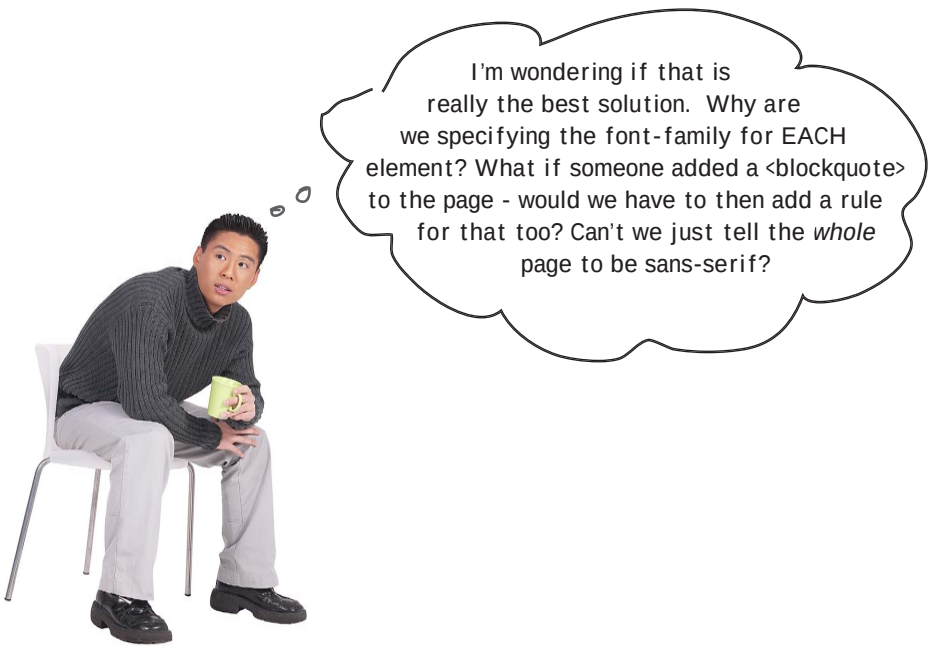
Sharpen your pencil
Solution

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

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

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

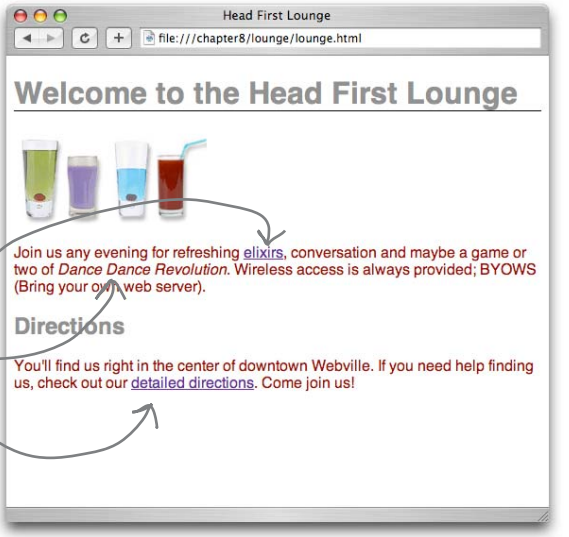
Just add a font-family property to your paragraph rule in the "lounge.css" file.



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>

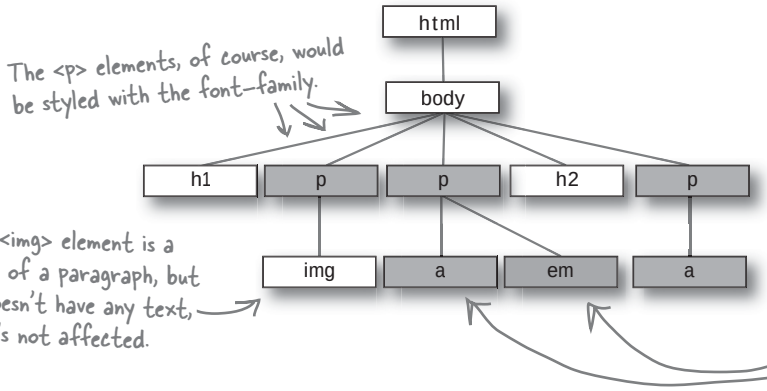
Just like you can inherit your blue eyes or brown hair from your parents, elements can inherit styles from their parents. In this case, the `<a>` and `<em>` elements inherited the `font-family` style from the `<p>` element, which is their parent element. It makes sense that changing your paragraph style 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 every style is inherited. Just some are, like font-family.

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.



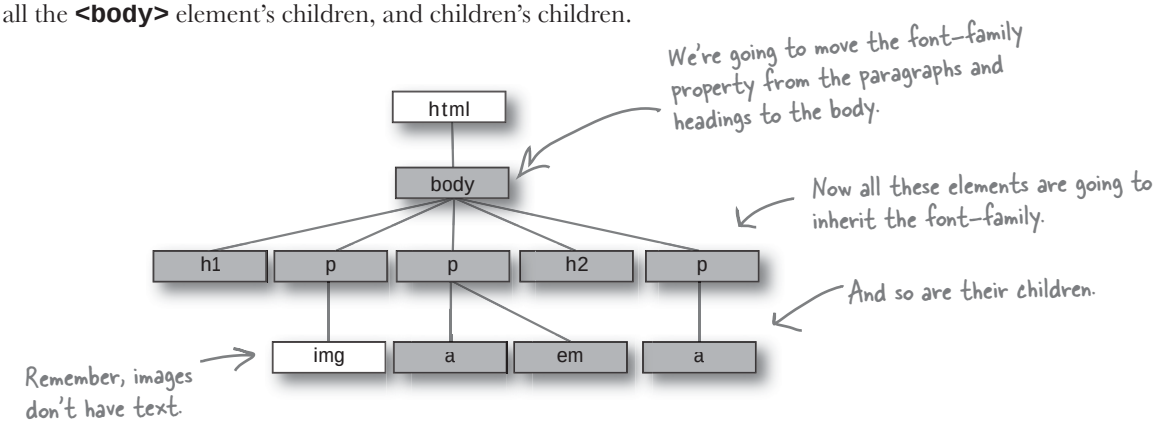
The <p> elements, of course, would be styled with the font-family.

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;
}
```

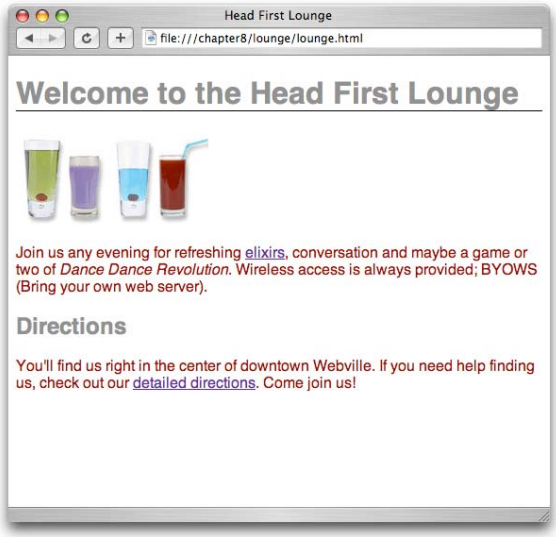
First, add a new rule that selects the <body> element. Then add the font-family property with a value of sans-serif.

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

Test drive your new CSS

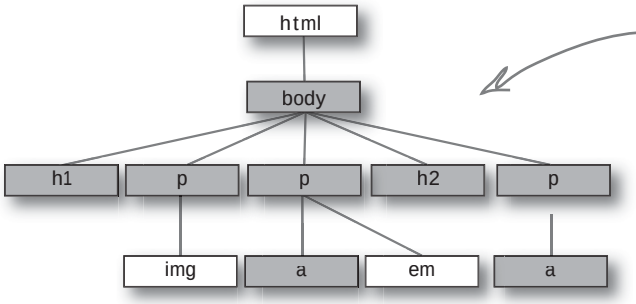
As usual, go ahead and make these changes in the "lounge.css" style sheet, save, and reload the "lounge.html" page. You shouldn't expect any changes, because the style is the same. It's just coming from a different rule. But you should feel better about your CSS because now you can add new elements to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

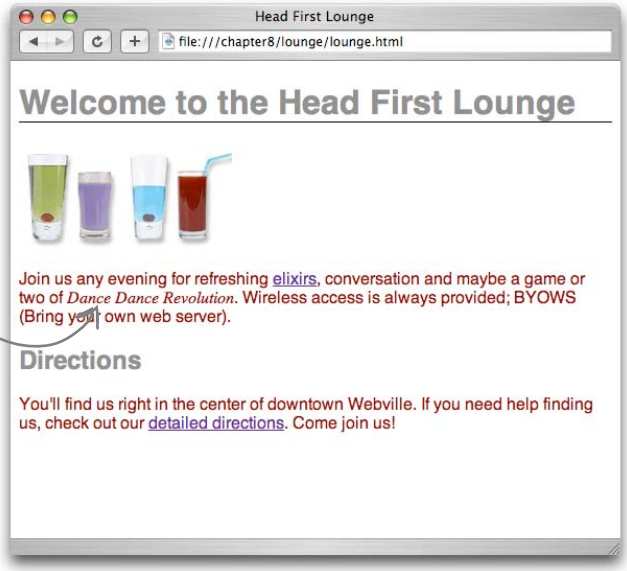
Well, then you can override the inheritance by supplying a specific rule just for ****. Here's how you add a rule for **** to override the font-family specified in the body:

```
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.

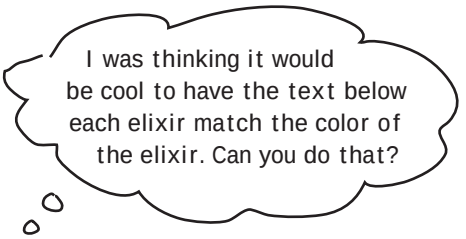
Q: This stuff gets complicated. Is there any way I can add comments to remind myself what the rules do?

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:

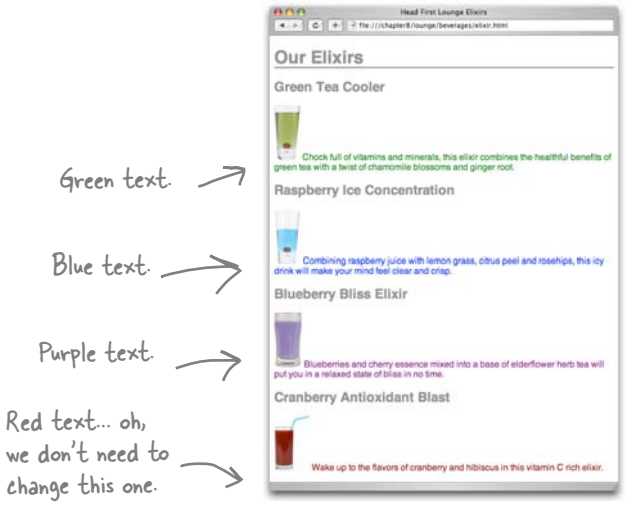
```
/* this rule will have no effect because it's in a comment

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:

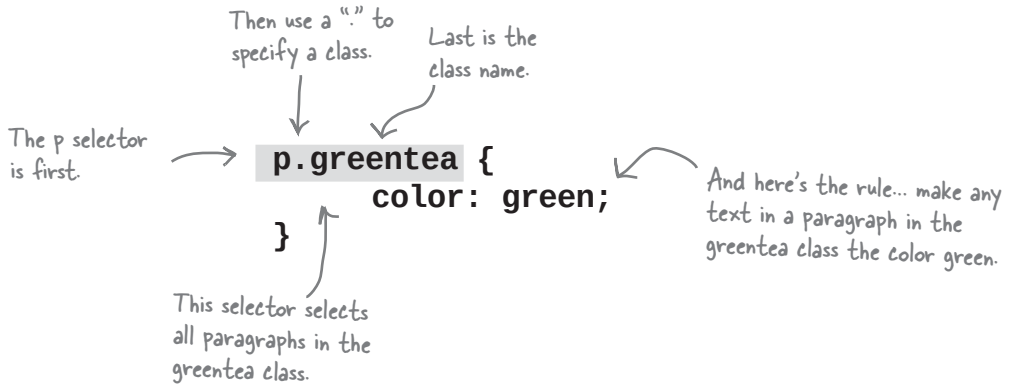
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
    <title>Head First Lounge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



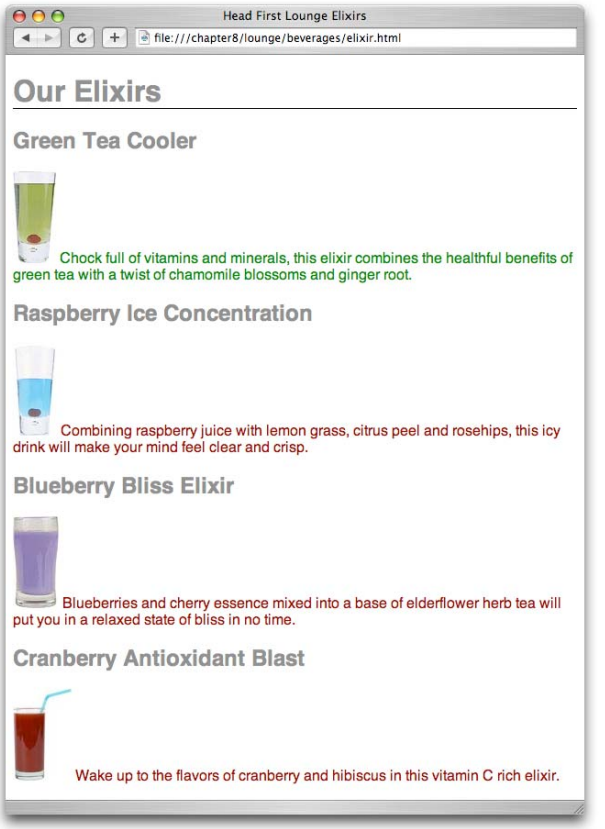
So now you have a way of selecting **<p>** elements that belong to a certain class. All you need to do is add the **class** attribute to any **<p>** elements you want to be green, and this rule will be applied. Give it a try: open your "lounge.css" file and add the **p.greentea** class selector to it.

```
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;
}
```

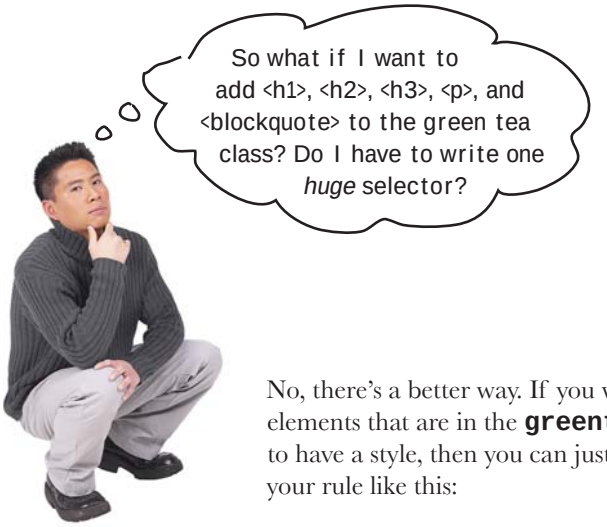
But what if you wanted to do the same to all **<blockquote>**s? Then you could do this:

```
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:

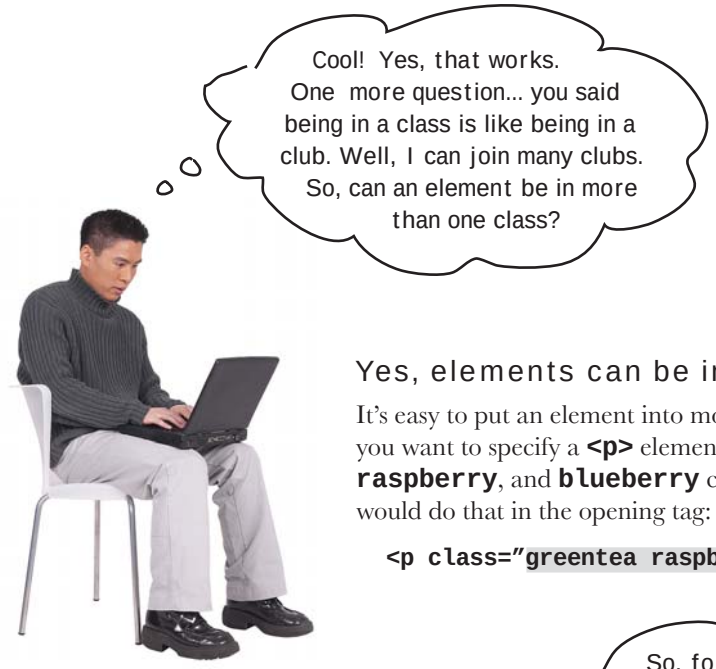
```
<blockquote class="greentea">
```



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.

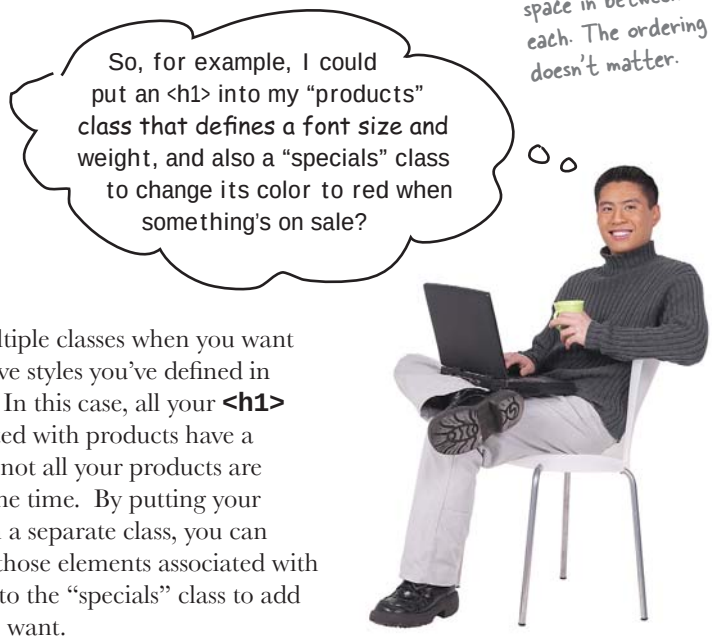


Yes, elements can be in more than one class.

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.



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?

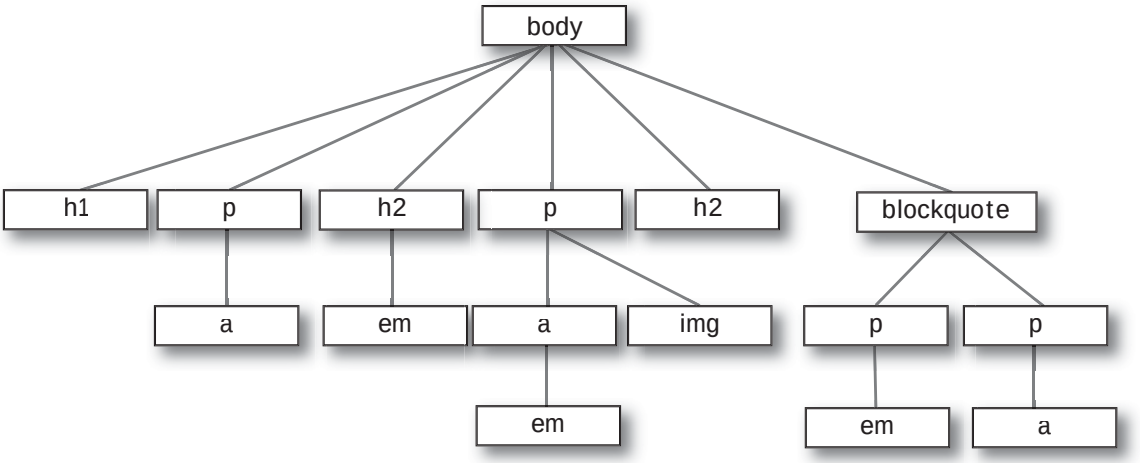
Oh yeah?

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).

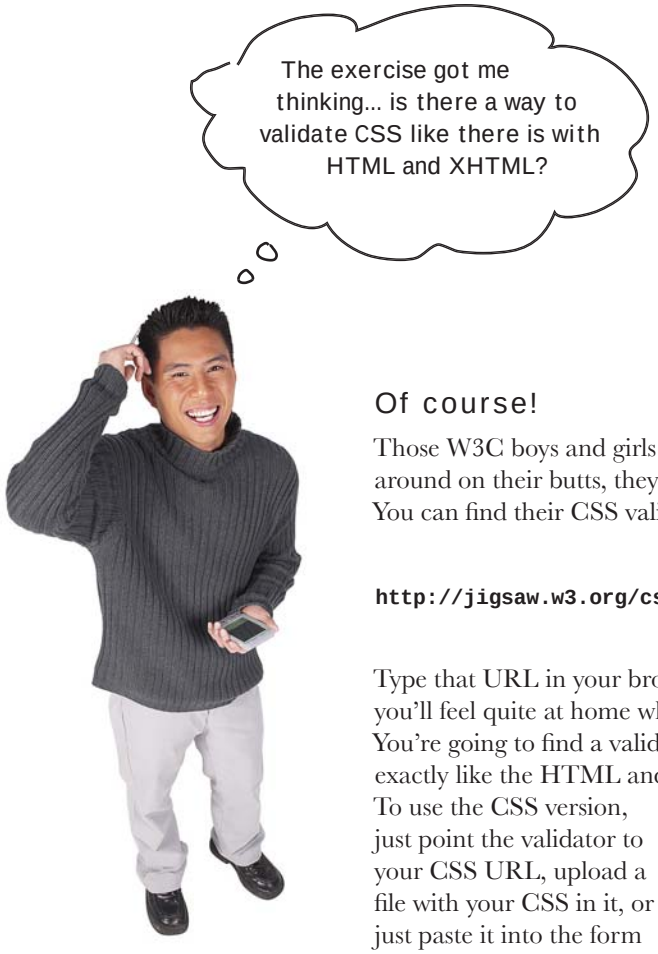
BE the Browser

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>
```



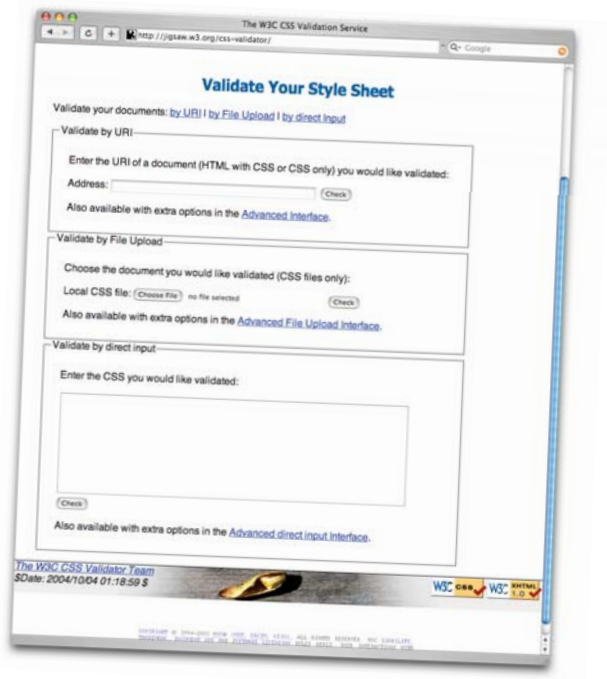
Of course!

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

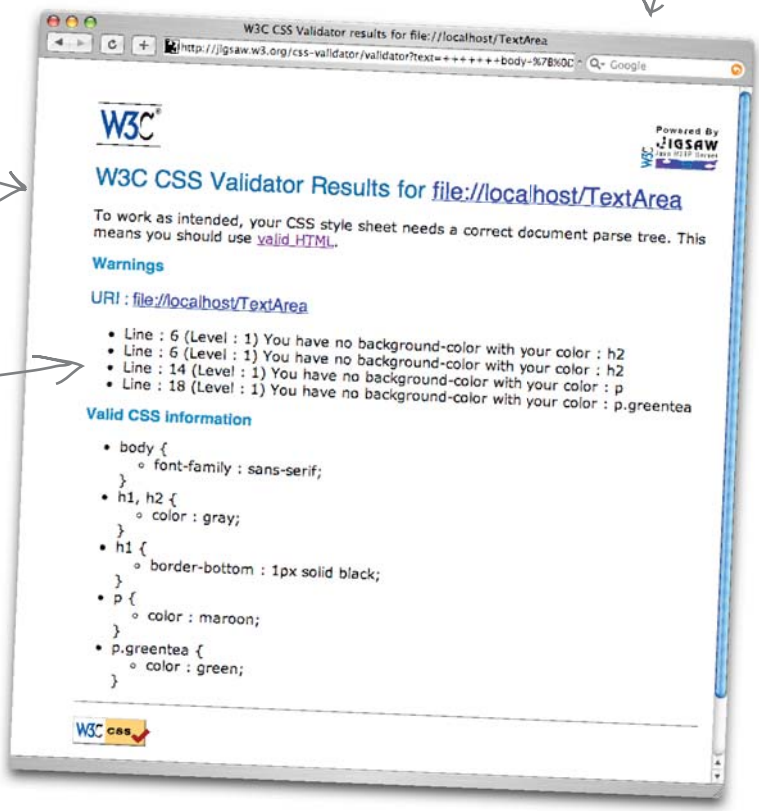
Before you wrap up this chapter, wouldn't you feel a lot better if all that Head First Lounge CSS validated? Sure you would. Use whichever method you want to get your CSS to the W3C. If you have your CSS on a server, type your URL into the form; otherwise, either upload your CSS file or just copy and 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.

Property Soup

top
Controls the position of the top of the element.

text-align
Use this property to align your text to the left, center, or right.

letter-spacing
This lets you set the spacing between letters. L i k e t h i s.

Use color to set the font color of text elements.
color

background-color
This property controls the background color of an element.

Use this property for italic or oblique text.
font-style

This property controls the weight of text. Use it to make text bold.
font-weight

border
This property puts a border around an element. You can have a solid border, a ridged border, a dotted border...

This property lets you change how list items look in a list.
list-style

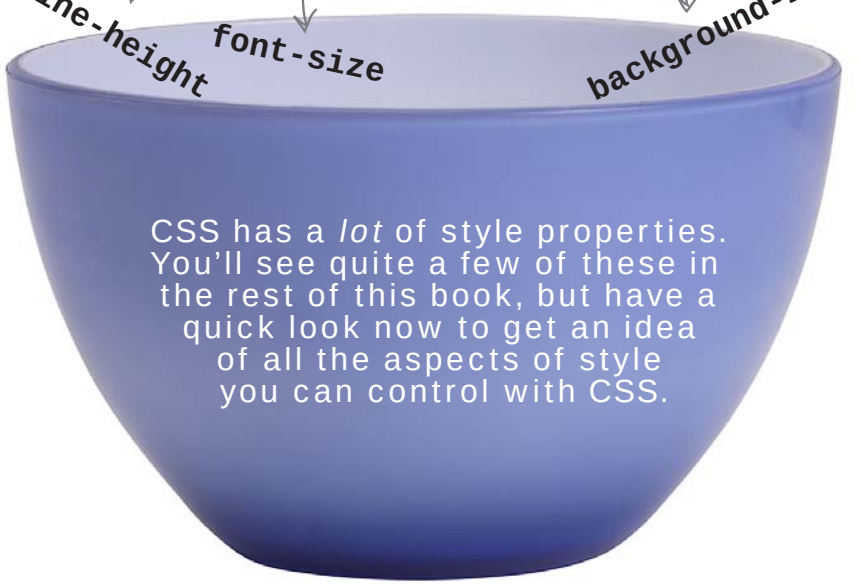
left
This is how you tell an element how to position its left side.

margin
If you need space between the edge of an element and its content, use margin.

This property sets the space between lines in a text element.
line-height

Makes text bigger or smaller.
font-size

Use this property to put an image behind an element..
background-image



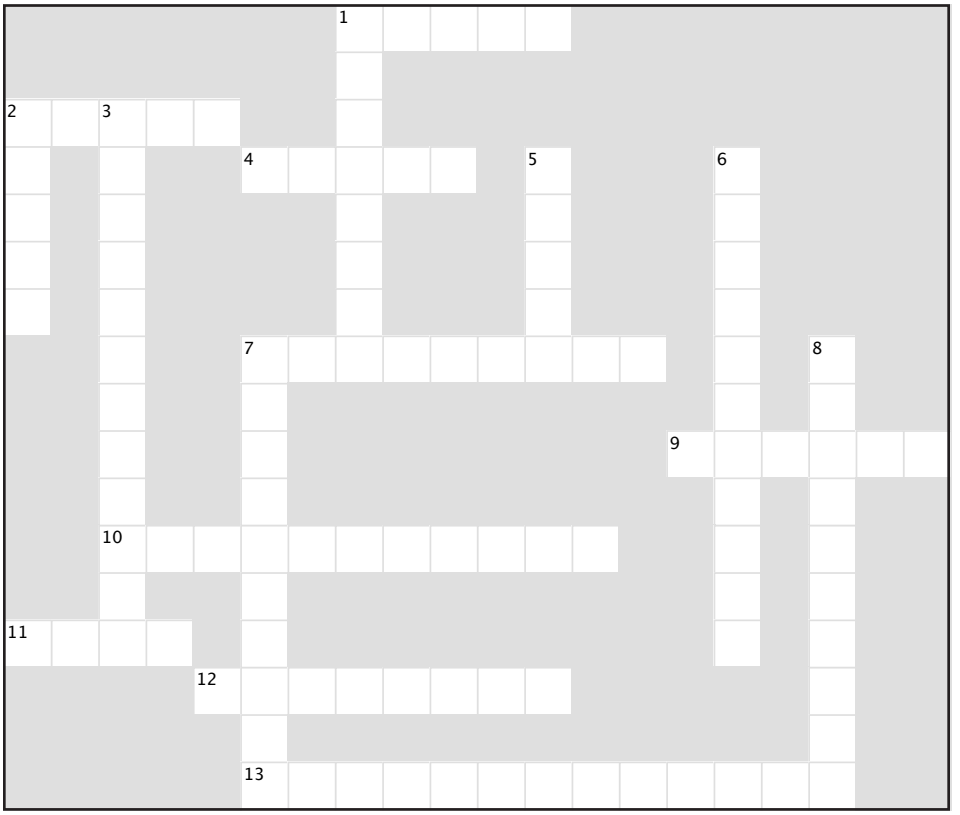
CSS has a *lot* of style properties. You'll see quite a few of these in the rest of this book, but have a quick look now to get an idea of all the aspects of style you can control with CSS.

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.

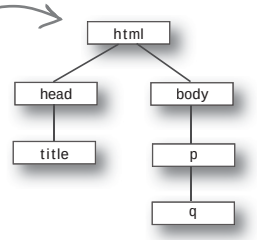
Down

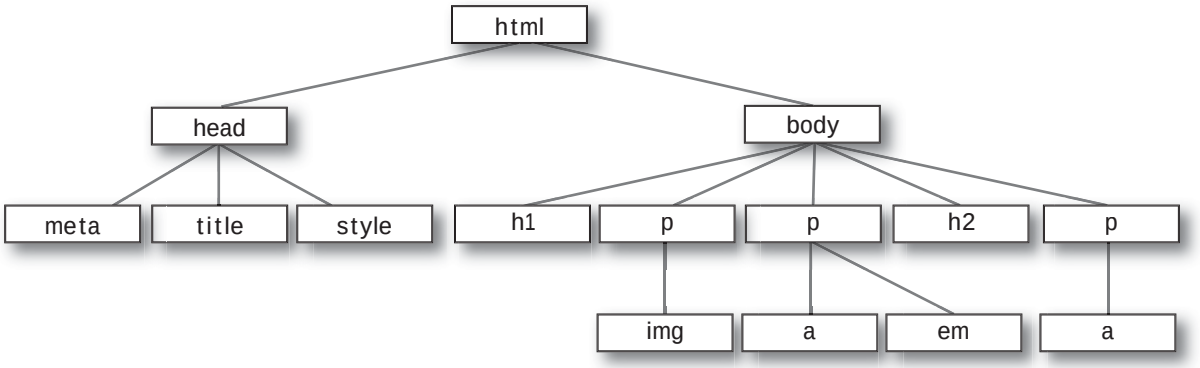
1. With inheritance, a property set on one element is also passed down to its _____.
2. You can place your CSS inside these tags in an HTML file.
3. Won this time because they used external style sheets.
5. Property that represents font color.
6. Property for font type.
7. An external style file is called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

Like this.

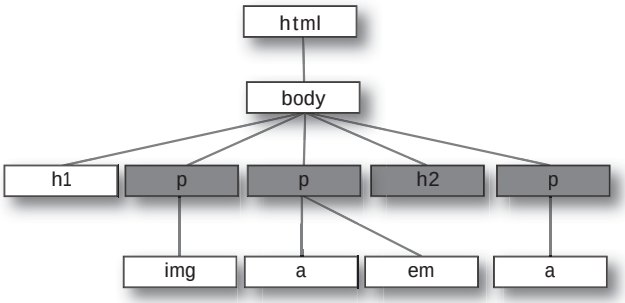




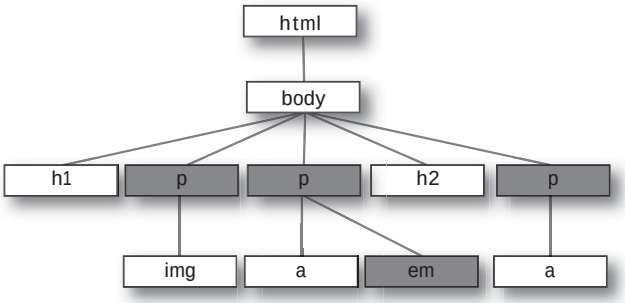
Sharpen your pencil
Solution

The selected elements are colored:

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



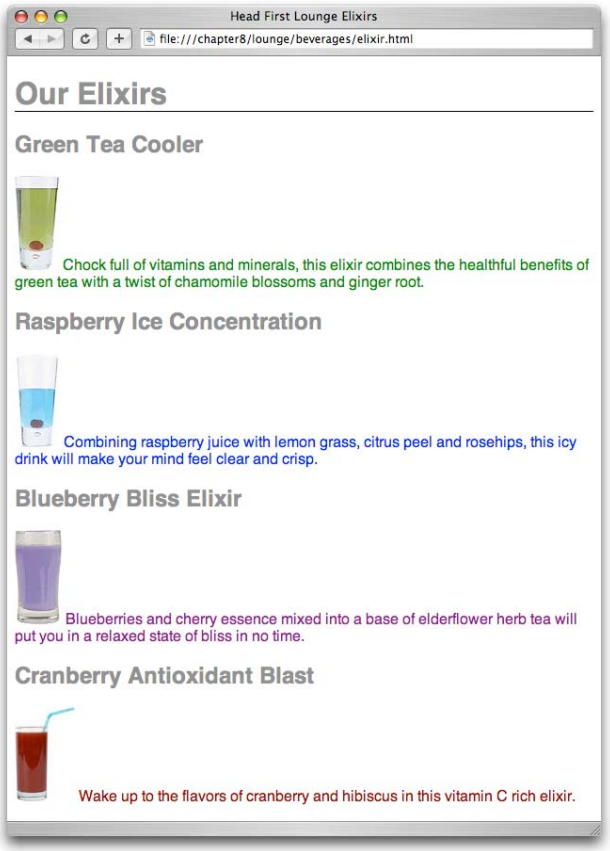

```
p, em {
    font-family: sans-serif;
}
```



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;
}
```



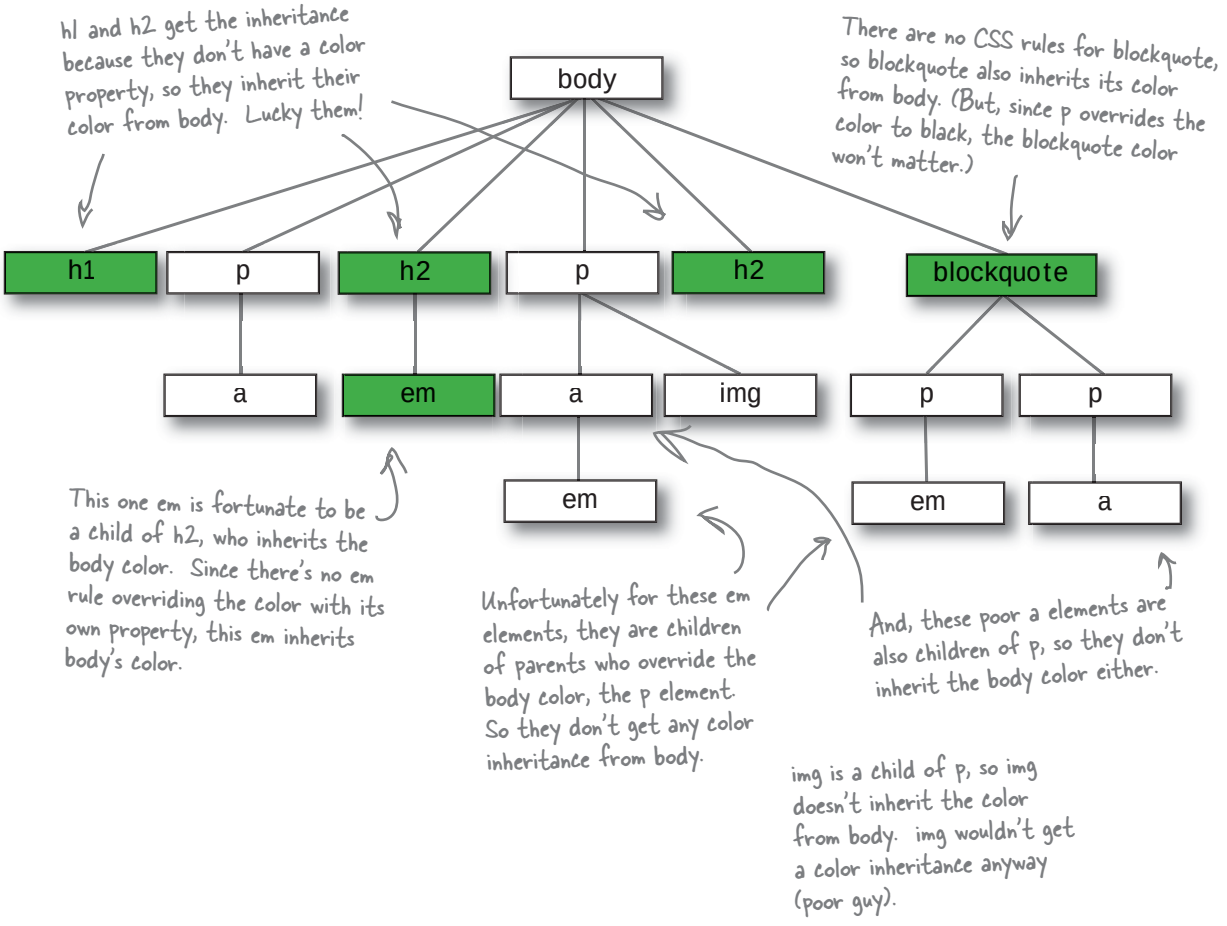
Solution

```
<!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 }

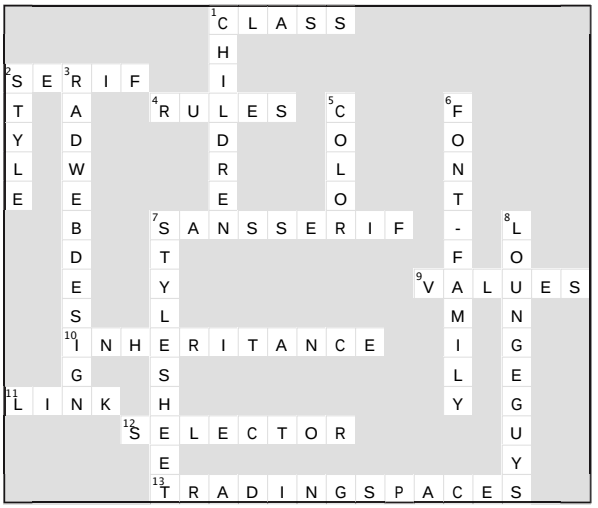
```
h1, {
    gray;
    font-family: sans-serif;
}
```
Extra comma.

Missing property name.

```
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.