# LAB 7

# ADVANCED CSS: LAYOUT

### What You Will Learn

- How to position and float elements outside of normal page flow
- How to construct multi-column layouts using flex and grid
- How to make a responsive layout using media queries
- How to use CSS transitions, transformations, filters, and animations
- How to use the SASS preprocessor

### Approximate Time

The exercises in this lab should take approximately 120 minutes to complete.

## Fundamentals of Web Development, 3rd Ed

Randy Connolly and Ricardo Hoar

Textbook by Pearson
http://www.funwebdev.com

Date Last Revised: February 7, 2022

### Preparing Directories

The starting `lab07` folder has been provided for you (within the zip file/folder downloaded from Gumroad).

*Note: these labs use the convention of* `blue background` *text to indicate filenames or folder names and* ***bold red*** *for content to be typed in by the student.*

## Older Forms of CSS Layout

Let's begin with a bit of a review about block vs inline vs inline-block display modes.

### Exercise 7.1 — Display property

1 Open, examine, and test `lab07-ex01.html`.

2 Resize the browser window and experiment by making the width smaller and larger.

3 Modify the following style in `lab07-ex01.css` and test.

```
h1 span {
  background-color: #E668A7;
  color: #87EAF2;
  font-size: 1.25em;
  margin: 50px;
  width: 200px;
  height: 100px;
  display: inline;
}
```

*You will notice that the top and bottom margin do not appear and the margin and height settings are ignored, because inline elements don't have top and bottom margins or widths and heights.*

4 Change the `display` property as follows and test:

```
h1 span {
  …
  display: inline-block;
}
```

*The margins, width, and height are now displayed. The <span> still sits on the same line.*

5 Change the `display` property as follows and test:

```
h1 span {
  …
  display: block;
}
```

*The margins, width, and height are displayed and the <span> now sits on its own line.*

6 Change the `display` property of the `<span>` back to `inline`.

7 Add the following style and test.

```
li {
  list-style: none;
  display: inline;
}
```

*This changes the <li> elements from block-level to inline (they will thus show up on a single line now.*

8 Change the `<li>` elements to `inline-block` and add 1em margins and test.

*Remember that inline-block keeps elements on the same line but gives them access to box model properties.*

9 Add the following style and test.

```
.card img {
  display: block;
  margin: 0.5em;
}
```

*By default, a block-level element exists on its own line.*

10 Change the display to `inline-block` and test.

## Exercise 7.2 — Relative Positioning

1 Open, examine, and test `lab07-ex02.html`.

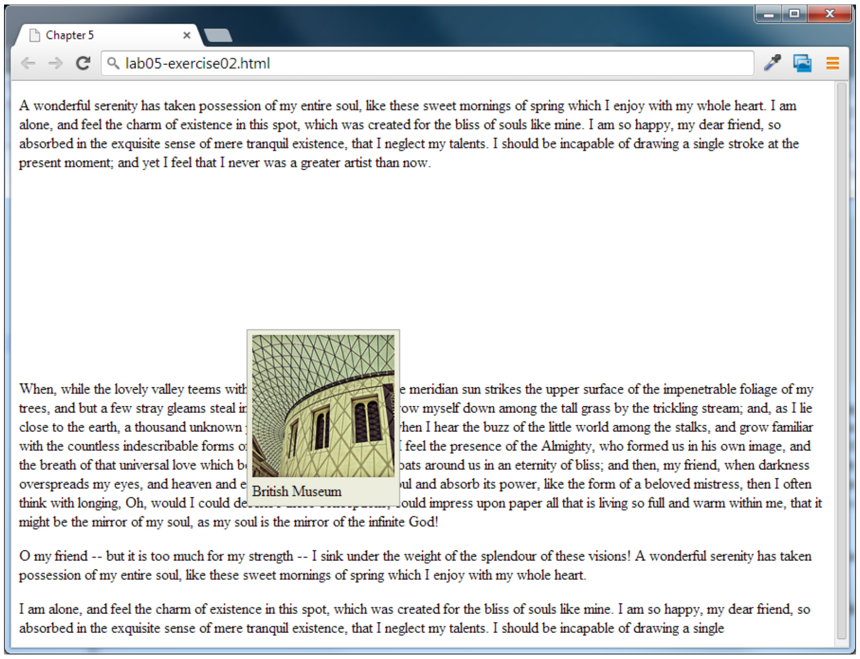
2 Modify the following style in `lab07-ex02.css` and test.

```
figure {
   background-color: #EDEDDD;
   border: 1pt solid #A8A8A8;
   padding: 5px;
   width: 150px;
   top: 150px;
   left: 200px;
}
```

3 Modify the following style and test.

```
figure {
   background-color: #EDEDDD;
   border: 1pt solid #A8A8A8;
   padding: 5px;
   width: 150px;
   top: 150px;
   left: 200px;
   position: relative;
}
```

*As you can see in Figure 7.1, the original space for the positioned <figure> element is* *preserved, as is the rest of the document's flow. As a consequence, the repositioned element now overlaps other content.*



A wonderful serenity has taken possession of my entire soul, like these sweet mornings of spring which I enjoy with my whole heart. I am alone, and feel the charm of existence in this spot, which was created for the bliss of souls like mine. I am so happy, my dear friend, so absorbed in the exquisite sense of mere tranquil existence, that I neglect my talents. I should be incapable of drawing a single stroke at the present moment; and yet I feel that I never was a greater artist than now.

When, while the lovely valley teems with e meridian sun strikes the upper surface of the impenetrable foliage of my trees, and but a few stray gleams steal in ow myself down among the tall grass by the trickling stream; and, as I lie close to the earth, a thousand unknown hen I hear the buzz of the little world among the stalks, and grow familiar with the countless indescribable forms o I feel the presence of the Almighty, who formed us in his own image, and the breath of that universal love which b oats around us in an eternity of bliss; and then, my friend, when darkness overspreads my eyes, and heaven and e ul and absorb its power, like the form of a beloved mistress, then I often think with longing, Oh, would I could de ould impress upon paper all that is living so full and warm within me, that it might be the mirror of my soul, as my soul is the mirror of the infinite God!

O my friend -- but it is too much for my strength -- I sink under the weight of the splendour of these visions! A wonderful serenity has taken possession of my entire soul, like these sweet mornings of spring which I enjoy with my whole heart.

I am alone, and feel the charm of existence in this spot, which was created for the bliss of souls like mine. I am so happy, my dear friend, so absorbed in the exquisite sense of mere tranquil existence, that I neglect my talents. I should be incapable of drawing a single

*Figure 7.1 – Exercise 7.2 complete*

## Exercise 7.3 — Absolute Positioning

1 Open, examine, and test `lab07-ex03.html`.

2 Modify the following style in `lab07-ex03.css` and test.

```
figure {
   background-color: #EDEDDD;
   border: 1pt solid #A8A8A8;
   padding: 5px;
   width: 150px;
   top: 150px;
   left: 200px;
   position: absolute;
}
```

*With absolute positioning, space is not left for the moved element, as it is no longer in the normal flow.*

3 Modify the following style and test.

```
figcaption {
   background-color: #EDEDDD;
   padding: 5px;
   top: 150px;
   left: 200px;
   position: absolute;
}
```

*A moved element via absolute position is actually positioned relative to its nearest positioned ancestor container (that is, a block-level element whose position is fixed, relative, or absolute). In this example, the <figcaption> is absolutely positioned; it is moved 150 px down and 200 px to the left of its nearest positioned ancestor, which happens to be its parent (the <figure> element).*

## Exercise 7.4 — Floating Elements

1 Open, examine, and test `lab07-ex04.html`.

2 Modify the following styles in `lab07-ex04.css` and test.

```
figure {
   border: 1pt solid #A8A8A8;
   background-color: #EDEDDD;
   padding: 5px;
   margin: 10px;
   width: 150px;
   float: left;
}
```

*Notice that a floated block-level element must also have a width specified.*

3 Modify the following styles and test.

```
figure {
   border: 1pt solid #A8A8A8;
   background-color: #EDEDDD;
   padding: 5px;
   margin: 10px;
   width: 150px;
   float: right;
}
```

## Exercise 7.5 — Floating in a Container

1 Open, examine, and test `lab07-ex05.html`.

*One of the most common styling requirements is to have content beside some other content. In this example, we want the text to be the right of the images.*

2 Add the following styles and test.

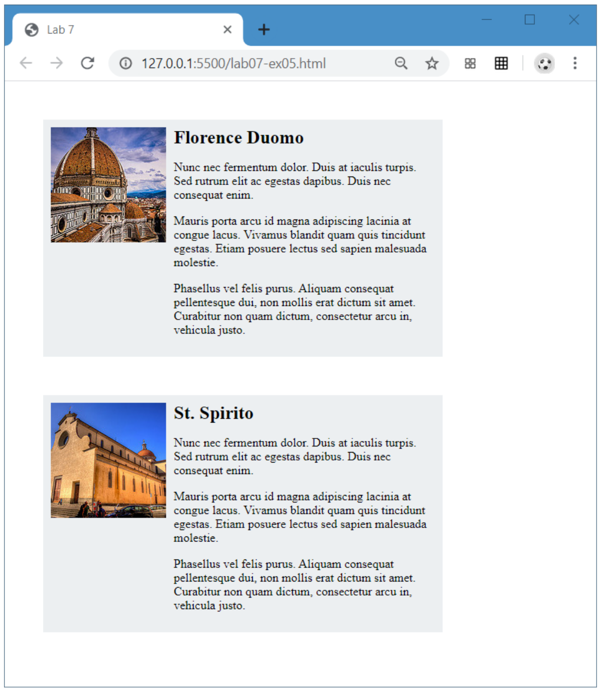
```
.media-image {
   float: left;
   margin-right: 10px;
}
.media-body {
   margin-left: 160px;
}
```

*Notice that the image within each media <div> is floating within its container, not the page as a whole, as shown in Figure 7.4.*

3 Modify `lab07-ex05.html` by changing the filename of the first image to `art-1.jpg` and test.

*Notice that such a floated layout requires very specific widths and or margins based on image sizes. They are thus not very generalizable.*

4 Return the filename of the first image to its original name.



**Florence Duomo**

Nunc nec fermentum dolor. Duis at iaculis turpis. Sed rutrum elit ac egestas dapibus. Duis nec consequat enim.

Mauris porta arcu id magna adipiscing lacinia at congue lacus. Vivamus blandit quam quis tincidunt egestas. Etiam posuere lectus sed sapien malesuada molestie.

Phasellus vel felis purus. Aliquam consequat pellentesque dui, non mollis erat dictum sit amet. Curabitur non quam dictum, consectetur arcu in, vehicula justo.

**St. Spirito**

Nunc nec fermentum dolor. Duis at iaculis turpis. Sed rutrum elit ac egestas dapibus. Duis nec consequat enim.

Mauris porta arcu id magna adipiscing lacinia at congue lacus. Vivamus blandit quam quis tincidunt egestas. Etiam posuere lectus sed sapien malesuada molestie.

Phasellus vel felis purus. Aliquam consequat pellentesque dui, non mollis erat dictum sit amet. Curabitur non quam dictum, consectetur arcu in, vehicula justo.

*Figure 7.2 – Exercise 7.5 with float and margins*

### Exercise 7.6 — Showing / Hiding Elements

1 Open, examine, and test `lab07-ex06.html`.

2 Modify the following styles in `lab07-ex06.css` and test by moving mouse over any of the colored items.

```
ul:hover .action {
  display: none;
}
```

*Notice that the space for second item is removed.*

3 Modify as follows and test.

```
ul:hover .action {
  visibility: hidden;
}
```

*Notice that space for hidden item is maintained.*

## FlexBox and Grid Layout

The exercises so far in this lab showed two different ways to move items out of the normal top-down flow, namely, by using positioning and by using floats. These older approaches have been supplanted by two newer forms of layout: flexbox and grid.

### Exercise 7.7 — Using Flexbox

1 Open, examine, and test `lab07-ex07.html`.

*You will be doing the same layout task as Exercise 7.5, but instead of floats you will use flex box.*

2 Notice the blue content at the top of the page. Add the following style then test.

```
.container {
   display: flex;
}
```

*Notice that both block and inline elements become flex items. Notice also that each flex item is placed along the main axis.*

3 Modify this tyle as follows and test.

```
.container {
   display: flex;
   justify-content: center;
}
```

*Changes the alignment of items within the main axis.*

4 Modify this style as follows and test.

```
.container {
   display: flex;
   justify-content: space-between;
}
```

5 Change this property as follows then test.

```
justify-content: space-around;
```

6 Change this property as follows then test.

```
justify-content: flex-end;
```

7 Add the following styles to `lab07-ex07.css` and test.

```
.media {
   display: flex;
   align-items: flex-start;
}
```

8 Add the following style and test.

```
.media-image {
   margin-right: 1em;
}
```

9 Change the filename of the first image to `art-1.jpg` and test.

*Notice that in comparison to the floated version in Exercise 7.5, the flexbox version still works even with a different image size.*

## Exercise 7.8 — FlexBox Direction

1 Open, examine, and test `lab07-ex08.html`.

2 Add the following styles to `lab07-ex08.css` and test.

```
 .row {
   display: flex;
   flex-direction: row;
}
.column {
   display: flex;
   flex-direction: column;
}
```

3 Add the following style (don't test yet).

```
.reverse {
   flex-direction: row-reverse;
}
```

4 Modify the markup as follows and test.

```
<section class="row reverse">
```

5 Add the following styles and test.

```
.box {
   flex-grow: 1;
}
```

*Flex grow specifies how a flex item should grow relative to the other items.*

6 Add the following style and test.

```
.c {
   flex-grow: 3;
}
```

7 Add the following style and test.

```
.d {
   flex-basis: 400px;
}
```

*Flex basis determines the initial size of the flex item of the flex item before the remaining space is distributed.*

## Exercise 7.9 — Centering with Flexbox

1 Open, examine, and test `lab07-ex09.html`.

2 Add the following styles to `lab07-ex09.css`.

```
.centered {
   display: flex;
   align-items: center;
   justify-content: center;
}
```

3 Modify the markup as follows and test.

```
<section>
   <div>A</div>
</section>
<section class="centered">
   <div>B</div>
</section>
<section class="centered">
   <div class="centered">C</div>
</section>
```

## Exercise 7.10 — FlexBox Navigation and Cards

1 Open, examine, and test `lab07-ex10.html`.

2 Modify the following style in `lab07-ex10.css` and test.

```
nav ul {
   background-color: var(--color-primary-1-700);
   display: flex;
   align-items: center;
   justify-content: stretch;
}
```

*This places each item along the same flex axis.*

3 Modify the following style and test.

```
nav li {
   list-style-type: none;
   flex: 1 1 auto;
   padding: 0.5em;
}
```

*The flex property is shorthand for flex-grow, flex-shrink, and flex-basis.*

4 Add the following style and test.

```
.right {
   text-align: right;
}
```

5 Modify the following styles and test.

```
.card {
   background-color: white;
   border: solid 1px #2D3A8C;
   border-radius: 4px;
   margin: 1em;
   text-align: center;
   display: flex;
   flex-direction: column;
}
section {
   display: flex;
   margin: 3em;
}
```

6 Add the following style and test.

```
.card .content {
   flex: 1 1 auto;
}
```

*This moves the footer to the bottom of the flex axis.*

## Test Your Knowledge #1

Modify `lab07-test01.html` by adding the CSS to implement the layout shown in Figure 7.4. Then add styles to `lab07-test01.css` (some of the styling as already been provided).

1 Set the background image on the `<body>` tag. Set the `height` to `100vh` so it will always fill the entire viewport. Set the `background-cover` and `background-position` properties (see Chapter 4 for a refresher if needed).

2 For the header set its `display` to `flex`. Set `justify-content` to `space-between` and `align-items` to `center`. This will make the `<h2>` and the `<nav>` elements sit on the same line, but will expand to be aligned with the outside edges.

3 To center the form in the middle of the viewport, set the `display` of the `<main>` element to `flex`, and `align-items` and `justify-contents` to center. Do the same for the `<form>` element.

4 Fine-tune the size of the form elements by setting the `flex-basis` of label to 16em, the search box to 36em, and the submit button to 10em. The final result should look similar to that shown in Figure 7.4.

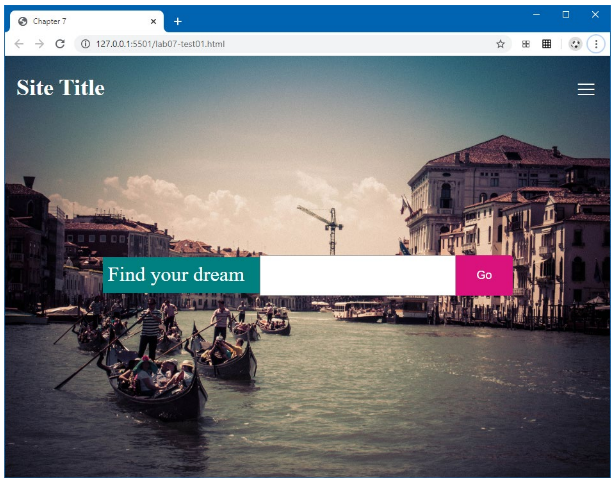


*Figure 7.3 – Completed Test Your Knowledge #1*

## Exercise 7.11 — Using Grid

1 Open, examine, and test `lab07-ex11.html`.

*Notice that the first container uses flex to distribute the items within the container.*

2 Add the following styles to `lab07-ex11.css` and test.

```
.container2 {
   display: grid;
}
```

*By default a grid container will behave like any container in that each block element will be on its own line.*

3 Modify the style as follows and test.

```
.container2 {
   display: grid;
   gap: 10px;
}
```

4 Modify the style as follows and test.

```
.container2 {
   display: grid;
   grid-template-columns: repeat(3, 1fr);
}
```

*This uses grid to layout the items within the container. Each row of the grid will contain three columns of equal size.*

5 Modify the following style property and test.

```
grid-template-columns: 2fr 1fr 1fr;
```

6 Modify the style as follows and test.

```
.container2 {
   display: grid;
   grid-template-columns: 2fr 1fr 1fr;
   grid-template-rows: 80px 200px;
}
```

*You generally don't need to set a row height, since the row height will size itself to fit the content.*

## Exercise 7.12 — Grid Column Widths

1 Open, examine, and test `lab07-ex12.html`.

2 Add the following styles to `lab07-ex12.css` and test.

```
.container {
   display: grid;
   grid-template-columns: 70px 70px 45px;
}
```

*Each column can have its own unique width value using px, %, em, fr, etc. units.*

3 Modify the following style property and test.

```
grid-template-columns: 70px auto 45px;
```

*An auto value indicates the width will fill the remaining space.*

4 Modify the following style property and test.

```
grid-template-columns: repeat(4, 1fr);
```

5 Modify the following style property and test.

```
grid-template-columns: repeat(2, 75px 150px) 300px;
```

6 Modify the following style property and test.

```
grid-template-columns: repeat(auto-fill, minmax(100px, 1fr));
```

*The auto-fill indicates the row should be filled with as many columns as can fit into the container space (which in our example, is 80% of browser width). The minmax function indicates the minimum column size should be 100px and the max size (1fr) is whatever size is necessary to allow each column to be equal sized.*

## Exercise 7.13 — Grid Item Properties

1 Open, examine, and test `lab07-ex13.html`.

2 Add the following to `lab07-ex13.css` and test.

```
.b {
   align-self: stretch;
   justify-self: start;
}
```

*These two properties modify the alignment of content within a grid cell.*

3 Add the following styles and test.

```
.c {
   align-self: start;
   justify-self: stretch;
}
.d {
   align-self: center;
   justify-self: center;
}
.e {
   align-self: end;
   justify-self: end;
}
```

4 Add the following style and test.

```
.a {
   grid-column-start: 1;
   grid-column-end: 3;
}
```

*Normally, grid items are placed automatically. This (and the following) steps illustrate how you can explicitly place content in a specific cell.*

5 Modify the following styles and test.

```
.b {
   align-self: stretch;
   justify-self: start;
   grid-row: 2;
   grid-column: 2;
}
.c {
   align-items: start;
   justify-items: stretch;
   grid-row-start: 1;
   grid-row-end: 3;
   grid-column: 3;
}
```

## Exercise 7.14 — Nested Grids

1 Open, examine, and test `lab07-ex14.html`.

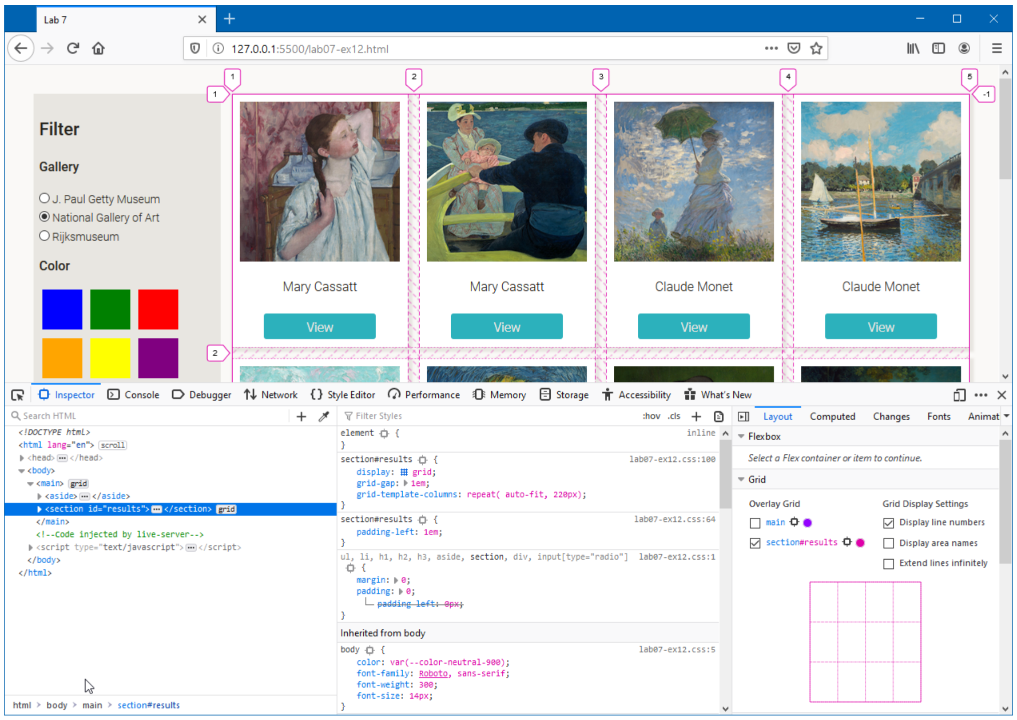
2 Add the following to `lab07-ex14.css` and test.

```
main {
   display: grid;
   grid-template-columns: 1fr 4fr;
}
```

3 Add the following CSS and test.

```
section#results {
   display: grid;
   grid-gap: 1em;
   grid-template-columns: repeat( auto-fit, 220px);
}
```

4 If you have Firefox browser, examine the file in the developer Inspector. Click on the Layout tab and then examine the grids (as shown in Figure 7.5). At the time of writing, FireFox has the superior tool set for investigating grids (and flexbox as well).



```
<!DOCTYPE html>
<html lang="en"> scroll
  <head> … </head>
  <body>
    <main> grid
      <aside> … </aside>
      <section id="results"> … </section> grid
    </main>
    <!--Code injected by live-server-->
    <script type="text/javascript"> … </script>
  </body>
</html>
```

html > body > main > section#results

```
element {                                              inline
}
section#results {                       lab07-ex12.css:100
   display: grid;
   grid-gap: 1em;
   grid-template-columns: repeat( auto-fit, 220px);
}
section#results {                        lab07-ex12.css:64
   padding-left: 1em;
}
ul, li, h1, h2, h3, aside, section, div, input[type="radio"]  lab07-ex12.css:1
 {
   margin: 0;
   padding: 0;
     └ ~~padding-left: 0px;~~
}
Inherited from body
body {                                    lab07-ex12.css:5
   color: var(--color-neutral-900);
   font-family: Roboto, sans-serif;
   font-weight: 300;
   font-size: 14px;
}
```

Flexbox

Select a Flex container or item to continue.

Grid

Overlay Grid
- [ ] main
- [x] section#results

Grid Display Settings
- [x] Display line numbers
- [ ] Display area names
- [ ] Extend lines infinitely

*Figure 7.4 – Inspecting grids in Firefox*

## Exercise 7.15 — Using Calc()

1 The starting files for this exercise are the same as the finished version of the previous exercise.

Change the following style in `lab07-ex14.css` and test.

```
.card img {
   width: 250px;
}
```

*Since you changed the image width within the card, your grid layout is all messed up. This illustrates how CSS can be difficult to maintain and modify.*

2 Add the following to the very top of the CSS file.

```
:root {
   --gapSize: 0.5em;
   --paintingWidth: 200px;
}
```

3 Modify the following styles to use these new variables and test.

```
.card {
   background-color: white;
   box-shadow: 3px 3px 3px 3px var(--color-neutral-100);
   padding: var(--gapSize);
   text-align: center;
}
.card img {
   width: var(--paintingWidth);
}
```

*The layout should be back to working correctly.*

4 Add the following variable to the very top of the CSS file.

```
:root {
   --gapSize: 0.5em;
   --paintingWidth: 200px;
   --cardWidth: calc(var(--gapSize) + var(--paintingWidth) +
                     var(--gapSize));
}
```

*Notice that you can perform math on px and em units together.*

5 Use this new variable by changing the following style.

```
section#results {
   display: grid;
   grid-gap: 1em;
   grid-template-columns: repeat( auto-fit, var(--cardWidth) );
}
```

6 Test. Experiment with different values for the `--paintingWidth` variable. Because the calculated `--cardWidth`, the layout will keep working.

7 Add the following variables:

```
:root {
   ...
   --baseFontSize: 14px;
   --is-size-2: calc(var(--baseFontSize) * 1.2);
   --is-size-3: calc(var(--baseFontSize) * 1.4);
   --is-size-4: calc(var(--baseFontSize) * 1.6);
}
```

*Notice that the other font sizes are proportional to the base font size.*

8 Change all the `font-size` references to these new variables.

```
body {
   ...
   font-size: var(--baseFontSize);
}
h2 {
   font-size: var(--is-size-4);
   ...
}
h3 {
   font-size: var(--is-size-2);
   ...
}
.card h2 {
   ...
   font-size: var(--is-size-3);
}
button {
   ...
   font-size: var(--is-size-2);
}
```

9 Test and experiment with changing the value of the `--baseFontSize` variable.

## Exercise 7.16 — Grid Areas

1. Open, examine, and test `lab07-ex16.html`.
2. Add the following styles in `lab07-ex16.css` and test.

```
.a1 { grid-area: a1; }
.a2 { grid-area: a2; }
.a3 { grid-area: a3; }
.a4 { grid-area: a4; }

.b1 { grid-area: b1; }
.b2 { grid-area: b2; }
.b3 { grid-area: b3; }

.c1 { grid-area: c1; }
.c2 { grid-area: c2; }
```

*It's not going to work correctly because we haven't specified the grid-template-areas yet.*

3. Add the following style and test.

```
.container {
   grid-template-areas: ". a1 a2 a3 a4"
                        "b1 b2 b2 b2 b3"
                        "b1 c1 c2 c2 c2";
}
```

*Note that this isn't replacing the existing styles for this class. It's in addition to it.*

4. Comment out the previous `grid-template-areas` and add the following.

```
grid-template-areas: "a1 a2 a3 a4 b3"
                     "b1 b2 b2 b2 b3"
                     "b1 c1 c2 c2 b3";
```

5. Comment out the previous `grid-template-areas` and add the following:

```
grid-template-areas: "b1 a2 a3 c2 a4"
                     "b1 b2 c1 c2 b3"
                     "a1 a1 c1 .. b3";
```

*You can use multiple periods to indicate an empty cell.*

## Exercise 7.17 — Flex and Grid Together

1. Open, examine, and test `lab07-ex17.html`.

   *Notice the uneven alignment of the View buttons within each cell. Using flexbox within each grid cell will solve this problem.*

2. Add the following styles and test.

```
.box {
   display: flex;
   flex-direction: column;
}
```

```
.box img {
   align-self: center;
}
.box h2 {
   text-align: center;
}
.box p {
   text-align: center;
}
.box button {
   margin-top: auto;
   justify-self: flex-end;
   align-self: center;
}
```

## Test Your Knowledge #2

Modify `lab07-test02.html` by adding CSS in `lab07-test02.css` to implement the layout shown in Figure 7.5 (some of the styling as already been provided).

1 This layout will require two nested grids. Create the outer grid that will have one row and three columns containing the `<nav>`, `<aside>`, and `<main>` elements. There should be no grid gap, and the first two columns should have a minimum size of 80px and a maximum size of 200px. The third column should fill the remaining space. To make the grid fill the entire vertical space, set the height of the container to 100vh.

2 The inner grid containing the four image squares should consist of two columns and rows. The images in the background of each square are 250px by 250px.

3 To center the text within each square, use flex layout along with `align-items` and `justify-content`.

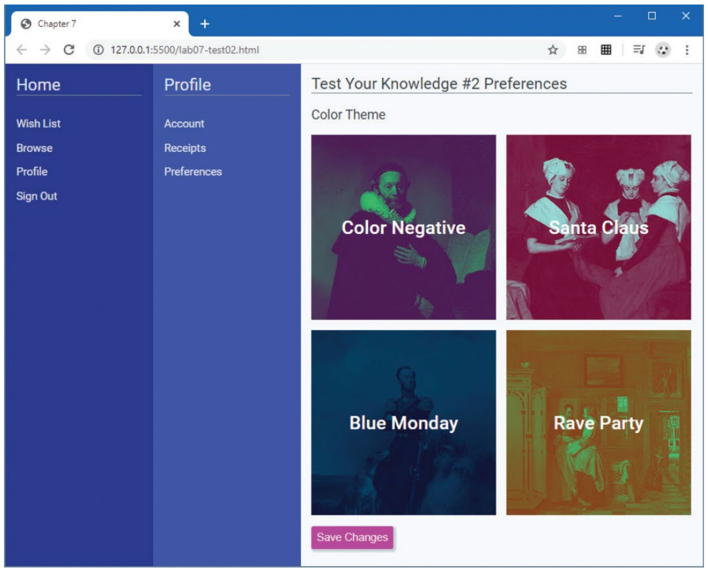
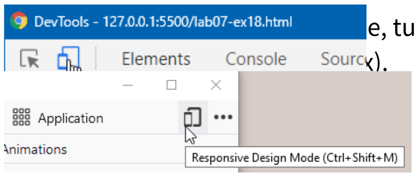


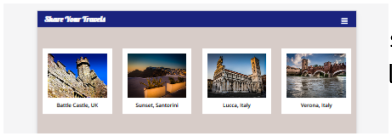
*Figure 7.5 – Finished Test Your Knowledge #2*

## Responsive Design

### Exercise 7.18 — Setting the Viewport

1 Open, examine, and test `lab07-ex18.html`.

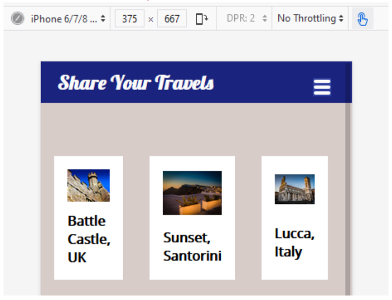
2 
DevTools - 127.0.0.1:5500/lab07-ex18.html
Elements
Console
Sourc
Application
Animations
Responsive Design Mode (Ctrl+Shift+M)
e, turn on the Device Toolbar (Chrome) or the x).

3 

smaller mobile device. Notice how the page by default lable width.

4 Add the following to the `<head>` section and test with the smaller device width.

```
<meta name="viewport" content="width=device-width, initial-
scale=1" />
```



*While this is an improvement, to truly improve the mobile experience, you will need to add media queries so there is different CSS for different sizes devices sizes.*

### Exercise 7.19 — Media Queries

1 Continue modifying your previous exercise.

2 Modify the following markup and test using different device widths.

```
<link rel="stylesheet" href="lab07-ex18-mobile.css"
  media="screen and (max-width:480px)" />
<link rel="stylesheet" href="lab07-ex18-tablet.css"
  media="screen and (min-width:481px) and (max-width:768px)" />
<link rel="stylesheet" href="lab07-ex18-desktop.css"
  media="screen and (min-width:769px)" />
```

3 Examine `lab07-ex18-desktop.css`.

*You will be modifying the grid structure and the margin in the other style files.*

4 Edit `lab07-ex18-mobile.css` (don't test yet).

```
section {
  display: grid;
  margin-top: 1em;
  grid-template-columns: 1fr;
}
```

*Notice that when the browser window shrinks to mobile size, the two-column layout is replaced with a single column. This requires removing the floats and the margins.*

5 Edit `lab07-ex18-tablet.css` then test at a variety of browser widths (See Figure 7.6).

```
section {
   display: grid;
   margin-top: 1em;
   grid-template-columns: 1fr 1fr;
 }
```

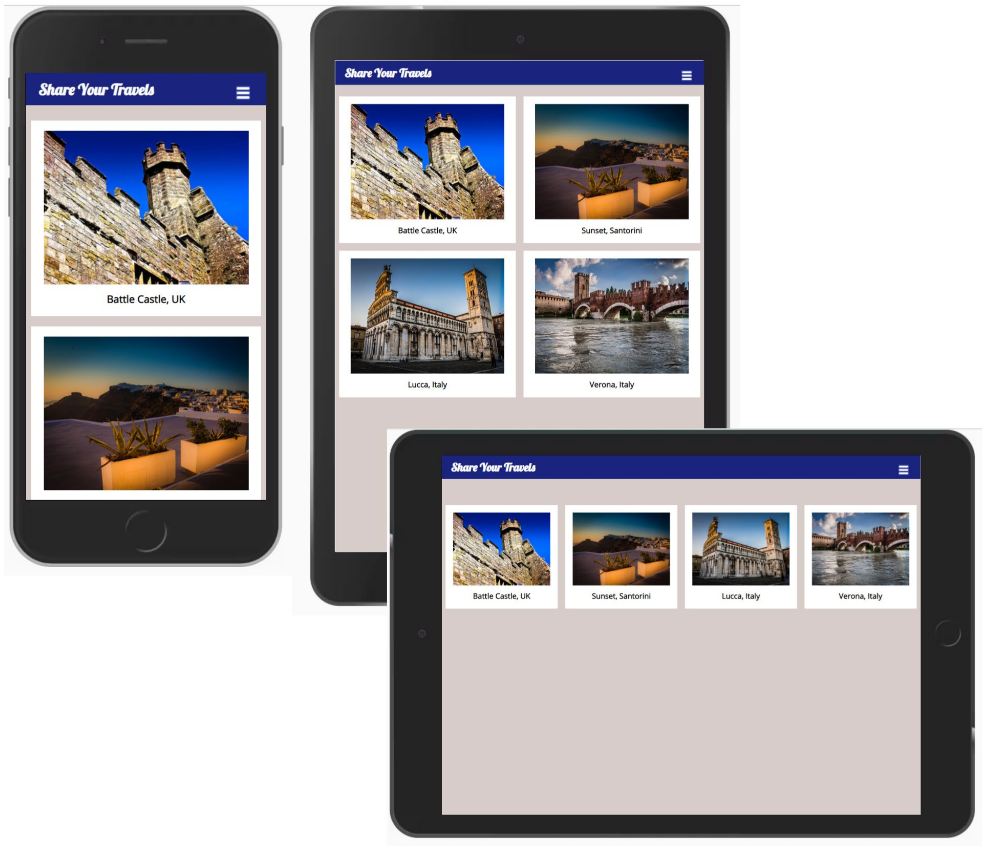


*Figure 7.6 – Viewing different media queries using Device Toolbar*

## Exercise 7.20 — Responsive Images

1 Open, examine, and test `lab07-ex20.html`.

2 Add the following to `lab07-ex20-queries.css`.

```
/* for tablet */
@media only screen and (min-width:481px) and (max-width:840px)
{
   .card {
      padding: calc( var(--gapSize) / 2 );
      width: 500px;
   }
   .card img {
      max-width: 100%;
   }
   button {
      background-color: var(--color-neutral-500);
   }
}

/* for phones */
@media only screen and (max-width:480px) {
   body {
      padding: 3px;
   }
   .card {
      padding: calc( var(--gapSize) / 4 );
      width: 300px;
   }
   .card img {
      max-width: 100%;
   }
   button {
      background-color: var(--color-support-2-500);
   }
}
```

*This example illustrates that you can add media queries within any given CSS file. It changes the button color to help you easily see when a media query is being used.*

3 Modify image element in `lab07-ex20.html` as follows:

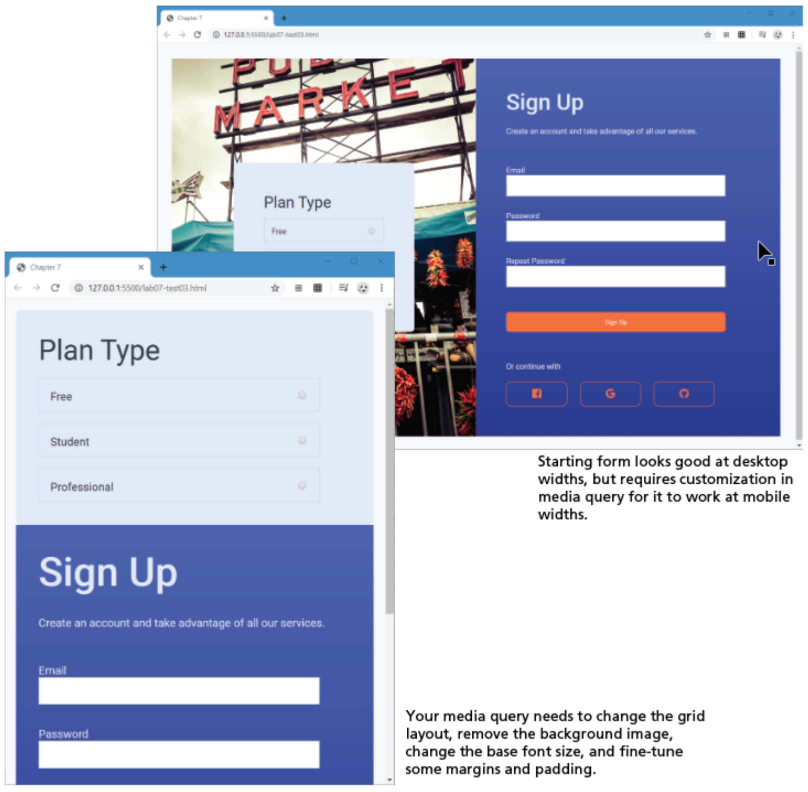
```
<picture>
   <source media="(min-width:481px) and (max-width:840px)"
srcset="images/albertHall-640.jpg" >
   <source media="(max-width:480px)" srcset="images/albertHall-
320.jpg" >
   <img src="images/albertHall-800.jpg" alt="Albert Hall">
</picture>
```

*The browser will now download the appropriate image file based on the browser width. This can dramatically improve perceived performance for mobile users.*

## Test Your Knowledge #3

Modify `lab07-test03.html` by adding CSS in `lab07-test03.css` to implement the layout shown in Figure 7.6 (some of the styling as already been provided).

1. You have been provided with a signup form layout that looks similar to the top screen in Figure 7.6. It uses a two-column grid layout (and flex within the grid cells). You need to add a media query that changes to a one-column grid when the browser width is below 1000 pixels. The second screen in Figure 7.6 illustrates how it should appear at the smaller browser widths.
2. Your media query will have to change the `margin` and `grid-template-columns` properties of the `container` class. The `formImage` class will also need to be modified in the media query so that it no longer has a background image and instead has a `background-color`.
3. It is quite common to increase the font size for smaller layouts. This can result in a lot of changes, but because the CSS uses variables, you only need to change the --base-font-size variable to 120% in your media query and all the other font sizes will also change. You will also need to change a few paddings and margins also (because of the use of CSS variables you should be able to simply make use of the `--space-med` and `--spacing-small` variables for those changes).


Chapter 7
127.0.0.1:5500/lab07-test03.html
PUBLIC MARKET
Sign Up
Create an account and take advantage of all our services.
Email
Password
Repeat Password
Sign Up
Or continue with
Plan Type
Free
Chapter 7
127.0.0.1:5500/lab07-test03.html
Plan Type
Free
Student
Professional
Sign Up
Create an account and take advantage of all our services.
Email
Password

Starting form looks good at desktop widths, but requires customization in media query for it to work at mobile widths.

Your media query needs to change the grid layout, remove the background image, change the base font size, and fine-tune some margins and padding.

*Figure 7.7 – Finished Test Your Knowledge #3*

## CSS3 Features

### Exercise 7.21 — Transforms

1 Open, examine, and test `lab07-ex21.html`.

2 Add the following style to the end of `lab07-ex21.css` and test.

```
figure {
    transform: rotate(45deg);
}
```

*Notice that the transform affects all the content within the transformed container.*

3 Modify the style as follows and test.

```
figure {
    transform: skew(-20deg);
}
```

4 Add the following style and test.

```
figure img {
    transform: translatex(100px) translatey(-30px);
}
```

*Notice that you can combine transforms and that the y-axis extends downwards.*

5 Modify the styles as follows and test.

```
figure {
    transform: rotate(15deg);
}
figure img {
    transform: rotate(45deg) scale(0.5);
}
```

### Exercise 7.22 — Transitions

1 Open, examine, and test `lab07-ex22.html`.

*In this example, we will add a transition to the button.*

2 Modify the following style in `lab07-ex22.css`.

```
button {
    ...
    transition-property: background-color;
    transition-duration: 1s;
    transition-timing-function: ease-out;
    transition-delay: 0s;
}
```

*This tells the browser to transition the background-color property. For this to work, however, we have to also indicate a different value of the background-color property to transition to or from.*

3 Add the following style to the end of `lab07-ex15.css` and test.

```
button:hover {
   background-color: #60b946;
}
```

## Exercise 7.23 — More Transitions

1 Open, examine, and test `lab07-ex23.html`.

*Notice the different hover state of the images. This is accomplished via four property settings in the figure:hover selector (see next step).*

2 Modify the following style in `lab07-ex23.css` and test.

```
figure:hover {
   background-color: #263238;
   color: white;
   box-shadow: 10px 10px 32px -4px rgba(0,0,0,0.75);
   transform: scale(1.75);
   transition: all 1s ease-in 0.25s;
}
```

*This transitions all four of these properties. When you test it, you will notice the transition happens on the hover over, but once you move the mouse off the image, it returns immediately to the non-hover property values. Let's add a transition to the figure as well.*

3 Modify the following style and test.

```
figure {
   ...
   align-items: center;
   justify-content: center;
   transition: all 0.6s ease-out 0.25s;
}
```

## Exercise 7.24 — Filters

1 Open, examine, and test `lab07-ex24.html`.

*Notice that each image (other than the first) has a different assigned CSS class. In this exercise we will define these classes to experiment with the different CSS3 filters.*

2 Add the following style to the end of `lab07-ex24.css` and test.

```
.saturate {
  filter: saturate(3);
}
```

3 Add the following style and test.

```
/* preview the difference */
img:hover {
  filter:none;
}
```

*To help you better see the difference the filter makes, this style definition removes the filter from the filtered image when you hover over it.*

4 Add the following styles and test.

```
.grayscale {
  filter: grayscale(100%);
}
.contrast {
  filter: contrast(160%);
}
.brightness {
  filter: brightness(30%);
}
.blur {
  filter: blur(3px);
}
```

5 Add the following styles and test.

```
.invert {
  filter: invert(100%);
}
.sepia {
  filter: sepia(100%);
}
.huerotate {
  hue-rotate(90deg);
}
.filter-opacity {
  filter: opacity(50%);
}
```

6 Add the following styles and test.

```
.combo-1 {
  filter: brightness(1.5) contrast(3) grayscale(0.6)
          invert(0.23) sepia(0.2);
}
.combo-2 {
  filter: brightness(1.3) contrast(1.1) hue-rotate(180deg)
          saturate(2);
}
```

*This example shows that you can combine multiple filters in once property setting.*

## Exercise 7.25 — Animations

1 Open, examine, and test `lab07-ex25.html`.

2 Modify the following style in `lab07-ex25.css`.

```
.box {
    background-color: green;
    width: 100px;
    height: 100px;
    margin: 100px;

    animation-name: animOne;
    animation-duration: 1.5s;
}
```

*This tells the browser to use the animation named animOne and have it run across 1.5 seconds. We still need to define this animation.*

3 Add the following style and test.

```
@keyframes animOne {
  from {
    opacity: 1;
    margin-left: 100px;
  }
  to {
    opacity: 0.5;
    transform: scale(1.5) rotate(90deg);
    margin-left: 400px;

  }
}
```

*This specifies a start state and an end state. This animation varies four things: the opacity, the size, the rotation, and the left margin.*

4 Modify the following style in `lab07-ex18.css` and test.

```
.box {
    ...
    animation-name: animOne;
    animation-duration: 1.5s;
    animation-iteration-count: infinite;
    animation-delay: 0.5s;
}
```

5 Add the following style.

```
@keyframes animTwo {
  25% {
    background-color: red;
    transform: scale(1.5) rotate(90deg);
    margin-left: 400px;
  }
  50% {
    background-color: yellow;
    transform: scale(1);
    margin-top: 400px;
    border-radius: 100%;
  }
  75% {
    background-color: blue;
    margin-left: 100px;
    transform: scale(0.5);
  }
  100% {
    background-color: green;
    border-radius: 0;
    margin-top: 100px;
    transform: scale(1) rotate(0);
  }
}
```

6 Comment out the animation styles entered in steps 2 and 4 and replace them with the following style and test:

```
animation: animTwo 5s ease-out 1s 3;
```

*This uses the shortcut property that allows you to combine the different animation properties into a single line.*

## CSS Pre-Processors

To complete the remaining exercises, you are going to need to install Sass or use an online environment such as CodePen that allows you to easily work with Sass.

The instructions below assume you have already installed Sass using npm as per the official documentation (`https://sass-lang.com/install`).

### Exercise 7.26 — Using Sass

1. Open and examine `lab07-ex26.html`.
2. Open and examine `lab07-ex26.scss`. Notice the extension is `scss` and not `css`.
3. Add the following to the top of this file.

```
$dark-color: #1A237E;
$nav-color: #3F51B5;
$mid-color: #9FA8DA;
$back-color: #E8EAF6;
$text-color: black;
$text-color-inverse: white;
$link-color: #F48FB1;

$mid-neutral : #9AA5B1;
$shadow : 0 0 3px 3px $mid-neutral;

$font-family-display: Lobster, "Times New Roman", serif;
$font-family-body: "Open Sans", Verdana, Arial, sans-serif;
```

*This defines a variety of Sass variables.*

4 Add the following after these variables then save.

```
h1, h2, h3 {
   font-family: $font-family-display;
}
body {
   font-family: $font-family-body;
   font-size: 100%;
   background-color: $back-color;
}
header {
   background: $dark-color;
   color: $text-color-inverse;
   flex-basis: 100%;
}
footer {
   background: $dark-color;
   color: $text-color-inverse;
   flex-basis: 100%;
}
```

*Notice the references to these Sass variables.*

5 Assuming you have already installed Sass, enter the following command (in Terminal, Command Window, PowerShell, Git Bash, or some other command-line environment).

```
sass lab07-ex26.scss lab07-ex26.css
```

*This will run the Sass pre-processor on `lab07-ex26.scss` and generate a CSS file named `lab07-ex26.css`.*

6 Open and examine `lab07-ex26.css`. Notice how the Sass preprocessor has generated valid CSS.

7 Test `lab07-ex26.html` in the browser.

8 Add the following to `lab07-ex26.scss`.

```
nav {
   background: $nav-color;
   color: $text-color-inverse;
   flex-basis: 9em;
   li {
      list-style: none;
      margin: 0.5em 0;
      padding: 0;
      a:link {
            color: $link-color;
      }
   }
}
```

*This demonstrates how CSS rules can be nested instead of using descendant selectors.*

9 Run the sass preprocessor (see step 5) and then test `lab07-ex26.html` in the browser.

## Exercise 7.27 — More Sass

1 In this exercise, you will be continuing to modify the same files as the previous exercise. Add the following mixin to `lab07-ex26.scss`.

```
@mixin imageBox($width) {
   border: solid 1px $mid-neutral;
   margin: 3px;
   box-shadow: $shadow;
   max-width: $width;
}
```

*A mixin is like a function that can be called or used in other locations in order to eliminate duplicate styling.*

2 Add the following after this mixin.

```
aside {
   background: $mid-color;
   flex-basis: 13em;

   h3 {
      text-align: center;
   }

   section {
      justify-content: space-around;
      div {
         margin: 0.25em;
         img {
            @include imageBox(50px);
         }
      }
   }
}

main {
   $spacing: 0.5em;
   margin: $spacing;
   background: $back-color;
   flex-grow: 1;
   h2 { margin-top: 0;  }
   div {
      img {
         @include imageBox(100px);
         margin-right: $spacing * 3;
      }
   }
}
```

*Notice how the imageBox mixin is used twice, but is passed different values. Notice also how the margin-right property is set to a calculated value.*

3 Run the sass preprocessor and then test `lab07-ex26.html` in the browser. The result should look similar to that shown in Figure 7.

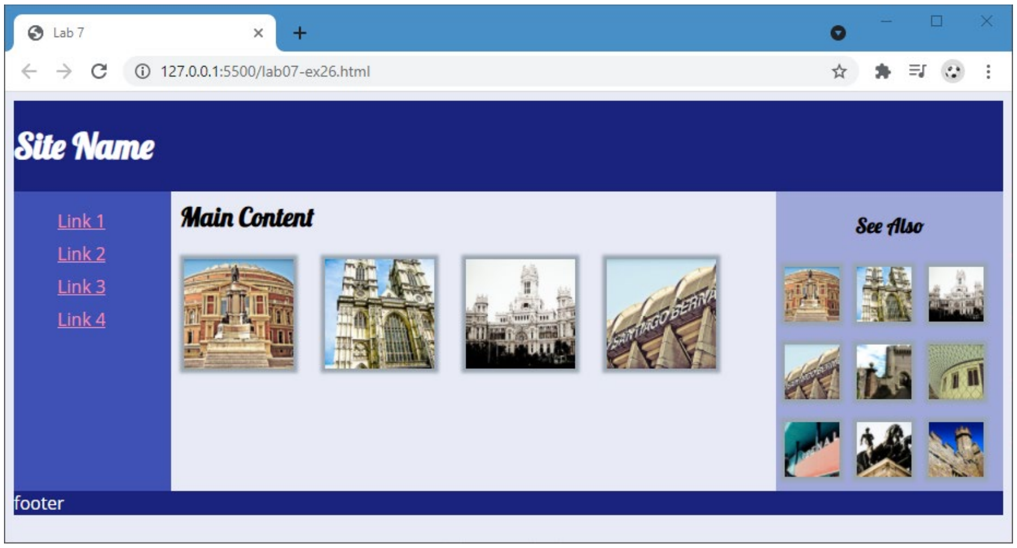


*Figure 7.8 – Finished Exercise 7.27*