

That should give you a feel for some of the aspects of a site that need special attention in a responsive design. We covered content hierarchy, various layout patterns, typography tweaks, responsive navigation patterns, and image strategies, and addressed tables, forms, and interactive features. I'd say that's enough lecturing. Now you'll get some hands-on time in [EXERCISE 17-1](#).

EXERCISE 17-1. Making the bakery home page responsive

We've done a lot of work on the Black Goose Bakery site over the last few chapters, but the resulting site works best on large screens. In this exercise, we're going to back up a few steps and build it again using a small-screen-first strategy, making changes to layout, navigation, typography, and more at strategic breakpoints.

I've done the heavy lifting of writing the necessary styles for each breakpoint, but I will talk you through each step and share the reasoning for the changes. The starting style sheet (*bakery-rwd.css*) as well as the finished style sheet (*bakery-rwd-finished.css*) and the other files for the site are provided with the materials for this chapter. The HTML file, *bakery.html*, hasn't changed since we added the container element to it in **Chapter 16, CSS Layout with Flexbox and Grid**, and we will not need to edit it again.

Getting Started

Open the HTML file (*bakery.html*) in a browser with a Responsive View (see the previous sidebar, “How Wide Is the Viewport?”) so you can expand the viewport window and watch the changing pixel dimensions. **FIGURE 17-12** shows the page at 320 pixels wide with the default, narrow-screen styles that will be the starting point for this design.

The content of the page is the same as in previous chapters, but if you worked on the exercises in **Chapter 16**, you'll notice that I've changed a few styles to make the initial layout suitable for small screens. Allow me to point out the characteristics of this baseline design:

- **Layout:** The page has a one-column layout for small screens. There are no borders around the main text area, and the Hours section has a scalloped edge on the top instead of the side. That maintains the look and feel, but is more appropriate when the sections are stacked.
- **Navigation:** The navigation menu, which was created with Flexbox, couldn't flex small enough to fit across a small screen. To make it fit, I turned on wrapping (**flex-wrap: wrap;**) and set the width of each **li** to 50% so there would be two on each row. I also made it so they can both grow and shrink as needed (**flex: 1 1 50%**).
- **Conditional header text:** The tagline was taking up a lot of vertical space, and I decided it wasn't critical. I hid the paragraph (**display: none;**) and I will make it visible again when there is more room.

- **Typography:** On small screens, I decided to use a legible sans-serif font for the text and not to employ my web font because it is likely to be difficult to read at small sizes.
- **Images:** I set the **img** elements for the bread and muffin images to **display: block** so they have the full width of the viewport to themselves with no text sneaking in next to them. Setting the side margins to **auto** keeps them centered horizontally.
- **Miscellaneous:**
 - The award appears at the bottom of the page because there is not enough space for it to be positioned at the top.
 - I highlighted a **span** from the 45th to 75th characters to reveal when the line lengths get too long.

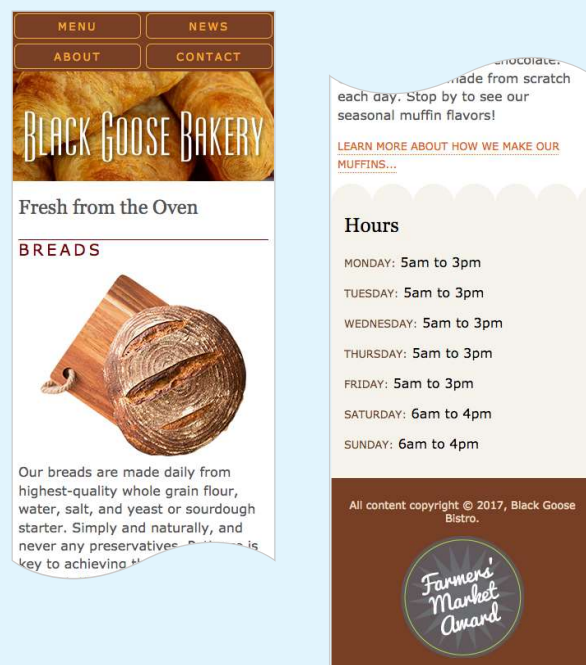


FIGURE 17-12. The small-screen design is our starting point.

Fixing the Navigation

Now we can start tailoring the design for other screen sizes. Using a Responsive View tool, I can resize the viewport and get an instant readout of the dimensions of the window. Give it a try on your browser. Keep making it wider, and you'll see that some things look OK, and some things start looking awkward pretty quickly.

One thing that looks awkward to me right away is the stacked navigation at the top. I'd like it to switch to one centered line as soon as there is room, which to my eye happens when the viewport is 400 pixels wide (FIGURE 17-13).

Are you ready to write your first media query? Open the style sheet (*bakery-rwd.css*) in a text editor. Remember that media queries need to come after other rules for the same declaration, so to keep this exercise simple, we'll add them at the end of the style sheet, before `</style>`. Add this query as you see it here. Remember to make sure you have the right number of nesting curly brackets:

```
@media screen and (min-width: 400px) {
  nav ul li {
    flex: none;
  }
  nav ul {
    justify-content: center;
  }
}
```

This tells the browser that when the page is on a screen and the viewport is 400 pixels or wider, set the “flex” of menu list items to “none.” The **none** keyword is equivalent to **flex: 0 0 auto;**, so the items are not allowed to grow or shrink and will be sized based on their content. I've centered the flexbox container by setting **justify-content: center.**

Save the style sheet and reload the page in the browser. Try resizing the viewport to see how it works at wider sizes. I think this



Before breakpoint change



After

FIGURE 17-13. The navigation started to look awkward, so I add a breakpoint at 400 pixels to switch it to one line.

centered arrangement will work for even the widest of screens, so navigation is all set. If you had navigation with additional elements such as an inline logo and a search box, you might find it best to create a few different arrangements over a number of breakpoints.

Floating Images

As I continue to make the viewport gradually wider, I notice that the main images start looking very lonely on a line alone, and that there is room to start wrapping text around them again at about 480 pixels wide. Let's take care of that awkward whitespace by floating the images to the left once the screen reaches 480 pixels (FIGURE 17-14):

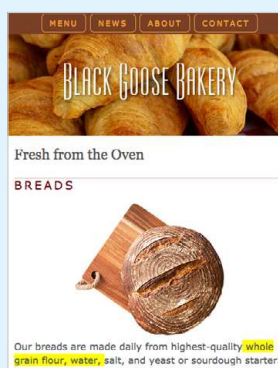
```
@media screen and (min-width: 480px) {
  main img {
    float: left;
    margin: 0 1em 1em 0;
  }
}
```

NOTE: If you like, you can include the CSS shapes from **Chapter 15, Floating and Positioning**, for a more interesting text wrap. I've omitted them here for brevity and because of limited browser support.

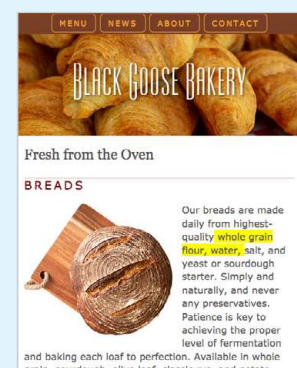
Text and Typography

Once the screen gets to be about 600 pixels wide, I feel like there is enough room to introduce some embellishments. There is room for the tagline in the header, so I'll set that to display again.

Now some attention to typography. I like the Stint Ultra Expanded web font, but it isn't key to the company's brand, so I omitted it on the narrow layout because of line length issues. At this breakpoint, I can begin using it because I know it will be more legible and result in comfortable line lengths. I've also loosened up the line height a little. I'll take advantage of the extra space to add a



A breakpoint is needed to fill in the awkward space around the image.



At 480 pixels wide, the image is floated to the left.

FIGURE 17-14. The images float left once there is enough width to accommodate wrapping text.

EXERCISE 17-1. Continued

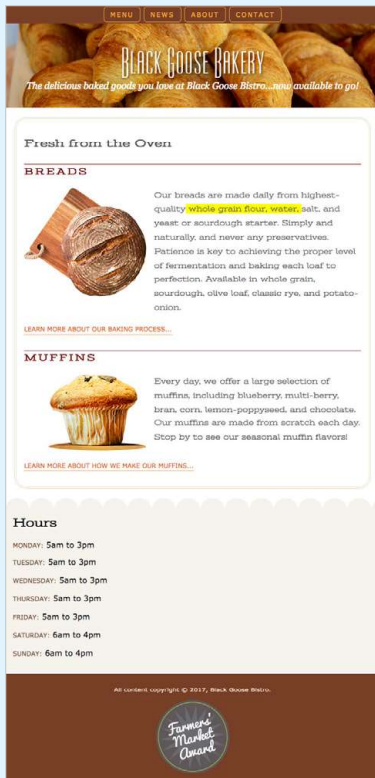


FIGURE 17-15. This medium size layout is well suited for tablet-sized devices.

rounded border around the main text area to bring it closer to the original brand identity for the site. The result is an enhanced one-column layout that is well suited for tablet-sized devices (**FIGURE 17-15**).

Here is the media query for the 600-pixel breakpoint. Add this to the bottom of the style sheet after the other two queries:

```
@media screen and (min-width: 600px) {
  header p {
    display: block;
    margin-top: -1.5em;
    font-family: Georgia, serif;
    font-style: italic;
    font-size: 1.2em;
  }
  main, h2, h3 {
    font-family: 'Stint Ultra Expanded', Georgia, serif;
  }
  h2, h3 {
    font-weight: bold;
  }
  main {
    line-height: 1.8em;
    padding: 1em;
    border: double 4px #EADDC4;
    border-radius: 25px;
    margin: 2.5%;
  }
}
```

Multicolumn Layout

As I continue to make the viewport wider and pay attention to the yellow highlighted span of characters, I see that the text line is growing longer than 75 characters. I could increase the font size or the margins, but I think this is a good point to introduce a second column to the layout. If you aren't targeting a specific device, the exact breakpoint is subjective. I've chosen 940 pixels as the point above which the page gets a columned layout.

I've simply taken the grid layout styles from the previous chapter and reapplied them here. On the **aside** element, I moved the scalloped background graphic to the left edge. In addition, I set a maximum width of 1200px on the container and set its side margins to **auto**, so if the browser window is wider than 1,200 pixels, the layout will stay a fixed width and get centered in the viewport. Finally, I absolutely positioned the award graphic at the top of the page now that there's enough room (**FIGURE 17-16**).

Add this final media query at the end of the style sheet. You can copy and paste them from the final exercise in **Chapter 16** (that's what I did) and make a few tweaks to the **#container** and **#aside** rules as shown:

```
@media screen and (min-width: 940px) {
  #container {
    display: grid;
    grid-template-rows: auto min-height 5em;
    grid-template-columns: minmax(25em, 1fr) 16em;
    grid-template-areas:
      "banner banner"
      "main hours"
      "footer footer";
    max-width: 1200px;
  }
```

```

    margin: 0 auto;
    position: relative;
  }
  header {
    grid-area: banner;
  }
  main {
    grid-area: main;
  }
  aside {
    grid-area: hours;
    background: url(images/scallop.png) repeat-y left top;
    background-color: #F6F3ED;
    padding: 1em;
    padding-left: 45px;
  }
  footer {
    grid-area: footer;
  }
  #award {
    position: absolute;
    top: 30px;
    left: 50px;
  }
}

```

And we're done! Is this the most sophisticated responsive site ever? Nope. Is there even more we could do to improve the design at various screen sizes? Certainly! But now you should have a feel for what it's like to start with a small-screen design and make changes that optimize for increasingly larger sizes. Consider it a modest first step to future adventures in RWD.

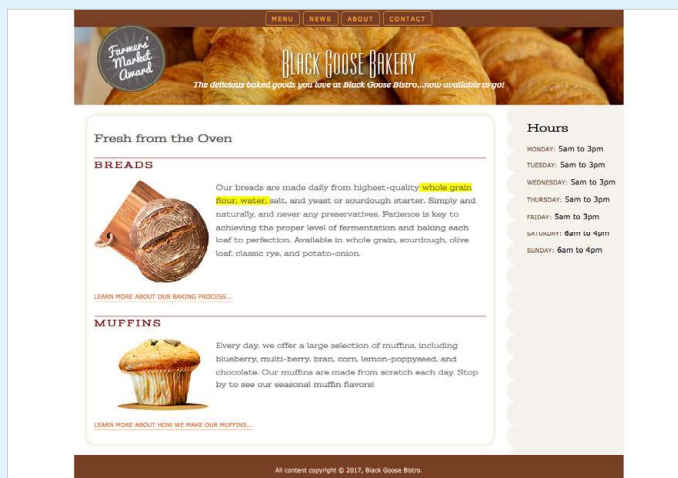


FIGURE 17-16. The two-column grid layout is appropriate for viewports over 940 pixels. On very wide screens, as shown here, the container stops expanding at 1,200 pixels wide and is centered horizontally.

NOTE

The highlighted background on the length span should be turned off before you publish, but I've left it visible in the figures so you can see how our line length is faring across layouts.