Introduction to CSS3 and HTML5: Lesson 4

Alan Simpson

Chapter 1

Borders, Corners, and Shadows

So far in this course, we've focused on different ways you can style the background of the pages in your site. In today's lesson, I want to start focusing on the *foreground*, which is going to be any words and pictures you put between the <body>...</body> tags. These elements will automatically appear in front of the background color and background picture.

To give your site some professional polish, you may want to consider using a wrapper div to contain the content, and then you might want to pick out some nice borders, rounded corners, and perhaps a drop shadow behind. You'll learn to do all those things in today's lesson starting in Chapter 2.

Chapter 2

Creating a Wrapper Div

In this chapter, I want to talk a little about programmer comments and wrapper divs. Much of this is review from the *Creating Web Pages* course, so I'll try to keep it brief and to the point. Let's start with the programmer comments.

Programmer Comments

People who write code often write notes to themselves or to other developers who are working on the same project to explain the purpose of various parts of the code. We refer to these notes as programmer comments or just comments for short. They don't affect how the code operates. They don't appear in the browser window for the public to see. They're just plain-English notes in the code.

You can't just write these notes at random in your code, though. You have to identify the comments so the browser can identify them. In HTML code, you start a comment with the <!-- characters, and end the comment with --> characters. You have to type them exactly like that, no spaces between them, and you can't leave any of them out. Here's an example:

```
<!-- This is a sample HTML comment -->
```

You can put any text at all inside the comment, and it can be any length. You can put the HTML comments anywhere between the <!DOCTYPE html> and </html> tags, except in an internal style sheet. An internal style sheet, like an external style sheet, contains no HTML code. A style sheet contains CSS code. CSS is a separate language from HTML. In CSS, a comment starts with /* and ends with */. Here's an example:

```
/* This is a sample CSS comment */
```

You can put a CSS comment anywhere in a style sheet, and you can put any amount of text inside the comment. I'll use comments in the code in this course. You can use them as reminders about what the code does, and you can reword them using your own words if you like. After all, comments are notes to yourself in code, so feel free to make them useful for you. We'll practice by adding some to your template page and style sheet shortly.

Okay, now let's start talking about how you can design the foreground of your page.

Defining the Wrapper Div

As you saw in the previous lesson, when you apply a background color or background image to the body element in your code, that color or pattern fills the entire browser window. The content (text and pictures) for people to read and look at goes in front of that background.

You can put all your content (text and pictures) right between the <body>...</body> tags of the page. But most professional developers don't do it that way. The majority of professional developers put all the content inside an additional pair of ... tags, because doing so provides much better control over the formatting and design of the page. In those tags, div is short for *division*, and it refers to anything on the page that might be considered a major division or section.

The div that contains all of the visible content on the page is often referred to as a container div (because it contains all the page content) or a wrapper div (because it wraps around all the content). It doesn't much matter what you call it, so we'll refer to it as the *wrapper* in this course.

The main purpose of the wrapper div is to constrain the width of the content, so it doesn't always stretch and shrink to fit the browser window. We used such a div in our final layout in *Creating Web Pages*. You can add one to this layout now, too. You add the div tags to your page the same way you add any other tags. You open the page in your editor and type them in. Let's add a pair of div tags to HTML5 Template.htm now, since we're using that page as kind of a working template to define the design of the Web pages for our site. Hopefully you know the drill, but here are the steps:

1. Open your *Intro HTML5* folder

2. Right-click or CTRL-Click HTML5 template.htm, choose **Open With** and then the name of the editor you're using (for example, *Notepad* or *TextEdit*).

3. Move the cursor down between the <body> and </body> tags, and type .

4. Press ENTER.

5. Type the words Page content goes here, and then press ENTER.

6. Type <!-- End wrapper --> .

The code below shows how the page should look. Your page should contain all that code. I highlighted only the new code you just typed to make it stand out. It won't be highlighted in your editor. I've also indented to make it easier to see matching start tags and end tags, but indenting is optional and you don't have to do that if you don't want to.

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <link href="stylesheet.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <div id="wrapper">
      Page content goes here
  </div><!-- End wrapper -->
</body>
</html>
```

It's important that you put the tags between <body> and </body> as above. Notice I added a comment that reads *End wrapper* after the closing tag. That comment has no effect on the code, and the code will work the same without it because it's just a comment. However, it's a good idea to mark your closing tags with a comment like that because there may be a lot more divs in the page by the time you're finished. Using a comment to track which goes with which right from the get-go is a good way to ward off confusing and frustrating errors caused later by forgetting the closing tags, or putting in too many of them, or putting them in the wrong places.

When you feel confident that you typed the code correctly, close and save the page exactly as you've done in the past. When you double-click the page's icon to view it in the Web browser, your background image will show just as it did before. The new text should show up too. Your background may be different from the one shown below, depending on how you left things in Lesson 3. That's okay, you don't have to use the leather background. But unless your background image is very dark, you should be able to see the new text inside the div.

Content added to the page

You can't see the new wrapper div we added because, by default, every div has the following characteristics:

- Its background is transparent.
- It's as tall as it needs to be to contain its content.
- It has no visible border around its edges.
- It's as wide as its containing element.

Default style is just what you get when you don't provide your own styling with CSS. And CSS will give us total control over the styling of the div. For starters, you may want to give it a background color so that any text you put into it is easier to read. You may want to give it a specific width so that you can better control how it looks on different screens. The main thing to keep in mind when you're choosing a width is to make sure people can read any text and see pictures without scrolling horizontally. Some designers will use a width of 780 pixels or so to accommodate lower resolution 800 x 600 monitors. Those are becoming rare, though. So some designers will go with a width of about 900 pixels. That's because even the smallest handheld devices like the iPhone, iPod Touch, Droid Razr, and other smartphones have a width of 960 pixels. Virtually all tablets, desktop PCs, and laptops have a width of at least 1,024 pixels. So 900 pixels wide is probably a safe bet, too.

You'll probably want to center the wrapper horizontally so things look balanced. You do that by setting the left and right margins to auto, which is short for "automatically calculate and make them the same as each other." When the left and right margins of an element are equal, that element is centered horizontally on the page.

To apply all this styling to the wrapper, we need a style rule for the wrapper. We can put that style rule in the external style sheet so it applies to all the pages in the site. That way, you get a nice, consistent look-and-feel across all the pages in the site.

Let's go ahead and add some programmer comments and a style rule for our wrapper div to our style sheet now. Follow these steps:

1. In your *Intro HTML5* folder, right-click or CTRL + click **stylesheet.css**, choose **Open With** and your editor as usual.

2. At the very top of the style sheet (above the body style rule), type */* stylesheet.css */* which is just a comment for you to look at when the file first opens to remind you of which file you're looking at.

3. Press ENTER, and then type */* Style rule for the body element */* as a comment just above the body{} style rule.

4. Move the cursor down past the body style rule so it's under the body style rule.

5. Type the comment */* Style rule for the wrapper div */* and then press ENTER.

6. Type *#wrapper{* making sure to use the open curly brace (not a parenthesis or square bracket) as that last character.

7. Press ENTER, and type *width: 900px;* and then press ENTER again.

8. Type *margin:0 auto;* and then press ENTER again.

9. Type *background-color:white;* and press ENTER again.

10. Type } (the closing curly brace) to end the wrapper style rule, and then press ENTER again.

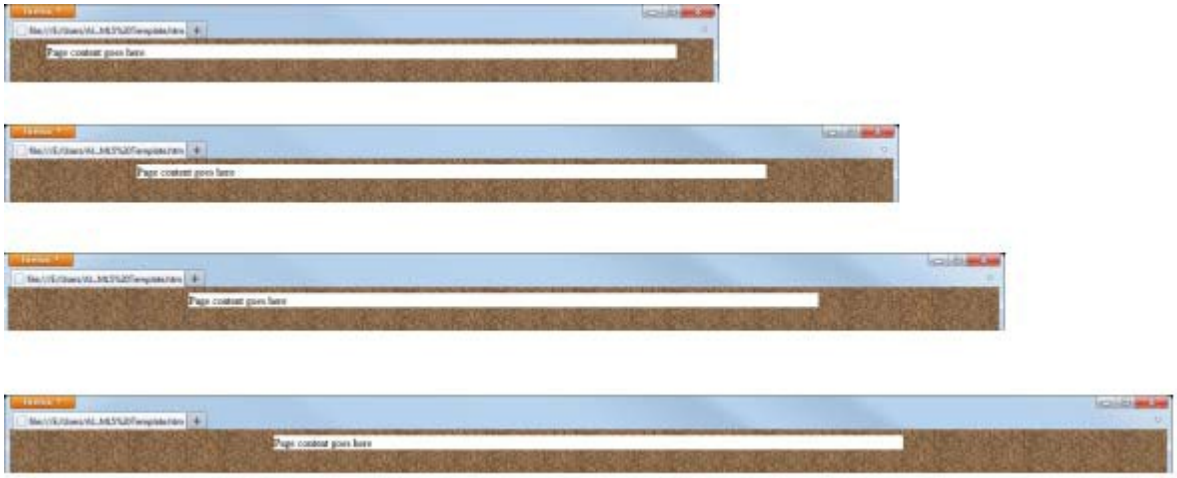
If you typed everything correctly, your entire style sheet should now look something like this (though feel free to use your own background color or image for the body element).

```
/* stylesheet.css */
/* Style rule for the body element */
body{
    background-color:yellow;
    background-image:url(pix/bkgleather2.jpg);
}
/* Style rule for the wrapper div */
#wrapper{
   width:900px;
   margin:0 auto;
   background-color:white;
}
```

You'll want to check your work in the browser, too, to make sure you typed all the code correctly. Remember how to do that? When you make a change to a style sheet, you have to do two things in the

correct order: 1) Save the changes you made to the style sheet, then 2) Open (and possibly Reload or Refresh) a linked page (HTML5 Template.htm) in a Web browser.

If you did everything correctly, you should see the wrapper div with its white background and a few words of content on the page in the browser. If your browser window is more that 900 pixels wide, you'll see the whole div centered horizontally. I can't say exactly what you'll see because I don't know what browser you're using or how wide its window is. But here are some possible examples using different window widths in the Firefox browser.



Wrapper div with white background, different browser widths

If your wrapper div looks much different, there may be a typographical error in your code somewhere. But assuming all is good, let's move on to Chapter 3 and talk about some cool ways to improve the appearance of that wrapper div, including a border line, rounded corners, and a drop shadow.

Chapter 3

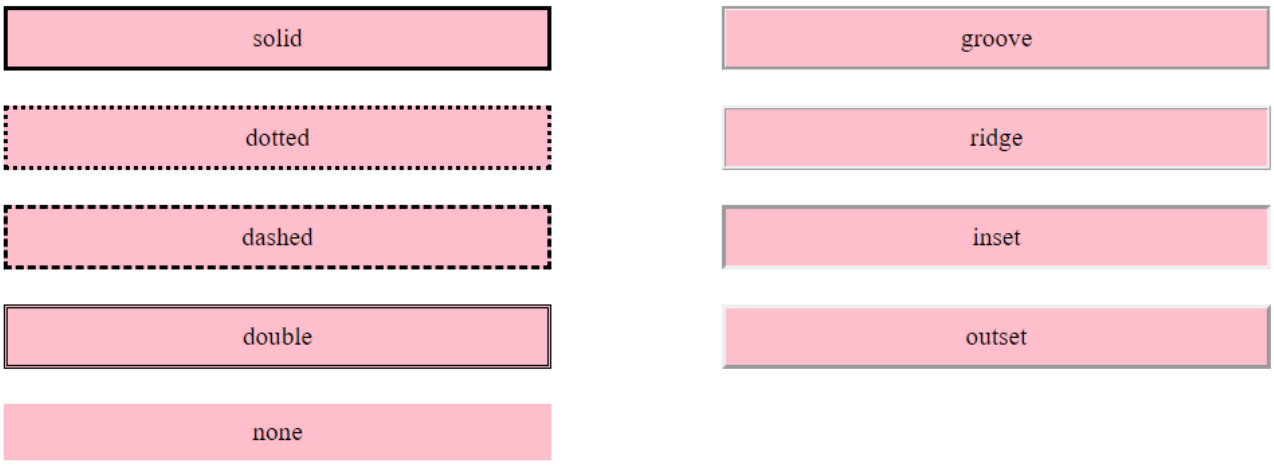
Designing Borders

In this chapter, we're going to talk about adding borders to elements. A *border* is a line that surrounds the element. A border is a decorative option, so it's never really required. Right now, our wrapper div has no border around it. That's because in HTML, no element gets a border automatically. If you want to put a border line around an element, you have to use a CSS border property to do so.

The quickest, easiest, and most common way to add a border is with the CSS border: property using the following syntax:

```
border: border-style [width] [color]
```

As always, the italics are just placeholders for values you provide. Square brackets indicate optional values that you can omit (you never actually type the square brackets in your own code). Replace *border-style* with the style of border you want, using one of the words shown in red below.



CSS border styles

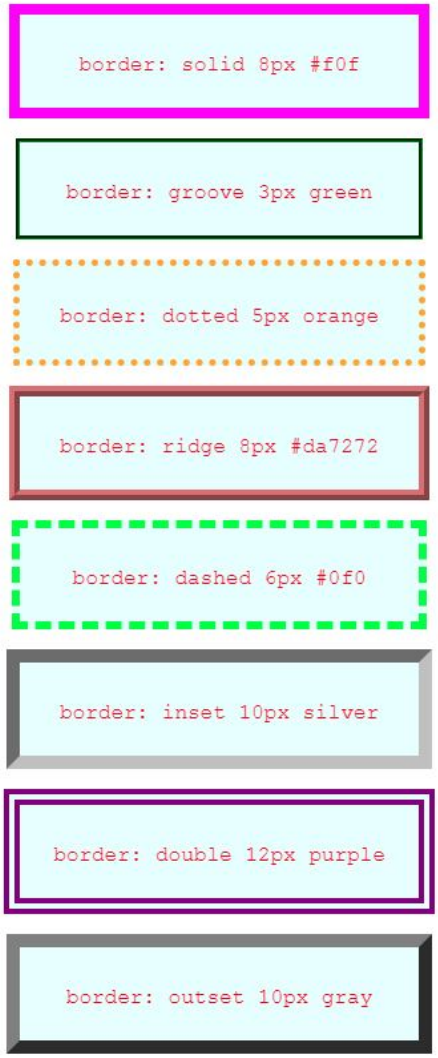
Note

CSS3 adds *dot-dash*, *dot-dot-dash*, and *wave* as border styles. However, there's currently no browser support for those new border styles, so I can't show you examples of their appearance.

For the *width* value, specify a number followed by *px* for pixels. The thinnest possible border is 1 px. As always, make sure you don't put a space between the number and the px. You can also provide *thin*, *medium*, or *thick* as values. But many designers avoid these values because they can give inconsistent results across browsers.

For the *color* value, provide any valid CSS color name or hex code.

Here are some examples of designing different borders by providing all three values.



CSS border styles

Note

In the example above, I've typed the CSS code inside each box (in red) to make the code for each border type easy to see. In real life, you'll put the CSS code in CSS style rules or inline styles, as always. You'll get some hands-on practice with this in the assignment for this lesson.

The order of values in the border descriptor really doesn't matter. For example, all of the descriptors below produce a solid red border that's 8 pixels wide.

```
border: solid 8px red
border: solid red 8px
border: red solid 8px
border: red 8px solid
border: 8px red solid
border: 8px solid red
```

The CSS border property is a quick and easy way to make all four borders around an element the same style, color, and width. But the folks who designed CSS realized that you may encounter situations where you don't want all four borders to be identical. So they've added a whole slew of border properties to style just one characteristic, or just one side, independently.

Border Properties	
property: *value(s)*	**What it sets**
border-bottom: *style width color*	Style, width, and color of the bottom border
border-bottom-color: *color*	Color of the bottom border
border-bottom-style: *style*	Style of the bottom border
border-bottom-width: *width*	Width of the bottom border
border-color: *color*	Color of all four borders
border-left: *style width color*	Style, width, and color of the left border
border-left-color: *color*	Color of the left border
border-left-style: *style*	Style of the left border
border-left-width: *width*	Width of the left border
border-right: *style width color*	Style, width, and color of the right border
border-right-color: *color*	Color of the right border
border-right-style: *style*	Style of the right border
border-right-width: *width*	Width of the right border
border-style: *style*	Style of all four borders
border-top: *style width color*	Style, width, and color of the top border
border-top-color: *color*	Color of the top border
border-top-style: *style*	Style of the top border
border-top-width: *width*	Width of the top border
border-width: *width*	Width of all four borders

At first glance, having so many border properties might seem intimidating. But each just lets you focus on one border (top, right side, bottom, or left side), or just the style, color, or width of a border. And that's just to give you more freedom and flexibility to design your borders the way you want. For instance, you could use four of those properties to style the width, style, and color of each border independently, like this:

```
border-top: solid 8px #ebddcb;
border-right: solid 6px #d2b48c;
border-bottom: solid 8px #b22222;
```

```
border-left: solid 6px #d2b4bc;
```

An element with those borders would have a unique appearance, like this.

Different CSS border properties

Keep in mind that not all browsers render borders exactly the same. So if you try to get too fancy with your borders, you may face some obstacles and frustrations trying to get them to look exactly the same across all versions and brands of browsers. (This is probably why you don't really see many fancy borders in websites.) Also, keep in mind that when styles conflict, the one that's closest to the tag wins. This can matter when you're trying to get real creative designing borders. For example, imagine these two descriptors applied to a single element:

```
border: solid 1px blue;
border-bottom-width: 20px;
```

The resulting borders look like this.

Bottom border thicker than others

The border looks exactly the way the code says that it should because the first descriptor says, "Make all four borders solid, 1-pixel wide, and blue"; and then the second one says, "Make the bottom border 20 pixels wide." There's a little bit of a conflict there because first we say, "Make all four borders 1px," and then we say, "Make the bottom border 20px." But since the second one is closer to the tags, it takes precedence over the first one, so the bottom border (only) is 20 pixels thick.

Note

As you move down the style sheet, you get closer to the tags, because the link tag that brings the style rules into the page is above the <body>...</body> tags, where all the visible tags being styled will be located in the page.

Now consider the same two descriptors, still applied to a single element but in reverse order, like this:

```
border-bottom-width: 20px;
border: solid 1px blue;
```

The end result is different, as below:

All borders same thickness

That's the correct way for the browser to interpret the code (even though it might not have been your intent when writing the code). Here's why it's correct. First, the browser is told to make the bottom border 20px thick. Nothing wrong with that. But the next line tells it to make *all four borders* solid*, 1 pixel wide,* and blue. Well, the bottom border can't be 20 pixels thick and also one pixel thick. So that's a conflict. The second descriptor wins the conflict because it's closer to the tags. So all four borders end up being 1-pixel wide, as the *border: solid 1px blue* says they should be.

After you've decided on a border style, you can add some more professional polish by rounding its corners and adding a drop shadow. Come on over to Chapter 4 to learn how.

Chapter 4

Rounded Corners and Drop Shadows

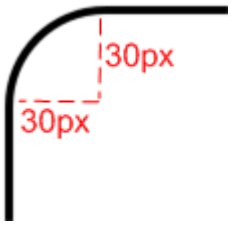
In this chapter, you'll learn to add some extra pizzazz to your design by using rounded corners and drop shadows. Both of these features use CSS3, which only works in the newest browsers. So make sure you use a current version of Safari, Firefox, Internet Explorer, Chrome, or other browser to see the full effect.

Rounding Corners

By default, the borders around an element meet at right angles, forming perfectly square corners. CSS3 provides some properties for rounding those corners. The simplest way to round all four corners of an element is to use this syntax in your style rule:

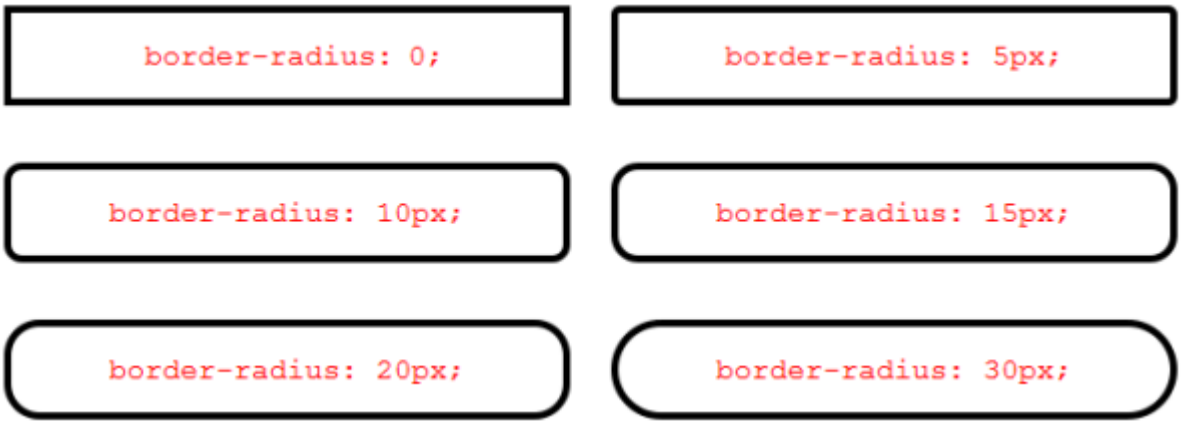
```
border-radius: length
```

In this example, *length* is a number, followed by a unit of measure (usually px for pixels). That number determines the height and width of the corner. For example, providing 30px as a number means each corner's rounding will be 30px tall and 30px wide, like this:



A 30px border radius

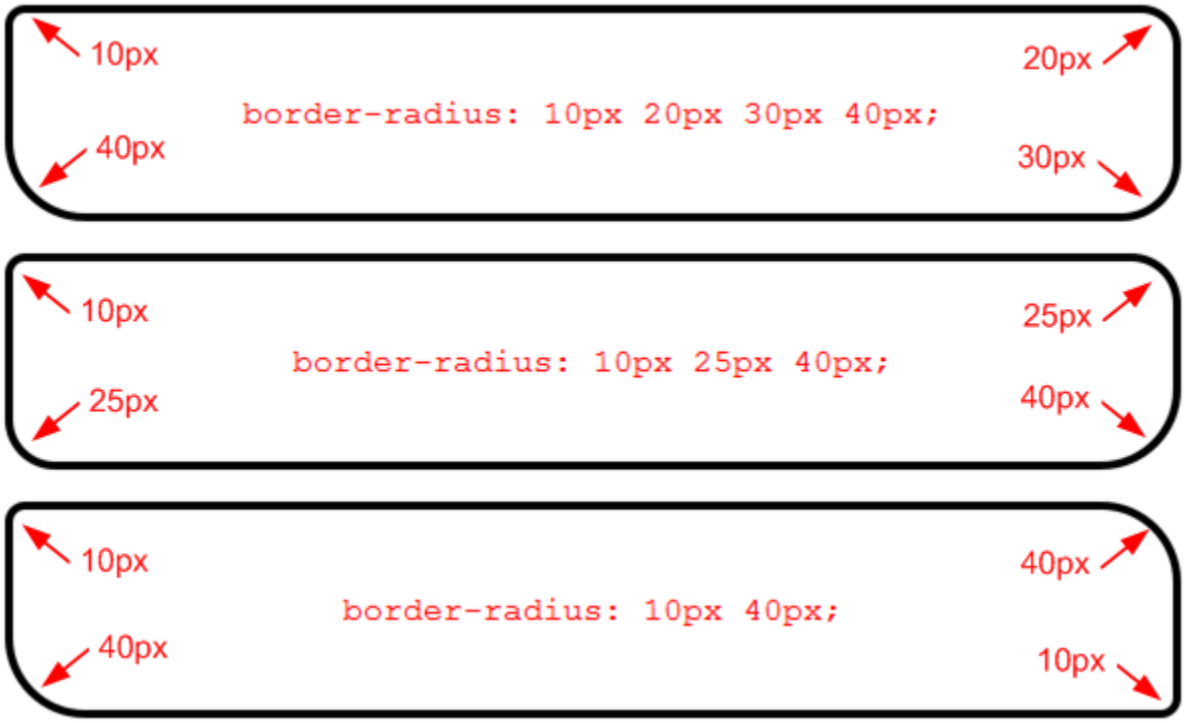
A value of zero means no rounding at all. The higher the number you go from there, the larger the rounding effect, as in these examples:



Some border-radius examples

The simple syntax you just learned will probably be sufficient for all of your corner-rounding needs. After all, it's unlikely that you'd ever need to have the corners of an element rounded differently. But of course, the folks at the W3C who created CSS know that people like to be creative and sometimes go beyond the basics. So CSS offers a few alternatives to the simple syntax above. For example, you can provide two, three, or four values (separated by spaces) to the border-radius property to style the corners independently. If you provide four values, they apply clockwise from the top-left corner. If you provide three values, they apply to the top-left corner, then the top-right and bottom-left corners, and then bottom right. If you provide two values, the first value applies to the top left and bottom right corners, and the second value applies to the top right and bottom left corners, as illustrated below.

Show Text Equivalent

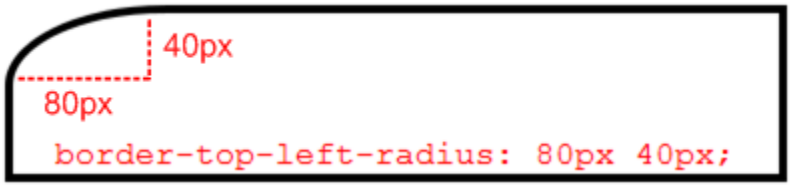


Some border-radius examples with multiple values

Your corners don't have to be perfectly round either. You can use more of a sloping curve by specifying a horizontal radius and a vertical radius for each corner. And you can do so for each corner independently using these CSS properties and syntaxes.

Sloping Corners Properties	
border-top-left-radius: *horizontal vertical*	Radii of top left corner
border-top-right-radius: *horizontal vertical*	Radii of top right corner
border-bottom-right-radius: *horizontal vertical*	Radii of bottom right corner
border-bottom-left-radius: *horizontal vertical*	Radii of bottom left corner

When you provide two numbers for the radius, the first number applies to the horizontal radius, and the second number applies to the vertical radius. For example, *border-top-left-radius: 80px 40px;* applies only to the top-left corner of the element, making the rounding 80 pixels wide but only 40 pixels tall, as below.



Sloping corner

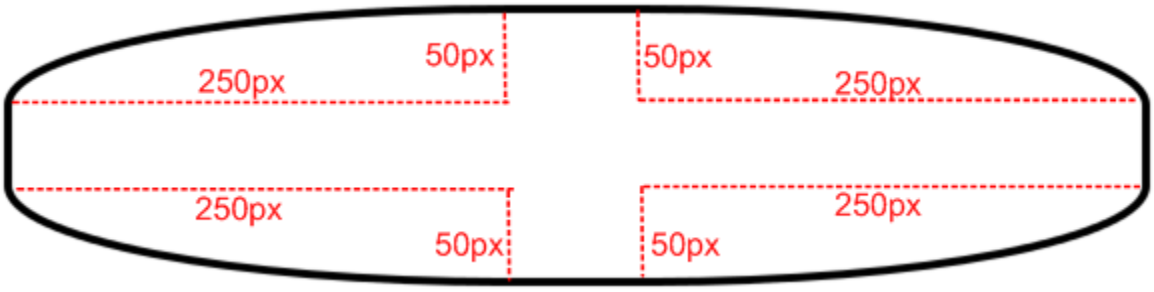
If you want all four sloping corners to be sloping rather than round, you don't have to specify two values for each corner one at a time. Instead, you can use border-radius with two values separated by a slash, like this:

```
border-radius: width/height
```

Where *width* is the width of each radius and *height* is the height of each radius, as in the example below.

```
border-radius: 250px/50px;
```

In the browser, that translates to making the radius of each corner 250 pixels wide and 50 pixels tall.



Four sloping corners

You may never need that much flexibility in designing your corners. So I'm not saying you must, should, or ever would use the more advanced features. But it's good to know they're available, just in case a situation ever arises where you want that kind of styling.

Using Drop Shadows

A drop shadow is an imaginary shadow behind an element that makes it look like it's raised above the page. CSS3 offers a box-shading property that allows you to design a box drop shadow. The syntax for using box-shading is:

```
box-shadow:horizontalOffset verticalOffset [blur] [spread] color [inset]
```

As always, the italics are placeholders for values you must provide. The square brackets indicate optional values that you can omit. Whether you provide a value or not, you should never type the square brackets into your code. Here's what you provide for the different values.

Replace *horizontalOffset* with a number and unit of measure (like px) indicating the width and horizontal location of the shadow. A negative number puts it on the left side; a positive number puts it in the right side.

Replace *verticalOffset* with a number and unit of measure (like px) indicating the width and vertical location of the shadow. A negative number puts it on the top; a positive number puts it in the bottom.

The *blur* value is optional—you can omit it for a value of zero. No blur means a sharp edge to the shadow. You can define a blur as a number followed by a unit of measure. The larger the blur value, the blurrier the edge.

The optional *spread* value is zero if you omit it. If you provide it, the shadow is spread out beyond the offset width by the amount you specify, even on the sides of the element that normally wouldn't have a shadow. You can use it with zero offset values to make the shadow appear all the way around the element, as you'll see in a later example.

Tip: To design rounded corners and drop shadows interactively, see https://alansimpson.me/html_css/tools/shadowmaker.html

Replace *color* with any valid color name or hex code. You can use any color you like. Though for a realistic shadow effect you may want to stick with a black or a shade of gray such as one of the options below.

Show Text Equivalent

Black
Gray
Silver
#111
#222
#333
#444
#555
#666
#777
#888
#999
#aaa
#bbb
#ccc
#ddd
#eee

Gray shades

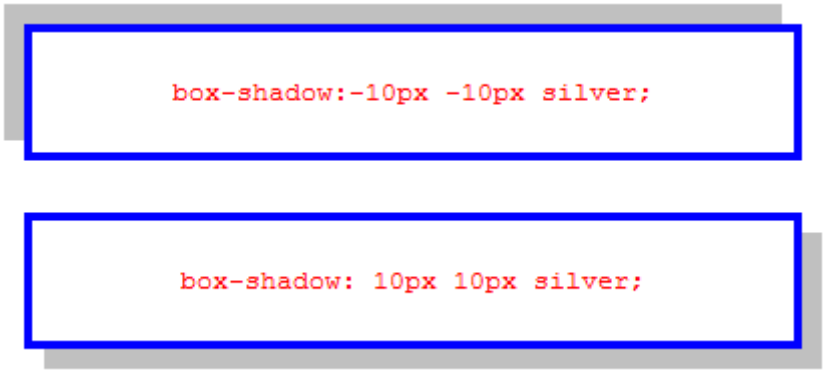
Tip

Darker shades work well against a dark background, while lighter shades generally look better against lighter backgrounds.

The option *inset* value is just the word *inset* typed literally into the code. If you omit it, the shadow appears outside the border. If you include it, the shadow will appear inside. You'll see examples in a moment.

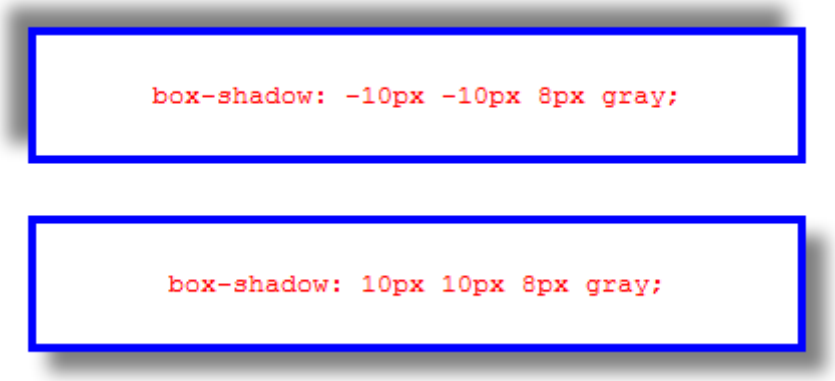
The image below shows a couple of examples using shadows that are 10 pixels wide, negative or positive numbers, and a color. I've omitted the optional blur, spread, and inset values to keep the examples as

simple as possible. I've made the border blue in each example to help it stand out from the shadow, which is outside the border. I'm showing the code in red inside the box so it's easy to see in the image. As always, the CSS code will go in the style rule, not in the element. You'll get some hands-on experience with that soon.



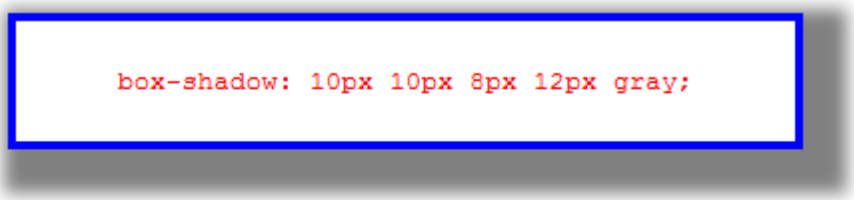
Simple shading

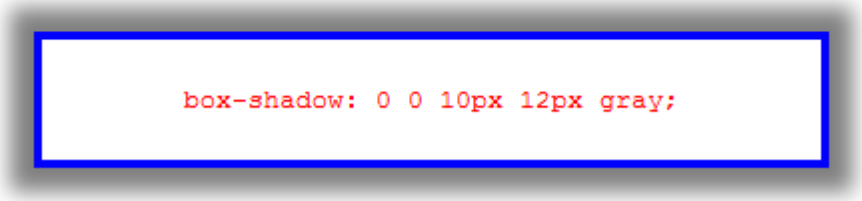
The effect makes the box look raised above the page. But it's not totally realistic because shadows rarely have such clean edges. That's where the optional blur value comes in. Assigning a blur value that's slightly less than, or equal to, the width of the shadow provides a more realistic look. Below I've added an 8-pixel blur to each shadow. I've also used gray, which is a slightly darker color. Though that's optional, and you can still use any color you like.



Blur value (8px) adds realism

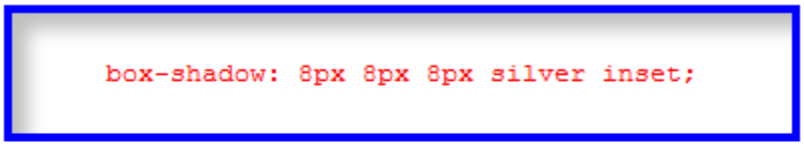
The blur adds some realism, but still provides the kind of shadow you'd get if you were projecting light onto the item from a single source only. In real life, shadows often spread out a little more because light is coming from more than one angle. That's where the options spread value comes in. It allows you to stretch the shadow out in all directions for added realism. For example, in the top example below, the shadow is offset 10 pixels down and to the right. But the 12-pixel spread makes a little bit of the shadow show above and to the left of the border as well. In the second example, the vertical and horizontal offset are both zero, so the shadow is directly behind the box with no offset. The spread makes the shadow spread out from all the borders.





Spread value (12px) extends shadow

And finally, if you want the shadow inside the border rather than outside, just add the word *inset* as the last value, as in the examples below.



Inset shadows

In the assignment for this lesson, you'll get some hands-on practice in applying borders, rounded corners, and a drop shadow to our wrapper div. And there you can try out different values until you find something that you really like. For now, that's enough new information to keep you busy for a while. So let's wander over to Chapter 5 and summarize what you've learned today.

Chapter 5

Conclusion

You can apply the borders, corners, and drop shadows you learned about in today's lesson to any block element in HTML. A block element is any element that starts on a new line in the page and includes things like divs, headings, paragraphs, and lists. The key points are:

You can use a wrapper div to constrain the width of the content portion of the layout and also to give it a separate background color.

- Any div can have a border, defined by the CSS border: property.
- To round the corners of an element, use the CSS border-radius property.
- To place a drop shadow behind an element, use the box-shadow CSS property.

Don't forget to complete the assignment for this lesson, where you'll get some hands-on practice in applying those features to your wrapper div.

Once you've decided on a nice design for your background and main content division, you may want to start designing the text in terms of fonts, colors, and other features. You'll learn to do that in Lesson 5. See you there!

Supplementary Material

Please see this page for updated Supplementary Material links.

FAQs

Q: My code is correct, but I don't see rounded corners or the drop shadow. What gives?

A: CSS3 is very new and supported only by the newer versions of the major Web browsers. Older browsers that don't recognize the code will just ignore it. The page will still look okay and be readable to anybody who views the page through an older browser. They just won't see the rounded corners or drop shadow.

Q: When looking at other peoples' code, I sometimes see stuff like this:

```
border-radius:20px;
-moz-border-radius:20px;
-webkit-border-radius:20px;
```

A: The -moz- and -webkit- parts are *vendor prefixes*, used by the browser manufacturers to experiment with new CSS properties before the W3C has made them official. The -moz- is for Mozilla browsers like Firefox and Firefox Mobile, while -webkit- is for Webkit browsers like Safari and Google Chrome. It's no longer necessary that you use those prefixes, even though they still appear in some older books and tutorials.

Q: Why does the wrapper style rule have a # in front of the selector, but the body style rule doesn't?

A: When you write a style rule for an element type that's built into HTML, you just use the name of the element. In other words, you just use whatever word appears between the < and > characters of the opening tag. For example, to style the <body> element, you just use the word *body* as the selector.

There's no such thing as a wrapper element or a <wrapper> tag. So we can't use a <wrapper> tag in our page, nor a *wrapper* selector in a style rule. We have to use a more generic div tag and give it an id like this:

```
<div id="wrapper">
```

The # in front of the selector tells the browser that it needs to apply the style rule to whatever tag in the page contains *id="wrapper"*.

Assignment

Let's add a nice border, rounded corners, and a drop shadow to your wrapper div in this assignment. You already added the wrapper div to your page in Chapter 2. And you already created a style rule for it in your stylesheet.css file in that same chapter. To add the new features, you just have to add them to the wrapper style rule in your style sheet, where you'll be putting most of your CSS code for this course and your website. Feel free to design your wrapper borders, corners, and drop shadow as you see fit. You just need to add three new descriptors to your existing wrapper style rule as in the example below.

```
border:solid 2px #4c2e16;
border-radius:20px;
box-shadow:8px 8px 8px black;
```

Note that you don't need to create a new style rule. You don't need to remove anything that's currently in your #wrapper style rule. And you don't need to make any changes to the page at all. You just have to open stylesheet.css in your editor and add those three lines, so the #wrapper style rule looks like this:

```
/* Style rule for the wrapper div */
#wrapper{
    width:900px;
    margin:0 auto;
```

```
    background-color:white;
    border:solid 2px #4c2e16;
    border-radius:20px;
    box-shadow:8px 8px 8px black;
}
```

After you make the changes, take a look at the page in the browser and see how things look. Here's how the code above should render in a modern Web browser if you're using the leather background. The fact that the wrapper div is so short right now makes things look a little odd. Text inside might touch the inner edges, and the corners might look too round. But don't worry about that, there's still plenty more we can (and will!) do to spruce things up. Just focus on the general appearance of the border, corners, and shadow for now.

New wrapper div styling

If the colors or other features don't really work for you and your design goals, consider trying some other options until you find a look you like. Remember, for the drop shadow, black or a dark gray will usually work best. Lighter shades of gray often look better against a light background. But use your own taste and judgment to come up with a look that works for you.

You should be able to make changes to the style sheet and try them out on your own by now. But if you need step-by-step instructions to get started, see the steps below.

Lesson 4: Assignment

Extra steps

Here are the steps to add the new styling to your wrapper style rule.

1. Open your *Intro HTML5* folder.
2. Right-click or CTRL+Click **stylesheet.css**, choose **Open With**, and then select the name of your editor (Notepad or TextEdit).
3. Move the cursor down just past the background color, and press ENTER to start a new line (but make sure you're still inside the closing curly brace for the #wrapper style rule).
4. Type (or copy and paste) in these three lines of CSS code:

```
border:solid 2px #4c2e16;
border-radius:20px;
box-shadow:8px 8px 8px black;
```

5. Close the style sheet, and choose **Yes** or **Save** if asked about saving your changes.

To see the effects of your changes . . .

6. Open your linked page (**HTML5 Template.htm**) by double-clicking its icon in your *Intro HTML5* folder.
7. If you don't see the new features right away, click the **Reload** or **Refresh** button in the browser. (In some browsers, you can also press the F5 key to reload or refresh the page.)