LAB 4

CSS 1: Selectors + Basic Styles

What You Will Learn

- How to use CSS to style HTML documents
- How to use CSS selectors
- The CSS box model

Approximate Time

The exercises in this lab should take approximately 90 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 `lab04` folder has been provided for you (within the zip file/folder downloaded from Gumroad).

1. If you haven't done so already, create a folder in your personal drive for all the labs for this book.
2. Copy the folder titled `lab04` from the zip file/folder to your course folder.

   *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.*

Quick Tour of CSS

CSS is a W3C standard for describing the appearance of HTML elements. Another common way to describe CSS's function is to say that CSS is used to define the presentation of HTML documents. CSS can be added directly to any HTML element (via the style attribute), within the `<head>` element, or in a separate text file that contains only CSS.

Exercise 4.1 — Adding Styles

1. Open, examine, and test `lab04-ex01.html` in browser.
2. Add the following internal style definition and test.

   ```
   <h1 style="color: red;">Featured</h1>
   ```

   Remember: these labs use the convention of `red bolded` text to indicate content to change/enter.

3. Modify your style as follows and test.

   ```
   <h1 style="color: #FF4500;">Featured</h1>
   ```

   This illustrates another approach to specifying color in CSS. It uses hexadecimal codes to indicate the R (red), G (Green), and B (Blue) recipe for the color. You will learn more about color in Chapter 6.

4. Modify your style as follows and test.

   ```
   <h1 style="color: rgb(255,0,0);">Featured</h1>
   ```

   This illustrates another way to specify color. It uses a CSS function to indicate the reg, green, and blue values.

5. Modify the following and test.

   ```
   <h1 style="color: red; background-color:gray;">Featured</h1>
   ```

Exercise 4.2 — Embedded Styles

1 Add the following embedded style to the `<head>` element from the previous exercise.

```
<title>lab04</title>
<style>
   h1 {
     color: blue;
     background-color: yellow;
   }
</style>
</head>
```

2 Test. Why didn't it seem to work?

It didn't work because of cascade specificity rules. The internal style created in last exercise overrides the embedded style we just created. To fix it, you will have to remove the internal styles.

3 Remove the internal styles created in Exercise 4.1. Test.

The `<h1>` element should now have blue text against a yellow background.

4 Add the following style rule and test.

```
h1, h2 {
 font-family: "Trebuchet MS", "Lucida Grande", sans-serif;
}
```

This is a grouped selector which selects both h1 and h2.

5 Add the following style rule **after** the one created in previous step and test.

```
h1, h2 {
 font-family: "Trebuchet MS", "Lucida Grande", sans-serif;
}
h1 {
 font-family: Georgia, Cambria, "Times New Roman", serif;
}
```

Notice that the new style rule for `h1` overrides the earlier one due to the cascade principle of location (i.e., when rules have the same specificity, then the latest are given more weight).

6 Change the previous style rule to the following. Before you test it, ask yourself whether this will affect the `font-family` of the headings.

```
h1, h2 {
 font-family: "Trebuchet MS", "Lucida Grande", sans-serif;
}
body {
 font-family: Georgia, Cambria, "Times New Roman", serif;
}
```

Exercise 4.3 —External Styles

1 Create a new text document with the following content:

```
body {
  font-family: Georgia, Cambria, "Times New Roman", serif;
  background-color: #FAF9F7;
  margin: 0;
  padding: 10px;
}
h1, h2 {
  font-family: "Trebuchet MS", "Lucida Grande", sans-serif;
  letter-spacing: 5px;
  text-transform: uppercase;
  margin: 0 5px;
  color:#423D33;
}
article {
  margin: 5px;
  background-color: white;
  box-shadow: 5px 5px 5px 0px #E8E6E1;
  padding: 10px;
  width: 810px;
}
figure {
  margin: 5px;
}
figcaption {
  font-style: italic;
  font-size: 14px;
  color: #857F72;
}
p {
  margin-left: 5px;
}
```

Notice that this document contains no HTML. Also notice that the second style rule uses a grouped selector, meaning that both the h1 and h2 elements will receive the same style.

***Note**: for the next several exercises, there will be style rules containing properties you have not learned yet. Rest assured, you will learn all these properties, mainly in this lab, and a few others in lab 7.*

2 Save your file as `lab04-ex03.css`.

3 Open `lab04-ex03.html` (which has the same content as the last exercise).

4 Add the following to the `<head>` element.

```
<head>
   <meta charset="utf-8">
   <title>Lab04title>
   <link rel="stylesheet" href="lab04-ex03.css" />
</head>
```

5 Save and test `lab04-ex03.html` in browser.

As you work your way through the chapter and lab, you will learn what all of these style rules do. For now, notice how just a little bit of CSS completely transforms the visual appearance of this page.

6 View the `lab04-ex03.css` file (yes, the css file) in the browser.

At some point everyone will mistakenly load a css file into the browser rather than the HTML file. What will happen will vary depending upon the browser and one's computer setup. If I open up the CSS file in Chrome or FireFox, the CSS text file is displayed; in the Internet Explorer, it starts my default CSS editor, which back then happened to be Adobe Dreamweaver.

7 Reopen `lab04-ex03.html` in browser. Right-click on the picture and choose the Inspect option. This provides an easy way to examine the CSS settings of any webpage.

8 In the inspector, click on the `<h2>` tag. You should be able to see the

CSS Selectors

When defining CSS rules you will need to first use a selector to tell the browser which elements will be affected by the styles. CSS selectors allow you to select individual or multiple HTML elements, elements that belong together in some way, or elements that are positioned in specific ways in the document hierarchy. The previous exercises used HTML selectors. The next exercises make use of other selectors.

Exercise 4.4 — Id and Class Selectors

1 Open `lab04-ex04.html` in the browser. Examine `lab04-ex04.css`. Notice that it already has some styling for the `<div>` element.

2 Add the following to the first `<div>` in the markup.

```
<div id="first"></div>
```

3 Open `lab04-ex04.css`, add the following style, and test (that is, view the HTML file in the browser).

```
#first {
   background-color: seagreen;
}
```

4 Change the previous selector to the following and test. Nothing will change.

```
div#first
```

In this example the selector in step 3 and 4 are functionally equivalent. However, the one in step 4 is more specific, so in some situations will be more preferable, but less perferable in others.

5 Switch to `lab04-ex04.html` and add the following to the next two `<div>` elements.

```
<div id="first"></div>
<div class="circle"></div>
<div class="circle"></div>
```

6 Switch to `lab04-ex04.css` and add the following style. Test.

```
.circle {
   background-color:  lightskyblue;
   border-radius: 50%;
}
```

Remember that whenever the lab says "test" it means view the relevant HTML file in a browser.

7 Add the following style.

```
.orange {
   background-color: orange;
}
```

8 Modify the next two `<div>` elements as follows and test.

```
<div class="orange"></div>
<div class="circle orange"></div>
```

Notice that multiple classes can be assigned to any element.

Attribute Selectors

An attribute selector provides a way to select HTML elements either by the presence of an element attribute or by the value of an attribute. This can be a very powerful technique, but because of uneven support by some of the browsers, not all web authors have used them.

Exercise 4.5 — Attribute Selectors

1 Open `lab04-ex05.html` and view in browser. Some styling has already been provided.

2 Open `lab04-ex05.css`, add the following style, and test.

```
input[type="text"] {
   margin-right: 20px;
   height: 25px;
   width: 200px;
   border-style: none;
   border-bottom: 2px solid #3D70B2;
   background-color: white;
}
```

This will select every <input> element whose type attribute is exactly equal to "text". Examine the HTML to see which elements this will target.

3 Modify the selector to the following and test.

```
[type="text"] {
```

This selects all the elements whose type attribute is exactly equal to "text". In our case, there is no change since the only elements in this exercise with type=text are our two <input> elements.

4 Modify the attribute (add the asterisk) to the following and test.

```
[type*="x"] {
```

*This selects all elements whose type attribute **contains** the text "x" within in it. In this case, this also selects the checkbox.*

5 Return the selector back to what you had in step 2:

```
input[type="text"] {
```

6 Add the following style and test.

```
a[href*="twitter.com"] {
  background: url(https://twitter.com/favicon.ico) no-repeat
                left center;
  padding-left:19px;
}
```

This selects any <a> elements whose href attribute contains "twitter.com".

7 Add the following style and test.

```
a[href$=".pdf"] {
  background: url(images/pdf_icon.svg) no-repeat left center;
  padding-left:19px;
}
```

This selects any <a> elements whose href attribute ends with ".pdf".

Pseudo-Class Selectors

The next exercise illustrates the use of pseudo-class selectors, which do not apply to an HTML element, but target either a particular state or, in CSS3, a variety of family relationships. The most common use of this type of selectors is for targeting link states. By default, the browser displays link text blue and visited text links purple. Exercise 4.6 illustrates the use of pseudo-class selectors to style not only the visited and unvisited link colors, but also the hover style, which is the color of the element when the mouse is over the element.

Exercise 4.6 — Pseudo Selectors and Lists

1. Open `lab04-ex06.html` and view in browser. Some styling has already been provided.
2. Add the following style to `lab04-ex06.css` and test. Be sure to move your mouse over the list elements.

```
li:hover {
   background-color: #E8368F;
   cursor: pointer;
}
```

3. Modify the selector as follows and test:

```
li:nth-child(2n) {
```

This selects every second <li> element.

4. Modify the selector as follows and test:

```
li:nth-child(2n-1) {
```

This selects every second <li> element starting with the first element.

5. Add the following style and test.

```
a { text-decoration: none; }
```

This removes the underline from all hyperlinks.

6. Add the following style and test.

```
a:link { color: #07818F; }
```

7. Add the following style and test.

```
a:visited { color: #0FB5BA; }
```

To test, click one of the links, then return to this page. Press refresh. It should have a different color.

8 Add the following and test.

```
a:hover {
   text-decoration: underline;
   font-weight: bold;
   background-color: #FFE3EC;
   color: #870557;
}
```

To test, you will need to hover the mouse over a link.

7 Add the following and test.

```
a:active {
   background-color: #DA127D;
   color:   #FFE3EC;
}
```

To see the style, click and hold the mouse down over a link.

Contextual Selectors

A contextual selector allows you to select elements based on their ancestors, descendants, or siblings. That is, it selects elements based on their context or their relation to other elements in the document tree. While some of these contextual selectors are used relatively infrequently, almost all web authors find themselves using descendant selectors.

Exercise 4.7 — Contextual Selectors

1 Open `lab04-ex07.html` (which has the same content as the last exercise).

2 Switch to `lab04-ex07.css` and add the following style and test.

```
p {
   color: #983C2A;
}
```

This changes the text color of every occurrence of the <p> element.

3 Modify the style as follows. Before you test, see if you can figure out which text will be affected by the change to this selector.

```
#main p {
   color: #983C2A;
}
```

This selects all <p> elements that exist somewhere within an element whose id=main. In this case, it selects all of them except for the one within the <footer> element.

4 Modify the style as follows and test.

```
hr+p {
   color: green;
}
```

This uses the adjacent sibling selector which selects the first <p> that follows any <hr> element.

5 Add the following style and test.

```
time {
   color: red;
}
```

6 Modify the style as follows and test. As with the previous steps, see if you can figure out which text will be affected by the change before you test it.

```
div time {
   color: red;
}
```

This doesn't select the <time> element inside the footer.

7 Modify the style as follows and test.

```
#main>time {
   color: red;
}
```

This uses the child selector which selects any <time> elements that are direct children

8 Add the style as follows and test.

```
h3~p {
   color: orange;
}
```

This uses the general sibling selector which selects all the <p> elements that share the same parent as an <h3> element.

Test Your Knowledge #1

1 You have been provided markup in `lab04-test01.html` and styles within comments in to `styles/lab10-test01.css`.

2 Uncomment the styles and add CSS selectors so that it looks similar to that shown in Figure 4.1. You **cannot** modify the markup, so this will require working with selectors.

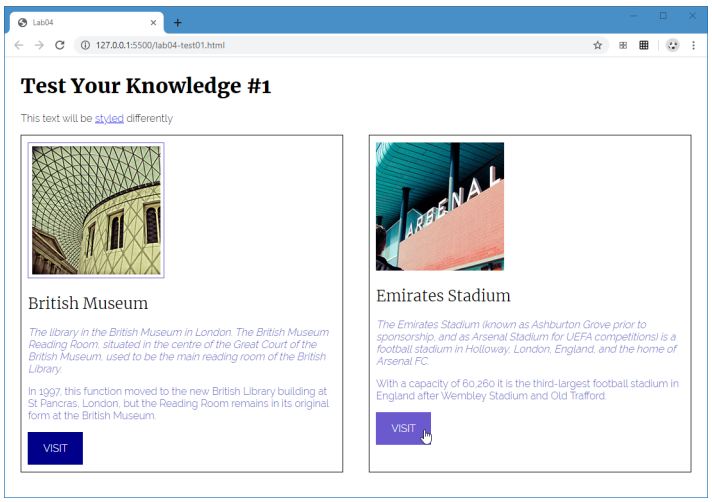

Figure 4.1 – Completed Test Your Knowledge #1.

CSS Cascade and Box Model

In Section 4.5 of the book, you learned that the cascade in CSS refers to the three principles used for adjudicating conflicting style rules: inheritance, location, and specificity.

Exercise 4.8 — CSS Cascade

1 Open and examine `lab04-ex08.html`.

2 Add the following style to `lab04-ex08.css` and test.

```
body {
  font-family: Georgia, Cambria, "Times New Roman", serif;
  font-weight: normal;
  font-size: 12pt;
  border: 2pt solid green;
  margin: 0;
  padding: 10px;
  color: grey;
}
```

*Notice that some of these properties are **inherited** by child elements. This is what is meant by "cascading" in the CSS name. Which properties are inherited and which are not?*

3 Add the following style to `lab04-ex08.css` and test.

```
div {
  font-weight: bold;
  color: magenta;
}
p {
  color: green;
}
```

Notice that these new styles will override the style of the parent (i.e., the <body> element).

4 Modify the following style and test.

```
p {
  color: green;
  border: inherit;
}
```

Notice that one of the <p> elements inherit's its parent's border property. Notice that the <p> elements within the elements do not.

5 Add (not modify) the following style after the already-existing body selector and test.

```
body {
  color: brown;
}
```

This illustrates the role of location in determining precedence: when rules have the same specificity, then the latest are given more weight, so the color specification in the second body style overrides the earlier one.

6 Use your browser's Inspect feature (in Chrome, right-click then select Inspect) to examine the style definitions for the `<body>` element.

As illustrated in Figure 4.2, the browser shows that a style definition has been overridden via a line through the overridden style.

7 Add the following style and test.

```
.last {
  font-size: 14pt;
  color: blue;
}
```

Notice that class selectors take precedence over element selectors. That is, the class selector (.last) specification overrides the element selector (p) specification.

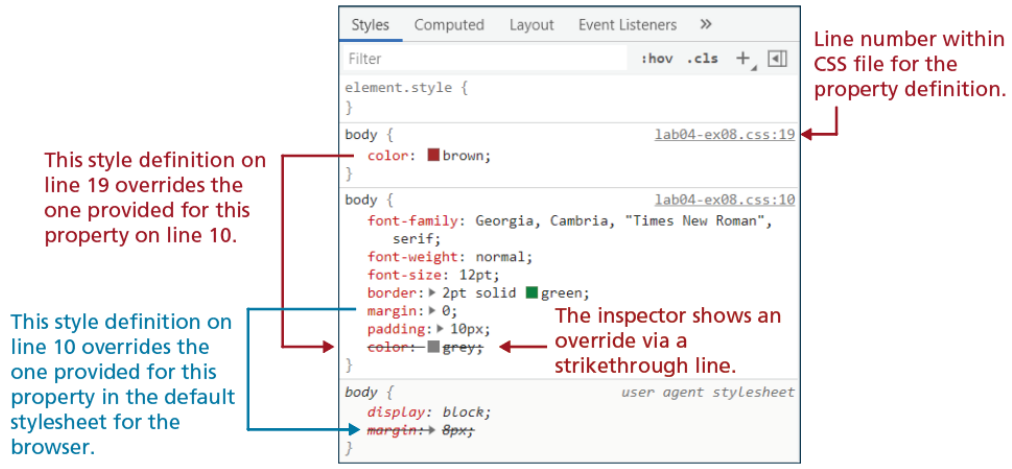

Figure 4.2 – Examining style overrides in browser's style inspector.

8 Add the following style and test.

```
#verylast {
  color: orange;
  font-size: 18pt;
}
```

Notice that id selectors take precedence over class selectors.

9 Add the following style at the end of your style document and test.

```
div p {
  color: red;
  font-size: 8pt;
}
```

*Notice that this style specification did **not** override the class or id specification (i.e., those from steps 6 and 7).*

10 Switch to the markup file and edit the following line and test.

```
<p class="last" id="verylast" style="color: yellow">This one has the last class and the verylast id attribute</p>
```

This example illustrates how an inline style attribute has the highest specificity, and can be used to override any style specification within a stylesheet.

11 Use the browser inspector to examine the overrides from steps 8, 9, and 10.

Exercise 4.9 — Block vs Inline

1. Open and examine `lab04-ex09.html`, then view in browser. It already has some styling to help visualize the difference between inline and block elements.

   The elements with the brown rectangle are inline elements, while those with the sea green dashed rectangle are block elements.

2. Modify the style in `lab04-ex09.css` as follows then test:

```
h1, h2, p, div, ul, li {
   border: 2px dashed lightseagreen;
   margin-top: 10px;
   padding: 5px;
   width: 75%;
}
```

   You will learn more about width, padding, and margins later in this lab, but this changes the width of the specified elements to be 75% of its parent's container, adds 10px margin space to the top of each block element, and adds 5px of inner space (inside the border) to each element.

3. Modify the following style and test:

```
span, a, em {
   border: 2px solid sienna;
   margin-top: 10px;
   padding: 5px;
   width: 75%;
}
```

   This has no effect on these inline elements. Margin top and bottom, padding, and width and height are ignored for inline elements.

4. Modify the following style and test:

```
img {
   border: 2px solid sienna;
   display: block;
}
```

   By changing the display property to block, the image now resides on a line to itself.

5. Modify the following style and test:

```
span, a, em {
   display: block;
   border: 2px solid sienna;
   margin-top: 10px;
   padding: 5px;
   width: 75%;
}
```

   Each of these elements now resides on its own line.

6 Modify the following style (some code is omitted) and test:

```
span, a, em {
   display: inline-block;
   ...
}
```

The inline-block value makes the element inline (rather than on a line by itself) but uses the margin, padding, and width and height property values.

7 Modify the following style (some code is omitted) and test:

```
span, a, em {
   ...
   width: 80px;
}
```

This shrinks the width to a set value. Notice that the content of an element still needs to be displayed, which means it might 'spill" outside of its width (as in the elements inside the <h1>.

8 Modify the following style and test:

```
img {
   border: 2px solid sienna;
   display: inline;
}
```

This restores the image to its default inline display.

9 Add the following style and test.

```
li {
   display: inline-block;
   list-style-type: none;
   width: 20%;
   height: 50px;
   margin: 0;
}
```

This demonstrates one of the most common uses of the inline-block display mode: to implement a horizontal list of elements.

In CSS, all HTML elements exist within an element box. It is absolutely essential that you familiarize yourself with the terminology and relationship of the CSS properties within the element box, which are covered in the Section 4.6 of the textbook.

Exercise 4.10 — Introducing the Box Model

1. Open and examine `lab04-ex10.html`, then view in browser.
2. Add the following style **to the very top** of `lab04-ex10.css`, then test.

```
header, footer, nav, main, article, section, figure,
figcaption, h1, h2, h3, ul, li, body, div, p, img
{
   margin: 0;
   padding: 0;
   font-size: 100%;
   vertical-align: baseline;
   border: 0;
}
```

This is an example of a CSS reset, that is, a way to remove any browser defaults. This way, any styling that is present is due to explicit styling rather than to browser defaults, which can potentially vary from browser to browser. Notice that it removes default margins for <p>, , <figure>, and heading elements.

3. Add the following style and test.

```
figure {
   margin: 2em;
}
```

4. Modify this style as follows then test.

```
figure {
   margin: 2em;
   background-color: #EEEEEE;
}
```

5. Modify this style as follows then test.

```
figure {
   margin: 2em;
   background-color: #EEEEEE;
   padding: 1.5em;
}
```

6. Modify this style as follows (some code omitted) then test.

```
figure {
   ...
   border: solid 1px #999999;
}
```

Figure 4.3 illustrates the relationship between the key box model properties.

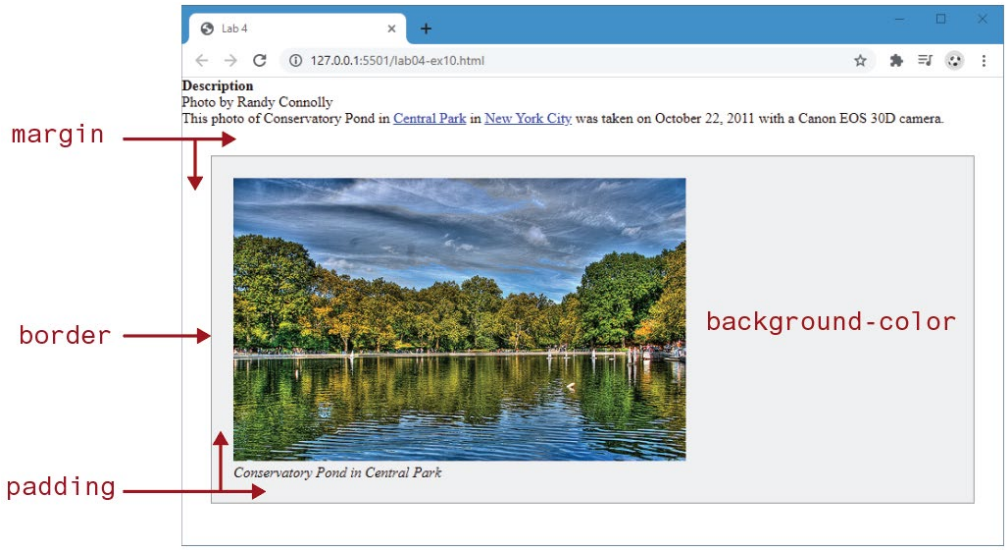

Figure 4.3 – Introducing the box model properties

Exercise 4.11 — Backgrounds and Shadows

1 Examine `lab04-ex11.html` then view in browser.

Notice that it has a `<section>` element; you will be adding background styles to it.

2 Modify the style in `lab04-ex11.css` as follows then test.

```
#container1 {
   width: 100%;
   height: 700px;
   background-color: powderblue;
}
```

3 Modify this same style as follows then test.

```
#container1 {
   width: 100%;
   height: 700px;
   background-color: powderblue;
   background-image: url(images/checkers.png);
}
```

Notice that the background color is hidden by the image. As well, the background image is repeated so as to fill the container.

4 Modify this same style as follows (some code omitted) then test.

```
#container1 {
   ...
   background-image: url(images/checkers.png);
   background-repeat: no-repeat;
}
```

5 Modify this same style as follows (some code omitted) then test.

```
#container1 {
   ...
   background-image: url(images/boating.jpg);
   background-repeat: no-repeat;
}
```

6 Modify as follows and test.

```
#container1 {
   ...
   background-size: cover;
}
```

This tells the browser to try to cover the entire width of the container with this image.

7 Change this property as follows and test.

```
background-size: contain;
```

This tells the browser to try to display the entirety of the image within the container.

8 Change this property as follows and test.

```
background-size: 500px 384px;
```

9 Add the following property to this same style and test.

```
background-position: 50px 100px;
```

This specifies where on the element the background image will be placed.

10 Modify this style and test.

```
#container1 {
   width: 50%;
   height: 500px;
   ...
   box-shadow: 2px 2px gray;
}
```

This adds a solid shadow offset by 2px to the section container.

11 Change this property as follows and test.

```
box-shadow: 2px 2px 10px 5px gray;
```

This adds a 10px blue radius and a 5px spread radius.

Exercise 4.12 — Borders, Margins, and Padding

1 Examine `lab04-ex12.html` then view in browser.

2 Add the following style to `lab04-ex12.css` then test.

```
p {
   border: solid 1pt red;
   margin: 0;
   padding: 0;
}
```

3 Modify this style as follows and test.

```
p {
   border: solid 1pt red;
   margin: 50px;
   padding: 0;
}
```

This sets the top, right, bottom, and left margins to 50px.

4 Modify this style as follows and test.

```
p {
   border: solid 1pt red;
   margin: 30px;
   padding: 30px;
}
```

5 Modify the following property and test.

```
margin: 50px 10px;
```

This sets the top and bottom margins to 50px and the left and right margins to 10px.

6 Modify the following property and test.

```
padding: 20px 10px 50px 5px;
```

When there are four values, they are in the order: top right bottom left.

7 Modify the following property and test.

```
margin: 50px 10px 0px 10px;
```

Notice that nothing changed.

8 Modify the property and test.

```
margin: 50px 10px 50px 10px;
```

Once again, nothing has changed. How much margin space is between each <p> element? That is, is the margin space 100px (50px top + 50px bottom)? In fact, it's just 50px. This is due to collapsing margins. When the vertical margins of two elements touch, only the largest margin value of the elements will be displayed, while the smaller margin value will be collapsed to zero (see Figure 4.4).

9 Add the following style and test.

```
div {
   border: dotted 1pt green;
   padding: 0;
   margin: 40px 20px;
}
```

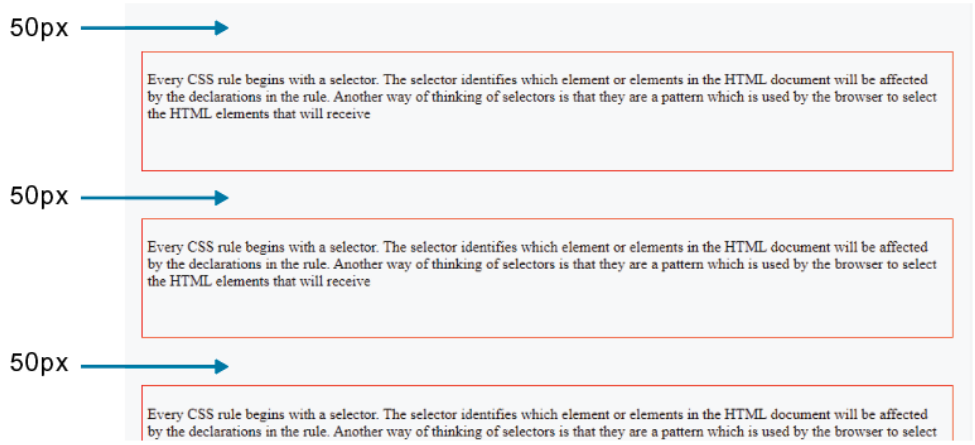

Figure 4.4 – Collapsing vertical margins

Exercise 4.13 — Box Sizing

1 Examine `lab04-ex13.html` then view in browser.

2 Modify the style in `lab04-ex11.css` as follows then test.

```
div {
   margin: 50px;
   border: solid 5px black;
   background-color: lightcyan;
   width: 400px;
   height: 100px;
}
```

3 Add the following and test.

```
div {
   ...
   padding: 5px;
}
```

4 Add the following styles and test.

```
#first {
   box-sizing: content-box;
}
#second {
   box-sizing: border-box;
}
```

By default, block elements have the content-box value for box-sizing. This means the true width of an element = borders + padding + width. By setting the box-sizing to border-box, the true width of the element = width (as shown in Figure 4.5). This is often very helpful.

5 To visualize the difference, switch to the markup and add the following. Test.

```
<div id="first">
   Every CSS rule begins with a selector...
</div>
<img src="images/box-sizing.png" />
<div id="second">
   Every CSS rule begins with a selector...
</div>
<img src="images/box-sizing2.png" />
```

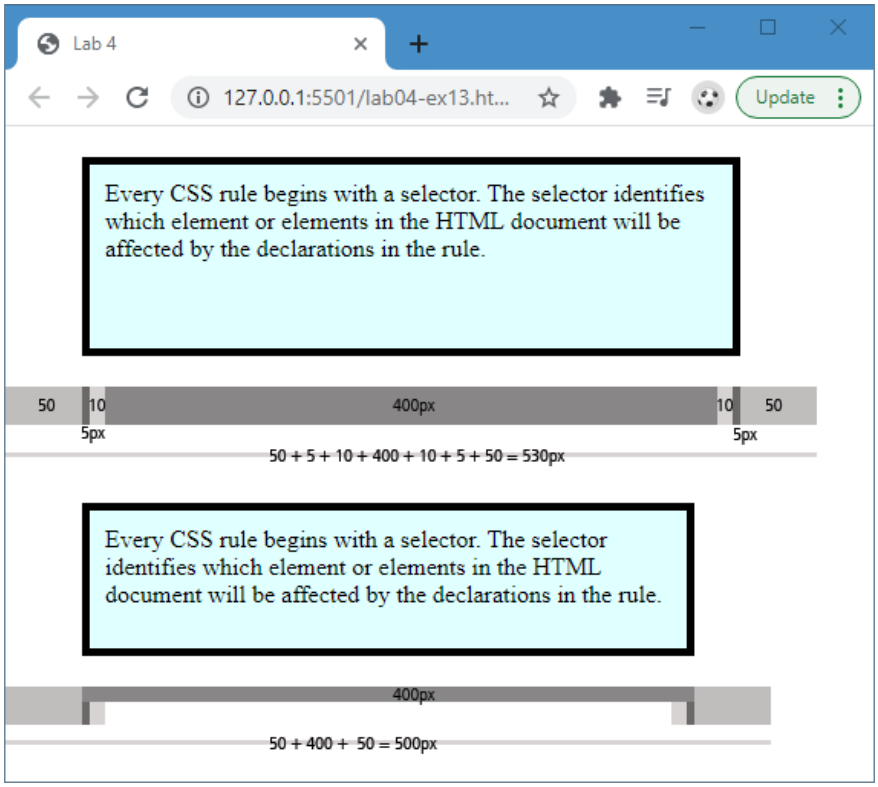

Figure 4.5 – content-box versus border-box

Exercise 4.14 — Overflow

1. Examine `lab04-ex14.html` then view in browser.
2. Modify the style in `lab04-ex11.css` as follows then test.

```
div {
    background-color: silver;
}
```

3. Add the following and test.

```
div {
    background-color: silver;
    width: 200px;
    height: 100px;
}
```

This example illustrates a limitation/feature of the height property: that the entire content of an element will be displayed regardless of the element's height. The extra content is said to ***overflow*** *out of the specified height. You can control how this overflow works in CSS.*

4. Add the following and test.

```
div {
    ...
    overflow: hidden;
}
```

5. Modify the following and test.

```
div {
    ...
    overflow: scroll;
}
```

6. Modify the following and test.

```
div {
    ...
    overflow: auto;
}
```

Test Your Knowledge #2

You have been provided markup in `lab04-test02.html`. You cannot modify the markup, so this will require working with selectors. This time, you will also be defining the styles within `styles/lab04-test02.css` so that it looks similar to that shown in Figure 4.5.

This exercise requires you to use margins, paddings, and borders. It's not important to make it exact, but do try to get it close.

1. Set the `background-image` property of the `<header>` image along with the `background-size` property.
2. For the three colored boxes, you will need to make use of the `nth-child()` pseudo class selector to give each box its own background color. In Chapter 7, you will learn about how to implement horizontal layouts with block-level elements. For now, you can get the `<div>` elements to sit horizontally next to each other by setting their `display` property to `inline-block`.

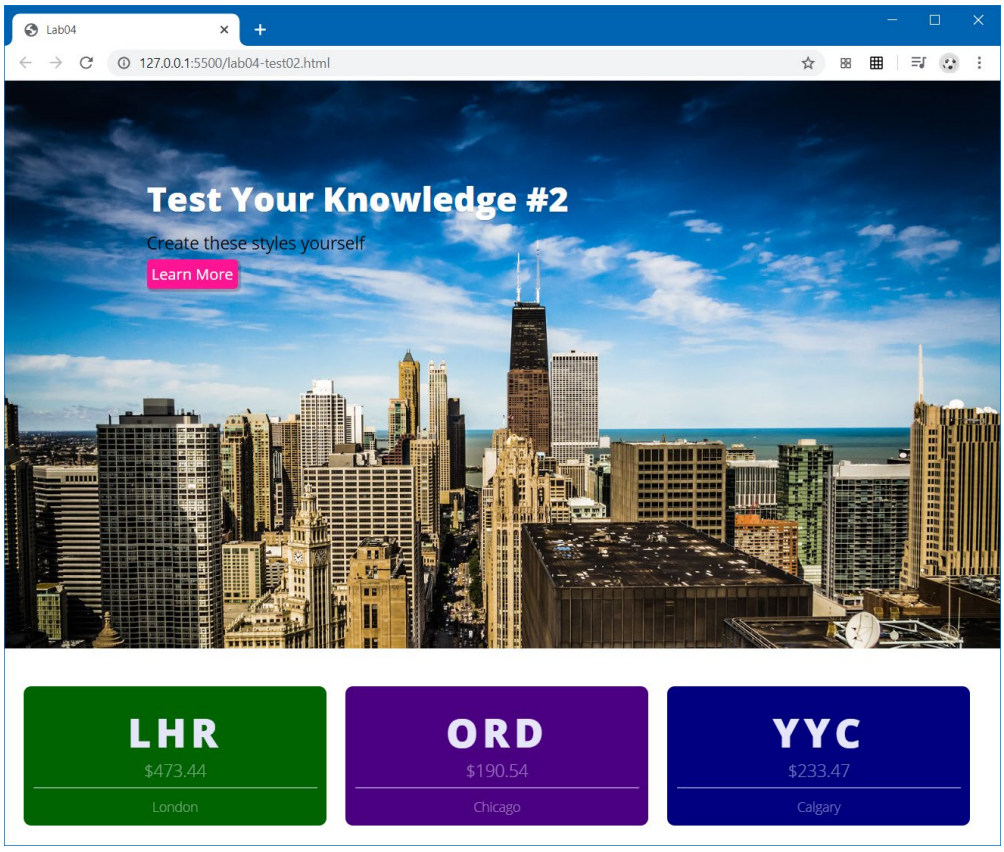

Figure 4.6 – Completed Test Your Knowledge #2

Exercise 4.15 — Font Families

1. Examine `lab04-ex15.html` then view in browser.
2. Modify the style in `lab04-ex15.css` as follows then test.

```
#first {
    font-family: Arial, "Helvetica Neue", Helvetica, sans-serif;
}
```

3. Use your browser to visit `https://fonts.google.com/`.
4. Search for the **Lobster** font, then select it. This should display the details for this font.
5. Use the **Select this style** selector, which should display a panel with the `<link>` element needed to use this font.

   Google changes the Google Fonts web interface relatively frequently, so it is possible that these instructions are no longer accurate.

6. Select the `<link>` element from the Google Font display and paste it into the `<head>` of `lab04-ex15.html` ***before*** the `<link>` for `lab04-ex15.css`.
7. Modify the style in `lab04-ex15.css` as follows then test.

```
#second {
    font-family: 'Lobster', cursive;
}
```

8. In the Google Font page, search and select the **Merriweather** font.

   Notice that this style has multiple weights (Light, Regular, Bold, Black)

9. Select Light, Regular, Bold, and Black styles. Notice the provided `<link>` element includes the weight numbers (e.g., 300,400,700,900). It should also still include the Lobster family.
10. Copy the provided `<link>` and replace the one pasted in step 6 with this one.
11. Modify the style in `lab04-ex15.css` as follows then test.

```
#third {
    font-family: 'Merriweather', serif;
    font-weight: 300;
}
#fourth {
    font-family: 'Merriweather', serif;
    font-weight: 900;
}
```

Exercise 4.16 — Character Styling

1 Examine `lab04-ex16.html` then view in browser.

2 Modify the styles in `lab04-ex16.css` as follows and test.

```
#parent1 {
   font-size: 100%;
}
#parent2 {
   font-size: 75%;
}
.first {
   font-size: 1em;
}
.second {
   font-size: 1.5em;
}
```

As you can see, em units are relative to the parent's % font-size setting.

3 Add the following style and test.

```
.third {
   font-size: 1rem;
}
```

When using a rem unit, the font-size stays the same regardless of the parent's % size.

Exercise 4.17 —Paragraph Styling

1 Examine `lab04-ex17.html` then view in browser.

2 Modify the styles in `lab04-ex17.css` as follows and test.

```
#p1 {
  line-height: 2;
}
#p2 {
  text-transform: uppercase;
}
#p3 {
  letter-spacing: 3px;
}
#p4 {
  text-align: center;
}
#p5 {
  text-align: right;
}
```

TEST YOUR KNOWLEDGE #3

You have been provided markup in `lab04-test03.html`. You cannot modify the markup, so this will require working with selectors. The markup includes links to two Google Fonts.

This exercise focuses on text and character styles. The final result should look similar to that shown in Figure 4.7.

1. For the `<h1>` elements, use a `font-weight` of 500, `letter-spacing` of 5px, a `color` of `deeppink` and center align the text.
2. To complete the `<span>` circles, set the `border-radius` to 50%, the `font-size` to 12px, and transform the content to upper case via the `text-transform` property.
3. Each colored circle `<span>` has its own class name (e.g., color1, color2). Define these classes and set the `background-color` to be the same as the span label.
4. Define the `details` class. It should have a `font-size` of 16px, be silver and italic.
5. Define each of the elements (h2, h3, h4) and classes (`bodytext` and `asidetext`). The style definitions for font, size, weight, and line height are contained in the markup itself.

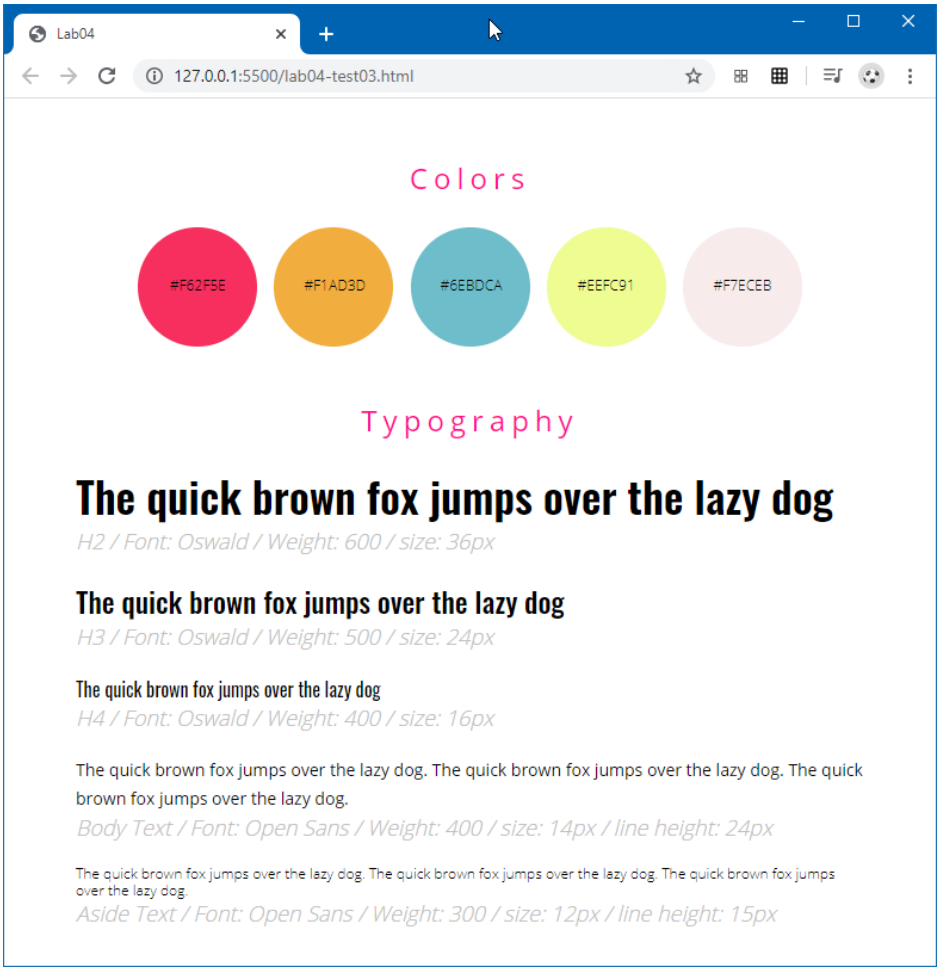

Figure 4.7 – Completed Test Your Knowledge #3