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
- How to make a responsive layout
- How to use a CSS Framework to simplify page layout

Approximate Time

The exercises in this lab should take approximately 75 minutes to complete.

Fundamentals of Web Development, 2nd Ed

Randy Connolly and Ricardo Hoar

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

Date Last Revised: Feb 20, 2017

CREATING TABLES

PREPARING DIRECTORIES

1. If you haven't done so already, create a folder in your personal drive for all the labs for this book.
2. From the main `labs` folder (either downloaded from the textbook's web site using the code provided with the textbook or in a common location provided by your instructor), copy the folder titled `lab07` to your course folder created in step one.

The `<table>` element in HTML represents information that exists in a two-dimensional grid. Tables can be used to display calendars, financial data, pricing tables, and many other types of data. Just like a real-world table, an HTML table can contain any type of data: not just numbers, but text, images, forms, even other tables.

Exercise 7.1 — DOCUMENT FLOW

1. Open, examine, and test `lab07-exercise01.html`.
2. Resize the browser window and experiment by making the width smaller and larger.
3. Modify the following style in`lab07-exercise01.css` and test.

```
h1 span {
  color: #8B0000;
  font-size: 1.5em;
  display: block;
}
```

This changes the element in the main heading from an inline element to a block-level element.

4. Add the following style and test.

```
li {
  display: inline;
}
```

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

5. Add the following style and test.

```
div img {
  display: block;
}
```

POSITIONING

Exercise 7.2 — Relative Positioning

1. Open, examine, and test `lab07-exercise02.html`.
2. Modify the following style in `lab07-exercise02.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.

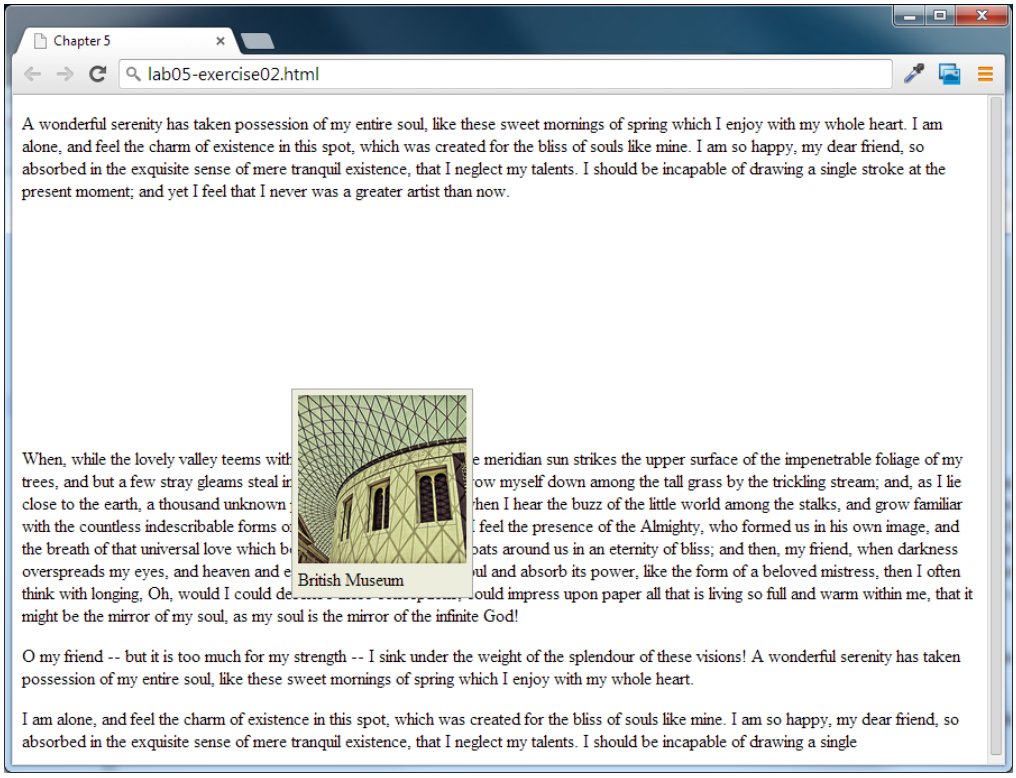

Figure 7.1 – Exercise 7.2 complete

Exercise 7.3 — Absolute Positioning

1. Open, examine, and test `lab07-exercise03.html`.
2. Modify the following style in `lab07-exercise03.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 — Stacking Using Z-Index

1 Open, examine, and test `lab07-exercise04.html`.

2 Modify the following style and test.

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

figcaption {
   font-size: 1.25em;
   background-color: yellow;
   padding: 5px;
   top: 90px;
   left: 140px;
   position: absolute;
}
```

Notice that the caption now overlaps both the underlying paragraph text and the figure image.

3 Modify the following styles (some CSS omitted here for clarity) and test.

```
figure {
   ...
   position: absolute;
   z-index: 5;
}

figcaption {
   ...
   position: absolute;
   z-index: 1;
}
```

Note that this did not move the <figure> on top of the <figcaption> as one might expect. This is due to the nesting of the caption within the figure.

4 Modify the following styles and test.

```
figure {
   ...
   position: absolute;
   z-index: 1;
}
```

```
figcaption {
   ...
   position: absolute;
   z-index: -1;
}
```

The <figure> zindex could be any value equal to or above 0 as long as the <figcaption> is below 0.

5 Modify the following styles and test.

```
figure {
   ...
   position: absolute;
   z-index: -1;
}

figcaption {
   ...
   position: absolute;
   z-index: 1;
}
```

If the <figure> zindex is given a value less than 0, then any of its positioned descendants change as well. Thus both the <figure> and <figcaption> move underneath the body text.

FLOATING

Exercise 7.5 — Floating Elements

1 Open, examine, and test `lab07-exercise05.html`.

2 Modify the following styles 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 7.6 — Floating in a Container

1 Open, examine, and test `lab07-exercise06.html`.

Notice the empty margin space on the left.

2 Modify the following styles and test.

```
figure {
   border: 1pt solid #262626;
   background-color: #c1c1c1;
   padding: 5px;
   width: 150px;
   margin: 10px;
   float: left;
}
```

Notice that a floated item moves to the left or right of its container. In this case the container for the floated <figure> is the <article> element.

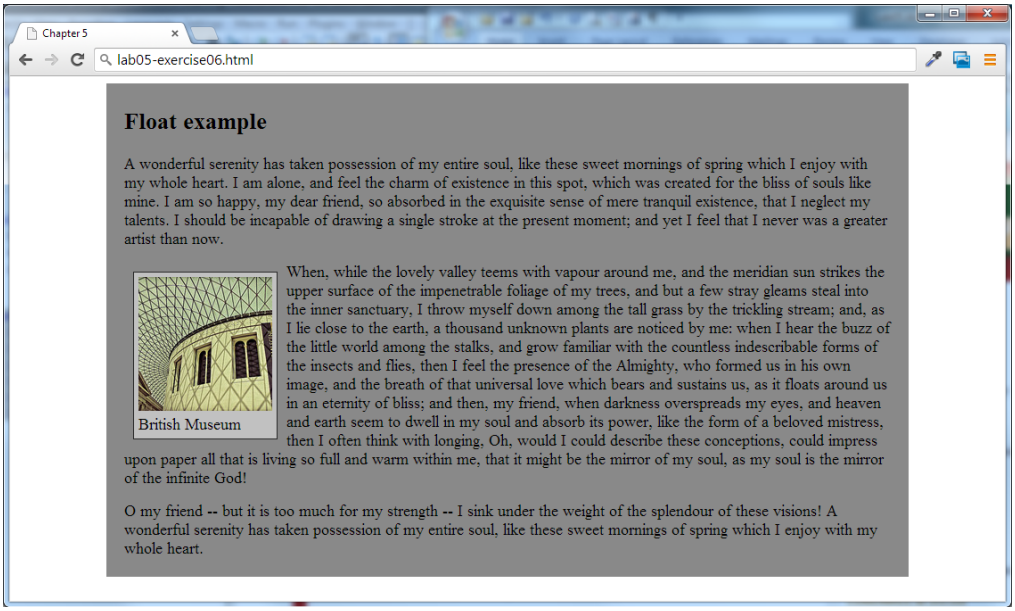

Figure 7.2 – Exercise 7.6 complete

Exercise 7.7 — Floating Multiple Items

1 Open, examine, and test `lab07-exercise07.html`.

Notice that each <figure> is a block-level element and hence on its own line.

2 Modify the following style and test.

```
figure {
   border: 1pt solid #262626;
   background-color: #c1c1c1;
   padding: 5px;
   width: 150px;
   margin: 10px;
   float: left;
}
```

When you float multiple items that are in proximity, each floated item in the container will be nestled up beside the previously floated item. All other content in the containing block (including other floated elements) will flow around all the floated elements.

3 Experiment by changing the width of the browser window.

This can result in some pretty messy layouts as the browser window increases or decreases in size (that is, as the containing block resizes). The solution is to use the clear property (next exercise).

Exercise 7.8 — Floating and Clearing

1 Open, examine, and test `lab07-exercise08.html`.

2 Add the following style and test.

```
.first { clear: left; }
```

3 Modify the third `<figure>` element as follows and test:

```
<figure class="first">
   <img src="images/tiny/828.jpg" alt="" />
   <figcaption>British Museum</figcaption>
</figure>
```

4 Modify the `<p>` element after the figures as follows and test:

```
<p class="first">When, while the lovely valley ...
```

Exercise 7.9 — Using Positioning

1 Open, examine, and test `lab07-exercise09.html`.

In this exercise, we are going to position the caption on top of the image.

2 Modify the following style and test.

```
figcaption {
   width: 100px;
   background-color: black;
   color: white;
   opacity: 0.6;
   width: 140px;
   height: 20px;
```

```
   padding: 5px;
   position: absolute;
   top: 130px;
   left: 10px;
}
```

Since absolute positioning is positioning in relation to the closest positioned ancestor, we will also need to position the <figure>.

3 Modify the following style and test.

```
figure {
   border: 1pt solid #262626;
   background-color: #c1c1c1;
   padding: 10px;
   width: 200px;
   margin: 10px;
   position: relative;
}
```

Notice that we are not moving the figure. Instead by setting its position to relative, we are creating a positioning context for the absolute positioning in the previous step.

4 Add the following markup and test.

```
<figure>
   <img src="images/828.jpg" alt="" />
   <figcaption>British Museum</figcaption>
   <img src="images/new-banner.png" alt="" class="overlayed"/>
</figure>
```

5 Add the following style and test.

```
.overlayed {
   position: absolute;
   top: 10px;
   left: 10px;
}
```

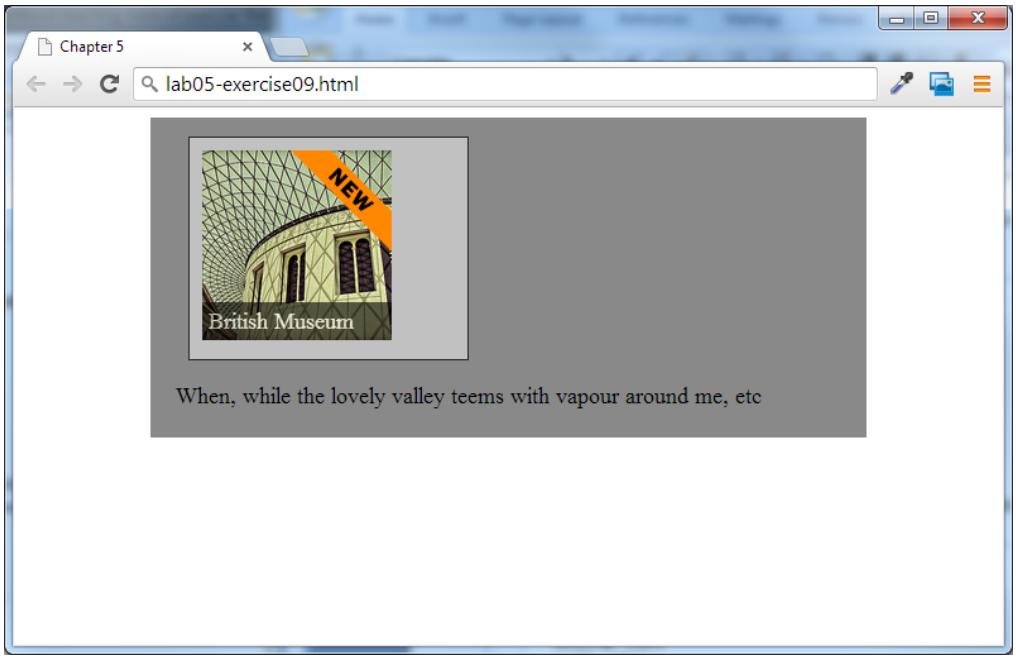

Figure 7.3 – Exercise 7.9 complete

LAYOUTS

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. They are the raw techniques that you can use to create more complex layouts.

Exercise 7.10 — Two-Column Layout

1. Open, examine, and test `lab07-exercise10.html`.
2. Modify the following styles and test.

```
div#main {
   padding: 0.5em;
   margin-left: 12em;
}
nav {
   color: white;
   background-color: #A9A9A9;
   padding: 0.5em;
   width: 10em;
}
```

3. Modify the following style and test.

```
nav {
   color: white;
```

```
    background-color: #A9A9A9;
    padding: 0.5em;
    width: 10em;
    float: left;
}
```

The previous exercise created a multi-column layout using floats. You can also create a multi-column layout using positioning, as shown in the next exercise.

Exercise 7.11 — Three-Column Layout

1. Open, examine, and test `lab07-exercise11.html`.
2. Modify the following styles and test.

```
nav {
  color: white;
  background-color: #A9A9A9;
  padding: 0.5em;
  width: 10em;
}
div#main {
    padding: 0.5em;
    margin-left: 12em;
    margin-right: 12em;
}
aside {
  color: white;
  background-color: #D3C6BA;
  padding: 0.5em;
  width: 10em;
}
```

3. Modify the following styles and test.

```
nav {
  color: white;
  background-color: #A9A9A9;
  padding: 0.5em;
  width: 10em;
  position: absolute;
  left: 0;
  top: 0;
}

aside {
  color: white;
  background-color: #D3C6BA;
  padding: 0.5em;
  width: 10em;
  position: absolute;
  right: 0;
  top: 0;
}
```

Can you figure out what is wrong (and the solution)? Remember that absolute positioning is relative to the last positioned element. There is no positioned element, so

the measurements are relative to the browser window.

4 Add the following styles and test.

```
div#container {
   position: relative;
}
```

RESPONSIVE DESIGN

Exercise 7.12 — MEDIA QUERIES

1 Open, examine, and test `lab07-exercise12.html`.

2 Modify the following markup and test by progressively reducing the width of the browser window .

```
<link rel="stylesheet" href="lab07-exercise12-desktop.css"
      media="screen and (min-width:481px)" />
```

Once the window shrinks below 481px wide, this stylesheet will no longer be in effect (and hence all CSS styling will disappear).

3 Add the following additional media query.

```
<head lang="en">
   <meta charset="utf-8">

   <title>Chapter 7</title>
   <link rel="stylesheet" href="lab07-exercise12-mobile.css"
         media="screen and (max-width:480px)" />
   <link rel="stylesheet" href="lab07-exercise12-desktop.css"
         media="screen and (min-width:481px)" />
</head>
```

We are adding an additional style sheet for mobile browser sizes.

4 Edit `lab07-exercise12-mobile.css` (this is the mobile style sheet) as follows and test.

```
main {
   width: 100%;
   margin-left: auto;
   margin-right: auto;
}

nav {
  color: white;
  background-color: #A9A9A9;
  padding: 0.5em;
  width: 100%;
  /* be sure to delete the float as well */
}

div#main {
    padding: 0.5em;
    margin-left: 0;
}
```

```
figure img {
   max-width: 100%;
}
```

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.

Exercise 7.13 — Setting the Viewport

1 Continue this exercise by using your files from Exercise 12.

2 Add the following viewport.

```
<meta name="viewport" content="width=device-width" />
```

3 Add the following markup to the `<nav>` element.

```
<nav>
   <h3>Navigation</h3>
   <ul>
     <li><a href="#">Australia</a></li>
     <li><a href="#">Belgium</a></li>
     <li><a href="#">Canada</a></li>
     <li><a href="#">France</a></li>
     <li><a href="#">United Kingdom</a></li>
     <li><a href="#">United States</a></li>
   </ul>
   <select>
      <option>Australia</option>
      <option>Belgium</option>
      <option>Canada</option>
      <option>France</option>
      <option>United Kingdom</option>
      <option>United States</option>
   </select>
</nav>
```

We are going to use CSS to use the <select> element for mobile sizes, and the <ul> for larger sizes.

4 Edit `lab07-exercise12-desktop.css` (this is the desktop style sheet) as follows by adding the following style.

```
nav select {
  display: none;
}
```

5 Edit `lab07-exercise12-mobile.css` (this is the mobile style sheet) as follows by adding the following styles and test in browser. Be sure to test using small and larger browser widths.

```
nav ul     { display: none; }
nav select {
   display: inline-block;
   width: 90%;
}
```

CSS3 FEATURES

Exercise 7.14 — TRANSFORMS

1 Open, examine, and test `lab07-exercise14.html`.

2 Add the following style to the end of `lab07-exercise14.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 7.15 — TRANSITIONS

1 Open, examine, and test `lab07-exercise15.html`.

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

2 Modify the following style in `lab07-exercise15.css`.

```
button {
    height: 3em;
    ...
    background-color: #146d37;
    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-exercise15.css` and test.

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

Exercise 7.16 — More Transitions

1 Open, examine, and test `lab07-exercise16.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-exercise16.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 in `lab07-exercise16.css` and test.

```
figure {
   margin: 1em;
   ...
   align-items: center;
   justify-content: center;
   transition: all 0.6s ease-out 0.25s;
}
```

Exercise 7.17 — Filters

1 Open, examine, and test `lab07-exercise17.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-exercise17.css` and test.

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

At the time of writing (February 2017), the additional specification with the -webkit-

prefix was necessary for Chrome browsers. You should test with and without this extra prefix and see if it works now without it. If so, then you can remove the additional prefix line in the following steps.

3 Add the following style and test.

```
/* preview the difference */
img:hover {
  filter:none;
  -webkit-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%);
  -webkit-filter: grayscale(100%);
}
.contrast {
  filter: contrast(160%);
  -webkit-filter: contrast(160%);
}
.brightness {
  filter: brightness(30%);
  -webkit-filter: brightness(30%);
}
.blur {
  filter: blur(3px);
  -webkit-filter: blur(3px);
}
```

5 Add the following styles and test.

```
.invert {
  filter: invert(100%);
  -webkit-filter: invert(100%);
}
.sepia {
  filter: sepia(100%);
  -webkit-filter: sepia(100%);
}
.huerotate {
  hue-rotate(90deg);
  -webkit-filter: hue-rotate(90deg);
}
.filter-opacity {
  filter: opacity(50%);
  -webkit-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);
  -webkit-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);
  -webkit-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 7.18 — Animations

1 Open, examine, and test `lab07-exercise18.html`.

2 Modify the following style in `lab07-exercise18.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-exercise18.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.

GRID SYSTEMS

Exercise 7.19 — USING BOOTSTRAP

1 Modify `lab07-exercise19.html` by adding the following markup, and then test.

```
<head>
   <meta charset="utf-8">
   <title>Chapter 7</title>
   <link href="bootstrap/css/bootstrap.css" rel="stylesheet">
</head>
<body>
<div class="container">
   <div class="row">
      <div class="col-md-2">
        left column
      </div>
      <div class="col-md-7">
        main content
      </div>
      <div class="col-md-3">
        right column
      </div>
   </div>
</div>
</body>
```

Notice that Bootstrap handles the entire layout without needing to bother with floats and positioning.

2 Modify the column layout as follows and test.

```
<div class="container">
   <div class="row">
      <div class="col-md-3">
        left column
      </div>
      <div class="col-md-9">
        main content
      </div>
   </div>
</div>
```

The columns in a Bootstrap row must add up 12. Any given layout can contain any number of rows. When testing, try shrinking the browser window. You will discover that Bootstrap uses media queries to stack columns on top of each other when there is not enough space to display them on the same row.

3 Add the following markup after the container and test.

```
<div class="container">
   <nav class="navbar navbar-default" role="navigation">
      <a class="navbar-brand" href="#">Logo</a>
      <ul class="nav navbar-nav">
         <li class="active"><a href="#">Home</a></li>
         <li><a href="#">Browse</a></li>
         <li><a href="#">Search</a></li>
         <li><a href="#">About Us</a></li>
```

```
      </ul>
    </nav>
    <div class="row">
      ...
```

Bootstrap also comes with many additional classes that can simplify the process of quickly creating a design. For more information, see http://getbootstrap.com/.

4 Modify the first column by adding a panel and list using some of Bootstrap's built-on formatting classes. Test.

```
<div class="row">
  <div class="col-md-3">
     <div class="panel panel-default">
        <div class="panel-heading">
           <h3 class="panel-title">Destinations</h3>
        </div>
        <div class="panel-body">
           <ul class="list-group">
               <li class="list-group-item">Austria</li>
               <li class="list-group-item">Canada</li>
               <li class="list-group-item">France</li>
               <li class="list-group-item">Italy</li>
               <li class="list-group-item">Spain</li>
           </ul>
        </div>
     </div>
  </div>
  ...
```

5 Modify the second column by making use of an additional Bootstrap class and test.

```
<div class="col-md-9">
    <div class="jumbotron">
        <h1>Main Content</h1>
        <p>
        CSS frameworks provide similar grid features. Bootstrap and
Material Lite both use a 12-column grid. The grid is constructed using div
elements with classes defined by the framework.
        </p>
    </div>
</div>
```

6 Add to the second column by adding a nested row after the jumbotron and test.

```
    <h2>Nested columns</h2>
    <div class="row">
        <div class="col-md-3">
            <img src="images/464.jpg" alt="London">
            <h5>London</h5>
            <button type="button" class="btn btn-primary">
                <span class="glyphicon glyphicon-info-sign"
                      aria-hidden="true"></span> View
            <button>
            <button type="button" class="btn btn-success">
                <span class="glyphicon glyphicon-shopping-cart"
                      aria-hidden="true"></span> Cart
            </button>
        </div>
        <div class="col-md-3">
```

```
            <img src="images/9498358806.jpg" alt="Florence">
            <h5>Florence</h5>
            <button type="button" class="btn btn-primary">
                <span class="glyphicon glyphicon-info-sign"
                      aria-hidden="true"></span> View
            </button>
            <button type="button" class="btn btn-success">
                <span class="glyphicon glyphicon-shopping-cart"
                      aria-hidden="true"></span> Cart
            </button>
        </div>
        <div class="col-md-3">
            <img src="images/9504449928.jpg" alt="Florence">
            <h5>Florence</h5>
            <button type="button" class="btn btn-primary">
                <span class="glyphicon glyphicon-info-sign"
                      aria-hidden="true"></span> View
            </button>
            <button type="button" class="btn btn-success">
                <span class="glyphicon glyphicon-shopping-cart"
                      aria-hidden="true"></span> Cart
            </button>
        </div>
        <div class="col-md-3">
            <img src="images/9494472443.jpg" alt="Venice">
            <h5>Venice</h5>
            <button type="button" class="btn btn-primary">
                <span class="glyphicon glyphicon-info-sign"
                      aria-hidden="true"></span> View
            </button>
            <button type="button" class="btn btn-success">
                <span class="glyphicon glyphicon-shopping-cart"
                      aria-hidden="true"></span> Cart
            </button>
        </div>
    </div>
```

Any given Bootstrap column can contain other Bootstrap rows. Every row, whether it is nested or not, will contain columns that add up to 12.

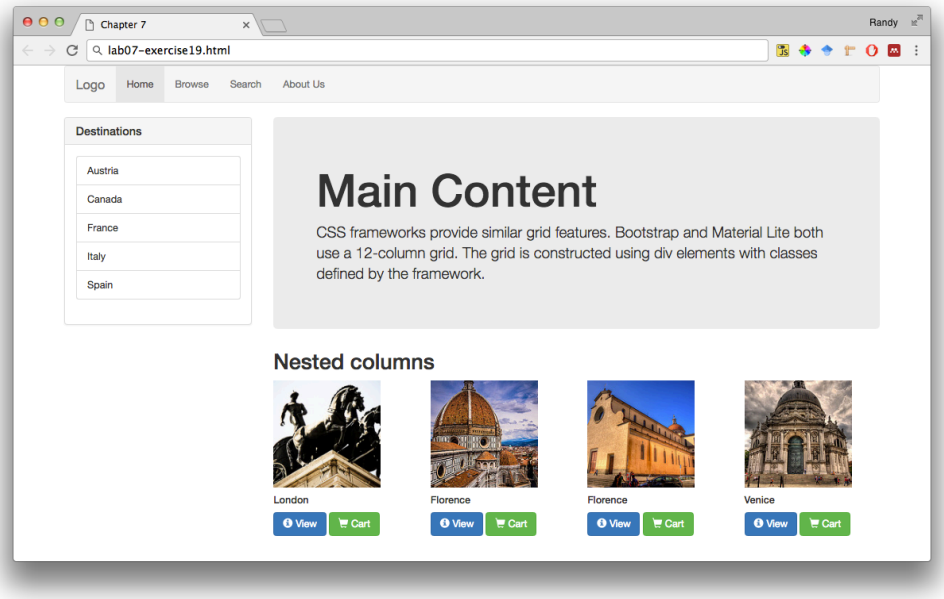

Figure 7.4 – Exercise 7.19 complete