

15

ADAPTING CONTENT TO RESPONSIVE DESIGN

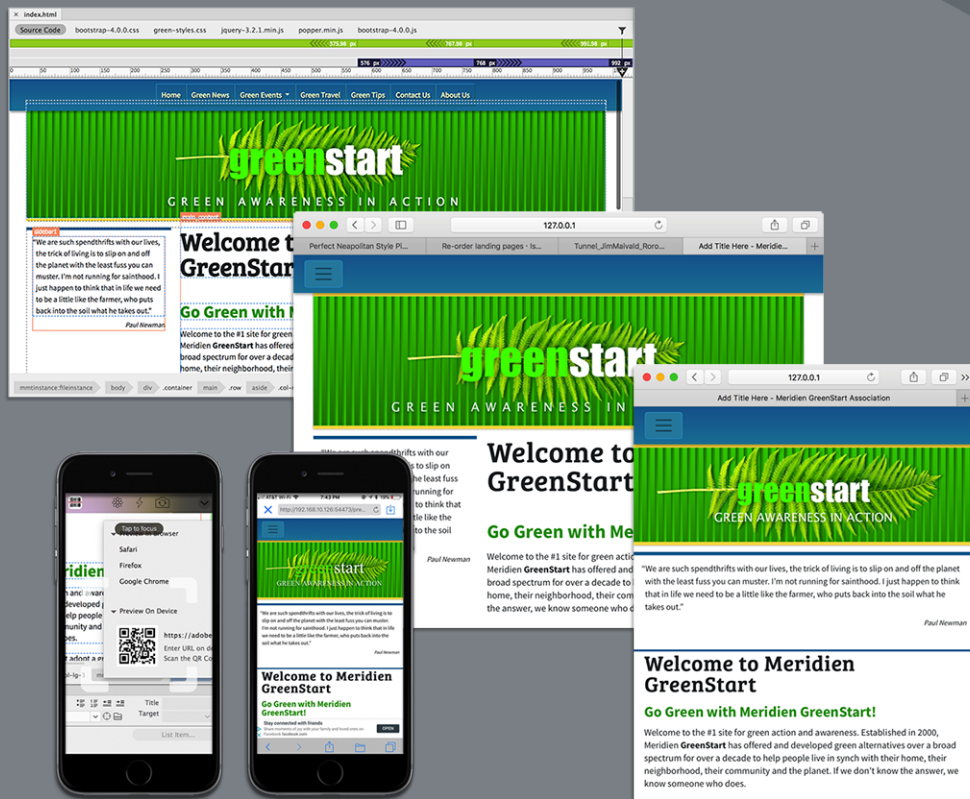
Lesson overview

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 and 15 minutes 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.



In most cases, text, images, and other HTML components will adapt automatically to a Bootstrap layout. When they don't, you'll have to know some techniques to get them to look right and work correctly.

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 with 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 often be your boss or client who finds the problem first. You never want to be in that position.

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 computer 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. And don't forget portrait and landscape orientations either. But even with all that effort you will still miss something. Be ready to jump on any report of a page or content that is misbehaving.

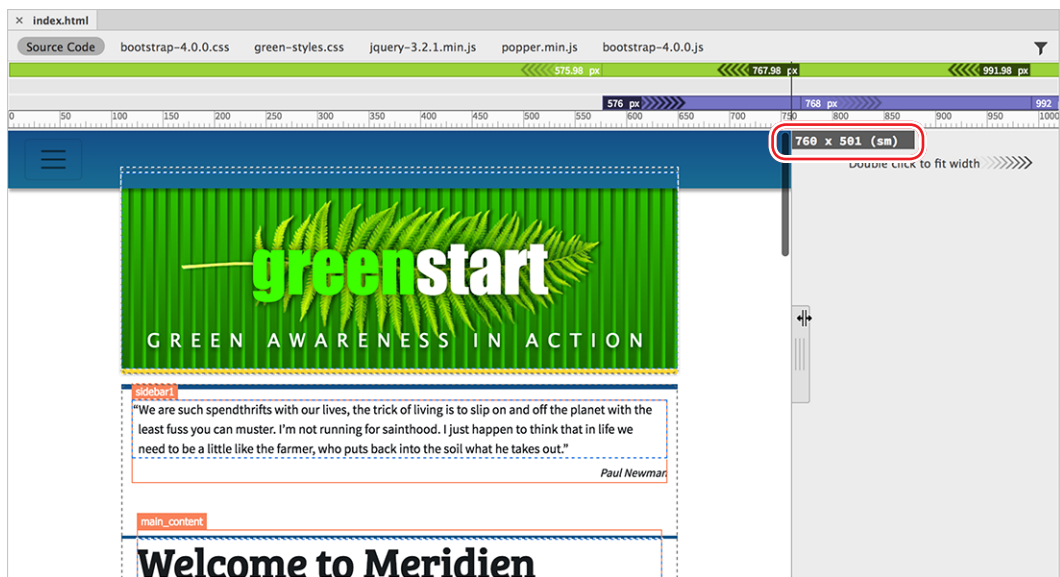
In the following exercises, we'll take look at some techniques for reviewing your pages and content.

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 home page 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.

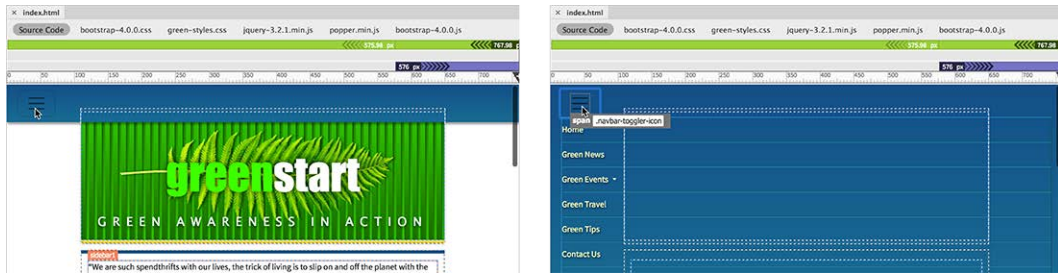
3 Drag the scrubber to the left to simulate the width of a screen 768 pixels wide or less.



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 left corner.

- 4 Click the sandwich icon to open the menu.

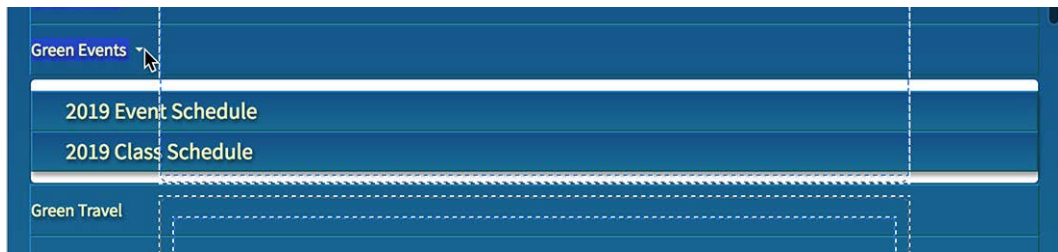


When you click the sandwich icon, the navigation menu drops down, showing all seven links stack vertically. Note that the *Green Events* menu item shows an arrow icon, indicating that it hosts its own dropdown menu.

- 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.

- 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 and the gap above and below them.

- 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

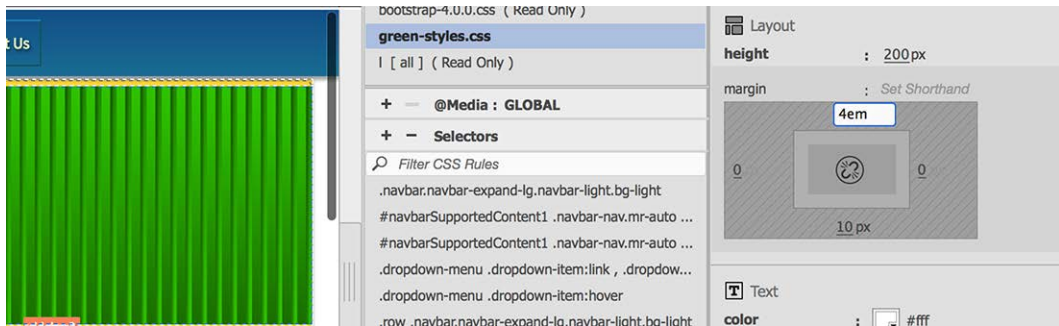
Unfortunately, the site did not pass the first test. The design has some major problems. But don't worry—all the issues observed are normal and expected in a new responsive site. Luckily, everything can be fixed pretty easily with some simple CSS tweaks.

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 and adding and styling content. You have to be constantly vigilant.

- 1 In CSS Designer, click the All button.
Select **green-styles.css** in the Sources panel.

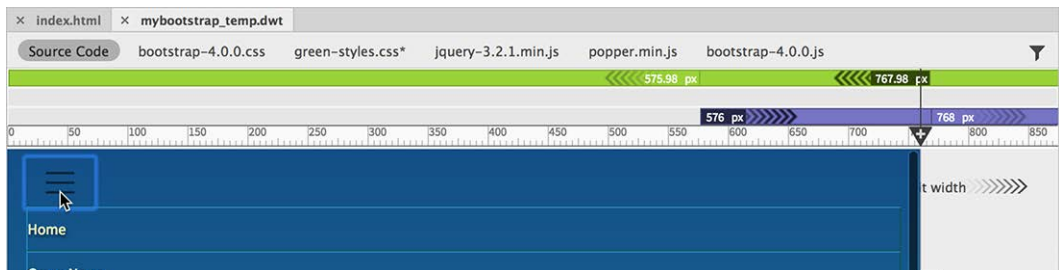
The Selectors panel displays all the rules defined within the style sheet. The margin setting that controls the placement of the border is defined in the rule header.

- 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 768 pixels or less.



You can see that the border around the sandwich icon is dark gray. The background color of the sandwich is the same as the rest of the menu bar. The whole color scheme makes it difficult to see amid the dark blue background. A lighter color scheme would make it easier to see. 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

Note: The border appears light blue when the element is selected.

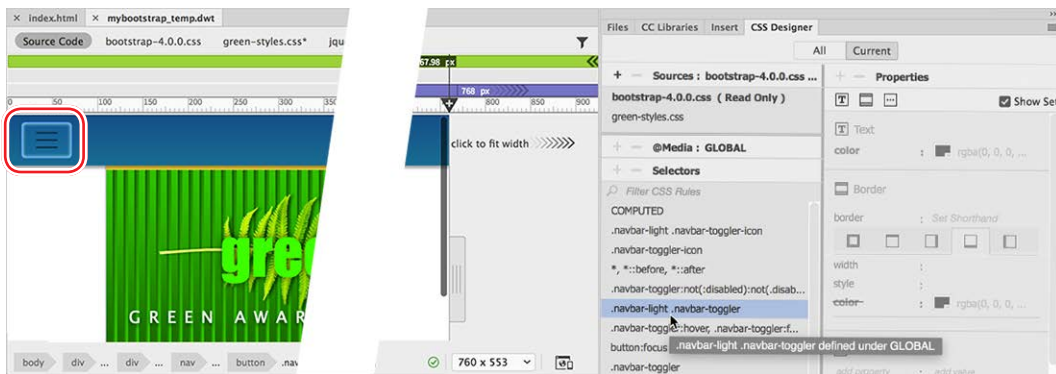
writing, the CSS Designer ignores elements in the locked areas. To track down the pertinent rules, you'll need to work with the template itself.

- 4 Open **mybootstrap_temp.dwt** in Live view from the lesson15/templates 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 the editable regions.

- 5 Drag the scrubber to 760 pixels.
- 6 In the CSS Designer, click the Current button. Click the sandwich icon and observe the Selectors pane.

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. What you may notice is that there are two different components that compose the menu icon: `navbar-toggler-icon` and `navbar-toggler`. Both items set border properties, but only `navbar-toggler` sets a border color. To override the border and other non-scheme colors, you need to create new matching rules in your custom style sheet. Note the exact name of the existing rule.



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

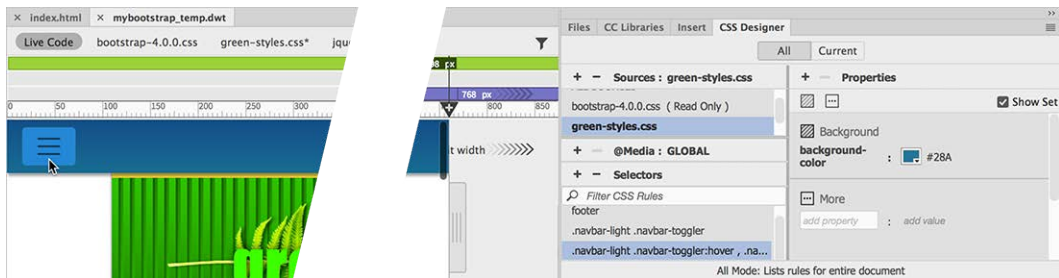
8 Add the following property to the new rule:

border: solid 1px #29C



This rule resets the borders around the outside of the icon. But the darker background color makes it hard to see the bars within the icon.

9 Add the following property: **background-color: #28A**



The lighter background color now makes the three bars within the sandwich icon easier to see. Let's add a hover behavior to indicate the nature of the interactivity.

10 Create the following rule:

**.navbar-light .navbar-toggler:hover ,
.navbar-light .navbar-toggler:focus**

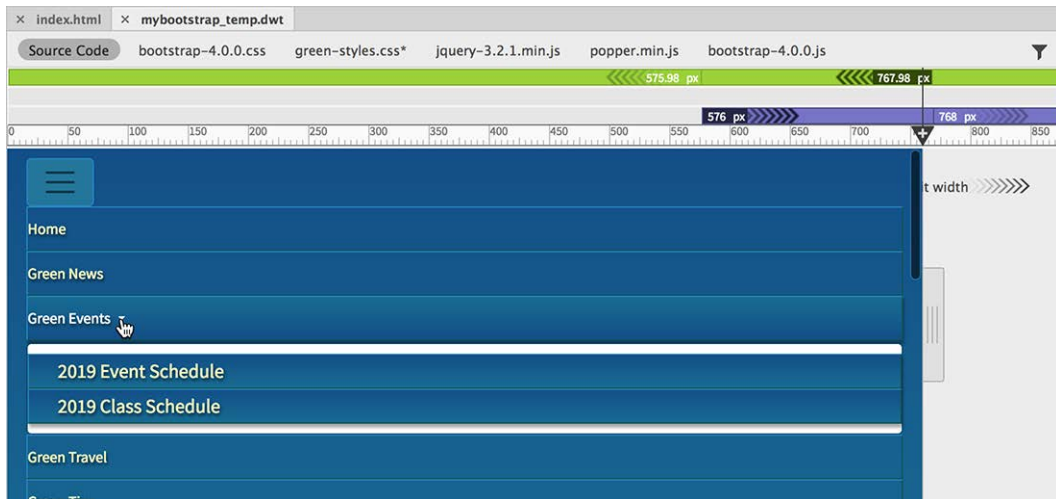
● **Note:** Don't forget the comma between the two parts of the selector.

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

11 Add the following property: **background-color: #29D**

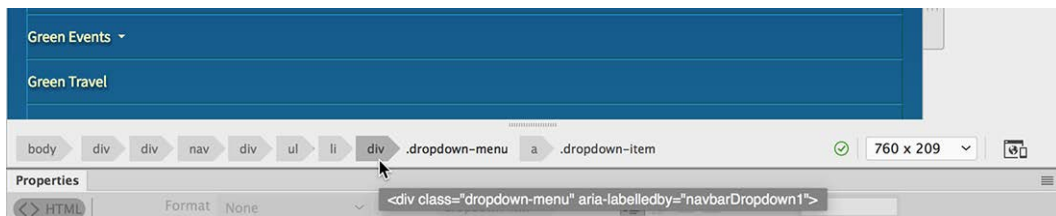
That addresses all the styling issues with the sandwich icon, but you may remember that the submenu also had some styling issues.

- 12** Click to open the horizontal menu.
Click to open the *Green Events* sub-menu.



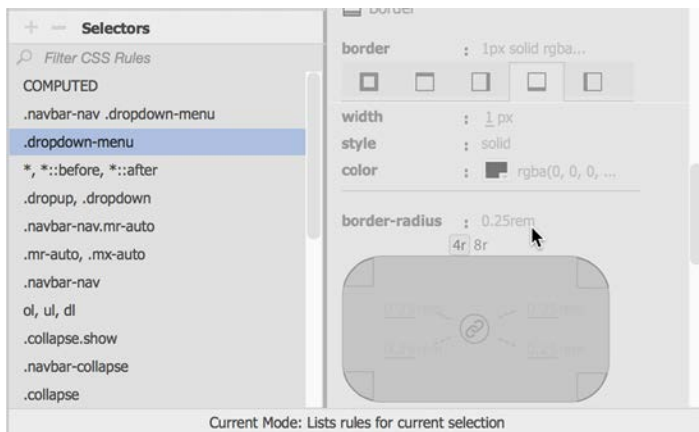
Notice the gap above and below the submenu items. One clue to help you find the appropriate rule to reset is the border radius visible on the corners of the dropdown menu.

- 13** Click one of the subitems in the dropdown menu.
Examine the tag selectors.



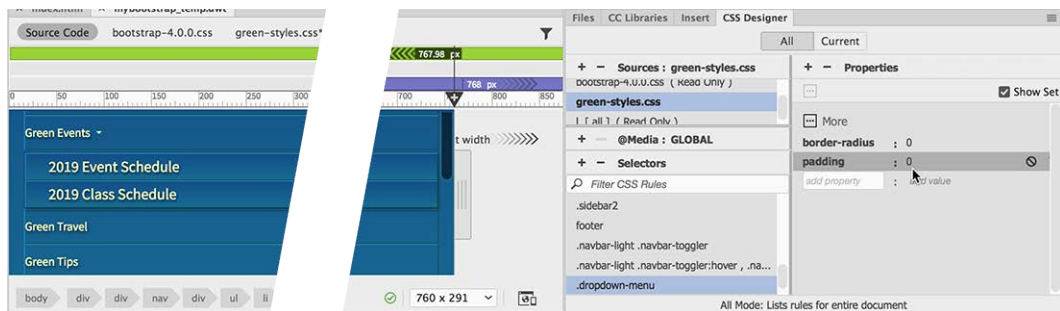
The dropdown menu is constructed using two `<a>` elements wrapped by a `<div>`.

- 14** Click the tag selector `div.dropdown-menu`.
Click the Current button and examine the rules styling the menu.



It's helpful when an HTML element has some sort of obvious styling. The border radius on the dropdown menu makes it easy to identify the specific rule styling it.

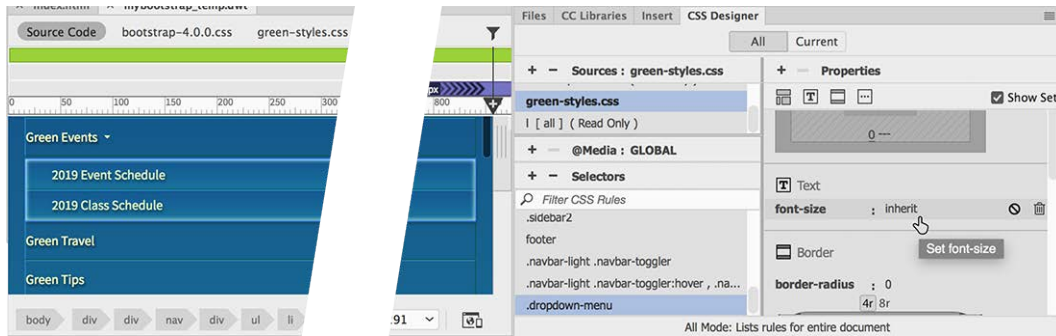
- 15 Click the All button.
- 16 Select **green-styles.css** and create the following rule:
.dropdown-menu
- 17 Create the following properties:
border-radius: 0
padding: 0



The new rule eliminates the gap above and below the submenu. You may also notice that the text in the submenu is slightly larger than the other menu items.

18 Create the following property:

font-size: inherit



Using the inherit setting allows the styling applied to the other menu items to pass through to the submenu. That way, if you change size of the main menu, the submenu will update automatically.

19 Save all files.

The changes in this exercise have brought the styling of the navigation menu within the site's design and color scheme. Close the menu and drag scrubber to the left and review the rest of the layout to identify any other styling issues.

Below 768 pixels in width, the content starts to display in a single column. If you watch carefully, at a certain point the company name and logo are too large for the available space and the motto breaks into two lines. You can see the second line displayed in the middle of the quotation.

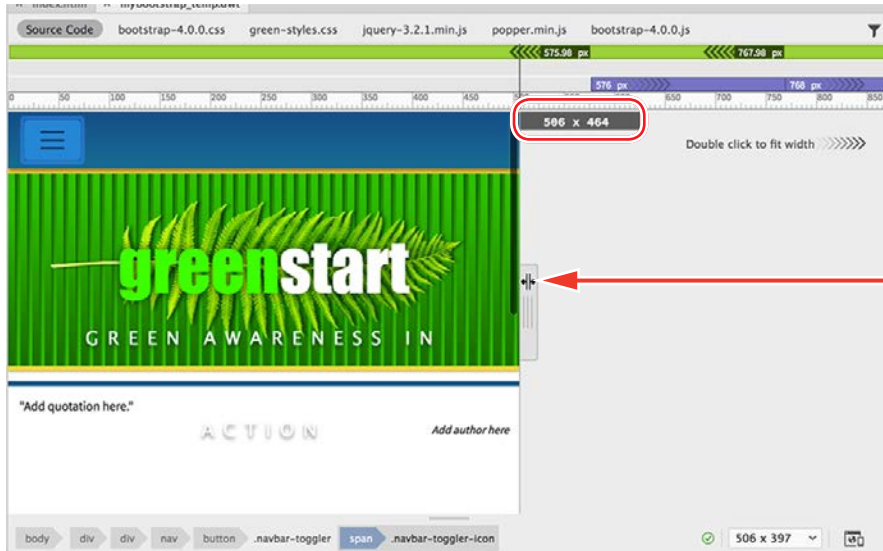
As you can plainly see, when the screen gets too small, the styling of the header is not scaling to fit the available space. There are several solutions to this issue, but the simplest method is to change the way the size of the logo and motto is applied. Instead of applying the font sizes as a global specification, you will apply them based on the size of the screen. To apply CSS styles based on a specific screen width, you first 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.

Adapting the header to smaller screens

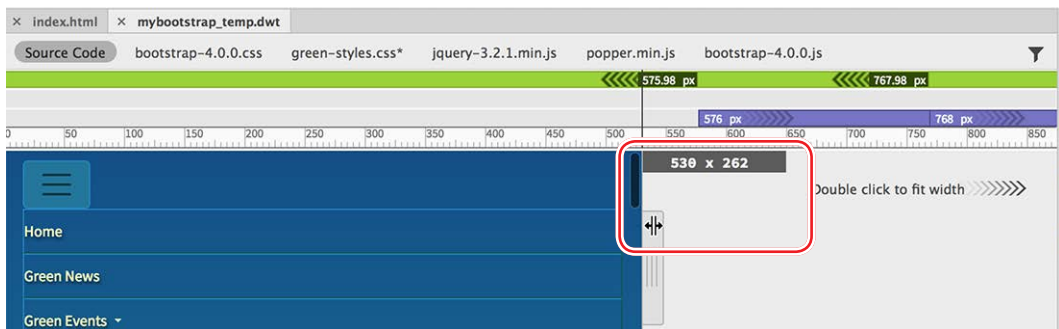
If the global styling cannot serve the purpose of applying the needed styling, you will often need to create custom media queries and rules. In this exercise, you will create another media query to adapt the header to smaller screens.

- 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.



On my computer, the text breaks around 520 pixels or so. The exact width 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 so that your new media query doesn't break right out of the box.

- 3 Drag the scrubber to 530 pixels.



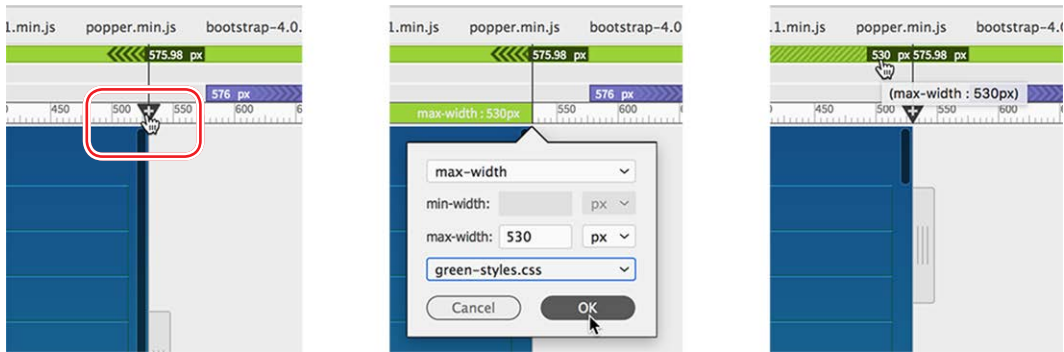
- 4 Click the Add Media Query icon  at the top of the scrubber.

● **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 screenshots. Substitute your measurement in the following exercises, as necessary.

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

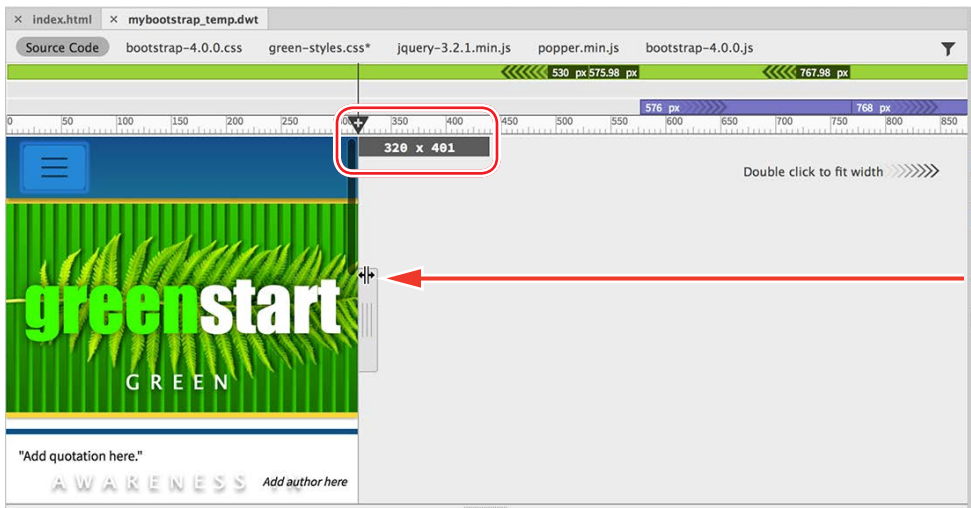
The Media Query Definition dialog appears. The max-width field is populated with the current scrubber position (530 px). The min-width field is grayed out, and **green-styles.css** is already selected in the Source menu. The new media query will apply its styles only when the screen is 530 pixels or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

5 Click OK to create the media query.



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 when adjusting the styling to set the scrubber at the smallest screen size you would need to support before making any new rules.

6 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.

7 Switch to Split view, if necessary.

8 Click the motto in Live view.

If the Element Display does not appear highlighting the motto in Live view, insert the cursor in the motto text in the Code view window.

9 Select the p tag selector.

When you create 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.

10 Click the All button in the CSS Designer.

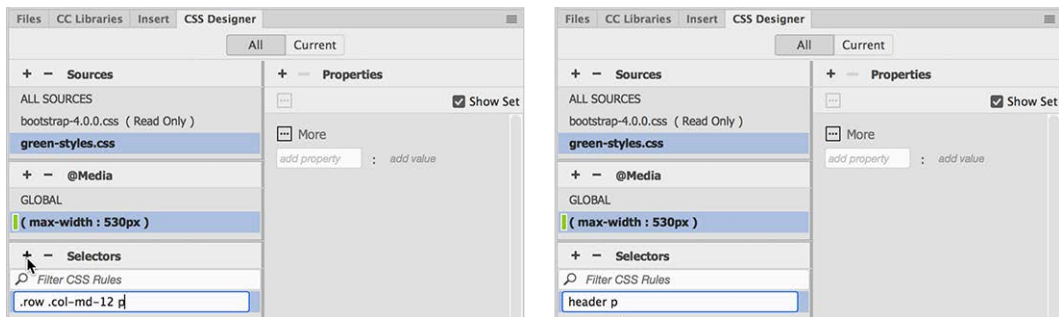
Select **green-styles.css** in the Sources pane.

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.

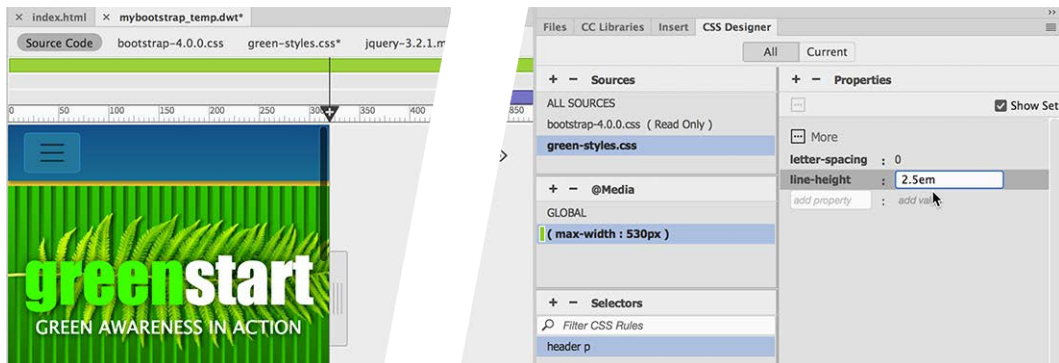
11 Create a new selector. Edit the selector name to **header p**



12 Create the following properties:

letter-spacing: 0

line-height: 2.5em



The motto should now appear on one line, although you may need to refresh the display in Live view. The new styling will work on screens smaller than 531 pixels.

- 13** 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.

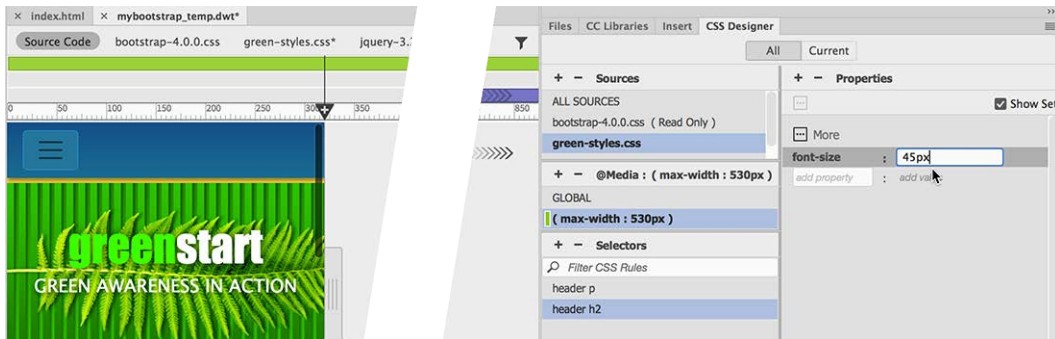
- 14** Drag the scrubber to 320 pixels.

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

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

Create the following selector: **header h2**

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



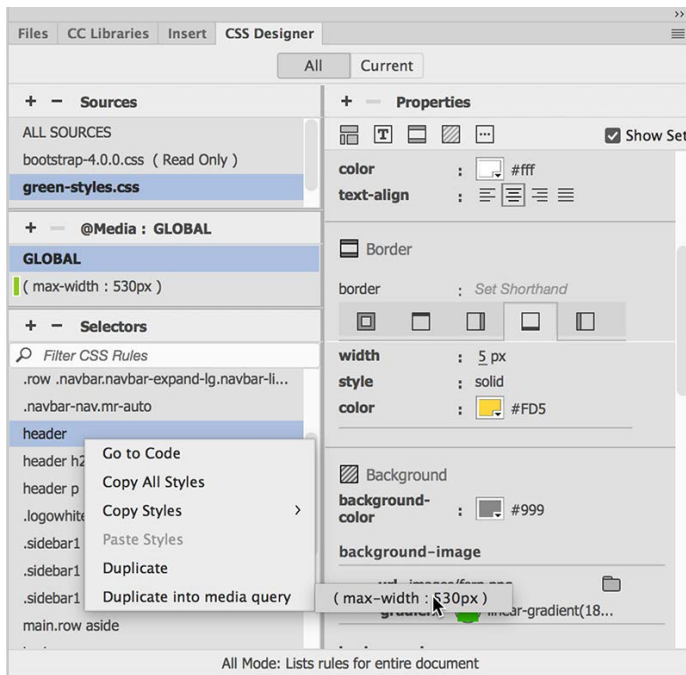
The heading reduces in size to fit the space better. The fern image is part of a complex background specification. The best way to modify a specification like this is to duplicate the entire rule into the media query and edit the appropriate settings.

Note: You may need to refresh Live view to see the changes.

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

- 17** 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.

- 18 Select **green-styles.css** > (max-width: 530px) in the CSS Designer. Right-click the rule header and select Go 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.

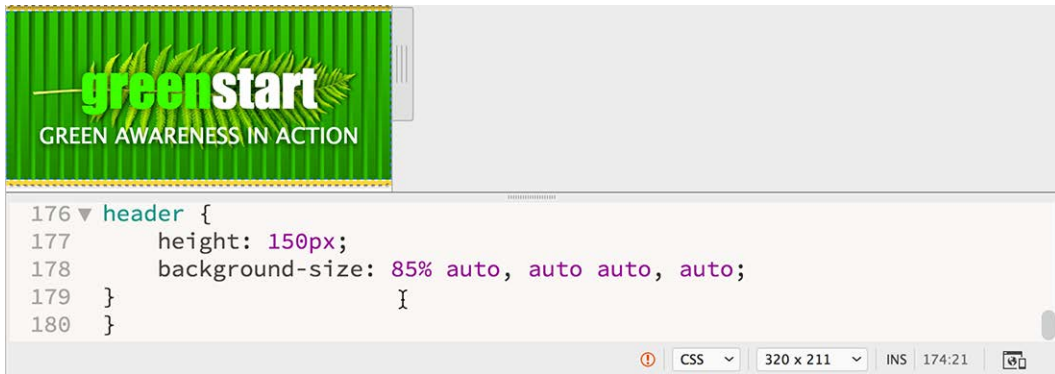
- 19 Delete the following properties:

```
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/stripe.png), -webkit-linear-gradient(270deg, rgba(0,153,0,1.00) 0%, rgba(0,204,0,1.00) 100%);
background-image: url(images/fern.png), url(images/stripe.png), linear-gradient(180deg, 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. The existing properties are set as global settings and are inherited by the header element at all screen sizes. The only properties you need to keep are those that will be reset.

- 20** Edit the following properties as highlighted:
- height: **150px**;
- background-size: **85% auto**, auto auto, auto;

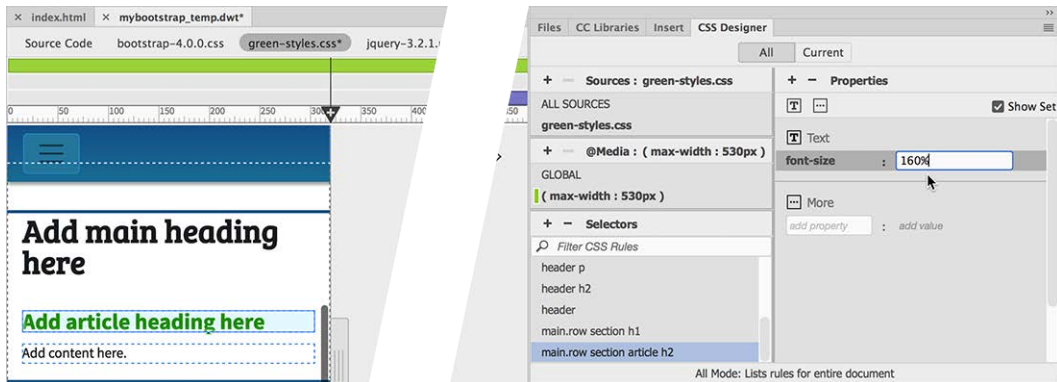


These changes have greatly reduced the header footprint, which is appropriate for smartphones and other mobile devices. It's also a good idea to reduce the size of the regular content.

- 21** Select **green-styles.css** > (max-width: 530px) and create the following selector:
- main.row section h1**
- 22** Create the following property:
- font-size: 225%**
- 23** In **green-styles.css** > (max-width: 530px) create the following selector:
- main.row section article h2**

24 Create the following property:

font-size: 160%



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

25 Save all files. Switch to Live view.

26 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 and needs.

27 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.

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

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.

● **Note:** Don't forget to select the GLOBAL option in the @Media pane.

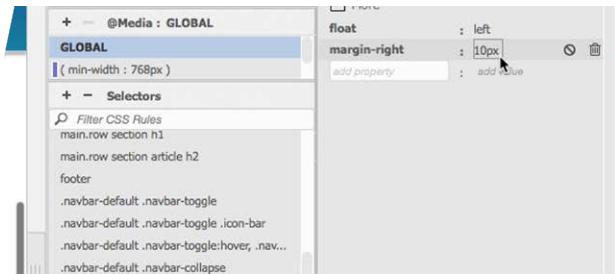
3 Select the All button in the CSS Designer.
Select **green-styles.css** > GLOBAL.

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.

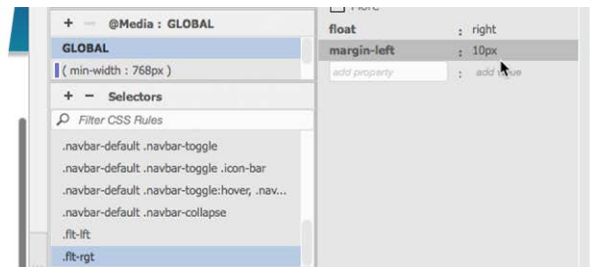
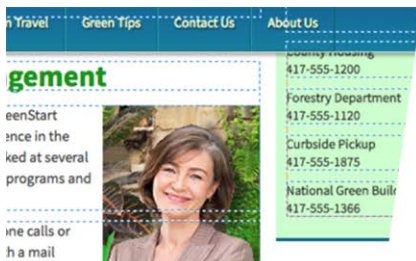
6 Select **green-styles.css** > GLOBAL.

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**

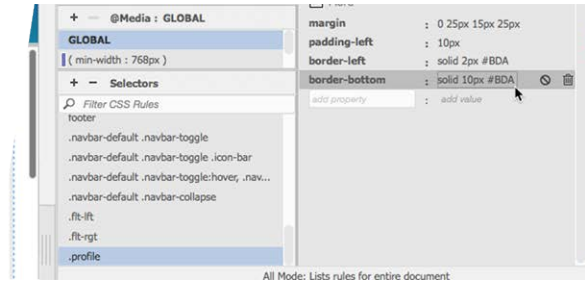
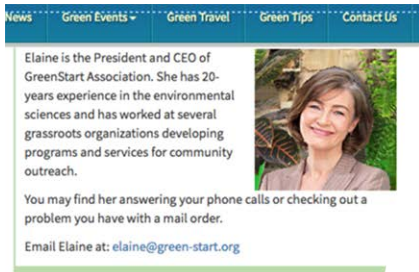
Create the following properties:

margin: 0 25px 15px 25px

padding-left: 10px

border-left: solid 2px #BDA

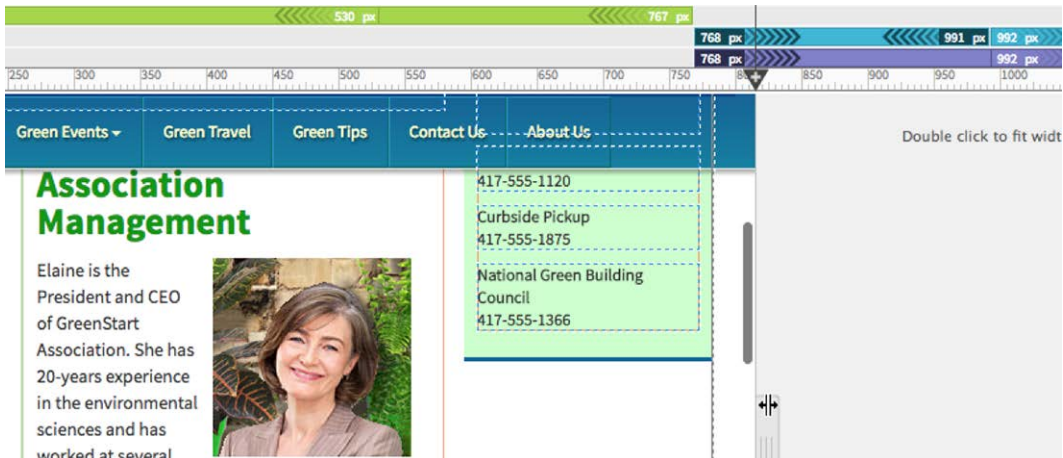
border-bottom: solid 10px #BDA



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 If necessary, 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 breakpoints: extra-small, small, medium, large, and extra-large. At this moment, the layout breaks only at a width of 767 pixels (medium).

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-md-6`. The part “col” refers to “column,” the part “md” refers to “medium,” and the part “6” refers to the number of grid sections. That means, in the current layout, the class `col-md-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.

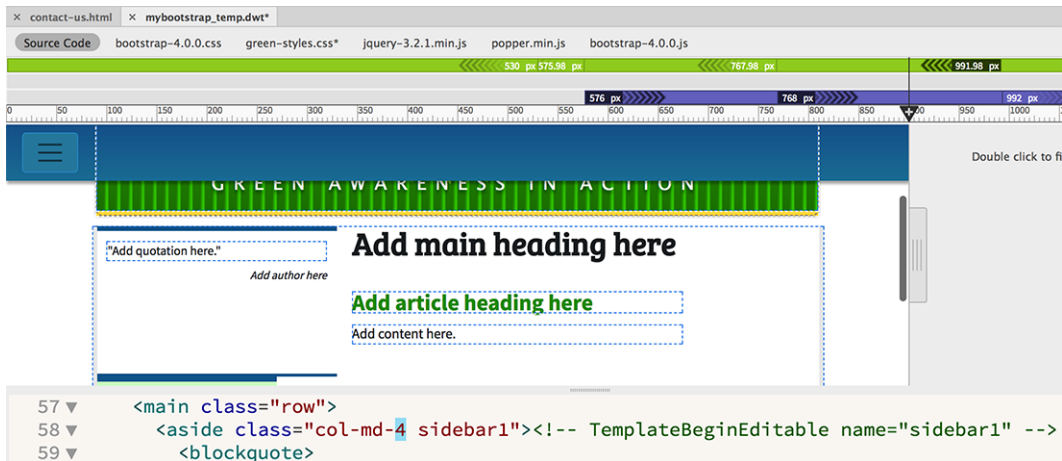
- 2 Open **mybootstrap_temp.dwt** in Split view. Make sure the document window is at least 1100 pixels in width.

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 up 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 medium breakpoint. You'll find that Live view doesn't respond properly with the template markup in place. The following changes will have to be made in the Code view window.

- 4 Locate the `<aside class="col-md-3 sidebar1">` element in the Code view window (around line 58).
- 5 Edit the class as highlighted: `.col-md-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 Locate the `<section class="col-md-6">` element (around line 63).
- 7 Edit the class as highlighted: `.col-md-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.

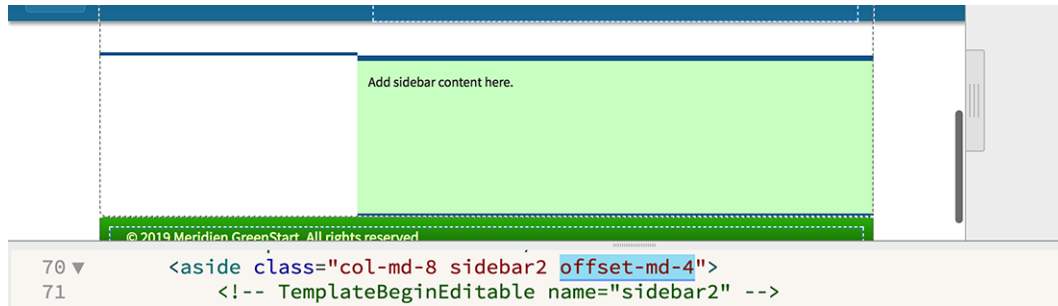
8 Locate the `<aside class="col-md-3 sidebar2">` element.

9 Edit the class name as highlighted: `.col-md-8`

The class change affects the width of the element. But to shift it over to the right, you will need to add another class.

10 Edit the class for Sidebar 2 as highlighted:

`<aside class="col-md-8 sidebar2 offset-md-4">`



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 medium 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 Locate the `<aside class="col-md-4 sidebar1">` element.

13 Edit the class as highlighted:

`<aside class="col-md-4 sidebar1 col-lg-3">`



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

14 Locate the `<section class="col-md-8">` element.

15 Edit the element class as highlighted:

```
<section class="col-md-8 col-lg-6">
```

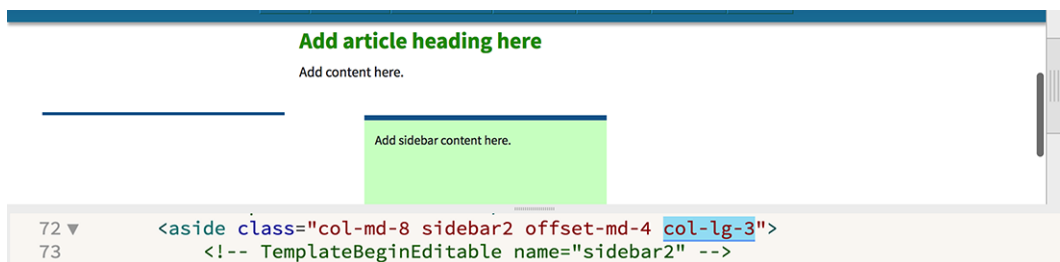


The main content area now returns to its previous size, leaving an open spot for Sidebar 2.

16 Locate the `<aside class="col-md-8 sidebar2 offset-md-4">` element.

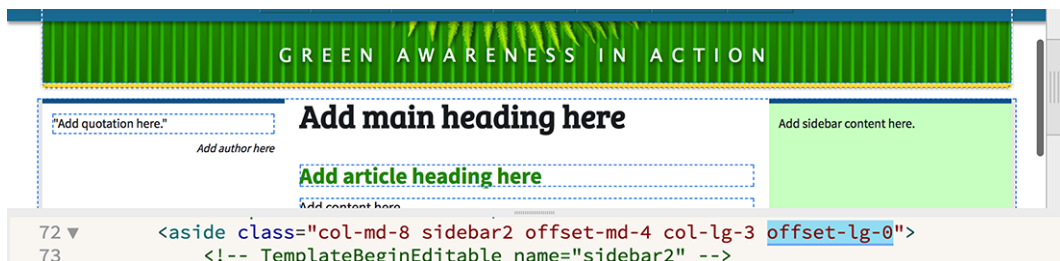
Add the following classes to Sidebar 2:

```
<aside class="col-md-8 sidebar2 offset-md-4 col-lg-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 offset 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: **offset-lg-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 narrows.

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

The layout converts from one to two to three columns as the window opens.

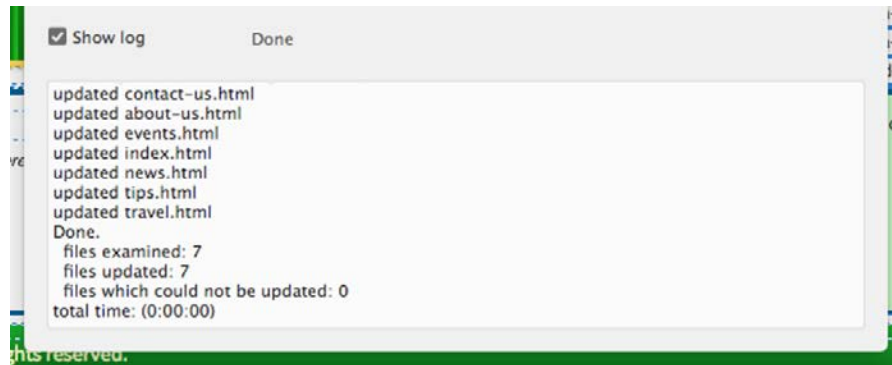
Once you are certain the styling is working successfully, you can apply it to all the pages in the site.

- 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 Update dialog. 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.

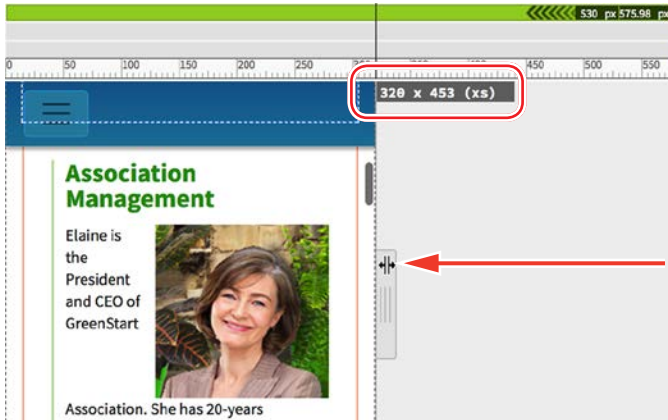
- 1** Drag the scrubber to 900 pixels.

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 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.

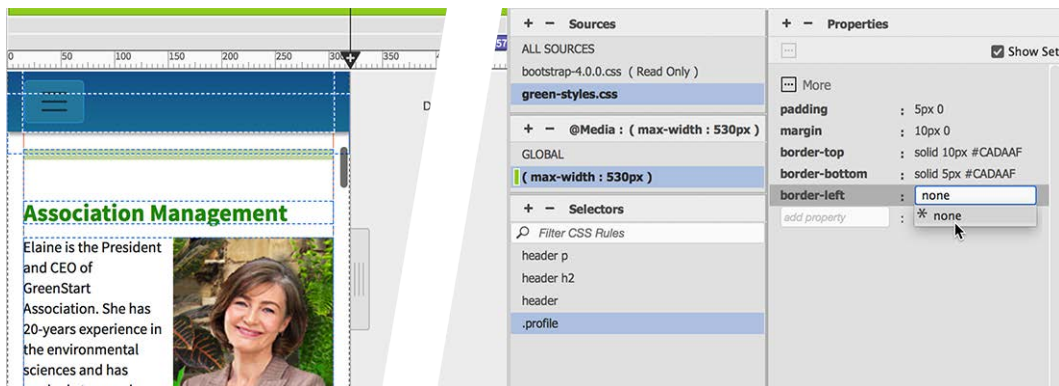


At 320 pixels, the layout is too cramped. The borders and indents are using up valuable space and no longer enhancing the content. Let's adjust the layout for the smallest screens.

- 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:

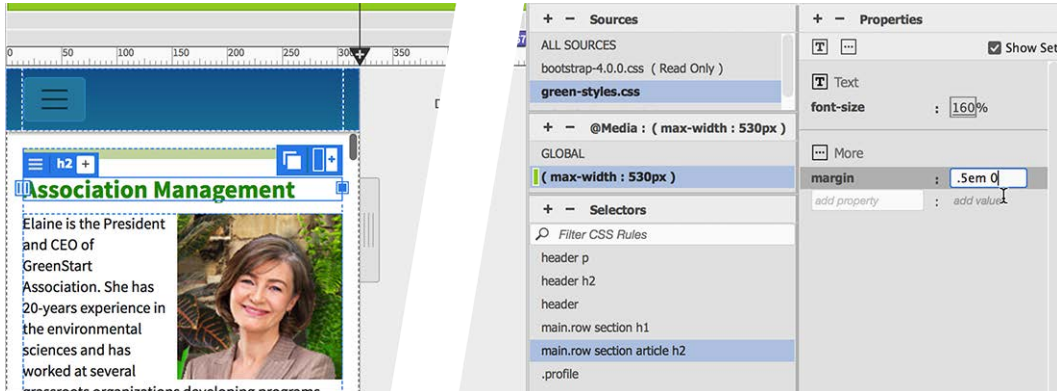
padding: 5px 0
margin: 10px 0
border-top: solid 10px #CADAAF
border-bottom: solid 5px #CADAAF
border-left: none



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.

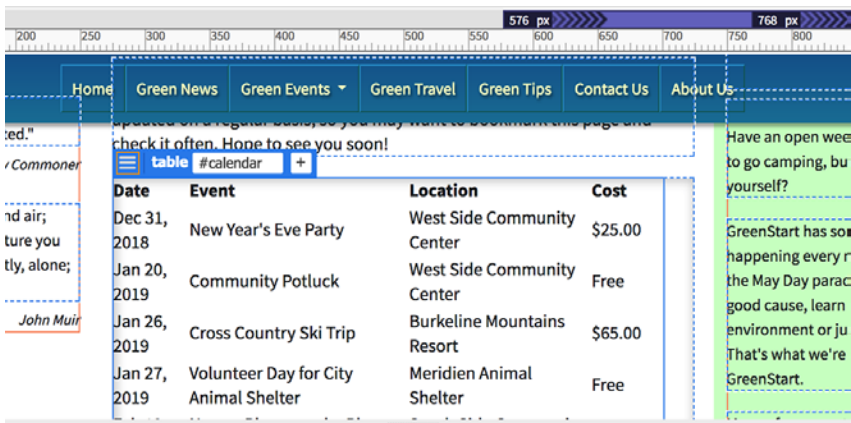
Adapting tables to a Bootstrap layout

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.



As soon as the page opens, you can see that 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.

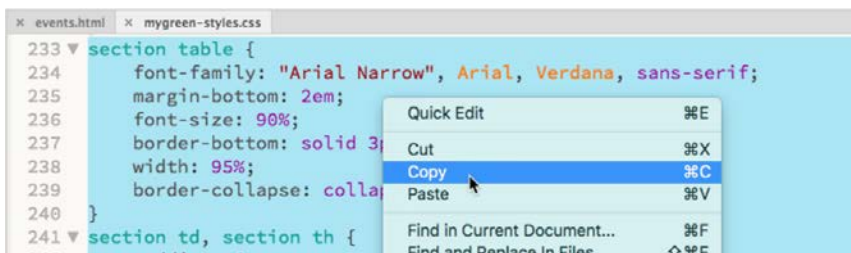
- 2 Open **mygreen-styles.css**.

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` (around line 233) down to the rule `table caption` (around line 284). 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.



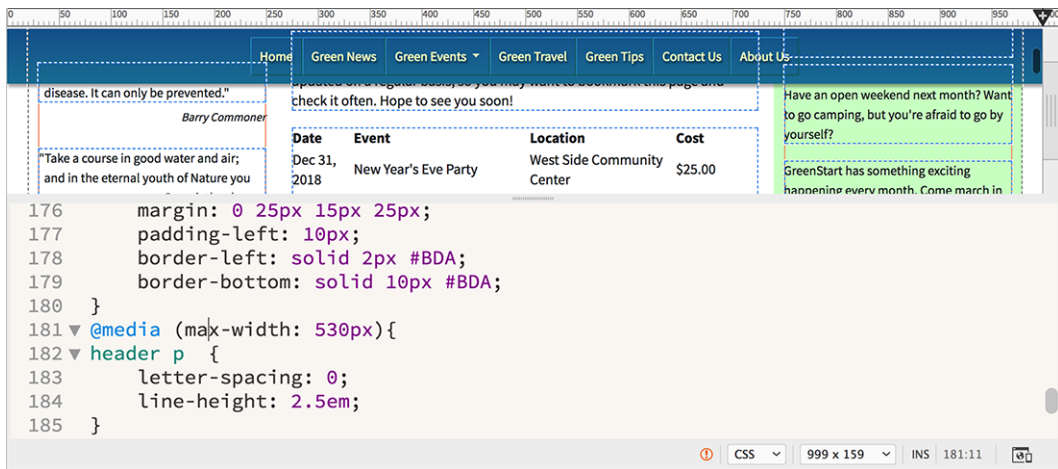
5 Switch to **events.html**.

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.

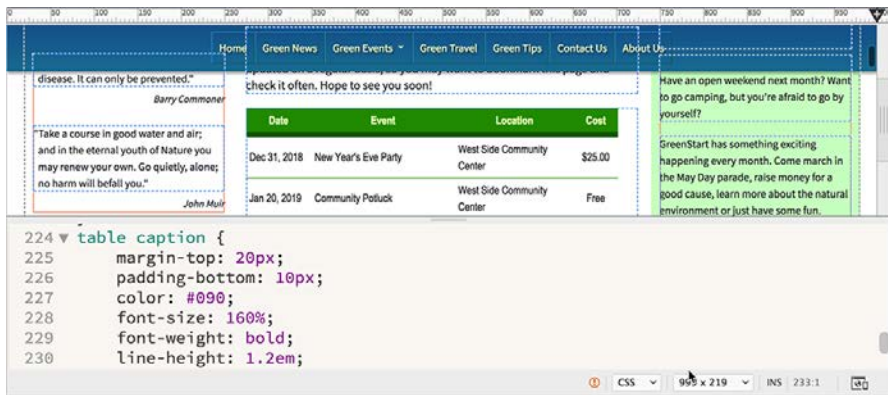
7 In the Code view window, scroll down to the entry `@media (max-width:530px)` (around line 181).



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 are contained after this opening brace and before the closing curly brace (}), which appears, in this case, around line 209.

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 in the media queries.

8 Insert the cursor before the media query and press Ctrl+V/Cmd+V to paste the CSS markup. Press Enter/Return to move the entry `@media (max-width:530px)` to a new line after the markup you just pasted, if necessary.



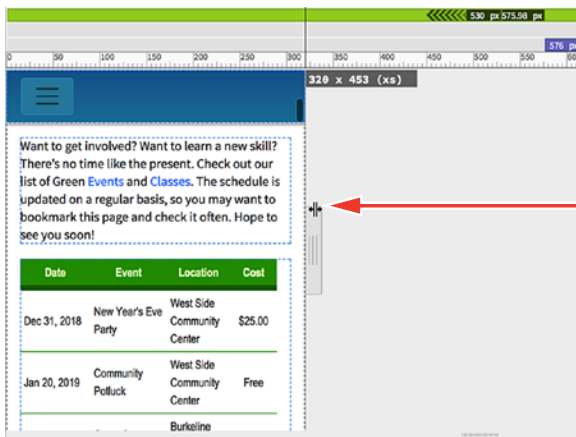
9 Save all files.

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 is 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.

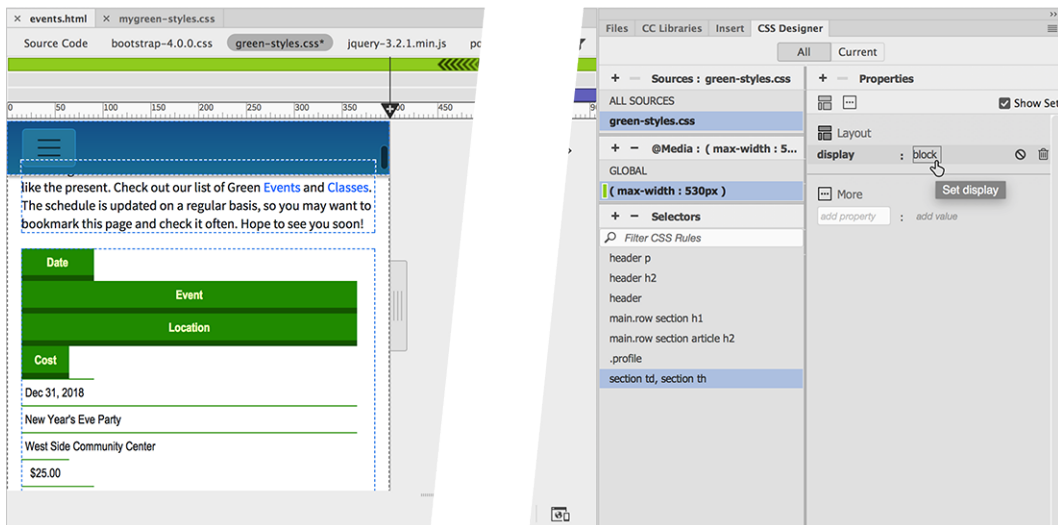
- 4 Open the CSS Designer, if necessary.
Select **green-styles.css** > (max-width: 530px).

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.

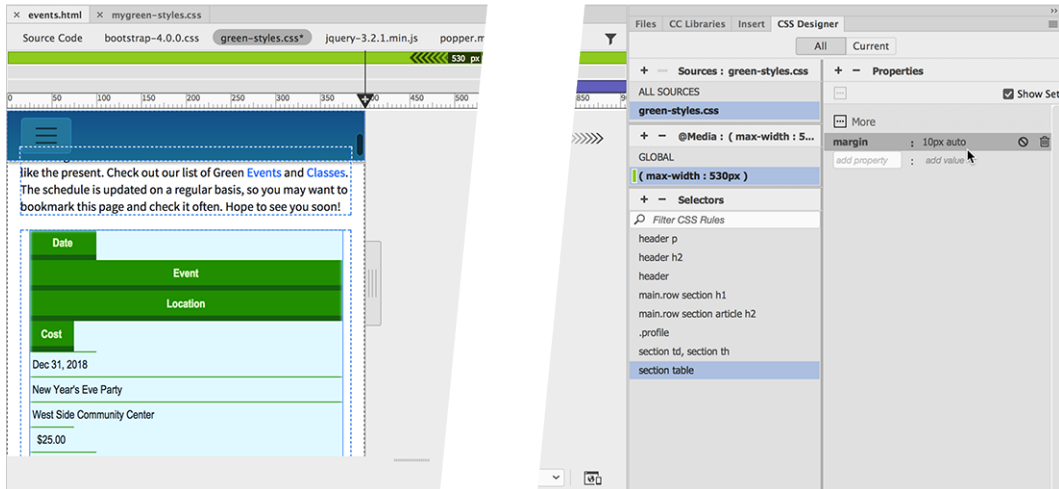
- 6 Add the **display: block** property to the new rule.



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**

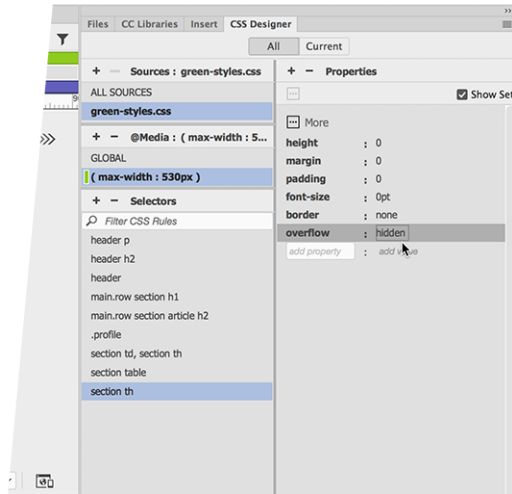
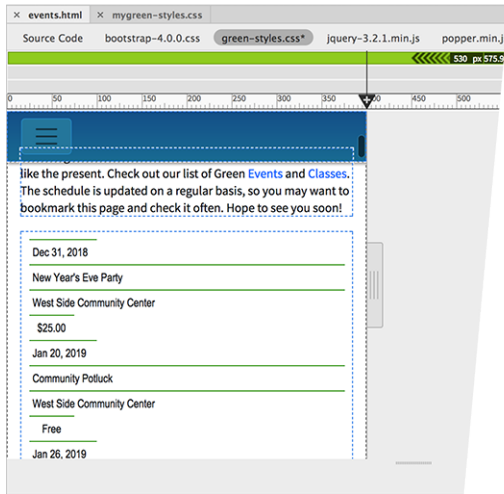


The tables are now centered in the layout.

With the data stacking vertically, it doesn't make much sense now to have a header row. You could set the header row to the `display:none` property 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



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

● **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.

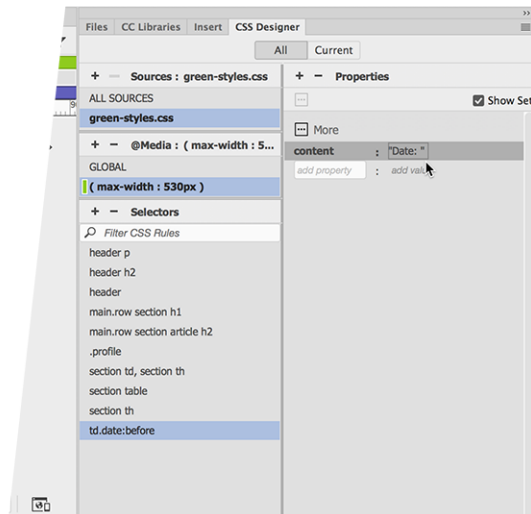
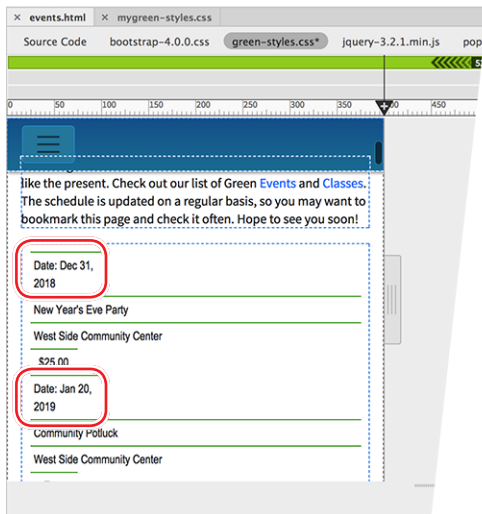
The header rows disappear visually but are still accessible to visitors using screen readers or other assistive devices. But now that they are invisible, you have to address the fact that there are no headers describing the data being displayed.

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.

9 Create the following rule: **td.date:before**

10 Enable the Show Set option.

Enter the following property–value combo: **content: "Date: "**

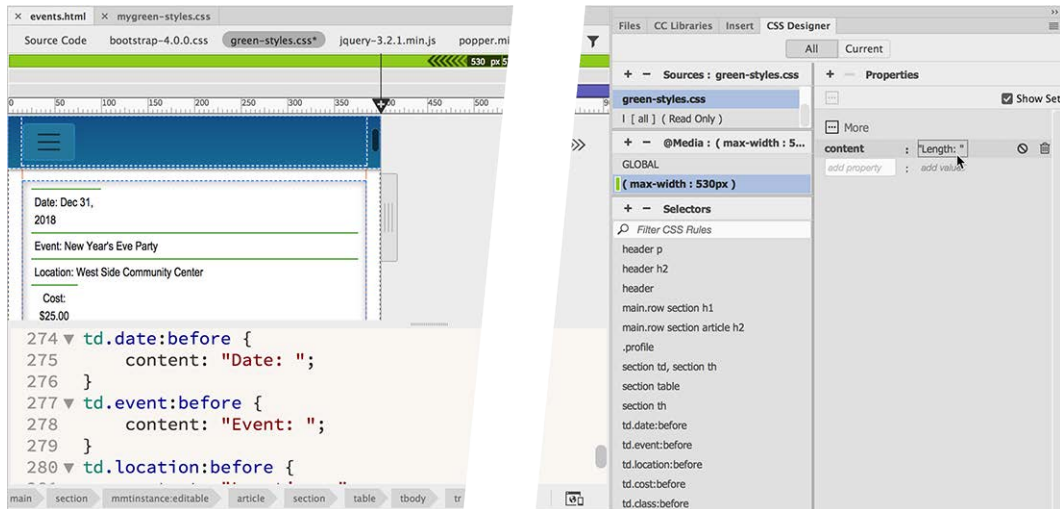


Notice that the label **Date:** appears in all the cells styled by the `date` class. You need to make a similar rule for each of the data elements.

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

| Rule | Property: Value |
|------------------------------------|---------------------------------------|
| <code>td.event:before</code> | <code>content: "Event: "</code> |
| <code>td.location:before</code> | <code>content: "Location: "</code> |
| <code>td.cost:before</code> | <code>content: "Cost: "</code> |
| <code>td.class:before</code> | <code>content: "Class: "</code> |
| <code>td.description:before</code> | <code>content: "Description: "</code> |
| <code>td.day:before</code> | <code>content: "Day: "</code> |
| <code>td.length:before</code> | <code>content: "Length: "</code> |

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



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

- 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

Note: If you make a single selector, as shown in step 12, 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.

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 Referenced file interface at the top of the document window.

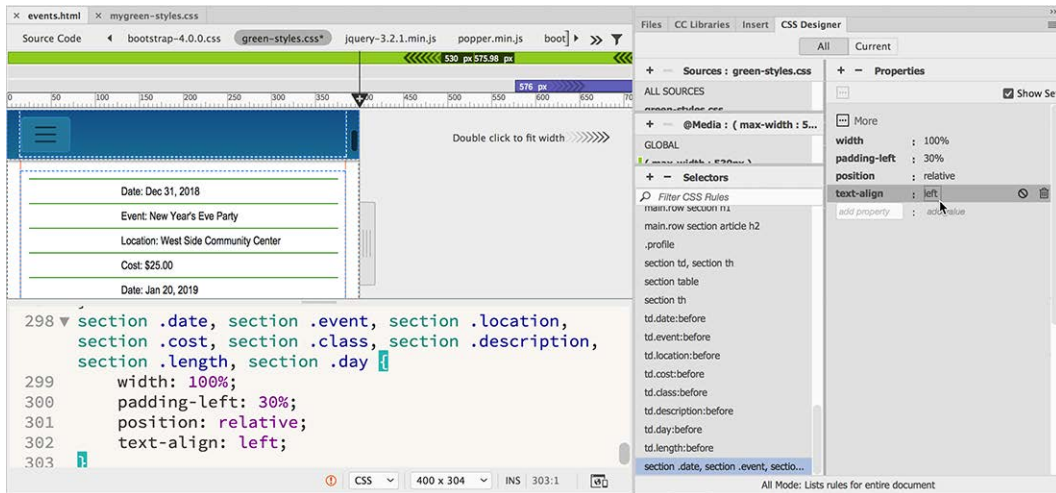
into a single rule, separated by commas. Remember to mind the punctuation and spelling carefully. Even a tiny error in the code can cause the formatting to fail. If you want the styling to be different in one or more of the elements, then create eight separate rules.

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

● **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.

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

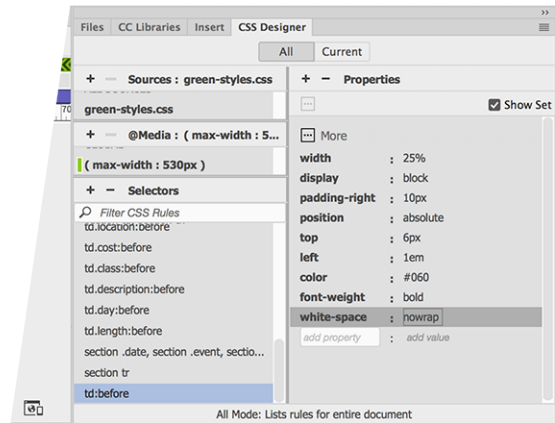
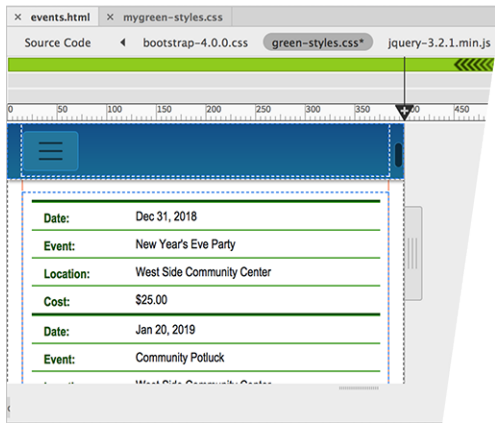
width: 100%
padding-left: 30%
position: relative
text-align: left



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.

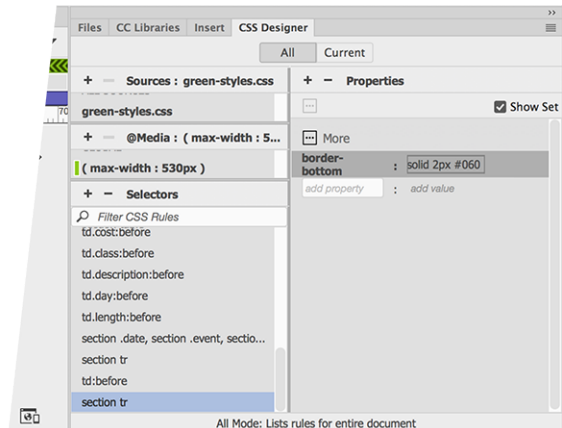
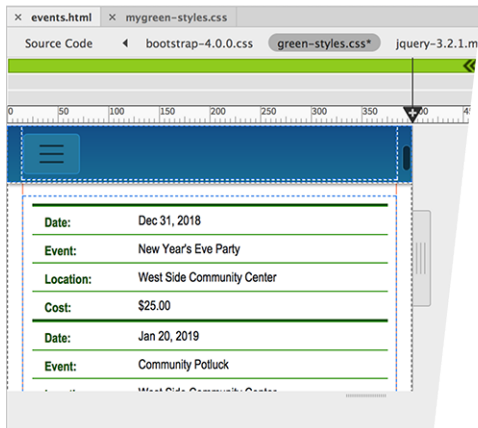
- 14 Create the following rule: **td:before**
Give the new rule the following properties:

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



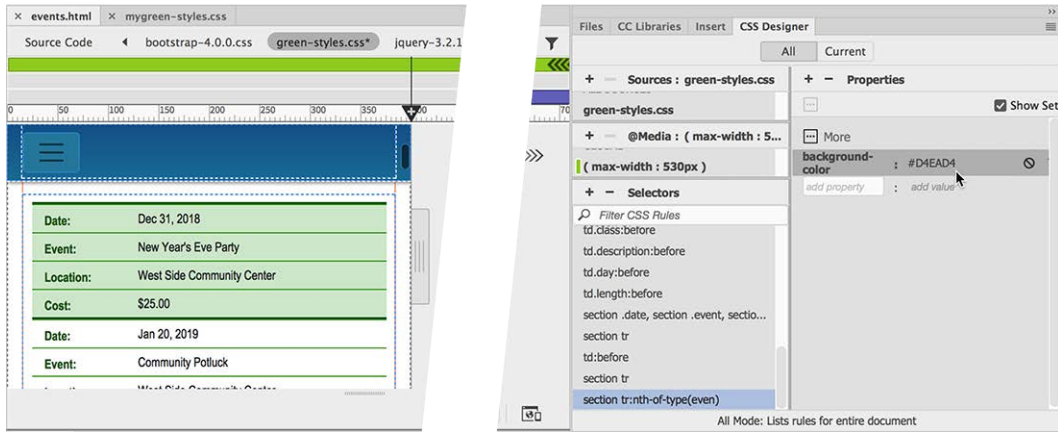
The labels now appear separated from the data and are styled in boldface and dark green. The only thing left to do now is differentiate one record from the next. One way is to simply add a darker border between each table row.

- 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 and make the data easier to read.

- 16** Create the following rule: **section tr:nth-of-type(even)**
Give the new rule the following property:
background-color: #D4EAD4



● **Note:** 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.

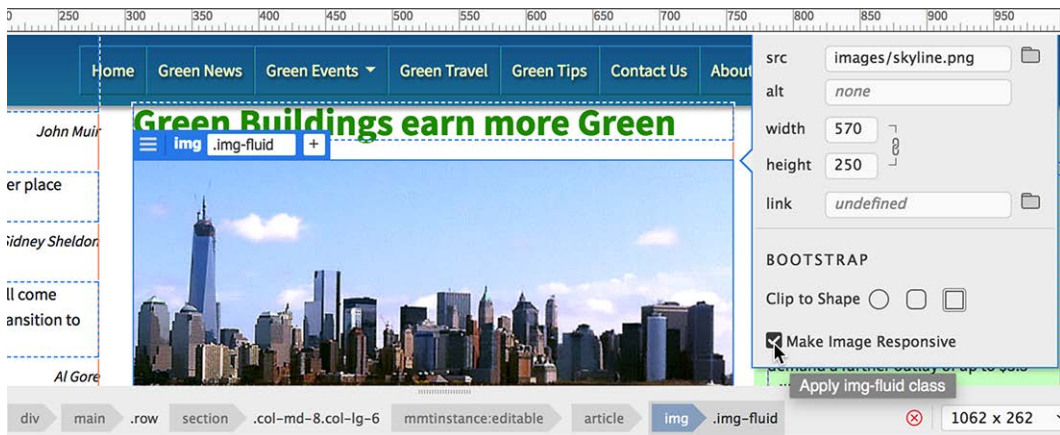


The Element Display appears on the image focused on the `img` element. The image is too large for the column. When it's selected, you can see that part of it is obscured behind Sidebar 2. Luckily, Dreamweaver has a new built-in way to deal with just this situation.

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

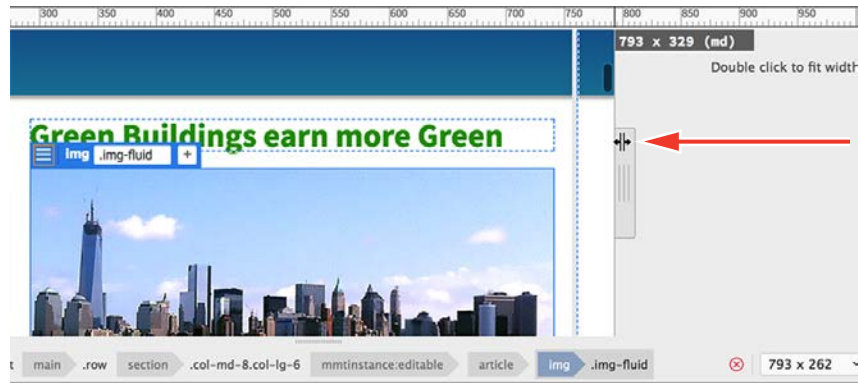
The Quick Property inspector appears.

- 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-fluid` 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.

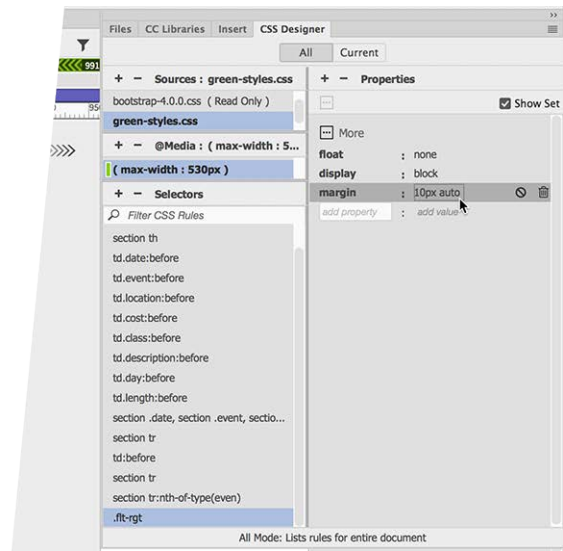
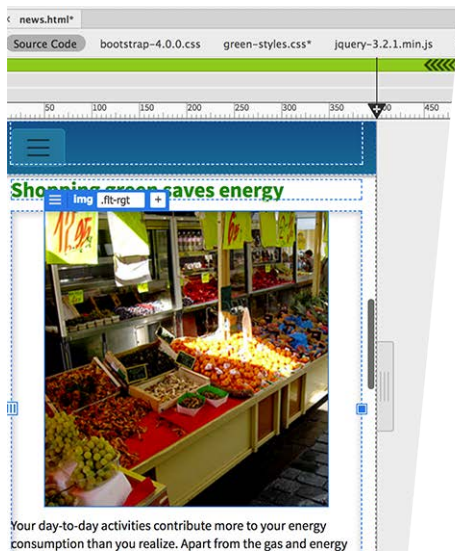
The second image displays acceptably until the width drops below 400 pixels. Then, there's not enough space on the left side of the image for the text to wrap effectively. On the smallest screens, you should turn off the float property and center the image.

- 7 Drag the scrubber to 400 pixels.
- 8 Choose **green-styles.css** > (max-width: 530px).
Create the following selector: **.flt-rgt**

The selector combines two classes to target this one image.

- 9 Create the following properties:

```
float: none  
display: block  
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 and close 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.

- 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 class field and press Enter/Return to complete the class name.



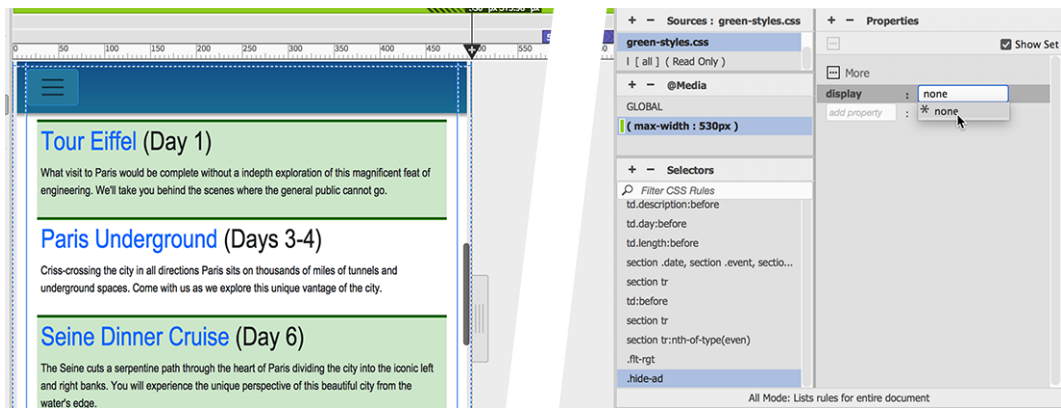
The CSS Source dialog appears.

● **Note:** If for any reason the CSS Source dialog does not appear, you must create the class reference in **green-styles.css** manually.

- 8 If necessary, select **green-styles.css** from the Select A Source menu.
Select `(max-width: 530px)` from the Select A Media Query menu.
Click away from the CSS Source dialog.

The CSS Source dialog closes. The new class should be added to the media query.

- 9 In the CSS Designer, select **green-styles.css** > `(max-width: 530px)`.
Create the following property in the rule `.hide-ad`:
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.”

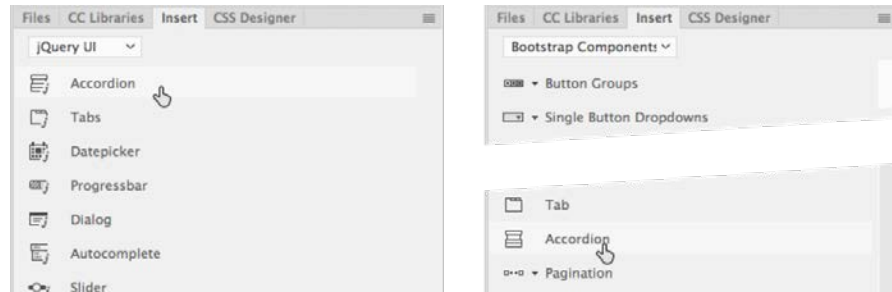
Trouble-shooting an interactive element

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.



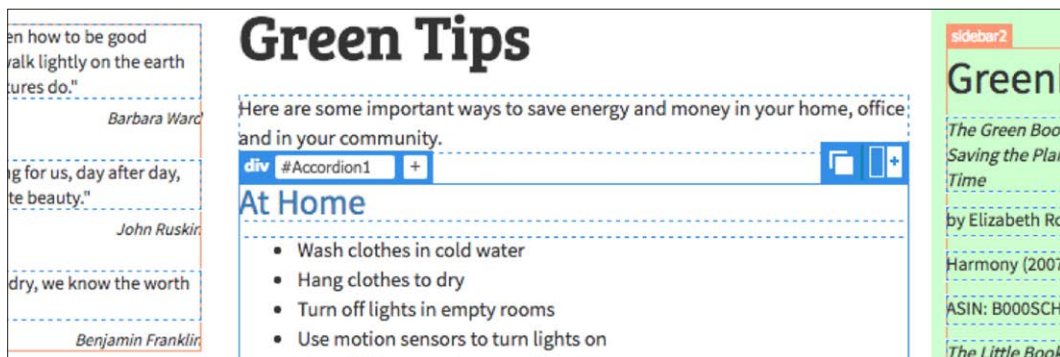
It's impossible to avoid conflicts like this. If you work on the web long enough, you will encounter many similar issues. So it's a good thing that you experience that kind of conflict here, where you can learn how to fix it. Let's take a look at the jQuery accordion widget on the *Green Tips* page.

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.

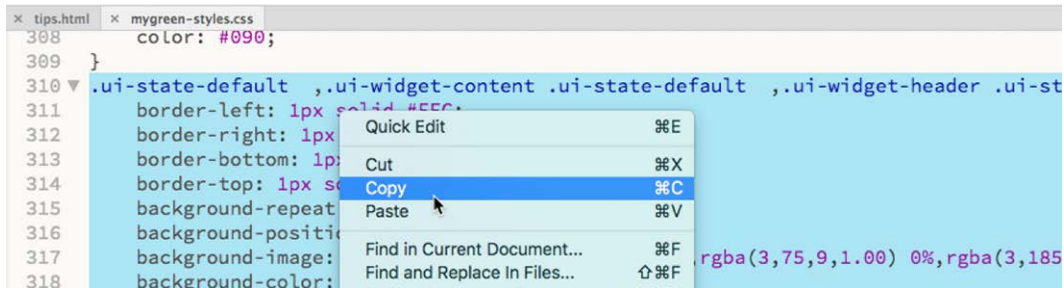
- 1 Open **tips.html** in Live view.
Make sure the document window is at least 1100 pixels in width.

The first thing you'll notice is that the styling for the accordion did not make it over to the new layout. As with 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.



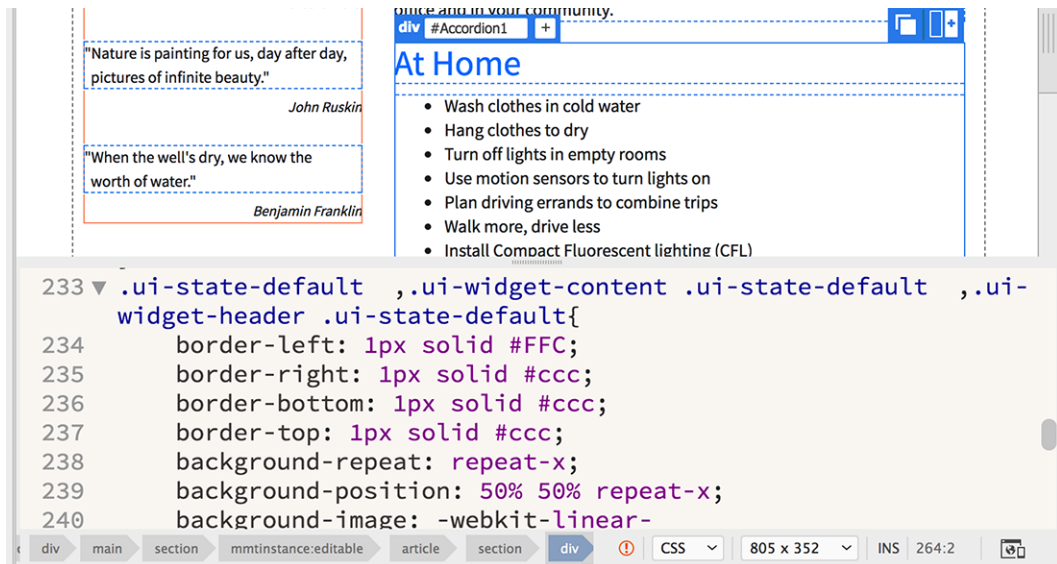
- 5 Switch to **tips.html**. Select **green-styles.css** in the Related Files interface.

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 (max-width: 530px)` (around line 233).

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

- 7 Insert the cursor before the media query. Paste the CSS code. Press Enter/Return to move the entry `@media (max-width: 530px)` 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/Templates folder in Code view.

The Bootstrap jQuery library (`jquery-3.2.1.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 the code. Look for an HTML comment and `<script>` tag loading the Bootstrap library `jquery-3.2.1.min.js` (around line 22).
- 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.

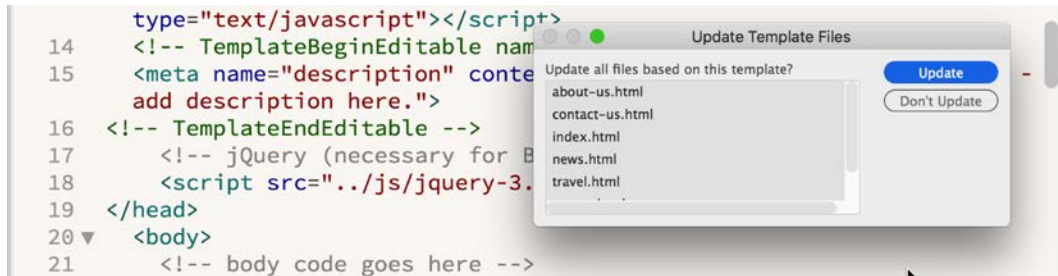
- 11 Scroll up to the closing `</head>` tag (around line 17).

● **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.

Notice the editable region just above the closing `</head>` tag. In the template, the editable region contains only the meta description, 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 closing `</head>` tag (around line 17).

- 13 Paste the code you 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 Update Template Files dialog. Close the template. Switch to **tips.html**.

- 16 Switch to Code view. 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.

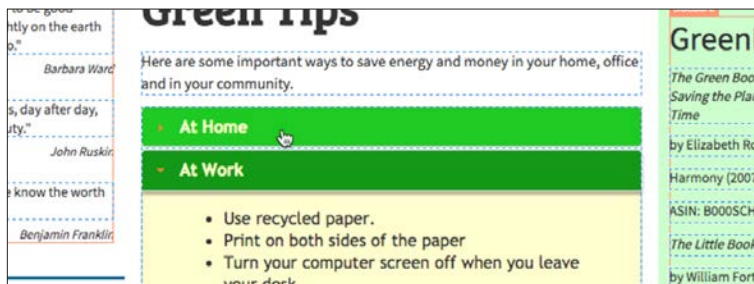
- 17 Delete the jQuery UI library script element:

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

```
17 <!-- InstanceBeginEditable name="head" -->
18 <meta name="description" content="Meridien GreenStart Association - Learn
the best eco-tips for your home, office and your community">
19 <script src="jQueryAssets/jquery-1.11.1.min.js"></script>
20 <script src="jQueryAssets/jquery.ui-1.10.4.accordion.min.js"></script>
21 <!-- InstanceEndEditable -->
22 <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
23 <script src="js/jquery-3.2.1.min.js"></script>
24 </head>
```

```
17 <!-- InstanceBeginEditable name="head" -->
18 <meta name="description" content="Meridien GreenStart Association - Learn
the best eco-tips for your home, office and your community">
19
20 <script src="jQueryAssets/jquery.ui-1.10.4.accordion.min.js"></script>
21 <!-- InstanceEndEditable -->
22 <!-- jQuery (necessary for Bootstrap's JavaScript plugins) -->
23 <script src="js/jquery-3.2.1.min.js"></script>
24 </head>
```

18 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.

● **Note:** If your accordion is not formatted properly at this point, make sure you copied and pasted all the rules you created back in Lesson 10 to format 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.

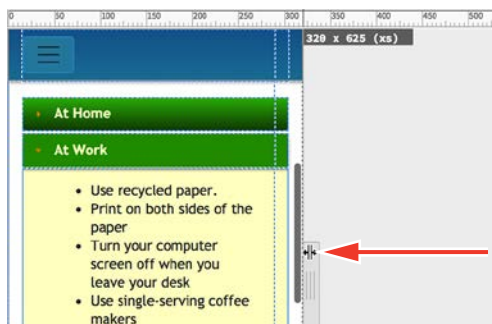
1 If necessary, switch to Live view. Drag the scrubber to 800 pixels.

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 can be indented on the `` element, the `` element, or both.

- 6 Select the `ul` tag selector.

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

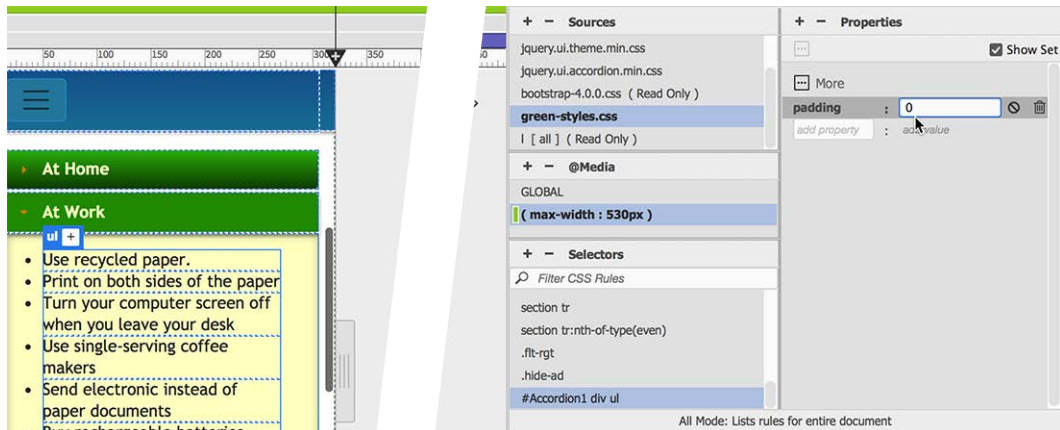
Select **green-styles.css** > (max-width: 530px).

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: 530px), it will kick in only on the smaller screens and devices.

- 8 Press Enter/Return as necessary to create the selector.

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 on any mobile devices you have at hand. Dreamweaver makes this process easy too.

Previewing pages using Real-Time Preview

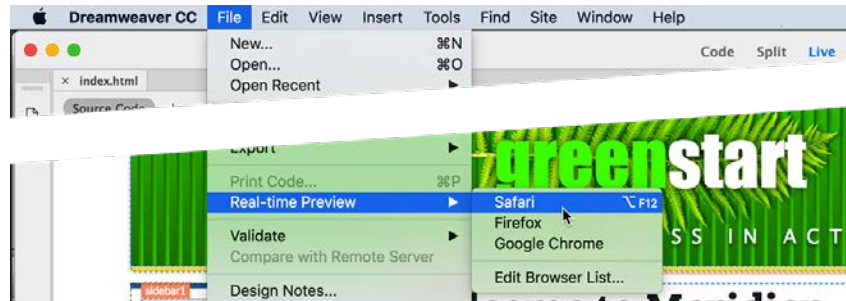
Since Live view is based on the WebKit web browser engine, it provides a pretty good facsimile of your pages and content. But nothing can take the place of an actual browser, smartphone, or tablet. You cannot be absolutely sure that your pages and design work properly until you have tested them all in 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 choosing File > Real-Time Preview > Edit Browser List.

- 2 Choose 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 re-launch Dreamweaver and try again.



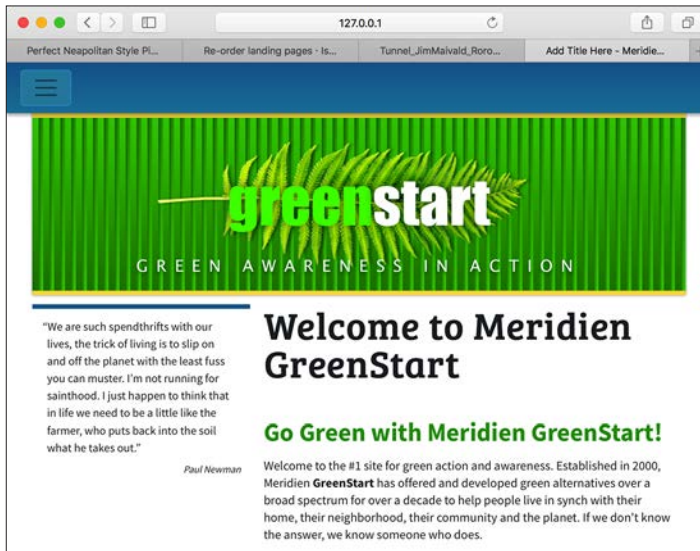
The page opens in the browser, displaying the content in three columns.

- 3 Observe the entire page and all the content.



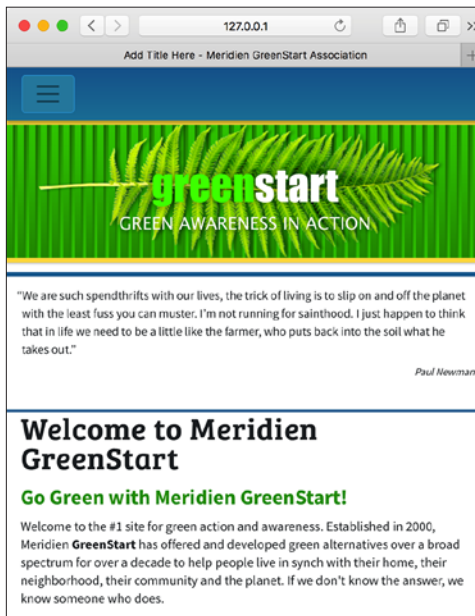
You should also test each page at various screen widths.

- 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.

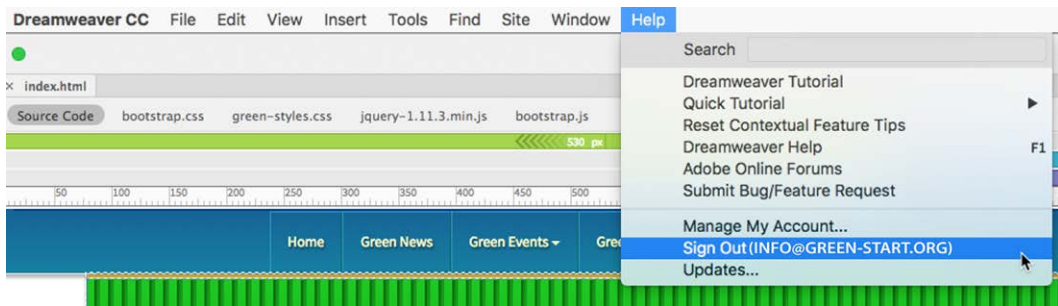
- 8 Test the *Green News* page at various widths.

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 a web server where you can upload all your pages?”

There’s no need to have your own web server when you are using Dreamweaver CC (2019 release). That’s because Real-Time Preview can upload your site to a preview area of Creative Cloud. 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, choose 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 If necessary, open **index.html**.

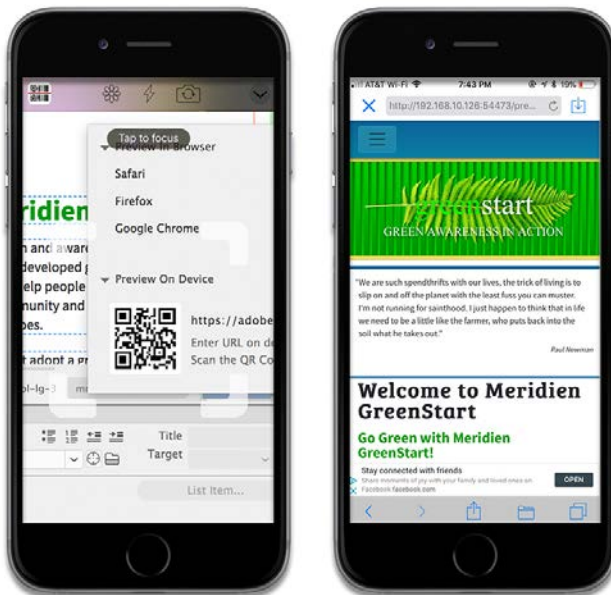
- 12 Click the Preview icon  in the lower-right corner of the document window.



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.

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



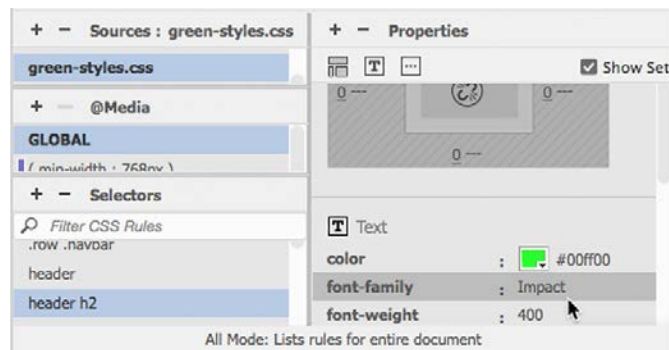
The home page should load in your device browser. The screenshot shows the preview on an iPhone 6. As you can see, there are problems with the fonts on the page. The logo and motto are not showing the correct fonts. You are seeing firsthand the reason it's vital to test all your pages on many different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

Fixing issues for mobile design and desktop

Have you tested all your pages? Did you test them on desktop and mobile devices? In multiple browsers? Did you identify all the issues on your pages and with the content?

In this exercise, you will fix the font problem you saw in the previous exercise, as well as address several others.

- 1 In Dreamweaver click the All button.
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. The same will happen in Android phones too, because that OS has a limited set of fonts installed. If you want the logo to look right, you have to create a font stack that will support a wide range of browsers and devices.

- 2 Deselect the Show Set option in the CSS Designer.
It's easier to create a font stack when this option is turned off.
- 3 Click the font-family property.
The font stack pop-up list appears.

4 Click Manage Fonts.

Create a custom font stack with the following fonts:

Impact

HelveticaNeue-CondensedBlack

Roboto Black

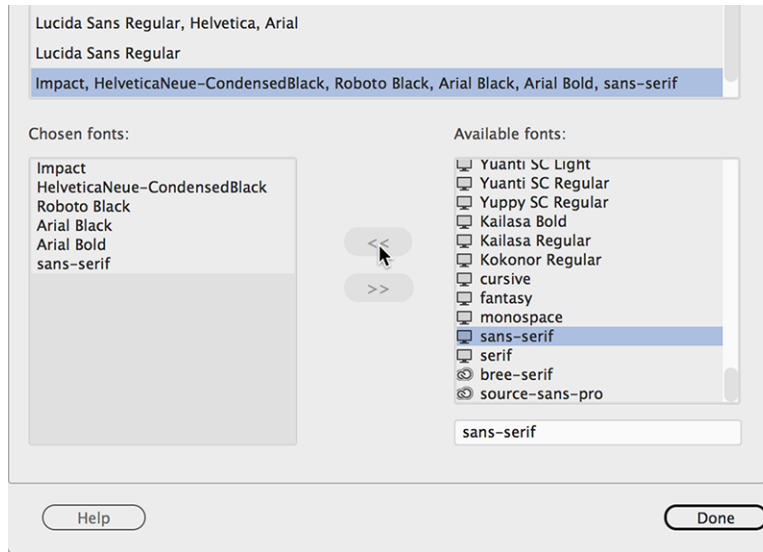
Arial Black

Arial Bold

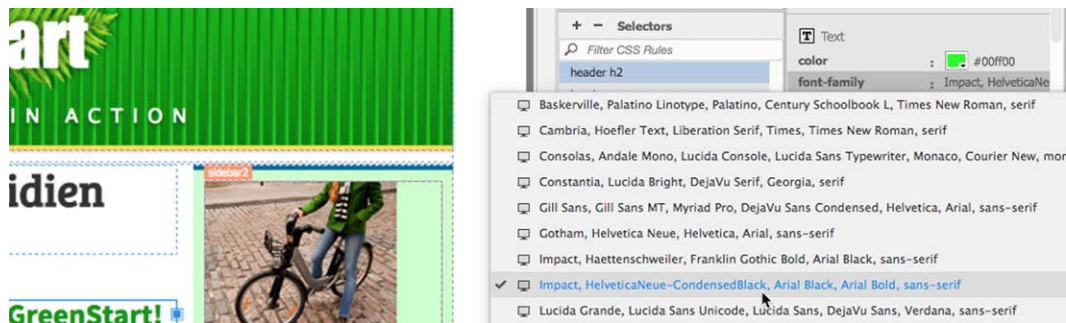
sans-serif

● **Note:** Remember that you can enter any font name manually even if it's not installed on your computer.

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



5 Select the new font stack for the rule header h2.



Now let's fix the motto.

- 6 Repeat steps 3 and 4 to create a new stack for the rule header p.

Add the following fonts to the custom font stack:

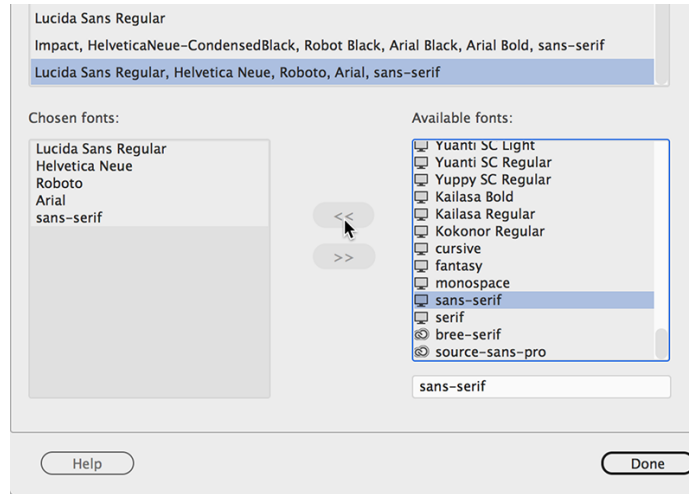
Lucida Sans Regular

HelveticaNeue

Roboto

Arial


sans-serif



- 7 Select the new font stack for the rule.

- 8 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.

- 9 Click the Preview icon  and scan the QR code with your mobile device to preview the changes on your smartphone or tablet.

As you can see from the new iPhone screenshot, the changes to the styling look much better.

- 10 Close all files.

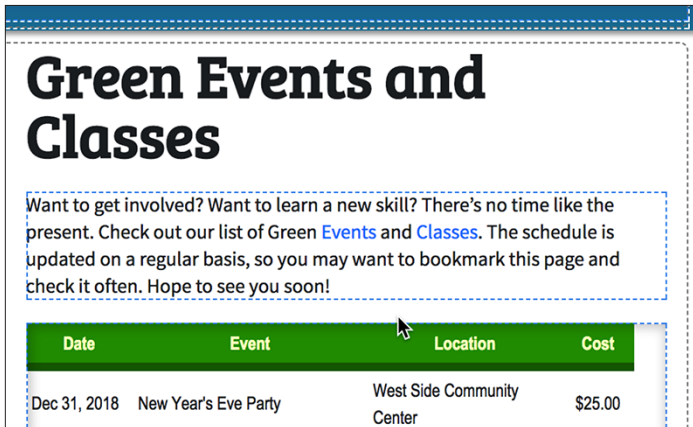
You've come a long way, but you are not finished. Next, we'll tackle an issue that is caused by the interaction of the Bootstrap layout and the table captions on the events page.



Addressing HTML5 styling

As we have seen throughout the book, HTML5 has brought many amazing things to web designers. But in this case, the new features have thrown us an unintended design issue with the table captions.

- 1 Open **events.html** in Live view from the lesson15 folder and scroll down to examine the calendar table.



| Want to get involved? Want to learn a new skill? There's no time like the present. Check out our list of Green Events and Classes. The schedule is updated on a regular basis, so you may want to bookmark this page and check it often. Hope to see you soon! | | | |
|--|----------------------|----------------------------|---------|
| Date | Event | Location | Cost |
| Dec 31, 2018 | New Year's Eve Party | West Side Community Center | \$25.00 |

You may not notice the issue if you only look at the top of the table.

- 2 Scroll down to the bottom of the calendar table.

Center

| | | | |
|--------------|----------------------|----------------------------|---------|
| Dec 15, 2019 | Holiday Party | West Side Community Center | Free |
| Dec 31, 2019 | New Year's Eve Party | West Side Community Center | \$25.00 |

caption +

2019 Event Schedule

[Return to Top](#)

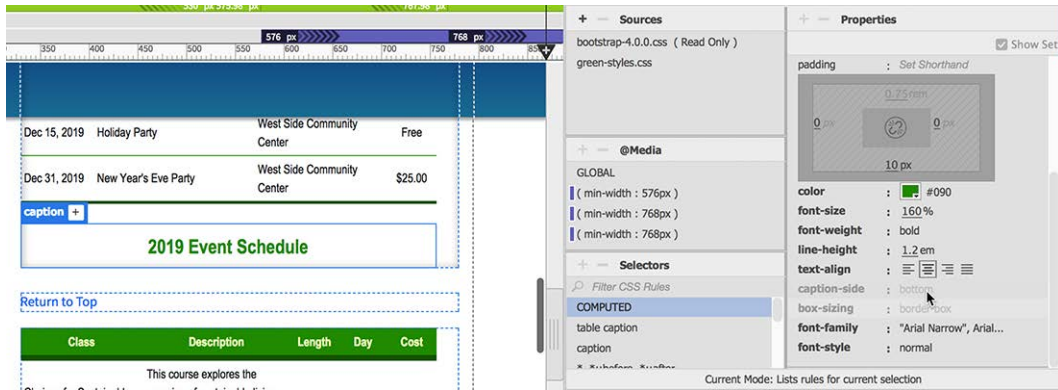
| Class | Description | Length | Day | Cost |
|-------|-------------|--------|-----|------|
|-------|-------------|--------|-----|------|

The captions for both tables are no longer displaying at the *top* of the tables, as they did in the original non-responsive HTML layout. For some reason, the captions are now appearing at the bottom of the tables in the Bootstrap-based design. Luckily, Dreamweaver provides an easy way to identify the cause.

- 3 Select the caption *2019 Event Schedule*.

The Element Display appears around the caption.

- 4 In CSS Designer, click the Current button, if necessary.
- 5 In the Selectors pane, click the COMPUTED option.
Examine the properties assigned to the <caption> element.

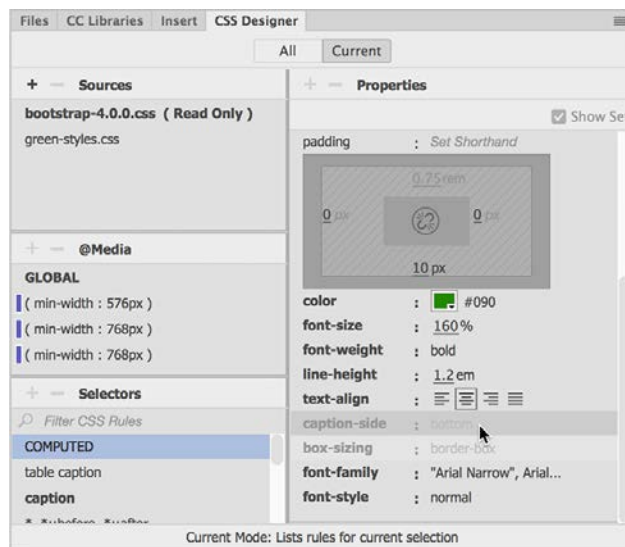


The Properties pane displays the styles compiled from all rules that are now formatting this element. The culprit affecting the <caption> element is easy to find.

● **Note:** Some users report that the Bootstrap style sheet is not marked "Read Only."

When the shift was made away from HTML attributes, CSS properties were created to replace them. Several of the properties are specific to the <caption> element. There are some recent additions in CSS3, but the one affecting the elements in your tables is older. By using the COMPUTED feature, you can see that the offending property is obviously caption-side. The item is grayed out in the CSS Designer, which indicates that it resides in a read-only style sheet.

- 6 Click the property caption-side in the Properties pane.



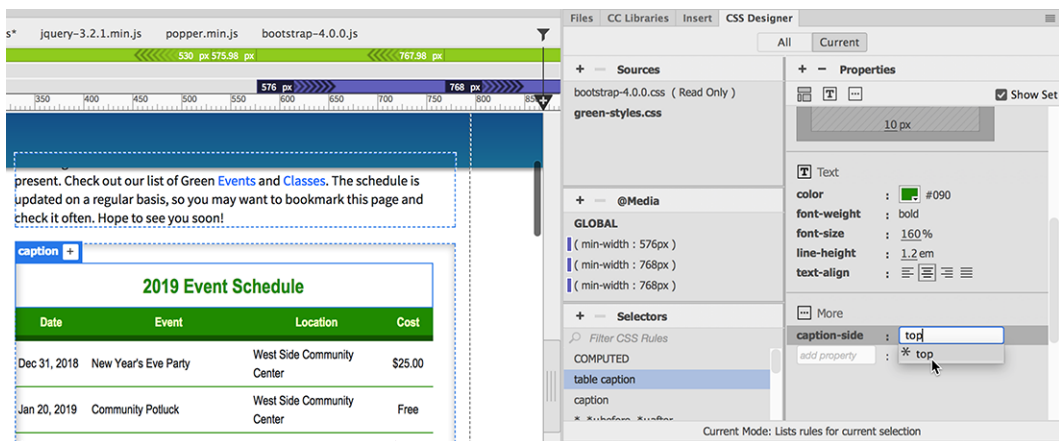
When you click it, CSS Designer highlights in bold the source and the media query, if any, the property is hosted in. In this case, the property is a GLOBAL attribute in `bootstrap-4.0.0.css`. Obviously, the creators of Bootstrap wanted captions to display at the bottom of the parent element.

Since that style sheet is read-only, you can override the property in **green-styles.css**. Notice that the first rule listed in the Selectors pane is `table caption`.

- 7 In CSS Designer, select the rule `table caption`.

This rule was created in Lesson 7 to format the table captions so that they would display more like headings. Since it is a global style, you can use that same rule to override the Bootstrap styling.

- 8 Create the property **caption-side: top** in the rule `table caption`.



As soon as the new property is complete, the captions return to the top of the tables. When you update frameworks like Bootstrap—or apply them to an existing site, like we did—you will often find many instances where elements are behaving badly. Don't take any styling for granted.

In fact, did you notice the other glaring issue in the site's styling? It affects every page and is so subtle you may have not recognized it. Give up? Take a look at the bottom border of the left and right sidebars. After we applied the Bootstrap framework, the borders started to display at the bottom of the page content, the full height of the tallest column. What's causing this?

Unfortunately, this problem won't be solved by a trip to the CSS Designer. It took several minutes of troubleshooting in a browser before I could track down the culprit in this case. This new bug was introduced by Bootstrap 4, which was recently added to Dreamweaver. Version 3 used floats to move items around to create the columns we're using. But that method has lots of issues and requires some tricks and workarounds to make it function properly in every browser and

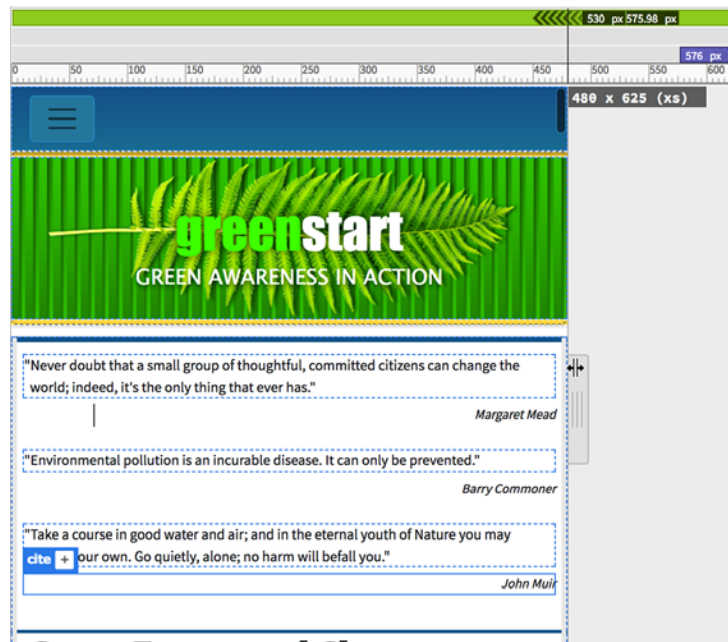
device. A technique dubbed Flexbox was developed in 2009 and added to CSS3 to address the problems designers had using floats.

The techniques used by Flexbox are too complex to discuss here, but a feature of this new layout mode is causing our issue. In Lesson 3: “CSS Basics Bonus,” we observed one of the problems with floating multiple columns of content over a colored or graphical background: If one column contains less content, it will be shorter than the others. In that bonus lesson, we used a simple solution; we just applied the same background color to all the columns and the containing element.

Designers who create multicolumn designs often want all columns to be the same height, even if they do not contain the same amount of content. They also don’t always want to apply the same background color. Enter Flexbox. The basic concept of Flexbox was to take all the guesswork and frustration out of creating HTML layouts. In this case, the current layout bug is actually the framework trying to *help* us.

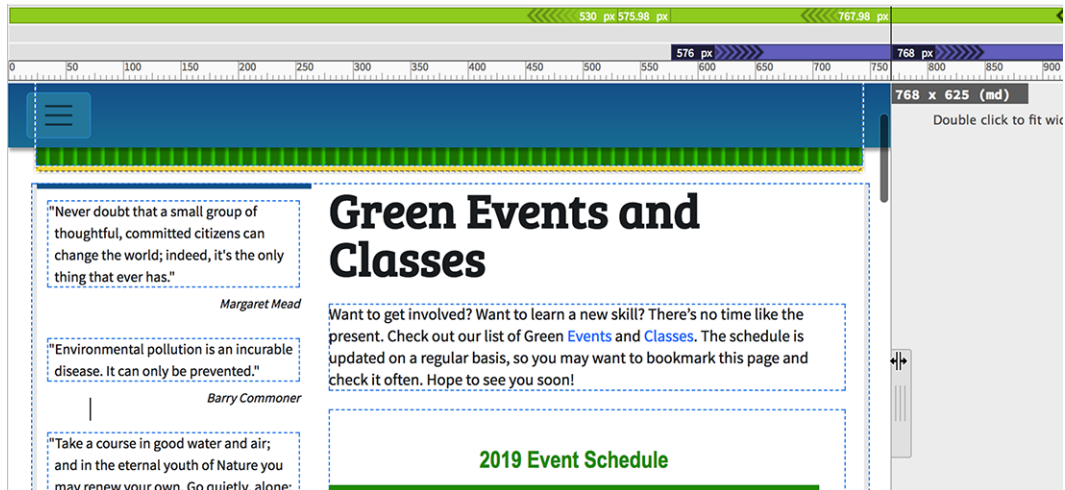
Unfortunately, the property that is creating this issue is a default setting of Bootstrap 4, which means you won’t be able to use any troubleshooting tool to track it down. Instead, we have to create a new media query and declare a new property to fix it. The first step is to identify the screen widths at which the height bug is relevant.

- 9 Drag the scrubber to the left.
Note the width where the page switches to one column.
Examine Sidebar 1 and Sidebar 2.




The layout converts to one column at 767 pixels. Note how the sidebars now collapse to the height of their content. The height bug is not affecting them in the one-column layout.

- 10 Drag the scrubber to the right until the layout appears in two columns. Examine Sidebar 1 and Sidebar 2.



At 768 pixels, the layout converts to two columns. In a two-column layout, Sidebar 1 matches the height of the main content. Sidebar 2 remains collapsed to the height of its content. So the height issue only affects the columns when the layout forms two or more columns. The solution is to create a rule that styles the column layout at hand. The two classes that need to be targeted are `.col-md-4` and `.col-lg-3`, but only at screen widths 768 pixels and larger.

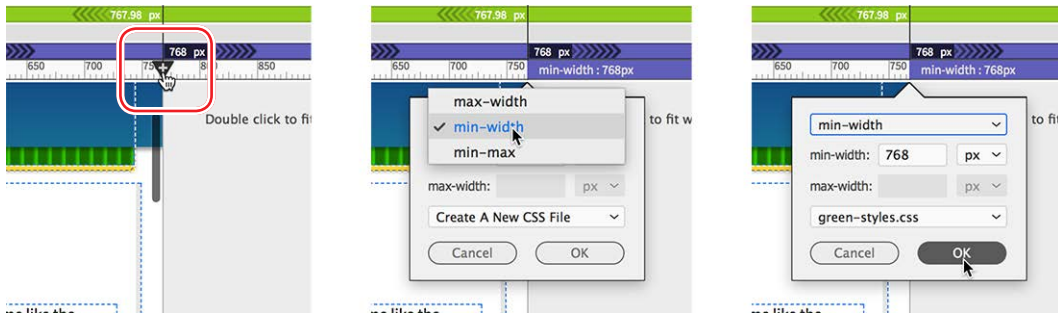
- 11 If necessary, drag the scrubber to 768 pixels.
- 12 Click the Add Media Query icon  at the top of the scrubber.

The Media Query Definition dialog appears. The `max-width` field is populated with the current scrubber position (768 px). The `min-width` field is grayed out. The new rule will need to apply to the elements 768 pixels and above, so we need to change the default settings in the dialog.

- 13 In the Media Type dropdown, select `min-width`.

As soon as you select the `min-width` option, the dialog reflects the selection, moving 768 pixels into the `min-width` field.

- 14 If necessary, select **green-styles.css** from the Source dropdown.
Click OK to create the new media query.




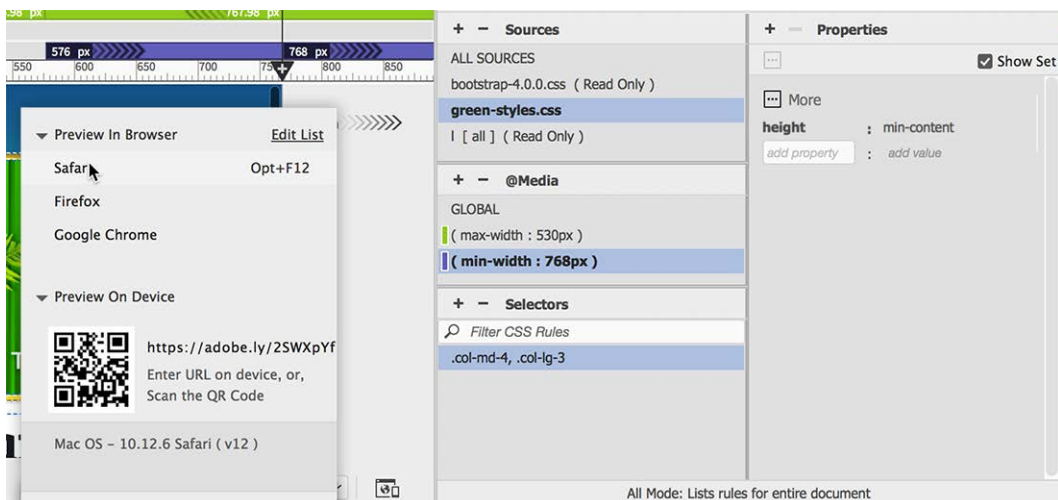
The new media query will apply its styles only when the screen is 768 pixels and wider.

- 15 In CSS Designer, select **green-styles.css > (min-width: 768)**.
Create the following selector: **.col-md-4 , .col-lg-3**

- 16 Create the following property in the new rule: **height: min-content**

The new rule may not affect the Live view display. At the time of this writing, the `min-content` value is not supported in Dreamweaver. Since all the Creative Cloud apps are updated from time to time, this property may be supported in the near future. Until that happens, this rule will format the height of the side-bars. To see the effect of the rule, you will have to use a browser.

- 17 Save all files.
- 18 Click the Preview icon  in the lower-right corner of the document window.
Select your favorite browser from the list.



The page should open in the selected browser. Check out the sidebars and change the width of the browser window to test the new styling at various widths.

19 Save and close all files.

The new rule and setting should fix the issue with the sidebar height for all screen widths above 768 pixels. To learn more about Flexbox check out <https://spaceninja.com/2015/08/24/what-is-flexbox>.

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 adapted your static HTML site for responsive design. By finishing all the exercises in this book up to this point, you have gained experience in all aspects of the design, development, and testing of a standard website compatible with desktop computers and mobile devices. In the final lesson, you will learn how to add web-compatible animation and video to your site.

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?
- 7 Can you use one set of fonts for all browsers and devices?

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.
- 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.
- 7 At this point, no single font technology will support every computer and mobile device. The best advice is to create a font stack that will target specific fonts compatible with as many browsers and devices as possible.