

# Responsive Styling

by Sophia



## WHAT'S COVERED

In this lesson, you will learn about the importance of responsive web design. Additionally, some techniques regarding responsive web design will be covered, including a discussion about the viewport attribute. You will learn how to rearrange content using media queries and flexbox. Lastly, you will learn how to implement a hamburger menu for mobile design and layouts.

Specifically, this lesson will cover the following:

1. [The Importance of Responsive Design](#)
2. [The Viewport](#)
3. [Rearranging Content](#)
4. [Implementing a Hamburger Menu](#)

## 1. The Importance of Responsive Design

As we discussed in a previous challenge, prior to the proliferation of cell phones equipped with web browsers, all websites were designed with the standard desktop computer screen in mind. Most HTML content was configured with fixed sizes set in pixels—not percentages—and the content was not rearranged when the browser window's size was reduced. Once cell phones started coming equipped with web browsers, site administrators needed a solution to make their site function properly on smaller screens.

Initially, the solution was to design a separate website for mobile devices, then use JavaScript to detect the browser type as mobile versus desktop, and then redirect the user to a subdomain that contains the mobile version.

🔗 **EXAMPLE** Attempting to access mywebsite.com using a mobile phone would redirect you to the subdomain m.mywebsite.com.

Today, CSS3 and HTML5 provide all of the necessary tools for making a website responsive. That is to say, a website will respond to the screen's size and rearrange its content to best fit the device's aspect ratio and orientation.

Media queries, as discussed previously, are powerful mechanisms that can react based on the user's screen size and either activate or deactivate blocks of CSS that override default CSS properties. The overriding CSS is what changes properties to rearrange the content, such as overriding flex-direction from "row" to "column" to allow sections of content to stack vertically rather than side by side. Another property that may be overridden by a media query is font size. Smaller screens with higher DPI may render text that is too small for the user to read. Instead, the font size may be increased for smaller screens.

In the next sections, we will look at the **viewport** setting, how to rearrange content based on screen size, as well as how to implement a content swap and deploy a hamburger menu for smaller screens.



### Viewport

The area of a computer program that is currently shown on the screen and an important setting for making a website responsive.

---

## 2. The Viewport

The first step in making a website responsive is to set the viewport value. The viewport value allows the browser to determine how to control the page's dimensions and scaling.

⇒ **EXAMPLE** We do this using a meta tag within the head section and give it the following HTML attributes:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Once this has been set, the browser will be able to automatically adjust elements to better fit on the screen, but this also gives it the ability to better handle media queries. In general, the reason this works is that all elements that use relative units of measurement, such as percentage and rem, need to have a parent with some sort of fixed size.

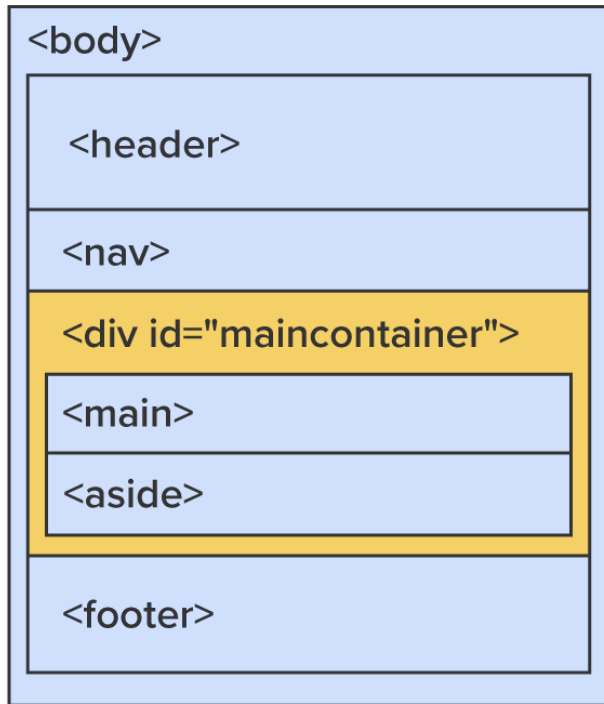
---

## 3. Rearranging Content

Returning to the page structure example wherein we used flexbox to make the main and aside sections sit side by side, we will prepare this page to adapt to narrower screen sizes by changing the flex-direction when the screen becomes too narrow.

The first step is to set the arrangement according to the initial design, that is, mobile first.

⇒ **EXAMPLE** We will set the mobile as the default CSS and make the media query adapt to a larger size.



## HTML source code:

```

<body>
  <header></header>
  <nav></nav>
  <div id="maincontainer">
    <main></main>
    <aside></aside>
  </div>
  <footer></footer>
</body>

```

## CSS source code:

```

#maincontainer
{
  display: flex;
  flex-direction: column;
}

@media all and (min-width: 480px)
{
  #maincontainer
  {
    display: flex;
    flex-direction: row;
  }
}

```

In the example, we set the `#maincontainer` as the flex container using “display:flex” and set the flex-direction to “column.” This allows the sections to stack vertically and allows for all content to fit on the screen and avoid forcing the user to scroll horizontally. Then, whenever the screen width reaches 480 pixels or more, the media query will activate its block of style rules and override the flex-direction with “row.” This allows both the main and aside sections to be displayed on screen and take advantage of the wider viewport.

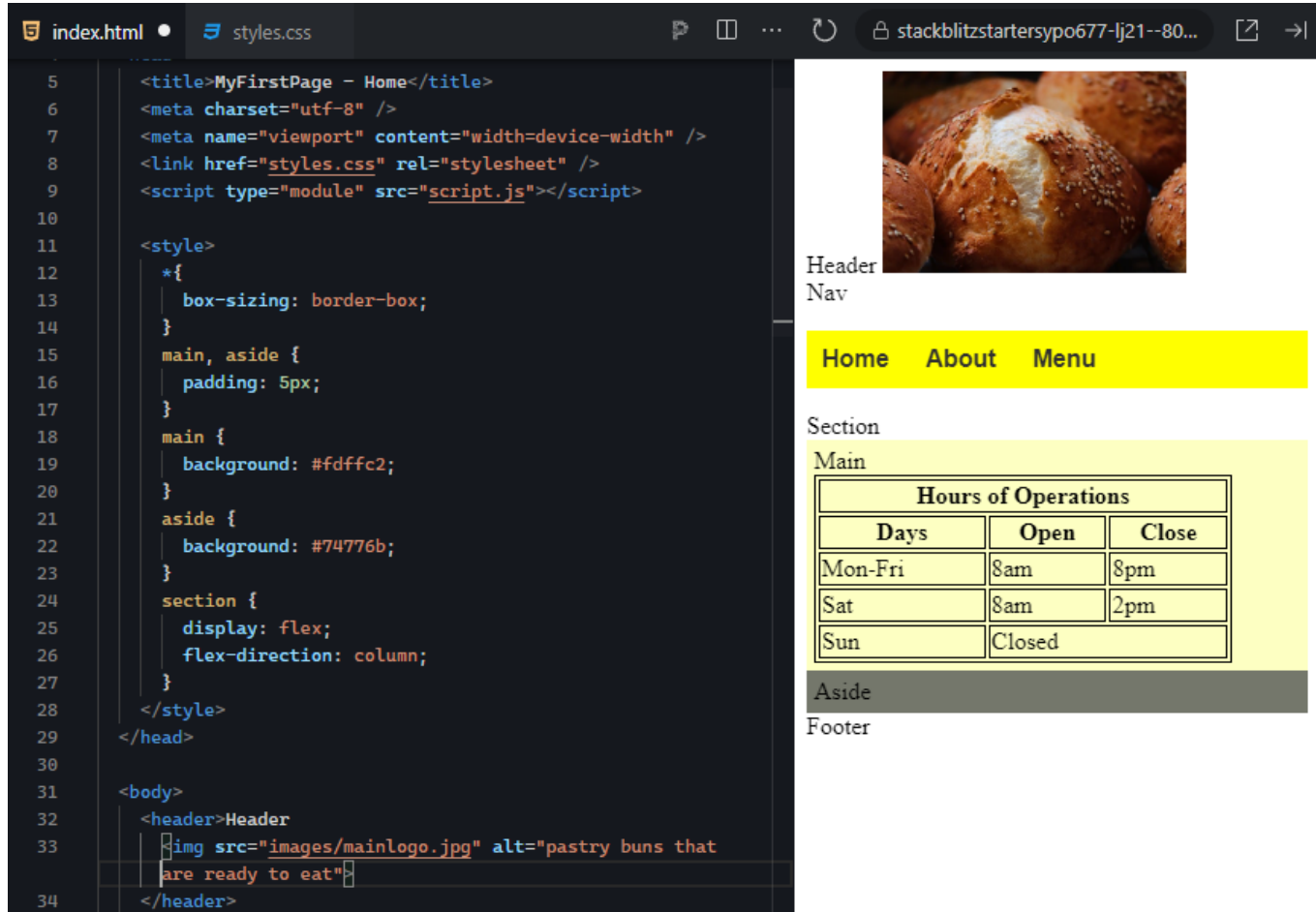


**Directions:** Now that you have learned about how content can be rearranged with flexbox and media queries, it is time to try it yourself.

**Note:** At any time if you do not see the change occur, make sure to select the refresh button in the IDE.

1. Return to your StackBlitz account and open the project you were working on in the Forms lesson. We will be adding background colors to the main and aside sections, setting their widths to 70% and 30%, respectively, and preparing them to respond to the width of the screen and change their orientation.
2. In the head section of the index.html, create a style element and add the following properties to the main and aside sections:

⇒ **EXAMPLE** The expected output is as follows:



The screenshot shows a StackBlitz IDE with two tabs: `index.html` and `styles.css`. The `index.html` file contains the following code:

```
5 <title>MyFirstPage - Home</title>
6 <meta charset="utf-8" />
7 <meta name="viewport" content="width=device-width" />
8 <link href="styles.css" rel="stylesheet" />
9 <script type="module" src="script.js"></script>
10
11 <style>
12   *{
13     box-sizing: border-box;
14   }
15   main, aside {
16     padding: 5px;
17   }
18   main {
19     background: #fddffc2;
20   }
21   aside {
22     background: #74776b;
23   }
24   section {
25     display: flex;
26     flex-direction: column;
27   }
28 </style>
29 </head>
30
31 <body>
32   <header>Header
33   | 
35 </header>
```

The preview on the right shows the rendered web page. It has a header with a logo of pastry buns and a navigation bar with links for Home, About, and Menu. The main content area is divided into a main section (yellow background) and an aside section (gray background). The main section contains a table titled "Hours of Operations" with the following data:

Days	Open	Close
Mon-Fri	8am	8pm
Sat	8am	2pm
Sun	Closed	

The footer is also visible at the bottom of the page.

Remember, we are going to assume a mobile-first approach, so the default styles should be set up for a narrow screen. In this case, we stack them in a column and give them widths of 100%.

- a. Using the universal selector `*`, set the box-sizing to border-box.
- b. Add a 5-pixel border to both the main and aside.
- c. Give the main the background color of `#FDDFC2`.
- d. Give the aside the background color of `#74776B`.
- e. Give the section the display value of `flex` and set flex direction to `column`.

At this point, you should see that your main and aside sections sit on top of each other and take up the full width.

3. Next, we will finish by adding the media query and overriding the flex-direction and the main and aside widths.

a. Add the following media query at the bottom of the style element. NOTE: Do *not* include a semi-colon after the 480px:

```
@media all and (min-width: 480px) { }
```

b. Next, within the media query's curly brackets, override the section's flex-direction.

c. Set the main to a width of 70%.

d. Set the aside to a width of 30%.

4. If everything was coded correctly, you should be able to click and drag the panel divider and see the results of your media query switching between the main and aside orientations.

🔗 EXAMPLE The expected output is as follows:

The screenshot shows a web browser with two panes. The left pane displays the CSS code for `index.html` and `styles.css`. The right pane shows the rendered output of the code.

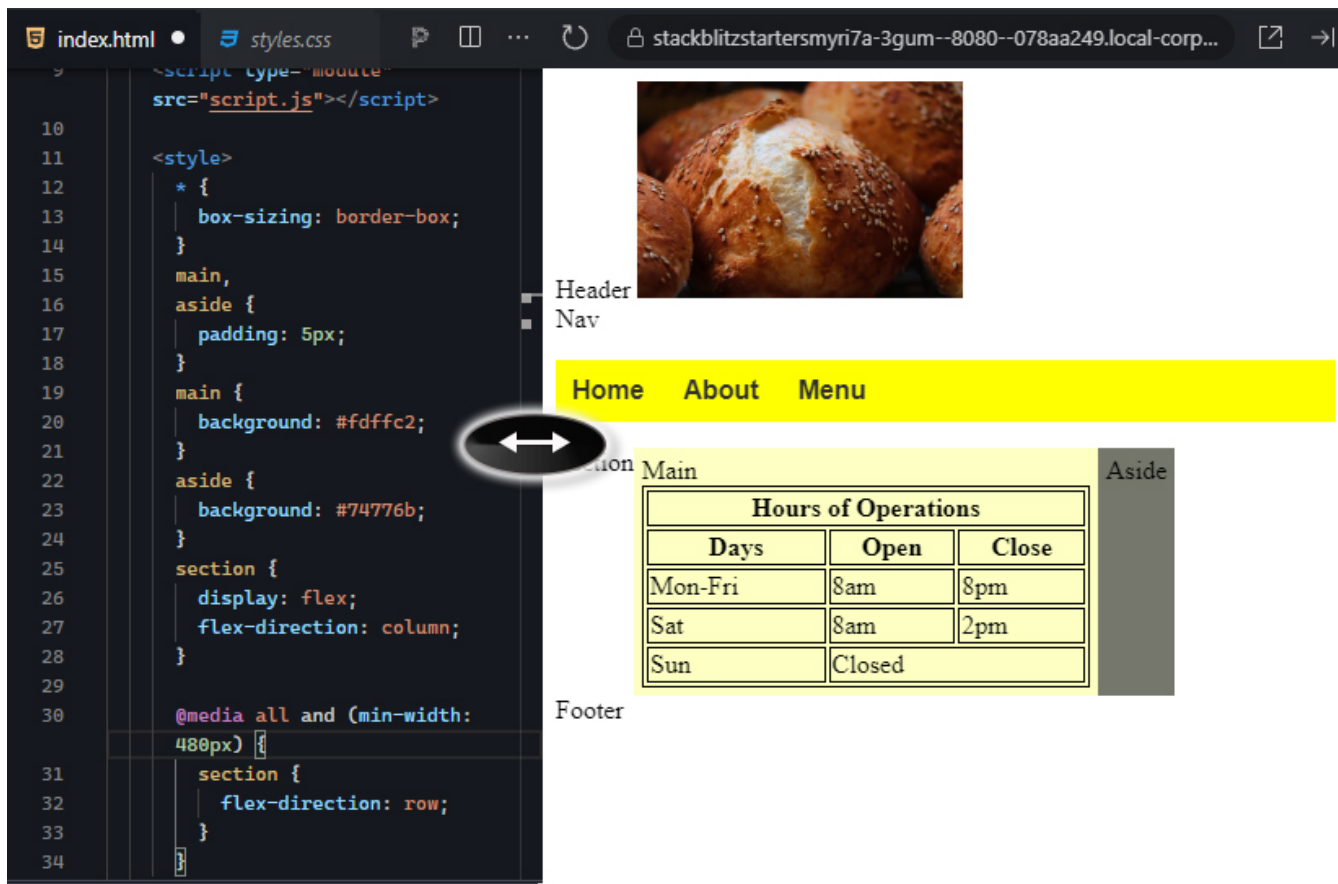
**Code in the left pane:**

```
10 <script type="module" src="script.js"></script>
11
12 <style>
13   * {
14     box-sizing: border-box;
15   }
16   main,
17   aside {
18     padding: 5px;
19   }
20   main {
21     background: #fdffc2;
22   }
23   aside {
24     background: #74776b;
25   }
26   section {
27     display: flex;
28     flex-direction: column;
29   }
30   @media all and (min-width: 480px) {
31     section {
32       flex-direction: row;
33     }
34   }
35 </style>
36 </head>
```

**Rendered output in the right pane:**

The output shows a header with a navigation menu (Home, About, Menu) and a main section. The main section contains a table titled "Hours of Operations" and an aside section. The footer is at the bottom.

Hours of Operations		
Days	Open	Close
Mon-Fri	8am	8pm
Sat	8am	2pm
Sun	Closed	



## REFLECT

Congratulations! You successfully set up your main and aside sections to be responsive to the width of the viewport. Think about how this may be useful in other areas of your website. Other content sections, such as the footer, galleries of images, and the navigation menu, which we will explore next.

## 4. Implementing a Hamburger Menu

Remember that the hamburger menu is a menu that saves screenspace while it is not in use, which is particularly helpful for mobile users. The technique for allowing a full-size menu to switch into a hidden popup menu requires creating two complete menus and always having one hidden. Then, when the media query affects the page, it swaps the visibility of the two menus.

## STEP BY STEP

The process for preparing a page with a responsive navigation menu is as follows:

1. Prepare the hamburger menu using the CSS tooltip technique.
2. Prepare the standard navigation menu.
3. Surround both in their own division container, each with a unique id, that is, "burgermenu" and "fullmenu." (NOTE: Keep in mind that class and id names cannot contain spaces. Follow the names as they are written in

the tutorial.)

4. Within the site's stylesheet, use the default CSS to hide the full menu using the id selector and set display to "none." Set the hamburger menu using its id selector and set its display property to "block."

5. Create a media query that reacts when the viewport width reaches at least 480 pixels. Use this media query to swap the display values, setting the burger menu to display "none" and the full menu to display "block."



**Directions:** Now that you have learned about the process of creating a responsive navigation menu using media queries, let us start by creating the hamburger menu below the regular navigation menu so that both are on the screen. We will then make them responsive by hiding one and using the media query to swap their visibility using the display property.

**Note:** At any time if you do not see the change occur, make sure to select the refresh button in the IDE.

1. Let us make some changes to the original menu (full menu). In the index.html, locate the unordered list of your full navigation menu, and surround it in a division. Give the division an id of "fullmenu."

2. Go to your stylesheet, and at the bottom of the stylesheet, create a new style rule with the #fullmenu selector and set the display to "none." This will make your full menu disappear as the default style (it will be made to reappear when the screen meets the media query requirements).

🔗 **EXAMPLE** The expected output is as follows:

```
index.html
</style>
</head>
<body>
  <header>Header
  
  </header>
  <nav>
    Nav
    <div id="fullmenu">
      <ul>
        <li><a href="index.html">Home</a></li>
        <li><a href="about.html">About</a></li>
        <li><a href="menu.html">Menu</a></li>
      </ul>
    </div>
  </nav>
  <section>
    Section
    <main>Main
    <table id="hours">
```

```
styles.css
background-color: #585917;
color: white;
}
header img {
  width: 280px;
}
#hours {
  width: 275px;
  border: 1px solid black;
}
#hours th, #hours td {
  border: 1px solid black;
}
#fullmenu {
  display: none;
}
```

Header  
Nav  
Section  
Main  
Hours of Operations  
Days | Open | Close  
Mon-Fri | 8am | 8pm  
Sat | 8am | 2pm  
Sun | Closed  
Aside  
Footer

3. Now, add the same media query as we did before:

```
@media all and (min-width: 480px) { }
```


4. Within the curly brackets, add the same style rule for the #fullmenu and set its display to "block." You should now be able to resize the viewport and see that your menu appears when the screen becomes wider than 480

pixels and disappears when it is less than 480 pixels.

⇒ EXAMPLE The expected output is as follows:

styles.css

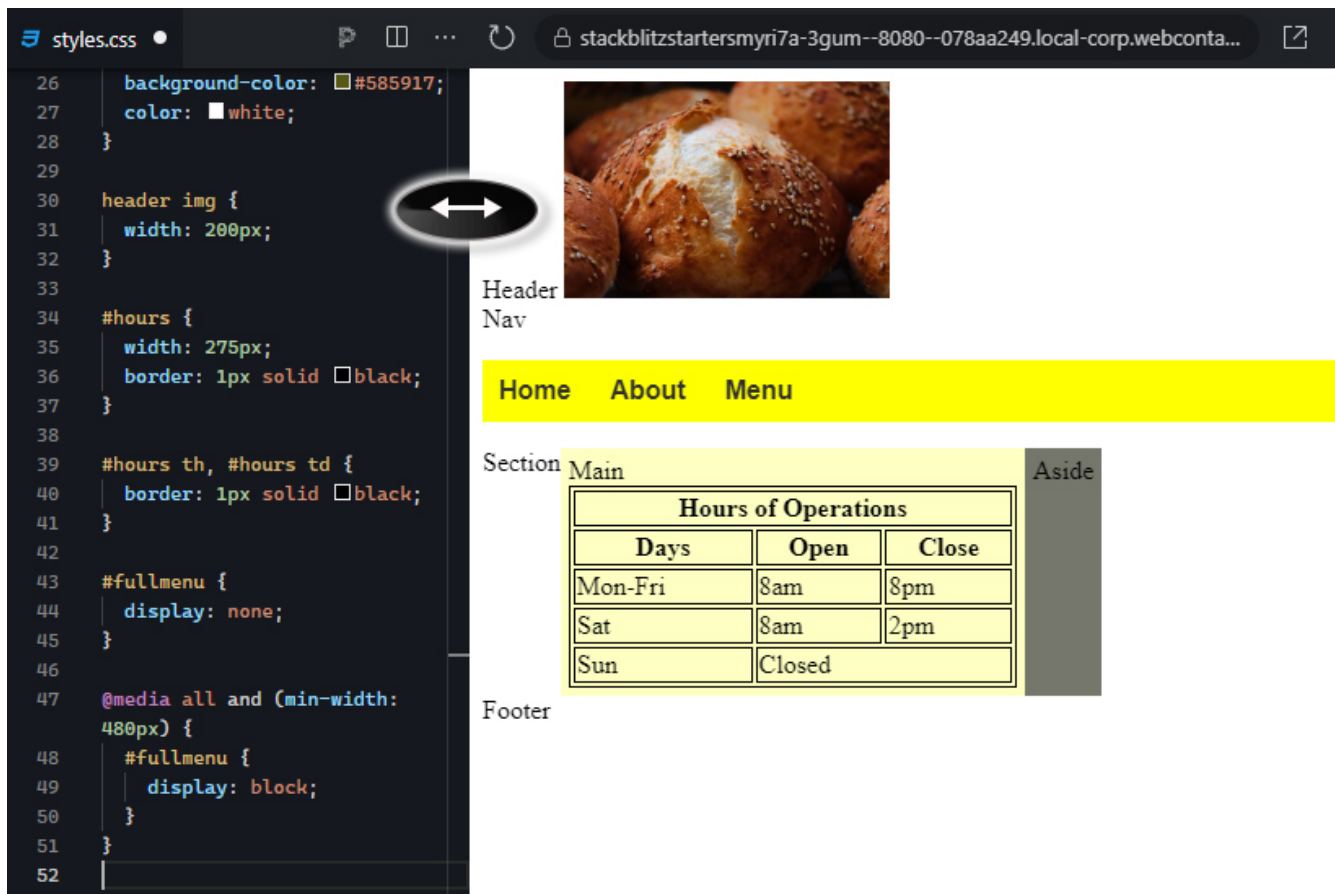
```
26 background-color: #585917;
27 color: white;
28 }
29
30 header img {
31   width: 200px;
32 }
33
34 #hours {
35   width: 275px;
36   border: 1px solid black;
37 }
38
39 #hours th, #hours td {
40   border: 1px solid black;
41 }
42
43 #fullmenu {
44   display: none;
45 }
46
47 @media all and (min-width: 480px) {
48   #fullmenu {
49     display: block;
50   }
51 }
52
```



Header  
Nav  
Section  
Main  
Aside  
Footer

Hours of Operations		
Days	Open	Close
Mon-Fri	8am	8pm
Sat	8am	2pm
Sun	Closed	





5. Next, we will create the burger menu. Copy and paste the entire full menu structure, including the entire division container, and paste the copy below the full menu division container.

6. Change the id value of the new menu to “burgermenu,” and give the unordered list within an id of “burgermenulinks.”

7. Surround the unordered list with a division element. Give the division element an id of “burgermenu.”

8. Give the unordered list tag an id of “burgermenulinks.”

9. Just below the burgermenu division and before the burgermenulinks, paste the following SVG code from Font Awesome. This is a burger menu icon and will be the hover target that will show the burger menu:

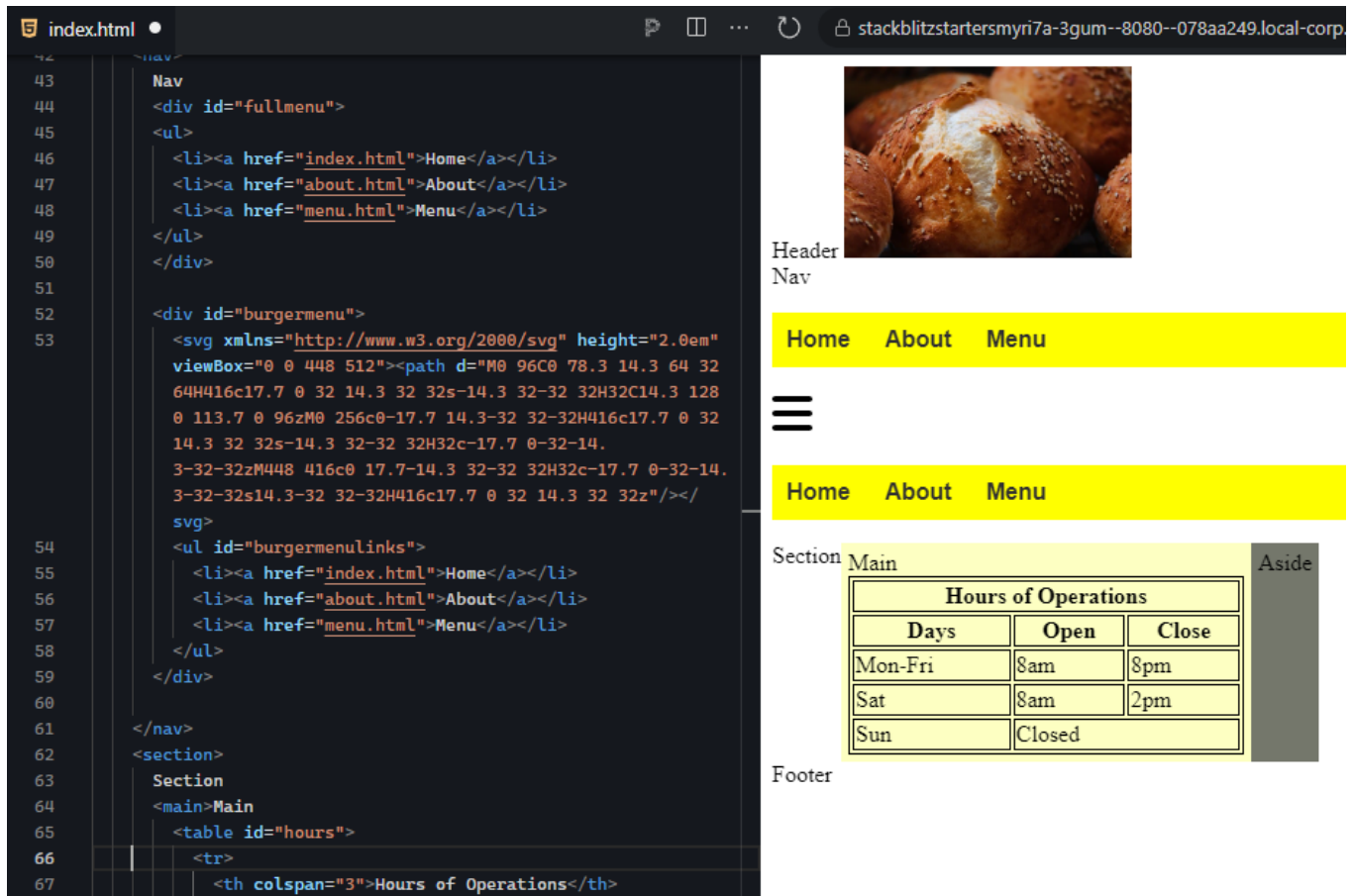
```

<svg xmlns="http://www.w3.org/2000/svg" height="2.0em" viewBox="0 0 448 512"><path d="M0 96C0
78.3 14.3 64 32 64H416c17.7 0 32 14.3 32 32s-14.3 32-32 32H32C14.3 128 0 113.7 0 96zM0 256c0-17.7 14.3-
32 32-32H416c17.7 0 32 14.3 32 32s-14.3 32-32 32H32c-17.7 0-32-14.3-32-32zM448 416c0 17.7-14.3 32-
32 32H32c-17.7 0-32-14.3-32-32s14.3-32 32-32H416c17.7 0 32 14.3 32 32z"/></svg>

```

If your viewport is wide enough, you will see two copies of your original full menu and the hamburger icon between them. The upper menu should disappear when you reduce the width of the viewport:

⇒ **EXAMPLE** The expected output is as follows:



Header  
Nav



Section	Main	Aside
Hours of Operations		
Days	Open	Close
Mon-Fri	8am	8pm
Sat	8am	2pm
Sun	Closed	

Footer

### Aside

However, we need to separate the two menus for styling purposes. To do this, we need to make the selectors for the original full menu more specific. Everywhere you see a selector that starts with “nav,” replace the “nav” with “#fullmenu.” If you refresh, you will see that the burger menu will appear as an unstyled unordered list.

- a. position: relative;
- b. display: block;
- c. height: 50px;
- d. width: 50px;
- e. margin: 0 auto;

This will set the hover area of the burger menu to 50 by 50 pixels and center it using “margin: 0 auto;”.

This is where you get to stylize your burger menu:

- e. Give it a padding of 25px 10px (25 pixels on the top and bottom; 10 pixels on the left and right).
- f. Set the width to 150px.
- g. Set the color to #171717.
- h. Set the background color to #9e9e9e.

These values can be adjusted after the next step in order to fine-tune the style of the menu.

12. Next, we will set up the burger menu to respond to a mouse hover, display the menu, and control the menu's position. Create a new style rule below the previous one with the following selector: `#burgermenu:hover #burgermenulinks`.

- a. Set visibility to "visible."
- b. Set display to "block."
- c. Set margin to "0 auto."
- d. Set left to -61px. (This will position the menu centered under the burger menu icon. The -61 was calculated by taking the width of the menu (150px), subtracting the width of the icon (28px), and then dividing by 2. If you used a different menu width or icon size, you may need to adjust the values.)

EXAMPLE The expected output is as follows:

The screenshot shows a web browser with two panes. The left pane displays CSS code, and the right pane shows the rendered web page.

**Left Pane (CSS Code):**

```

46
47 #burgermenu {
48   position: relative;
49   display: block;
50   height: 50px;
51   width: 50px;
52   margin: 0 auto;
53 }
54
55 #burgermenu #burgermenulinks {
56   visibility: hidden;
57   position: absolute;
58   list-style-type: none;
59   padding: 25px 10px;
60   text-align: center;
61   width: 150px;
62   color: #171717;
63   background: #9e9e9e;
64 }
65
66 #burgermenu:hover #burgermenulinks {
67   visibility: visible;
68   display: block;
69   margin: 0 auto;
70   left: -61px;
71   /* left: calc((-150px - 28px) / 2); */
72 }
73
74 @media all and (min-width: 480px) {
75   #fullmenu {
76     display: block;
77   }
78 }

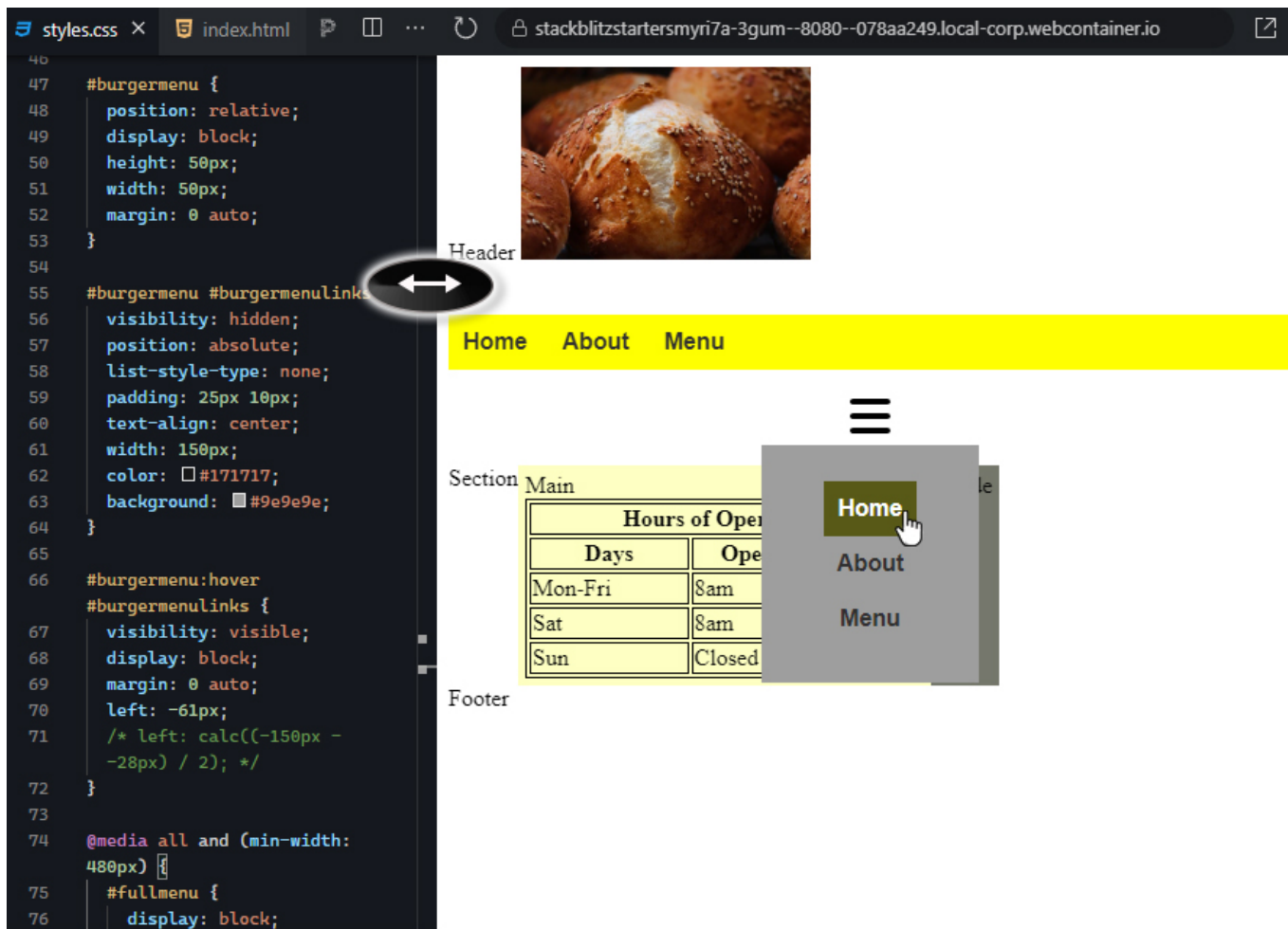
```

**Right Pane (Visual Output):**

The visual output shows a web page with a header, a section, and an aside/footer area. The header contains a burger icon (three horizontal lines) and a circular arrow icon. The section contains a table titled "Hours of Operations".

Hours of Operations		
Days	Open	Close
Mon-Fri	8am	8pm
Sat	8am	2pm
Sun	Closed	

The aside/footer area is a gray bar.



13. The second to last step is to hide the burger menu when the full menu is visible. Go to the media query section, and below the `#fullmenu` style rule, add a new rule for `#burgermenu` and set its display to “none.” Your navigation menu is now responsive.

14. Finally, copy and paste the entire nav section from the `index.html` and paste it into the `about.html`, overwriting the old nav section. Now, your `about.html` page’s navigation menu is also responsive. Remember to refresh your project.

⇒ EXAMPLE The expected output is as follows:


index.html
styles.css
about.html

stackblitzstartersmyri7a-3gum--8080--078aa249.lo...

```

7   <link href="styles.css" rel="stylesheet" />
8
9   <style>
10    * {
11      box-sizing: border-box;
12    }
13  </style>
14
15 </head>
16 <body>
17   <header></header>
18
19   <nav>
20     Nav
21     <div id="fullmenu">
22       <ul>
23         <li><a href="index.html">Home</a></li>
24         <li><a href="about.html">About</a></li>
25         <li><a href="menu.html">Menu</a></li>
26       </ul>
27     </div>
28     <div id="burgermenu">
29       <svg xmlns="http://www.w3.org/2000/svg" height="2.0em" viewBox="0 0 448
30         512"><path d="M0 96C0 78.3 14.3 64 32 64H416C17.7 0 32 14.3 32 32s-14.3
31         32 32 32H32C14.3 128 0 113.7 0 96zM0 256C0 17.7 14.3 32 32 32H416C17.7
32         0 32 14.3 32 32s-14.3 32 32 32H32C17.7 0 32 14.3 32 32s14.3 32 32
33         32H416C17.7 0 32 14.3 32 32s14.3 32 32 32H416C17.7 0 32 14.
34         3 32 32z"/></svg>
35       <ul id="burgermenulinks">
36         <li><a href="index.html">Home</a></li>
37         <li><a href="about.html">About</a></li>
38         <li><a href="menu.html">Menu</a></li>
39       </ul>
40     </div>
41   </nav>
42
43   <main>

```



Header  
Nav

Section

Main

Hours of Operations		
Days	Open	Close
Mon-Fri	8am	8pm
Sat	8am	2pm
Sun	Closed	

Aside

Footer


index.html
styles.css
about.html

stackblitzstartersmyri7a-3gum--8080--078aa249.local-corp.webco...

```

6   <title>Harvest Bakery - About Us</title>
7   <link href="styles.css" rel="stylesheet" />
8
9   <style>
10    * {
11      box-sizing: border-box;
12    }
13  </style>
14
15 </head>
16 <body>
17   <header></header>
18
19   <nav>
20     Nav
21     <div id="fullmenu">
22       <ul>
23         <li><a href="index.html">Home</a></li>
24         <li><a href="about.html">About</a></li>
25         <li><a href="menu.html">Menu</a></li>
26       </ul>
27     </div>
28     <div id="burgermenu">
29       <svg xmlns="http://www.w3.org/2000/svg" height="2.0em"
30         viewBox="0 0 448 512"><path d="M0 96C0 78.3 14.3 64 32 64 32
31         64H416C17.7 0 32 14.3 32 32 32 32H32C14.3 128 0 113.7 0 96zM0 256C0
32         17.7 14.3 32 32 32H416C17.7 0 32 14.3 32 32 32 32H32C17.7 0 32
33         14.3 32 32s14.3 32 32 32H416C17.7 0 32 14.3 32 32s14.3 32 32
34         32H416C17.7 0 32 14.3 32 32s14.3 32 32 32H416C17.7 0 32 14.3
35         32 32z"/></svg>
36       <ul id="burgermenulinks">
37         <li><a href="index.html">Home</a></li>
38         <li><a href="about.html">About</a></li>
39         <li><a href="menu.html">Menu</a></li>
40       </ul>
41     </div>
42   </nav>
43
44   <main>

```



Header  
Nav

Home About Menu

Section

Main

Hours of Operations		
Days	Open	Close
Mon-Fri	8am	8pm
Sat	8am	2pm
Sun	Closed	

Aside

Footer

index.html

styles.css

about.html

stackblitzstartersmyri7a-3gum--8080--078aa249.local-corp.webco...

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

8

9<style>

10  \* {

11    box-sizing: border-box;

12  }

13</style>

14

15</head>

16<body>

17  <header></header>

18

19  <nav>

20    Nav

21    <div id="fullmenu">

22      <ul>

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

24        <li><a href="about.html">About</a></li>

25        <li><a href="menu.html">Menu</a></li>

26      </ul>

27    </div>

28    <div id="burgermenu">

29      <svg xmlns="http://www.w3.org/2000/svg" height="2.0em" viewBox="0 0 448 512"><path d="M0 96C0 78.3 14.3 64 32 64H416c17.7 0 32 14.3 32 32s-14.3 32-32 32H32C14.3 128 0 113.7 0 96zM0 256c0-17.7 14.3-32 32-32H416c17.7 0 32 14.3 32 32s-14.3 32-32 32H32c-17.7 0-32-14.3-32-32zM416 416c0 17.7 14.3 32 32 32s-14.3 32-32 32H32c-17.7 0-32-14.3-32-32zM416 448c0 17.7 14.3 32 32 32s-14.3 32-32 32H32c-17.7 0-32-14.3-32-32Z"/></svg>

30      <ul id="burgermenulinks">

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

32        <li><a href="about.html">About</a></li>

33        <li><a href="menu.html">Menu</a></li>

34      </ul>

35    </div>

36  </nav>

37

38  <main>

39    <h1>Learn a little about us!

40    </h1>

41  </main>

Nav

Home About Menu

Learn a little about us!

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Contact Us

Personal Information:

First Name:

Last Name:

Phone:

Email:

How many days per week do you visit the shop?:

Reason for communication:

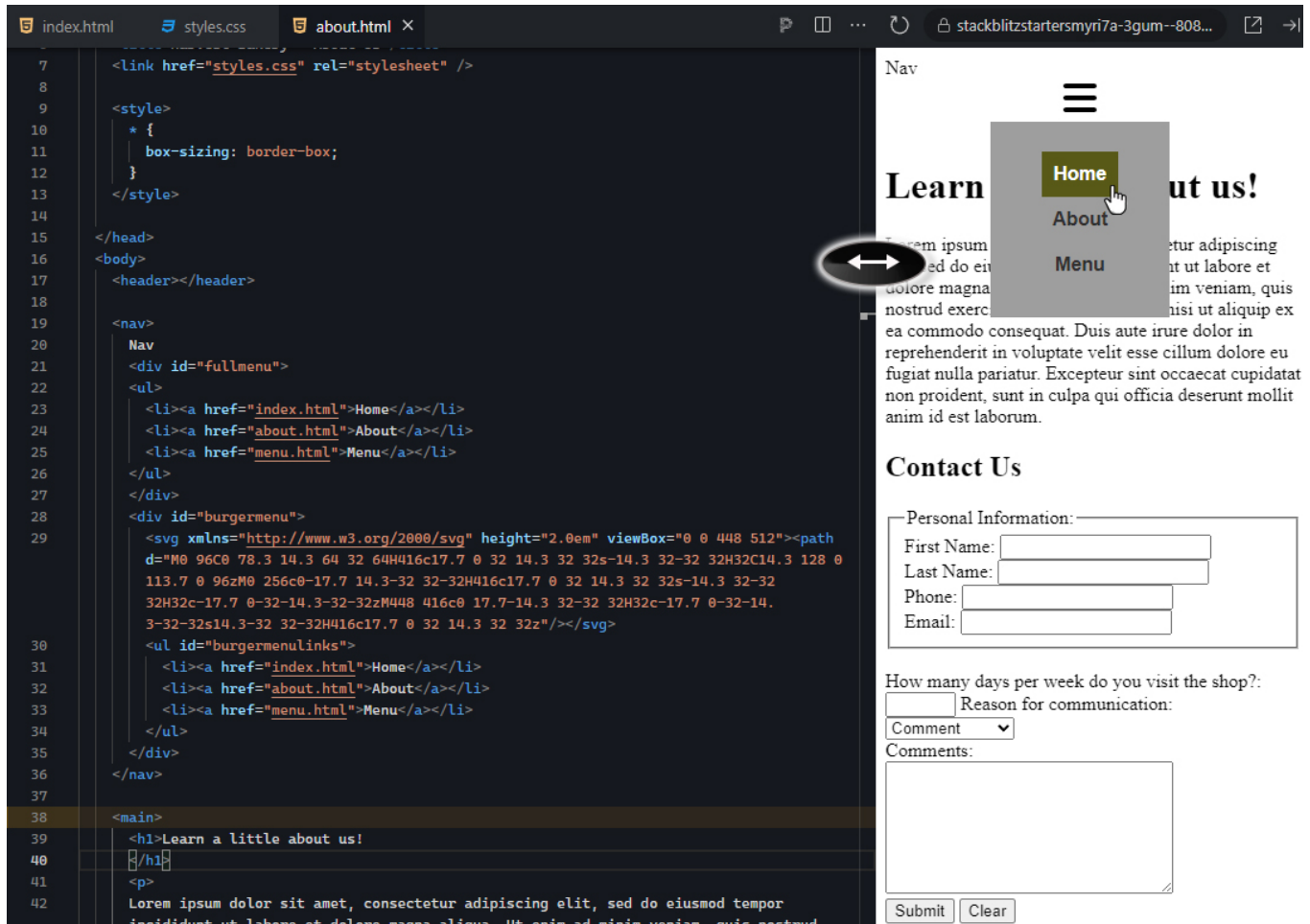
Comments:

Submit Clear

© 2024 SOPHIA Learning, LLC. SOPHIA is a registered trademark of SOPHIA Learning, LLC.

Page 14





**Note:** In the about.html page, if your #burgermenulinks in the hover state are not vertically aligned, add the same style element that you added in the head section of the index.html. As in the index.html, use the universal selector “\*,” set the box-sizing to border-box.

15. Make sure to save your project.

## REFLECT

Congratulations! You have successfully set up a navigation menu to be responsive and convert into a compact hamburger menu. The popup menu itself is just a hidden section of HTML content that is positioned relative to its parent and is only visible when the mouse hovers over the icon. This is a valuable skill set to have, as all websites today really should be built using responsive web design.

At this point, you would want to stylize the popup menu according to the design of the website.

## WATCH

View the following video for more on how you can create content for all screen sizes with responsive design.

## SUMMARY

In this lesson, you learned about the **importance of responsive web design**. You learned about the **viewport** attribute and its importance in building a responsive webpage. Lastly, you learned about two common techniques, including **rearranging content** using media queries and flexbox, as well as how to perform a content swap to **implement the hamburger menu**, which is a more compact version of the navigation menu for mobile users.

Source: This Tutorial has been adapted from "The Missing Link: An Introduction to Web Development and Programming " by Michael Mendez. Access for free at <https://open.umn.edu/opentextbooks/textbooks/the-missing-link-an-introduction-to-web-development-and-programming>. License: **Creative Commons attribution: CC BY-NC-SA**.



## TERMS TO KNOW

### Viewport

The area of a computer program that is currently shown on the screen and an important setting for making a website responsive.