

15

ADAPTING CONTENT TO RESPONSIVE DESIGN

Lesson overview

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

- Review content to identify styling issues based on screen size
- Create custom media queries to target styling to specific screen sizes.
- Create alternate Bootstrap column configurations
- Create custom table styling to support small screens
- Preview pages in various browsers and devices using Real-Time Preview



This lesson will take about 3 hours to complete. 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. Follow the instructions under “Accessing the Lesson Files and Web Edition.” Define a site based on the lesson15 folder.

Updating responsive design

Nothing in life is perfect. Although you have created a responsive template based on a powerful responsive web framework, you will still have issues with the basic design as well as the content you try to deploy. The purpose of this lesson is to walk you through the process of identifying these issues and to demonstrate some techniques for adapting the layout and content to various screens and devices.

In some cases, the issues you will see are intrinsic to the layout. Others will be based on specific content or components used within the layouts. And finally, some issues you will discover affect only a specific screen size or device. There is no way to find all the issues within this short lesson. Some will be discovered only after weeks of use and testing. Others may be reported by conscientious visitors or staff. But don't get complacent or lazy. It's easy to think that your HTML is fine because it works on your own computer and devices.

Testing and correcting errors is an ongoing and seemingly never-ending process. And what's worse is that HTML, CSS, JavaScript, and the Internet itself are constantly changing and evolving. What works today may not work next week or next year. The teams of developers working to improve the web always try to make their changes backward compatible, but that's not always possible or practical. There's only one thing you can be certain of: Things will break.

Inspecting the current site

The first step in troubleshooting is to put eyes on all the pages and review all the content. Yes, all of it. I can guarantee from personal experience that the first time you think that an item is okay and that you don't have to test it, it will be broken or display badly. And, worse, it will be your boss or client who finds the problem first.

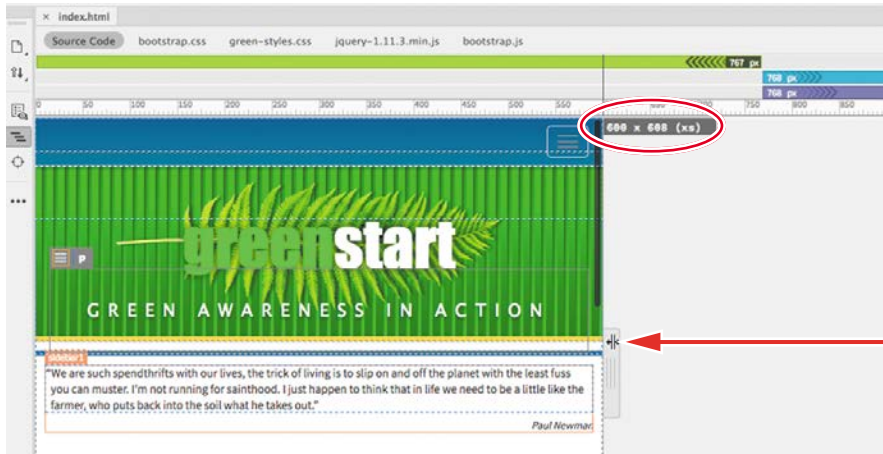
You must test every page and review them on multiple screen sizes and devices. I work on a Mac laptop, but I've installed Windows on my Mac and I even have a separate desktop with Windows installed on it. I own an iPhone and an iPad, but I went out and bought a used Android phone and tablet so I could test all my sites in both worlds. I try to test every page and every component in all those environments and in multiple browsers. But even with all that effort, you will still miss something. Be ready to jump on any report of a page or some content that is misbehaving.

Let's take a look at some techniques for reviewing your pages and content.

- 1 Open **index.html** in Live view from the lesson15 folder.
Make sure the document window is open to a minimum width of 1100 pixels.
- 2 Scroll down the page to observe the footer. Scroll back to the top of the page.
The homepage contains various kinds of text and an image. Scan the page carefully, looking over all the elements and styling. Try to note the spacing for

margins and padding, as well as line height and the spacing between paragraphs. If a page is longer than your screen height, make sure you scroll all the way to the bottom. When reviewing content, it's important to test the pages at different screen sizes too.

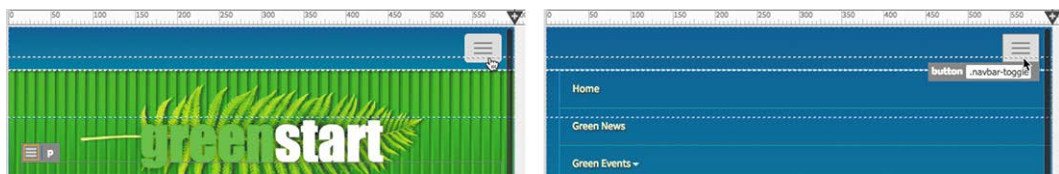
- 3 Drag the scrubber to the left to simulate the width of a 600-pixel wide screen.



Note how the three-column layout reacts to the narrowing screen. Pay close attention to the widths and balance of each column.

As the screen narrows, the columns resize to share the space evenly until the screen width reaches 768 pixels. As soon as the screen drops below 768 pixels, the three-column design converts to a one-column design, with all the content stacking vertically. At that moment, the horizontal menu switches from seven individual buttons to a solid blue bar with a sandwich icon appearing in the right corner.

- 4 Click the sandwich icon to open the menu.



When you click the sandwich icon, the navigation menu drops down, showing all seven links stack vertically. Note that the *Green Events* menu item shows an arrow icon, indicating that it hosts its own drop-down menu.

- 5 Position the cursor over each menu item and note the behavior of each item and the appearance of the cursor.

The menu items react with the hover behavior as the cursor passes over them.

● **Note:** The styling of the submenu items has changed between the full-screen layout and this smaller screen. You will address this styling issue later in this lesson.

- 6 Click the *Green Events* menu item.
The submenu opens, showing the two sublinks to the event calendar and class schedule. Note the styling of the sublinks.
- 7 Click the *Green Events* menu item a second time to close it.
- 8 Drag the scrubber all the way to the right side. Observe how the page reacts to the screen as it widens.

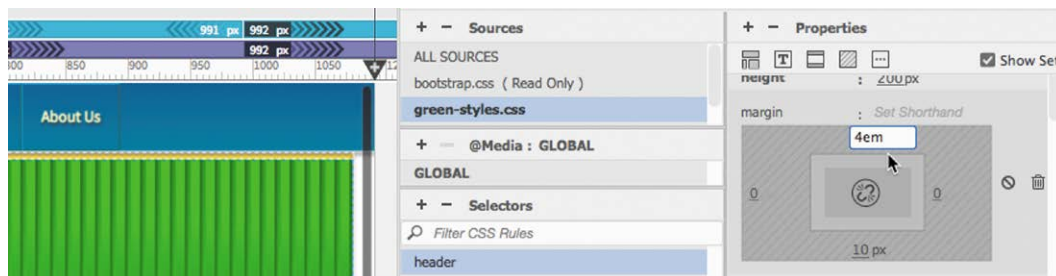
How many issues did you see in the basic layout or design? Did you catch them all? Let's take a look at the major issues and see how you can correct them.

Identifying site design issues

Unfortunately, the site did not pass the first test. The design has some major problems. But don't worry—all the issues observed are normal and expected in a new responsive site. Luckily, everything can be fixed pretty easily with some simple CSS tweaks.

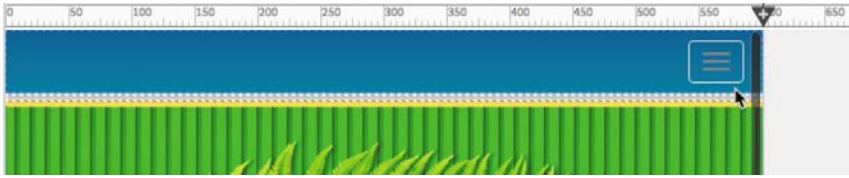
The first item you should have noticed is that the top border of the header element is hidden beneath the navigation menu. When you built the responsive menu in the previous lesson, it aligned perfectly. Something happened to it, and now the border is no longer visible. Things like this happen all the time when you are building pages, adding content, and styling it. You have to be constantly vigilant.

- 1 In CSS Designer, click the All button.
Select **green-styles.css** in the Sources panel.
The Selectors panel displays all the rules defined within the style sheet. The margin setting that controls the placement of the border is defined in the rule header.
- 2 Select the rule header.
Edit the following property: `margin-top: 4em`



This added spacing fixes the header display. The next issue we can correct is the gray border on the sandwich icon of the navigation menu when the screen is less than 768 pixels.

- 3 Drag the scrubber to a width of 600 pixels.

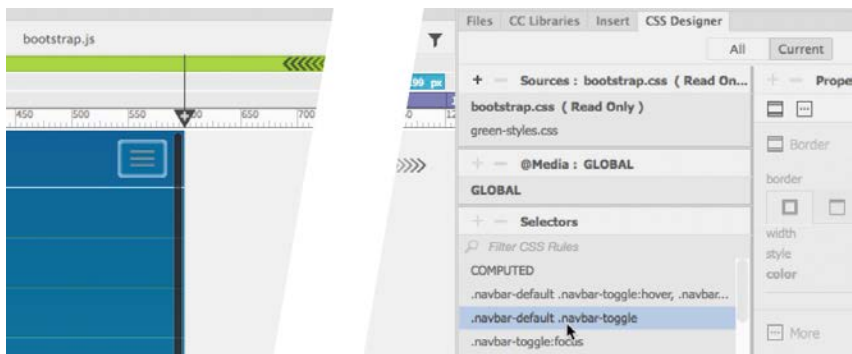


You can see that the border on the sandwich icon is gray. This doesn't follow the site color theme and looks a bit jarring amid the dark-blue background. Normally, you could use the Current button in the CSS Designer to test and identify components on the page, but the horizontal menu is part of the locked portion of the site template. As of this writing, the CSS Designer ignores elements in the locked areas. To track down the pertinent rules, you'll need to work with the template itself.

- 4 Open **mybootstrap_temp.dwt** in Live view from the lesson15 folder.

In the Dreamweaver templates, all elements are selectable and editable. You should be able to identify the specific rules formatting the sandwich icon and other components within editable regions.

- 5 Drag the scrubber to 600 pixels.
- 6 In the CSS Designer, click the Current button. Click the sandwich icon and observe the Selectors panel.

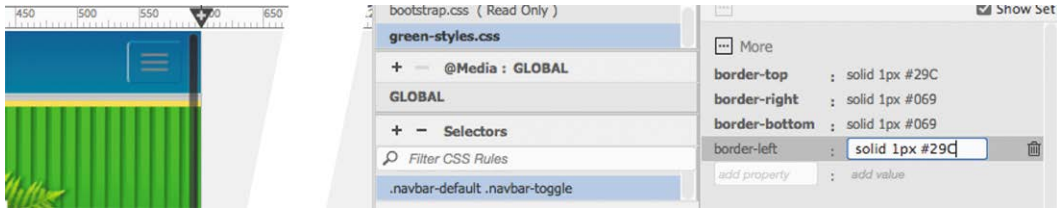


The Selectors panel lists the rules supplying some form of styling to the sandwich icon. Start at the top of the list and examine each until you find the rule, or rules, formatting the borders and other aspects of the sandwich icon. To override the border and other nonscheme colors, you need to create new matching rules in your custom style sheet.

- 7 Click the All button.
- Select **green-styles.css** and create the following rule:
- .navbar-default .navbar-toggle**

- 8 Add the following properties to the new rule:

border-top: solid 1px #29C
border-right: solid 1px #069
border-bottom: solid 1px #069
border-left: solid 1px #29C



This rule resets the borders around the outside of the icon. But you also need to reset the background-color of the element and even of the bars within the icon.

- 9 Create the following rule: **.navbar-default .navbar-toggle .icon-bar**

- 10 Add the following property: **background-color: #29C**



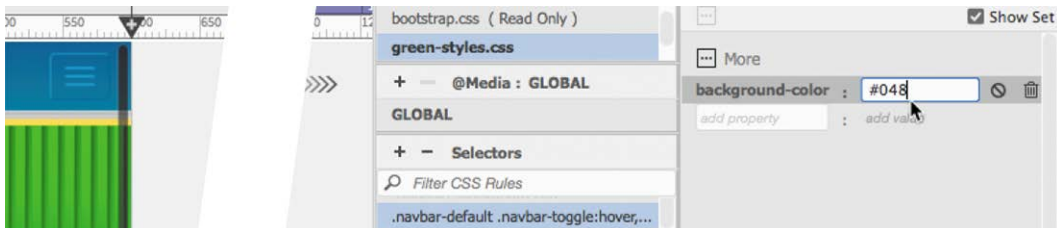
The three bars within the icon now comport with the site color theme.

● **Note:** Don't forget the comma between the two parts of the selector.

- 11 Create the following rule:
.navbar-default .navbar-toggle: hover,
.navbar-default .navbar-toggle: focus

This rule targets the `:hover` and `:focus` states of the sandwich icon.

- 12 Add the following property: **background-color: #048**



That fixes all the styling issues with the sandwich icon, but you may have also noticed that there is a gray border between the navbar and the menu items when it's open.

13 Create the following rule: `.navbar-default .navbar-collapse`

14 Add the following property: `border-top: solid 2px #048`



The changes in this exercise have brought the styling of the entire navigation menu within the site's color scheme. This corrects the color palette of the menu, but there's still a structural issue that you can see only when the drop-down menu opens.

Working with media queries

By default, the Bootstrap menu is designed to adapt to the width of any screen. But in the previous lesson, you centered and froze the new responsive menu at the top of the screen by using a fixed-width measurement. Since that specification was created as a global style, it affects the menu on all screens and devices. That means that when the screen is less than 768 pixels the menu items do not scale with the page, as they were originally intended to do. If you open the navigation menu when the screen is 767 pixels or smaller, you will see that the menu items do not touch the left edge of the menu interface. On very small screens, a scroll bar will appear below the menu items because the menu is much wider than the visible workspace.

There are several solutions to this the issue, but the simplest method is to change the way the width of the navigation menu is applied. Instead of applying the desktop width as a global specification, you will apply it based on the width of the screen. To apply CSS styles based on a specific screen width, you need to learn how to create a custom media query in your style sheet.

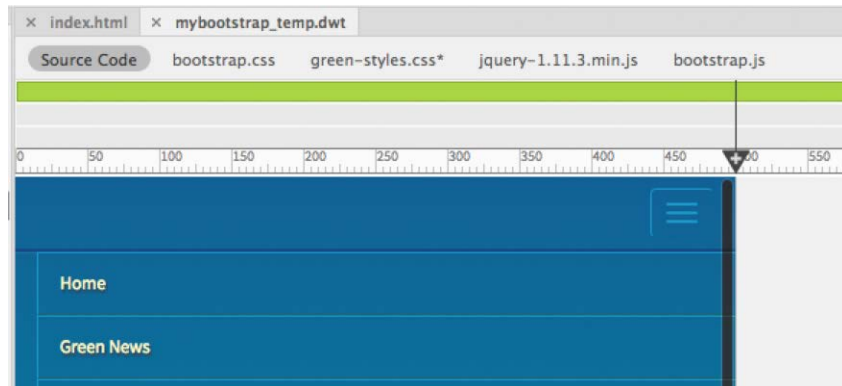
As you learned earlier, a media query is a tool you can use to target styling to specific screen sizes, orientations, and devices. In most cases, to deal with conflicts with your page content or with specific design requirements, you can simply add custom media queries and CSS rules to your own custom style sheet.

Creating a custom media query

Once you identify the dimension that needs some special handling, Dreamweaver makes it easy to create new custom media queries.

1 Open `mybootstrap_temp.dwt` in Live view, if necessary.

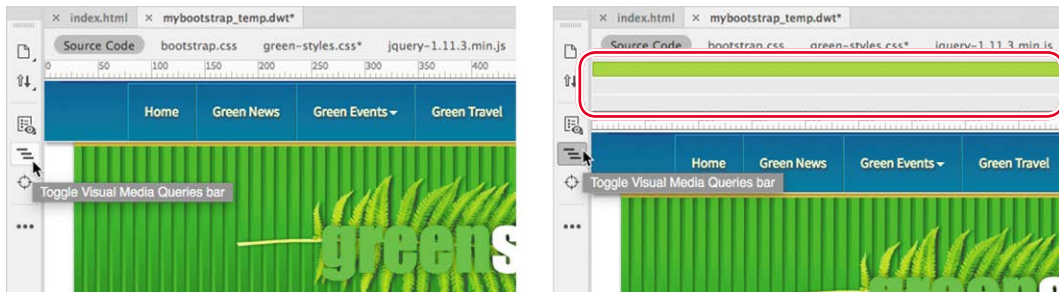
- 2 Drag the scrubber to 500 pixels.
Click to open the navigation drop-down menu.



The menu items are slightly indented from the left. You may also see a scroll bar displayed at the bottom of the menu. Because the menu is set to be wider than the current screen size, the menu items could accidentally scroll off the screen, making it difficult, if not impossible, to use. One way to fix this is by moving the rule that's applying the menu width into a custom media query that applies only at specific screen sizes.


To work with media queries, Dreamweaver provides the Visual Media Query (VMQ) interface.

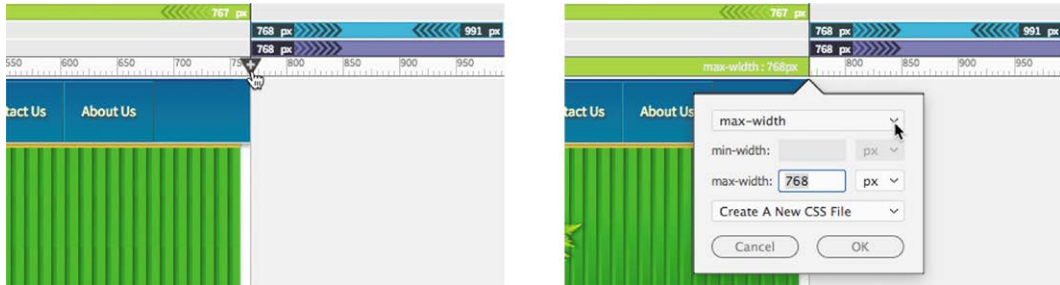
- 3 If necessary, click the Toggle Visual Media Query bar icon  in the Common toolbar to display the VMQ.



Depending on the width of your screen, the VMQ should display five media queries that are defined in the Bootstrap framework. Since the Bootstrap style sheet is locked, you won't be able to use any of the existing media queries. You will have to define your own.

You can use the scrubber to create a custom media query for a specific screen width.

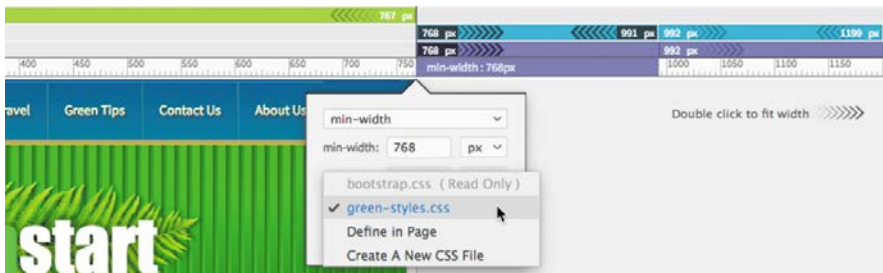
- 4 Drag the scrubber to 768 pixels.
Make sure the document window is at least 1100 pixels wide.
- 5 Click the Add Media Query icon  at the top of the scrubber.



A Media Query Definition dialog appears. The max-width field is already populated with the pixel position of your scrubber (768 pixels). The min-width field is grayed out. That means that if you use these specifications, this media query will apply to screens 768 pixels or smaller. This is not the intention of the new media query. Instead, you want to apply the fixed width to the navigation menu on screens 768 pixels or larger.

Note: The purpose of the media query is to apply styles only when the screen is 768 pixels or wider.

- 6 Open the first drop-down menu and select the min-width condition.
The ruler position should appear in the min-width field now. If the number is not 768, you can enter the correct position manually.
- 7 Select **green-styles.css** in the source pull-down menu.



- 8 Click OK to create the media query.
- 9 If necessary, open the CSS Designer and click the All button.
- 10 Click **green-styles.css** in the Sources panel.

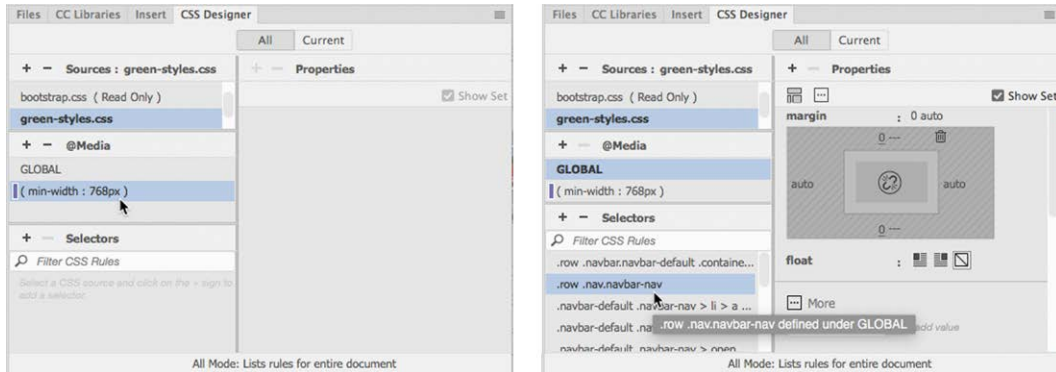
A new (min-width) media query has been added to the style sheet. You will use this media query to apply the fixed-width measurement to the navigation menu.

- 11 Select the (min-width: 768px) media query.
The media query is empty.

12 Select GLOBAL.

Locate the rule `.row .nav .navbar-nav`.

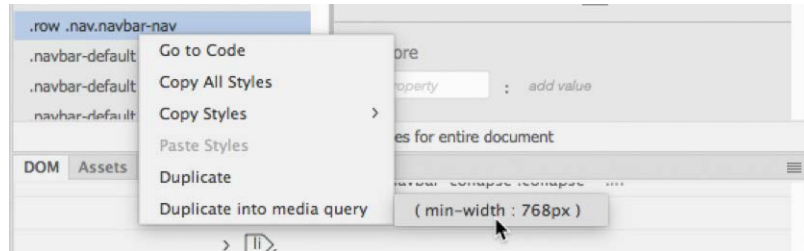
Examine the properties.



To ensure you are working on the correct item, always check the properties assigned by a rule before you edit or move it. The selected rule should feature the property `width: 660px`.

13 Right-click rule `.row .nav .navbar-nav`.

Select Duplicate Into Media Query > (min-width: 768px).



This command makes an exact duplicate of the rule in the new media query.


14 Select (min-width: 768px) in the @Media panel.

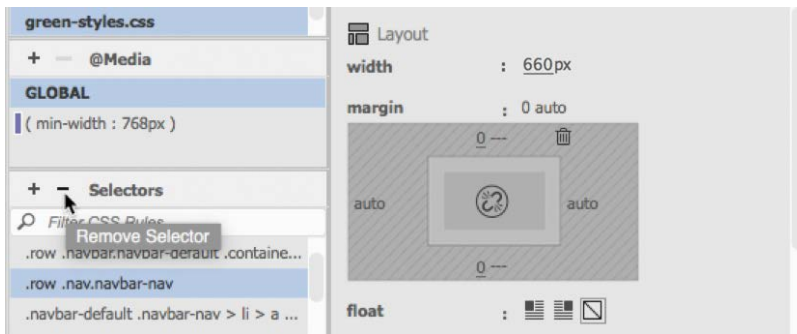
The rule `.row .nav .navbar-nav` now appears in the media query. At the moment, you have two copies of this rule. Before you delete the global version, you should examine all the properties defined within it to make sure that none of them are still needed as global attributes.

The rule defines width, margin, and float properties that are required only when the menu appears on screens 768 pixels or larger. You can freely delete the original version without harming the design.

15 Select GLOBAL in the @Media panel.

When this option is selected, only global rules are displayed in the Selectors panel.

- 16** Select the rule `.row .nav .navbar-nav` and click the Remove Selector icon  to delete it.



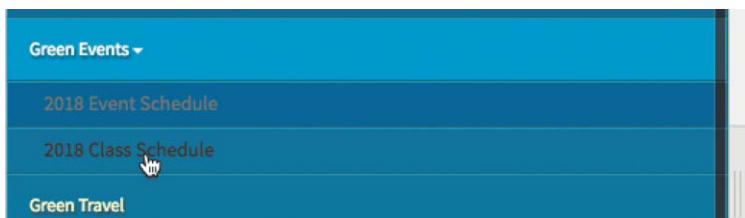
Once the global rule is deleted, the other version formats the menu only when the screen size is 768 pixels or larger.

- 17** Drag the scrubber back to 500 pixels. Click to open the menu.



The menu items are no longer indented, and no scroll bar is visible. The new media query seems to be working. The menu seems to be working now, but it's always a good idea to test all aspects of the menu. The *Green Events* menu item still features a submenu with links pointing to the event calendar and class schedule.

- 18** Click to open the submenu.




The styling of the submenu is different at this screen width. The text is formatted as a dark gray, not pale yellow. Web frameworks often supply unique styling at different screen sizes. That's why it's so important to test all aspects of your website before you let the public in. Let's correct the styling issue. In this exercise, you learned how to apply styles based on `min-width` specifications; in the next exercise, you will learn how to apply styles using a `max-width` setting.

Creating a max-width media query

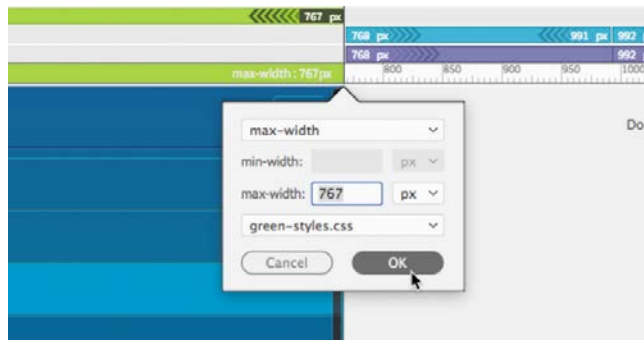
► **Tip:** Instead of dragging the scrubber, you can also click the green media query to set the window size.

The drop-down menu is activated at 767 pixels and below. In this exercise, you will fix the text styling by creating a new custom media query for screens smaller than 768 pixels.

- 1 Drag the scrubber to 767 pixels.
Click the Add Media Query icon  at the top of the scrubber.

The Media Query Definition dialog appears. The `max-width` field is populated with the current scrubber position (767 px).

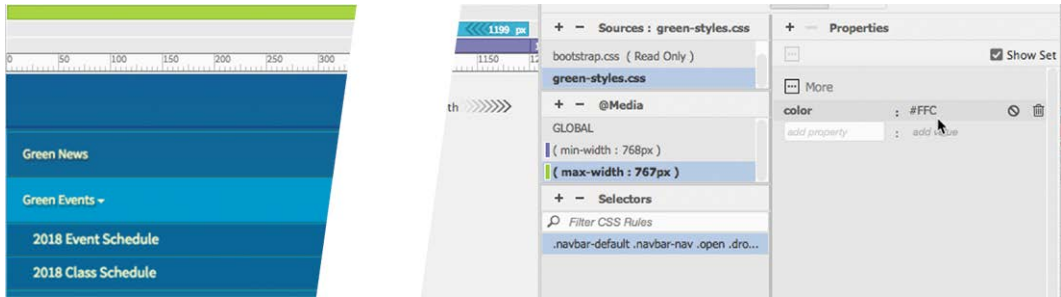
- 2 Select **green-styles.css**, if necessary, in the Source drop-down menu.
Click OK to create the new media query.



- 3 Select **green-styles.css** in the Sources panel.
Select (max-width: 767px) in the @Media panel.
Create the following rule:

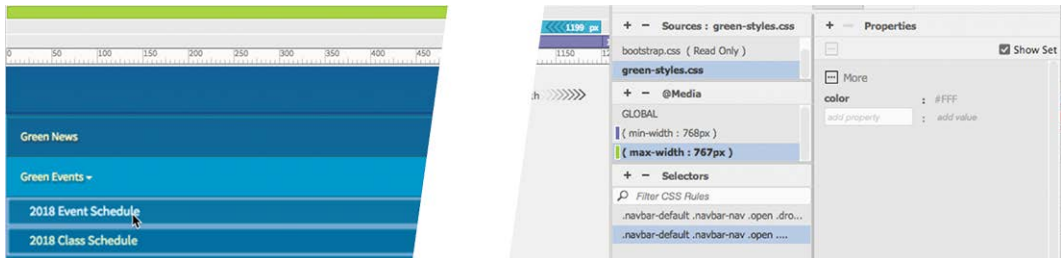
```
.navbar-default .navbar-nav .open .dropdown-menu >
li >a:link , .navbar-default .navbar-nav .open
.dropdown-menu > li >a:visited
```

- 4 Create the following property: **color: #FFC**



- 5 Create the following rule in media query (max-width: 767px):
.navbar-default .navbar-nav .open .dropdown-menu > li > a: hover, .navbar-default .navbar-nav .open .dropdown-menu > li > a: focus

- 6 Create the following property: **color: #FFF**



Test the submenu. When open, the links should now be pale yellow by default. Then they turn white when you position the cursor over them.

- 7 Save all files.
- 8 If necessary, click to close the navigation menu.
Review the rest of the layout to identify any other layout issues.



At 500 pixels in width, the content is displayed in a single column. At this size, the company name and logo are very large and the motto is breaking into two lines. The second line is displayed in the middle of the quotation.


When the screen gets too small, the styling of the header is not scaling down to fit the available space. You need to identify the screen size where the content starts to break and create additional custom rules in the appropriate media query to fix it.

Adapting the header to smaller screens

If an existing media query cannot serve the purpose of applying the needed styling, you will often need to create additional media queries and rules. In this exercise, you will create another media query to adapt the header to smaller screens.

- 1 Open **mybootstrap_temp.dwt** in Live view, if necessary.
- 2 Drag the scrubber left and right to identify the exact screen width where the motto breaks to two lines.

The text breaks around 523 pixels or so. The exact size will depend on many factors, including, but not limited to, operating system, screen type, browser type, and version. In other words, don't count on whatever number you see on your system at the moment. Always add some extra space into consideration so that your brand-new media query doesn't break right out of the box.

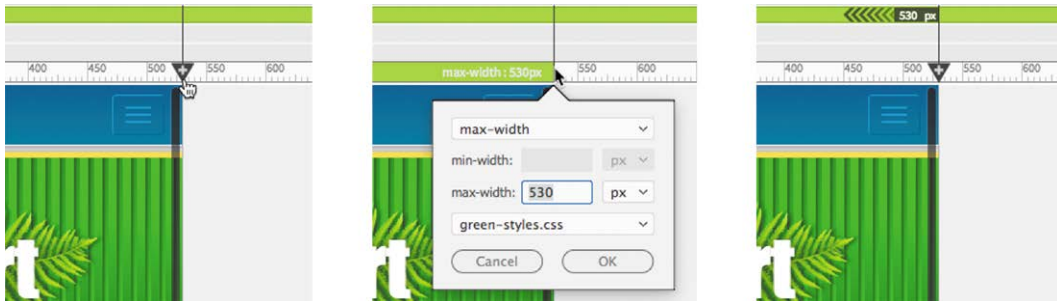
- 3 Drag the scrubber to 530 pixels.
Click the Add Media Query icon  at the top of the scrubber.

The Media Query Definition dialog appears. The **max-width** field is populated with current scrubber position (530 px). The min-width field is grayed out and **green-styles.css** is already selected in the Source menu. The new media query will apply its styles only when the screen is 530 pixels or narrower.

● **Note:** If **green-styles.css** does not appear in the pull-down menu automatically, select it manually.

- 4 Click OK to create the media query.

● **Note:** The purpose of the media query is to keep the motto on one line. Your pixel position may differ from the one described here or pictured in subsequent screen shots. Substitute your measurement in the following exercises, as necessary.



A new media query appears in the VMQ. The query is active immediately, but at this width, the motto is currently displayed on one line. It would help to set the scrubber at the smallest screen size you need to support before making any new rules.

- 5 Drag the scrubber to 320 pixels.

The width of 320 pixels is the size of the original iPhone and should be the smallest device you have to worry about. Notice that the logo extends off the edges of the screen and that the motto is showing a single word. Let's deal with the motto first.

- 6 Click the motto in Live view. Select the `p` tag selector.

When you're creating new rules in a media query, there is a specific way to do it to make sure it gets added to the right place in the style sheet. The CSS Designer is integrated closely with the document window and the items selected therein. If you use the following method, your rules will always be added properly.

- 7 Click the All button in the CSS Designer.
Select **green-styles.css** in the Sources pane.
Select `(max-width: 530px)` in the @Media pane.
Click the Add Selector icon **+**.

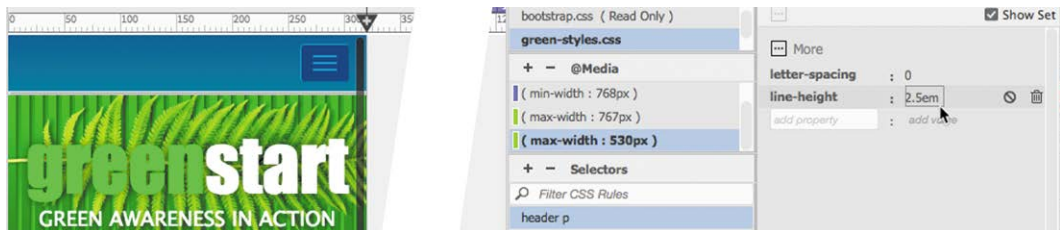
A new selector, `.row .col-sm-12 p`, is created in the Selector panel. It is based on the Bootstrap structure and does not match the original rule you created. To reset styling, it is essential that the selectors be identical.

- 8 Create the following name: **header p**

- 9 Create the following properties:

letter-spacing: 0

line-height: 2.5em



The motto now appears on one line. The new styling will work on screens smaller than 531 pixels.

- 10 Drag the scrubber to 600 pixels.

The motto shows no letter spacing until the scrubber passes 530 pixels, and then it returns to its original styling. Next you will adjust the size of the logo.

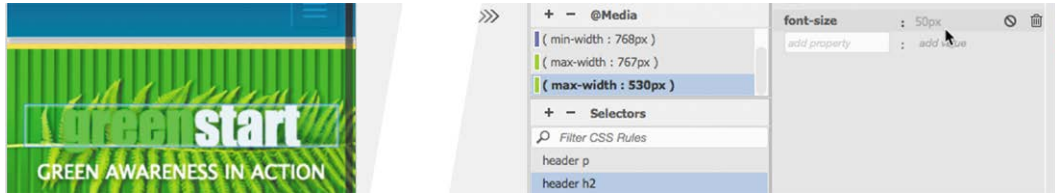
- 11 Drag the scrubber to 320 pixels.

The logo is composed of the word *greenstart* and the fern image. You will have to reset the rules for both elements.

- 12 Select **green-styles.css** > (max-width: 530px) in the CSS Designer.

Create the following selector: **header h2**

Create the following property: **font-size: 45px**

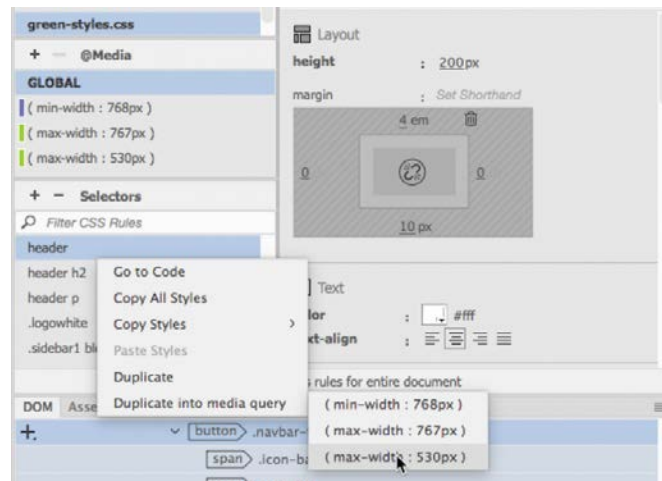


The heading reduces in size to fit the space better. The fern image is part of a complex background specification. The best way to modify a specification like this is to duplicate the entire rule into the media query and edit the appropriate settings.

- 13 Select **green-styles.css** > GLOBAL in the CSS Designer.

- 14 Right-click the rule header.

Select Duplicate Into Media Query > (max-width: 530px) from the context menu.



The entire rule is duplicated in the new media query. Since the background specification is so complex, the best place to edit it is in Code view.

- 15 Select **green-styles.css** > (max-width: 530px) in the CSS Designer. Right-click the rule header and select Code To Code from the context menu.

The document window switches to Split view, if necessary, and loads **green-styles.css** focused on the header rule. Since all the styles are coming from a global rule, you need to keep only the specifications that will reset portions of the background. The goal of every web designer should be to minimize code whenever possible. You should delete any specification that is unneeded.

- 16** Delete the following properties:

```
text-align: center;
background-color: #999;
color: #fff;
border-bottom: 5px solid #FD5;
border-top: 5px solid #FD5;
margin-bottom: 10px;
background-image: url(images/fern.png) , url(images/stripes.png) , -webkit-linear-gradient(270deg, rgba(0,153,0,1.00) 0%, rgba(0,204,0,1.00) 100%);
webkit-box-shadow: 0px 0px 3px 0px rgba(0, 0, 0, 0.55);
box-shadow: 0px 0px 3px 0px rgba(0, 0, 0, 0.55);
background-repeat: no-repeat , repeat-x;
background-position: 45% center, 0 0;
margin-top: 4em;
```

When these properties are deleted, you should see no changes in the header display.

- 17** Edit the following properties as highlighted:

```
height: 150px;
background-size: 85% auto, auto auto, auto;
```



These changes have greatly reduced the header footprint, which is appropriate for smartphones and other mobile devices. It's also a good idea to reduce the size of the regular content.

- 18** Select **green-styles.css > (max-width: 530px)** and create the following selector:

```
main.row section h1
```

19 Create the following property:

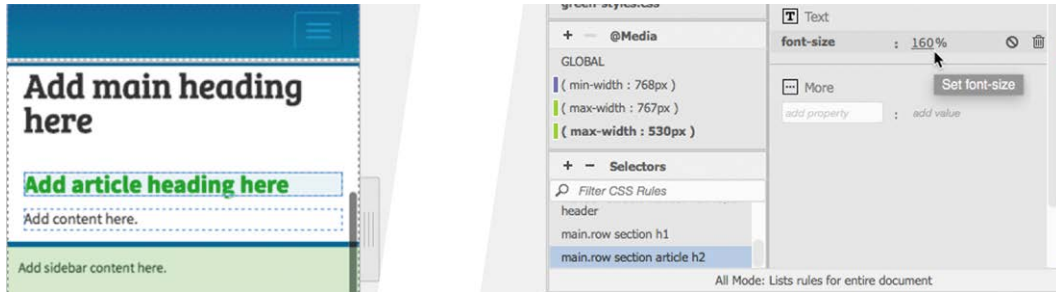
font-size: 225%

20 In **green-styles.css** > (max-width: 530px), create the following selector:

main.row section article h2

21 Create the following property:

font-size: 160%



Once you have updated the CSS, you should test the styling.

● **Note:** If the Update Template Files dialog appears, go ahead and click the Update button.

22 Save all files. Switch to Live view.

23 Drag the scrubber to the right and left and observe how the content adapts to the screen width.

The text and background effects in the header and the headings in the main content change sizes seamlessly as you increase and decrease the width of the document window. Feel free to adjust any of the specifications to meet your own tastes.

24 Close all files.

This should take care of the styling for the basic content on this page for smaller screens. There are still other pages you need to review.

Adapting content to responsive design

There are seven pages in the GreenStart site. Each page contains text, images, and other types of HTML components. Converting the layout to a responsive design was only your first step. In the next few exercises, you will open each of the pages and address various issues to adapt the content to the new responsive design.

Re-creating float and indentation styles

On the *Contact Us* page, you created custom styling to highlight and indent the staff profiles. In this exercise, you will review the *profile* elements at multiple screen sizes and adapt them as needed.

1 Open **contact-us.html** from the lesson15 folder in Live view.
Make sure the document window is at least 1100 pixels in width.

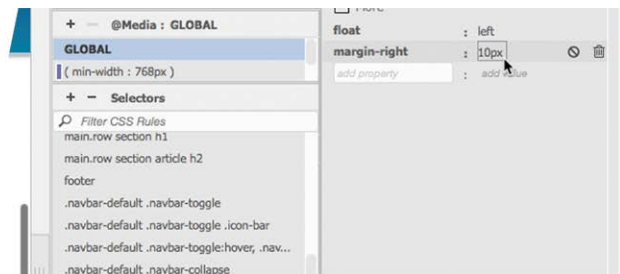
2 Examine the page and content at full width.

Do you see any immediate issues? Since we're not using the GreenStart style sheet, the classes `.flt-lft`, `.flt-rgt`, and `.profile` do not exist. That means that the text in each profile is not wrapping around the staff photo. The first step is to re-create these styles.

3 Select the All button in the CSS Designer.
Select **green-styles.css** > GLOBAL.

4 Create a new selector: **.flt-lft**

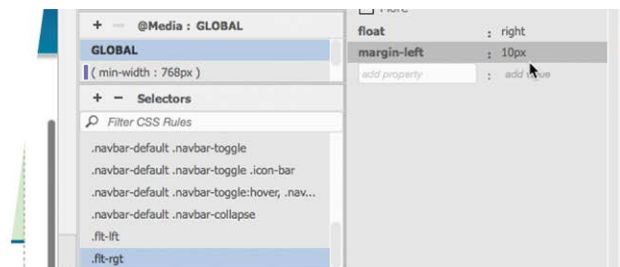
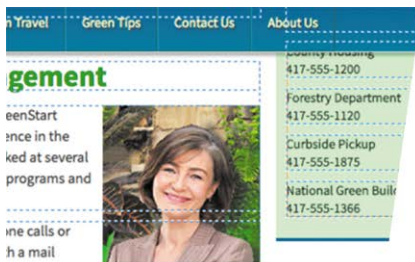
5 Create the following properties:
float: left
margin-right: 10px



As soon as you finish the properties, the images styled with the class `.flt-lft` appear correctly formatted.

6 Select **green-styles.css** > GLOBAL.
Create a new selector: **.flt-rgt**

7 Create the following properties:
float: right
margin-left: 10px



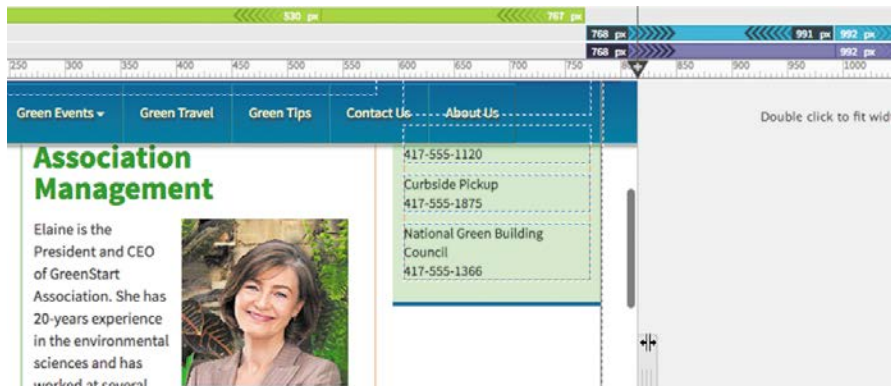
The remaining images appear correctly formatted. The next step is to create the `.profile` rule.

- 8 Select **green-styles.css** > GLOBAL.
Create a new selector: **.profile**
Create the following properties:
margin: 0 25px 15px 25px
padding-left: 10px
border-left: solid 2px #BDA
border-bottom: solid 10px #BDA



The original styling is complete and displayed in the layout. Let's test the formatting to see how it responds to the new responsive layout.

- 9 Save all files.
- 10 Drag the scrubber to the left to make the document window narrower. Test the layout down to a width of 320 pixels. Observe how the profile structure responds to the changing width.



The `.profile` section looks fine until you get down to widths between 768 and 991 pixels. In that range the profile sections start to seem very cramped, with the text wrapping around the images. At 767 pixels, the content stacks in one column. It looks fine down to a width of 400 pixels. Below that, the left indent and border waste too much space. The interaction of the floated images and indented text with the changing width would best be fixed by first adjusting the basic layout. For the smallest screens, you'll provide alternate styling for the indents and floated images.

Creating alternate Bootstrap layouts

As the layout adapts to smaller screens, you may have noticed that the columns scale down to share the available space. At a certain width, the layout switches, or breaks, from three columns into a single column. This *breakpoint* is specified in the Bootstrap settings established when you first created the basic layout in Lesson 13, “Designing for Mobile Devices.”

Unfortunately, the three-column design, which looks fine at full size, doesn’t work very well at the intermediate sizes between 768 and 991 pixels. In this range, the layout would be better split into two columns instead of three. Luckily, Bootstrap makes it easy to create alternate layouts by simply changing some of the classes assigned to the layout elements.

In this exercise, you will create an alternate two-column layout by editing the existing classes and adding new classes to the Bootstrap structural elements.

- 1 Open **contact-us.html** from the lesson15 in Live view.

Make sure the document window is at least 1100 pixels in width.

The underlying Bootstrap framework features five basic break points: extra-small, small, medium, large, and extra-large. At the moment, the layout breaks only at a width of 767 pixels (extra-small).

If you look at the code, you will see the classes assigned to the basic column elements. Each class is typically broken into three parts, such as `col-sm-6`. The part “col” refers to “column,” the part “sm” refers to “small,” and the part “6” refers to the number of grid sections. That means, in the current layout, the class `col-sm-6` causes the element to span six grid divisions when the screen is 768 pixels or larger.

To change the layout from three to two columns, you first need to change the existing classes. The classes you need to change are outside the editable regions, so you will have to make the changes in the template.

- 2 Open **mybootstrap_temp.dwt** in Live view.

Make sure the document window is at least 1100 pixels in width.

You want to change the layout from three columns to two columns between 768 and 991 pixels. It’s always a good idea to set the workspace to that environment so that you can see the changes visually.

- 3 Drag the scrubber to 900 pixels.

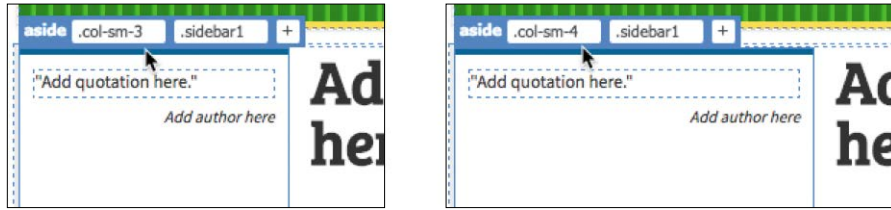
The scrubber is positioned within the small breakpoint.

- 4 Click the quotation placeholder.

Select the `aside.col-sm-3.sidebar1` tag selector.

The Element Display appears focused on the `<aside>` element.

- 5 Edit the class as highlighted: `.col-sm-4`



This change causes the first column to span four grid divisions. Sidebar 2 doesn't have enough space to appear on the same line and drops down below the first two columns. Now you will style the second column to use the remaining space.

- 6 Click the heading *Add main heading here*.
Select the `section.col-sm-6` tag selector.

- 7 Edit the class as highlighted: `.col-sm-8`



The main content area now occupies the remaining space, creating a two-column layout. Sidebar 2 appears below the two columns, but it is aligned to the left. A preferable design would be to move it under the main content and allow it to fill the second column. Bootstrap enables you to push and pull columns using another class.

- 8 Click in Sidebar 2. Select the `.aside.col-sm-3.sidebar2` tag selector.
9 Edit the class name as highlighted: `.col-sm-8`
10 Click the Add Class/ID icon `+`.
Type `.col-sm-push-4` and press Enter/Return to complete the new class.



The class shifts Sidebar 2 under the main content area. The two-column layout is complete, but you now have to re-create the three-column layout for larger screens.

- 11** Drag the scrubber to fully open the document window.

When fully open, the two-column layout scales to use the entire width of the screen. The classes you changed format the small breakpoint. If no other classes exist, the larger breakpoints simply use the existing styles. To restore the three-column design, you'll need to add new classes to the same elements. In this case, you'll restore the three-column layout for the medium breakpoint.

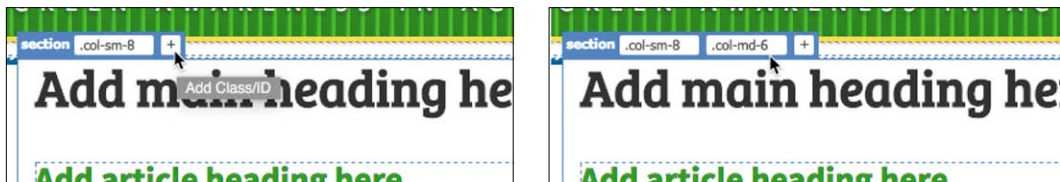
- 12** Click the quotation placeholder.
Select the `aside.col-sm-4.sidebar1` tag selector.

- 13** Click the Add Class/ID icon .
Type `.col-md-3` and press Enter/Return to complete the new class.



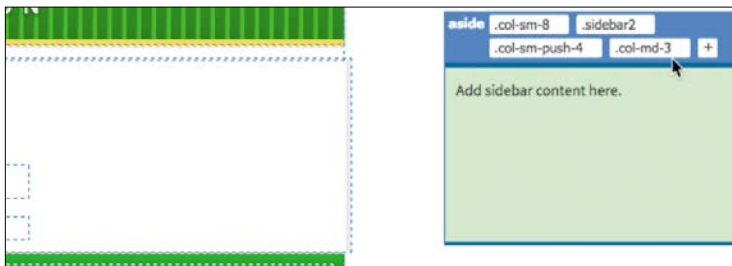
The new class reapplies the original width to the first column. Next you'll restore the second column to the original width.

- 14** Click the *Add main heading here* placeholder.
Select the `section.col-sm-8` tag selector.
- 15** Add the following class to the element: `.col-md-6`



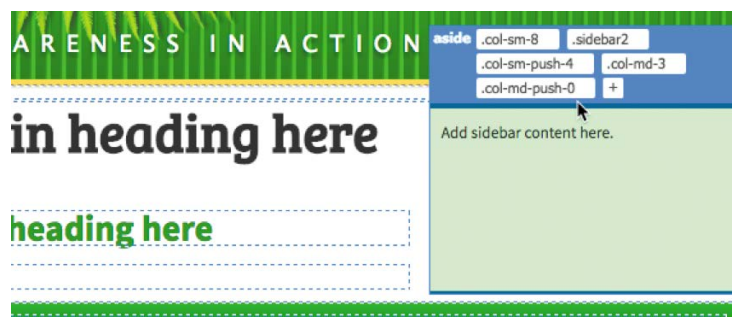
The main content area now returns to its previous size, leaving an open spot for Sidebar 2.

- 16** Click in Sidebar 2.
Select the `aside.col-sm-8.sidebar2.col-sm-push-4` tag selector.
Add the following class to the element: `.col-md-3`



The class restores the width of Sidebar 2, but the element doesn't return to its original position. It appears outside the layout on the right side. Since you *pushed* the element to align it to the second column, that class is still being applied. To reset the position of Sidebar 2, you have to apply another class to cancel out that styling.

- 17** Add the following class to the element: `.col-md-push-0`



Sidebar 2 moves into its expected position in the layout. Once the classes are applied, you should test the styling again.

- 18** Drag the scrubber to 320 pixels.

Observe the layout as it adapts to the width of the document window.

The layout converts from three columns to two columns to one as the width drops to 320 pixels.

- 19** Drag the scrubber to open the document window fully.

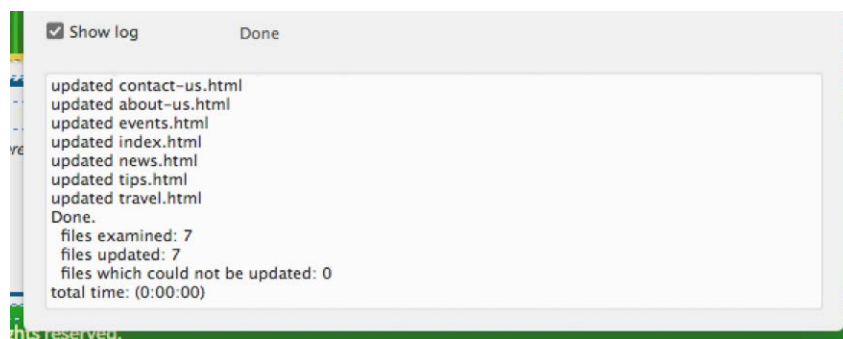
The layout converts from one to two to three columns as the window opens. Once you are certain the styling is working successfully, you can apply it to all the pages in the site.

- 20** Save the template.

The Update Template Files dialog appears.

- 21** Click Update to update all child pages.

All pages in the site are updated with the new classes and design scheme.



22 Close the template.

When the template closes, the *Contact Us* page remains open. The asterisk in the document tab indicates that the layout was updated. Whenever you make global changes to the site, you should always review and test every page.

Adapting custom indents to responsive design

The *Contact Us* page now has the new layout scheme applied to it. Let's take a look now at how the new layout styling works with the staff profiles.

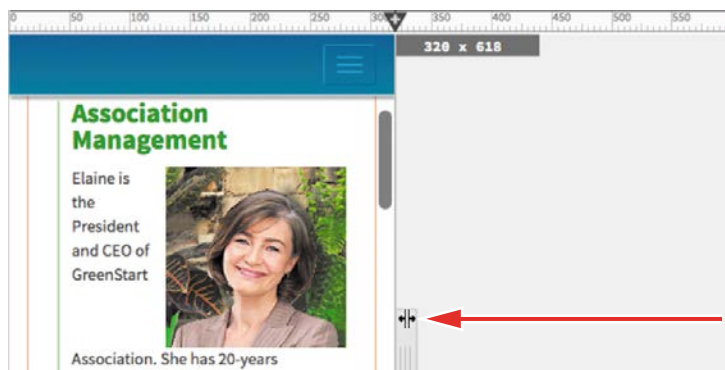
1 Drag the scrubber to 900 pixels.

The layout changes from three to two columns. The content fits nicely in the new layout.

2 Drag the scrubber to 500 pixels.

The layout changes from two columns to one column. The content looks fine when the document window is at 500 pixels, but how will it handle smaller screen sizes?

3 Drag the scrubber to 320 pixels.



At 320 pixels, the layout is too cramped. The text is not wrapping properly around the images. The borders and indents are using up valuable space and no longer enhancing the content. Let's adjust the layout for the smallest screens.

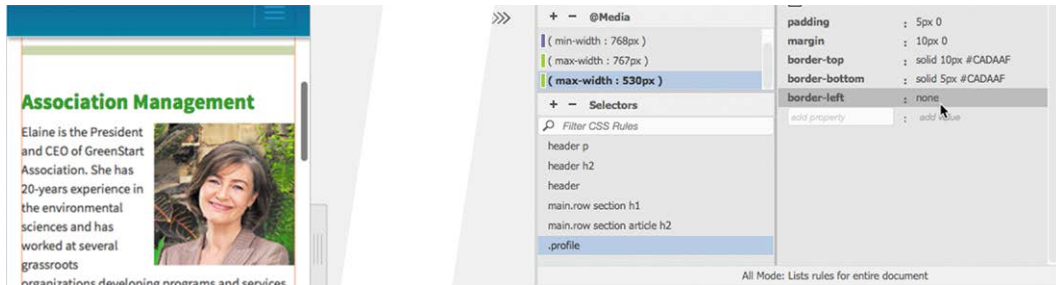
4 In the CSS Designer, click the All button.

Choose **green-styles.css** > (max-width: 530px).

Create the following selector: **.profile**

- 5 Create the following properties in the new rule:

padding: 5px 0
margin: 10px 0
border-top: solid 10px #CADAAF
border-bottom: solid 5px #CADAAF
border-left: none



Below 530 pixels, the `.profile` section now expands nearly to the full width of the screen and drops the indents and the left border. There's a bit too much space above the heading in each profile. Luckily, you created a new selector for this element earlier.

- 6 Select **green-styles.css** > (max-width: 530px) > `main.row section article h2`.
Add the following property: **margin:** .5em 0



● **Note:** You may need to click the Refresh button to see the changes in the layout properly.

- 7 Test the new styles by dragging the scrubber left and right.

The new styling for the profiles works perfectly and looks good, too. Remember to test all new components at every screen size and orientation and make changes to the styling as needed.

- 8 Save and close all files.

The next items you have to review are the tables created for the events and class calendars and the one used on the travel page.

Adapting tables to a Bootstrap layout

Before you tackle the concept of making the tables responsive, let's review the existing tables and see how they fared moving to the new Bootstrap layout.

Moving CSS rules between style sheets

In this exercise, you will identify the rules needed for styling HTML tables in one style sheet and move them into another.

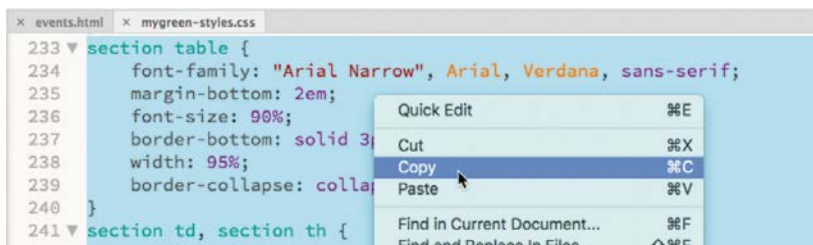
- 1 Open **events.html** in Live view.
Make sure the document window is at least 1100 pixels in width.



Date	Event	Location	Cost
Dec 31, 2017	New Year's Eve Party	West Side Community Center	\$25.00
Jan 20, 2018	Community Potluck	West Side Community Center	Free

As soon as the page opens, you can see that all the CSS styling was lost in the transition to responsive design. The tables still bear all the content and CSS classes, but the new style sheet has none of the specifications that you created earlier. Luckily, those rules are all still available in **mygreen-styles.css**. Instead of creating all those rules again, let's just copy and paste them into the new style sheet.

- 2 Open **mygreen-styles.css**.
The file contains all the styles from the original static GreenStart website. The table styles were created in the same lesson, one after the other. That means they should all appear consecutively in the style sheet.
- 3 Scroll down through the style sheet to locate the first table style.
Around line 233 you will find the rule **section table**. You will move all the table rules to the new style sheet.
- 4 Select the CSS markup from **section table** down to the rule **table caption**.
Right-click the selected code and select **Copy**, or press **Ctrl+C**/**Cmd+C**.



Be sure you include the properties for the **table caption** rule.

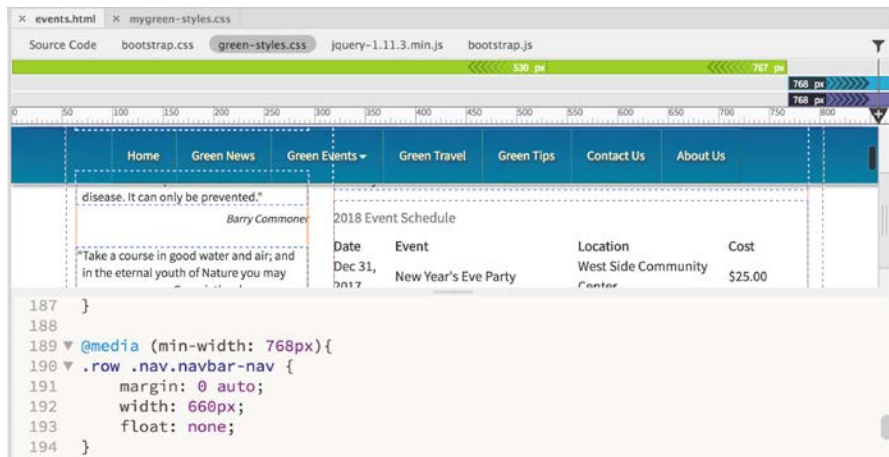
5 Switch to **events.html**.

You don't need to open **green-styles.css** if you have one of the webpages that is linked to it already open.

6 Select **green-styles.css** in the Related Files interface.

The document window switches to Split view and loads **green-styles.css** in the Code view window. It is very important to insert the styles in the proper spot of the style sheet.

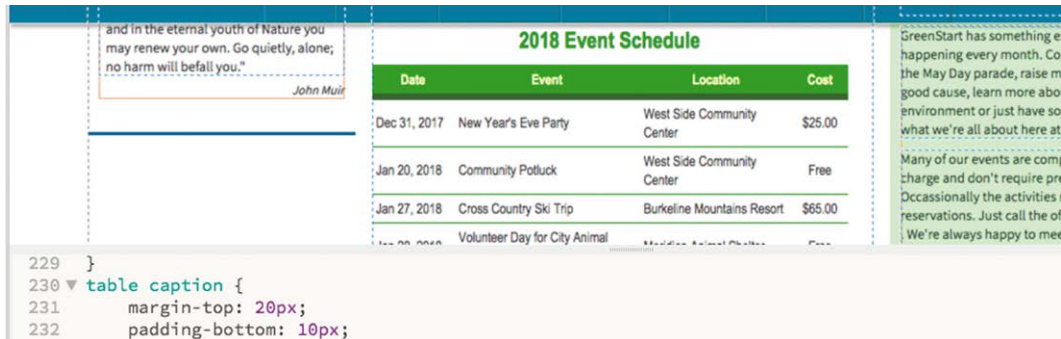
7 In the Code view window, scroll down to the entry `@media (min-width: 768px)` (around line 189).



This is the first custom media query you created earlier. It is essential that you insert the CSS you copied *before* this media query. You will see an opening curly brace { after this entry. All the rules for the media query appear after this opening brace and before the closing curly brace }, which appears, in this case, around line 194.

The table styling is considered global styling. It is automatically applied to and inherited by the tables. The styles in the media queries are designed to override the global styles. But that means the global styles have to appear in the style sheet before any media queries. Global rules inserted accidentally after the media queries could actually cancel out the styles contained within the media queries.

- 8 Insert the cursor before this entry and press Ctrl+V/Cmd+V to paste the CSS markup. Press Enter/Return to move the entry @media (min-width: 768px) to a new line after the markup you just pasted, if necessary.



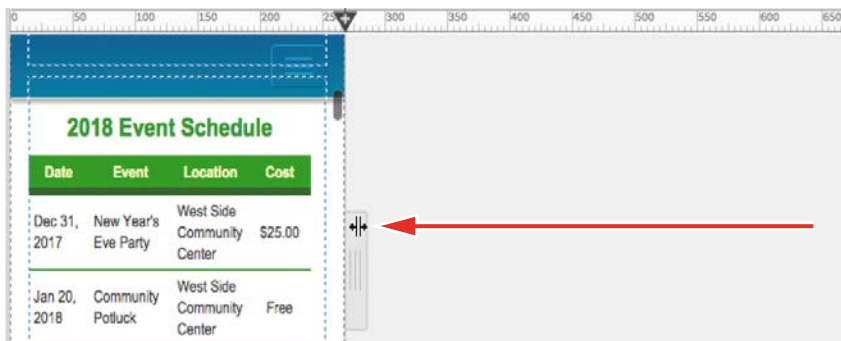
- 9 Save all files.

There's no functional need to move the media query entry to a separate line. It just makes the code easier to read and edit. As soon as you paste the CSS you should see that the tables are now formatted as you left them in Lesson 7, "Working with Text, Lists, and Tables." Once the tables are styled again, you can start adapting them to the new layout.

Making tables responsive

Tables are probably the last HTML element you'd think of when you think of responsive design. Tables are notoriously ill suited to smaller screens because they don't naturally adapt to them. But using some new CSS tricks, you'll learn how even tables can be made to be responsive.

- 1 Open **events.html**, if necessary. Switch to Live view. Make sure the document window is at least 1100 pixels in width.
- 2 Drag the scrubber to the left. Watch carefully how the tables respond, or don't respond, to the changing window size.



As the screen becomes narrower, the media queries kick in and reformat the page and components to adapt to the smaller screen. Since the table widths are set to 95%, they mostly scale down with the page. But once the screen width drops below 500 pixels, the text within the table becomes very cramped.

We need to rethink the whole concept of table design and display. You need to change the basic nature of the elements that compose tables so you can display them in a completely different way. During this process, you may sometimes find it easier to work in the CSS Designer; at other times you may want to enter the settings directly in Code view. Feel free to use whichever method feels more comfortable to you.

- 3 Drag the scrubber to 400 pixels.

All the changes will be applied only to the smallest screen sizes.

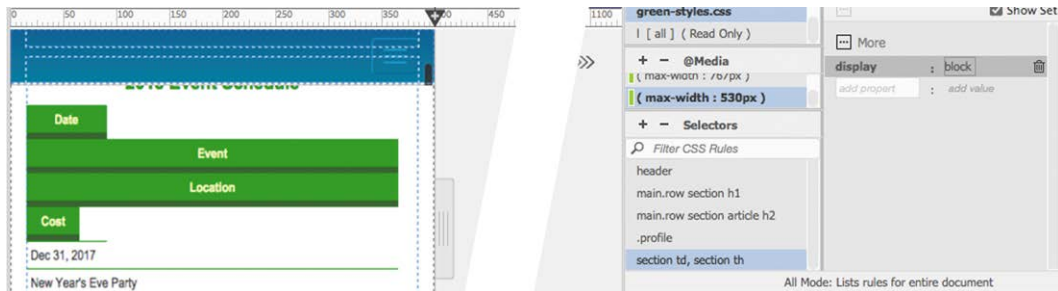
- 4 Open the CSS Designer, if necessary.
Select **green-styles.css** > (max-width: 530px).

The basic element of the table is the cell, or `td`, element. Cells default to inline block styling. The first step is to alter their basic nature.

- 5 Create the following selector: **section td, section th**

Table headers, `<th>`, are basically the same as table cells. You will format both the same way at first.

- 6 Add the **display: block** property to the new rule.



The table cells are now displayed vertically, stacking one atop the other when the width of the document window is narrower than 530 pixels.

This rule resets the default behavior of the table elements so that you can control their appearance on smaller screens. Some cells appear narrower than others because formatting is still being inherited from other parts of the style sheet. You'll have to create additional rules to override these specifications.

- 7 Create the following rule: **section table**
Create the following property: **margin: 10px auto**

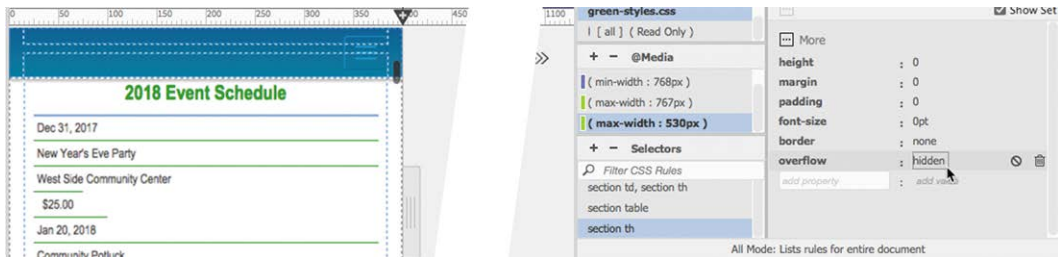


The tables are now centered in the layout.

With the data stacking vertically it doesn't make much sense now to have a header row. You could set the header row property to `display:none` to hide it, but that's not recommended for accessibility standards. The next best thing would be to simply format it to take up no space.

Note: Be sure that all subsequent rules and properties are added only to the custom media query.

- 8 Create the following rule: **section th**
Create the following properties:
- height: 0**
 - margin: 0**
 - padding: 0**
 - font-size: 0pt**
 - border: none**
 - overflow: hidden**

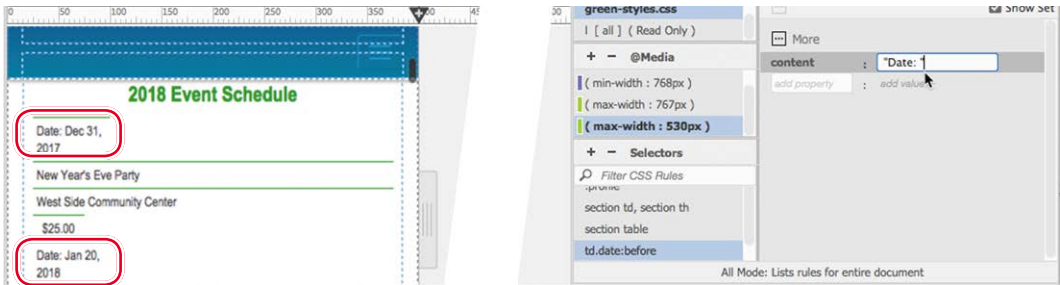


The header rows disappear visually but are still accessible to visitors using screenreaders or other assistive devices. But now that they are invisible, you have to address the fact that there are no headers describing the data being displayed.

For this purpose, you'll resort to a new CSS3 property that can actually create labels based on the CSS class applied to the cell. Some of the latest CSS3 properties are not directly available in the CSS Designer, but you can enter them manually in the Properties window or in Code view, and Dreamweaver may provide hinting support for them as well.

● **Note:** This type of selector is called a pseudo-class and is related to the classes you created for link behaviors.

- 9 Create the following rule: **td.date:before**
- 10 Enable the Show Set option.
Enter the following property–value combo: **content: "Date: "**
- **Note:** Make sure you add a space after the colon in the label. This will ensure that there is a space between the label and the cell content.



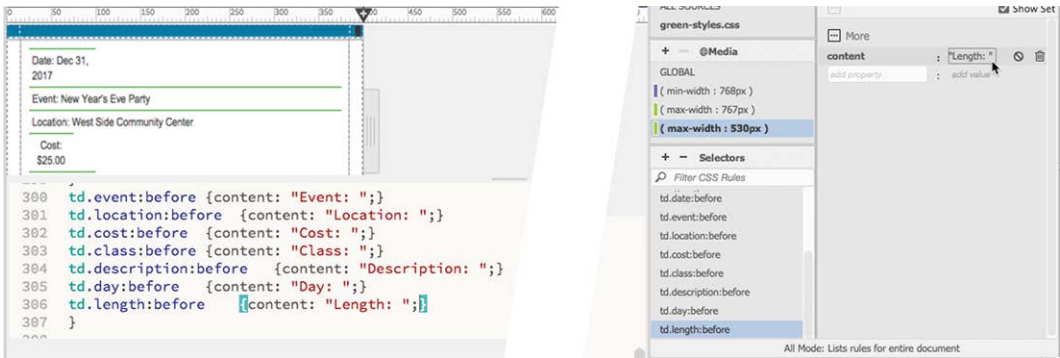
● **Note:** Be sure that all subsequent rules and properties are added only to the custom media query.

▶ **Tip:** Creating long selectors may be easier to do in Code view. You can access the **green-styles.css** file by clicking its name in the Related file interface at the top of the document window.

Notice that the label Date: appears in all the cells styled by the date class. You need to make a similar rule for each of the data elements.

- 11 Repeat steps 9 and 10 to create the following rules and properties:

Rule Name	Property: Value
td.event:before	content: "Event: "
td.location:before	content: "Location: "
td.cost:before	content: "Cost: "
td.class:before	content: "Class: "
td.description:before	content: "Description: "
td.day:before	content: "Day: "
td.length:before	content: "Length: "



Each data cell now shows the appropriate labels. CSS can also style the data and labels.

● **Note:** If you make a single selector, as shown in step 11, do not add a comma on the last selector, which would disable the rule altogether. If you don't make a single selector, be sure to remove the comma from each one. No comma is added at the end.

- 12 Create the following selector:

```
section .date,  
section .event,  
section .location,  
section .cost,  
section .class,  
section .description,  
section .length,  
section .day
```

I typed this rule on separate lines to make it easier to read, but you should enter it as one long string in the selector name field or in Code view. As long as the styling for all the data cells is identical, you can combine all the selectors into a single rule, separated by commas. Remember to mind the punctuation and spelling carefully. Even a tiny error in the code can cause the formatting to fail. If you want the styling to be different in one or more of the elements, then create eight separate rules.

Next, let's apply some styling to the labels themselves to make them stand out more distinctly.

- 13 Apply the following properties to the new rule or in each of the separate rules:

```
width: 100%  
padding-left: 30%  
position: relative  
text-align: left
```

● **Note:** Although the formatting is identical for these classes at this point, you may want to adjust the styling for one or more items later. Making separate rules can add flexibility even though it adds to the amount of code that has to be downloaded.

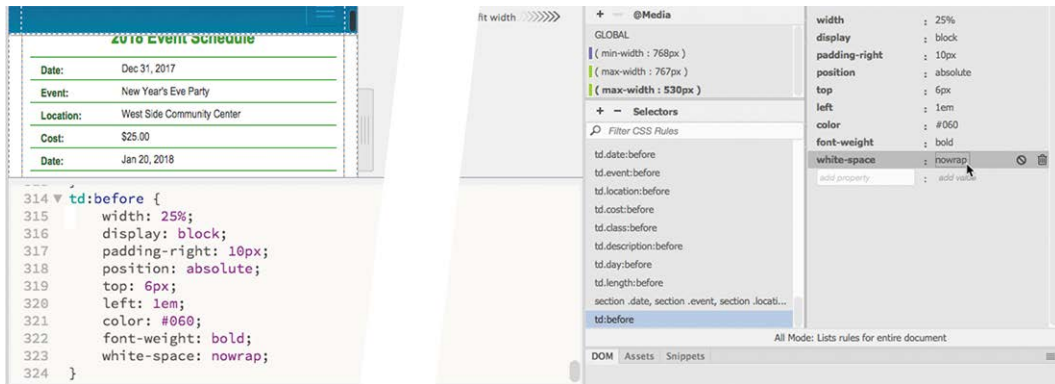


The event and class entries are now indented, and all appear at the same width. Next, you'll add a rule to differentiate the labels from the content of the tables themselves.

- 14 Create the following rule: **td:before**

Give the new rule the following properties:

width: 25%
display: block
padding-right: 10px
position: absolute
top: 6px
left: 1em
color: #060
font-weight: bold
white-space: nowrap

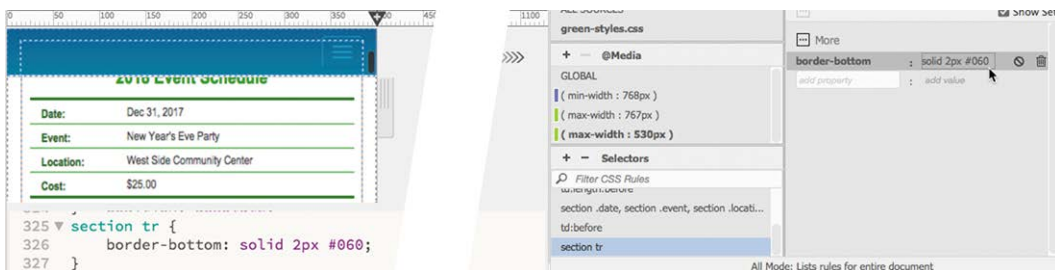


The labels now appear separated from the data and are styled in boldface and dark green. The only thing left to do now is differentiate one record from the next. One way is to simply add a darker border between each table row.

15 Create the following rule: **section tr**

Give the new rule the following property:

border-bottom: solid 2px #060

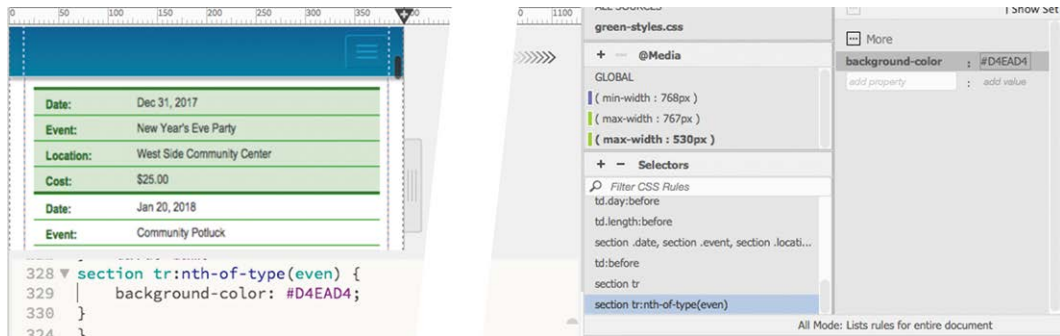


Using a CSS3 selector, you will add a little more pizzazz to the table.

16 Create the following rule: `section tr:nth-of-type(even)`

Give the new rule the following property:

`background-color: #D4EAD4`



◆ **Warning:** Advanced selectors like `nth-of-type(even)` may not be supported by older browsers.

This CSS3-based selector actually applies the background only on even rows of the table.

Both tables are now slickly styled and responsive to any changes in the screen size. Although they look good in Live view, don't get complacent; it's vital to test the design in a variety of browsers and mobile devices too.

17 Save all files. Preview the page in the default browser. Test the media queries and the responsive table styling by changing the size of the browser window.

It's likely that everything you tried in this exercise worked perfectly in both Dreamweaver and any browser you tested it in. But you have to remember that CSS3 is still fairly new and has not been fully adopted within the industry.

The good news is that most of the mobile devices you're targeting should support the various settings used in this exercise. And you can be sure Dreamweaver will stay current with the latest updates.

Adapting images to smaller screens

Today, the modern web designer has to contend with a multitude of visitors using different browsers and devices. Depending on the size of the images and how they are inserted, you may need to use several different strategies to get them to work effectively in your page design.

For example, the images used on the *Contact Us* page are small enough that they should be usable all the way down to the size needed on a smartphone. But that's not true for every page.

- 1 Open **news.html** in Live view.


Make sure the document window is at least 1100 pixels in width.

There are two images on the page. The one at the top stretches all the way across the column; the second floats to the right, with the text wrapping around to the left.

- 2 Click the top image to select it.

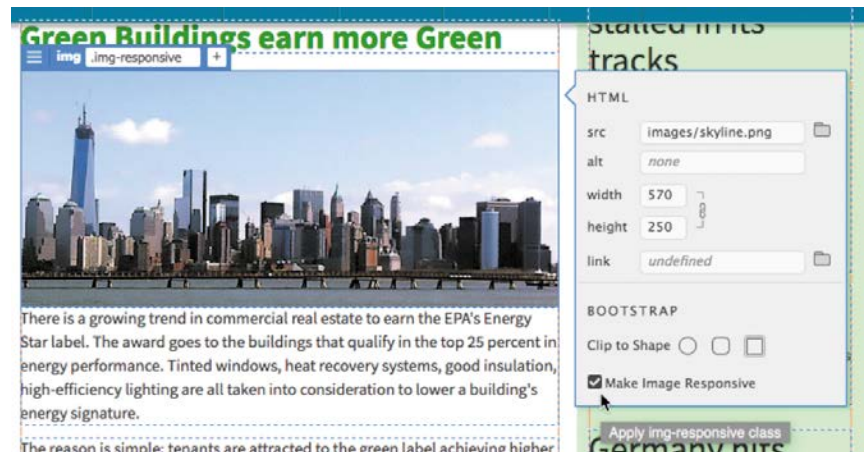


The Element Display appears on the image focused on the `img` element. The image is too large for the column. When selected, you can see that part of it is obscured behind Sidebar 2. Luckily, Dreamweaver has a new built-in way to deal with just this situation.

- 3 Click the Edit HTML Attributes icon  on the new image.

The Quick Property inspector appears.

- 4 Select the Make Image Responsive checkbox.



The image now conforms to the width of the column, but what happens when the screen gets smaller?

- 5 Drag the scrubber to the left and observe how the image adapts to the changes to the layout.



The image scales automatically as the document window changes sizes. The option in the Quick Property inspector selected in step 4 applies the Bootstrap `.img-responsive` class to the image. This class forces the image to fit within the existing element and to scale as needed.

- 6 Scroll down to observe the second image.
Test the image at various screen widths.

The second image displays acceptably until the width drops below 400 pixels. Then, there's not enough space on the left side of the image for the text to wrap effectively. On the smallest screens you should turn off the float property and center the image.

- 7 Repeat steps 2–4 to apply the Make Image Responsive attribute to the second image. Drag the scrubber to 400 pixels.

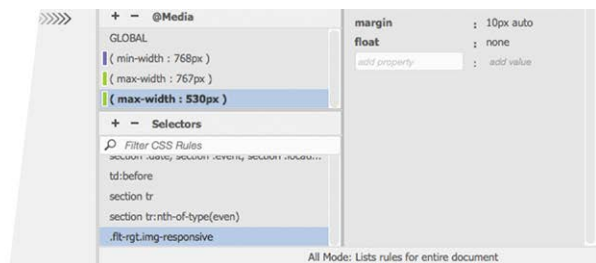
- 8 Choose **green-styles.css** > (max-width: 530px).
Create the following selector: **.flt-rgt.img-responsive**

The selector combines two classes to target this one image.

- 9 Create the following properties:

float: none

margin: 10px auto



These settings will turn off the float property and center the image in the column. The responsive Bootstrap style will now work properly.

- 10 Save all files.

The last items you need to review are the interactive elements added to the *Green Travel* page and the *Green Tips* page.

Hiding components on a responsive layout

Let's open the *Green Travel* page to see how it works in the new layout.

- 1 Open **travel.html** in Live view.

Make sure the document window is at least 1100 pixels in width.

The page contains a table with interactive links that swap the Eco-Tour ad with images from specific Paris tours.

- 2 Drag the scrubber to the left to test how the table responds to the responsive layout.

The table adapts to the changing screen in a fashion similar to the tables created and styled on the *Green Events* page. Everything seems to display fine, although the Eco-Tour ad does not scale or resize in any way.

At a width of 530 pixels, the table's two columns merge and the cells begin to stack one atop the other, including the cell containing the Eco-Tour ad.

The image no longer appears beside the text describing the individual tours.

Although the rollover effect still functions, the purpose of it is lost completely, as is the need for the ad itself. The simplest plan would be just to hide the ad on screens smaller than 530 pixels.

At this moment, there's a custom ID applied to the Eco-Tour ad but not to the cell containing it. CSS can hide the image, but that would leave the blank cell behind. Instead, you will create your own custom class to hide the ad.

● **Note:** As you type, you may notice existing classes called `.hidden` and `.hidden-xs`. These are predefined Bootstrap classes that are designed to hide content at specific screen sizes. You cannot use them in this instance, but keep them in mind for other applications.


- 3 Drag the scrubber to 500 pixels.

- 4 Select the Eco-Tour image.

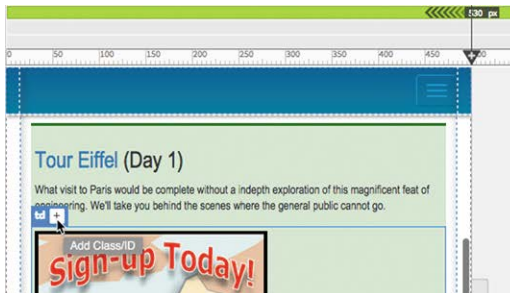
The Element Display appears focused on the `img` element.

- 5 Press the up arrow once.

The Element Display now displays the `td` element.

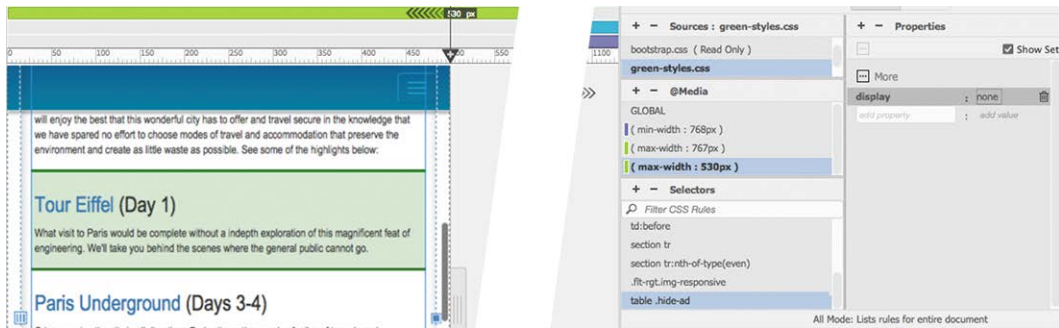
- 6 Click the Add Class/ID icon .

- 7 Type `.hide-ad` in the Element Display text field and press Enter/Return to complete the class name.



The CSS Source dialog appears. We do not want to add this class name to the style sheet as is.

- 8 Press Esc to close the dialog without adding the class to the style sheet.
- 9 Select **green-styles.css** > (max-width: 530px).
Create the following selector: **table .hide-ad**
Create the following property: **display: none**



The table cell and all its contents disappear as soon as the property is created. Whenever the screen is 530 pixels or narrower, the Eco-Tour ad will hide.

- 10 Drag the scrubber to 800 pixels.
Observe the changes to the table and its content. Once the screen is 531 pixels or wider, the normal table design returns and the Eco-Tour ad appears again.
- 11 Save all files.
- 12 Close **travel.html**.

The last item you need to review is the jQuery Accordion widget you created in Lesson 10, “Adding Interactivity.”

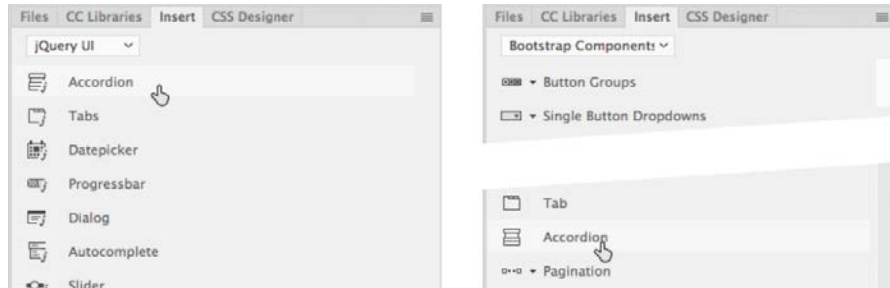
Troubleshooting an interactive element

From time to time you will encounter conflicts between different scripts or web frameworks. You may come across a neat widget or interactive element on the Internet that you’d like to deploy on your website only to find that something in the new component is not compatible with your existing system. This is even more likely when using the Bootstrap framework.

Many interactive elements load their own resources to enable their interactivity. Since the new widget has no idea you are already using a web framework, it becomes more likely that the new element will encounter some issues with the existing software.

This is the case with the accordion used on the *Green Tips* page. You used an accordion from the jQuery UI category in Dreamweaver’s Insert panel. Since you

were not using Bootstrap at that moment, it was your best option. But if you look at the Bootstrap Components category in the Insert panel, you will see it has its own Accordion widget. Both widgets use jQuery libraries, but unfortunately, they are not using the same one. There's no way of predicting how this will affect the widget, although such conflicts rarely produce acceptable results.



It's impossible to avoid conflicts like this. If you work on the web long enough, you will encounter many similar issues. So it's a good thing that you experience that kind of conflict here, where you can learn how to fix it. Let's take a look at the jQuery Accordion widget on the *Green Tips* page.

Styling a jQuery Accordion widget in a Bootstrap layout

In this exercise, you will inspect the jQuery Accordion widget and see how it fared in the transition to the Bootstrap layout.

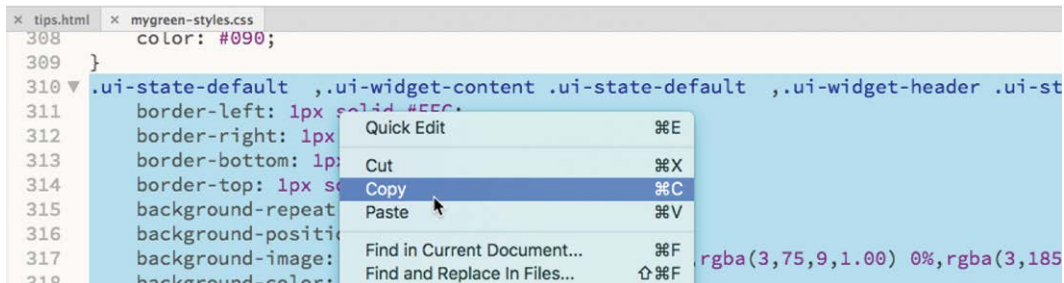
- 1 Open **tips.html** in Live view.
Make sure the document window is at least 1100 pixels in width.

The first thing you'll notice is that the styling for the accordion did not make it over to the new layout. Like the table styling you encountered earlier in this lesson, it's a simple matter to move the styling for the accordion over from **mygreen-styles.css**.
- 2 Open **mygreen-styles.css**.
- 3 Scroll down through the style sheet to locate the CSS rules formatting the jQuery Accordion widget.



Around line 310 you will see styles starting with the class `.ui-state-default`. These are the styles you created to format the accordion in Lesson 10. You will need to move all of these styles over to **green-styles.css**.

- 4 Select all the CSS code formatting the accordion (approximately lines 310 to 341). Copy the code.



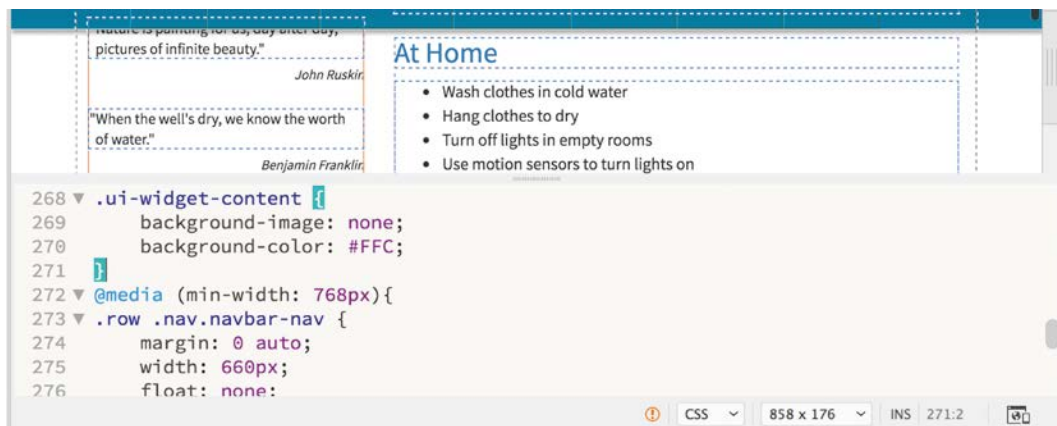
- 5 Switch to **tips.html**. Select **green-styles.css** in the Related Files interface.

The document window switches to Split view and loads **green-styles.css** in the Code view window.

- 6 In the Code view window scroll down to the entry `@media (min-width: 768px)` (around line 240).

As you did earlier, you'll paste the CSS code preceding the media query.

- 7 Insert the cursor before the media query. Paste the CSS code. Press Enter/Return to move the entry `@media (min-width: 768px)` to a new line, if necessary.



The CSS is in place, but the accordion still appears unstyled. If you look more closely, you will also notice that the HTML lists are all visible and no accordion or interactivity is visible. What happened?

There are two problems with the current component. The accordion and the Bootstrap framework are both based on jQuery, but the two systems implemented in Dreamweaver use different versions. That's the first problem. The second problem is that both versions are being called at the same time. That's not allowed. You can call one library or the other. You can even call one library twice. But you can't load two different libraries at the same time and expect everything to work. Luckily, there's an easy fix.

- 8 Open **mybootstrap_temp.dwt** from the lesson15 folder in Code view.

The Bootstrap jQuery library (`jquery-1.11.3.min.js`) is being called in the site template to support the responsive layout as well as other Bootstrap components used on the page.

- 9 Scroll down to the bottom of the file. Look for an HTML comment and `<script>` tag loading the Bootstrap library **jquery-1.11.3.min.js** (around line 88).

- 10 Select the related comment and the entire script markup and cut it into memory.

Typically, scripts like this are loaded near the bottom of the code. In this case, it will work fine loading up in the `<head>`, where it needs to be to support and activate the accordion code. Be mindful that you can't always move such scripts.

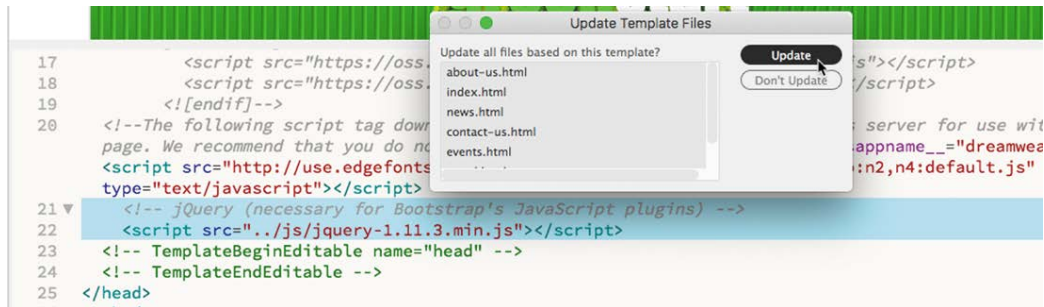
● **Note:** It's not always possible to use a newer library to power every widget or component. Sometimes you may have to scrap an older application and rebuild it from scratch using the newer library.

- 11 Scroll up to the closing `</head>` tag (around line 25).

Notice the editable region just above the closing tag. In the template the editable region is empty, but this is where the jQuery UI library (`jquery-1.11.1.min.js`) is inserted in **tips.html**. The Bootstrap library is a slightly newer version than this one, so you will use the Bootstrap library and delete the older one.

- 12 Insert the cursor before the comment `<!-- TemplateBeginEditable name="head" -->`.

- 13 Paste the code cut in step 10. Save the template.



The Update Template Pages dialog appears.

- 14 Click Update.

All child pages are updated. You just moved the Bootstrap jQuery library up into the <head> section of every page.

- 15 Close the template. If necessary, save changes to **green-styles.css**. Switch to **tips.html**.

- 16 Switch to Code view. Select Source Code in the Related Files interface, if necessary. Scroll down to the closing </head> tag (around line 31).

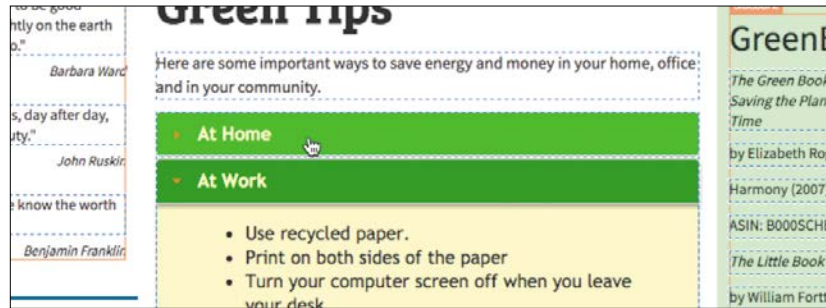
You can see the comment and script elements you moved just above the editable region. In the template, and on all the other pages of the site, the editable region was empty. In **tips.html** it features two scripts, one loading the Bootstrap jQuery library and the other loading a specific library supporting the jQuery UI accordion.

- 17 Delete the jQuery UI library script:

```
<script src="jQueryAssets/jquery-1.11.1.min.js"></script>
```



- 18** If necessary, switch to Live view.



As soon as the conflicting script is deleted, the accordion is fully styled and seems to be back in business. Let's test it.

- 19** Scroll down so you can see more than one panel in the accordion.
Click one of the closed panels.

The closed panel opens; the open panel closes.

- 20** Click the other closed panel.

The other closed panel opens and the open panel closes. By deleting the conflicting library and moving the Bootstrap library into the <head> section, you corrected the issues in the accordion widget. Now let's see how the accordion looks at various screen sizes.

Adapting an Accordion widget to responsive design

The jQuery Accordion widget looks and functions properly now in a desktop environment. Let's see how it functions at all screen sizes.

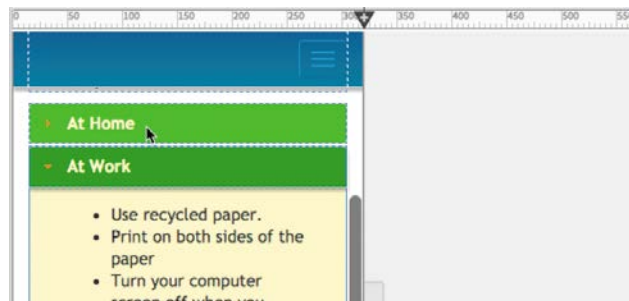
- 1** Drag the scrubber to 800 pixels.

The layout switches to two columns.

- 2** Click a closed tab in the accordion. Test each panel.

The accordion looks correct and functions properly.

- 3** Drag the scrubber to 320 pixels.



The layout switches to one column. At 320 pixels, this is the smallest screen size you should have to support.

- 4 Click a closed tab in the accordion. Test each panel.

The accordion functions properly, but the bulleted lists are not using the space effectively. The lists are indented too far. Let's reduce the indent a bit.

- 5 Click one of the list items.

The Element Display appears focused on the `li` element. Lists may be indented on the `` element, the `` element, or both.

- 6 Select the `ul` tag selector.

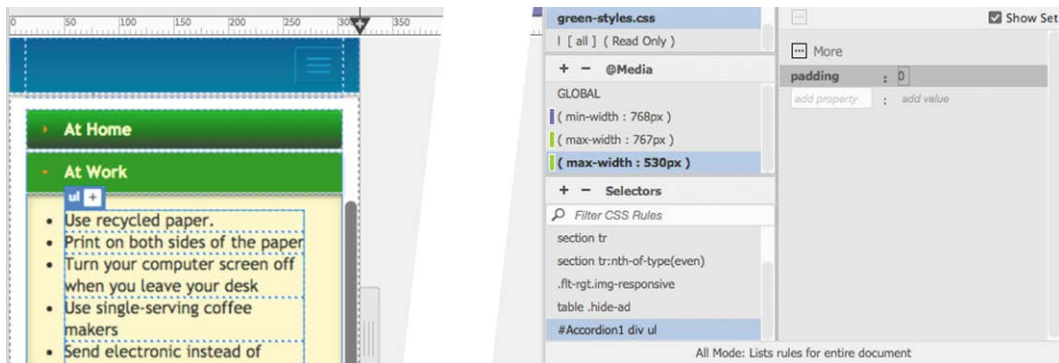
- 7 In the CSS Designer, click the All button.

Select **green-styles.css** > (max-width: 520px).

Click the Add Selector icon **+**.

A new selector, `#Accordion1 div ul`, appears in the Selectors panel. It targets only the lists in this accordion. By adding it to the media query (max-width: 520px), it will kick in only on the smaller screens and devices.

- 8 Create the following property: **padding: 0**



The list items move to the left and occupy most of the screen.

- 9 Drag the scrubber to the right to open the document window fully.

As you drag it open, the accordion adapts to the layout seamlessly, in one, two, or three columns.

- 10 Save and close all files.

Once you complete the review of the site in Dreamweaver, you should review every page, component, and piece of content in a variety of browsers and any mobile devices you have at hand. Dreamweaver makes this process easy too.

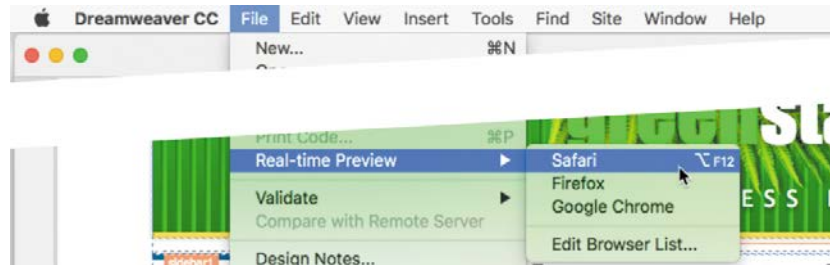
Previewing pages using Real-Time Preview

Since Live view is based on the WebKit web browser engine, it provides a pretty good facsimile of your pages and content, but nothing can take the place of an actual browser, smartphone, or tablet. You cannot be absolutely sure that your pages and design work properly until you have tested them all in the actual environments. Real-Time Preview was added to Dreamweaver to make testing your pages and site a one-step operation.

- 1 Open **index.html** from the lesson15 folder.

The first part of Real-Time Preview allows you to preview pages in any browser installed on your computer that is registered with Dreamweaver. New ones can be added by selecting File > Real-Time Preview > Edit Browser List.

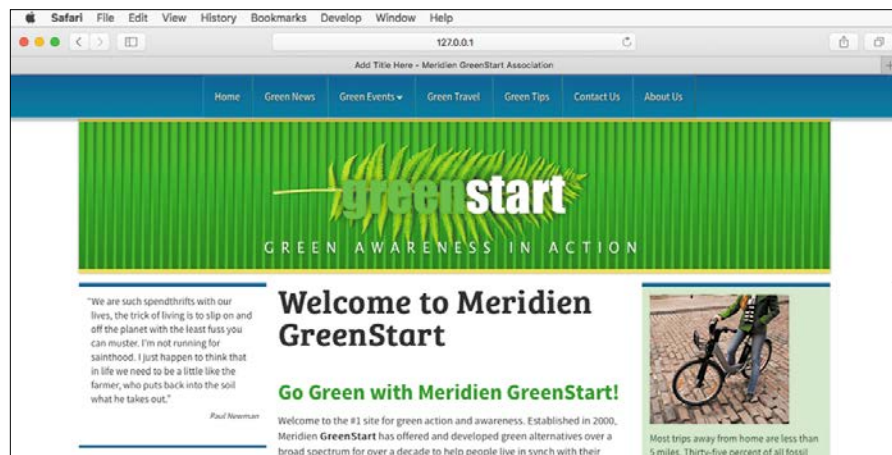
- 2 Select File > Real-Time Preview > Safari or whatever is your default browser. Maximize the browser to fit the entire computer display.



● **Note:** When using Real-Time Preview, you may receive an error message in your browser. If you do, try clearing your browser cache. Close and relaunch Dreamweaver and try again.

The page opens in the browser, displaying the content in three columns.

- 3 Observe the entire page and all the content.



You should also test each page at various screen widths.

- 4 Drag the right edge of the browser window to reduce the width.
Stop dragging the edge when the layout switches to two columns.
Review the content again to look for anything that doesn't work or doesn't look acceptable.

- 5 Reduce the width again until the layout switches to one column.
Review the content.

- 6 Click the sandwich icon on the main menu.
The hyperlinks should all work.

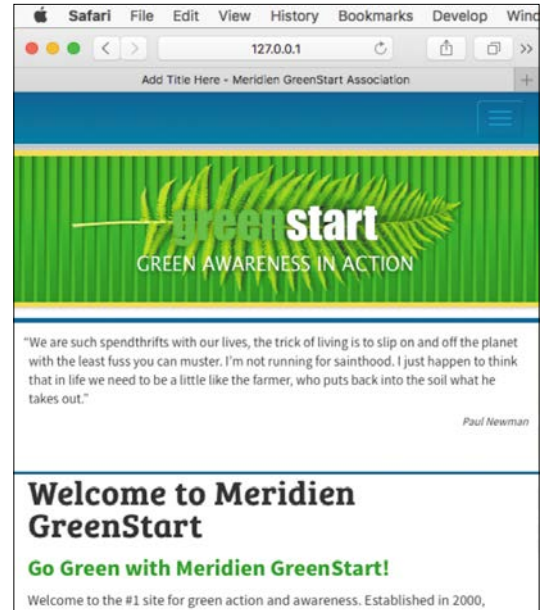
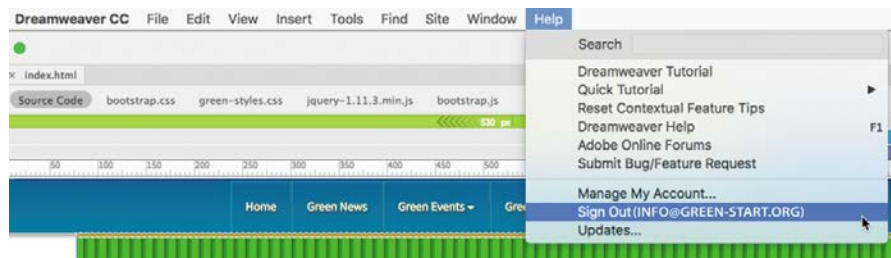
- 7 Click the *Green News* link.
The *Green News* page loads in the browser, replacing the home page.

- 8 Test the *Green News* page at various widths.
When you finish testing the page, click the next menu item and start the process again. Once you are done testing the site in one browser, select the next browser on your system and begin the process all over from scratch.


Once you finish testing the website on your desktop browsers and fix any issues that you find, you should test everything again on a range of smartphones and other mobile devices. But you might ask, “How do you test pages without having a web server where you can upload all your pages?”

While it's nice to have, there's no need to have your own web server for testing purposes when you are using Dreamweaver CC (2018 release). That's because Real-Time Preview can upload your site to a preview area of Creative Cloud where you can test your pages on the Internet. All you need is a live connection to the Internet and an active Creative Cloud account. Typically, you are always logged in when you are using Dreamweaver, but you can see your status by checking the Help menu.

- 9 In Dreamweaver, select Help.



The menu should say “Sign Out” and your login name. If it says “Sign In,” you will have to log in to your account before you can use Real-Time Preview.

- 10 If necessary, log in to your Creative Cloud account in Dreamweaver; otherwise, skip to step 11.
- 11 Open **index.html**.
- 12 Click the Real-Time Preview icon  in the lower-right corner of the document window.

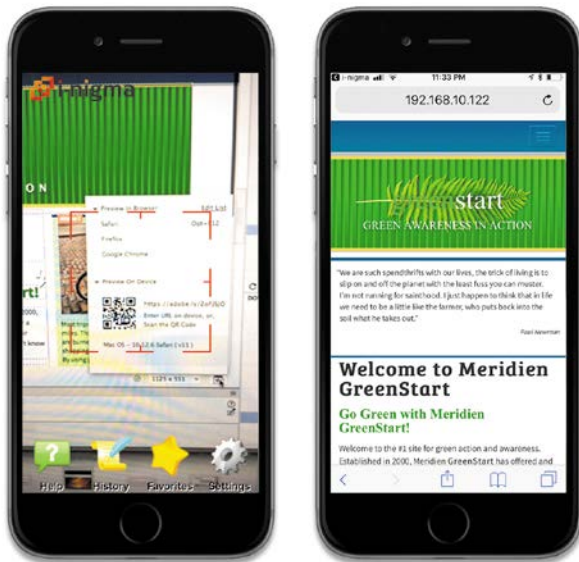


A pop-up window should appear showing a Quick Response (QR) code and a custom Adobe URL. If you have a QR app on your phone or tablet, you simply have to scan the code and you will be redirected to a preview copy of your entire site uploaded to Creative Cloud. You can even share the URL with your coworkers or clients.

► **Tip:** To obtain a QR app for your phone or tablet, just go to the app store for your device and search for QR or quick response.

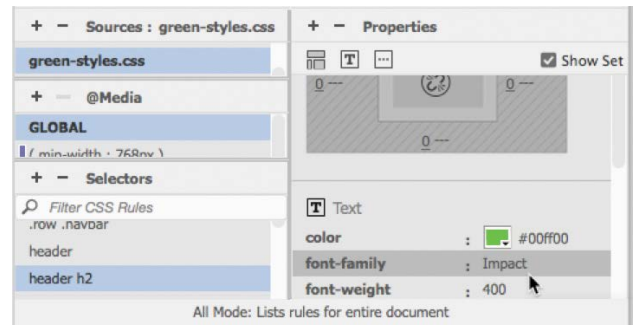
- 13 Scan the QR code with your phone or tablet, or enter the URL into a browser on your device.

The home page should load in your device browser. The screenshot shows the preview on an iPhone 6. As you can see, there are problems with the page. The logo and motto are not showing the correct fonts. You are seeing firsthand the reason it's vital to test all your pages off of your computers and on as many different browsers and devices before taking a site live. Luckily, this is one of the easier issues you can have with a new site.



- 14** In Dreamweaver, select **green-styles.css** > GLOBAL in the CSS Designer.
Select the rule header **h2**.

The rule calls only the font Impact. Since that font is not supported by the iPhone, or any other iOS device, the logo falls back to the default font of the iPhone browser. If you want the logo to appear properly, you have to create a font stack that will support a wide range of browsers and devices.



- 15** Deselect the Show Set option in the CSS Designer.

It's easier to create a font stack when this option is turned off.

- 16** Click the font-family property.

The font stack pop-up list appears.

- 17** Click Manage Fonts.

Create a custom font stack with the following fonts:

Impact

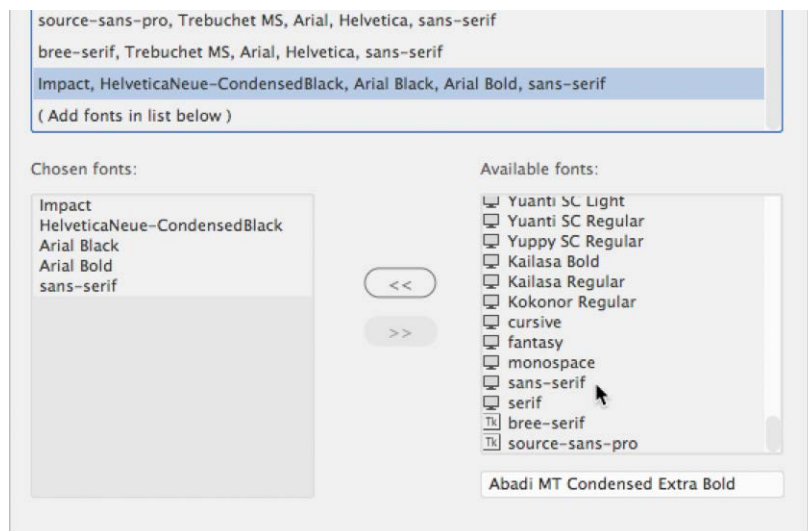
HelveticaNeue-CondensedBlack

Arial Black

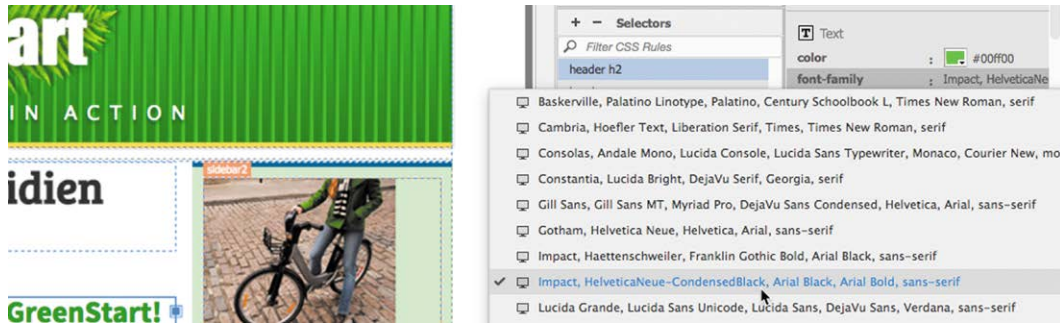
Arial Bold

sans-serif

Note: You must enter the name for HelveticaNeue-CondensedBlack exactly as shown or it will not load properly in an iOS device.



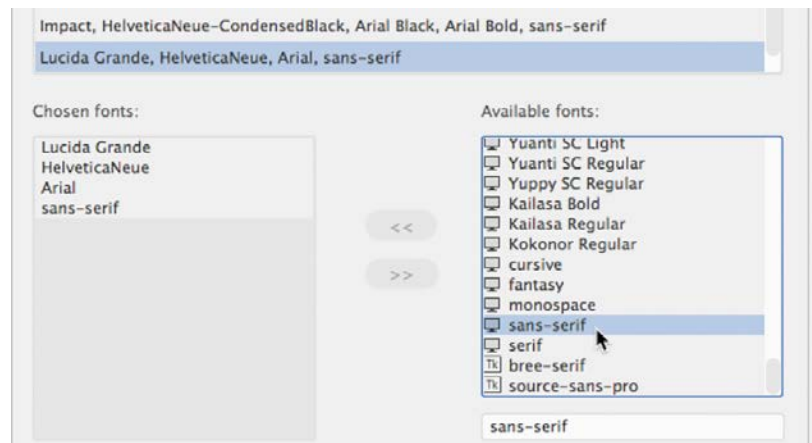
- 18 Select the new font stack for the rule header h2.



Now let's fix the motto.

- 19 Repeat steps 14–18 to create a new stack for the rule header p.
Add the following fonts to the custom font stack:

Lucida Grande
HelveticaNeue
Arial
sans-serif



- 20 Select the new font stack for the rule.
21 Save all files.

Once the new font stacks are created and applied to the logo and motto, you can preview them again using Real-Time Preview.

- 22 Repeat steps 12–13 to preview the changes on your smartphone or tablet.

As you can see from the new screenshot from an iPhone, the changes to the styling look much better. You've come a long way, but you are not finished. You now know how to test and fix a variety of issues with your new website. Continue testing the pages and tweaking the content for as many different devices as you can lay hands on.

Congratulations! You've designed, developed, and built an entire website and uploaded it to your remote server. By finishing all the exercises in the book and online, you have gained experience in all aspects of the design, development, and testing of a standard website compatible with desktop computers and mobile devices. Now you should be ready to build and publish a site of your own. Good luck!



Review questions

- 1 How can you target a specific device size or orientation for styling content?
- 2 True or false? When resetting the styling of a rule, you must duplicate all the properties of the rule in the media query.
- 3 How can you control the number of columns displayed for different screen sizes in a Bootstrap layout?
- 4 Can tables be made to be responsive?
- 5 How can you test your webpages for responsive design if you don't have a web server?
- 6 What do you need to have to enable Real-Time Preview?

Review answers

- 1 To target a specific size screen or device, you can create a custom media query in your style sheet.
- 2 False. You have to create only the properties that need to be reset or changed.
- 3 In Bootstrap, you can control the number of columns displayed at a specific breakpoint by assigning a predefined class to the structural elements.
- 4 Yes. Using some CSS3 properties you can format table content to be viewable on any size screen.
- 5 Dreamweaver provides Real-Time Preview, which enables you to preview your pages on desktop browsers and any available mobile device.
- 6 You must have a live connection to the Internet and be logged in to an active Creative Cloud account to use Real-Time Preview.