Presented by TechRepublic

A Brain-Friendly Guide to HTML & CSS

Head First HTML with CSS & XHTML

Launch your Web career in one chapter

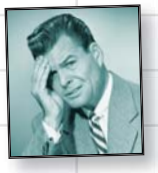
A learner's guide to creating standards-based Web pages

Watch out for common HTML & CSS traps and pitfalls

Learn why everything your friends know about style is probably wrong

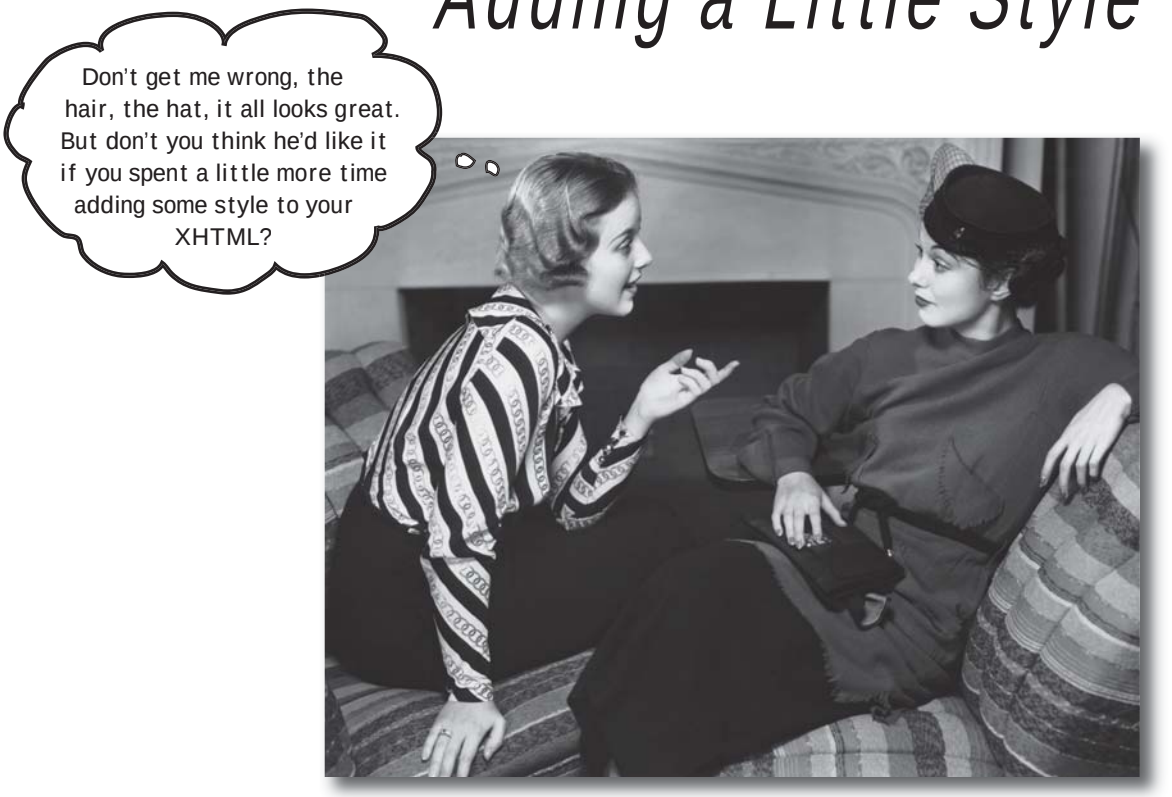
Bend your mind around 100 puzzles & exercises

Avoid embarrassing validation mistakes



Elisabeth Freeman & Eric Freeman

Adding a Little Style



I was told there'd be CSS in this book. So far you've been concentrating on learning XHTML to create the structure of your Web pages. But as you can see, the browser's idea of style leaves a lot to be desired. Sure, we could call the fashion police, but we don't need to. With CSS, you're going to completely control the presentation of your pages, often without even changing your XHTML. Could it really be so easy? Well, you *are* going to have to learn a new language; after all, Webville is a bilingual town. After reading this chapter's guide to learning the language of CSS, you're going to be able to stand on *either* side of Main Street and hold a conversation.

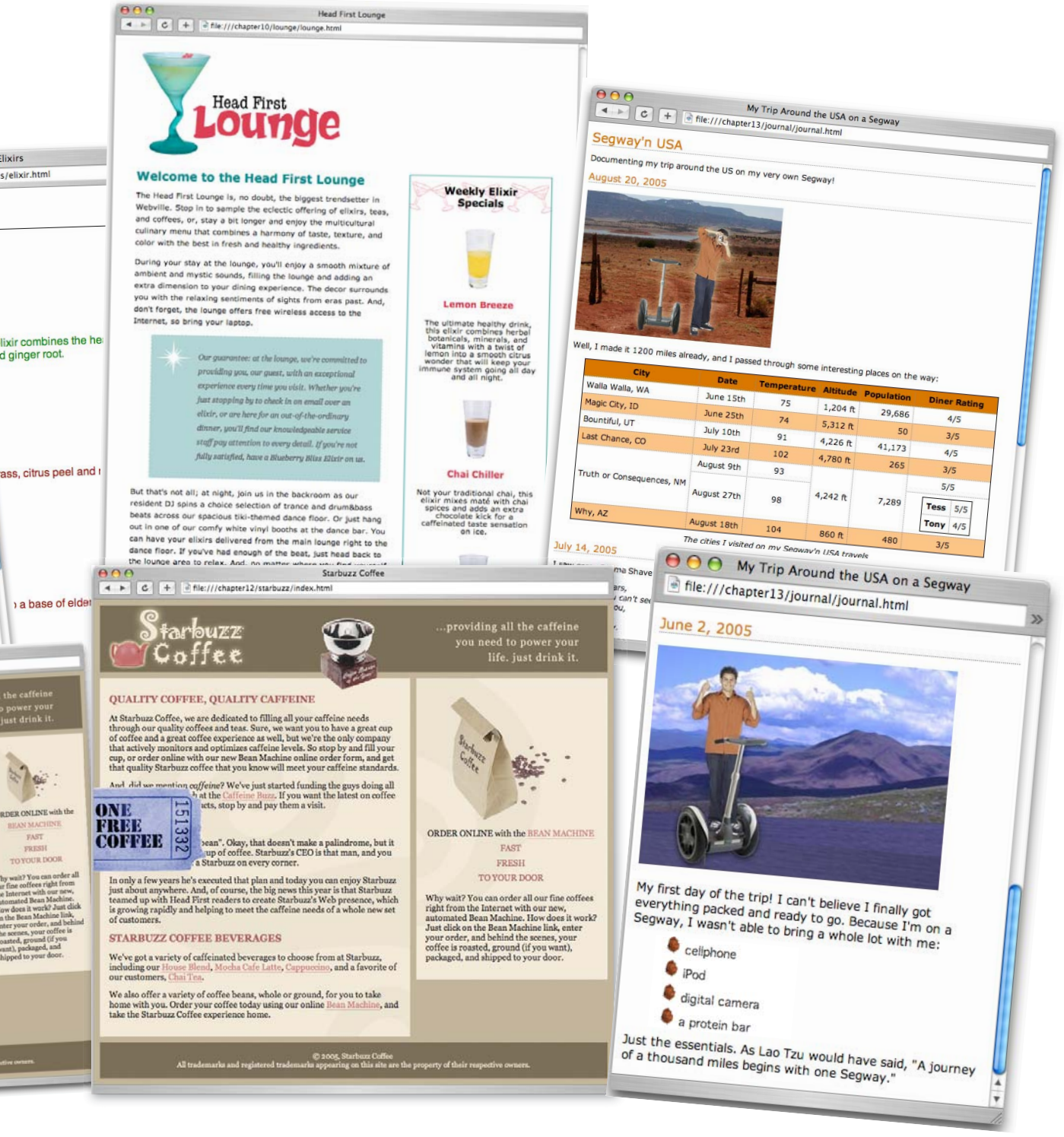
Remember the Wizard of Oz? Well, this is the part of the book where things go from black & white to color.

You're not in Kansas anymore

You've been a good sport learning about markup and structure and validation and proper syntax and nesting and compliance, but now you get to really start *having some fun* by styling your pages. But no worries, all those XHTML pushups you've been doing aren't going to waste. In fact, you're going to see that a solid understanding of XHTML is crucial to learning (and using) CSS. And, learning CSS is just what we're going to do over the next several chapters.

Just to tease you a bit, on these two pages we've sprinkled a few of the designs you're going to work with in the rest of the book. Quite a difference from the pages you've been creating so far, isn't it? So, what do you need to do to create them? Learn the language of CSS of course.

Let's get started...



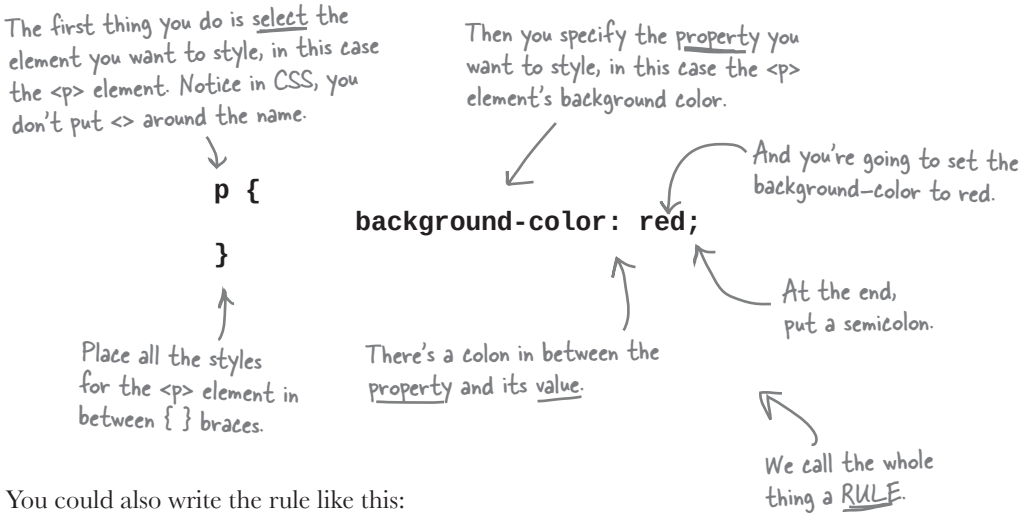
Overheard on Webville's "Trading Spaces"

Not up on the latest reality TV? No problem, here's a recap: take two neighbors, two homes, and $1,000. The two neighbors switch homes, and using the $1,000, totally redesign a room or two in 48 hours. Let's listen in...

Of course, in the Webville edition of the show, everyone talks about design in CSS. If you're having trouble understanding them, here's a little translation tip: each statement in CSS consists of a location (like bedroom), a property in that location (like drapes, or carpet), and a style to apply to that property (like the color blue, or 1 inch tiles).

Using CSS with XHTML

We're sure CSS has a bright future in the home design category, but let's get back to XHTML. XHTML doesn't have rooms, but it does have elements and those elements are going to be the locations that we're styling. Want to paint the walls of your **<p>** elements red? No problem; only paragraphs don't have walls, so you're going to have to settle for the paragraph's **background-color** property instead. Here's how you do that:



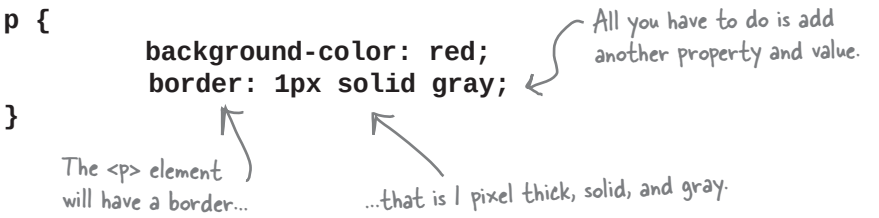
You could also write the rule like this:

```
p { background-color: red; }
```

Here, all we've done is remove the linebreaks. Like XHTML, you can format your CSS pretty much as you like. For longer rules you'll usually want to add some linebreaks and indenting to make the CSS more readable (for you).

Wanna add more style?

You can add as many properties and values as you like in each CSS rule. Say you wanted to put a border around your paragraphs, too. Here's how you do that:



there are no Dumb Questions

Q: Does every <p> element have the same style? Or can I, say, make two paragraphs different colors?

A: The CSS rules we've used so far define the style for *all* paragraphs, but CSS is very expressive: it can be used to specify styles in lots of different ways, for lots of different elements – even subsets of elements. You'll see how to make paragraphs two different colors later in this chapter.

Q: How do I know what properties I can set on an element?

A: Well, there are *lots* of properties that can be set on elements, certainly more than you'd want to memorize, in any case. You're going to get quite familiar with the more common properties in the next few chapters. You'll probably also want to find a good CSS reference. There are plenty of references online, and O'Reilly's *CSS Pocket Reference* is a great little book.

Q: Remind me why I'm defining all this style in a separate language, rather than in XHTML. Since the elements are written in XHTML, wouldn't it be easier just to write style in XHTML, too?

A: You're going to start to see some big advantages to using CSS in the next few chapters. But, here's a quick answer: CSS really is better suited for specifying style information than XHTML. Using just a small bit of CSS, you can create fairly large effects on the style of your XHTML. You're also going to see that CSS is a much better way to handle styles for multiple pages. You'll see how that works later in this chapter.

Say you have an **** element inside a paragraph. If you change the background color of the paragraph, do you think you also have to change the background of the **** element so it matches the background color of the paragraph?

Getting CSS into your XHTML

Okay, you know a little about CSS syntax now. You know how to select an element and then write a rule with properties and values inside it. But you still need to get this CSS into some XHTML. First, we need some XHTML to put it in. In the next few chapters, we're going to revisit our old friends – Starbuzz, and Tony and his Segway journal – and make things a little more stylish. But, who do you think is dying to have their site styled first? Of course, the Head First Lounge guys. So, here's the XHTML for the Head First Lounge main page. Remember, in the last chapter we fixed things up a little and made it strict XHTML (would you have expected any less of us?). Now, we're adding some style tags, the easiest way to get style into your pages.

But not necessarily the best way. We'll come back to this later in the chapter and see another way.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en" >
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <title>Head First Lounge</title>

    <style type="text/css">

    </style>

  </head>
  <body>
    <h1>Welcome to the Head First Lounge</h1>
    <p>
       <img src="images/drinks.gif" alt="Drinks" />
    </p>
    <p>
       Join us any evening for refreshing
       <a href="beverages/elixir.html">elixirs</a>,
       conversation and maybe a game or two
       of <em>Dance Dance Revolution</em>.
       Wireless access is always provided;
       BYOWS (Bring your own web server).
    </p>
    <h2>Directions</h2>
    <p>
       You'll find us right in the center of downtown
       Webville. If you need help finding us, check out our
       <a href="about/directions.html">detailed directions</a>.
       Come join us!
    </p>
  </body>
</html>
```

Here's what we're interested in: the <style> element.

To add CSS style directly to your XHTML, add opening and closing style tags in the <head> element.

And a style type of "text/css".

And your CSS rules are going to go right in here.

Adding style to the lounge

Now that you've got the **<style>** element in your XHTML, you're going to add some style to the Lounge to get a feel for writing CSS. This design probably won't win you any "design awards," but you gotta start somewhere.

The first thing we're going to do is change the color (something to match those red lounge couches) of the text in the paragraphs. To do that, we'll use the CSS **color** property like this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
  <head>
    <meta http-equiv="Content-Type"
          content="text/html; charset=ISO-8859-1" />
    <title>Head First Lounge</title>
    <style type="text/css">

      p {
          color: maroon;
      }

    </style>
  </head>
  <body>
    <h1>Welcome to the Head First Lounge</h1>
    <p>
       <img src="images/drinks.gif" alt="Drinks" />
    </p>
    <p>
       Join us any evening for refreshing
       <a href="beverages/elixir.html">elixirs</a>,
       conversation and maybe a game or two
       of <em>Dance Dance Revolution</em>.
       Wireless access is always provided;
       BYOWS (Bring your own web server).
    </p>
    <h2>Directions</h2>
    <p>
       You'll find us right in the center
       of downtown Webville. If you need
       help finding us, check out our
       <a href="about/directions.html">detailed directions</a>.
       Come join us!
    </p>
  </body>
</html>
```

Here's the rule that is going to specify the font color of the paragraphs.

We're selecting just the <p> element to apply this style to.

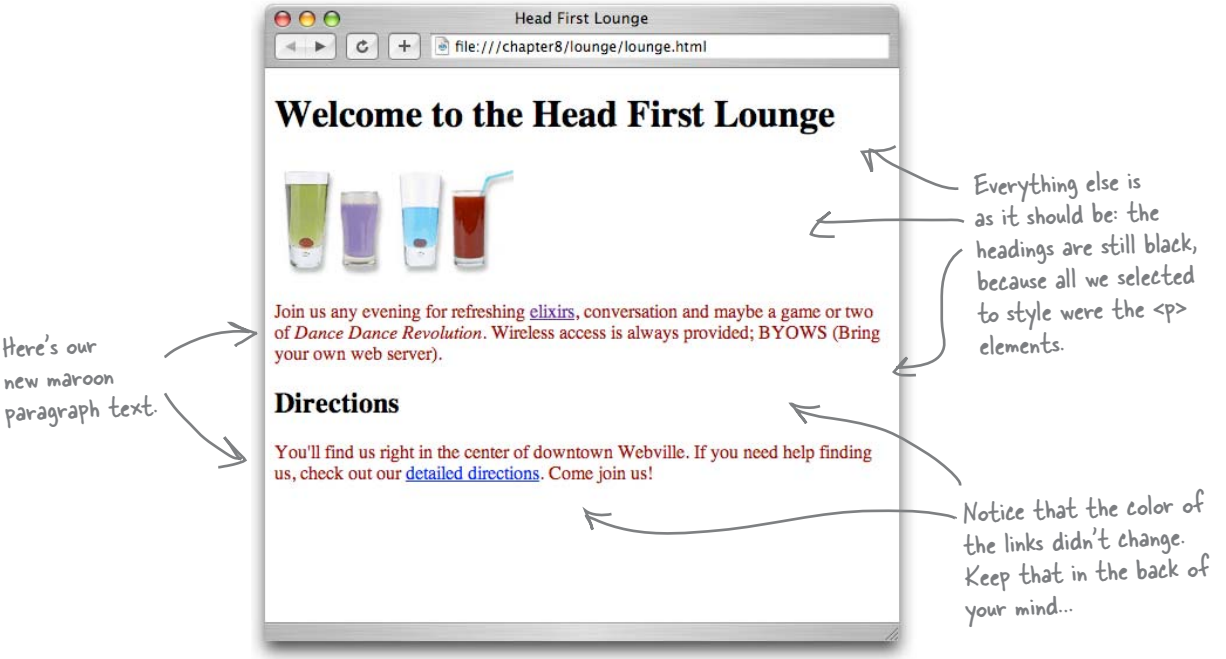
The property to change the font color is named "color" (you might think it would be "font-color" or "text-color", but it's not).

We're setting the text to a lovely maroon color that happens to match the lounge couches.

The p selector selects all the paragraphs in the XHTML.

Cruising with style: the test drive

Go ahead and make all the changes from the last couple of pages to your "lounge.html" file in the "chapter8/lounge" folder, save, and reload the page in your browser. You'll see that the paragraph text color has changed to maroon:



BRAIN POWER

Instead of setting the **color**, what if you set **background-color** of the **<p>** elements to **maroon** instead? How would it change the way the browser displays the page?

Style the heading

Now let's give those headings some style. How about changing the font a bit? Let's change both the type of font, and also the color of the heading fonts:


```
h1 {
     font-family: sans-serif;
     color:       gray;
}

h2 {
     font-family: sans-serif;
     color:       gray;
}

p {
     color: maroon;
}
```

Here's the rule to select <h1> elements and change the font-family to sans-serif and the font color to gray. We'll talk a lot more about fonts later.

And here's another rule to do the exact same thing to the <h2> element.

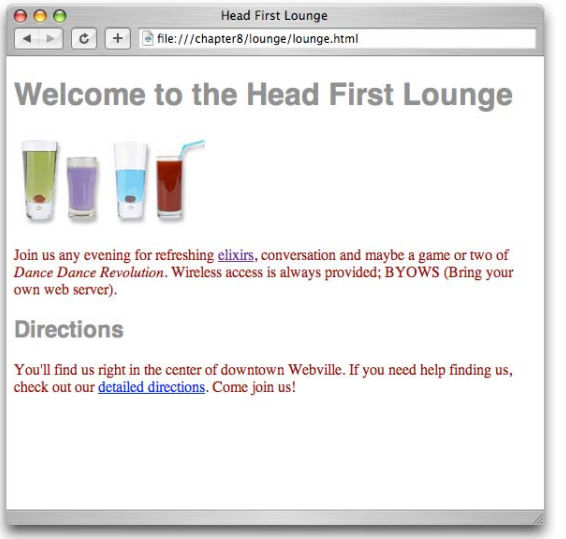
Actually, since these rules are *exactly* the same, we can combine them, like this:

```
h1, h2 {
     font-family: sans-serif;
     color:       gray;
}

p {
     color: maroon;
}
```

To write a rule for more than one element, just put commas between the selectors, like "h1, h2".

Test drive...

Add this new CSS to your "lounge.html" file and reload. You'll see that with one rule, you've selected both the **<h1>** and **<h2>** headings.



Both of the headings on the page are now styled with a sans-serif font and colored gray.

Let's put a line under the welcome message too

Let's touch up the welcome heading a bit more. How about a line under it? That should set the main heading apart visually and add a nice touch. Here's the property we'll use to do that:

```
border-bottom: 1px solid black;
```

This property controls how the border under an element looks.

We're going to style the bottom border so that it is a 1 pixel thick, solid black line.

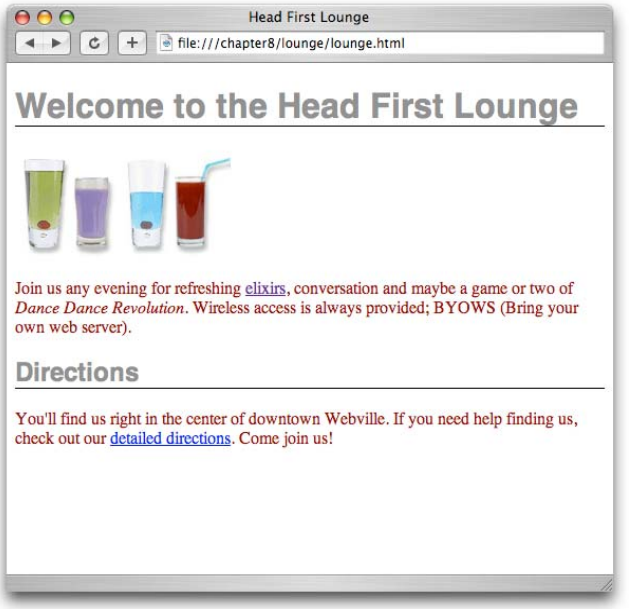
The trouble is, if we add this property and value to the combined "h1, h2" rule in our CSS, we'll end up with borders on both our headings:

```
h1, h2 {
     font-family:   sans-serif;
     color:         gray;
     border-bottom: 1px solid black;
}

p {
     color: maroon;
}
```

Here we're adding a property to change the bottom border for both the <h1> and <h2> elements.

If we do this...

... we get bottom borders on both our headings. Not what we want.



So, how can we set the bottom border on *just* the **<h1>** element, without affecting the **<h2>** element? Do we have to split up the rules again? Turn the page to find out...

We have the technology: specifying a second rule, just for the <h1>

We don't have to split the "h1, h2" rule up, we just need to add another rule that is only for "h1" and add the border style to it.

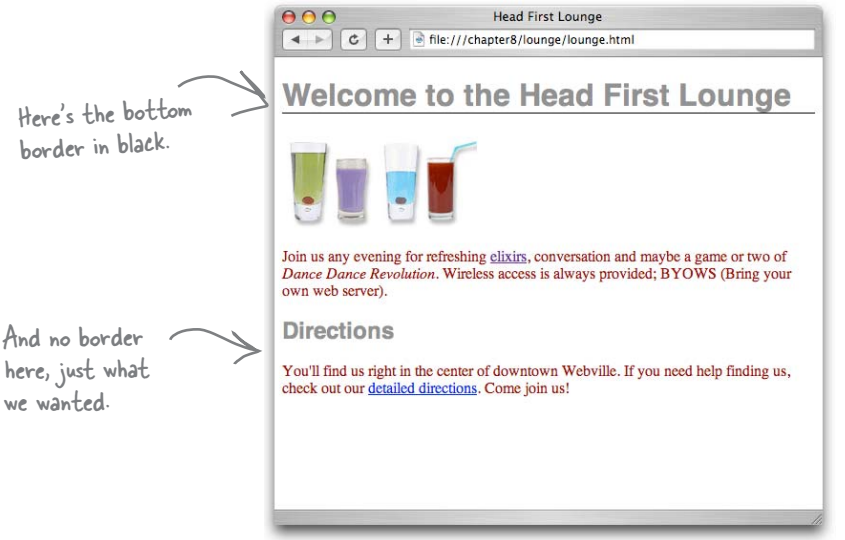
```
h1, h2 {
     font-family:    sans-serif;
     color:          gray;
}

h1 {
     border-bottom: 1px solid black;
}

p {
     color: maroon;
}
```

The first rule stays the same. We're still going to use a combined rule for the font-family and color for both <h1> and <h2>.

But now we're adding a second rule that adds another property just to <h1>: the border-bottom property.

Another test drive...

Change your CSS and reload the page. You'll see that the new rule added a black border to the bottom of the main heading, which gives us a nice underline on the heading and really makes it stand out.



there are no Dumb Questions

Q: So how does that work when you have more than one rule for an element?

A: You can have as many rules as you want for an element. Each rule adds to the style information of the rule before it. In general, you try to group together all the common styles between elements, like we did with <h1> and <h2>, and then any style that is specific to an element, you write in another rule, like we did with the border-bottom style for the main heading.

Q: What's the advantage of that approach? Isn't it better to organize each element separately, so you know exactly what styles it has?

A: Not at all. If you combine common styles together, then if they change, you only have to change them in one rule. If you break them up, then there are many rules you have to change, which is error-prone.

Q: Why do we use a bottom border to underline text? Isn't there an underline style for text?

A: Good question. There is an underline style for text and we could use that instead. However, the two styles have slightly different effects on the page: if you use border-bottom then the line will extend to the edge of the page. An underline is only shown under the text itself. The property to set text underline is called text-decoration and has a value of "underline" for underlined text. Give it a try and check out the differences.

So, how do selections really work?

You've seen how to select an element to style it, like this:

We call this the selector.

```
h1 {
      color: gray;
}
```

The style is applied to the elements described by the selector – in this case, <h1> elements.

Or, how to select more than one element, like this:

Another selector. The style is applied to <h1> and <h2> elements.

```
h1, h2 {
      color: gray;
}
```

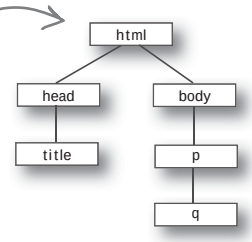
You're going to see that CSS allows you to specify all kinds of selectors that determine which elements your styles are applied to. Knowing how to use these selectors is the first step in mastering CSS, and to do that you need to understand the organization of the XHTML that you're styling. After all, how can you select elements for styling if you don't have a good mental picture of what elements are in the XHTML, and how they relate to one another?

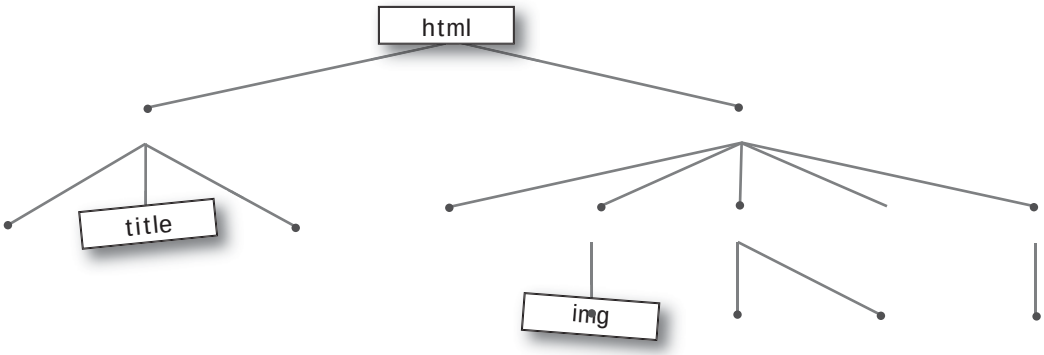
So, let's get that picture of the Lounge XHTML in your head, and then we'll dive back into selectors.

Markup Magnets

Remember drawing the diagram of HTML elements in Chapter 3? You're going to do that again for the Lounge's main page. Below, you'll find all the element magnets you need to complete the diagram. Using the Lounge's XHTML (on the right), complete the tree below. We've done a couple for you already. You'll find the answer in the back of the chapter.

Like this.





h1 body p a p p h2 meta head em a style

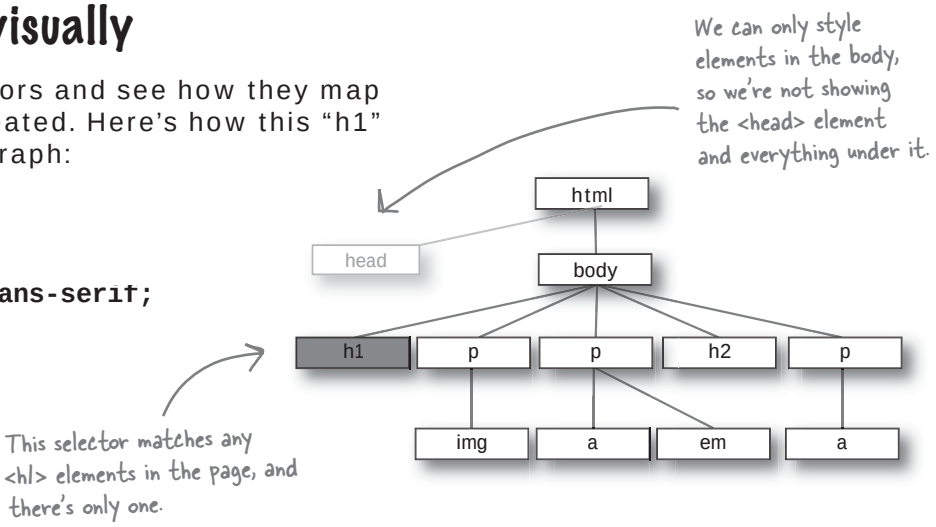
The Head First Lounge XHTML.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <title>Head First Lounge</title>

    <style type="text/css">
       h1, h2 {
              font-family:   sans-serif;
              color:         gray;
       }

       h1 {
              border-bottom: 1px solid black;
       }

       p {
              color: maroon;
       }
    </style>

  </head>
  <body>
    <h1>Welcome to the Head First Lounge</h1>
    <p>
       <img src="images/drinks.gif" alt="Drinks" />
    </p>
    <p>
       Join us any evening for refreshing
       <a href="beverages/elixir.html">elixirs</a>,
       conversation and maybe a game or two
       of <em>Dance Dance Revolution</em>.
       Wireless access is always provided;
       BYOWS (Bring your own web server).
    </p>
    <h2>Directions</h2>
    <p>
       You'll find us right in the center of downtown
       Webville. If you need help finding us, check out our
       <a href="about/directions.html">detailed directions</a>.
       Come join us!
    </p>
  </body>
</html>
```

Seeing selectors visually

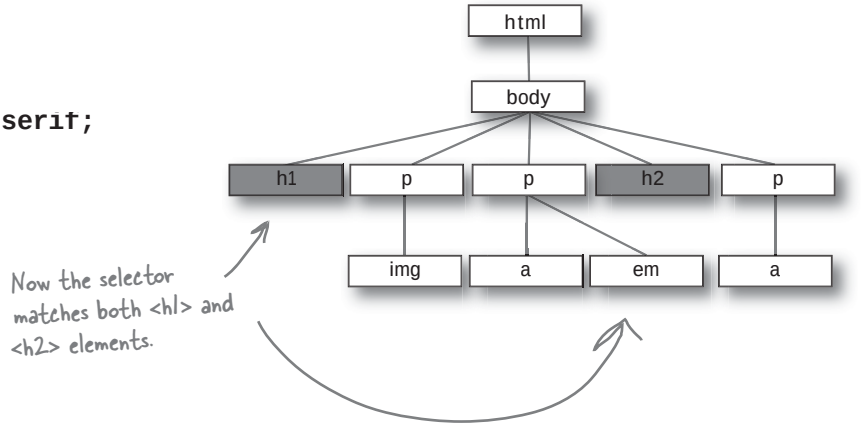
Let's take some selectors and see how they map to the tree you just created. Here's how this "h1" selector maps to the graph:

```
h1 {
    font-family: sans-serif;
}
```



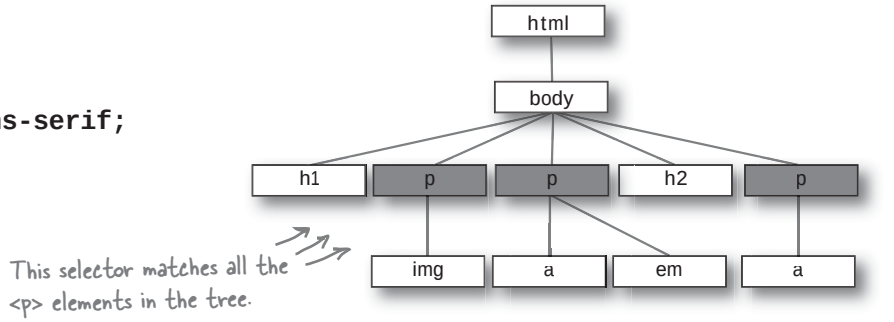
And here's how the "h1, h2" selector looks:

```
h1, h2 {
    font-family: sans-serif;
}
```



If we use a "p" selector, here's how that looks:

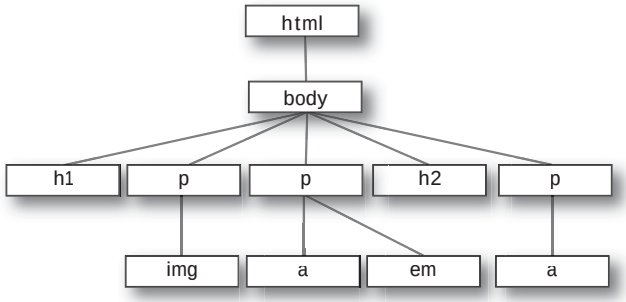
```
p {
    font-family: sans-serif;
}
```



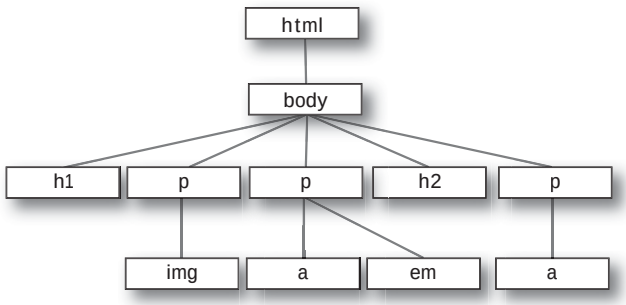
Sharpen your pencil

Color in the elements that are **selected** by these selectors:

```
p, h2 {
    font-family: sans-serif;
}
```




```
p, em {
    font-family: sans-serif;
}
```



The Case of Brute Force versus Style

When we last left RadWebDesign in Chapter 4, they had just blown the corporate demo and lost RobotsRUs' business. CorrectWebDesign was put in charge of the entire RobotsRUs site and got to work getting everything nailed down before the site launch later in the month. But, you'll also remember that RadWebDesign decided to bone up on their XHTML & CSS. They decided to rework the RobotsRUs site on their own, using strict XHTML and style sheets, just to get some experience under their belt before they took on another consulting job.

Five-Minute Mystery

As fate would have it, just before RobotsRUs' big site launch, it happened again: RobotsRUs called CorrectWebDesign with an urgent message. "We're changing our corporate look and we need all the colors, backgrounds, and fonts changed on our site." At this point, the site consisted of almost a hundred pages, so CorrectWebDesign responded that it would take them a few days to rework the site. "We don't have a few days!" the CEO said. Desperate, the CEO decided to call in RadWebDesign for help. "You flubbed up the demo last month, but we really need your help. Can you help the CorrectWebDesign guys convert the site over to the new look and feel?" RadWebDesign said they could do better than that; in fact they could deliver the entire site to them in less than an hour.

How did RadWebDesign go from disgrace to Web page superheroes? What allowed them to change the look and feel of a hundred pages faster than a speeding bullet?

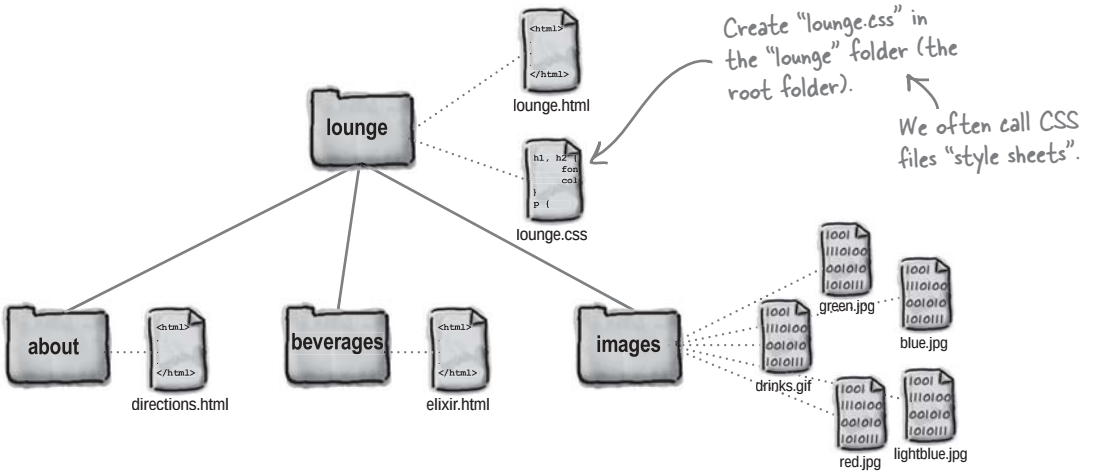
Getting the Lounge style into the elixirs and directions pages

It's great that we've added all this style to "lounge.html", but what about "elixir.html" and "directions.html"? They need to have a look that is consistent with the main page. Easy enough... just copy the style element and all the rules into each file, right? **Not so fast.** If you did that, then whenever you needed to change the style of the site, you'd have to change *every single file* – not what you want. But, luckily, there is a better way. Here's what you're going to do:

1. Take the rules in "lounge.html" and place them in a file called "lounge.css".
2. Create an *external link* to this file from your "lounge.html" file.
3. Create the same external links in "elixir.html" and "directions.html".
4. Give all three files a good test drive.

Creating the "lounge.css" file

You're going to create a file called "lounge.css" to contain the style rules for all your Head First Lounge pages. To do that, create a new text file named "lounge.css" in your text editor.



Now type, or copy and paste from your "lounge.html" file, the CSS rules into the "lounge.css" file. Delete the rules from your "lounge.html" file while you're at it.

Note that you should *not* copy the **<style>** and **</style>** tags because the "lounge.css" file contains only CSS, not XHTML.

```
h1, h2 {
   font-family: sans-serif;
   color: gray;
}

h1 {
   border-bottom: 1px solid black;
}

p {
   color: maroon;
}
```

Your "lounge.css" file should look like this. Remember, no <style> tags!

Linking from "lounge.html" to the external style

Now we need a way to tell the browser that it should style this page with the styles in the external style sheet. We can do that with an XHTML element called **<link>**. Here's how you use the **<link>** element in your XHTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
  <head>
    <meta http-equiv="Content-Type"
          content="text/html; charset=ISO-8859-1" />
    <title>Head First Lounge</title>
    <link type="text/css" rel="stylesheet" href="lounge.css" />
    <style type="text/css">
    </style>

  </head>
  <body>
    <h1>Welcome to the Head First Lounge</h1>
    <p>
       <img src="drinks.gif" alt="Drinks" />
    </p>
    .
    .
    .
    </p>
  </body>
</html>
```

Here's the XHTML that links to the external style sheet.

You don't need the <style> element any more – just delete it.

The rest of the XHTML is the same.

XHTML Up Close

Let's take a closer look at the **<link>** element since you haven't seen it before:

Use the link element to "link in" external information.

The type of this information is "text/css". In other words, a CSS style sheet.

And the style sheet is located at this href (in this case we're using a relative link, but it could be a full-blown URL).

```
<link type="text/css" rel="stylesheet" href="lounge.css" />
```

The rel attribute specifies the relationship between the XHTML file and the thing you're linking to. We're linking to a style sheet, so we use the value "stylesheet".

<link> is an empty element.

Linking from "elixir.html" and "directions.html" to the external style sheet

Now you're going to link the "elixir.html" and "directions.html" files just as you did with "lounge.html". The only thing you need to remember is that "elixir.html" is in the "beverages" folder, and "directions.html" is in the "about" folder, so they both need to use the relative path "../lounge.css".

So, all you need to do is add the following **<link>** element to both files:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <title>Head First Lounge Elixirs</title>
    <link type="text/css" rel="stylesheet" href="../lounge.css" />
  </head>
  <body>
  .
  .
  .
  </body>
</html>
```

This is "elixir.html". Just add the <link> line.

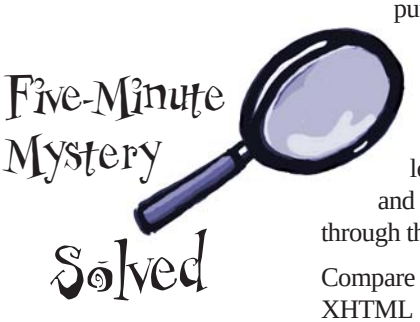
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <title>Head First Lounge Directions</title>
    <link type="text/css" rel="stylesheet" href="../lounge.css" />
  </head>
  <body>
  .
  .
  .
  </body>
</html>
```

Same for "directions.html". Add the <link> line here.

Test driving the entire lounge...

Save each of these files and then open "lounge.html" with the browser. You should see no changes in its style, even though the styles are now coming from an external file. Now click on the "elixirs" and "detailed directions" links.

Wow! We have a whole new style for the Elixirs and Directions pages with only a *one line change* to the HTML in each file! Now you can really see the power of CSS.



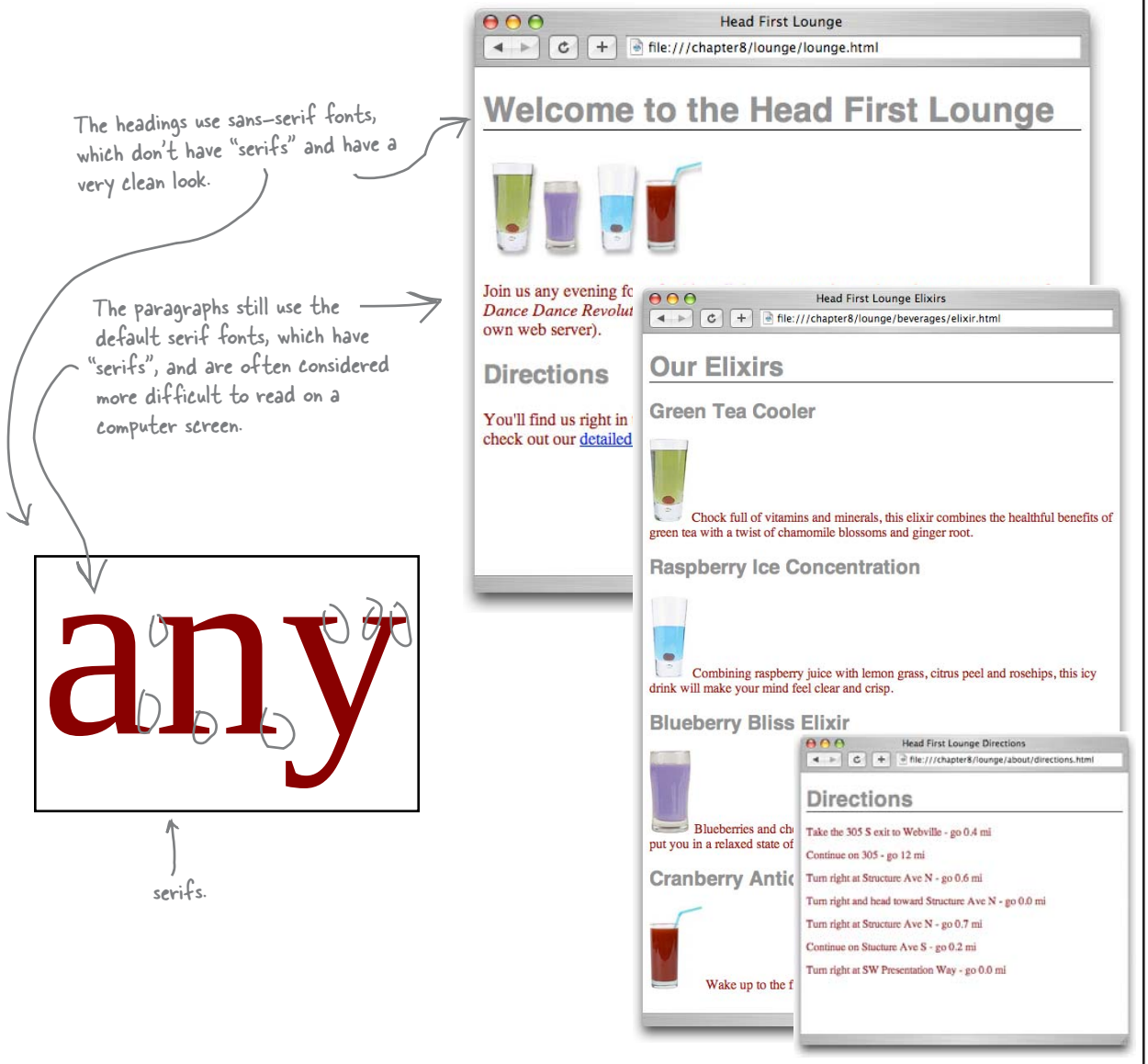
The Case of Brute Force versus Style

So, how did RadWebDesign become Web page superheroes? Or, maybe we should first ask how the "do no wrong" CorrectWebDesign firm flubbed things up this time? The root of the problem was that CorrectWebDesign was creating the RobotsRUs pages using circa 1998 techniques. They were putting their style rules right in with their HTML (copying and pasting them each time), and, even worse, they were using a lot of old HTML elements like **<font>** and ** ** that have now been deprecated. So, when the call came to change the look and feel, that meant going into *every* Web page and making changes to the CSS. Worse, it meant going through the HTML to change elements as well.

Compare that with what RadWebDesign did: they used strict XHTML 1.0, so they had no old presentation HTML in their pages, and they used an external style sheet. The result? To change the style of the entire site, all they had to do was go into their external style sheet and make a few changes to the CSS, which they easily did in minutes, not days. They even had time to try out multiple designs and have three different versions of the CSS ready for review before the site launch. Amazed, the RobotsRUs CEO not only promised RadWebDesign more business, but he also promised them the first robot that comes off the assembly line.

Sharpen your pencil

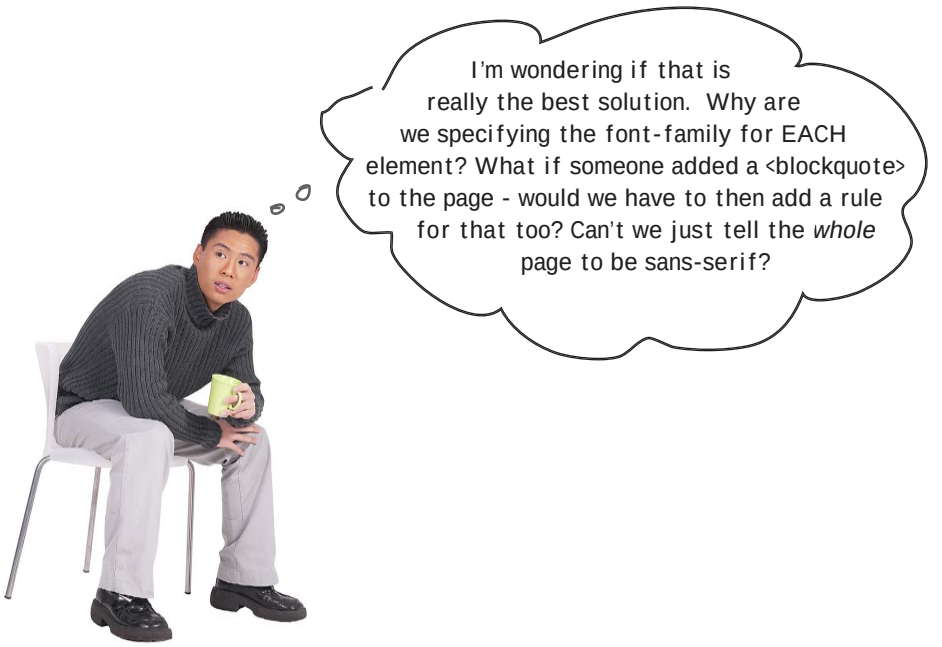
Now that you've got one external style file (or "style sheet"), use it to change all the paragraph fonts to "sans-serif" to match the headings. Remember, the property to change the font style is "font-family", and the value for sans-serif font is "sans-serif". You'll find the answer on the next page.



Sharpen your pencil
Solution

```
h1, h2 {
       font-family:  sans-serif;
       color:        gray;
}

h1 {
       border-bottom: 1px solid black;
}

p {
       font-family:  sans-serif;
       color:        maroon;
}
```

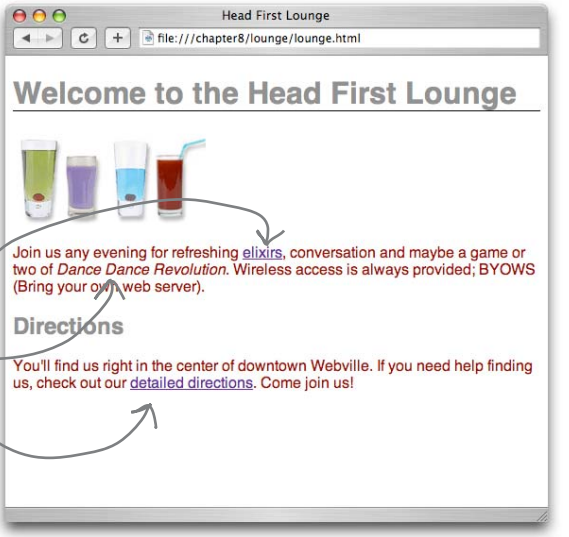
Just add a font-family property to your paragraph rule in the "lounge.css" file.



It's time to talk about your inheritance...

Did you notice when you added the **`font-family`** property to your "p" selector that it also affected the font family of the elements inside the **<p>** element? Let's take a closer look:

When you added the font-family property to your CSS p selector, it changed the font family of your <p> elements. But it also changed the font family of the two links and the emphasized text.



The elements inside the <p> element inherit the font-family style from <p>

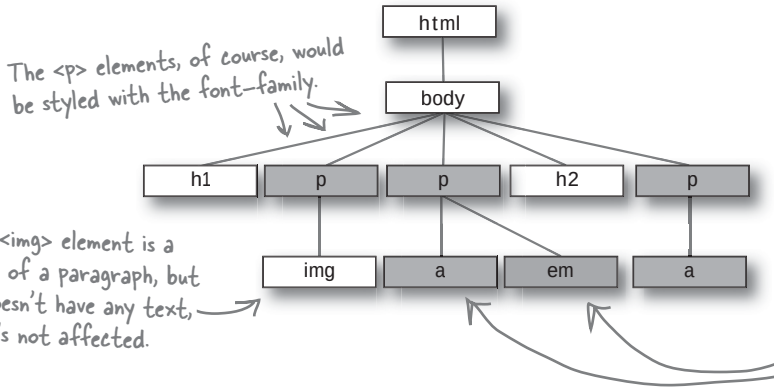
Not every style is inherited. Just some are, like font-family.

Just like you can inherit your blue eyes or brown hair from your parents, elements can inherit styles from their parents. In this case, the **<a>** and **** elements inherited the **`font-family`** style from the **<p>** element, which is their parent element. It makes sense that changing your paragraph style would change the style of the elements in the paragraph, doesn't it? After all, if it didn't, you'd have to go in and add CSS rules for every inline element in every paragraph in your whole site... which would definitely be so NOT fun.

Not to mention, error-prone, tedious, and time-consuming.

Let's take a look at our XHTML tree to see how inheritance works:

If we set the font-family of all the <p> elements, here are all the elements that would be affected.



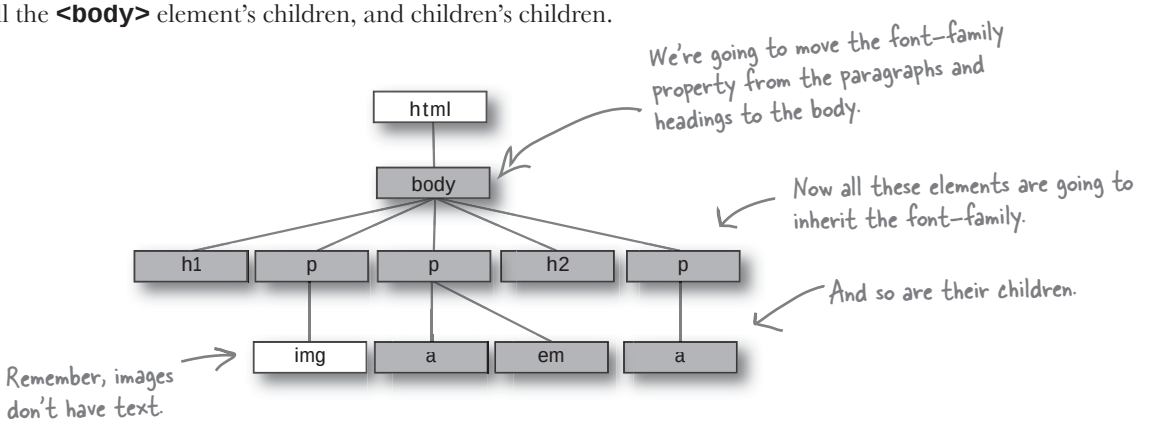
The <p> elements, of course, would be styled with the font-family.

The <a>, , and <a> elements in the two paragraphs inherit the font-family from their parent elements, the <p> elements.

The <img> element is a child of a paragraph, but it doesn't have any text, so it's not affected.

What if we move the font up the family tree?

If most elements inherit the **font-family** property, what if we move it up to the **<body>** element? That should have the effect of changing the font for all the **<body>** element's children, and children's children.



Wow, this is powerful. Simply by changing the font-family property in the body rule, we could change the font for an entire site.

What are you waiting for... give it a try

Open your "lounge.css" file and add a new rule that selects the **<body>** element. Then remove the **font-family** properties from the headings and paragraph rules, because you're not going to need them anymore.

Here's what you're going to do.

```
body {
       font-family:  sans-serif;
}

h1, h2 {
       font-family:  sans-serif;
       color:        gray;
}

h1 {
       border-bottom: 1px solid black;
}

p {
       font-family:  sans-serif;
       color:        maroon;
}
```

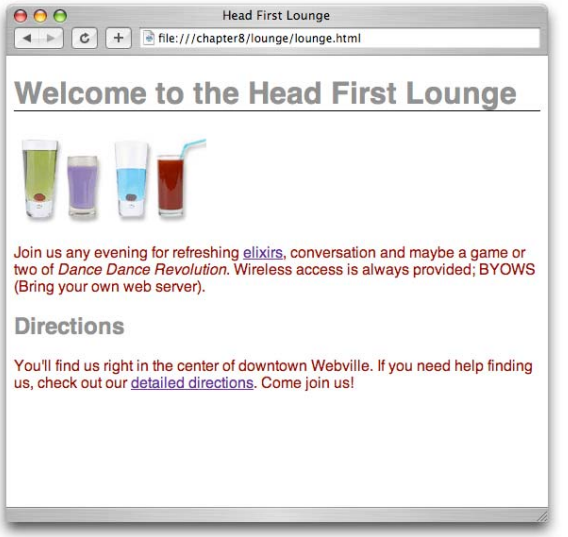
First, add a new rule that selects the <body> element. Then add the font-family property with a value of sans-serif.

Then, take the font-family property out of the h1, h2 rule, as well as the p rule.

Test drive your new CSS

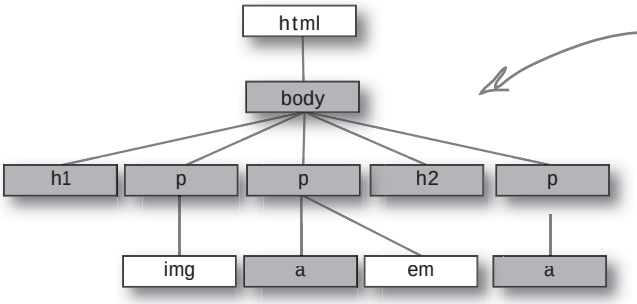
As usual, go ahead and make these changes in the "lounge.css" style sheet, save, and reload the "lounge.html" page. You shouldn't expect any changes, because the style is the same. It's just coming from a different rule. But you should feel better about your CSS because now you can add new elements to your pages and they'll automatically inherit the sans-serif font.

Surprise, surprise. This doesn't look any different at all, but that is exactly what we were expecting, isn't it? All you've done is move the sans-serif font up into the body rule and let all the other elements inherit that.



Overriding inheritance

By moving the **font-family** property up into the body, you've set that font style for the entire page. But what if you don't want the sans-serif font on every element? For instance, you could decide that you want **** elements to use the serif font instead.



The font-family property is set in the body rule, so every element inside the body inherits the sans-serif font-family property from <body>.

But you've decided you want your elements to have the serif font-family instead. You need to override the inheritance with a CSS rule.

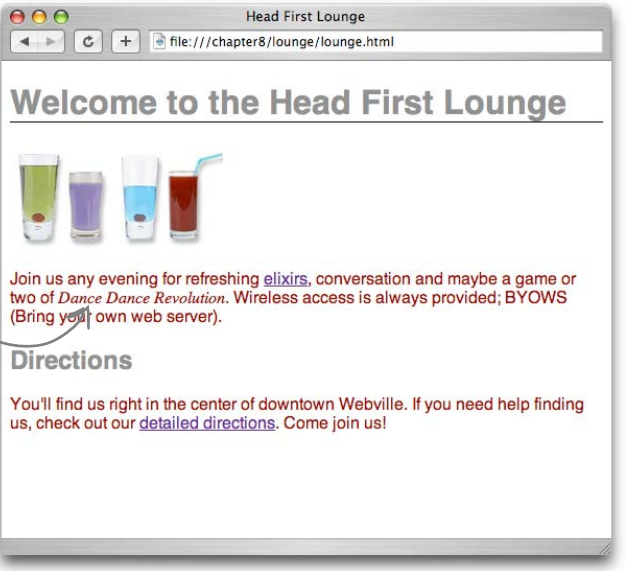
Well, then you can override the inheritance by supplying a specific rule just for ****. Here's how you add a rule for **** to override the font-family specified in the body:

```
body {
      font-family:  sans-serif;
}

h1, h2 {
      color:        gray;
}

h1 {
      border-bottom: 1px solid black;
}

p {
      color:        maroon;
}

em {
      font-family: serif;
}
```

To override the font-family property inherited from body, add a new rule selecting em with the font-family property value set to serif.

Test drive

Add a rule for the **** element to your CSS with a **font-family** property value of **serif**, and reload your "lounge.html" page:



Notice that the "Dance Dance Revolution" text, which is the text in the element, is now a serif font.

As a general rule, it's not a good idea to change fonts in the middle of a paragraph like this, so go ahead and change your CSS back to the way it was (without the em rule) when you're done testing.

there are no Dumb Questions

Q: How does the browser know which rule to apply to when I'm overriding the inherited value?

A: With CSS, the most specific rule is always used. So, if you have a rule for <body>, and a more specific rule for elements, it is going to use the more specific rule. We'll talk more later about how you know which rules are most specific.

Q: How do I know which CSS properties are inherited and which are not?

A: This is where a good reference really comes in handy, like O'Reilly's *CSS Pocket Reference*. In general, all of the styles that affect the way your text looks, such as font color (the color property), the font-family, as you've just seen, and other font related properties such as font-size, font-weight (for bold text), and font-style (for italics) are inherited. Other properties, such as border, are not inherited, which makes sense, right? Just because you want a border on your <body> element doesn't mean you want it on *all* your elements. A lot of the time you can follow your common sense (or just try it and see), and you'll get the hang of it as you become more familiar with the various properties and what they do.

Q: Can I always override a property that is being inherited when I don't want it?

A: Yes. You can always use a more specific selector to override a property from a parent.

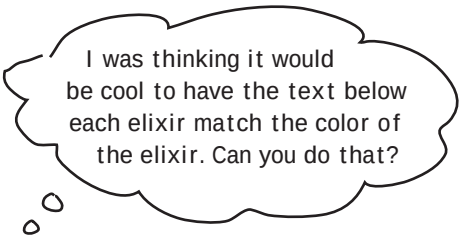
Q: This stuff gets complicated. Is there any way I can add comments to remind myself what the rules do?

A: Yes. To write a comment in your CSS just enclose it between /* and */. For instance:

```
/*  this rule selects all paragraphs and colors them blue */
```

Notice that a comment can span multiple lines. You can also put comments around CSS and browsers will ignore it, like:

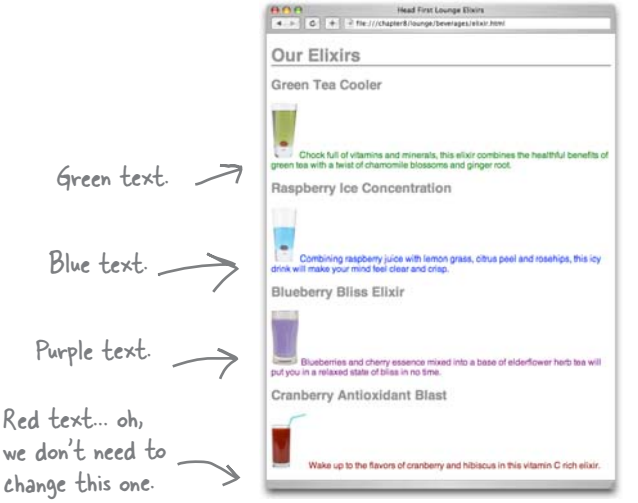
```
/* this rule will have no effect because it's in a comment

p { color: blue; }  */
```



We're not sure we agree with the aesthetics of that suggestion, but, hey, you're the customer.

Can you style each of these paragraphs separately so that the color of the text matches the drink? The problem is that using a rule with a "p" selector applies the style to *all* **<p>** elements. So, how can you select these paragraphs individually?

That's where *classes* come in. Using both XHTML and CSS, we can define a class of elements, and then apply styles to any element that belongs to that class. So, what exactly is a class? Think of it like a club – someone starts a "greentea" club, and by joining you agree to all the rights and responsibilities of the club, like adhering to their style standards. Anyway, let's just create the class and you'll see how it works.



Adding a class to "elixir.html"

Open up the "elixir.html" file and locate the "Green Tea Cooler" paragraph. This is the text we want to change to green. All you're going to do is add the **<p>** element to a class called **greentea**. Here's how you do that:

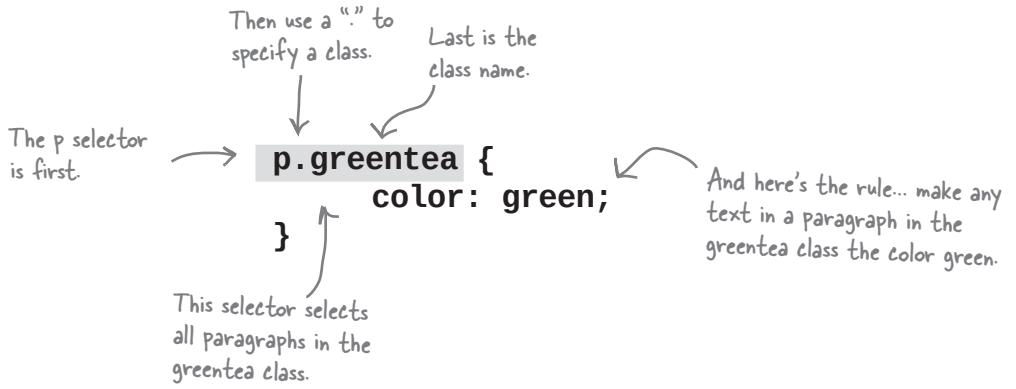
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <title>Head First Lounge Elixirs</title>
    <link type="text/css" rel="stylesheet" href="../lounge.css" />
  </head>
  <body>
    <h1>Our Elixirs</h1>
    <h2>Green Tea Cooler</h2>
    <p class="greentea">
            <img src="../images/green.jpg" />
            Chock full of vitamins and minerals, this elixir
            combines the healthful benefits of green tea with
            a twist of chamomile blossoms and ginger root.
    </p>
    <h2>Raspberry Ice Concentration</h2>
    <p>
            <img src="../images/lightblue.jpg" />
            Combining raspberry juice with lemon grass,
            citrus peel and rosehips, this icy drink
            will make your mind feel clear and crisp.
    </p>
    <h2>Blueberry Bliss Elixir</h2>
    <p>
            <img src="../images/blue.jpg" />
            Blueberries and cherry essence mixed into a base
            of elderflower herb tea will put you in a relaxed
            state of bliss in no time.
    </p>
    <h2>Cranberry Antioxidant Blast</h2>
    <p>
            <img src="../images/red.jpg" />
            Wake up to the flavors of cranberry and hibiscus
            in this vitamin C rich elixir.
    </p>
  </body>
</html>
```

To add an element to a class, just add the attribute "class" along with the name of the class, like "greentea".

And, now that the green tea paragraph belongs to the **greentea** class, you just need to provide some rules to style that class of elements.

Creating a selector for the class

To select a class, you write the selector like this:



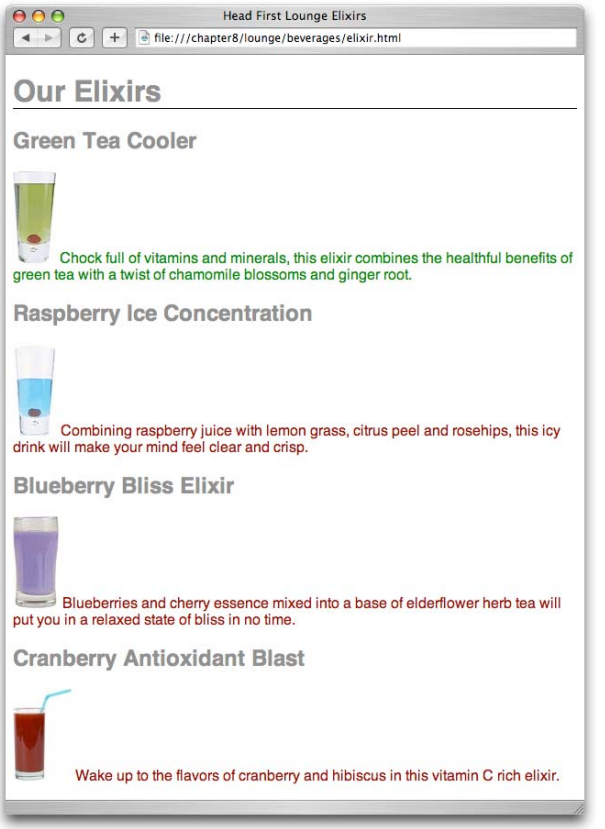
So now you have a way of selecting **<p>** elements that belong to a certain class. All you need to do is add the **class** attribute to any **<p>** elements you want to be green, and this rule will be applied. Give it a try: open your "lounge.css" file and add the **p.greentea** class selector to it.

```
body {
     font-family: sans-serif;
}

h1, h2 {
     color: gray;
}

h1 {
     border-bottom: 1px solid black;
}

p {
     color: maroon;
}

p.greentea {
     color: green;
}
```

A greentea test drive

Save, and then reload to give your new class a test drive.

Here's the new greentea class applied to the paragraph. Now the font is green and matches the Green Tea Cooler. Maybe this styling wasn't such a bad idea after all.



Sharpen your pencil

Your turn: add two classes, "raspberry" and "blueberry", to the correct paragraphs in "elixir.html", and then write the styles to color the text blue and purple, respectively. The property value for raspberry is "blue" and for blueberry is "purple". Put these at the bottom of your CSS file, under the greentea rule: raspberry first, and then blueberry.

Yeah, we know you're probably thinking, how can a raspberry be blue? Well, if Raspberry Kool-aid is blue, that's good enough for us. And seriously, when you blend up a bunch of blueberries, they really are more purple than blue. Work with us here.

Taking classes further...

You've already written one rule that uses the **greentea** class to change any paragraph in the class to the color "green":

```
p.greentea {
       color: green;
}
```

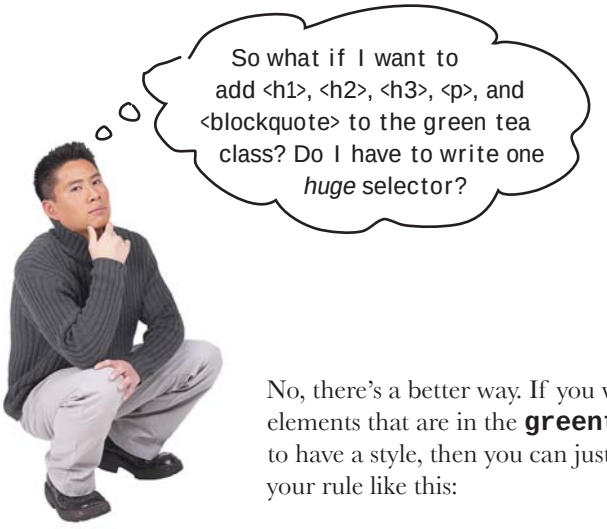
But what if you wanted to do the same to all **<blockquote>**s? Then you could do this:

```
blockquote.greentea, p.greentea {
       color: green;
}
```

Just add another selector to handle <blockquote>s that are in the greentea class. Now this rule will apply to <p> and <blockquote> elements in the greentea class.

And in your XHTML you'd write:

```
<blockquote class="greentea">
```



No, there's a better way. If you want all elements that are in the **greentea** class to have a style, then you can just write your rule like this:

```
.greentea {
       color: green;
}
```

If you leave out all the element names, and just use a period followed by a class name, then the rule will apply to all members of the class.

Cool! Yes, that works. One more question... you said being in a class is like being in a club. Well, I can join many clubs. So, can an element be in more than one class?

Yes, elements can be in more than one class.

It's easy to put an element into more than one class. Say you want to specify a **<p>** element that is in the **greentea**, **raspberry**, and **blueberry** classes. Here's how you would do that in the opening tag:

```
<p class="greentea raspberry blueberry">
```

Place each class name into the value of the class attribute, with a space in between each. The ordering doesn't matter.

So, for example, I could put an <h1> into my "products" class that defines a font size and weight, and also a "specials" class to change its color to red when something's on sale?

Exactly. Use multiple classes when you want an element to have styles you've defined in different classes. In this case, all your **<h1>** elements associated with products have a certain style, but not all your products are on sale at the same time. By putting your "specials" color in a separate class, you can simply add only those elements associated with products on sale to the "specials" class to add the red color you want.

Now you may be wondering what happens when an element belongs to multiple classes, all of which define the *same* property – like our **<p>** element up there. How do you know which style gets applied? You know each of these classes has a definition for the **color** property. So, will the paragraph be green, blue (raspberry), or purple?

We're going to talk about this in great detail after you've learned a bit more CSS, but on the next page you'll find a quick guide to hold you over.

The world's smallest & fastest guide to how styles are applied

Elements and document trees and style rules and classes... it can get downright confusing. How does all this stuff come together so that you know which styles are being applied to which elements? As we said, *to fully answer that* you're going to have to know a little more about CSS, and you'll be learning that in the next few chapters. But before you get there, let's just walk through some common sense rules-of-thumb about how styles are applied.

First, do any selectors select your element?

Let's say you want to know the **font-family** property value for an element. The first thing to check is: is there a selector in your CSS file that selects your element? If there is, and it has a **font-family** property and value, then that's the value for your element.

What about inheritance?

If there are no selectors that match your element, then you rely on inheritance. So, look at the element's parents, and parents' parents, and so on, until you find the property defined. When and if you find it, that's the value.

Struck out again? Then use the default

If your element doesn't inherit the value from any of its ancestors, then you use the default value defined by the browser. In reality, this is a little more complicated than we're describing here, but we'll get to some of those details later in the book.

What if multiple selectors select an element?

Ah, this is the case we have with the paragraph that belongs to all three classes:

```
<p class="greentea raspberry blueberry">
```

There are multiple selectors that match this element and define the same **color** property. That's what we call a "conflict". Which rule breaks the tie? Well, if one rule is more *specific* than the others, then it wins. But what does more specific mean? We'll come back in a later chapter and see *exactly* how to determine how specific a selector is, but for now, let's look at some rules and get a feel for it:

```
p { color: black;}

.greentea { color: green; }

p.greentea { color: green; }

p.raspberry { color: blue; }

p.blueberry { color: purple; }
```

Here's a rule that selects any old paragraph element.

This rule selects members of the greentea class. That's a little more specific.

And this rule selects only paragraphs that are in the greentea class, so that's even more specific.

These rules also select only paragraphs in a particular class. So they are about the same in specificity as the p.greentea rule.

And if we still don't have a clear winner?

So, if you had an element that belonged only to the **greentea** class there would be an obvious winner: the **p.greentea** selector is the most specific, so the text would be green. But you have an element that belongs to *all three* classes: **greentea**, **raspberry**, and **blueberry**. So, **p.greentea**, **p.raspberry**, and **p.blueberry** all select the element, and are of equal specificity. What do you do now? You choose the one that is listed *last* in the CSS file. If you can't resolve a conflict because two selectors are equally specific, you use the ordering of the rules in your style sheet file. That is, you use the rule listed last in the CSS file (nearest the bottom). And in this case, that would be the **p.blueberry** rule.

In your "lounge.html" file, change the greentea paragraph to include all the classes, like this:

```
<p class="greentea raspberry blueberry">
```

Save, and reload. What color is the Green Tea Cooler paragraph now? _____

Next, reorder the classes in your XHTML:

```
<p class="raspberry blueberry greentea">
```

Save, and reload. What color is the Green Tea Cooler paragraph now? _____

Next, open your CSS file and move the p.greentea rule to the bottom of the file.

Save, and reload. What color is the Green Tea Cooler paragraph now? _____

Finally, move the p.raspberry rule to the bottom of the file.

Save, and reload. What color is the Green Tea Cooler paragraph now? _____

After you've finished, rewrite the green tea element to look like it did originally:

```
<p class="greentea">
```

Save, and reload. What color is the Green Tea Cooler paragraph now? _____

Tonight's talk: **CSS & XHTML compare languages**

CSS

Did you see that? I'm like Houdini! I broke right out of your **<style>** element and into my own file. And you said in Chapter 1 that I'd never escape.

Have to link me in? Come on; you know your pages wouldn't cut it without my styling.

If you were paying attention in this chapter, you would have seen I'm downright powerful in what I can do.

Well now, that's a little better. I like the new attitude.

XHTML

Don't get all excited; I still have to link you in for you to be at all useful.

Here we go again... while me and all my elements are trying to keep things structured, you're talking about hair highlights and nail color.

Okay, okay, I admit it; using CSS sure makes my job easier. All those old deprecated styling elements were a pain in my side. I do like the fact that my elements can be styled without inserting a bunch of stuff in the XHTML, other than maybe an occasional class attribute.

But I still haven't forgotten how you mocked my syntax... <remember>?

CSS

You have to admit XHTML is kinda clunky, but that's what you get when you're related to an early '90s technology.

Are you kidding? I'm very expressive. I can select just the elements I want, and then describe exactly how I want them styled. And you've only just begun to see all the cool styling I can do.

Yup; just wait and see. I can style fonts and text in all kinds of interesting ways. I can even control how each element manages the space around it on the page.

Bwahahahaa. And you thought you had me controlled between your **<style>** tags. You're going to see I can make your elements sit, bark, and rollover if I want to.

XHTML

I call it standing the test of time. And you think CSS is elegant? I mean, you're just a bunch of rules. How's that a language?

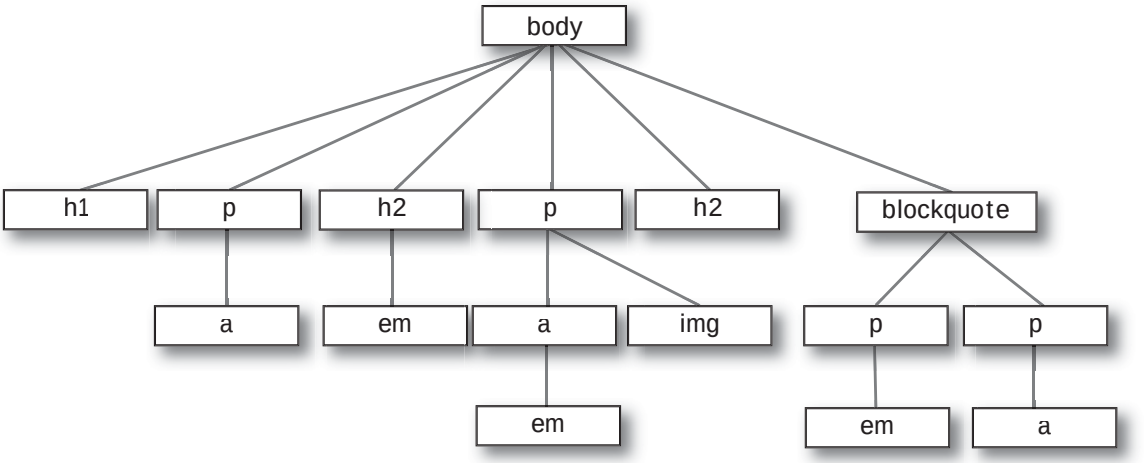
Oh yeah?

Hmmm... sounds as if you have a little too much power; I'm not sure I like the sound of that. After all, my elements want to have some control over their own lives.

Whoa now! Security... security?!

Who gets the inheritance?

Sniff, sniff; the **<body>** element has gone to that great browser in the sky. But he left behind a lot of descendants and a big inheritance of **color** "green". Below you'll find his family tree. Mark all the descendants that inherit the **<body>** element's color green. Don't forget to look at the CSS below first.




```
body {
        color: green;
}

p {
        color: black;
}
```

Here's the CSS. Use this to determine which of the above elements hit the jackpot and get the green (color).

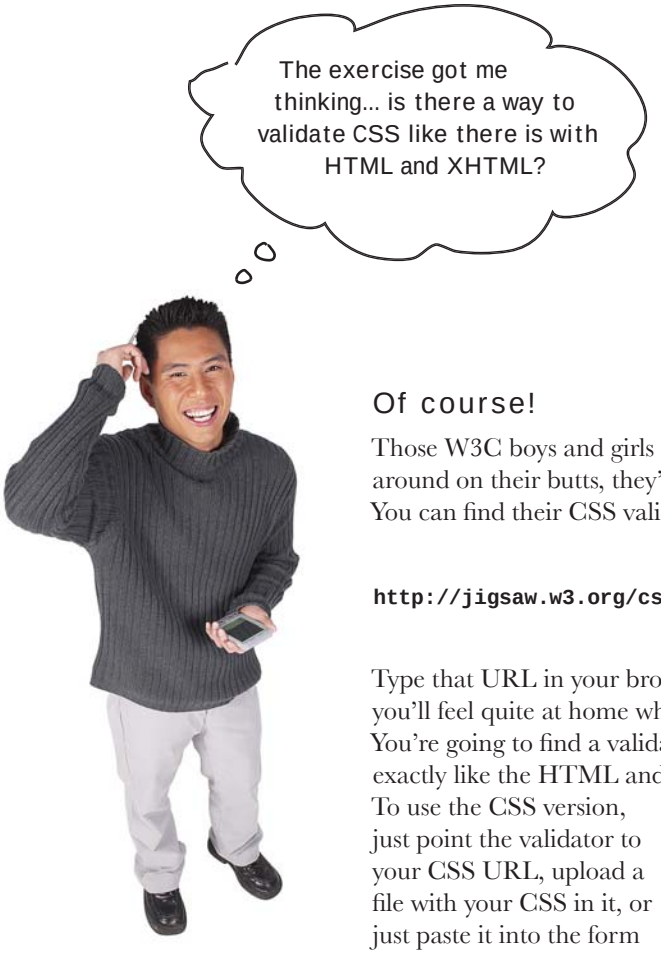
BE the Browser

Below, you'll find the CSS file "style.css", with some errors in it. Your job is to play like you're the browser and locate all the errors. After you've done the exercise look at the end of the chapter to see if you caught all the errors.

If you have errors in your CSS, usually what happens is all the rules below the error are ignored. So, get in the habit of looking for errors now, by doing this exercise.

The file "style.css"

```
<style>

body {
    background-color: white

h1, {
    gray;
    font-family: sans-serif;
}

h2, p {
    color:
}

<em> {
    font-style: italic;
}

</style>
```



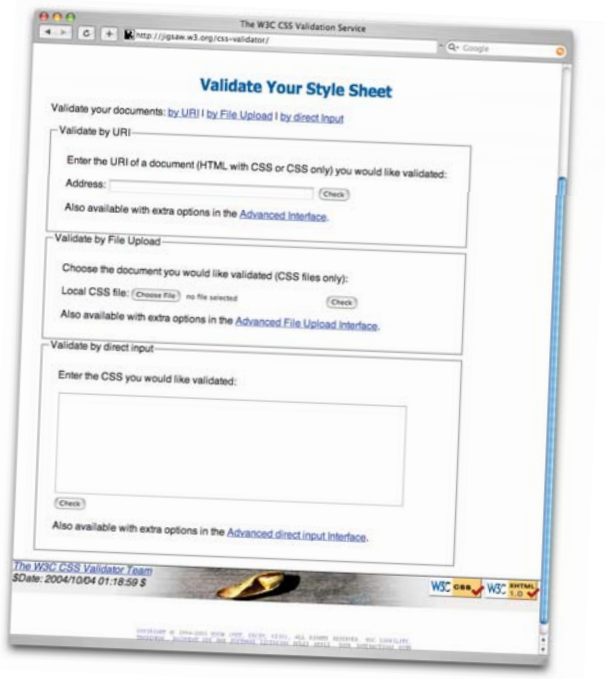
Of course!

Those W3C boys and girls aren't just sitting around on their butts, they've been working hard. You can find their CSS validator at:

http://jigsaw.w3.org/css-validator/

Type that URL in your browser and we think you'll feel quite at home when you get there. You're going to find a validator that works almost exactly like the HTML and XHTML validators. To use the CSS version, just point the validator to your CSS URL, upload a file with your CSS in it, or just paste it into the form and submit.

You shouldn't encounter any big surprises, like needing DOCTYPEs or character encodings with CSS. Go ahead, give it a try (like we're not going to make you do it on the next page, anyway).



Making sure the Lounge CSS validates

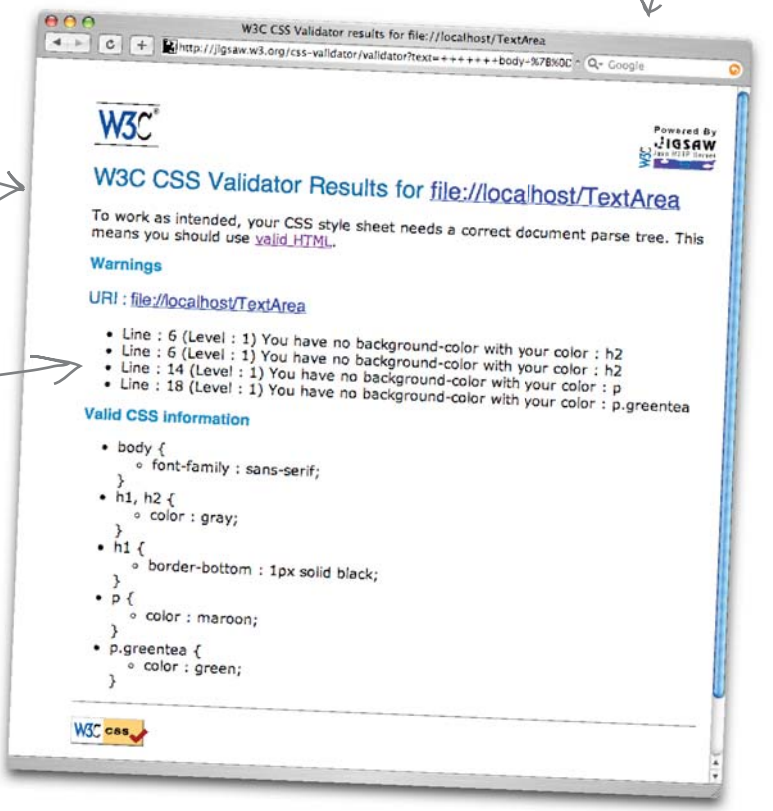
Before you wrap up this chapter, wouldn't you feel a lot better if all that Head First Lounge CSS validated? Sure you would. Use whichever method you want to get your CSS to the W3C. If you have your CSS on a server, type your URL into the form; otherwise, either upload your CSS file or just copy and paste the CSS into the form. (If you upload, make sure you're directing the form to your CSS file, not your XHTML file.) Once you've done that, click on "Check".

If your CSS didn't validate, check it with the CSS a few pages back and find any small mistakes you've made, then resubmit.

This is just telling you that the CSS needs correct XHTML to style, so make sure your XHTML (or HTML) also validates.

Here are some warnings about the CSS. These are more suggestions than real warnings. For instance, all these warnings are telling you to set a background color on the headings and paragraphs.

And here's all the valid CSS, which is ALL your CSS, so this means your CSS validates.

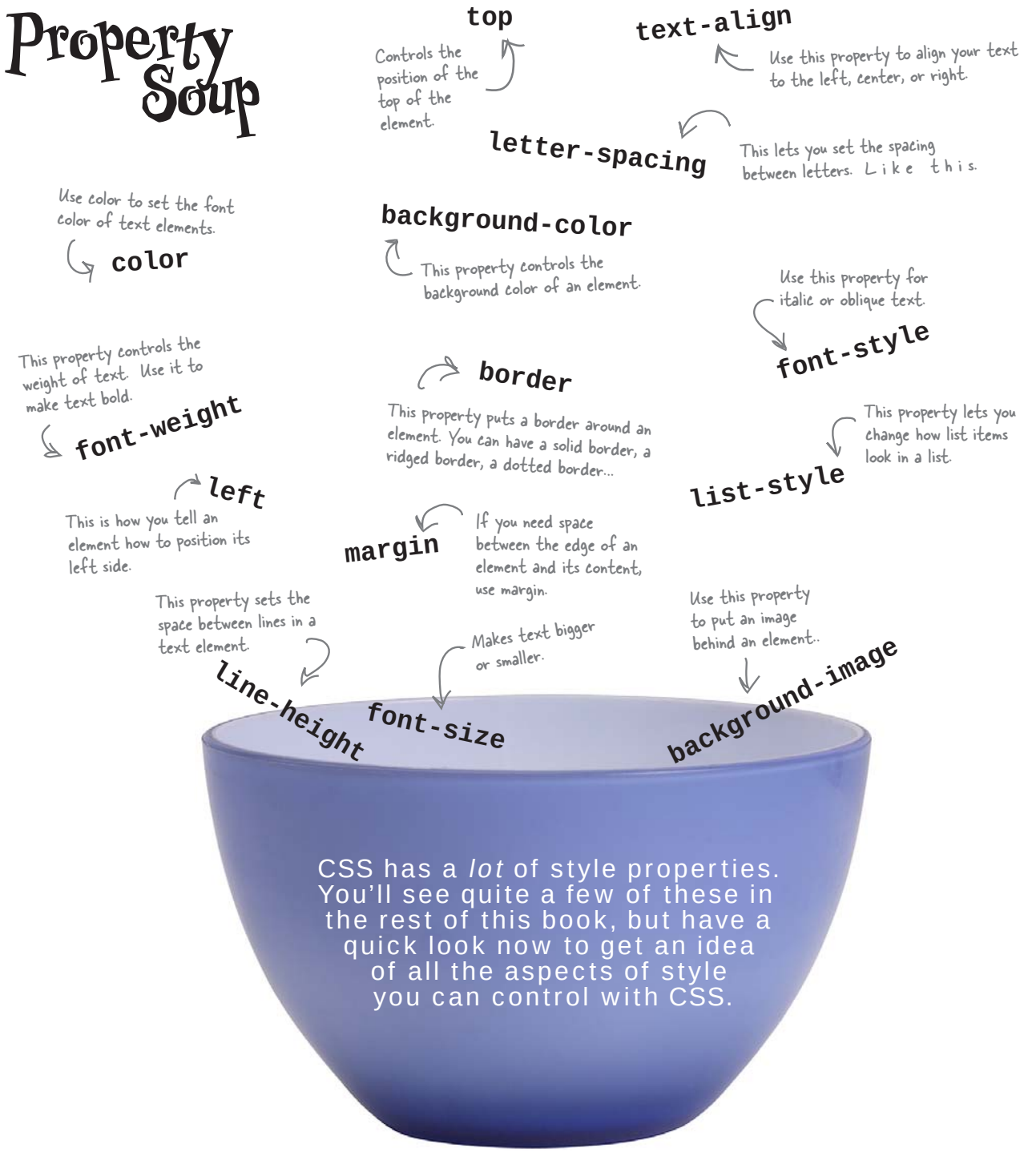


There's no "green badge of success" when you pass validation like there is when you validate XHTML. So check the top of the page for "Errors". If you don't see that, your CSS validated!

there are no Dumb Questions

Q: Do I need to worry about those warnings? Or do what they say?

A: It's good to look them over, but you'll find some are more in the category of suggestions than "must do's". The validator can err on the side of being a little anal, so just keep that in mind.

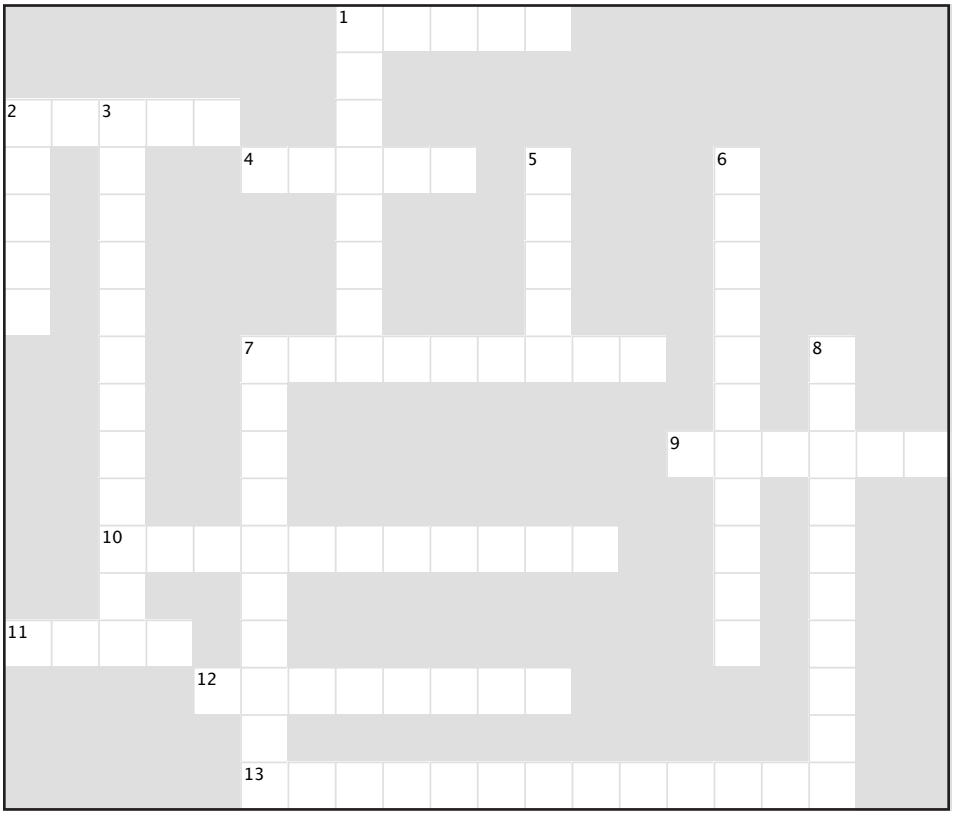


BULLET POINTS

- CSS contains simple statements, called rules.
- Each rule provides the style for a selection of XHTML elements.
- A typical rule consists of a selector along with one or more properties and values.
- The selector specifies which elements the rule applies to.
- Each property declaration ends with a semicolon.
- All properties and values in a rule go between { } braces.
- You can select any element using its name as the selector.
- By separating element names with commas, you can select multiple elements at once.
- One of the easiest ways to include a style in HTML is the <style> tag.
- For XHTML and for sites of any complexity, you should link to an external style sheet.
- The <link> element is used to include an external style sheet.
- Many properties are inherited. For instance, if a property that is inherited is set for the <body> element, all the <body>'s child elements will inherit it.
- You can always override properties that are inherited by creating a more specific rule for the element you'd like to change.
- Use the `class` attribute to add elements to a class.
- Use a "." between the element name and the class name to select a specific element in that class.
- Use ".classname" to select any elements that belong to the class.
- An element can belong to more than one class by placing multiple class names in the class attribute with spaces between the names.
- You can validate your CSS using the W3C validator, at http://jigsaw.w3.org/css-validator.

XHTMLcross

Here are some clues with mental twist and turns that will help you burn alternative routes to CSS right into your brain!



Across

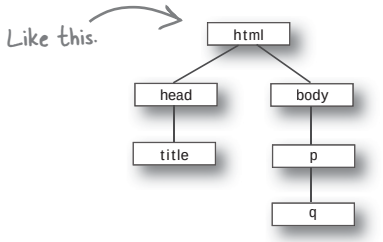
1. Defines a group of elements.
2. Ornamental part of some fonts.
4. Styles are defined in these.
7. Fonts without serifs.
9. Each rule defines a set of properties and?
10. How elements get properties from their parents.
11. Use this element to include an external style sheet.
12. Selects an element.
13. Reality show.

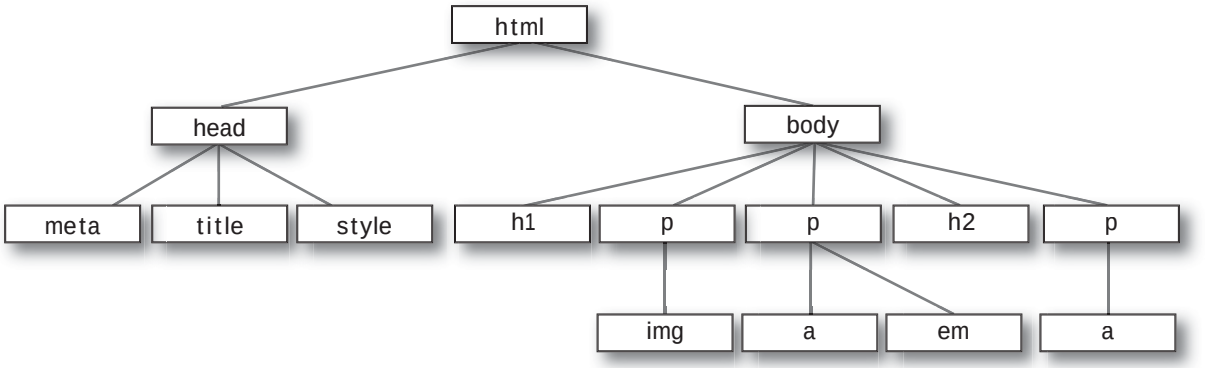
Down

1. With inheritance, a property set on one element is also passed down to its _____.
2. You can place your CSS inside these tags in an HTML file.
3. Won this time because they used external style sheets.
5. Property that represents font color.
6. Property for font type.
7. An external style file is called this.
8. They really wanted some style.

Markup Magnets Solution

Remember drawing the diagram of XHTML elements in Chapter 3? You did that again for the Lounge's main page. Here's the answer:

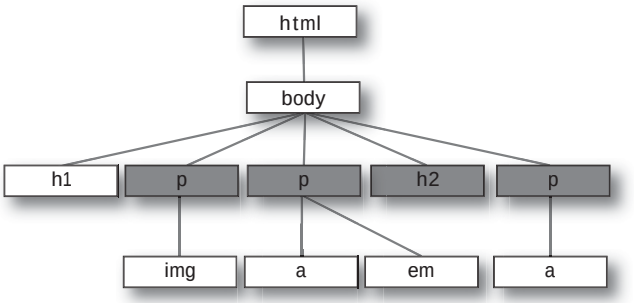




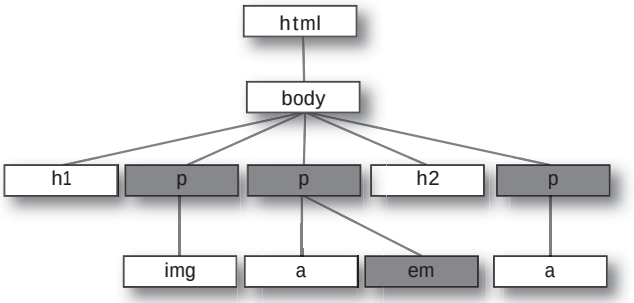
Sharpen your pencil
Solution

The selected elements are colored:

```
p, h2 {
    font-family: sans-serif;
}
```



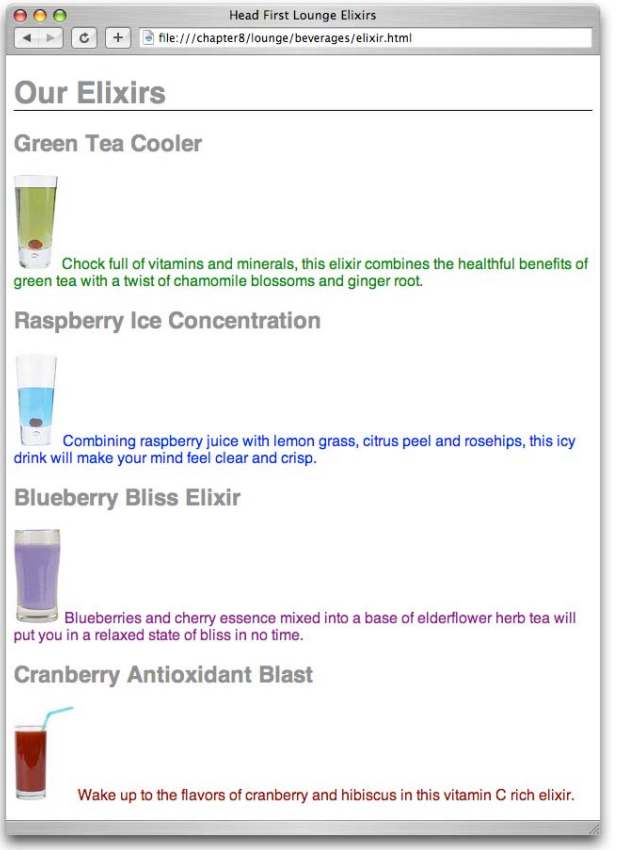

```
p, em {
    font-family: sans-serif;
}
```



Sharpen your pencil
Solution

Your turn: add two classes, “raspberry” and “blueberry” to the correct paragraphs in “elixir.html” and then write the styles to color the text blue and purple respectively. The property value for raspberry is “blue” and for blueberry is “purple”.

```
body {
        font-family: sans-serif;
}

h1, h2 {
        color: gray;
}

h1 {
        border-bottom: 1px solid black;
}

p {
        color: maroon;
}

p.greentea {
        color: green;
}

p.raspberry {
        color: blue;
}

p.blueberry {
        color: purple;
}
```

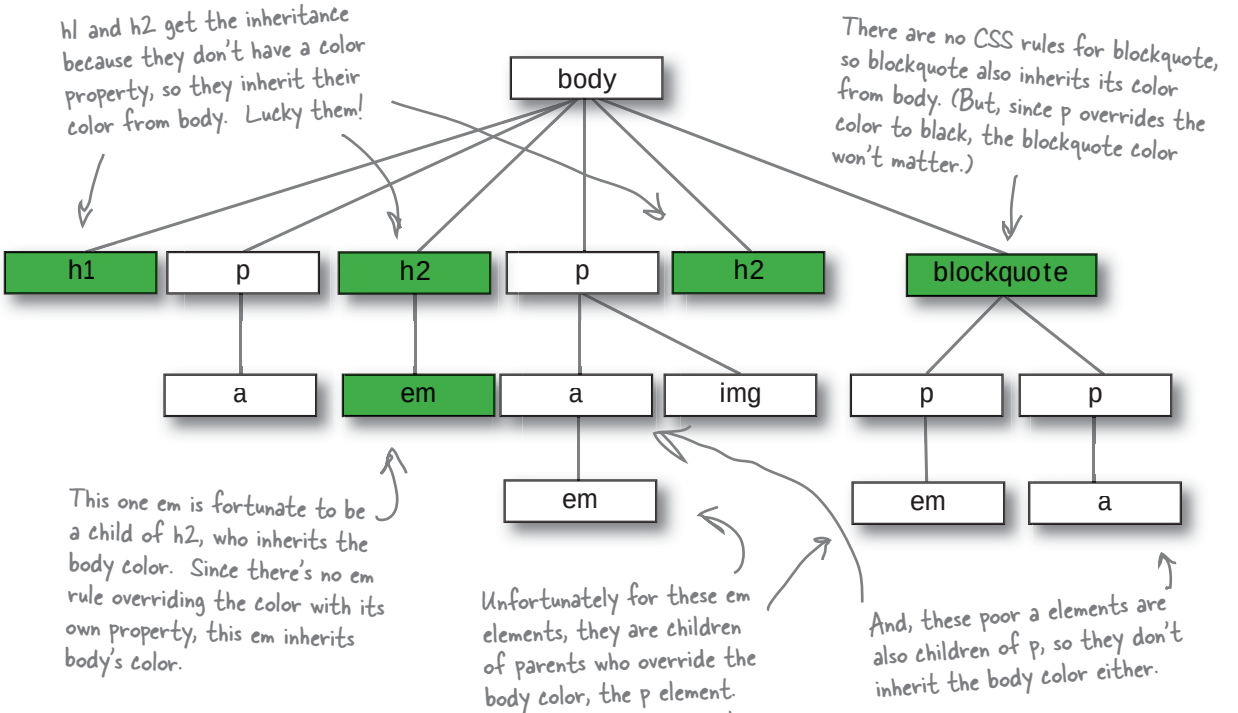



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <title>Head First Lounge Elixirs</title>
    <link type="text/css" rel="stylesheet" href="../lounge.css" />
  </head>
  <body>
    <h1>Our Elixirs</h1>
    <h2>Green Tea Cooler</h2>
    <p class="greentea">
              <img src="../images/green.jpg" />
              Chock full of vitamins and minerals, this elixir
              combines the healthful benefits of green tea with
              a twist of chamomile blossoms and ginger root.
    </p>
    <h2>Raspberry Ice Concentration</h2>
    <p class="raspberry" >
              <img src="../images/lightblue.jpg" />
              Combining raspberry juice with lemon grass,
              citrus peel and rosehips, this icy drink
              will make your mind feel clear and crisp.
    </p>
    <h2>Blueberry Bliss Elixir</h2>
    <p class="blueberry" >
              <img src="../images/blue.jpg" />
              Blueberries and cherry essence mixed into a base
              of elderflower herb tea will put you in a relaxed
              state of bliss in no time.
    </p>
    <h2>Cranberry Antioxidant Blast</h2>
    <p>
              <img src="../images/red.jpg" />
              Wake up to the flavors of cranberry and hibiscus
              in this vitamin C rich elixir.
    </p>
  </body>
</html>
```

Exercise solutions

Who gets the inheritance?

```
body {
        color: green;
}

p {
        color: black;
}
```



Exercise solutions

BE the Browser

Below, you'll find a CSS file with some errors in it. Your job is to play like you're the browser and locate all the errors. Did you find them all?

```
<style>
```
No XHTML in your CSS! The <style> tags are XHTML and don't work in a CSS style sheet.

```
body {
    background-color: white
```
Missing semicolon.

Missing }

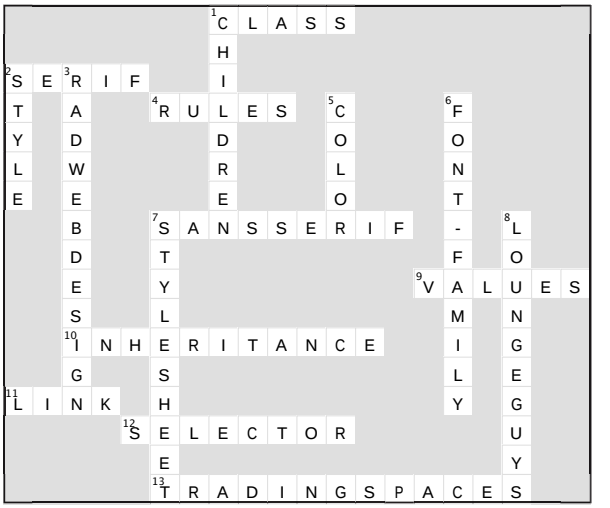
```
h1, {
```
Extra comma.

Missing property name.

```
    gray;
    font-family: sans-serif;
}

h2, p {
    color:
}
```
Missing property value and semicolon.

```
<em> {
    font-style: italic;
}
```
Using the XHTML tag instead of just the element name. This should be em.

```
</style>
```
No </style> tags needed in the CSS stylesheet.



Exercise Solutions

In your "lounge.html" file, change the greentea paragraph to include all the classes, like this:

```
<p class="greentea raspberry blueberry">
```

Save, and reload. What color is the Green Tea Cooler paragraph now?

purple

It's purple because the blueberry rule is last in the CSS file.

Next reorder the classes in your XHTML:

```
<p class="raspberry blueberry greentea">
```

Save, and reload. What color is the Green Tea Cooler paragraph now?

purple

It's still purple because the ordering of the names in the class attribute doesn't matter.

Next open your CSS file and move the p.greentea rule to the bottom of the file.

Save, and reload. What color is the Green Tea Cooler paragraph now?

green

Now, it's green, because the greentea rule comes last in the CSS file.

Finally, move the p.raspberry rule to the bottom of the file.

Save, and reload. What color is the Green Tea Cooler paragraph now?

blue

Now, it's blue, because the raspberry rule comes last in the CSS file.

After you've finished, rewrite the green tea element to look like it did originally:

```
<p class="greentea">
```

Save, and reload. What color is the Green Tea Cooler paragraph now?

green

Okay, now the <p> element only belongs to one class, so we use the most specific rule, which is p.greentea.