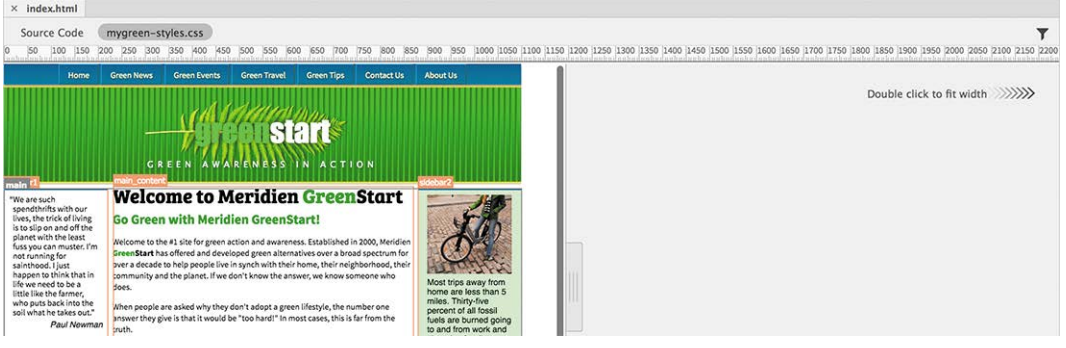
As the window decreases in width, Dreamweaver simulates what the page would look like if it were loaded in a browser or device 800 pixels in width.

6 Drag the scrubber to set the document window to 600 pixels.

Once the width of the window decreases from full size, the right side of the page is obscured and unusable. The page does not respond in any way to the changing width. Since the site template is based on a fixed-width design, it was never intended to adapt automatically to smaller screens. Webpages like this react to different screen sizes in one of two ways.

On desktop displays, the page may display at actual size and simply scroll off the screen to the right, as you see it now in Dreamweaver. The other option, which will occur on smartphones and tablets, is that the page may simply scale in size to fit the available screen. You can simulate that effect in the document window by zooming out.

7 Press Ctrl+–(minus)/Cmd+– (minus) twice to zoom out.

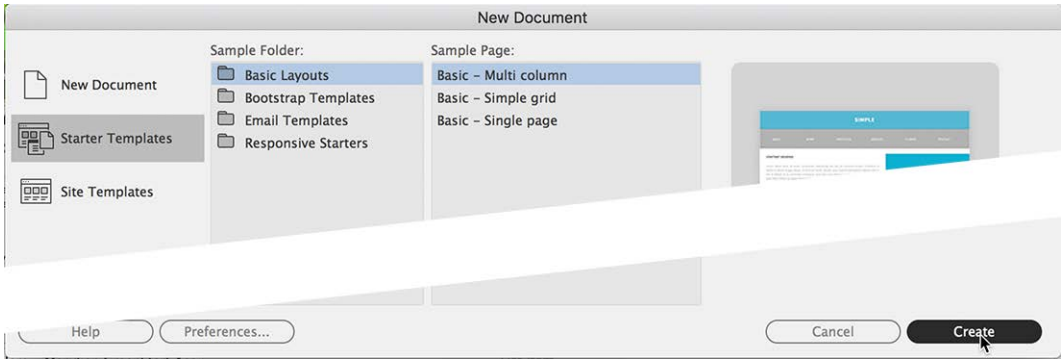


The webpage scales smaller to fit into the reduced space.

Whether the article scrolls off the screen or scales down to fit, the result is a webpage that is useless to many visitors. The alternative to a fixed-width design is one that can adapt automatically to different screens and devices. Dreamweaver provides some built-in examples of such designs.

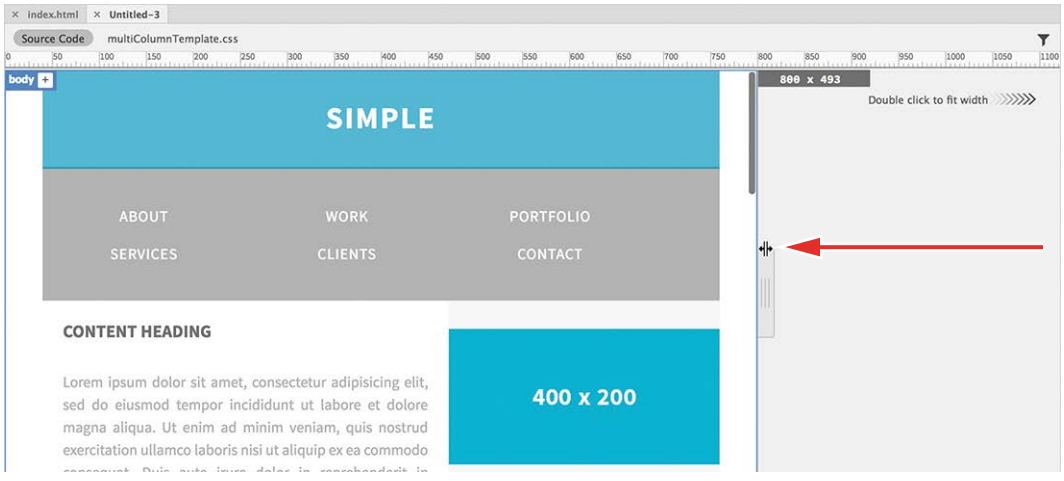
8 Select File > New.

9 In the New Document dialog, select Starter Templates > Basic Layouts > Basic – Multi Column and click Create.



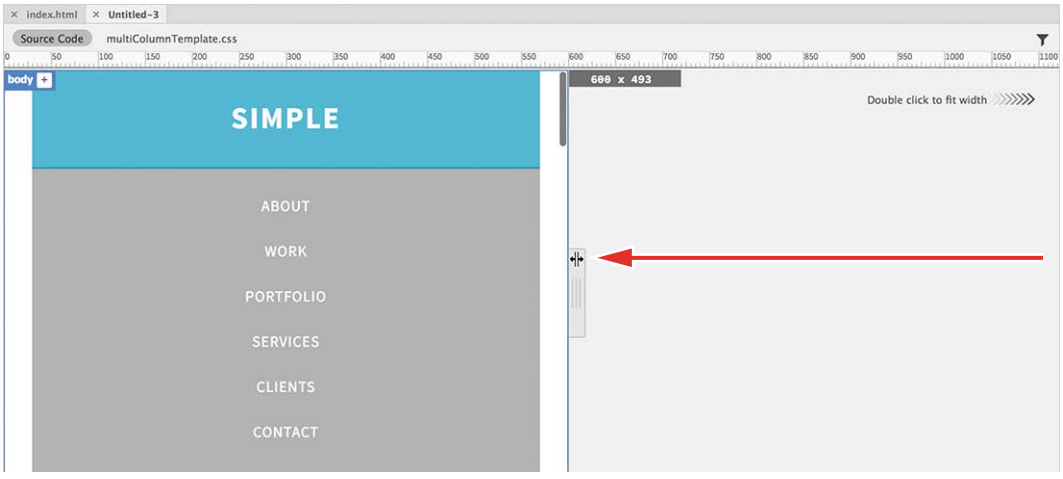
The document window displays a new untitled page based on the selected starter template. All the new Dreamweaver templates provide various levels of responsiveness.

10 Drag the scrubber to the left to 800 pixels.
Observe the changes to the page content.



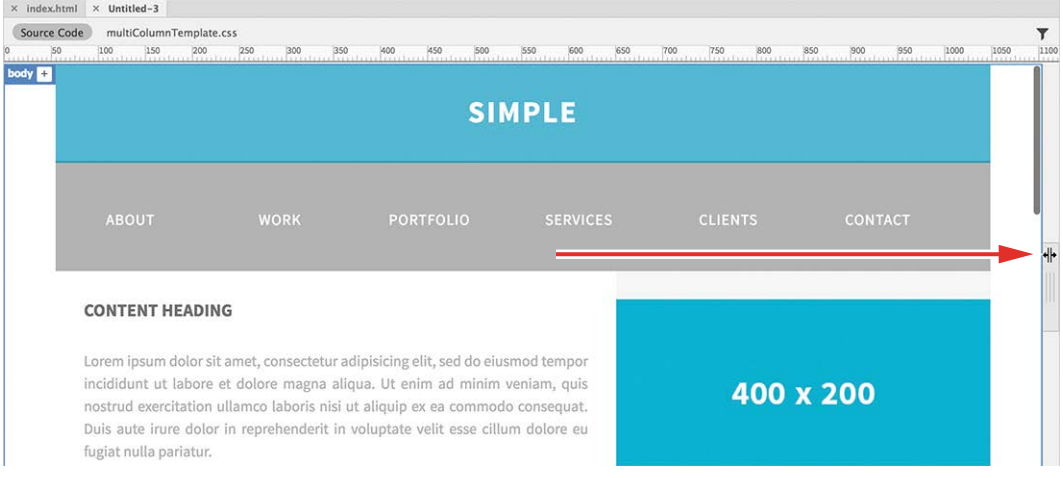
As the width decreases, the page content automatically reacts to changes. The navigation menu switches from a single row to two rows. The text and image placeholders resize to share the remaining space. At no time while you are resizing the window does any of the content scroll off to the right. It all remains within the visible area and accessible.

11 Drag the scrubber to 600 pixels.



As you drag the scrubber, you will see the content continue to adapt. Eventually, the content shifts to form a single column. The original horizontal navigation now stacks in a single vertical column. This behavior also works in reverse.

12 Drag the scrubber to the right.



As you drag the scrubber, the document reformats on the fly, putting everything back where it originally came from. This isn't magic; it's responsive design. It's based on some concepts you already know and some you are about to learn.

No matter whether you decide to design for desktop first or mobile first, you still need to learn how to make webpages that can respond automatically to devices of any size. To start, you have to learn about two basic CSS parameters: *media type*

and *media query*. These parameters enable browsers to identify what size and type of device is accessing the webpage and then load the appropriate style sheet, if one exists.

Media type properties

The media type property was added to the CSS2 specifications and adopted in 1998. It was intended to address the proliferation of non-computer devices that were able to access the web and web-based resources at that time. The media type is used to target customized formatting to reformat or optimize web content for different media or output.

In all, CSS includes ten individually defined media types, as shown in **Table 13.1**.

Table 13.1 Media type properties

PROPERTY	INTENDED USE
`all`	All devices. "All" is the default media type if one is not specified in the code.
`aural`	Speech and sound synthesizers.
`braille`	Braille tactile feedback devices.
`embossed`	Braille printers.
`handheld`	Handheld devices (small screen, monochrome, limited bandwidth).
`print`	Documents viewed onscreen in print preview mode and for printing applications.
`projection`	Projected presentations.
`screen`	Primarily for color computer screens.
`tty`	Media using a fixed-pitch character grid, such as Teletypes, terminals, or portable.
`tv`	Television-type devices (low resolution, color, limited-scrollability screens, sound available).

Although the media type property works fine for desktop screens, it never really caught on with browsers used on cellphones and other mobile devices. Part of the problem is the sheer variety of devices available in all shapes and sizes. Add to this smorgasbord an equally diverse list of hardware and software capabilities and you've produced a nightmare environment for the modern web designer. But all is not lost.

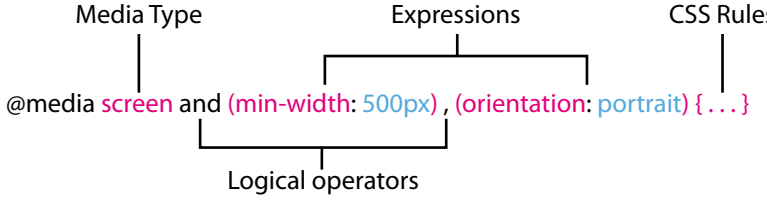
Media queries

A media query is a newer CSS development that enables a webpage to interactively determine what formatting to use based not only on what kind of device (media type) is displaying the page but also on what dimensions and orientation it's using. Once the browser knows the type or size of the device it has encountered, it reads the media query to know how to format the webpage and content. This process is as fluid and continuous as a precision dance routine, even allowing the visitor to switch orientations during a session and have the page and content adapt seamlessly without other intervention. The key to this ballet is the creation of style sheets optimized for specific browsers, specific devices, specific orientation, or all three.

Media query syntax

Like the CSS it controls, a media query requires a specific syntax to work properly in the browser. It consists of one or more media types and one or more expressions, or media features, which a browser must test as true before it applies the styles it contains. Currently, Dreamweaver supports 22 media features. Others are being tested or are still under development and may not appear in the interface, but you can manually add them to the code, if necessary.

The media query creates a set of criteria to determine whether a specific set of rules contained within it is applied in a webpage.



You can create media queries in a variety of ways. For example, they can be designed to work exclusively (by completely resetting the existing styling) or in tandem (by inheriting some styles and modifying specifications only as necessary). The latter method requires less CSS code and is typically more efficient. We will favor that method in the upcoming exercises and for the new responsive site design.

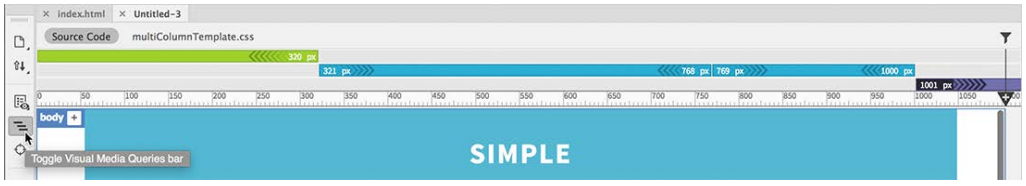
To learn more about media queries and how they work, check out www.w3schools.com/cssref/css3_pr_mediaquery.asp.

Working with the Visual Media Queries interface

One of the best aspects of working with Dreamweaver is that it provides visual tools to help you style and troubleshoot various CSS properties. The Visual Media Queries (VMQ) interface is one such tool. It allows you to identify and interact with media queries instantly using the cursor.

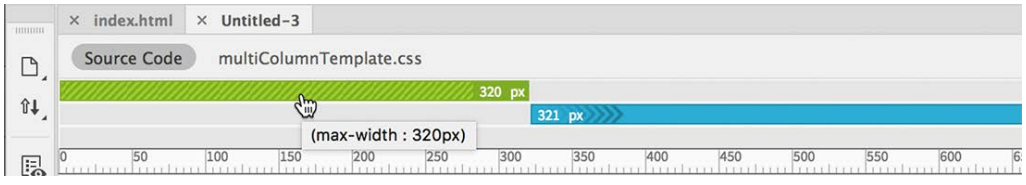
Note: The following exercise requires the sample document created from the Basic – Multi Column starter template in a previous exercise.

1 If necessary, click the Toggle Visual Media Queries Bar icon in the toolbox on the left side of the screen.



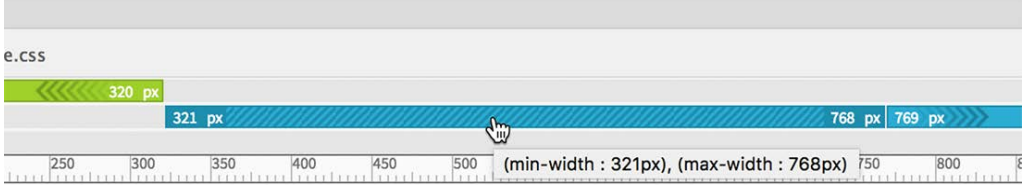
Depending on the width of your screen, the toolbar displays four media queries applied to this document.

2 Position the cursor over the first media query.



A tooltip appears, identifying that the media query is set for a max-width of 320 pixels. In other words, the styling controlled by this media query applies only to screens 320 pixels wide or narrower.

3 Position the cursor over the second media query.



The second media query is set for a min-width of 321 pixels and max-width of 768 pixels. By using both min-width and max-width, this media query targets screens between 321 and 768 pixels. The interface displays the media query settings using colored bars labeled with the pixel dimensions so that you can see the specifications instantly.

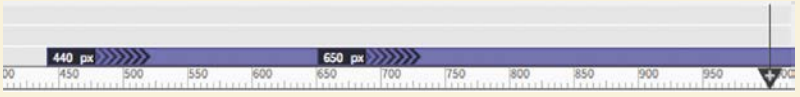
You can use the cursor to activate or switch the document view between each media query.

VMQ enables you to see the differences

Media queries enable your webpage and its content to adapt to a variety of different types of screens and devices. They do this by loading custom style sheets created for specific screen sizes, devices, or even orientations. Later, you'll learn more about media queries, how they work, and how to create them. For now, let's review how the Visual Media Queries interface works.

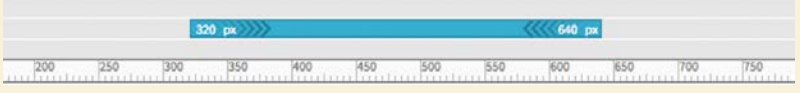
The VMQ interface identifies all media queries defined on the page or within style sheets linked to it. The media queries are displayed in color based on their specifications.

Media queries that define only a minimum width are displayed in purple.



Media queries that define only a maximum width are displayed in green.

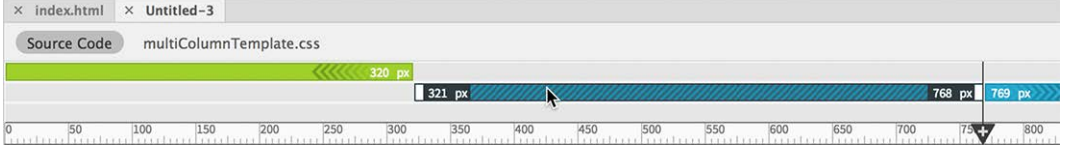
Media queries that define both minimum and maximum widths are displayed in blue.



The Dreamweaver workspace is fully responsive and will display the specific CSS styling appropriate to the screen size and orientation within CSS Designer. To display the styling associated with a specific media query, simply click the media query notation in the @Media pane of CSS Designer.

Note: Don't click directly on the number itself, because it will open a field that allows you to edit the width setting.

4 Click the cursor to the right of the number 321 px in the VMQ interface.

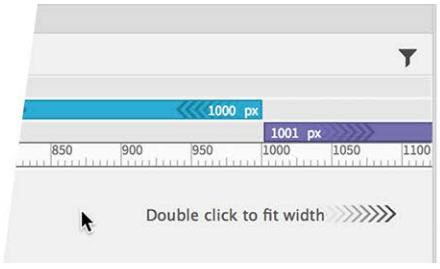


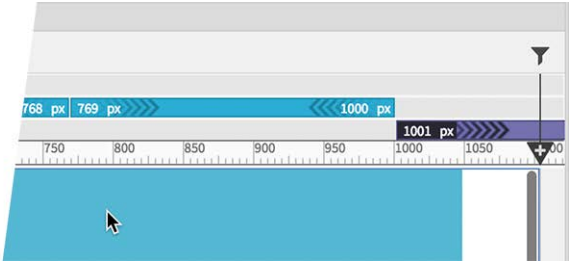
The document window expands instantly to 768 pixels. The content adapts to the window size based on the applicable CSS styles.

5 Click the same media query again.

The document window narrows to 321 pixels. Clicking the media query a second time toggles the display between its beginning and end. To return to full size, drag the scrubber to the right edge of the window or double-click in the gray area to the right of the scrubber.

6 Double-click anywhere in the gray area on the right side of the scrubber.

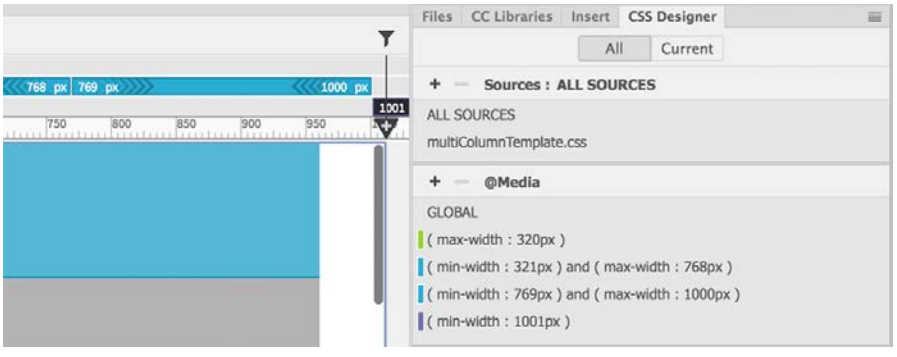




The document window opens to the full size of the workspace.

The Visual Media Query interface display is integrated closely with CSS Designer and reflects the various specifications defined there.

7 If necessary, select Window > CSS Designer to display the panel. Open the @Media pane to display the settings defined within it.



In the @Media pane, you will see the GLOBAL attribute and four predefined media queries. Within each of these queries are sets of CSS rules that are geared to style and reformat the page on different screen dimensions. Switch between the two open documents and compare the display of the VMQ interface and CSS Designer. The main difference between the two is obviously the lack of media queries and the associated rules defined within them. The advantage of using the starter layouts in Dreamweaver is that many of them are based on powerful web frameworks.

Introducing web frameworks

If you look through the style sheet and at each media query displayed in CSS Designer in the Untitled document, you will see the various ways this starter template is formatted for different screen sizes. Creating CSS rules this way, to handle all types of content for every type of screen and device, can be a daunting task. Even experienced web designers think twice about building responsive designs from scratch. Because the number and types of mobile devices have proliferated exponentially in the last few years, adapting content successfully to every device and screen size is nearly impossible without a little help. That's why developers have created a number of front-end frameworks to make that task much easier.

A framework is a collection of predefined HTML and CSS code—often including JavaScript—that permits the rapid development and deployment of webpages and web applications. Most frameworks are built from the ground up to display content responsively on a wide range of devices. That way, you can concentrate on what the content says, instead of how, or whether, it displays.

Dreamweaver CC (2018 release) supports three web frameworks out of the box: jQuery UI, jQuery Mobile, and Bootstrap. The jQuery Accordion used in Lesson 10, "Adding Interactivity," is part of the jQuery UI framework, which provides a set of GUI widgets, animated visual effects, and themes. It's not designed to build an entire site but can add useful interactivity to existing pages.

jQuery Mobile is a touch-enabled web framework specifically optimized for smartphones and tablets. It's geared for building mobile-only websites or for integration with frameworks like PhoneGap or Worklight.

Created by Twitter, Bootstrap was released to the public in 2011 and quickly became one of the most popular frameworks in use. It was incorporated into Dreamweaver in a previous version and provides a powerful set of tools for building complete websites from the ground up that are fully optimized for mobile devices and desktop screens. One advantage of using Bootstrap is that there are plenty of resources and add-ons available that extend its capabilities independent of Adobe and Dreamweaver.

In the upcoming exercises, you'll learn how to build a custom layout using the Bootstrap framework to replace the current non-responsive site template and then update the existing site pages to support all size screens and devices.

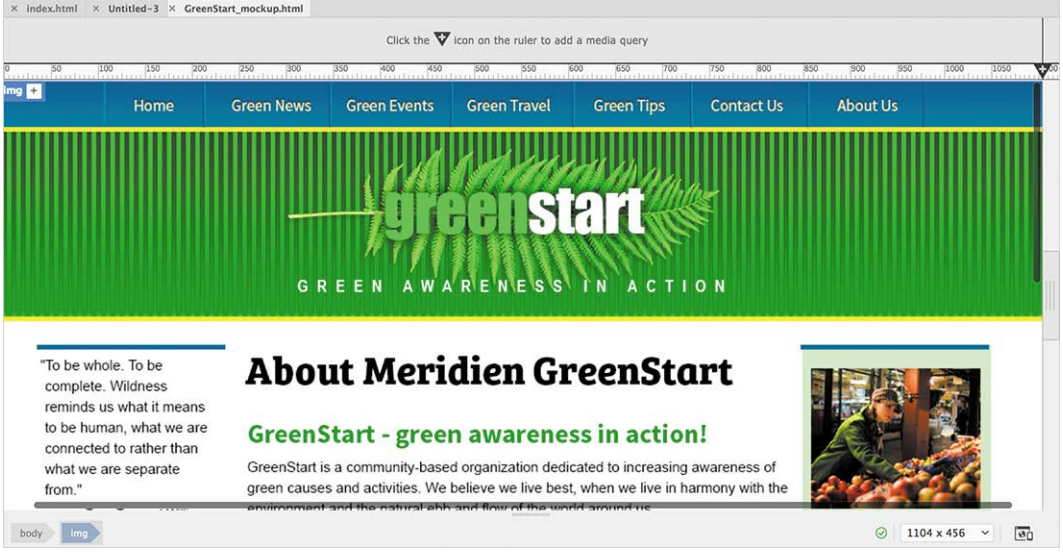
Identifying the Bootstrap structure

Underlying all the power and flexibility of Bootstrap is a basic grid system of rows and columns. This concept harkens back to the earlier days of web design, before the advent of CSS, when we would use tables to cobble together our layouts. The only way to impose order on our webpages was to organize our text and pictures into table rows and cells.

No, we're not going back to the bad old days. Tables were not responsive. Although they could scale up or down in size, they would not automatically adapt to the screen and knew nothing about mobile devices. Instead, the rows and columns of Bootstrap have been carefully engineered to work in most modern browsers and devices.

The first step in Bootstrap is to identify the basic grid structure you need to build, or impose, on the proposed site design. This should be quite easy when you reexamine the site mockup.

1 If necessary, open **GreenStart_mockup.html** in Live view.



To study the site mockup, start by marking up the content that would be grouped together in rows, as in the following figure.

Next, identify the columns within the content. Remember that the columns are divvied up in the rows you already created.

2 Close all open files. Do not save any changes.

Once you have the basic plan established, building the Bootstrap structure is a simple procedure using Dreamweaver.

Creating a Bootstrap layout

In this exercise, you will build the basic structure of the new template using the Bootstrap framework and the Insert panel.

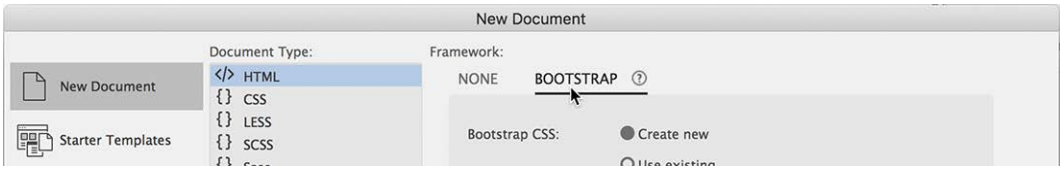
1 Select File > New.

The New Document dialog appears.

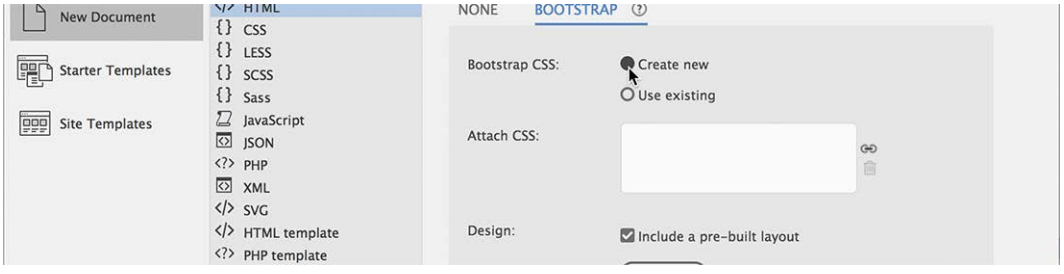
2 Select the New Document tab in the New Document dialog.

3 Select HTML in the Document Type window.

4 Click the Bootstrap tab.

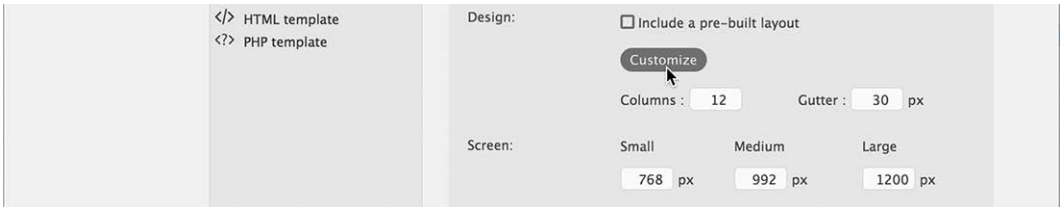


5 For Bootstrap CSS, select Create New.



6 Deselect the Design option Include A Pre-built Layout.

7 Click the Customize button.

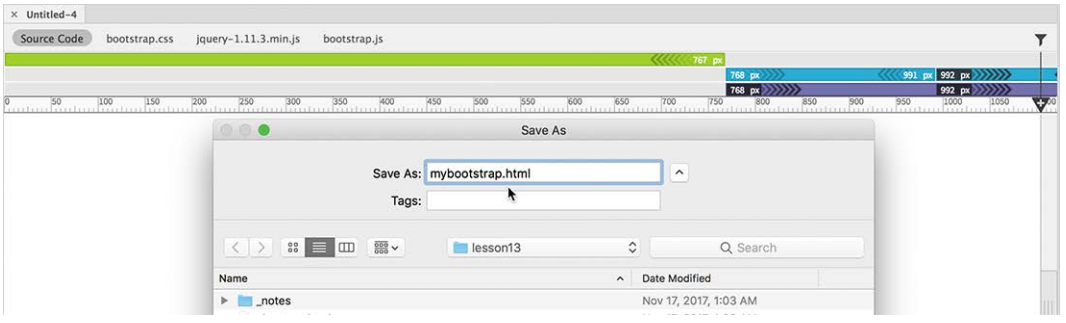


The Customize options allow you to change the number of columns, the gutter width, or the predefined screen sizes. For this layout, you will leave the default settings as they are, but if you use Bootstrap in the future, you should make sure these numbers reflect the needs of your site and its visitors.

8 Click Create.

A new, untitled document appears in the document window.

9 Select File > Save. Name the file **mybootstrap.html** and save it in the lesson13 folder.



Note: The supporting JavaScript and CSS files for various frameworks, like Bootstrap, are updated from time to time and may be different from the ones shown in this lesson.

Although the file seems to be entirely empty, lots of things are already going on. You can see that the VMQ interface displays at least six media queries and that the Related Files interface shows one CSS and two JavaScript files. But you've only started.

Adding Bootstrap components

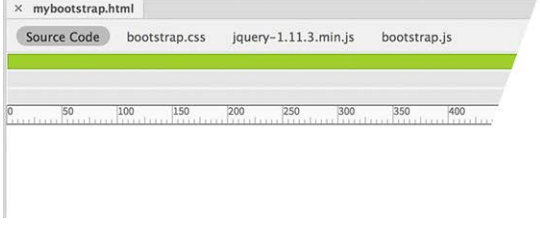
The new layout is set up to support the Bootstrap framework. The next step is to add some basic structures. In this exercise, you will add the basic page skeleton.

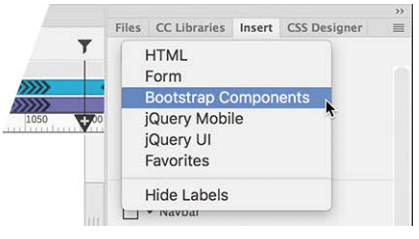
1 If necessary, open **myBootstrap.html** in Split view.
Click in the Live view window.

The <body> element is selected in the window.

2 Display the Insert panel. If it's not visible on the screen, you can select it from the Window menu.

3 In the Insert panel, select **Bootstrap Components** from the drop-down menu.



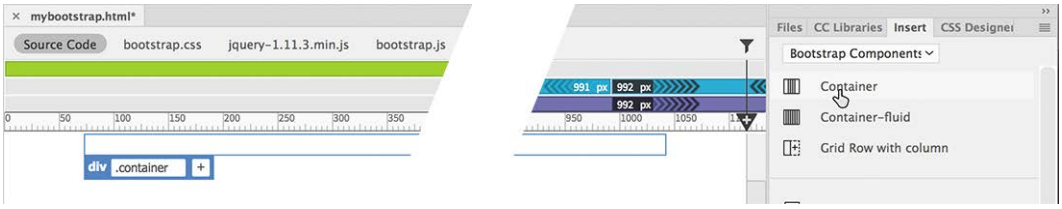


The panel displays a list of 26 main items and more than 80 subitems supported by the framework. Although the list is not exhaustive of all the possible Bootstrap components and widgets, it's a good start. And whatever you can't find in the Insert panel can always be manually added by using the Code window.

4 Click in the Live view document window.

The Element Display appears focused on the <body> element.

5 Click the **Container** item at the top of the Insert panel.

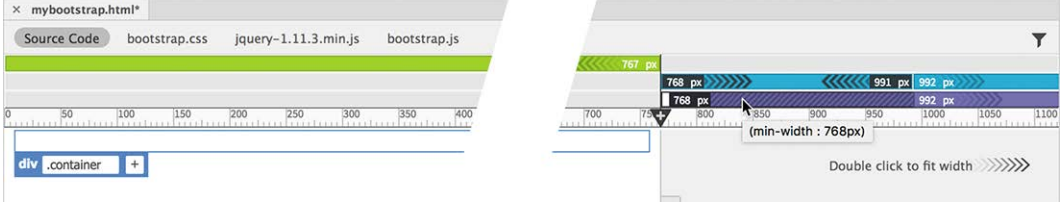


This option inserts a fixed-width element in the page. In Bootstrap it is, by default, 1170 pixels wide in a full-screen browser on a desktop computer. On smaller screens or devices, this container will display at various smaller fixed widths or scale as necessary to fit the screen.

Once you've established your overall container, you can start creating the row and column scheme devised earlier, but first you will target the default page width.

6 Click the *Small* (purple) media query (min-width 768 pixels) in the VMQ interface.

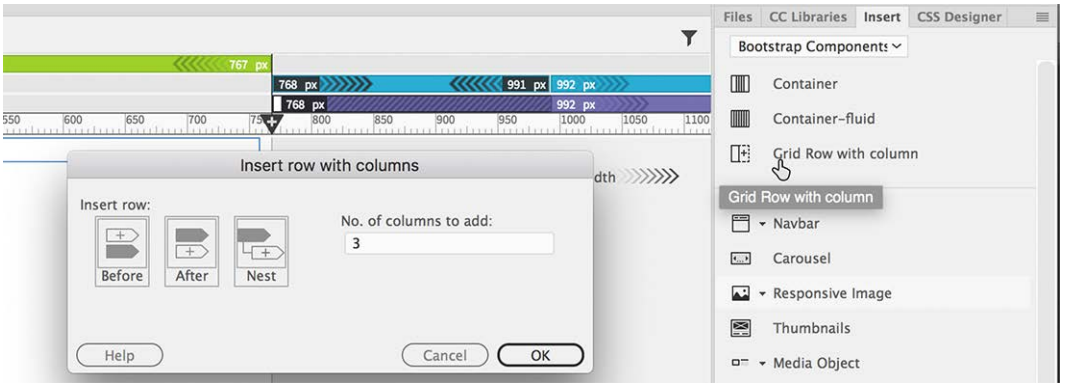
Note: To see all the media queries, the document window will have to be at least 1100 pixels in width.



The document window resizes to match the dimensions of the media query. Targeting a specific width first determines what classes the Bootstrap framework automatically assigns to the components when you add columns to the rows.

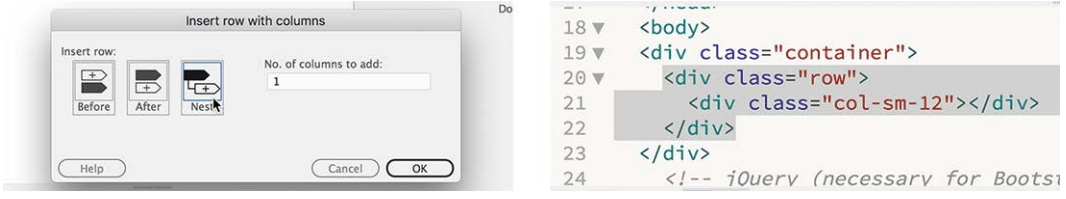
Note: The div element should be selected in the document window by default.

7 Click the **Grid Row With Column** item in the Insert panel.



The Insert Row With Columns dialog appears.

8 Click the **Nest** option, which inserts the row inside the container element created in step 4. Enter **1** in the No. Of Columns To Add field. Click OK.



In the Code window, you can see that Dreamweaver inserted two `<div>` elements, one with a class of `row` and the other with a class of `col-sm-12`, nested one inside the other in the initial container. Since the *Small* media query was targeted in step 5, the class says `sm`. *Medium* classes will say `md`, and *Large* classes will say `lg`.

The structure doesn't look very remarkable, but this is the key to the power of Bootstrap. These classes apply predefined styles to the elements that will allow them to adapt to different screens and devices. By adding more classes or manipulating the existing ones, you can provide different types of formatting and behaviors as desired.

As complex and elaborate as Bootstrap might be, one aspect of this scheme is easy to understand. If you remember what you saw in the New Document dialog, the Bootstrap specifications called for 12 columns in the grid. The class `col-sm-12` speaks to this grid by telling the `<div>` to be *12 columns* wide on *small* devices. A small device is considered to be a tablet at least 768 pixels wide. But don't let that fool you. It's important to know that some Bootstrap classes, like this one, are based on inheritance theory and format elements even

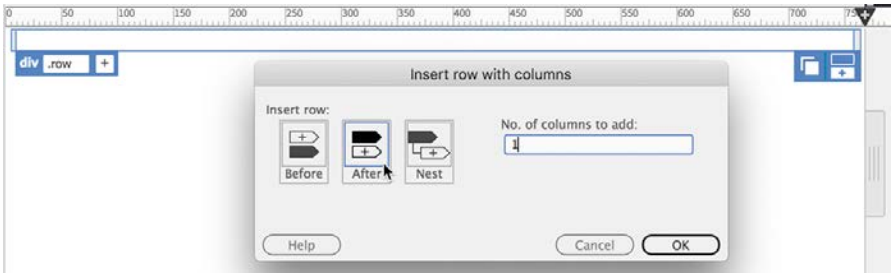
on larger devices. In other words, this class will continue to format the page unless another class overrides the styling.

As we work through this layout and the upcoming lessons, you will learn how to add to the main components other Bootstrap classes that will specify their behavior in each target environment you want to support. The first row will hold the main site navigation menu. Let's continue building this layout by adding a new row for the page header next. The first row should still be selected.

9 Click the **Grid Row With Column** item again.

The Insert Row With Columns dialog appears.

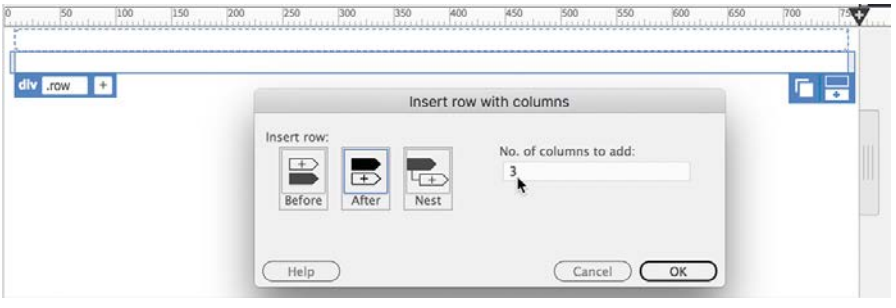
10 Enter **1** for the number of columns.
Select **After**, which will insert a row after the current one. Click OK.



A new row is created, inserted after the first. The second is a duplicate of the first and will eventually hold the header and company logo. You'll add that later, but now let's create the next row.

Note: Pay close attention to where the new element appears in the responsive structure. Make sure the new row appears separate from and below the first but wholly inside the container element.

11 Insert another Grid Row With Column, as in step 7.
In the Insert Row With Columns dialog, enter **3** this time for the number of columns and select After.



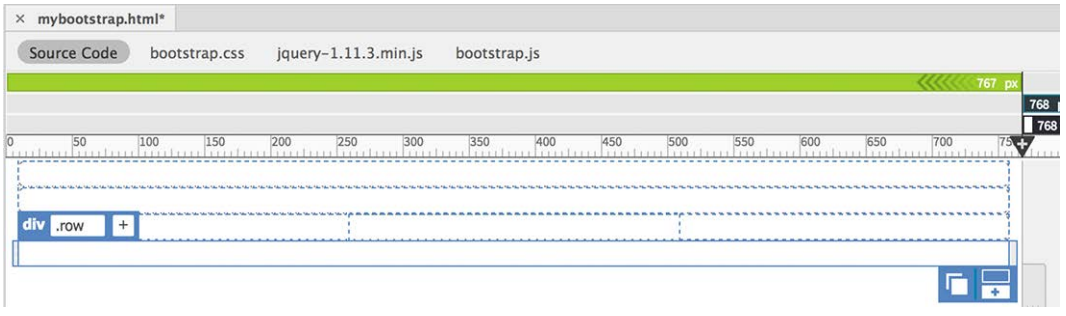
12 Click OK.

A new row appears with three nested `<div>` elements. The new elements have a class of `col-sm-4`. Because 4 divides into 12 three times, the new elements form three columns that divide the available space into three equal parts. Later,

you will modify these classes to change the widths and the relationships of these elements to one another. But let's finish the layout first. There's one more row needed for the page footer.

```
19 ▼  <div class="container">
20 ▼    <div class="row">
21        <div class="col-sm-12"></div>
22      </div>
23 ▼    <div class="row">
24        <div class="col-sm-12"></div>
25      </div>
26 ▼    <div class="row">
27        <div class="col-sm-4"></div>
28        <div class="col-sm-4"></div>
29        <div class="col-sm-4"></div>
30      </div>
31    </div>
```

13 Repeat steps 9 and 10 to create a row with one column.



A new row is added for the last row, which will hold the footer. The basic Bootstrap structure for the site design is now complete.

14 Save the file. If you are continuing to the next exercise, leave **myBootstrap.html** open, but close any other open documents.

In the upcoming exercises, you will modify the basic layout to add HTML5 elements and content placeholders for the site template.

Adding semantic elements to Bootstrap

As you can see from the previous exercise, Bootstrap relies heavily on the `<div>` element. There's nothing wrong with this technique, since the framework can't intuit the purpose of the elements in the rows and columns and automatically add the appropriate tags. But as a generic container, the `<div>` element conveys no semantic value, or other information, to search engines or other web applications.

Once you have created your basic structure, it makes sense to go back and swap out these generic structures with HTML5 semantic elements that more closely match your intended usage or content model, as you did in the existing website. Dreamweaver makes it easy to edit structural elements.

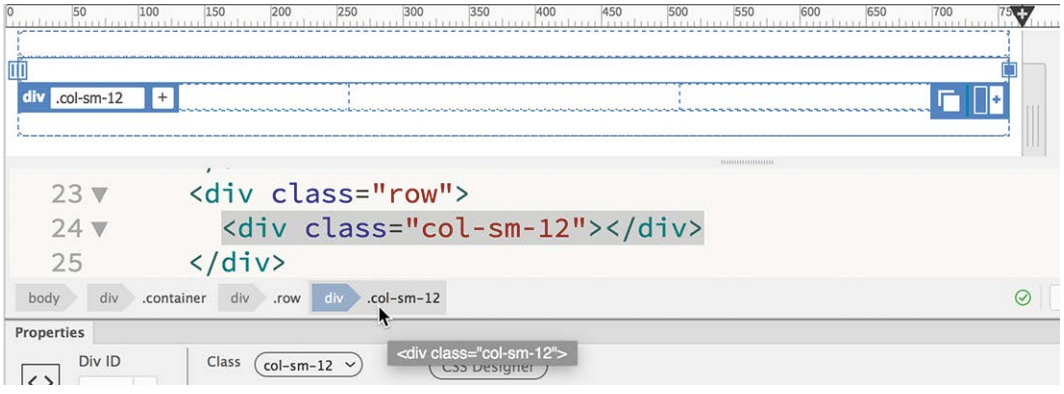
1 If necessary, launch Dreamweaver CC (2018 release) or later.
Open **myBootstrap.html** from the lesson13 folder.

The file has a basic Bootstrap structure containing four rows, three with one column and one with three columns.

2 Select Split view so that the workspace displays the Code and Live view windows at the same time.

The Bootstrap borders, rows, and columns should be visible in the Live view window as faint blue lines. Since we're going to add a dedicated navigation menu to the first row later, let's start on the second row.

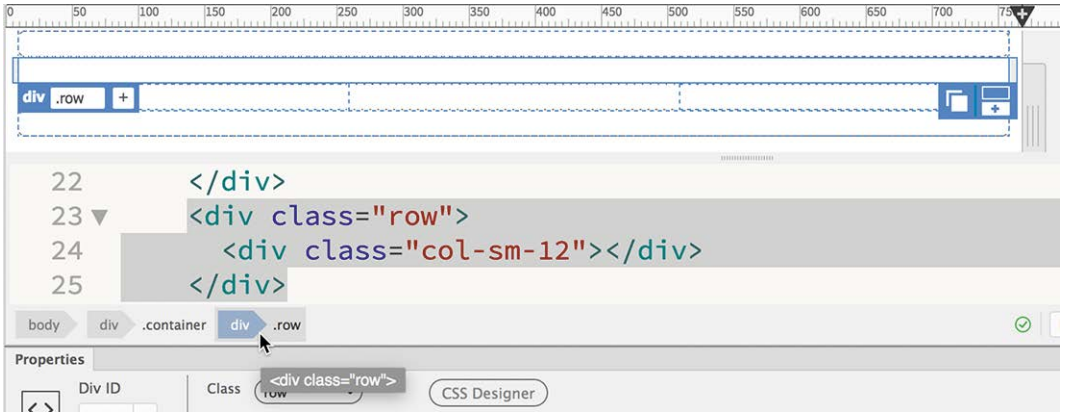
3 Click in the second row of the layout and examine the tag selectors at the bottom of the document window.



One of the elements in the row is selected. The Element Display appears focused on one of the nested elements.

Depending on where or how you click, you might select the row itself or the column nested within it. You can determine which element is selected by looking at the class name displayed in either the Element Display or the tag selectors. If it displays `div.row`, it indicates you have selected a row, whereas `div.col-sm-12` means you have a column selected. The selected element will be highlighted in the tag selectors interface.

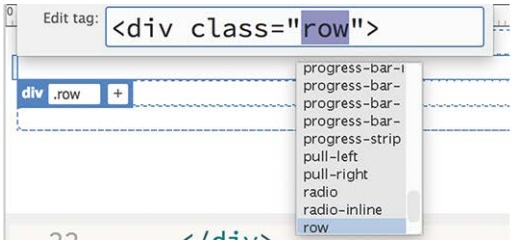
4 Click the tag selector for `div.row`.

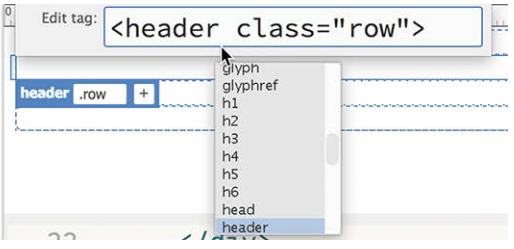


5 Press Ctrl+T/Cmd+T to activate the Quick Tag Editor.

The Quick Tag Editor appears, populated by the code for the row element. As you might recall from the original page diagram, this element should be designated as an HTML5 `<header>`.

6 Edit the element code as highlighted:
`<header class="row">`



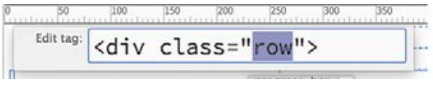


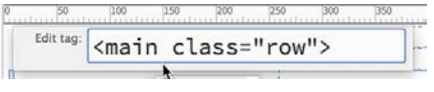
7 Press Enter/Return twice to complete the change.

The structure is now updated to use the new `header` element.

▶ **Tip:** You can also use the DOM viewer to select the column element.

8 Repeat steps 4 through 7 to edit the third row as highlighted:
`<main class="row">`



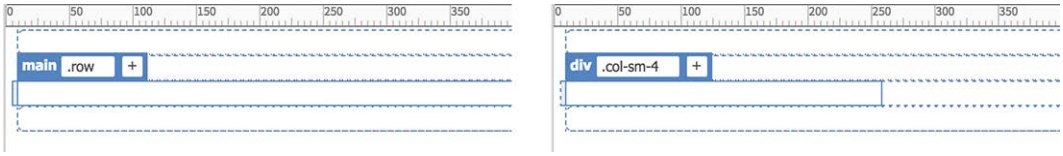


This row also contains three columns: one `article` element and two sidebars, or `aside` elements.

9 If necessary, click the third row.

The Element Display appears for `main.row`.

10 Press the Down Arrow key once.

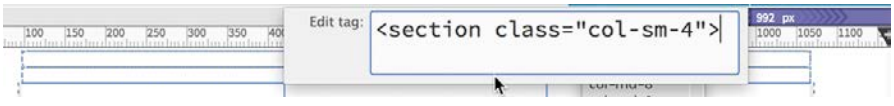


You can use the Up and Down Arrow keys in Live view to change the selection focus on consecutive elements in the HTML code. The first `div.col-sm-4` element in the row should be selected. We'll refer to this element as Sidebar 1 from this point on.

11 Press Ctrl+T/Cmd+T.
Edit the element as highlighted:
`<aside class="col-sm-4">`

This element will contain Sidebar 1.

12 Click the third row and press the Down Arrow twice to select the second column in `main.row`. Edit the column element as highlighted:
`<section class="col-sm-4">`



This element will contain the main content of each page.

13 Edit the third column as highlighted:
`<aside class="col-sm-4">`

This element will contain Sidebar 2.

14 Edit the fourth row as highlighted:
`<footer class="row">`

15 Save the file.

```
<div class="container">
  <div class="row">
    <div class="col-sm-12"></div>
  </div>
  <header class="row">
    <div class="col-sm-12"></div>
  </header>
  <main class="row">
    <aside class="col-sm-4"></aside>
    <section class="col-sm-4"></section>
    <aside class="col-sm-4"></aside>
  </main>
  <footer class="row">
    <div class="col-sm-12"></div>
  </footer>
</div>
```

The Bootstrap layout has now been updated to use HTML5 semantic elements. In the next lesson, you'll learn how to convert this file into an alternate site template that will then be applied to the existing site pages, as well as to format various elements.

Review questions

1 What is responsive design?

2 How are web frameworks used in responsive design?

3 What does the Visual Media Queries (VMQ) interface do?

4 What do the colors in the VMQ interface signify?

5 How does the scrubber work in conjunction with the VMQ interface?

6 Why should you consider using Bootstrap for your next website?

7 True or false: You have to use one of the six predefined templates if you want to use Bootstrap.

8 Why should you replace the `<div>` elements created by Bootstrap with HTML5 semantic elements?

Review answers

1 Responsive design is a method for designing webpages so that they automatically adapt to any size screen or device.

2 A web framework is a set of predefined HTML and CSS code, often including JavaScript, designed to support responsive design from the ground up and enable designers to quickly and easily build webpages and applications.

3 The VMQ interface provides a visual representation of the existing media queries in a file and allows you to create new media queries and interact with them in a point-and-click interface.

4 The colors displayed indicate whether the media query is defined with min-width specifications, max-width specifications, or a combination of both.

5 The scrubber allows you to quickly preview the page design at varying screen sizes to test the predefined media queries and pertinent styling.

6 Bootstrap and other web frameworks use predefined CSS and scripting to provide built-in support for multiple screen sizes and mobile devices.

7 False. The six templates provide a quick way to jumpstart a Bootstrap design, but you can create your own Bootstrap layout from scratch at any time in Dreamweaver.

8 The `<div>` element is a generic container that passes no semantic information to search engines or other applications. Semantic elements like `header`, `nav`, `article`, and `aside` help search engines identify what type of content they contain.