

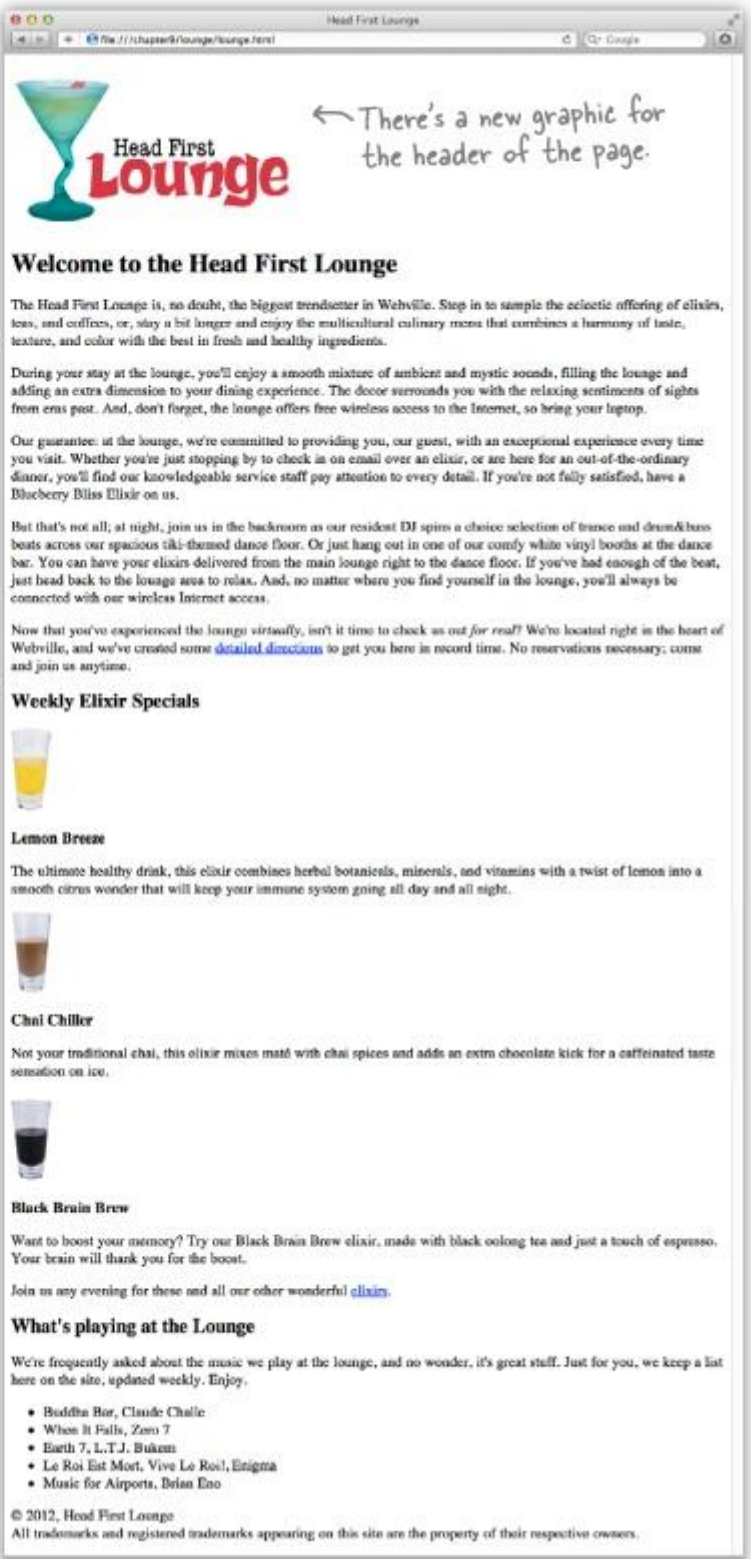
# Chapter 9. The Box Model: Getting Intimate with Elements



**To do advanced web construction, you really need to know your building materials.** In this chapter we're going to take a close look at our building materials: the HTML elements. We're going to put block and inline elements right under the microscope and see what they're made of. You'll see how you can control just about every aspect of how an element is constructed with CSS. But we don't stop there—you'll also see how you can give elements unique identities. And, if that weren't enough, you're going to learn when and why you might want to use multiple stylesheets. So, turn the page and start getting intimate with elements.

## The lounge gets an upgrade

You've come a long way in eight chapters, and so has the Head First Lounge. In fact, over the next two chapters, we're giving it a total upgrade with all new content for the main page and restyling it from scratch. And, just to entice you, we're going to give you a little sneak peek before we even get started. Check this out—on this page, you'll find the new unstyled lounge page with all the new content. On the next page, you'll find the stylized version that we're going to create by the end of the next chapter.



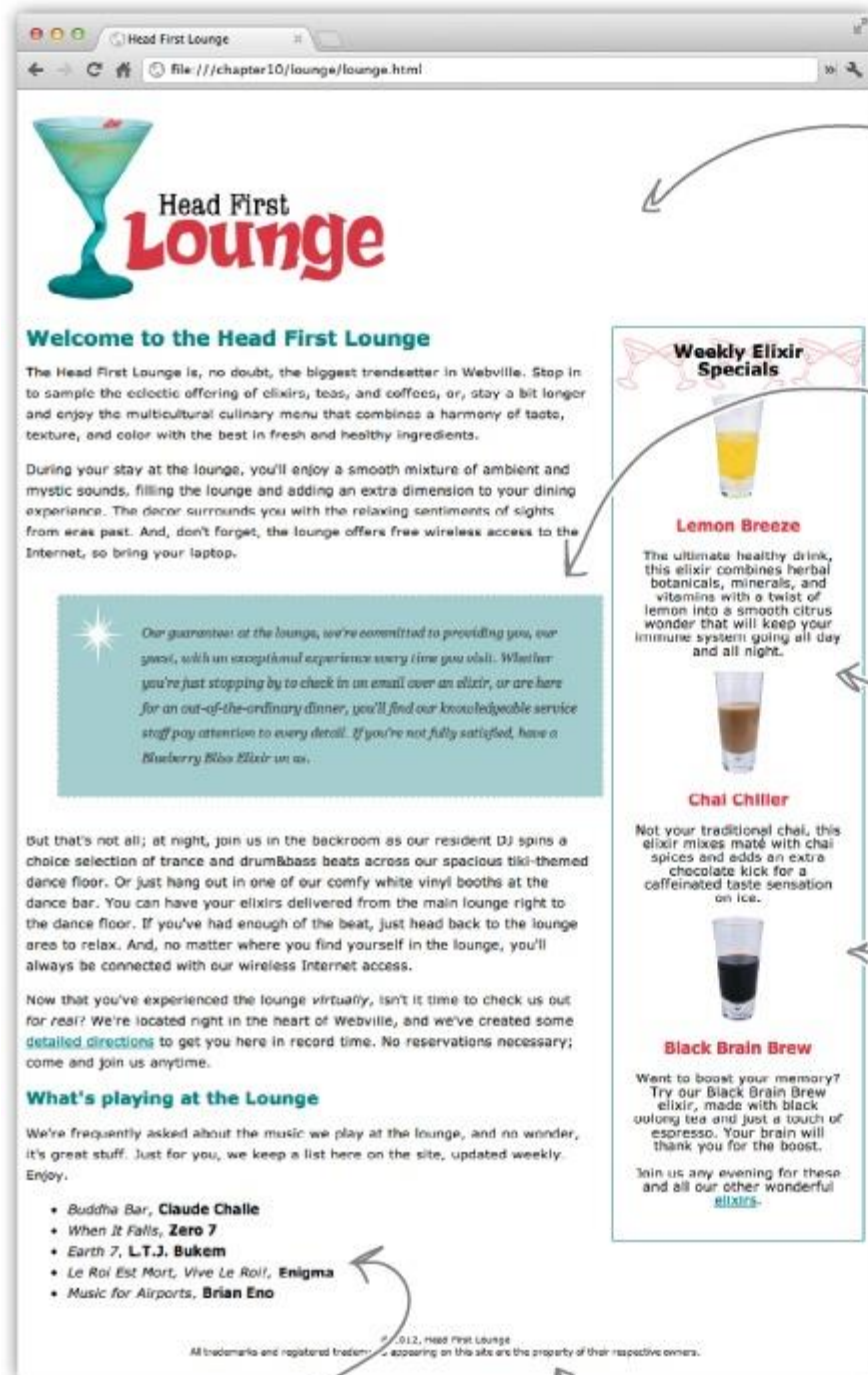
There's a new graphic for the header of the page.

The lounge guys have supplied a lot of new text describing the lounge and what it offers.

They've included a set of elixir specials for the week.

And they even let visitors sample some of the music that is played in the lounge each week, a common request of customers.

Finally, they've got some legalese in the footer of the page with a copyright.



We've got headings that match the site's color theme, an aquamarine. The fonts are also a very readable sans-serif.

This paragraph has been highly stylized, which helps set it off from the text and gives the page an attractive look. It also looks like the font is a serif font, which is different from the main text.

The elixirs have been dramatically restyled into an appetizing display of drinks.

The elixirs have also been moved over to the side. How did that happen?

The music CDs and artists are styled now too.

And the footer is centered and displayed in a small font.

The new and improved, ultra-stylish lounge

Not too shabby. Now the lounge design might be a tad on the, well, “ultra-stylish” side for you, but hey, it *is* a lounge. And we’re sure that you can see this design is starting to look downright sophisticated—just think what the same techniques could do for your pages. Well, after this chapter and the next, designs like this are going to be easily within your reach.

## Setting up the new lounge

Before we start the major construction, let’s get familiar with the new lounge. Here’s what you need to do:

1. Take a look at the “chapter9/lounge” folder and you’ll find the file “lounge.html”, with all new content. Open the file in your editor and have a look around. Everything should look familiar: head, paragraphs, a few images, and a list.
2. You’re going to spend most of this chapter adding style to this HTML, so you need a place for your CSS. You’re going to create all new styles for the lounge in the stylesheet file “lounge.css”, so you’ll find your `<link>` element in the `<head>` of “lounge.html” is still there, but the previous version of “lounge.css” stylesheet is gone.  
`<link type="text/css" rel="stylesheet" href="lounge.css">`  
Remember, this `<link>` element tells the browser to look for an external stylesheet called “lounge.css”.
3. Next, you need to create the new “lounge.css” in the “chapter9/lounge” folder. This file is going to hold all the new CSS for the new lounge.

## Starting with a few simple upgrades

Now you’re all ready to start styling the lounge. Let’s add a few rules to your CSS just to get some basics out of the way—like the font family, size, and some color—that will immediately improve the lounge (and give you a good review from the last chapter). So, open your “lounge.css” file and add the following rules.

```
body {  
  font-size:    small;  
  font-family:  Verdana, Helvetica, Arial, sans-serif;  
}  
  
h1, h2 {  
  color: #007e7e;  
}  
  
h1 {  
  font-size: 150%;  
}  
  
h2 {  
  font-size: 130%;  
}
```

Here's the default font size for the page.

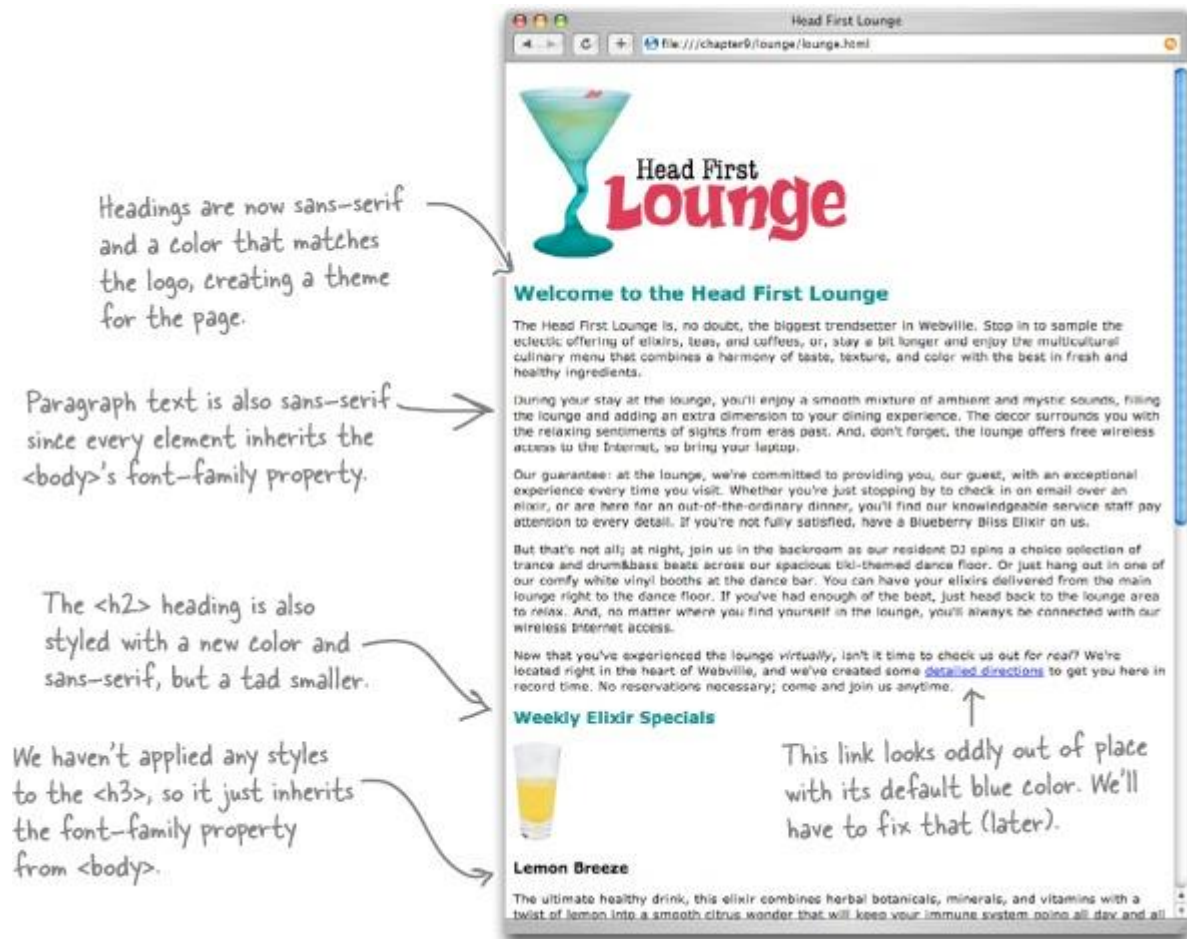
We're going to go with a sans-serif font family for the lounge. We've picked a few font alternatives, and ended the declaration with the generic sans-serif font.

We're going to set the color of the <h1> and <h2> elements to an aquamarine to match the glass in the logo.

Now let's get some reasonable heading sizes for <h1> and <h2>. Since we're setting two different sizes for these, we need separate rules and can't add them to the combined rule for <h1> and <h2>.

## A very quick test drive

Let's do a quick test drive just to see how these styles affect the page. Make sure you've made all the changes; then save and test.



## One more adjustment

We're going to make one more adjustment to the lounge before we move on to start making some bigger changes. This adjustment involves a new property you haven't seen before, but at this point, you've got enough experience under your belt that we're not going to treat you with kid gloves every time a new property comes along. So, just jump in and give it a try.

Here's what we're going to do: we're going to adjust the line height of the text on the entire page so that there's more vertical space between each line. To do that, we add a `line-height` property in the `body` rule:

### NOTE

Increasing the line height of your text can improve readability. It also gives you another way to provide contrast between different parts of your page (you'll see how that works in a bit).



```
body {
  font-size: small;
  font-family: Verdana, Helvetica, Arial, sans-serif;
  line-height: 1.6em;
}
```

Here we're changing the space between each line to 1.6em—in other words, 1.6 times the font size.

## Checking out the new line height

As you might have guessed, the `line-height` property allows you to specify the amount of vertical space between each line of your text. Like other font-related properties, you can specify the line height in pixels, or using an em or percent value that's relative to the font size.

Let's see what the effect of the `line-height` property is on the lounge. Make sure you add the `line-height` property to your CSS file and then save. You should see the line height increase when you refresh.

Using the `line-height` property, we've increased the space between each line of text from the default to 1.6em.

Before

During your stay at the lounge, you'll enjoy a smooth mixture of ambient and mystic sounds, filling the lounge and adding an extra dimension to your dining experience. The decor surrounds you with the relaxing sentiments of sights from eras past. And, don't forget, the lounge offers free wireless access to the Internet, so bring your laptop.

The space between lines is known as "leading" (pronounced "ledding") in the publishing industry.

After



The `line-height` property is inherited, so by setting it in the body, all the elements on the page now have a line height of 1.6em.



## EXERCISE

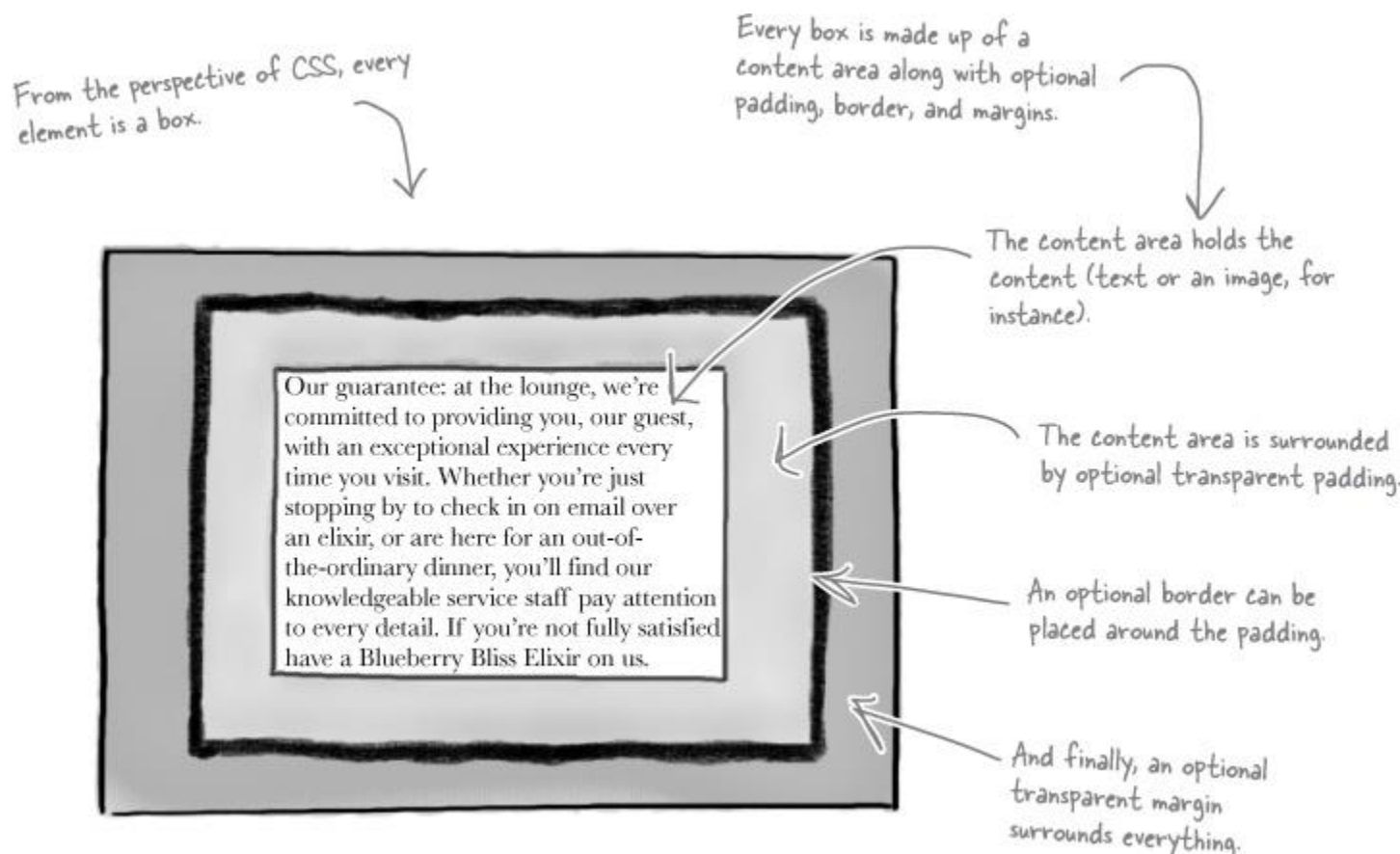
Try a few different values for line-height, like 200%, .5em, and 20px to see the effect. Which looks the best? The worst? Which is most readable? When you're done, make sure you change the line-height back to 1.6em.

## Getting ready for some major renovations

After only a few pages of this chapter, you already have a ton of text style on the new lounge. Congrats!

Now things are going to get really interesting. We're going to move from changing simple properties of elements, like size, color, and decorations, to really tweaking some fundamental aspects of how elements are displayed. This is where you move up to the big leagues.

But to move up to the big leagues, you've got to know *the box model*. What's that? It's how CSS sees elements. CSS treats every single element as if it were represented by a box. Let's see what that means.

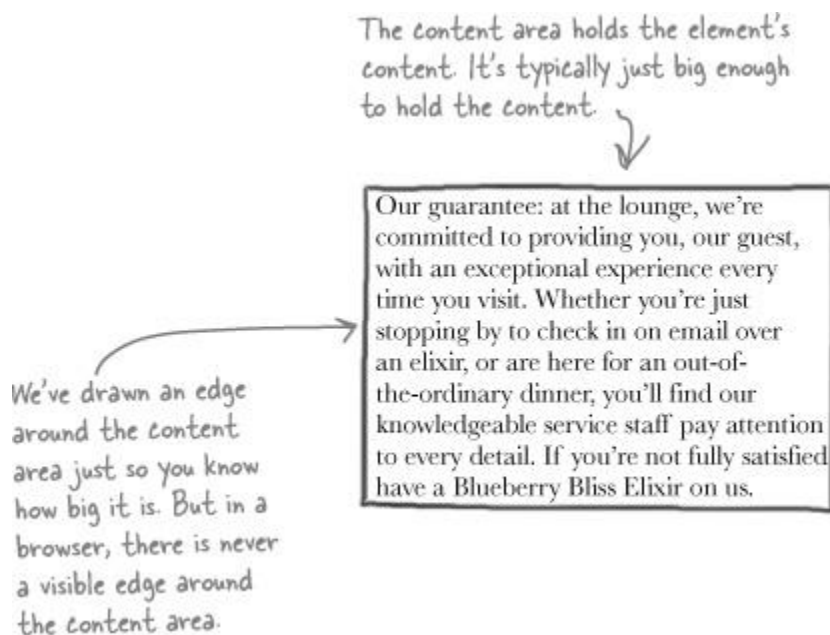


## NOTE

All elements are treated as boxes: paragraphs, headings, block quotes, lists, list items, and so on. Even inline elements like `<em>` and links are treated by CSS as boxes.

## A closer look at the box model

You're going to be able to control every aspect of the box with CSS: the size of the padding around the content, whether or not the element has a border (as well as what kind and how large), and how much margin there is between your element and other elements. Let's check out each part of the box and its role:



### What is the content area?

Every element starts with some content, like text or an image, and this content is placed inside a box that is just big enough to contain it. Notice that the content area has no whitespace between the content and the edge of this box.

The browser adds optional padding around the content area.



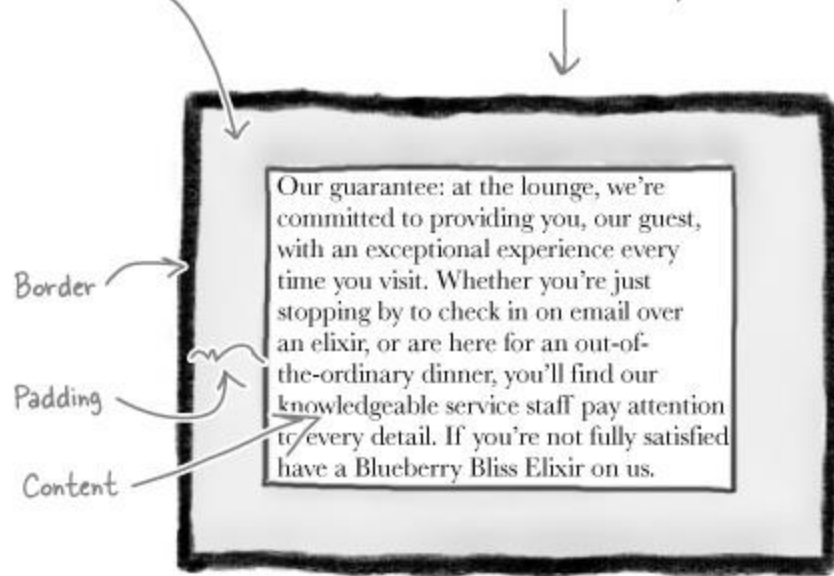
Using CSS, you're going to be able to control the width of the padding around the entire content area, or even control the padding on any one side (top, right, bottom, or left).

## What is the padding?

Any box can have a layer of padding around the content area. Padding is optional, so you don't have to have it, but you can use padding to create visual whitespace between the content and the border of the box. The padding is transparent and has no color or decoration of its own.

Notice that the padding separates the content area from the border.

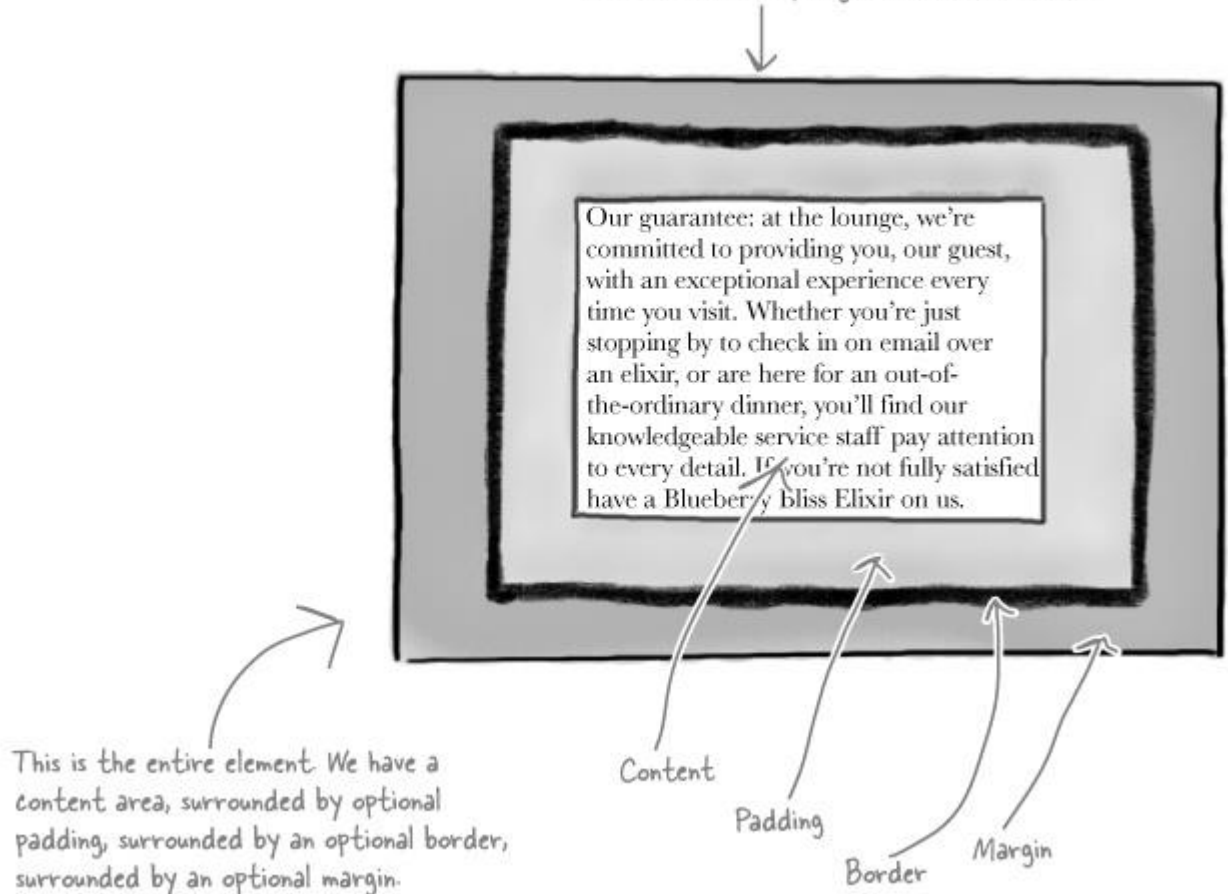
Using CSS, you're going to be able to control the width, color, and style of the border.



## What is the border?

Elements can have an optional border around them. The border surrounds the padding and, because it takes the form of a line around your content, provides visual separation between your content and other elements on the same page. Borders can be various widths, colors, and styles.

Using CSS, you're going to be able to control the width of the entire margin, or of any particular side (top, right, bottom, or left).



## What is the margin?

The margin is also optional and surrounds the border. The margin gives you a way to add space between your element and other elements on the same page. If two boxes are next to each other, the margins act as the space in between them. Like padding, margins are transparent and have no color or decoration of their own.

## What you can do to boxes

The box model may look simple with just the content, some padding, a border, and margins. But when you combine these all together, there are endless ways you can determine the layout of an element with its internal spacing (padding) and the spacing around it (margins). Take a look at just a few examples of how you can vary your elements.

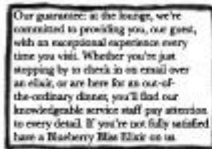
# Boxes



You can style a box to have padding, a border, and a margin.



Or just padding and a border



Or just a border



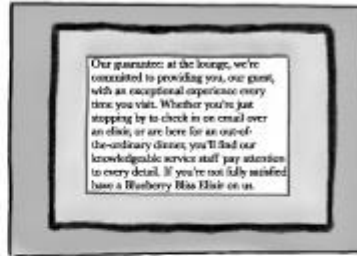
Or a margin with no border and no padding





# Borders

You can have solid borders, thick or thin. →



Or no border at all →



Or choose from eight different styles of borders, like dashed →



Or color your borders →



Or even create rounded corners on your borders →



# Padding



With CSS, you can control padding on any side of the content area. Here we've got a lot of left and right padding.



And here there's a lot of top and bottom padding.



And here the content is offset to the bottom right with padding on the top and left.



# Margins

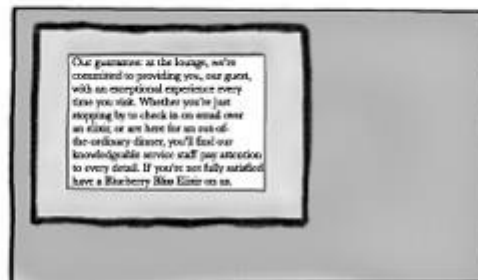
You have the same level of control over the margins. Here there's a lot of top and bottom margin.



And here's a lot of left and right margin.

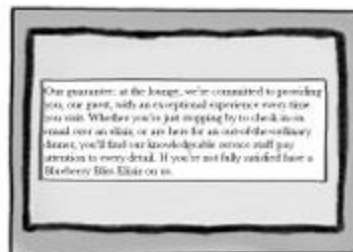


And as with padding, you can specify all sides independently to create margins like this.



# Content Area

You can even control width and height in a variety of ways. Here, the content area has been made wide.



And here the content area is tall but thin.



THERE ARE NO DUMB QUESTIONS

**Q: It seems like knowing this box stuff would be important if I were someone creating the software for a web browser, but how is this going to help me make better web pages?**

**A:** *To go beyond simple web pages that use the browser's default layout, you need to be able to control how elements sit on the page, as well as the relative position of other elements. To do that, you alter various aspects of each element's padding and margins. So to create interesting web page designs, you definitely need to know something about the box model.*

**Q: What's the difference between padding and margin? They seem like the same thing.**

**A:** *The margin provides space between your element and other elements, while padding gives you extra space around your content. If you have a visual border, the padding is on the inside of the border and the margin on the outside. Think of padding as part of the element, while the margin surrounds your element and buffers it from the things around it.*

**Q: I know they are all optional, but do you need to have padding to have a border or a margin?**

**A:** *No, they are all totally optional and don't rely on each other. So you can have a border and no padding, or a margin and no border, and so on.*

**Q: I'm not sure I get how elements get laid out and how margins fit into that.**

**A:** *Hold that thought. While you're going to see a little of how margins interact with other elements in this chapter, we'll get way into this topic in [Chapter 11](#) when we talk about positioning.*

**Q: So other than size, it sounds like I can't really style padding and margins?**

**A:** *That's basically right. Both are used to provide more visual space, and you can't give the padding or margin a direct color or any kind of decoration. But because they are transparent, they will take on the color of any background colors or background images. One difference between padding and margins is that the element's background color (or background image) will extend under the padding, but not the margin. You'll see how this works in a bit.*

**Q: Is the size of the content area determined solely by the size of the content in it?**

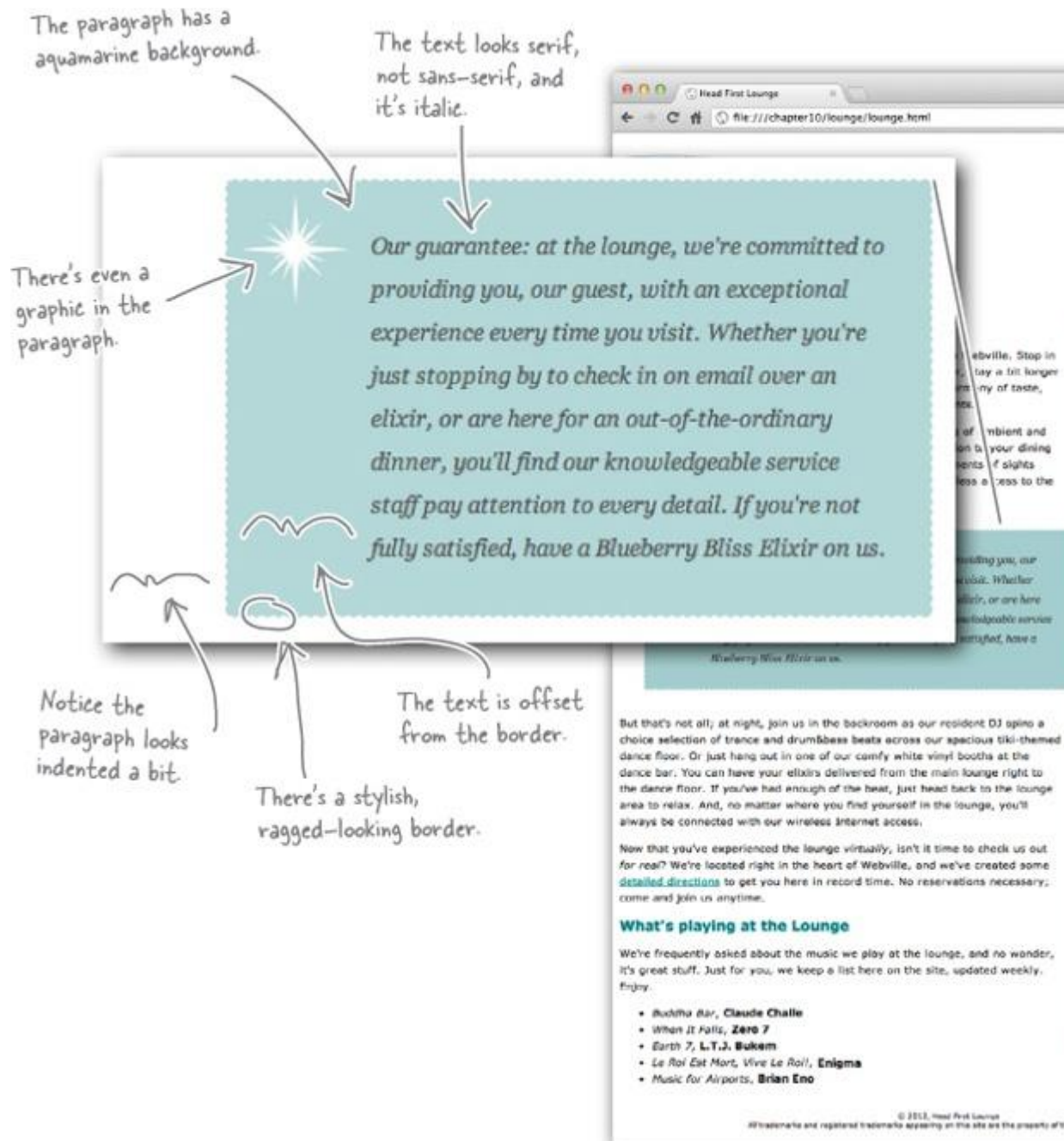
**A:** *Browsers use several rules to determine the width and height of the content area, and we'll be looking at that more in-depth later. The short answer is that while the*

*content is the primary way the size of an element is determined, you can set the width and height yourself if you need control over the size of the element.*



## Meanwhile, back at the lounge...

We do have our work cut out for us on the lounge page, so let's get back to it. Did you notice the blue, stylized paragraph when you looked at the final version of the lounge page in the beginning of the chapter? This paragraph contains text with the lounge's guarantee to their customers, and obviously they want to really highlight their promise. Let's take a closer look at this paragraph, and then we'll build it.



## SHARPEN YOUR PENCIL

See if you can identify the padding, border, and margins of this paragraph. Mark all the padding and margins (left, right, top, and bottom):



don't forget, the lounge offers free wireless access to the Internet, so bring your laptop.



*Our guarantee: at the lounge, we're committed to providing you, our guest, with an exceptional experience every time you visit. Whether you're just stopping by to check in on email over an elixir, or are here for an out-of-the-ordinary dinner, you'll find our knowledgeable service staff pay attention to every detail. If you're not fully satisfied, have a Blueberry Bliss Elixir on us.*

But that's not all; at night, join us in the backroom as our resident DJ spins a choice selection of trance and drum&bass beats across our spacious tiki-themed dance floor. Or just hang

## BRAIN POWER

Before going on to the next page, think about how you might use padding, borders, and margins to transform an ordinary paragraph into the “guarantee paragraph.”

## Creating the guarantee style

Let's get started by making a few small changes to the style of the guarantee paragraph just to get a feel for how the paragraph's box is set up. To do that, you're going to add the paragraph to a class called `guarantee` so that you can create some custom styles for just this paragraph. You're then going to add a border along with a little background color, which will let you see exactly how the paragraph is a box. Then we'll get to work on the rest of the style. Here's what you need to do:

1. Open your “lounge.html” file and locate the paragraph that starts “Our guarantee”. Add a class attribute “guarantee” to the element like this:

Add the class attribute with a value of "guarantee". Remember, a class will allow you to style this paragraph independently of the other paragraphs.

```
<p class="guarantee">
```

Our guarantee: at the lounge, we're committed to providing you, our guest, with an exceptional experience every time you visit. Whether you're just stopping by to check in on email over an elixir, or are here for an out-of-the-ordinary dinner, you'll find our knowledgeable service staff pay attention to every detail. If you're not fully satisfied, have a Blueberry Bliss Elixir on us.

```
</p>
```

2. Save your "lounge.html" file and open the "lounge.css" file. You're going to add a border and background color to the guarantee paragraph. Add the following CSS to the bottom of your stylesheet and then save.

```
.guarantee {  
    border-color:    black;  
    border-width:    1px;  
    border-style:    solid;  
    background-color: #a7cece;  
}
```

The first three properties add a border to any element that is in the guarantee class. So far, that's just this paragraph.

We're making the color of the border black...

...and one pixel thick...

...and solid.

We're also giving the element a background color, which will help you see the difference between padding and margins, and make the guarantee look good.

## A test drive of the paragraph border



Reload the page in your browser, and you'll now see the guarantee paragraph with an aquamarine background and a thin black border around it. Let's examine this a little more closely...

It doesn't look like the paragraph has any padding around the content—there is no space between the text and the border.



the lounge and adding an extra dimension to your dining experience. The decor surrounds you with the relaxing sentiments of sights from eras past. And, don't forget, the lounge offers free wireless access to the Internet, so bring your laptop.

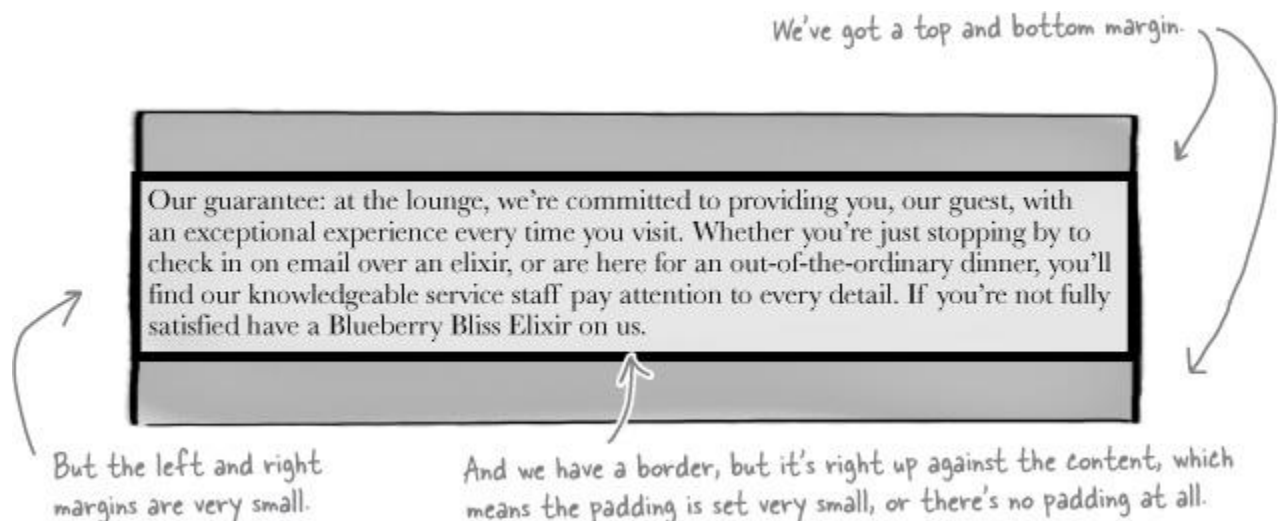
Our guarantee: at the lounge, we're committed to providing you, our guest, with an exceptional experience every time you visit. Whether you're just stopping by to check in on email over an elixir, or are here for an out-of-the-ordinary dinner, you'll find our knowledgeable service staff pay attention to every detail. If you're not fully satisfied, have a Blueberry Bliss Elixir on us.

But that's not all; at night, join us in the backroom as our resident DJ spins a choice selection of trance and drum&bass beats across our spacious tiki-themed dance floor. Or just hang out in one of our comfy white vinyl booths at the dance bar. You can have your elixirs delivered from the main

But there does seem to be a margin on the top and bottom of the paragraph element.

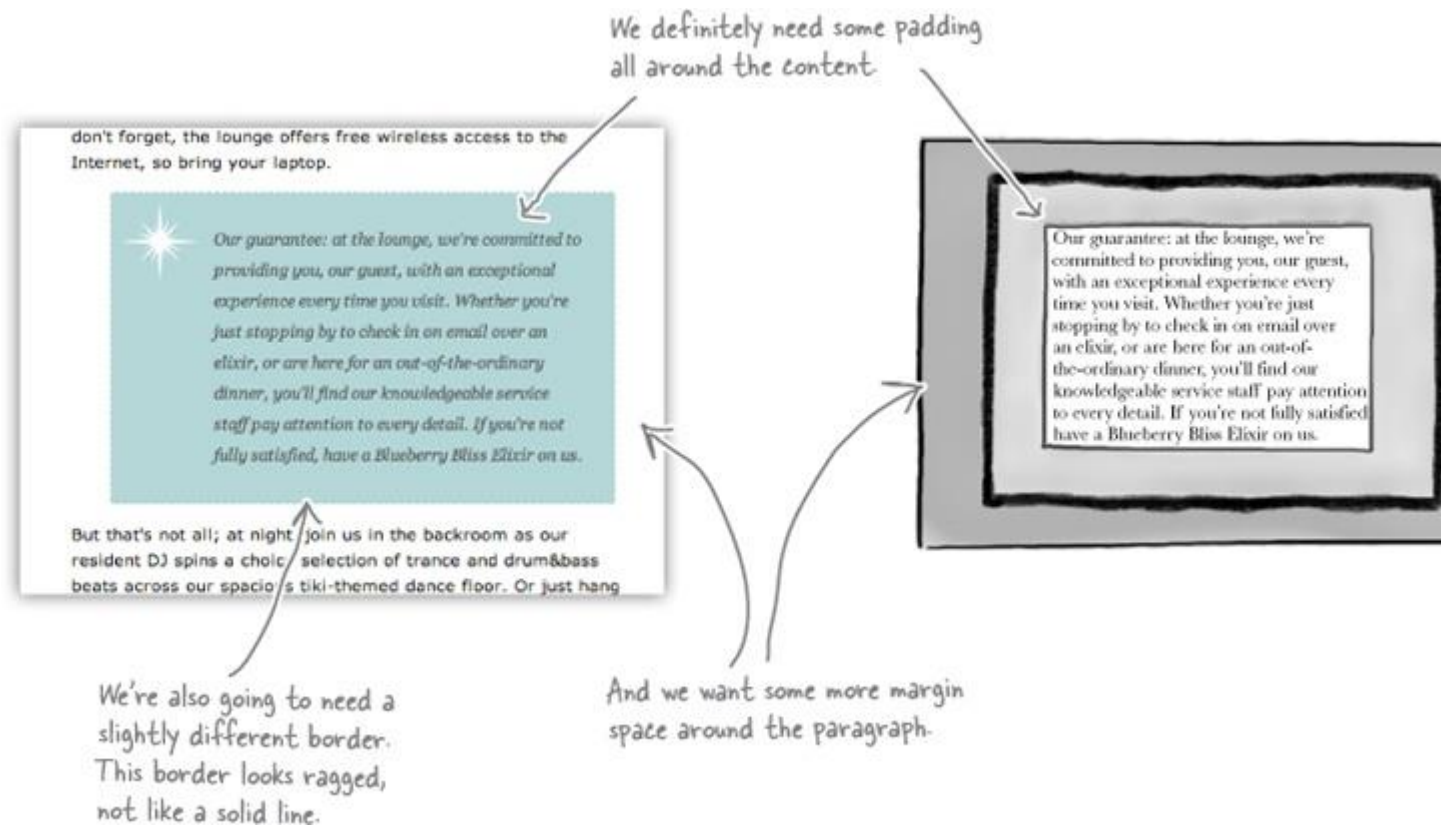
There isn't a noticeable margin between the left and right edges of the paragraph and the browser window edge.

Here's what the paragraph would look like if we drew it as a box model diagram:



# Padding, border, and margins for the guarantee

Now that you've seen how the padding, border, and margins are currently set on the guarantee paragraph, let's think more about how we'd actually like them to look.



## Adding some padding

Let's start with the padding. CSS has a `padding` property that you can use to set the same padding for all four sides of the content. You can set this property either to a number of pixels or a percentage. We'll use pixels and set the padding to 25 pixels.



```
.guarantee {
  border-color:    black;
  border-width:    1px;
  border-style:    solid;
  background-color: #a7cece;
  padding:         25px;
}
```

We're adding 25 pixels of padding to all sides of the content (top, right, bottom, and left).

## A test drive with some padding



When you reload the page in your browser, you'll notice the text in the guarantee paragraph has a little more breathing room on the sides now. There's some space between the text and the border, and it's much easier to read.

Now you can see 25 pixels of space between the edge of the text content and the border.

Notice that the background color is under both the content and the padding. But it doesn't extend into the margin.

the relaxing sentiments of sights from eras past. And, don't forget, the lounge offers free wireless access to the Internet, so bring your laptop.

Our guarantee: at the lounge, we're committed to providing you, our guest, with an exceptional experience every time you visit. Whether you're just stopping by to check in on email over an elixir, or are here for an out-of-the-ordinary dinner, you'll find our knowledgeable service staff pay attention to every detail. If you're not fully satisfied, have a Blueberry Bliss Elixir on us.

But that's not all; at night, join us in the backroom as our resident DJ spins a choice selection of trance and drum&bass beats across our spacious tiki-themed dance floor. Or just hang out in one of

## Now let's add some margin

Margins are easy to add using CSS. Like padding, you can specify the margin as a percentage or in pixels. You're going to add a 30-pixel margin around the entire guarantee paragraph. Here's how you do that:

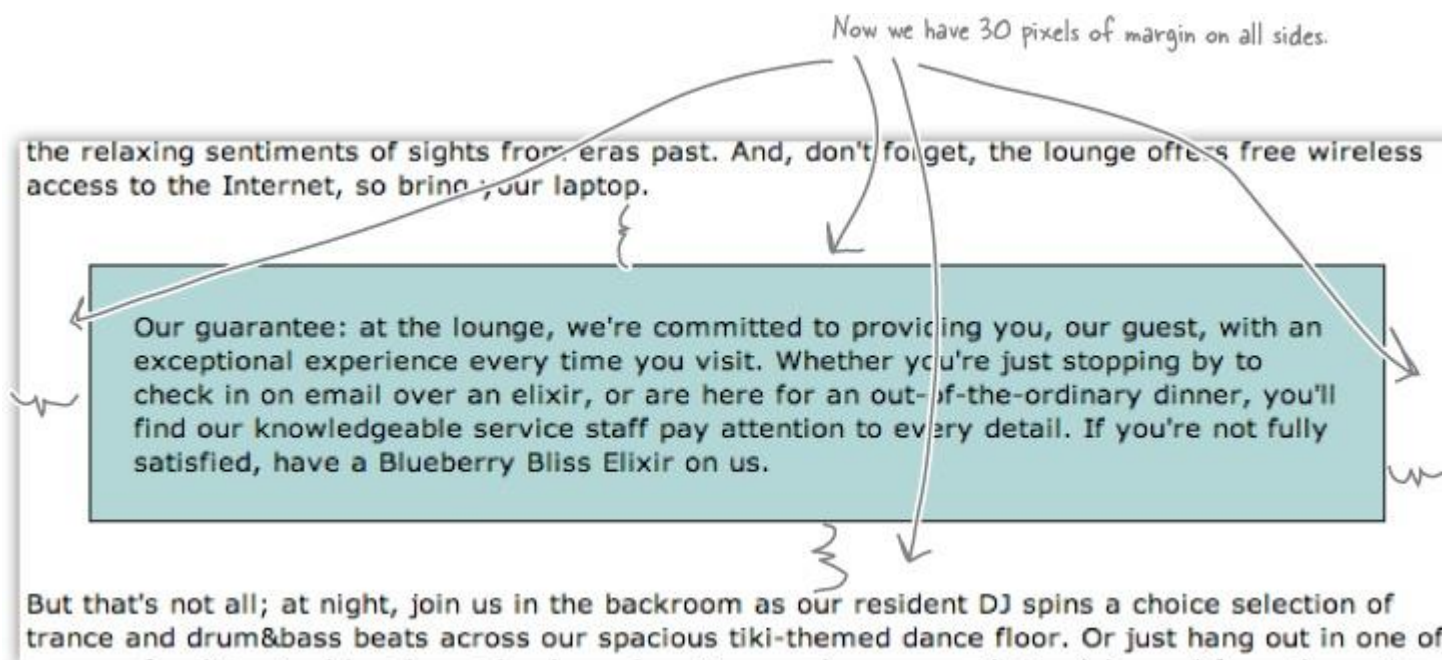
```
.guarantee {  
    border-color:    black;  
    border-width:    1px;  
    border-style:    solid;  
    background-color: #a7cece;  
    padding:         25px;  
    margin:          30px;  
}
```

← We're adding 30 pixels of margin to all sides of the content (top, right, bottom, and left).

## A test drive with the margin



When you reload the lounge page, you'll see the paragraph is really beginning to stand out on the page. With the margins in place, the paragraph looks inset from the rest of the text, and that, combined with the background color, makes it look more like a "callout" than an ordinary paragraph. As you can see, with only a few lines of CSS, you're doing some powerful things.





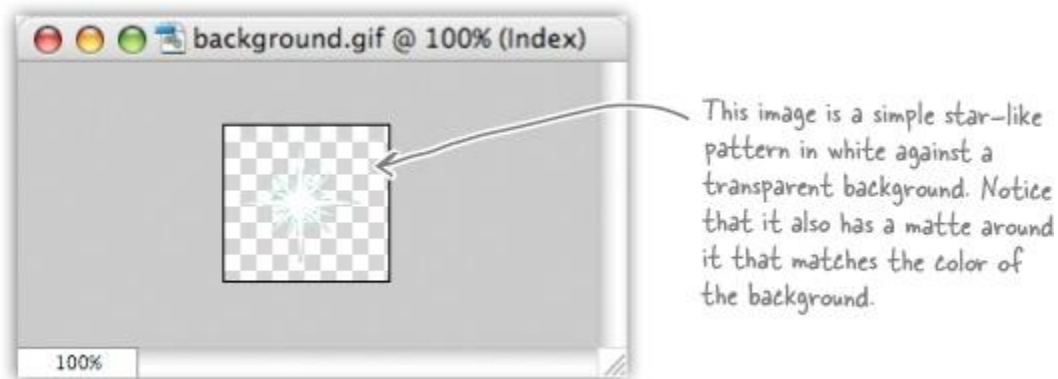
## EXERCISE

If you look at the guarantee paragraph as it's supposed to look in its final form, it has an italic, serif font, a line height greater than the rest of the page, and (if you're looking really close) gray text. Write the CSS below to set the line height to 1.9em, the font style to italic, the color to #444444, and the font family to Georgia, "Times New Roman", Times, serif. Check your CSS with the answers in the back of the chapter, then type it in and test.

## Adding a background image

You're almost there. What's left? We still need to get the white "guarantee star" graphic into the paragraph and work on the border, which is a solid, black line. Let's tackle the image first.

If you look in the "chapter9/lounge/images" folder, you'll find a GIF image called "background.gif" that looks like this:



Now you just need to get that image into your paragraph element, so you'll be using an `<img>` element, right? Not so fast. If you're adding an image to the background of an element, there is another way. Using CSS, you can add a background image to any element using the `background-image` property. Let's give it a try and see how it works:

```
.guarantee {  
    line-height: 1.9em;  
    font-style: italic;  
    font-family: Georgia, "Times New Roman", Times, serif;  
    color: #444444;  
    border-color: black;  
    border-width: 1px;  
    border-style: solid;  
    background-color: #a7cece;  
    padding: 25px;  
    margin: 30px;  
    background-image: url(images/background.gif);  
}
```

Here are the properties you added  
in the exercise on the previous page.



↖ Add this to your CSS, save, and reload your page.



Wait a sec, it seems like we have two ways to put images on a page. Is `background-image` a replacement for the `<img>` element?

No, the `background-image` property has a very specific purpose, which is to set the background image of an element. It isn't for placing images in a page—for that, you definitely want to use the `<img>` element.

Think about it this way: a background image is pure presentation, and the only reason you would use a `background-image` property is to improve the attractiveness of your element. An `<img>` element, on the other hand, is used to include an image that has a more substantial role in the page, like a photo or a logo.

Now, we could have just placed the image inside the paragraph, and we could probably get the same look and feel, but the guarantee star is pure decoration—it has no real meaning on the page, and it's only meant to make the element look better. So, `background-image` makes more sense.

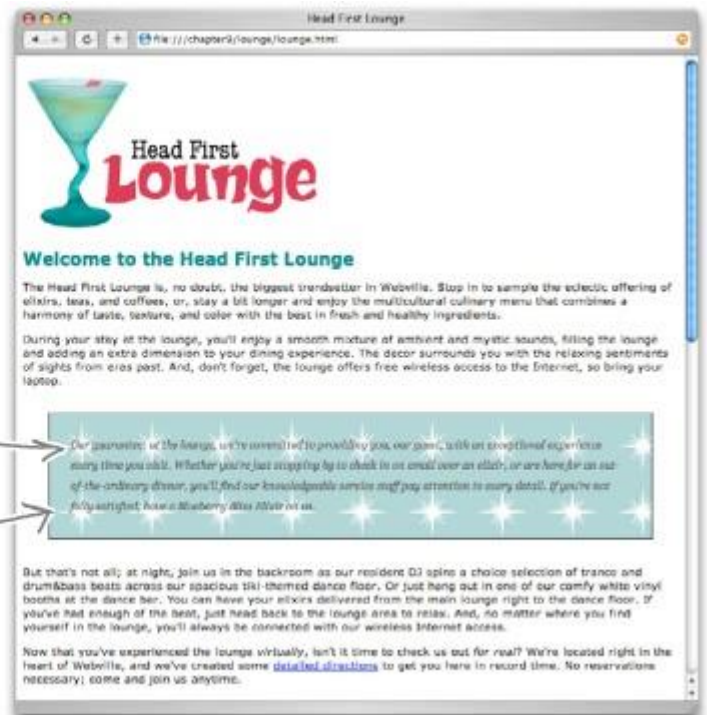
# Test driving the background image



Well, this is certainly an interesting test drive—we have a background image, but it appears to be repeated many times. Let's take a closer look at how to use CSS background images, and then you'll be able to fix this.

Here's the guarantee star image in the background. Notice that it sits on top of the background color, and because it has a transparent background, it lets the color show through.

Also notice that background images, like the background color, only show under the content area and padding, and not outside the border in the margin.



## CSS UP CLOSE

The `background-image` property places an image in the background of an element. Two other properties also affect the background image: `background-position` and `background-repeat`.

```
background-image: url(images/background.gif);
```

The `background-image` property is set to a URL, which can be a relative path or a full-blown URL (`http://...`).

Notice that no quotes are required around the URL.

# Fixing the background image

By default, background images are repeated. Luckily, there is a `no-repeat` value for the `background-repeat` property. Also, by default, browsers position a background image in the top left of the element, which is where we want it, but let's also add a `background-position` property just to give it a try.

```
.guarantee {  
    line-height:      1.9em;  
    font-style:       italic;  
    font-family:      Georgia, "Times New Roman", Times, serif;  
    color:            #444444;  
    border-color:      black;  
    border-width:      1px;  
    border-style:      solid;  
    background-color:  #a7cece;  
    padding:          25px;  
    margin:            30px;  
    background-image:  url(images/background.gif);  
    background-repeat: no-repeat;  
    background-position: top left;  
}
```

Handwritten annotations:

- ← You've got two new properties to add. (pointing to `background-repeat` and `background-position`)
- ← We want the background image to not repeat. (pointing to `no-repeat`)
- ← And we want it in the top-left corner. (pointing to `top left`)

The `background-position` property sets the position of the image and can be specified in pixels, or as a percentage, or by using keywords like `top`, `left`, `right`, `bottom`, and `center`.

```
background-position: top left;
```

Handwritten annotations:

- ← Places the image in the top left of the element. (pointing to `top left`)
- ← There are many different ways to position things in CSS, and we'll be talking more about that in two chapters. (pointing to the entire line)

By default, a background image is “tiled,” or repeated over and over to fill the background space. The `background-repeat` property controls how this tiling behaves.

**background-repeat: repeat;**

Sets the image to repeat both horizontally and vertically. This is the default behavior.

Here are the other background-repeat values you can use.

**no-repeat**

Display the image once; don't repeat the image at all.

**repeat-x**

Repeat the image only horizontally.

**repeat-y**

Repeat the image only vertically.

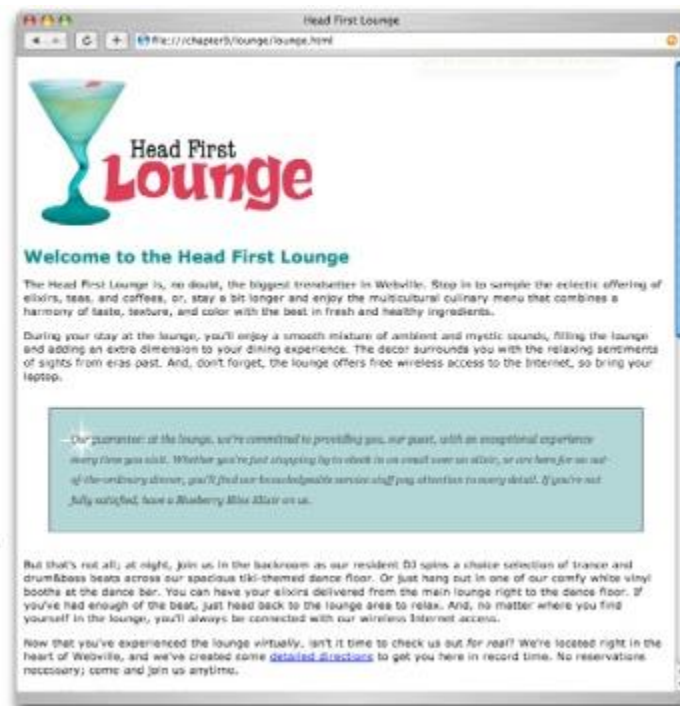
**inherit**

Just do whatever the parent element does.

## Another test drive of the background image



Here we go again. This time, it looks like we're much closer to what we want. But since this is a background image, the text can sit on top of it. How do we fix this? That's exactly what padding is for! Padding allows you to add visual space around the content area. Let's increase the padding on the left and see if we can nail this down once and for all.



This is much better. Now the image isn't repeated.

But we'd really like for the text not to run over the top of the image.

## How do you add padding only on the left?



For padding, margins, and even borders, CSS has a property for every direction: top, right, bottom, and left. To add padding on the left side, use the `padding-left` property, like this:

```
.guarantee {
    line-height:      1.9em;
    font-style:       italic;
    font-family:      Georgia, "Times New Roman", Times, serif;
    color:            #444444;
    border-color:      black;
    border-width:      1px;
    border-style:      solid;
    background-color:  #a7cece;
    padding:           25px;
    padding-left:      80px;
    margin:            30px;
    background-image:  url(images/background.gif);
    background-repeat: no-repeat;
    background-position: top left;
}
```

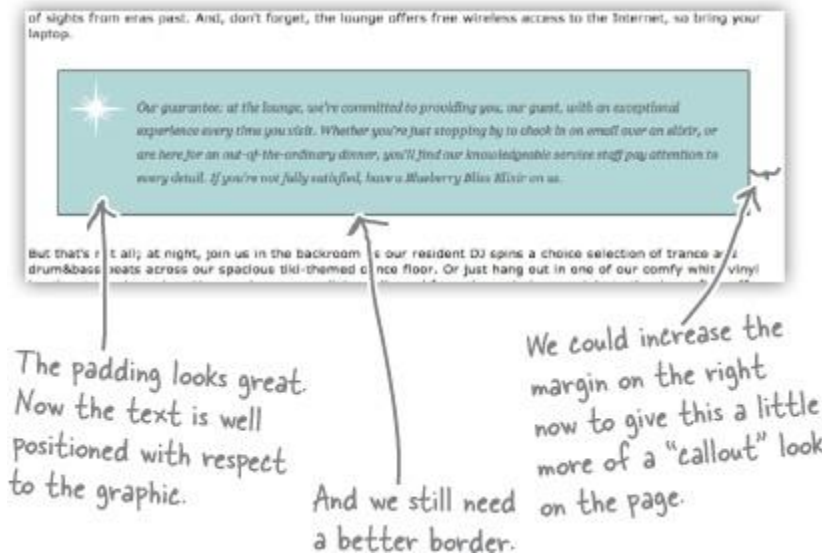
We're using the `padding-left` property to increase the padding on the left.

Notice that we first set the padding on all sides to 25 pixels, and then we specify a property for the left side.

Order matters here—if you switch the order, then set the padding for the left side first, and then the general padding property will set all sides back to 25 pixels, including the left side!

## Are we there yet?

Make sure you save your changes and reload the page. You should see more padding on the left side of the paragraph, and the text is now positioned well with respect to the guarantee star. This is a great example of where you use padding instead of margins. If you need more visual space around the content area itself, use padding, as opposed to if you want space between elements or the sides of the page, in which case, use margin. In fact, we could actually use a little more margin on the right side to set the paragraph off even more. Let's do that, and then all we need to do is fix the border.



## How do you increase the margin just on the right?

You do this just like you did with the padding: add another property, `margin-right`, to increase the right margin.

### NOTE

See the pattern? There's a property to control all sides together, and properties for each side if you want to set them individually.

```

.guarantee {
    line-height:      1.9em;
    font-style:       italic;
    font-family:      Georgia, "Times New Roman", Times, serif;
    color:            #444444;
    border-color:      black;
    border-width:      1px;
    border-style:      solid;
    background-color:  #a7cece;
    padding:          25px;
    padding-left:      80px;
    margin:            30px;
    margin-right:      250px;
    background-image:  url(images/background.gif);
    background-repeat: no-repeat;
    background-position: top left;
}

```

Remember, we're already setting the margins to be 30 pixels.

And now we're going to override that setting for the right side, and set it to 250 pixels.

Add the new `margin-right` property and reload. Now the paragraph should have 250 pixels of margin on the right side.



## A two-minute guide to borders

There's only one thing left to do to perfect the guarantee paragraph: add a better border. Before you do, take a look at all the different ways you can control the border of an element.

### BORDER STYLE

The `border-style` property controls the visual style of the border. There are eight border styles available, from a solid line to dotted lines to ridges and grooves.

**border-style: groove;**

To specify a border style, just use the border-style property and a value of one of the available styles.

The solid style is just what it sounds like: a solid border.

Go with **solid**, the original.

The dotted style looks like a series of dots.

Once you go **dotted**, you'll never go back.

The double style uses two lines.

Go with **double**; I'm twice the fun.

And the dashed style is just a set of dashes around the border.

Ignore dotted; use **dashed**.

A groove style looks like a groove in the page (difficult to see in a book).

I'm the border that's got the **groove**.

The inset style looks like an inset that sinks into the page.

I'm the only "in" style: **inset**.

The outset style looks like an outset that rises from the page.

Go with me; I've been better since the **outset**.

The ridge style looks like a raised ridge on the page.

I'm more fun; I've got **ridges**.

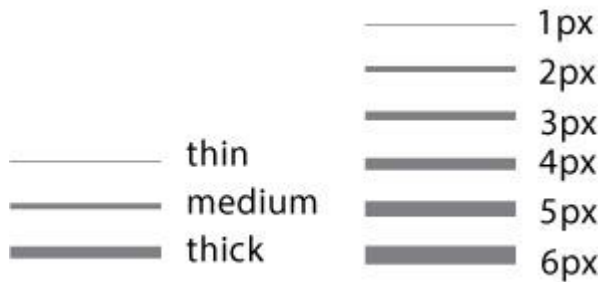
## BORDER WIDTH

The border-width property controls the width of the border. You can use keywords or pixels to specify the width.

```
border-width: thin;
```

```
border-width: 5px;
```

You can specify widths using the keywords thin, medium, or thick, or by the number of pixels.



## BORDER COLOR

The border-color property sets the color of the border. This works just like setting font colors; you can use color names, rgb values, or hex codes to specify color.

```
border-color: red;
```

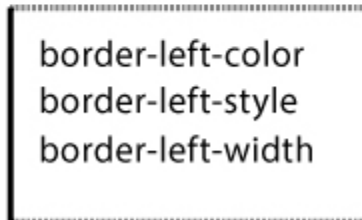
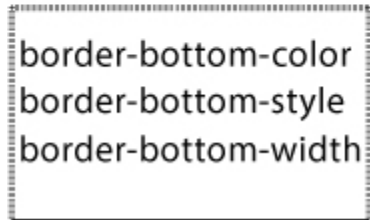
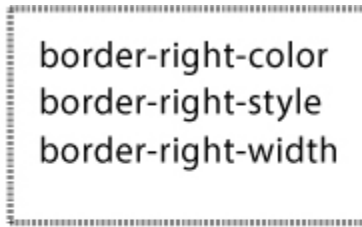
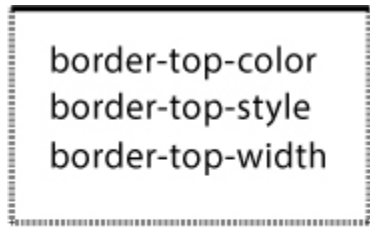
```
border-color: rgb(100%, 0%, 0%);
```

```
border-color: #ff0000;
```

Use border-color to specify the color of a border. You can use any of the common ways to specify color.



## SPECIFYING BORDER SIDES



Just as with margins and padding, you can specify border style, width, or color on any side (top, right, bottom, or left):

```
border-top-color: black;  
border-top-style: dashed;  
border-top-width: thick;
```

↑  
These properties are for the top border only. You can specify each side of the border independently.

## SPECIFYING BORDER CORNERS

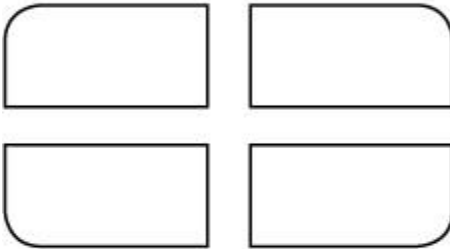
You can create rounded corners on all four corners, or just one corner, or any combination.





You can specify all four corners with one number.

**border-radius: 15px;**



Or you can specify each corner separately. Notice that you can use px or em to specify the radius size.

**border-top-left-radius: 3em;  
border-top-right-radius: 3em;  
border-bottom-right-radius: 3em;  
border-bottom-left-radius: 3em;**

If you use em, the measurement of the border radius is relative to the font size of the element, just like when you use em for font-size.



**border-top-left-radius: 15px;  
border-top-right-radius: 0px;  
border-bottom-right-radius: 0px;  
border-bottom-left-radius: 15px;**

You can get all kinds of interesting shapes using border-radius.

## Border fit and finish

It's time to finish off the guarantee paragraph. All we need to do is give it a ragged-looking border. But which style is that? The available styles are solid, double, dotted, dashed, groove, ridge, inset, and outset. So how do we make it look ragged? It's actually just a trick: we're using a dashed border that has its color set to white (matching the background color of the page). Here's how you do it. Begin by just making the border dashed. Find the `border-style` property in your "lounge.css" and change it, like this:

**border-style: dashed;**

Here we've changed the border from solid to dashed.

Go ahead and save the file and reload. You should see a border like this:



Now, to get a ragged-looking border, just set the color of the border to white. This makes the border look like it is cutting into the background color. Give it a try: find the `border-color` property and set it to `white`.

```
border-color: white;
```

And here we've changed  
the border color from  
black to white.

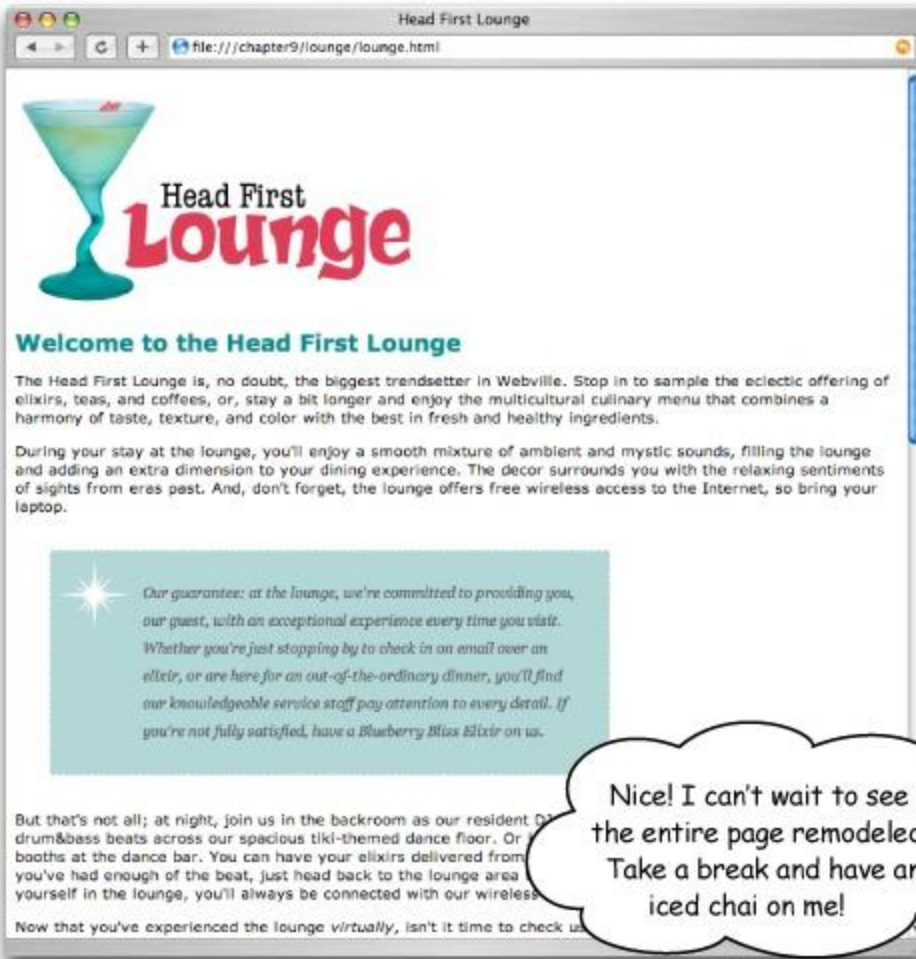
Save the file and reload again. Now you should see the ragged border:



WATCH IT!

**Browsers don't always agree on the size of thin, medium, and thick.**

*Browsers can have different default sizes for the keywords thin, medium, and thick, so if the size of your border is really important to you, consider using pixel sizes instead.*



Nice! I can't wait to see  
the entire page remodeled.  
Take a break and have an  
iced chai on me!



# Congratulations!

**Bravo! You've taken an ordinary HTML paragraph and transformed it into something a lot more appealing and stylish using only 15 lines of CSS.**

**It was a long trip getting here, so at this point we encourage you to take a little break. Grab yourself an iced chai and take a little time to let things sink in—when you come back, we'll nail down a few more of the fine points of CSS.**

## EXERCISE

While you're drinking that iced chai, try your hand at adding a border-radius to the guarantee paragraph. We've got some examples below of the guarantee paragraph with a variety of border-radius values set. Write the CSS to create the border you see in the example. For each example, we've provided the size of the border-radius used to create the rounded corners in that example.

30px



Our guarantee: at the lounge, we're committed to providing you, our guest, with an exceptional experience every time you visit. Whether you're just stopping by to check in on email over an elixir, or are here for an out-of-the-ordinary dinner, you'll find our knowledgeable service staff pay attention to every detail. If you're not fully satisfied, have a Blueberry Bliss Elixir on us.

Write your CSS



40px



Our guarantee: at the lounge, we're committed to providing you, our guest, with an exceptional experience every time you visit. Whether you're just stopping by to check in on email over an elixir, or are here for an out-of-the-ordinary dinner, you'll find our knowledgeable service staff pay attention to every detail. If you're not fully satisfied, have a Blueberry Bliss Elixir on us.

40px



Our guarantee: at the lounge, we're committed to providing you, our guest, with an exceptional experience every time you visit. Whether you're just stopping by to check in on email over an elixir, or are here for an out-of-the-ordinary dinner, you'll find our knowledgeable service staff pay attention to every detail. If you're not fully satisfied, have a Blueberry Bliss Elixir on us.

2em



Our guarantee: at the lounge, we're committed to providing you, our guest, with an exceptional experience every time you visit. Whether you're just stopping by to check in on email over an elixir, or are here for an out-of-the-ordinary dinner, you'll find our knowledgeable service staff pay attention to every detail. If you're not fully satisfied, have a Blueberry Bliss Elixir on us.

**Welcome back, and good timing. We're just about to listen in on an interview with a class...**

THE CLASS EXPOSED

**This week's interview: Are classes always right?**

**Head First:** Hey, Class; you know we've been making good use of you, but we still don't know a lot about you.

**Class:** Well, there's not all that much to know. If you want to create a "group," so to speak, that you can style, just come up with a class, put your elements in it, and then you can style all the elements in that class together.

**Head First:** So the class lets you take sets of elements and apply one or more style properties to them?

**Class:** Exactly. Say you have some holiday-themed areas in your page—one Halloween, one Christmas. You could add all Halloween elements to the `halloween` class and all Christmas elements to the `christmas` class. Then you can style those elements independently—say, orange for Halloween and red for Christmas—by writing rules that apply to each class.

**Head First:** That makes a lot of sense. We just saw a good example of that in this chapter, didn't we?

**Class:** I'm not sure; I was off working. You'll have to catch me up.

**Head First:** Well, we have a paragraph on the Head First Lounge page that contains a written guarantee from the owners, and they want this paragraph to stand out independently of the other paragraphs.

**Class:** So far, so good...but let me ask you this: are there a few of these paragraphs, or just the one?

**Head First:** Well, I don't think there is any reason to have multiple guarantee paragraphs, and I don't see the same style being applied anywhere else in the page, so just the one.

**Class:** Hmmm, I don't like the sound of that. You see, classes are meant to be used for styles that you want to reuse with multiple elements. If you've got one unique element that you need styled, that's not really what classes are for.

**Head First:** Wait a second—a class seemed to work perfectly...how can this be wrong?

**Class:** Whoa, now, don't freak out. All you need to do is switch your `class` attribute to an `id` attribute. It will only take you a minute.

**Head First:** An `id` attribute? I thought those were for those link destinations, like in [Chapter 4](#)?

**Class:** `ids` have lots of uses. They're really just unique identifiers for elements.

**Head First:** Can you tell us a little more about `id` attributes? This is all news to me. I mean, I just went through an entire chapter using `class` incorrectly!



**Class:** Hey, no worries; it's a common mistake. Basically, all you need to know is that you use a `class` when you might want to use a style with more than one element. And if what you need to style is unique and there's only one on your page, then use an `id`. The `id` attribute is strictly for naming unique elements.

**Head First:** Okay, I think I've got it, but why does it really matter? I mean, `class` worked just fine for us.

**Class:** Because there are some things you really want *only one* of on your page. The guarantee paragraph you mentioned is one example, but there are better examples, like the header or footer on your page, or a navigation bar. You're not going to have two of those on a page. Of course, you can use a class for just one element, but someone else could come along and add another element to the class, and then your element won't have a unique style anymore. It also becomes important when you are positioning HTML elements, which is something you haven't gotten to yet.



**Head First:** Well, okay, Class. This conversation has certainly been educational for us. It sounds like we definitely need to convert that paragraph from a `class` to an `id`. Thanks again for joining us.

**Class:** Any time, Head First!

## BRAIN POWER

Choose whether you'd use class or id for the following elements:

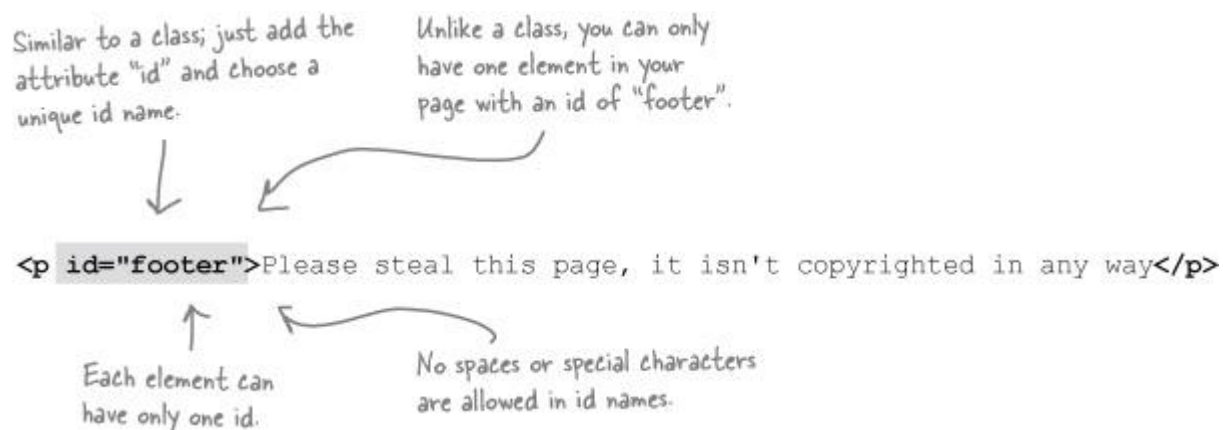
id	class	
<input type="checkbox"/>	<input type="checkbox"/>	A paragraph containing the footer of a page.
<input type="checkbox"/>	<input type="checkbox"/>	A set of headings and paragraphs that contain company biographies.
<input type="checkbox"/>	<input type="checkbox"/>	An <code>&lt;img&gt;</code> element containing a "picture of the day."
<input type="checkbox"/>	<input type="checkbox"/>	A set of <code>&lt;p&gt;</code> elements containing movie reviews.
<input type="checkbox"/>	<input type="checkbox"/>	An <code>&lt;ol&gt;</code> element containing your to-do list.

id	class	
		<q> elements containing <i>Buckaroo Banzai</i> quotes.

Answer: The footer, the picture of the day, and the to-do list are great candidates for using id.

## The id attribute

Because you’ve already used ids on <a> elements, and because you already know how to use a `class` attribute, you’re not going to have to learn much to use the `id` attribute. Say you have a footer on your page. There’s usually only one footer on any page, so that sounds like the perfect candidate for an `id`. Here’s how you’d add the identifier `footer` to a paragraph that contains the footer text:



Giving an element an id is similar to adding an element to a class. The only differences are that the attribute is called `id`, not `class`; an element can’t have multiple ids; and you can’t have more than one element on a page with the same id.

### THERE ARE NO DUMB QUESTIONS

**Q: What’s the big deal? Why do I need an id just to prove something is unique on the page? I could use a class exactly the same way, right?**

**A:** Well, you can always “simulate” a unique id with a class, but there are many reasons not to. Say you’re working on a web project with a team of people. One of your teammates is going to look at a class and think it can be reused with other elements. If, on the other hand, she sees an id, then she’s going to know that’s for a unique element.

**A:** There are a couple of other reasons ids are important that you won’t see for a few

chapters. For instance, when you start positioning elements on a page, you'll need each element you want to position to have a unique id.

**Q: Can an element have an id and also belong to a class?**

**A:** Yes, it can. Think about it this way: an id is just a unique identifier for an element, but that doesn't prevent it from belonging to one or more classes (just like having a

**A:** unique name doesn't prevent you from joining one or more clubs).

## But how do I use id in CSS?

You select an element with an id almost exactly like you select an element with a class. To quickly review: if you have a class called `specials`, there are a couple of ways you can select elements using this class. You could select just certain elements in the class, like this:

```
p.specials {  
    color: red;  
}
```



This selects only paragraphs that are in the specials class.

Or you can select all the elements that belong to the `specials` class, like this:

```
.specials {  
    color: red;  
}
```



This selects all elements in the specials class.

Using an id selector is very similar. To select an element by its id, you use a # (hash mark) character in front of the id (compare this to class, where you use a . [dot] in front of the class name). Say you want to select any element with the id `footer`:

```
#footer {  
    color: red;  
}
```



This selects any element that has the id "footer".

Or you could select only a `<p>` element with the id `footer`, like this:

```
p#footer {  
    color: red;  
}
```



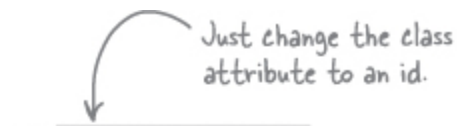
This selects a <p> element if it has the id "footer".

The other difference between class and id is that an id selector should match *only one* element in a page.

## Using an id in the lounge

Our guarantee paragraph really should have an id since it's intended to be used just once in the page. While we should have designed it that way from the beginning, making the change is going to be quite simple.

### Step one: Change the class attribute to an id in your “lounge.html” file



```
<p id="guarantee">
    Our guarantee: at the lounge, we're committed to providing
    you, our guest, with an exceptional experience every time you
    visit. Whether you're just stopping by to check in on email
    over an elixir, or are here for an out-of-the-ordinary dinner,
    you'll find our knowledgeable service staff pay attention to every
    detail. If you're not fully satisfied, have a Blueberry Bliss Elixir
    on us.
</p>
```

### Step two: Change the “.guarantee” class selector in “lounge.css” to an id selector

Just change the . to a # in the selector.

```
#guarantee {  
    line-height:      1.9em;  
    font-style:       italic;  
    font-family:      Georgia, "Times New Roman", Times, serif;  
    color:            #444444;  
    border-color:      white;  
    border-width:      1px;  
    border-style:      dashed;  
    background-color:  #a7cece;  
    padding:          25px;  
    padding-left:      80px;  
    margin:            30px;  
    margin-right:      250px;  
    background-image:  url(images/background.gif);  
    background-repeat: no-repeat;  
    background-position: top left;  
}
```

**Step three: Save your changes and reload the page**



Well, everything should look  
EXACTLY the same. But don't  
you feel much better now that  
everything is as it should be?



## THERE ARE NO DUMB QUESTIONS

**Q:** **Q:** So why did you make the selector `#guarantee` rather than `p#guarantee`?

**A:** We could have done either, and they both would select the same thing. On this page, we know that we will always have a paragraph assigned to the id, so it doesn't really matter (and `#guarantee` is simpler). However, on a more complex set of pages, you might have some pages where the unique id is assigned to, say, a paragraph, and on others it's assigned to a list or block quote. So you might want several rules for the id, like `p#someid`, and `blockquote#someid`, depending on which kind of element is on the

**A:** page.

**Q:** **Q:** Should I always start with a class, and then change it to an id when I know it's going to be unique?

**A:** No. You'll often know when you design your pages if an element is going to be unique or not. We only did things this way in the chapter because, well, you didn't know about id when we started. But don't you think we tied id into the story rather nicely?

**A:** 😊

**Q:** **Q:** What are the rules for class and id names?



**A:** *Class names should begin with a letter, but id names can start with a number or a letter. Both id and class names can contain letters and numbers as well as the \_ character, but no spaces. So “number1” works, as does “main\_content”, but not “header content”. Just remember, ids must be unique!*

## Remixing stylesheets

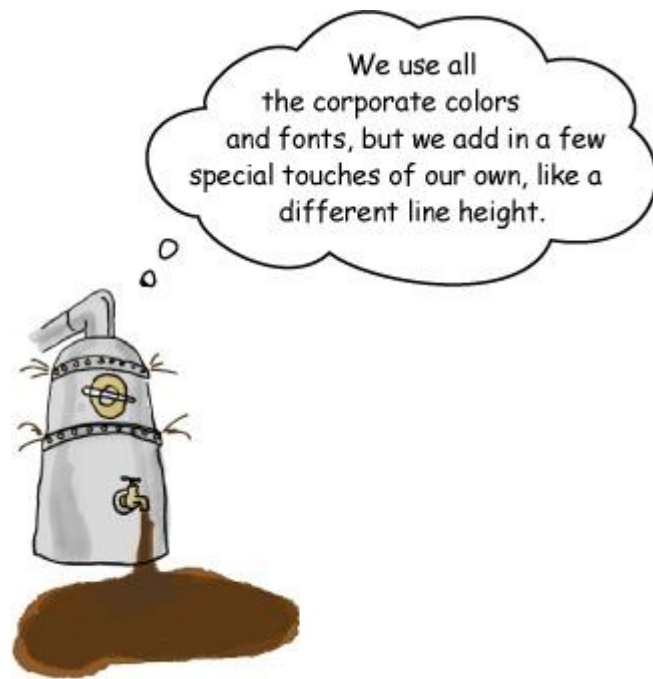
Before we wind this chapter down, let’s have a little fun remixing some stylesheets. So far, you’ve been using only one stylesheet. Well, who ever said you can’t use more than one stylesheet? You can specify a whole set of stylesheets to be used with any HTML. But you may be wondering why anyone would want to. There are a couple of good reasons. Here’s the first one...



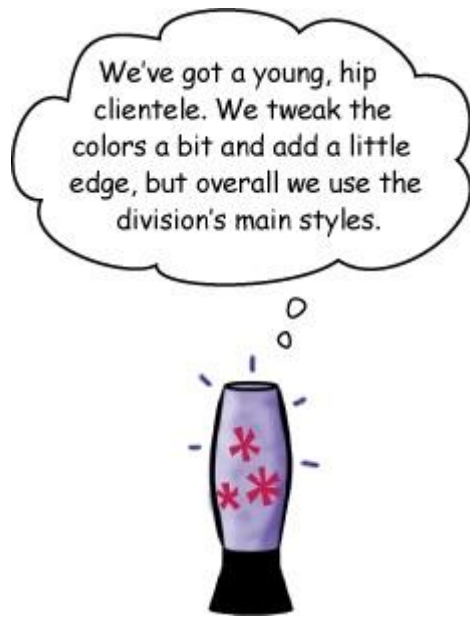
Imagine that the Head First Lounge takes off, gets franchised, does the IPO, and so on (all thanks to you and your terrific web work, of course). Then there would be a whole corporate website with hundreds of pages, and obviously you’d want to style those pages with external CSS stylesheets. There would be various company divisions, and they might want to tweak the styles in individual ways. And the lounge franchises also might want some control over style. Here’s how that might look:



*Corporate*



*Beverage Division*



*Seattle Lounge (part of the Beverage Division)*

## Using multiple stylesheets

So how do you start with a corporate style and then allow the division and the lounge franchises to override and make changes to the styles? You use several stylesheets, like this:

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Head First Lounge</title>
    <link type="text/css" href="corporate.css" rel="stylesheet">
    <link type="text/css" href="beverage-division.css" rel="stylesheet">
    <link type="text/css" href="lounge-seattle.css" rel="stylesheet">
  </head>
  <body>
    .
    .
    .
  </body>
</html>

```

In your HTML, you can specify more than one stylesheet. Here, we've got three.

One stylesheet for the entire corporation.

And the lounge in Seattle has its own tweaks in its stylesheet.

The beverage division can add to the corporate style a little, or even override some of the corporate styles.

Order matters! A stylesheet can override the styles in the stylesheets linked above it.

## THERE ARE NO DUMB QUESTIONS

**Q: Q: So the order of the stylesheets matters?**

**A:** Yes, they go top to bottom, with the stylesheets on the bottom taking precedence. So if you have, say, a font-family property on the <body> element in both the corporate and the division stylesheets, the division's takes precedence, since it's the last one

**A:** linked in the HTML.

**Q: Q: Do I ever need this for a simple site?**

**A:** You'd be surprised. Sometimes there's a stylesheet you want to base your page on, and rather than changing that stylesheet, you just link to it, and then supply your own

**A:** stylesheet below that to specify what you want to change.

**Q: Q: Can you say more about how the style for a specific element is decided?**

**A:** We talked a little about that in [Chapter 7](#). And for now, just add to that knowledge

**A:** that the ordering of the stylesheets linked into your file matters. In the next chapter,

*after you've learned a few other CSS details, we're going to go through exactly how the browser knows which style goes with which element.*

## Stylesheets—they're not just for desktop browsers anymore...



There's actually another reason you might want to have multiple style files: let's say you want to tailor your page's style to the type of device your page is being displayed on (desktops, laptops, tablets, smartphones, or even printed versions of your pages). Well, to do that there is a `media` attribute you can add to the `<link>` element that lets you use only the style files that are appropriate for your device. Let's look at an example:

The `media` attribute allows you to specify the type of device this stylesheet is for.

You specify the type of device by creating a "media query," which is matched with the device.

```
<link href="lounge-mobile.css" rel="stylesheet" media="screen and (max-device-width: 480px)"
```

Here our query specifies anything with a screen (as opposed to, say, a printer, or 3D glasses, or a braille reader)...

...and any device that has a width of at most 480 pixels.

Likewise, we could create a query that matches the device if it is a printer, like this:

```
<link href="lounge-print.css" rel="stylesheet" media="print">
```

The `lounge-print.css` file is only going to be used if...

...the media type is "print", which means we're viewing it on a printer.

There are a variety of properties you can use in your queries, like `min-device-width`, `max-device-width` (which we just used), and the `orientation` of the display (landscape or portrait), to name just a few. And keep in mind you can add as many `<link>` tags to your HTML as necessary to cover all the devices you need to.

## Add media queries right into your CSS

There's another way to target your CSS to devices with specific properties: rather than using media queries in link tags, you can also use them right in your CSS. Here's an example:

Use the `@media` rule...      ...followed by your query.

```
@media screen and (min-device-width: 481px) {  
  #guarantee {  
    margin-right: 250px;  
  }  
}  
  
@media screen and (max-device-width: 480px) {  
  #guarantee {  
    margin-right: 30px;  
  }  
}  
  
@media print {  
  body {  
    font-family: Times, "Times New Roman", serif;  
  }  
}  
  
p.specials {  
  color: red;  
}
```

And then put all the rules that apply to devices matching this query within curly braces.

So, these rules will be used if the screen is wider than 480px...

...these rules will be used if the screen is 480px or less...

...and these rules will be used if you're printing the page.

All other rules apply to all pages because they aren't contained within a `@media` rule.

So, the way this works, only the CSS rules that are specific to a media type are included in an `@media` rule. All the rules that are common to all the media types are included in the CSS file below the `@media` rules, so that way you don't have any unnecessarily repeated rules. And, when a browser loads the page, it determines through the media queries the rules that are appropriate for the page, and ignores any that don't match.