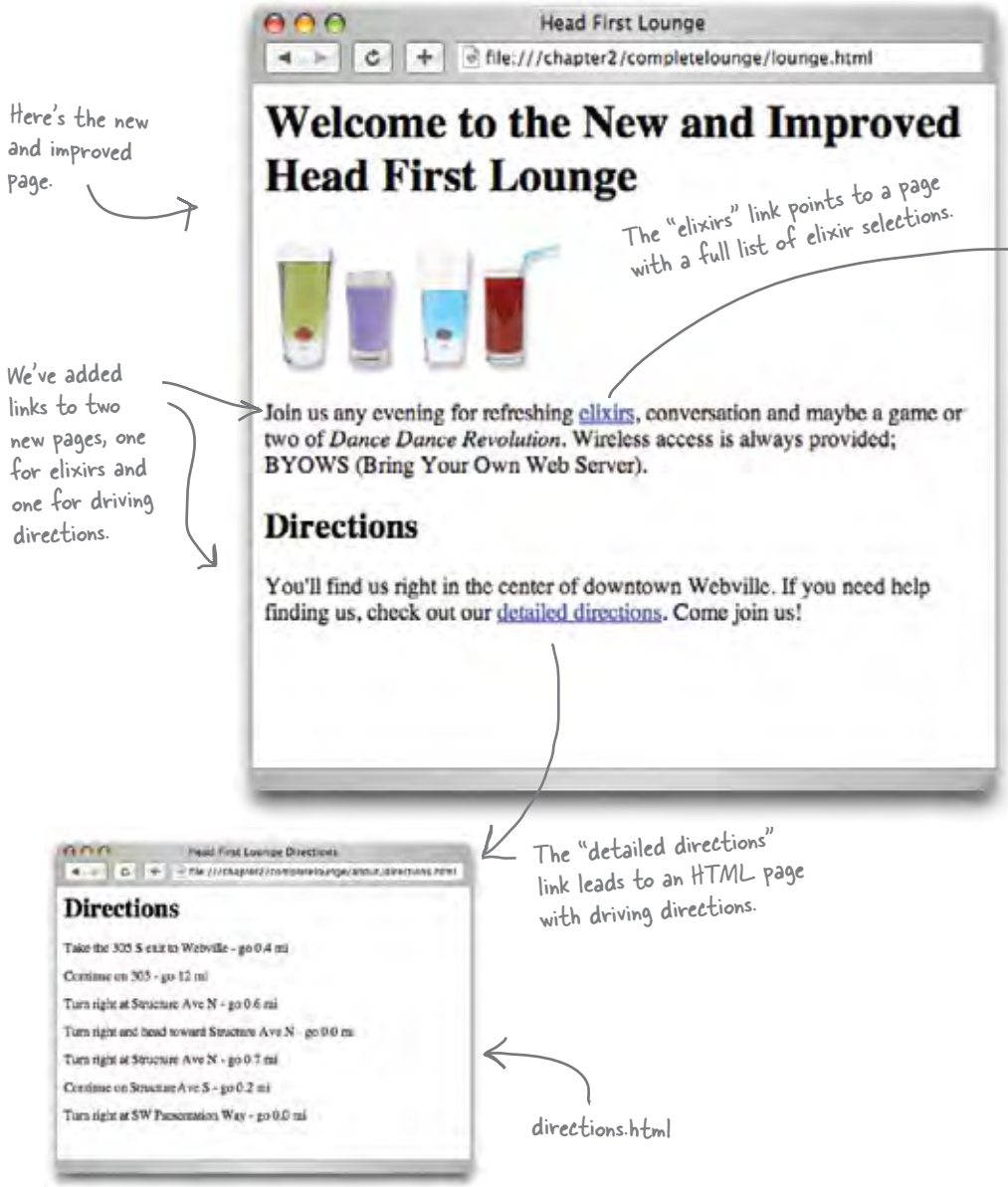
2 going further, with hypertext

Meeting the 'HT' in HTML

Did someone say "hypertext?" What's that? Oh, only the *entire basis* of the Web. In Chapter 1 we kicked the tires of HTML and found it to be a nice *markup language* (the 'ML' in HTML) for describing the structure of Web pages. Now we're going to check out the 'HT' in HTML, *hypertext*, which will let us break free of a single page and link to other pages. Along the way we're going to meet a powerful new element, the <a> element, and learn how being "relative" is a groovy thing. So, fasten your seat belts – you're about to learn some hypertext.

Head First Lounge, *New and Improved*

Remember the Head First Lounge? Great site, but wouldn't it be nice if customers could view a list of the refreshing elixirs? Even better, we should give customers some real driving directions so they can find the place.



elixir.html

A page listing some refreshing and healthy drinks. Feel free to grab one before going on.

Creating the new and improved lounge in three steps...

Let's rework the original Head First Lounge page so it links to the two new pages.

1. The first step is easy because we've already created the "directions.html" and "elixir.html" files for you. You'll find them in the source files for the book, which are available at http://www.headfirstlabs.com.

2. Next you're going to edit the "lounge.html" file and add in the HTML needed to link to "directions.html" and "elixir.html".

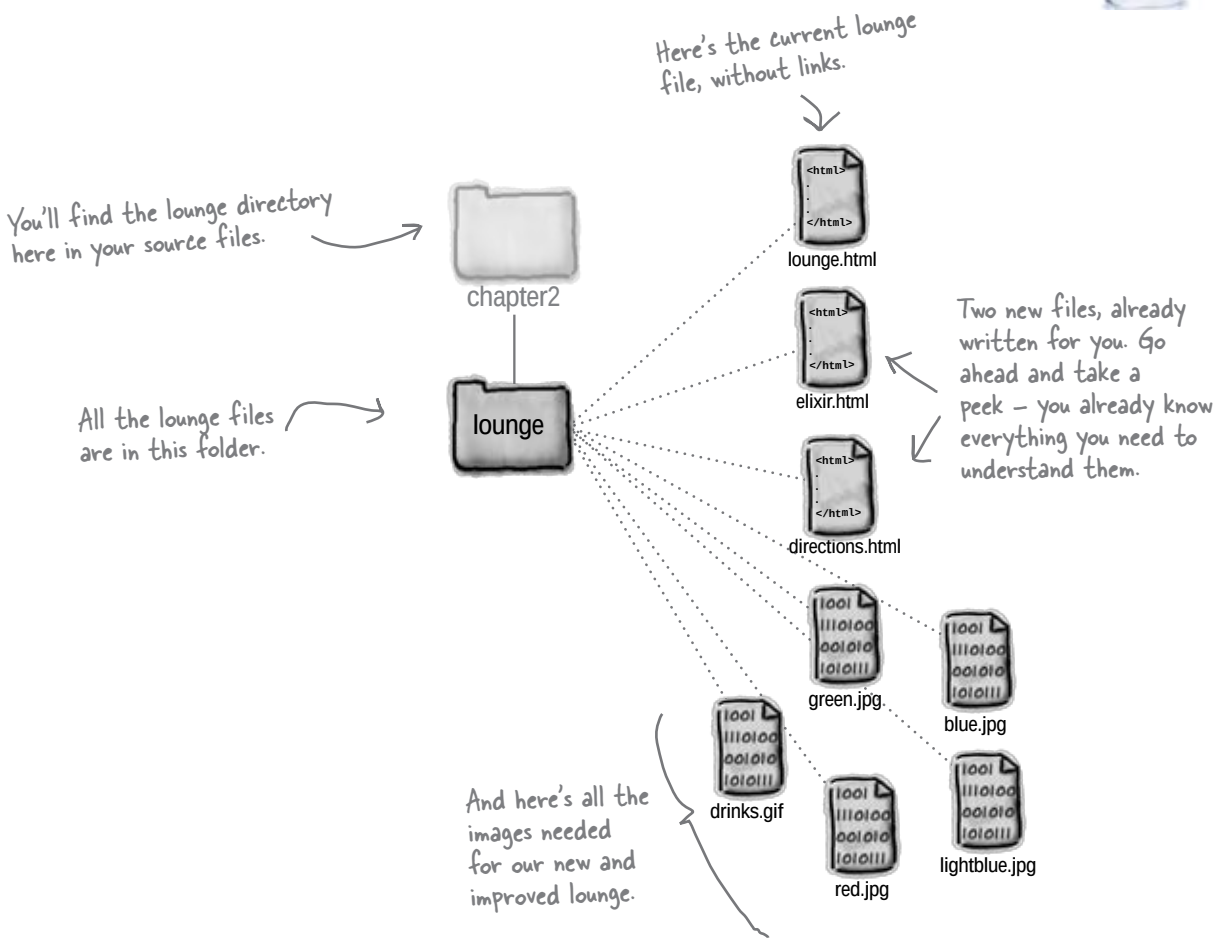
3. Last, you'll give the pages a test drive and try out your new links. When you get back we'll sit down and look at how it all works.

Flip the page and let's get started...

Creating the new lounge

1 Grab the source files

Go ahead and grab the source files from http://www.headfirstlabs.com. Once you've downloaded them, look under the folder "chapter2/lounge" and you'll find "lounge.html", "elixir.html", and "directions.html" (and a bunch of image files).



BRAIN POWER

The Head First Lounge is already growing; do you think that keeping all the site's files in a single directory is a good way to organize the site? What would you do differently?

2 Edit lounge.html

Open "lounge.html" in your editor. Add the new text and HTML that is highlighted below. Go ahead and type this in; we'll come back and see how it all works on the next page.

```
<html>
  <head>
    <title>Head First Lounge</title>
  </head>
  <body>
    <h1>Welcome to the New and Improved Head First Lounge</h1>
    <img src="drinks.gif">
    <p>
       Join us any evening for
       refreshing <a href="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="directions.html">detailed directions</a>.
      Come join us!
    </p>
  </body>
</html>
```

Let's add "New and Improved" to the heading.

Here's where we add the HTML for the link to the elixirs.

To create links we use the <a> element; we'll take a look at how this element works in just a sec...

We need to add some text here to point customers to the new directions.

And here's where we add the link to the directions, again using an <a> element.

3 Save lounge.html and give it a test drive.

When you're finished with the changes, save the file "lounge.html" and open it in your browser. Here are a few things to try...

1. Click on the elixir link and the new elixir page will display.
2. Click on the browser's back button and "lounge.html" should be displayed again.
3. Click on the directions link and the new directions page will display.

Okay, I've loaded the new lounge page, clicked the links, and everything worked. But, I want to make sure I understand how the HTML works.

Behind the Scenes

What did we do?

1. Let's step through creating the HTML links. First we need to put the text we want for the link in an <a> element, like this:

```
<a>elixirs</a>
```

```
<a>driving directions</a>
```

The <a> element is used to create a link to another page.

The content of the <a> element acts as a label for the link. In the browser the label appears with an underline to indicate you can click on it.

2. Now that we have a label for each link, we need to add some HTML to tell the browser where the link points to:

```
<a href="elixir.html">elixirs</a>
```

For this link, the browser will display an "elixirs" label that, when clicked, will take the user to the "elixir.html" page.

The href attribute is how you specify the destination of the link.

```
<a href="directions.html">driving directions</a>
```

And for this link, the browser will display a "driving directions" link that, when clicked, will take the user to the "directions.html" page.

What does the browser do?

1. First, as the browser renders the page, if it encounters an <a> element, it takes the content of the element and displays it as a clickable link.

```
<a href="elixir.html">elixirs</a>
```

Both "elixirs" and "detailed directions" are between the opening and closing <a> tags, so they end up being clickable labels in the web page.


```
<a href="directions.html">detailed directions</a>
```

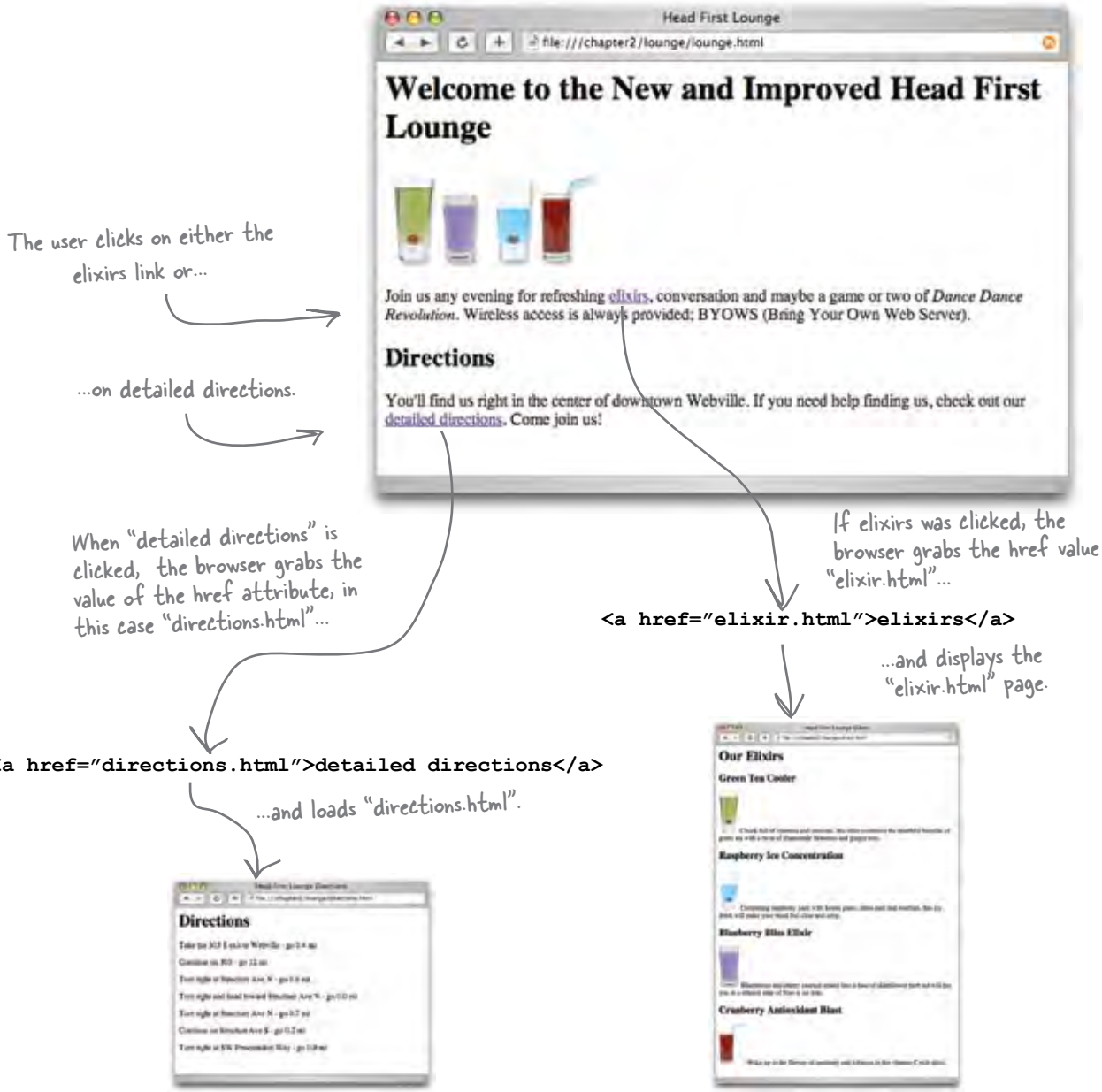
Use the <a> element to create a hypertext link to another web page.

The content of the <a> element becomes clickable in the web page.

The href attribute tells the browser the destination of the link.

Behind the Scenes

2 Next, when a user clicks on a link, the browser uses the "href" attribute to determine the page the link points to.



Understanding attributes

Attributes give you a way to specify additional information about an element. While we haven't looked at attributes in detail, you've already seen a few examples of them:

```
<style type="text/css">
<a href="irule.html">
<img src="sweetphoto.gif">
```

The type attribute specifies which style language we're using, in this case CSS.

The href attribute tells us the destination of a hyperlink.

The src attribute specifies the filename of the picture an img tag displays.

Let's cook up an example to give you an even better feel for how attributes work:

What if <car> was an element?

If **<car>** was an element, then you'd naturally want to write some markup like this:

```
<car>My Red Mini</car>
```

With no attributes, all we can supply is a descriptive name for the car.

But this **<car>** element only gives a descriptive name for your car – it doesn't tell us the make, precise model, whether it is a convertible, or a zillion other details we might want to know. So, if **<car>** were really an element, we might use attributes like this:

But with attributes, we can customize the element with all kinds of information.

```
<car make="BMW" model="Mini Cooper" convertible="no">My Red Mini</car>
```

Better, right? Now this markup tells us a lot more information in an easy to write, convenient form.

SAFETY FIRST

Attributes are always written the same **way: first comes the attribute name,** followed by an equals sign, and then the attribute value surrounded in double quotes.

You may see some sloppy HTML on the Web that leaves off the double quotes, but don't get lazy yourself. Being sloppy can cause you a lot of problems down the road (as we'll see later in the book).

Do this (correct form)

```
<a href="top10.html">Great Movies</a>
```

attribute name, equals sign, double quote, attribute value, double quote

Not this (incorrect form)

```
<a href=top10.html>Great Movies</a>
```

WRONG – no double quotes around the attribute value.

there are no Dumb Questions

Q: Can I just make up new attributes for an HTML element?

A: No, because Web browsers only know about a predefined set of attributes for each element. If you just made up attributes, then Web browsers wouldn't know what to do with them, and as you'll see later in the book, doing this will very likely get you into trouble. When a browser recognizes an element or an attribute, we like to say that it "supports" that element or attribute. You should only use attributes that you know are supported.

Q: Who decides what is "supported?"

A: There are standards committees that worry about the elements and attributes of HTML. These committees are made up of people ~~with nothing better to do~~ who generously give their time and energy to make sure there's a common HTML roadmap that all companies can use to implement their browsers.

Q: How do I know what attributes and elements are supported? Or, can all attributes be applied to any element?

A: Only certain attributes can be used with a given element. Think about it this way: you wouldn't use an attribute "convertible" with the element <toaster>, would you? So, you only want to use attributes that make sense and are supported by the element.

We're going to be learning which attributes are supported by which elements as we make our way through the book. After you've finished the book there are lots of great references you can use to refresh your memory, such as *HTML & XHTML: The Definitive Guide* (O'Reilly).

Attributes Exposed

This week's interview: Confessions of the href attribute

HeadFirst: Welcome, href. It's certainly a pleasure to interview as big an attribute as you.

href: Thanks. It's good to be here and get away from all the linking; it can wear an attribute out. Every time someone clicks on a link, guess who gets to tell the browser where to go next? That would be me.

HeadFirst: We're glad you could work us into your busy schedule. Why don't you take us back to the beginning... What does it mean to be an attribute?

href: Sure. Well, attributes are used to customize an element. It's easy to wrap some **<a>** tags around a piece of content, like "Sign up now!" – we do it like this: **<a>Sign up now!** – but without me, the href attribute, you have no way to tell the **<a>** element the destination of the link.

HeadFirst: Got it so far...

href: ...but with an attribute you can provide additional information about the element. In my case, that's where the link points to: **Sign up now!**. This says that the **<a>** element, which is labeled "Sign up now!", links to the "signup.html" page. Now, there are lots of other attributes in the world, but I'm the one you use with the **<a>** element to tell it where it points to.

HeadFirst: Nice. Now, I have to ask, and I hope you aren't offended, but what is with the name? href? What's with that?

href: It's an old Internet family name. It means "hypertext reference", but all my friends just call me "href" for short.

HeadFirst: Which is?

href: A hypertext reference is just another name for a resource that is on the Internet or your computer. Usually the resource is a Web page, but I can also point to audio, video... all kinds of things.

HeadFirst: Interesting. All our readers have seen so far are links to their own pages; how do we link to other pages and resources on the Web?

href: Hey, I gotta get back to work, the whole Web is getting gunked up without me. Besides, isn't it your job to teach them this stuff?

HeadFirst: Okay okay, yes, we're getting to that in a bit... thanks for joining us, href.

Exercise

You've created links to go from "lounge.html" to "elixir.html" and "directions.html"; now we're going to go back the other way. Below you'll find the HTML for "elixir.html". Add a link with the label "Back to the Lounge" at the bottom of the elixir page that points back to "lounge.html".

```
<html>
  <head>
    <title>Head First Lounge Elixirs</title>
  </head>
  <body>
    <h1>Our Elixirs</h1>

    <h2>Green Tea Cooler</h2>
    <p>
                <img src="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="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="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="red.jpg">
                Wake up to the flavors of cranberry and hibiscus
                in this vitamin C rich elixir.
    </p>
```

Your new HTML goes here.

```
  </body>
</html>
```

When you are done, go ahead and do the same with "directions.html" as well.

We need some help constructing and deconstructing <a> elements. Given your new knowledge of the <a> element, we're hoping you can help. In each row below you'll find some combination of the label, destination, and the complete <a> element. Fill in any information that is missing. The first row is done for you.

Label	Destination	What you write in HTML
Hot or Not?	hot.html	`<a href="hot.html">Hot or Not?</a>`
Resume	cv.html	
	candy.html	`<a href="_____">Eye Candy</a>`
See my mini	mini-cooper.html	
let's play		`<a href="millionaire.html">_____</a>`

there are no Dumb Questions

Q: I've seen many pages where I can click on an image rather than text. Can I use the <a> element for that?

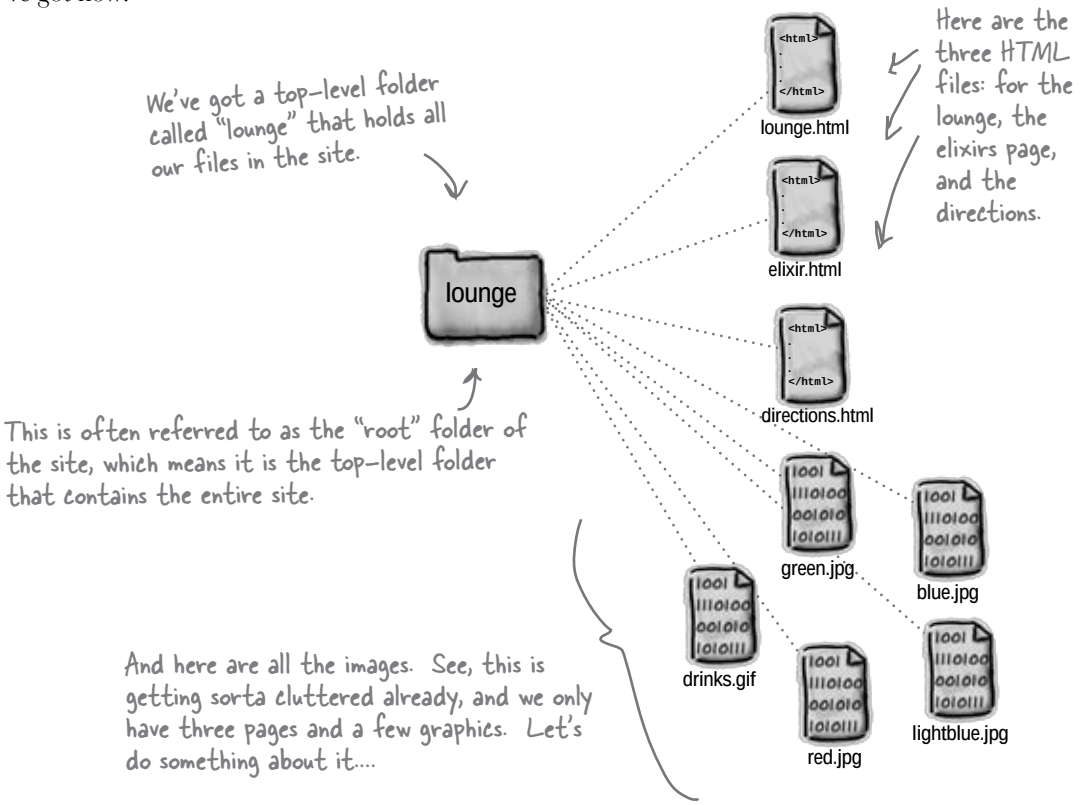
A: Yes, if you put an **<img>** element between the **<a>** tags then your image will be clickable just like text. We're not going to talk about images in depth for a few chapters, but they work just fine as links.

Q: So I can put anything between the <a> tags and it will be clickable? Like, say, a paragraph?

A: Whoa now. Not so fast. Not every element can be placed inside an **<a>** element. In general you'll just be using text and images (or both) within the **<a>** element. What tags will go inside other tags is a whole other topic, but don't worry; we'll get there soon enough.

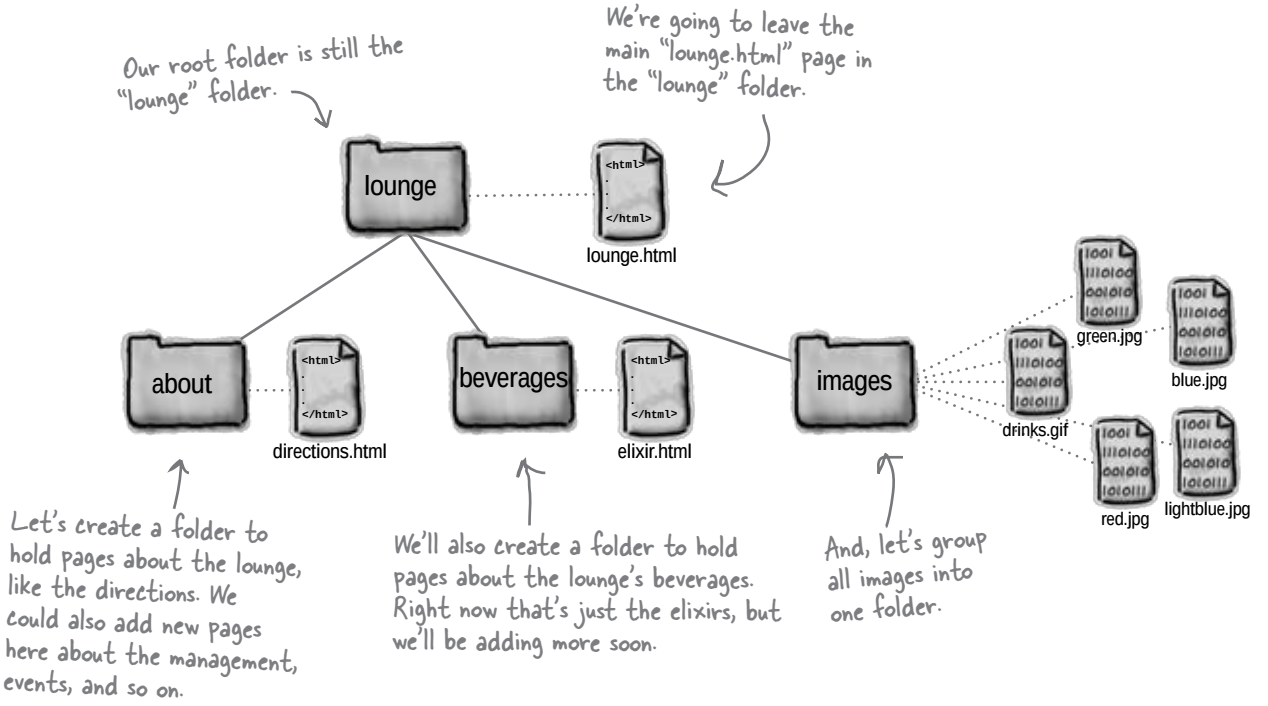
Getting organized

Before you start creating more HTML pages, it's time to get things organized. So far, we've been putting all our files and images in one folder. You'll find that even for modestly-sized Web sites, things are much more manageable if you organize your Web pages, graphics, and other resources into a set of folders. Here's what we've got now:



Organizing the lounge...

Let's give the lounge site some meaningful organization now. Keep in mind there are lots of ways to organize any site; we're going to start simple and create a couple of folders for pages. We'll also group all those images into one place.



there are no Dumb Questions

Q: Since you have a folder for images, why not have another one called "html" and put all the HTML in that folder?

A: You could. There aren't any "correct" ways to organize your files; rather, you want to organize them in a way that works best for you and your users. As with most design decisions, you want to choose an organization scheme that is flexible enough to grow, while keeping things as simple as you can.

Q: Or, why not put an images folder in each other folder, like "about" and "beverages."

A: Again, we could have. We expect that some of the images will be reused among several pages, so we put images in a folder at the root (the top level) to keep them all together. If you have a site that needs lots of images in different parts of the site, you might want each branch to have its own image folder.

Q: "Each branch"?

A: You can understand the way folders are described by looking at them as upside down trees. At the top is the root and each path down to a file or folder is a branch.

Exercise

Now you need to create the file and folder structure shown on the previous page. Here's exactly what you need to do:

1. Locate your "lounge" folder and create three new subfolders named "about", "beverages", and "images".
2. Move the file "directions.html" into the "about" folder.
3. Move the file "elixir.html" into the "beverages" folder.
4. Move all the images into the "images" folder.
5. Finally, load your "lounge.html" file and try out the links. Compare with how ours worked below.

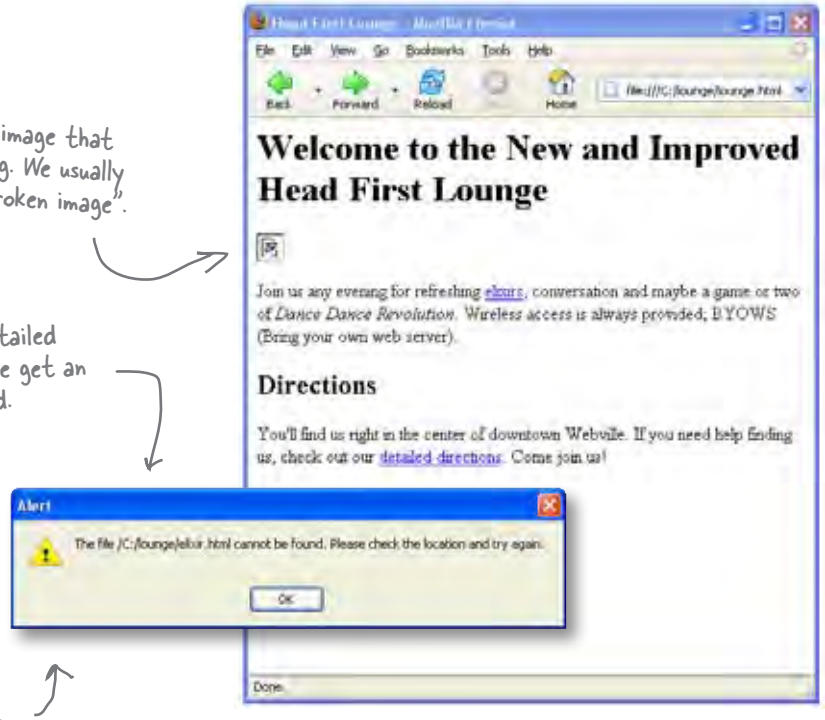
Technical difficulties

It looks like we've got a few problems with the lounge page after moving things around.

We've got an image that isn't displaying. We usually call this a "broken image".

And, when you click on elixirs (or detailed directions) things get much worse: we get an error saying the page can't be found.

Some browsers display this error as a web page rather than a dialog box.



Right. We need to tell the browser the new location of the pages.

So far you've used **`href`** values that point to pages in the *same folder*. Sites are usually a little more complicated, though, and you need to be able to point to pages that are in *other folders*.

To do that, you trace the path from your page to the destination file. That might mean going down a folder or two, or up a folder or two, but either way we end up with a *relative path* that we can put in the **`href`**.

Planning your paths...

What do you do when you're planning that vacation in the family truckster? You get out a map and start at your current location, and then trace a path to the destination. The directions themselves are *relative* to your location – if you were in another city, they'd be different directions, right?

Okay, you'd really go to Google maps, but work with us here!

To figure out a relative path for your links, it's the same deal: you start from the page which has the link, and then you trace a path through your folders until you find the file you need to point to.

There are other kinds of paths too. We'll get to those in later chapters.

Let's work through a couple of relative paths (and fix the lounge at the same time):

Linking down into a subfolder

1 Linking from "lounge.html" to "elixir.html".

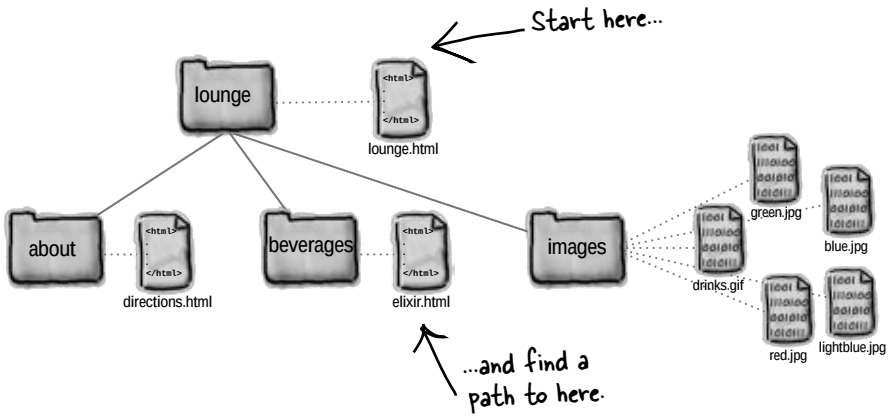
We need to fix the elixirs link in the "lounge.html" page. Here's what the **<a>** element looks like now:

```
<a  href="elixir.html">elixirs</a>
```

Right now we're just using the filename "elixir.html", which tells the browser to look in the same folder as "lounge.html".

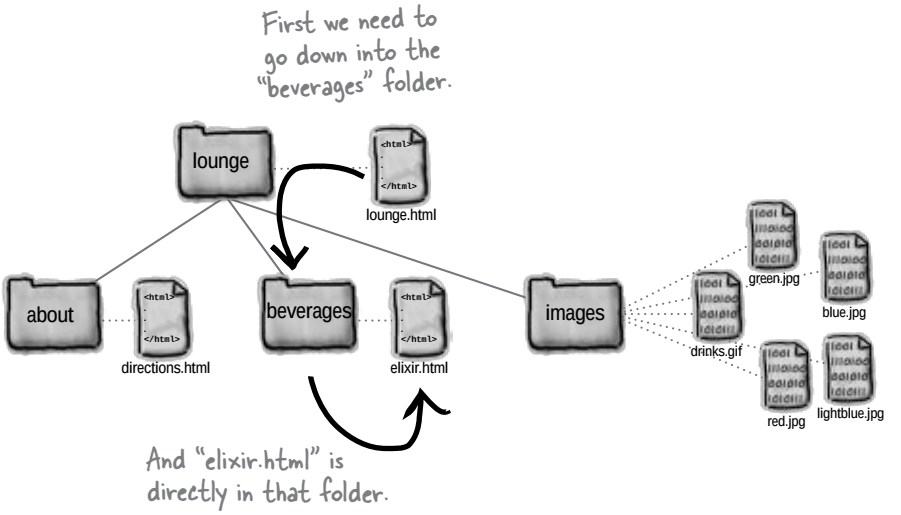
2 Identify the source and the destination.

When we re-organized the lounge, we left "lounge.html" in the "lounge" folder, and we put "elixir.html" in the "beverages" folder, which is a subfolder of "lounge".



③ Trace a path from the source to the destination.

Let's trace the path. To get from the "lounge.html" file to "elixir.html", we need to go into the "beverages" folder first, and then we'll find "elixir.html" in that folder.



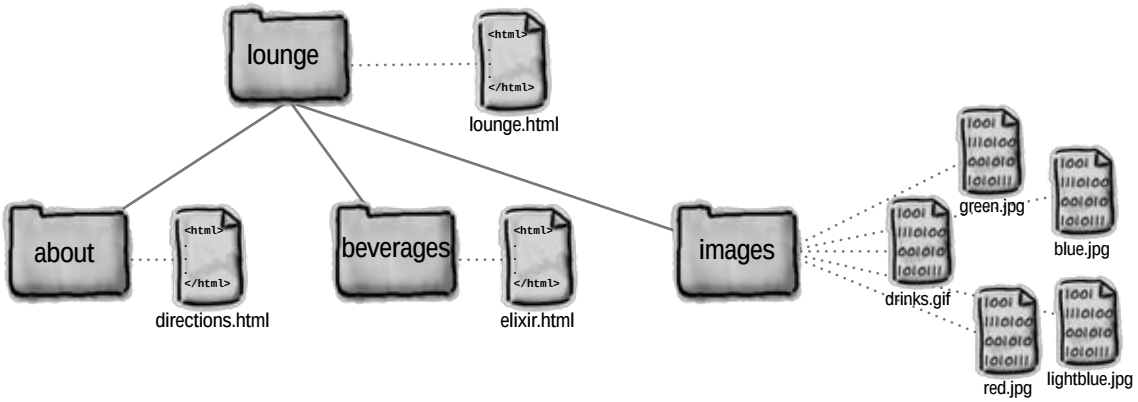
④ Create an href to represent the path we traced.

Now that we know the path, we need to get it into a format the browser understands. Here's how you write the path:

We put the relative path into the href value. Now when the link is clicked on, the browser will look for the "elixir.html" file in the "beverages" folder.

Sharpen your pencil

Your turn: trace the relative path from "lounge.html" to "directions.html". When you've discovered it, complete the **<a>** element below. Check your answer in the back of the chapter, and then go ahead and change both **<a>** elements in "lounge.html."




```
<a  href="_____">detailed directions</a>
```

YOUR ANSWER HERE

Going the other way; linking up into a "parent" folder

1 Linking from "directions.html" to "lounge.html".

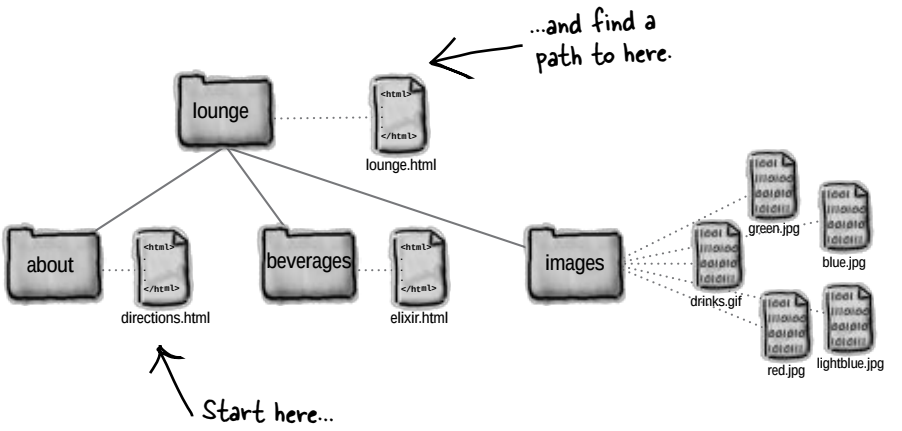
Now we need to fix those "Back to the Lounge" links. Here's what the **<a>** element looks like in the "directions.html" file:

```
<a  href="lounge.html">Back to the Lounge</a>
```

Right now we're just using the filename "lounge.html", which tells the browser to look in the same folder as "directions.html". That's not going to work.

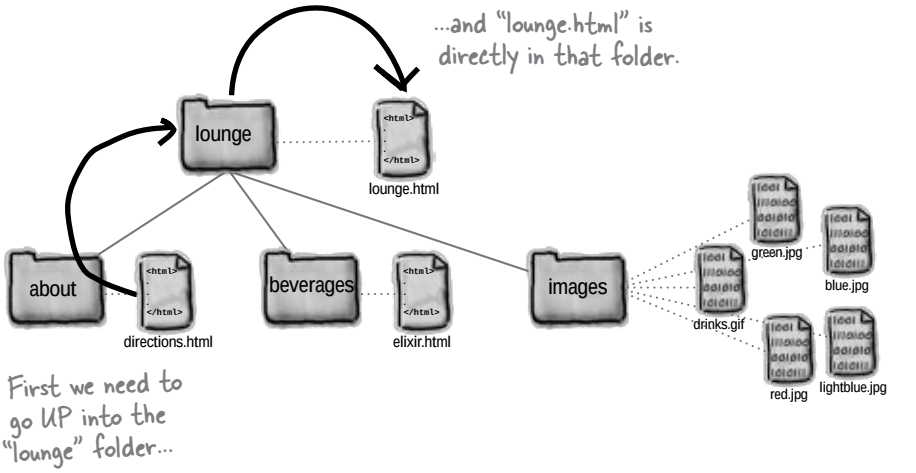
2 Identify the source and the destination.

Let's take a look at the source and destination. The source is now the "directions.html" file, which is down in the "about" folder. The destination is the "lounge.html" file that sits above the "about" folder, where "directions.html" is located.



3 Trace a path from the source to the destination.

Let's trace the path. To get from the "directions.html" file to "lounge.html", we need to go up one folder into the "lounge" folder, and then we'll find "lounge.html" in that folder.



4 Create an href to represent the path we traced.

We're almost there. Now that you know the path, you need to get it into a format the browser understands. Let's work through this:

First you need to go up one folder. How do you do that? With a ".." . That's right, two periods. Go with it, we'll explain in a sec.

Separate all parts of the path with a "/".

Finally you have the file name.

```
.. / lounge.html
```

Pronounce ".." as "dot dot".

Putting it all together...

```
<a  href="../lounge.html">Back to the Lounge</a>
```

Now when you click on the link, the browser will look for the "lounge.html" file in the folder above.

there are no Dumb Questions

Q: What's a parent folder? If I have a folder "apples" inside a folder "fruit", is "fruit" the parent of "apples"?

A: Exactly. Folders (you might have heard these called directories) are often described in terms of family relationships. For instance, using your example, "fruit" is the parent of "apples", and "apples" is the child of "fruit". If you had another folder "pears" that was a child of "fruit", it would be a sibling of "apples." Just think of a family tree.

Q: Okay, parent makes sense, but what is ".."?

A: When you need to tell the browser that the file you're linking to is in the parent folder, you use ".." to mean "move UP to the parent folder." In other words, it's browser-speak for parent.

In our example, we wanted to link from "directions.html", which is in the "about" folder, to "lounge.html", which is in the "lounge" folder, the parent of "about". So we had to tell the browser to look UP one folder. ".." is the way we tell the browser to go UP.

Q: What do you do if you need to go up two folders instead of just one?

A: You can use ".." for each parent folder you want to go up. Each time you use ".." you're going up by one parent folder. So, if you want to go up two folders, you'd type "../..". You still have to separate each part with the "/", so don't forget to do that (the browser won't know what "...." means!).

Q: Once I'm up two folders, how do I **tell the browser where to find the file?**

A: You combine the "../.." with the filename. So, if you're linking to a file called "fruit.html" in a folder that's two folders up, you'd write "../../fruit.html". You might expect that we'd call "../.." the "grandparent" folder, but we don't usually talk about them that way, and instead say, "the parent of the parent folder," or "../.." for short.

Q: Is there a limit to how far up I can go?

A: You can go up until you're at the root of your Web site. In our example, the root was the "lounge" folder. So, you could only go up as far as "lounge".

Q: What about in the other direction – is there a limit to how many folders I can go down?

A: Well, you can only go down as many folders as you have created. If you create folders that are 10 deep, then you can write a path that takes you down 10 folders. But we don't recommend that – when you have that many folder levels, it probably means your website organization is *too* complicated!

In addition, there is a limit to the number of characters you can have in a path: 255 characters. That's a lot of characters, so it's unlikely you'll ever need that many, but if you have a large site, it's something to be aware of.

Q: My operating system uses "\" as a separator; shouldn't I be using that instead of "/"?

A: No; in Web pages you always use "/". Don't use "\". Various operating systems use different file separators (for instance, Windows uses "\" instead of "/") but when it comes to the Web, we pick a common separator and all stick to it. So, whether you're using Mac, Windows, Linux, or something else, always use "/" in the paths in your HTML.

Your turn: trace the relative path from "elixir.html" to "lounge.html" from the "Back to the Lounge" link. How does it differ from the same link in the "directions.html" file?

Answer: It doesn't, it is exactly the same.

Fixing those broken images...

You've almost got the lounge back in working order; all you need to do now is fix those images that aren't displaying.

We haven't looked at the **<img>** element in detail yet (we will in a couple of chapters), but all you need to know for now is that the **<img>** element's **src** attribute takes a relative path, just like the **href** attribute.

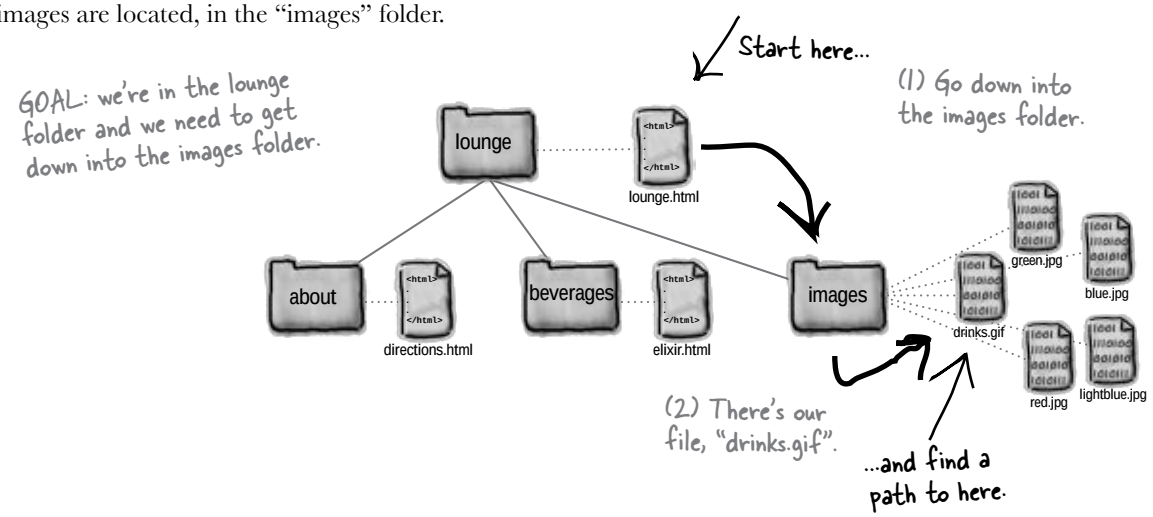
Here's the image element from the "lounge.html" file:

```
<img src="drinks.gif">
```

Here's the relative path, which tells the browser where the image is located. We specify this just like we do with the href attribute in the <a> element.

Finding the path from "lounge.html" to "drinks.gif"

To find the path, we need to go from the "lounge.html" file to where the images are located, in the "images" folder.

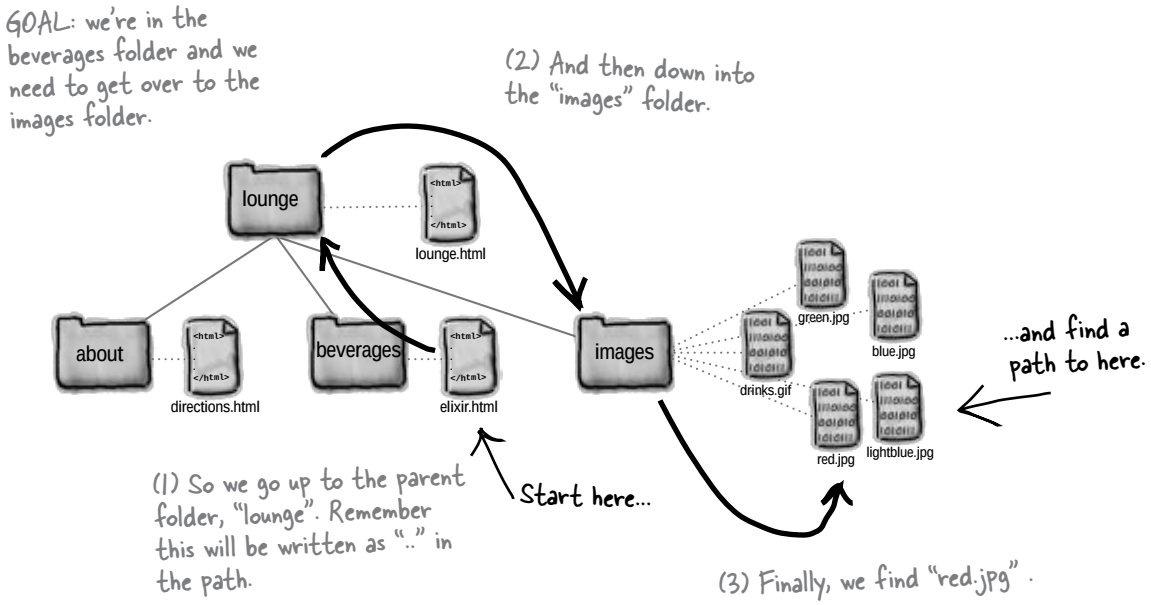


So putting (1) and (2) together our path looks like "images/drinks.gif", or:

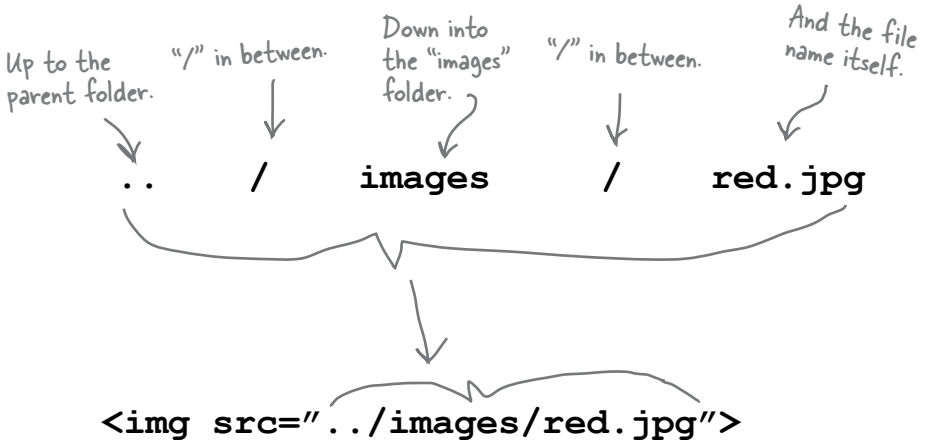
```
<img src="images/drinks.gif">
```

Finding the path from "elixir.html" to "red.jpg"

The elixirs page contains images of several drinks: "red.jpg", "green.jpg", "blue.jpg", and so on. Let's figure out the path to "red.jpg" and then the rest will have a similar path because they are all in the same folder:

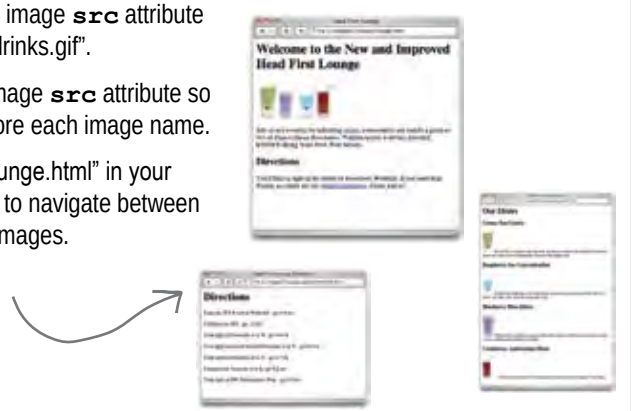


So putting (1) , (2), and (3) together we get:



That covers all the links we broke when we reorganized the lounge, although you still need to fix the images in your "lounge.html" and "elixir.html" files. Here's exactly what you need to do:

1. In "lounge.html", update the image **`src`** attribute to have the value "images/drinks.gif".
2. In "elixir.html", update the image **`src`** attribute so that "../images/" comes before each image name.
3. Save both files and load "lounge.html" in your browser. You'll now be able to navigate between all the pages and view the images.



P.S. If you're having any trouble, the folder "chapter2/completelounge" contains a working version of the lounge. Double-check your work against it.

BULLET POINTS

- When you want to link from one page to another, use the **<a>** element.
- The **href** attribute of the **<a>** element specifies the destination of the link.
- The content of the **<a>** element is the label for the link. The label is what you see on the Web page. By default, it's underlined to indicate you can click on it.
- You can use words or an image as the label for a link.
- When you click on a link, the browser loads the Web page that's specified in the **href** attribute.
- You can link to files in the same folder, or files in other folders.
- A relative path is a link that points to other files on your Web site relative to the Web page you're linking from. Just like on a map, the destination is relative to the starting point.
- Use ".." to link to a file that's one folder above the file you're linking from.
- ".." means "parent folder."
- Remember to separate the parts of your path with the "/" character.
- When your path to an image is incorrect, you'll see a broken image on your Web page.
- Don't use spaces in names when you're choosing names for files and folders for your Web site.
- It's a good idea to organize your Web site files early on in the process of building your site, so you don't have to change a bunch of paths later when the Web site grows.
- There are many ways to organize a Web site; how you do it is up to you.

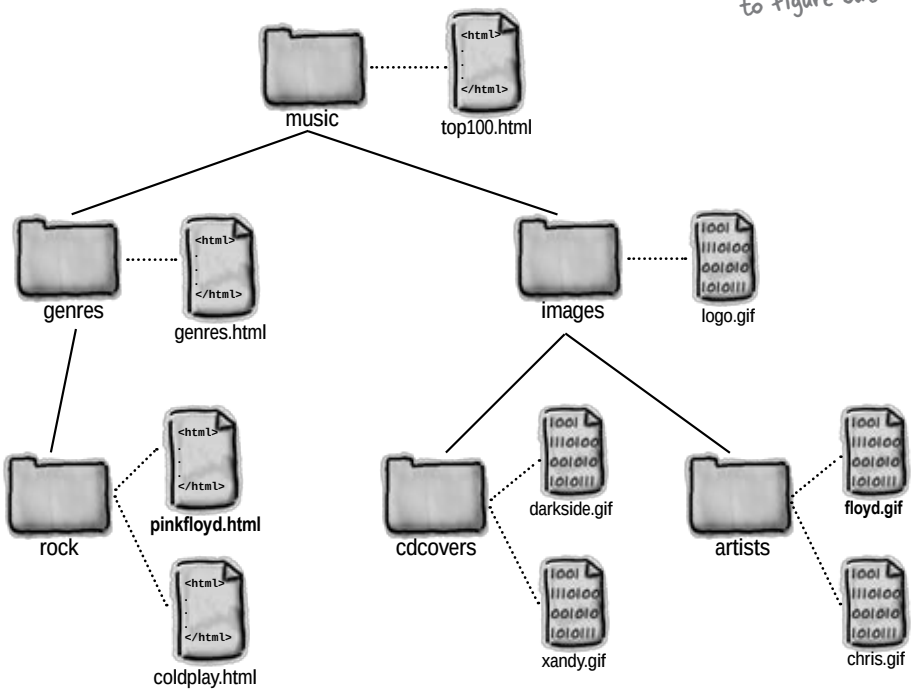
The Relativity Grand Challenge

Here's your chance to put your relativity skills to the test. We've got a Web site for the top 100 albums in a folder named "music". In this folder you'll find HTML files, other folders and images. Your challenge is to find the relative paths we need so we can link from our Web pages to other Web pages and files.

On this page, you'll see the Web site structure; on the next page you'll find the tasks to test your skills. For each source file and destination file, it's your job to make the correct relative path. If you succeed, you will truly be champion of relative paths.

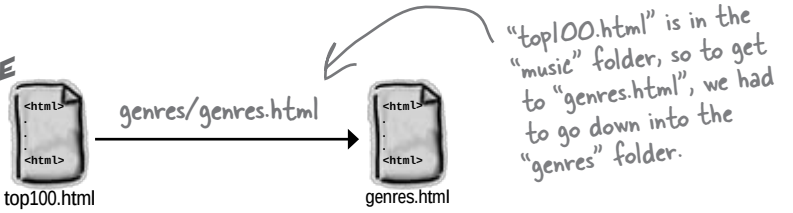
Good luck!

Feel free to draw right on this Web site picture to figure out the paths.

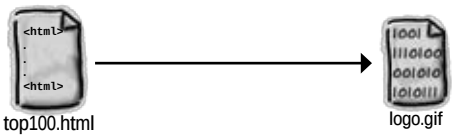


It's time for the competition to begin. Ready... set... write!

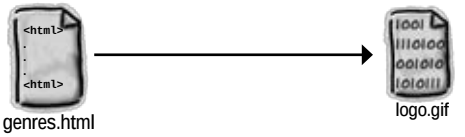
Example



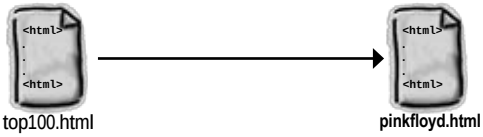
Round One



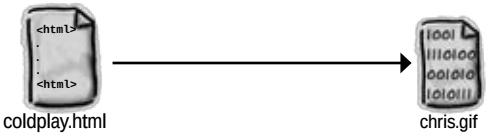
Round Two



Round Three

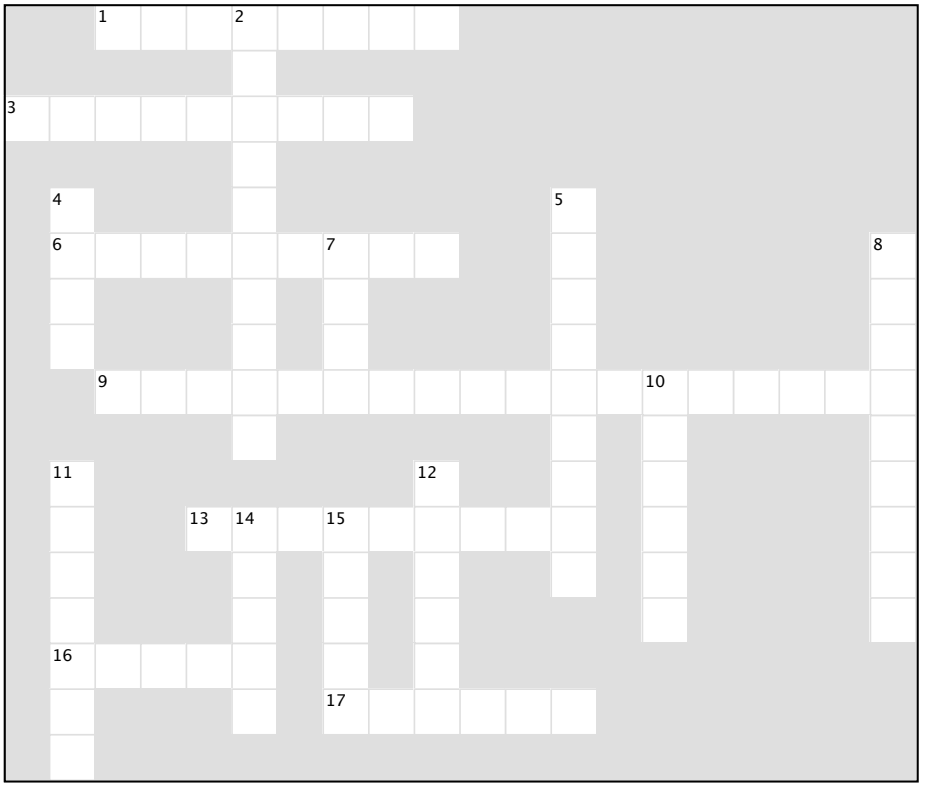


Bonus Round



HTMLcross

How does a crossword help you learn HTML? Well, all the words are HTML-related and from this chapter. In addition, the clues provide the mental twist and turns that will help you burn alternative routes to HTML right into your brain!



Across

1. ../myfiles/index.html is this kind of link.
3. Another name for a folder.
6. Flavor of blue drink.
9. what href stands for.
13. Everything between the <a> and is this.
16. Can go in an <a> element, just like text.
17. Pronounced "..".

Down

2. href and src are two of these.
4. Hardest working attribute on the web.
5. Rhymes with href.
7. Top folder of your site.
8. The "HT" in HTML.
10. Healthy drink.
11. A folder at the same level.
12. Use .. to reach this kind of directory.
14. Text between the <a> tags acts as a _____.
15. A subfolder is also called this.

Exercise Solutions

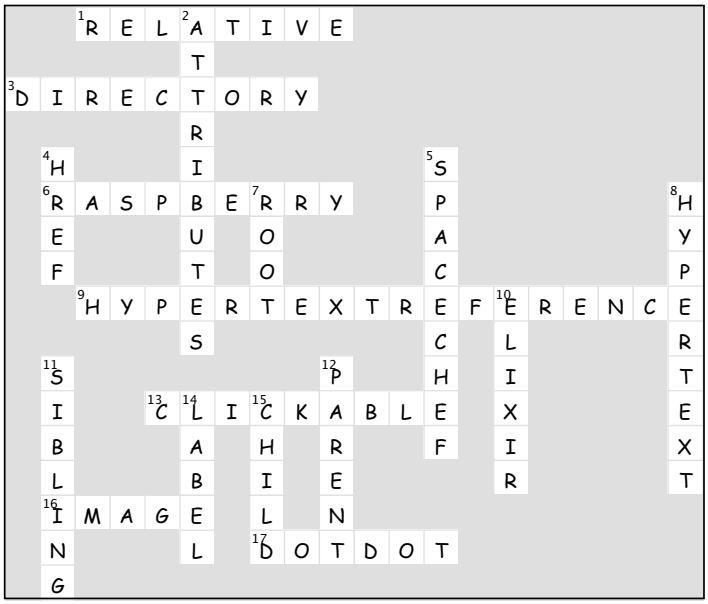
```
<html>
  <head>
    <title>Head First Lounge Elixirs</title>
  </head>
  <body>
    <h1>Our Elixirs</h1>

    <h2>Green Tea Cooler</h2>
    <p>
                <img src="green.jpg">
                Chock full of vitamins and minera
                combines the healthful benefits of
                a twist of chamomile blossoms and
    </p>
    <h2>Raspberry Ice Concentration</h2>
    <p>
                <img src="lightblue.jpg">
                Combining raspberry juice with le
                citrus peel and rosehips, this ic
                will make your mind feel clear an
    </p>
    <h2>Blueberry Bliss Elixir</h2>
    <p>
                <img src="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="red.jpg">
                Wake up to the flavors of cranberry and hibiscus
                in this vitamin C rich elixir.
    </p>
    <p>
        <a href="lounge.html">Back to the Lounge</a>
    </p>
  </body>
</html>
```

We put the link inside its own paragraph to keep things tidy. We'll talk more about this in the next chapter.

Here's the new <a> element pointing back to the lounge.

Exercise solutions

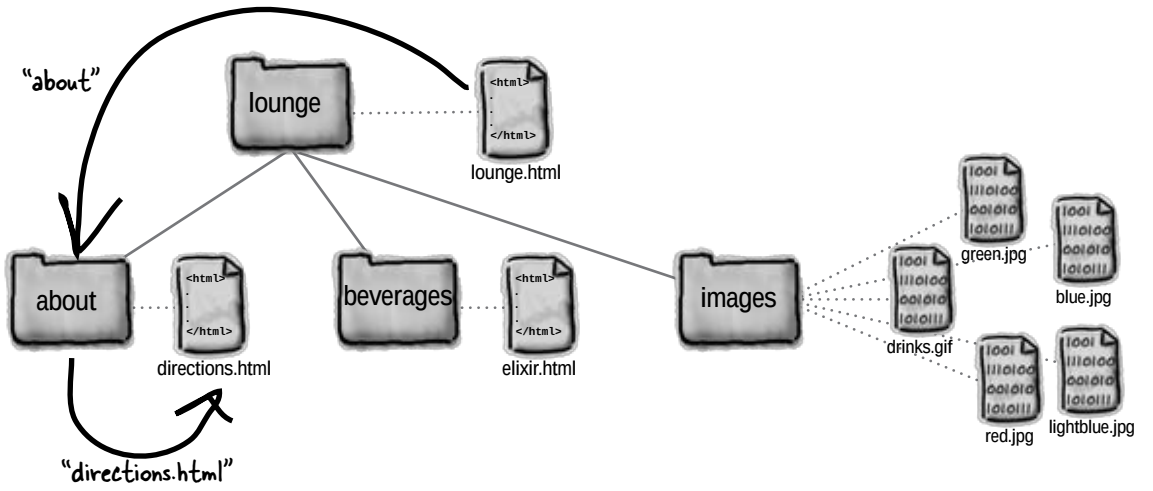
Label	Destination	Element
Hot or Not?	hot.html	`<a href="hot.html">Hot or Not?</a>`
Resume	cv.html	`<a href="cv.html">Resume</a>`
Eye Candy	candy.html	`<a href="candy.html">Eye Candy</a>`
See my mini	mini-cooper.html	`<a href="mini-cooper.html">See my mini</a>`
let's play	millionaire.html	`<a href="millionaire.html">let's play</a>`



Sharpen your pencil
Solution

Trace the relative path from "lounge.html" to "directions.html". When you've discovered it, complete the **<a>** element below.

Here's the solution. Did you change both **<a>** elements in "lounge.html"?




```
<a href="about/directions.html">detailed directions</a>
```

YOUR ANSWER HERE

The Relativity Grand Challenge Solution

Round One

top100.html is in the music folder, so to get to logo.gif, we had to go down into the images folder.

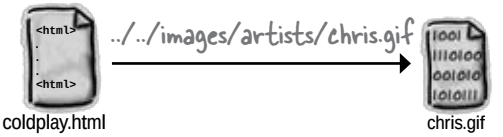
Round Two

genres.html is down in the genres directory, so to get to logo.gif, we first had to go up to music, and then down into the images folder.

Round Three

From top100.html, we go down into genres, then down into rock, and find pinkfloyd.html.

Bonus Round



This was a tricky one. From coldplay.html, which is down in the rock folder, we had to go up TWO folders to get to music, and then go down into images, and finally artists to find the image chris.gif. Whew!