Intermediate CSS3 and HTML5: Lesson 5

Alan Simpson

Chapter 1

Introduction

Welcome back. Today you're going to put some of the things you've learned to practical use by designing some fun features for a site. For starters, I'll show you how to create an interactive FAQs page. Then we'll look at a couple of cool photo thumbnail galleries with larger images that pop-up on hover (mouse-over). And then later in the lesson, I'll show you a method for dealing with troublesome hover techniques on the iPhone and iPad.

You could use all of the items we discuss in this lesson to fill out the entire main content area of a Web page. Remember that in modern layouts, that's the area between the <article> and </article> tags. So if you were to use one of the items we'll go over in this lesson, you'd probably put all the code in that part of a larger layout, and you could put all the style rules into an external style sheet. But to keep things relatively simple and focused, I'm just going to show you the exact code you need for the exact task at hand—not all the other code for the page layout. And I'll also provide complete page examples in the FAQ for this lesson.

So, let's hop over to Chapter 2 and get started with techniques for creating an interactive FAQs page.

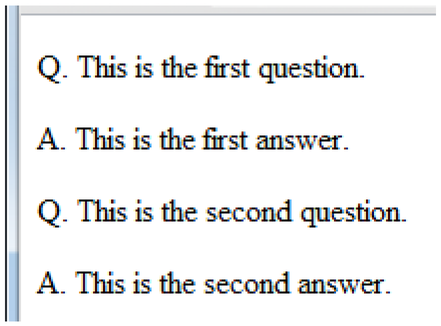
Chapter 2

Fun With FAQs

FAQs (Frequently Asked Questions) are fairly common in websites, and many users will look there before sending in a question. FAQs can be as simple paragraphs, like in the example below.

```
<p>Q. This is the first question.</p>
<p>A. This is the first answer.</p>
<p>Q. This is the second question.</p>
<p>A. This is the second answer.</p>
```

The default styling isn't much—just a blank line between each paragraph.


Q. This is the first question.

A. This is the first answer.

Q. This is the second question.

A. This is the second answer.

Simple FAQs in a browser

There are many ways you could make these more interesting. A good starting point might be to put them into their own div with a class name, like this:

```
<div class="faqs">
<p>Q. This is the first question.</p>
<p>A. This is the first answer.</p>
<p>Q. This is the second question.</p>
<p>A. This is the second answer.</p>
</div><!-- End FAQs -->
```

That div alone wouldn't have any immediate effect on the appearance of the FAQs. But it would make it easy to create some style rules that apply only to paragraphs inside that div. You'll see some examples in this chapter.

Styling the First Letter

The CSS :first-letter pseudo-class allows you to define the style of just the first letter in paragraphs or other elements that contain text. If that first letter is followed by a period, comma, or other punctuation mark, the style is applied to that character, too. The basic syntax is:

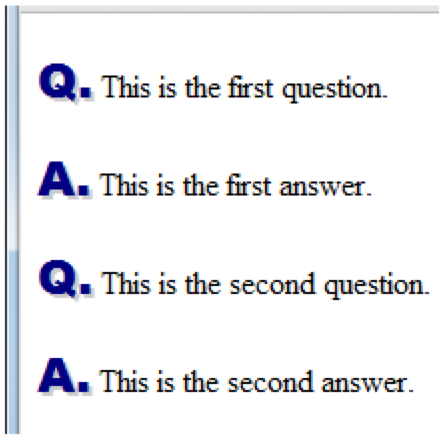
```
type:first-letter{
 ...
}
```

Type is the *type* of element to which you want to apply the styling. Typically, that would be *p* for paragraphs in <p>...</p> tags, but you're free to apply it to other element types. The ... stands for the CSS *property*:*value* pairs that define the style, and there can be any number of them.

For our working example, we can use a descendant selector to apply the styling only to descendants of the faqs class. In other words, make the styling apply only to paragraphs inside the <div class="faqs">...</div> tags rather than to every paragraph on every page. For example, we could use this to style the first letter of every paragraph in the faqs div.

```
.faqs p:first-letter {
  font: bold 1.5em Arial Black, Gadget, sans-serif;
  color: navy;
  text-shadow: 2px 2px #ccc;
 }
```

The .faqs, followed by a space, at the start of the selector limits the scope of the style rule to items inside the element that has the class name faqs assigned to it. The p:first-letter applies to the first letter in each paragraph inside that faqs division. From there you can style the first letter however you like. I used the font:, color:, and text-shadow properties. The result is that the first letter of each paragraph, (and period after it) has styling that's different from the rest of the paragraph.



Q. This is the first question.

A. This is the first answer.

Q. This is the second question.

A. This is the second answer.

First letter styled

Of course, you can style the first letter however you want. And you can put that style rule in your external style sheet, so long as the page that contains the FAQs is linked to that style sheet.

That styling alone may be enough to add some pizzazz to your FAQs. But now that you understand a bit more about visibility, you could make all the answers invisible when the page first opens so only the questions show. The answer will only appear when the user touches the mouse pointer to a question. We'll look at that next.

Making Pop-Up Answers

It might be fun to set up your FAQs so that when the page first opens, only the questions appear. The answer will only appear when the user points to (rests the mouse pointer on) the question. Any time you want things to appear or disappear on mouseover, using CSS, you have to make sure that the item that appears and disappears is contained within the element that makes it appear and disappear. A simple way to do that is to use a div to contain both elements. For example, instead of setting up each of our questions and answers as paragraphs, we can define the entire question and answer pair as a div and only define the answer as a paragraph, like this:

```
<div class="faqs">
  <div><!-- Start a qa -->
  Q. This is the first question.
  <p>A. This is the first answer.</p>
  </div><!-- End a qa -->
  <div><!-- Start a qa -->
  Q. This is the second question.
  <p>A. This is the second answer.</p>
  </div><!-- End a qa -->
 </div><!-- End FAQs -->
```

I've added some comments to help you see how each question and answer pair is defined as a div, with only the answer defined as a paragraph. So the paragraph is now inside the same div that contains the question. And that's important, because both the text that the user points to (the question) and the text that appears when they point (the answer) have to be contained within a single element. You'll see why in a moment.

One bad side effect of changing the tags is that if you were to save the change and look at the page in a browser, the first letter of each question is no longer styled like that answer. The reason for that is pretty plain: the style rule that makes the first letter fancy uses **p:first-letter** as the selector, and after changing the tags as above, only the answer is contained in <p>...</p> tags. But it's not a big problem. We can just change the style rule so it applies to the first letter of every div, as well as the first letter of every paragraph. Here's how the style rule looks after making that change:

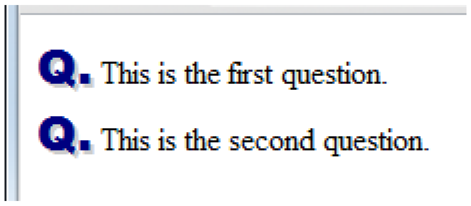
```
.faqs div:first-letter,
 .faqs p:first-letter {
  font: bold 1.5em Arial Black, Gadget, sans-serif;
  color: navy;
  text-shadow: 2px 2px #ccc;
 }
```

That just gets us back to where we were before—the first letter of each question and each answer is styled. To control visibility, we need more style rules.

Within the FAQs section, each answer is a paragraph that's contained within a larger div that also contains the question. So once again, we can use descendant selectors to write a style rule that applies only to paragraphs that are contained within divs. And we can limit it to the FAQs section, but use that class name up front. Here's how that style rule looks. I've put a comment above it to describe its purpose.

```
/* Hide the paragraph within each qa div */
 .faqs div p{
  display: none;
 }
```

With that style rule in play, opening the page displays the questions, but the answers are hidden from view. (If you're actually following along, remember you need to save the style sheet, open the page in the browser, and perhaps refresh or reload the page in the browser as well).


Q. This is the first question.

Q. This is the second question.

Answers hidden

So far so good. There's just one remaining problem. The answers are hidden, but we haven't yet provided any means of making them visible. That part is actually kind of easy, though. All we have to do is write a style rule that says when the user hovers the mouse pointer over a question in the FAQs section, make the answer paragraph for that div visible. And here's how you could type that style rule:

```
/* Show the paragraph when hovering on the qa div */
  .faqs div:hover p {
  display: block;
  }
```

The .faqs div:hover applies to any div inside the FAQs section of the page when the mouse pointer is over the div. The style actually applies to the paragraph (p) that's inside the div. Earlier, we created a style rule that uses display:none to hide the answer. In that new style rule, we're using display:block to make it visible again, but only when the cursor is inside the div that contains the paragraph.

After making and saving that change and refreshing or reloading the page in the browser, the page still opens with all the answers hidden. But touching the mouse pointer to a question makes that answer to that question appear below the question.

To make the section a little more user-friendly, you could add a style rule that replaces the default mouse pointer with a "help" pointer, which shows a little question mark in most systems. Just add this style rule:

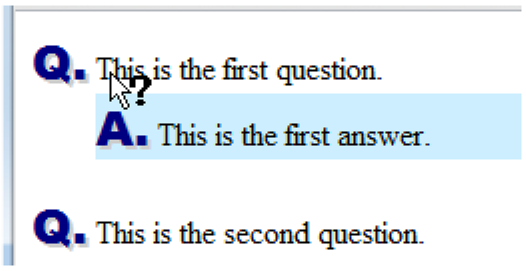
```
/* Style of mouse pointer over qa */
  .faqs div {
  cursor: help;
 }
```

Making the answer visually different from the question can also help the user understand what's going on as they type. For example, in addition to just making the answer visible on hover, you could indent it, give it a background color, and perhaps remove its top margin so it's closer to the question. That would be a matter of adding more descriptors to the style rule that makes it visible, like this:

```
/* Show the paragraph when hovering on the qa div */
 .faqs div:hover p {
  display: block;
  margin-left: 2em;
  background-color: #cef;
  margin-top: 0;
}
```

Style for the answers

With that styling in play, when the user points to a question, the answer stands our more clearly, like this:


Q. This is the first question.

A. This is the first answer.

Q. This is the second question.

Point to question for answer

The exact appearance of the mouse pointer varies depending on the operating system you're using and mouse pointer scheme. But in most cases, it will contain a question mark.

You could also put some instructions above the FAQs list, telling the user what to do. To help with that, and to make sure you can see the big picture in terms of what code goes where, I've included a complete sample FAQs page in the FAQs for this lesson. Of course, you'd still have to come up with your own questions and answers. But all the basic code for the FAQs is there.

On a touchscreen, which has no mouse pointer, tapping the item will have the same effect as hovering with a mouse. Some older Apple iOS devices don't always get it right. But later in this lesson I'll show a small bit of code you can add to your page to make sure the tap acts like a hover on even those older Apple devices.

Let's head over to Chapter 3 now, and I'll show you some other fun ways you can use :hover and visibility.

Chapter 3

Thumbnail Photo Galleries

Thumbnail photo galleries are a fun way to show off favorite photos and other pictures. A thumbnail photo (or just *thumbnail*, for short) is a small image that's about an inch wide on the screen. The basic idea is that you can show many thumbnail images on the page. When the user points to or taps on a thumbnail, an enlarged version of that image pops up.

As a developer, it's important to keep in mind that photo files tend to be large. The larger a file is, the longer it takes to download. Exactly how long depends primarily on two things: the size of the file and the speed of the user's Internet connection. So you always have to be careful not to load up a page with so many pictures that the user loses interest and navigates away to some other page instead of waiting for them to download.

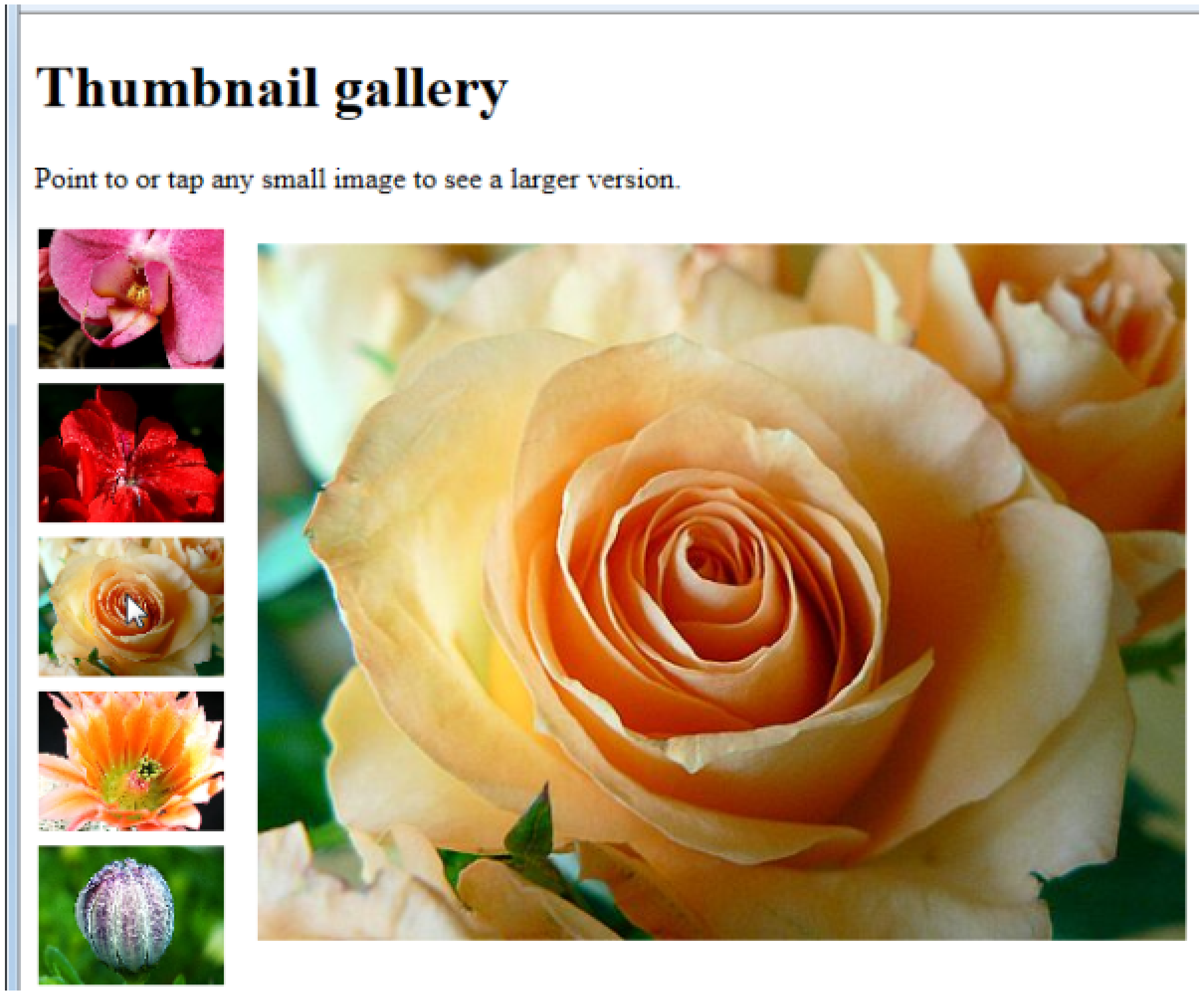
One way to keep an image file to a reasonable size is to limit the dimensions of the picture. You always want to keep the large original of every photo for yourself because that's best for editing and printing. But for use in a website, you'd be wise to make copies that are perhaps in the range of 320 to 640 pixels wide. You can probably put between five and 15 photos of that size on any page and still get a reasonable download time for most users.

Note

You can use most graphics programs, including Photoshop, Paint.Net, Microsoft Paint, and Preview in Mac OS, to make smaller copies of large photos. Or you can use free websites like www.shrinkpictures.com and www.picresize.com.

For practice, I've included a dozen flower photos in the pix2 folder you downloaded for this course. Each is about 500 x 275 pixels in size, with a file size well under 500KB in most cases. We'll use those in a couple of thumbnail gallery examples you may want to incorporate into your own websites.

My first thumbnail example works best for smaller displays and offers five thumbnail images down the side. Pointing to (or tapping) a thumbnail image displays a larger version of it to the right. For example, here the mouse pointer is on the rose thumbnail, so the larger version of that image shows to the right.



A thumbnail gallery example

The code for the gallery can be fairly simple. Create a single large div to contain the whole thumbnail gallery. I opted to give that div a class name of *gallery*. That div in turn contains more divs, and each of those smaller divs contains two img tags, both of which show the same image. (You'll see why two img tags per image are required in a moment).

```
<div class="gallery">
  <div>
  <img src="pix2/flower01.png" alt="">
  <img src="pix2/flower01.png" alt="">
  </div>
  <div>
  <img src="pix2/flower02.png" alt="">
  <img src="pix2/flower02.png" alt="">
  </div>
  <div>
  <img src="pix2/flower03.png" alt="">
  <img src="pix2/flower03.png" alt="">
  </div>
  <div>
  <img src="pix2/flower04.png" alt="">
  <img src="pix2/flower04.png" alt="">
  </div>
  <div>
  <img src="pix2/flower05.png" alt="">
  <img src="pix2/flower05.png" alt="">
  </div>
  </div><!-- End gallery -->
```

I used alt="" in each tag because the alt attribute is required. It contains alternate text that screen readers for the blind can read aloud. But since this is just an example to illustrate a concept, I won't complicate the code by putting in more text. The key thing is that each image in the gallery has two <img> tags in the page—one of which displays the thumbnail, the other displays the larger image. And each pair of img tags is contained in its own smaller div.

Without any CSS styling, that code just puts two copies of each image on the page, with each image at its natural dimensions. That's not good. To create the thumbnail gallery, we need to add some CSS. I'll just show you all of the style rules, and then we'll go through them one-by-one.

```
/* Entire gallery */
  .gallery {
  position: relative;
  }
  /* Smaller divs with 2 img tags each */
  .gallery div {
  padding: 2px;
  width: 120px;
  }
  /* First image in each smaller div */
  .gallery div img:nth-child(1) {
  width: 100px;
  }
  /* Second image in each smaller div */
  .gallery div img:nth-child(2) {
  position: absolute;
  top: 10px;
  left: 120px;
  z-index: 10;
  visibility: hidden;
  }
  /* Hover on any smaller div */
  .gallery div:hover img:nth-child(2) {
  visibility: visible;
  }
```

Let's discuss it one style rule at a time. This first style rule . . .

```
/* Entire gallery */
  .gallery {
  position: relative;
 }
```

. . . applies to the entire gallery, and it just makes it a relatively-positioned element so that we can absolutely position other elements inside of it. Then this next style rule:

```
/* Smaller divs with two img tags each */
  .gallery div {
  padding: 2px;
  width: 120px;
  }
```

. . . applies to each of the smaller divs within the gallery. That style rule just puts a little padding around each div to put some space between each picture. It also limits the width of the div to 120 pixels, which is about the width of each thumbnail photo in this example.

This next style rule applies to the first img tag in each of the smaller divs:

```
/* First image in each smaller div */
  .gallery div img:nth-child(1) {
  width: 100px;
 }
```

That style rule uses the nth-child pseudo element to make the first image inside each smaller div 100 pixels wide. In other words, that style rule makes the first image in each of the smaller divs into a thumbnail-sized image.

This next style rule applies to the second larger image in each div of img tag pairs:

```
/* Second image in each smaller div */
```

```
  .gallery div img:nth-child(2) {
  position: absolute;
  top: 10px;
  left: 120px;
  z-index: 10;
  visibility: hidden;
  }
```

In that style rule, we use img:nth-child(2) to target the styling to the second img tag in each of the smaller divs. That style rule absolutely-positions each of the larger images up near the top left corner of the larger gallery div. I didn't specify a size for those larger images because I already pre-sized them to the size I want them to be on the page. The z-index ensures that the larger image is in front of any other elements on the page. I also made the larger image invisible with visibility:hidden because I don't want it to be visible on the page all the time. I only want it to be visible when the user points to or taps the thumbnail image.

Last but certainly not least, we need a style rule that makes the larger image visible when the user points to the thumbnail. That style rule follows the same pattern as other examples you see. You just have to place a :hover element on the element that's just before the hidden element's selector. And then within that style rule, set the visibility property to visible as we did in the final style rule.

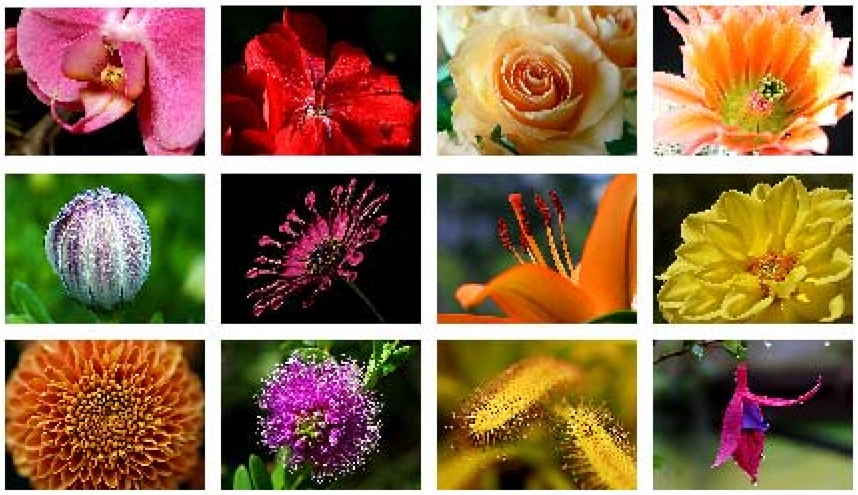
```
/* Hover on any smaller div */
  .gallery div:hover img:nth-child(2) {
  visibility: visible;
  }
```

In that style rule, the first part of the selector, *.gallery div:hover*, says what has to be hovered over in order for the style rule to kick in. In this case, it's any smaller div inside the gallery. The second part of the style rule, *img:nth-child(2)*, says what item receives the styling when the hover kicks in. And in this case, it's the second img tag within the smaller div that's being hovered over.

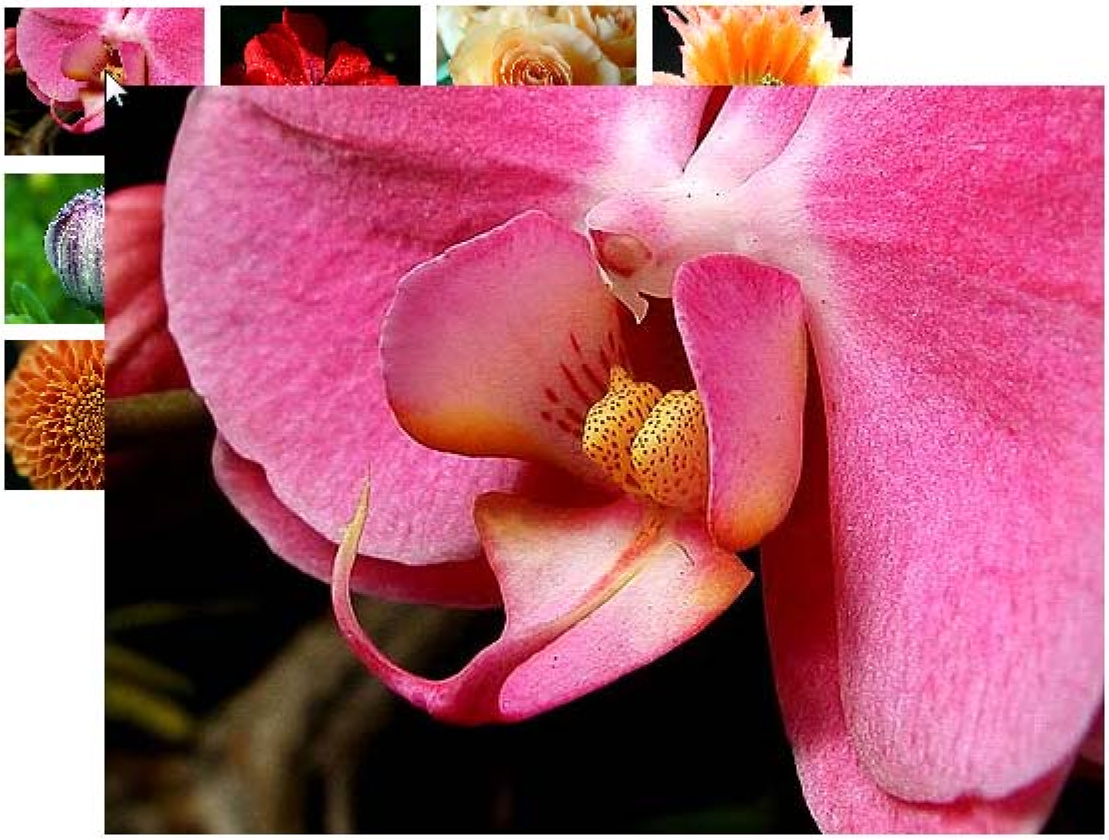
If you have more than just a few images to show in your gallery, you may want to show them in a table near the center of the page rather than a small strip down the left side. Let's look at how you could code that.

Thumbnails Gallery Table

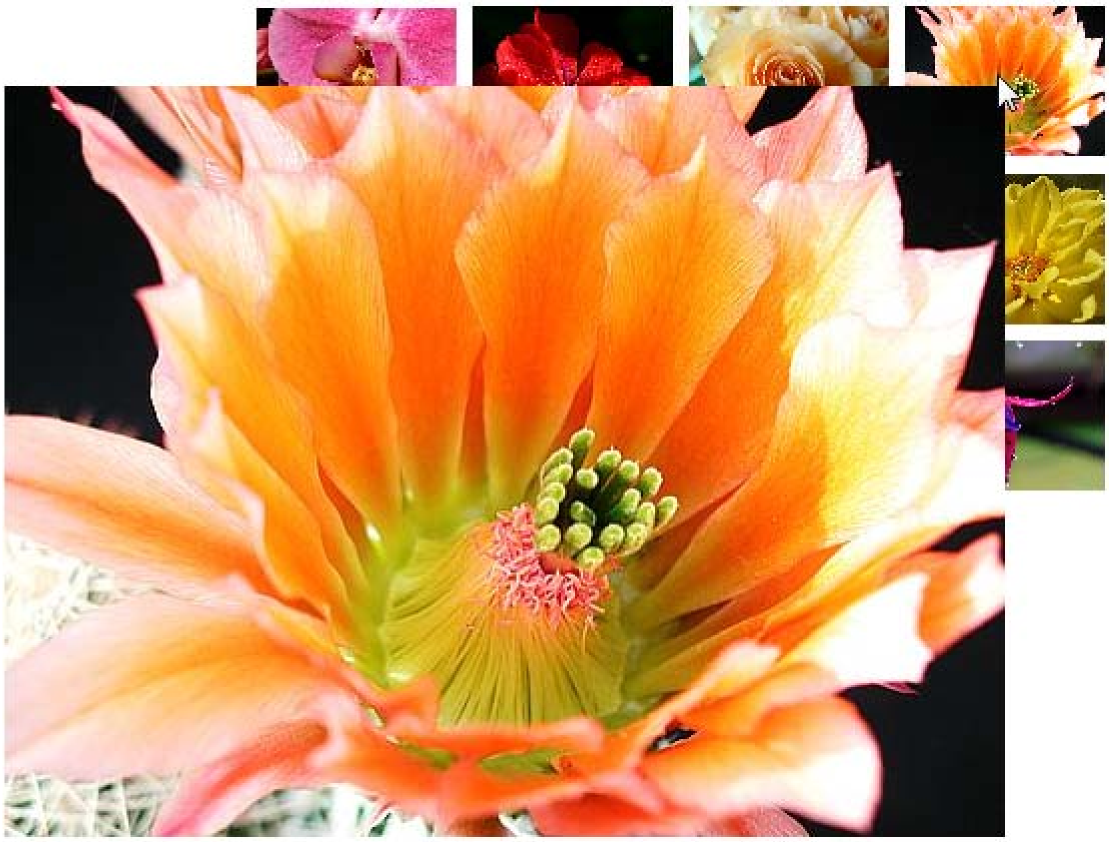
If you want to display more than half a dozen or so thumbnails, you can use a table to arrange them into rows and columns, like this.


A thumbnail gallery in a table

When the user points to (or taps on) a thumbnail image, the larger image appears. To ensure that the larger image fits on the screen even for people using small devices or browsers window, larger images for thumbnails show off to the right of the thumbnail like this:


Larger image to right of thumbnail

When they point to or tap a thumbnail on the right, the larger image shows off to the left.


Larger image to left of thumbnail

For this example, I used a table to display 12 images. The table uses the class name *thumbs*, which I'll use to help define all the style rules in a moment. Each cell contains two img tags for the same image. One img tag contains class="small". That one shows the small thumbnail image. The second img tag in each cell is for the larger version of the same image that appears when the mouse pointer is on the smaller image. That second tag uses class="left" if the cell is in the left half of the table, or class="right" if the cell is in the right half of the table, to show the larger image on one side of the table or the other. The class names *thumbs*, *small*, *left*, and *right* are all class names that I just made up. They could just as easily be *goober*, *gomer*, *howdy*, and *wingnut* or anything else because they're just made-up names. But the names I chose are more descriptive than totally random names, which will hopefully make the code a little easier to understand.

```
<table class="thumbs">
  <tr>
  <td>
  <img src="pix2/flower01.png" alt="" class="small">
  <img src="pix2/flower01.png" alt="" class="left"></td>
  <td>
  <img src="pix2/flower02.png" alt="" class="small">
  <img src="pix2/flower02.png" alt="" class="left"></td>
  <td>
```

```
<img src="pix2/flower03.png" alt="" class="small">
<img src="pix2/flower03.png" alt="" class="right"></td>
<td>
<img src="pix2/flower04.png" alt="" class="small">
<img src="pix2/flower04.png" alt="" class="right"></td>
</tr>
<tr>
<td>
<img src="pix2/flower05.png" alt="" class="small">
<img src="pix2/flower05.png" alt="" class="left"></td>
<td>
<img src="pix2/flower06.png" alt="" class="small">
<img src="pix2/flower06.png" alt="" class="left"></td>
<td>
<img src="pix2/flower07.png" alt="" class="small">
<img src="pix2/flower07.png" alt="" class="right"></td>
<td>
<img src="pix2/flower08.png" alt="" class="small">
<img src="pix2/flower08.png" alt="" class="right"></td>
</tr>
<tr>
<td>
<img src="pix2/flower09.png" alt="" class="small">
<img src="pix2/flower09.png" alt="" class="left"></td>
<td>
<img src="pix2/flower10.png" alt="" class="small">
<img src="pix2/flower10.png" alt="" class="left"></td>
<td>
<img src="pix2/flower11.png" alt="" class="small">
<img src="pix2/flower11.png" alt="" class="right"></td>
<td>
<img src="pix2/flower12.png" alt="" class="small">
<img src="pix2/flower12.png" alt="" class="right"></td>
</tr>
</table>
```

If you were to put that table in a page and view it without any styling, you'd see two copies of each picture in a very large table. The thumbnail sizes, hover action, and general magic of it all requires, as usual, some CSS code. For this particular example, we start off with some general styling for the thumbs table. The margin: 0 auto; for the table sets the top and bottom margins to zero and side margins to auto to center the table horizontally on the page.

```
/* Center the table */
  table.thumbs {
  margin: 0 auto;
  }
```

The second style rule applies to each table cell (each <td>...</td> tag pair) inside the thumbs table. It sets the positioning method to relative, which will allow us to absolutely position the larger image within each cell in later style rules. The padding: 0 2px; puts no padding on the top and bottom of each cell, and 2 pixels of padding on the sides. That's not really necessary, and you could replace those numbers with whatever you think looks good. I just thought those numbers provided nice looking spacing between the images.

```
/* Style for each table cell */
  table.thumbs td {
  position: relative;
  padding: 0 2px;
  }
```

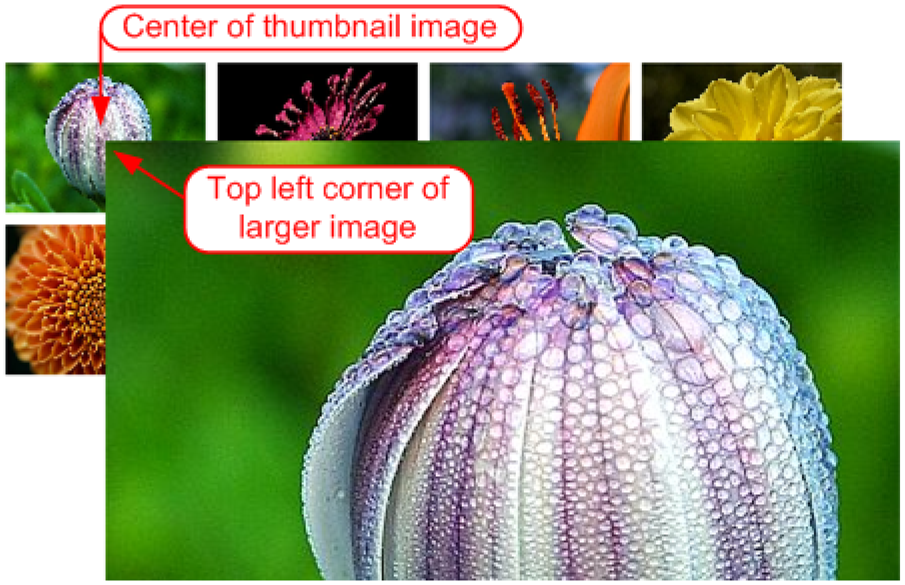
Next we come to a style rule that applies to every <img> tag that contains class="small". It sets the width of each image to 100 pixels, which is a good size for a thumbnail image.

```
/* Small image in each cell */
 table.thumbs img.small {
  width: 100px;
 }
```

Now we can move onto the larger images. For these, I'll need two style rules—one for images in the left side of the table (which have class="left" in their img tags) and another for images on the right side of the table (which have class="right" in their img tags). Here are the style rules for those larger images:

```
/* Large images in left two columns */
  table.thumbs img.left {
  position: absolute;
  top: 50%;
  left: 50%;
  z-index: 10;
  visibility: hidden;
  }
  /* Large images in right two columns */
  table.thumbs img.right {
  position: absolute;
  top: 50%;
  right: 50%;
  z-index: 10;
  visibility: hidden;
  }
```

For both types of images, I use absolute positioning to position the larger image within its cell. For images on the left side, I set the top and left both to 50%, which puts the top left corner in the center of the thumbnail image.



Upper left corner in middle of thumbnail

For images on the right side of the table, I set top and right each to 50%, which puts the top right corner of the larger image in the center of the thumbnail.

Again, the two different ways of showing the larger image is to prevent the entire display when a large image is showing from becoming too wide for narrower screens.

The CSS also gives each of the larger images a high z-index to ensure that they're in front of other content on the page when they're visible. Each large image is initially hidden with visibility:hidden so that when the page first opens, none of the larger images in visible.

The last style rule for the table of thumbnail images looks like this:

```
/* Hover on any table cell */
 table.thumbs td:hover img.left,
 table.thumbs td:hover img.right {
  visibility: visible;
  }
```

That style rule says that whenever the mouse pointer is hovering over a cell in the thumbnails table, make visible the image that has the class name of "left" or the class name of "right" in that cell so the larger image in that cell becomes visible when the mouse pointer is anywhere inside the cell.

Again, I'll provide the complete page for that code in the FAQs for this lesson. But before I do, you need to be aware that the examples may not work as-is on iOS devices like Apple's iPhone, iPad, and iPod Touch. Follow me to Chapter 4 and we'll take care of that pesky little problem right now.

Chapter 4

Making iOS Play Nice

In this chapter, we'll discuss a potential problem with using :hover, and its solution. First, we'll need a little background.

As you probably know, all computers and mobile devices have a central piece of software called an *operating system* (OS) that defines how you interact with the computer or device. Windows is an operating system that's widely used in the business world, as well as in many personal desktop and portable computers. Apple's Mac computers use Apple's Mac OS as their operating system.

Most smartphones and tablets use either the Android or iOS operating system. Android is a Google product, and it's used in Droid phones and popular tablets like the Kindle Fire, Barnes and Noble Nook, Google Nexus, and others. Apple's iPhone, iPad, and iPod Touch all use Apple's iOS operating system.

All of the operating systems have Web browsers, and all are very motivated to be 100% compliant with CSS3 and HTML5 as soon as possible. Though one sticking point has been the fact that the :hover doesn't always work correctly in the Safari browser on iOS devices (iPhone, iPad, iPod Touch). Fortunately, there's a fairly easy workaround involving code that you can pretty much just copy and paste right into any page that used :hover. But in the interest of being professional about it, you should know what you are copying and pasting and why. So before I show you the code, I'd like to explain a bit about what's going on.

JavaScript and jQuery

All Web browsers are capable of executing three languages: CSS, HTML, and JavaScript. You never use one in place of the other. The vast majority of websites use all three languages. Simply stated, we use HTML to describe what an element *is*, CSS to describe how it *looks*, and JavaScript to describe how it *acts*.

You have to learn CSS and HTML before JavaScript, because JavaScript is mostly about manipulating those other two languages. Even though we don't cover JavaScript in great depth in the CSS and HTML courses, we have been forced to use a little bit of it in pages that use the new HTML layout tags. For example, in pages that use the modern HTML5 page layout tags, you have to put this little snippet of JavaScript code right under the <body> tag to help older Web browsers recognize and handle the tags correctly.

```
<script>
 //Make older browsers aware of new HTML5 layout tags
 'header nav aside article footer section'.replace(/\w+/g, function (n) {
document.createElement(n) })
 </script>
```

In addition to JavaScript, there's a product called jQuery that you can use in your pages. Technically, it's still JavaScript, not a language in its own right. In fact, jQuery is a library of prewritten JavaScript code to perform common programming tasks with less code and less effort. It's a freebie that every Web developer can use, no strings attached. jQuery is very popular, and very widely used because it greatly simplifies and speeds up the process of using some common JavaScript capabilities in your pages. In fact, it's becoming so common, it's one of those things that everyone in the Web development biz just assumes you already know and understand, like CSS and HTML.

A full discussion of jQuery is way beyond the scope of a course on CSS and HTML. And it's a very large and feature-rich tool that could be worth some study after you complete this course. But for here and now, our only concern with it will be the extent that it can help to ensure that the :hover capability works correctly in our FAQs and thumbnails examples. And, of course, the same code and concepts will apply to any such code you develop on your own, should you find that any iOS device is ignoring the :hover code.

To use jQuery in a page, the first step is to add the code that makes the jQuery library accessible to the page. That part is pretty easy—you just copy and paste the code that you put right into the head of the page with no tweaking or modification. Here's that code:

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.0/jquery.min.js">
</script>
```

To fix a broken :hover for iOS devices, the next step is to copy and paste all of the code below into the same page, right below the tags above, and still between the <head>...</head> tags in the page:

```
<script type="text/javascript">
 $(document).ready(function () {
 //ipad and iphone fix
 if ((navigator.userAgent.match(/iPhone/i)) || (navigator.userAgent.match(/iPod/i))
|| (navigator.userAgent.match(/iPad/i))) {
 $("PRE:HOVER CSS").click(function () {});
 }
 });
 </script>
```

It's a lot of complex-looking code that basically uses an "if" state to determine if the current device is an iPhone, iPod, or iPad. And if so, it makes a click (tap) kind of work like hover for the *PRE:HOVER CSS* code. The code won't work as-is, though. You have to replace the part that reads PRE:HOVER CSS with part of the CSS selector in the style rule that makes visible whatever you have hidden on your page. I know that's a lot to grasp, so let's take it one step at a time.

In all of our sample pages that show something on hover, there's a style rule that contains :hover in the selector, and that style rule uses display:block or visibility:visible to make visible the part of the element that is initially invisible. All we need from that style rule is everything up to, but excluding, the :hover in the selector at the start of the style rule. That small bit of code has to replace the *PRE:HOVER CSS* in the copy and paste code. Everything else, including the quotation marks and parentheses, stays exactly the same as in the original code.

So, in our sample FAQs page, we use the following style rule to make the FAQ answer visible on hover:

```
/* Show the paragraph when hovering on the qa div */
 .faqs div:hover p {
 display: block;
 margin-left: 2em;
 background-color: #cef;
 margin-top: 0;
 }
```

The selector in that style rule is *.faqs div:hover p*, but for our jQuery code, we only want the part that's before :hover in the selector. So in other words, we only want *.faqs div* out of there, and we want to replace *PRE:HOVER CSS* in the jQuery code with that. So the line in the jQuery code that normally looks like this . . .

```
$("PRE:HOVER CSS").click(function () {});
```

. . . looks like this instead:

```
$(".faqs div").click(function ()  {});
```

That's all there is to it. That would fix the problem in the FAQs page where tapping on a question has no effect when you're using an iPhone, iPad or iPod Touch.

For the thumbnail image examples, the tapping seems to work correctly for me on the iPad even without adding the jQuery code. But just to play it safe, and to provide more examples, you could include the jQuery code on those pages as well. In the page that shows the small strip of thumbnails down the left side of the page, we use this style rule to make a large image visible on hover:

```
/* Hover on any div that contains an image pair */
 .gallery div:hover img:nth-child(2) {
 visibility: visible;
 }
```

In order for that page to work in iOS, we need all the jQuery code in the page, with *.gallery div* filled in where indicated earlier. Which means we need all of this code up between the <head>...</head> tags in that page:

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.0/jquery.min.js"></script>
```

```
 <script type="text/javascript">
 $(document).ready(function () {
 if ((navigator.userAgent.match(/iPhone/i)) || (navigator.userAgent.match(/iPod/i))
|| (navigator.userAgent.match(/iPad/i))) {
 $(".gallery div").click(function () {});
 }
 });
 </script>
```

For the second, larger thumbnail gallery page, we use this style rule to make larger images visible on hover:

```
/* Hover on any table cell */
 table.thumbs td:hover img.left,
 table.thumbs td:hover img.right {
 visibility: visible;
 }
```

Again, we only need the part before the :hover in our jQuery code. And we only need it once. So the jQuery code at the top of the page for that sample code would look like this:

```
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.0/jquery.min.js"></script>
 <script type="text/javascript">
 $(document).ready(function () {
 if ((navigator.userAgent.match(/iPhone/i)) || (navigator.userAgent.match(/iPod/i))
|| (navigator.userAgent.match(/iPad/i))) {
 $("table.thumbs td").click(function () {});
 }
 });
 </script>
```

As mentioned, I'll show you the complete code for all page examples in this lesson in the FAQs for this lesson, including the jQuery code. So don't worry if you're getting a little nervous about remembering where the code goes. In the Lesson 5 FAQ I'll also show you how to speed the loading of the page for testing, in case you notice a slowdown after adding that code. For now, let's pop over to Chapter 5 and review all that you've learned in this lesson.

Chapter 5

Conclusion

Combining advanced techniques like visibility, absolute positioning, and CSS :hover can help you add some interactivity to your Web pages. In today's lesson you saw some examples of that in the form of a FAQs page with answers that appear only when requested, and a couple of thumbnail photo galleries. With computers, simply touching the mouse pointer to an item makes the FAQ answer or larger image pop into view. On devices that don't have a mouse or mouse pointer, tapping the item will usually suffice. Once exception to that rule is iOS devices like the iPhone, iPad, and iPod Touch. Those devices don't always seem to handle the CSS :hover properly. But you can usually fix that with a little bit of jQuery code, as we went over in this lesson.

In the next lesson, you'll learn another advanced technique that you can use to add some real pizzazz to your pages. You'll learn to use fonts that you want in your page without having to rely on the common fonts like Arial, Helvetica, and Times. See you there!

Supplementary Material

CSS Initial Caps

http://webdesign.about.com/od/advancedcss/a/aa090307.htm

Here's an in-depth description of the CSS :first-letter pseudo-class, and some things you can do with it.

Meet the Pseudo Class Selectors

http://css-tricks.com/pseudo-class-selectors/

Click this link for a nice introduction to all the CSS3 pseudo-classes.

Five Cool CSS Hover Effects You Can Copy and Paste

http://designshack.net/?p=19746

As the title implies, here are more fun things you can do with CSS :hover. Make sure you use a modern browser, because some require good CSS3 support.

How To: iOS and CSS Hover Events

http://niteodesign.com/web-design/iphone-ipad-ios-and-the-css-hover-event/

Here's an article on using jQuery to get stubborn :hover events to work in iOS.

iOS

http://www.apple.com/ios/

This is the home page for Apple's iOS operating system.

Android

http://www.android.com/

This is the home page for Google's Android operating system.

Microsoft Windows

http://windows.microsoft.com/en-US/windows/home

This is the home page for the Windows operating system.

Mac OS X

http://www.apple.com/osx/

This is the home page for Apple's Mac OS operating system.

Three Reasons Why You Should Let Google Host jQuery For You

http://encosia.com/3-reasons-why-you-should-let-google-host-jquery-for-you/

Here's an article about using the tags we used in this lesson to load the jQuery library.

jQuery

http://jquery.com/

This is the home of the jQuery JavaScript library. You don't need to learn anything from there now. But you may want to add it to your browser's Bookmarks or Favorites for future reference.

FAQs

Q: All of a sudden it's taking a really long time for my page to open in a browser. What's causing that?

A: If you add the jQuery code for iOS devices to your pages, you may find that it takes much longer to open the page in a web browser when viewing the page on your own computer (as opposed to from a published website). You can speed that up by adding http: to the front of the URL from which the jQuery library is downloaded, like this:

```
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.0/jquery.min.js"></script>
```

The http isn't required on a web server, but usually won't cause any problems there either. However, if you notice problems with the page after you've published to a web server, try removing the http: from the front of the URL and re-publish the page.

More information is available at this page:
http://encosia.com/3-reasons-why-you-should-let-google-host-jquery-for-you/

Q: The interactive FAQs are kind of fun. But what if somebody wants to print the whole page? Wouldn't the printed page just contain the questions?

A: Yes, as it stands, the printed page would contain just the questions. To get around that, you could add the following code to the bottom of the style sheet.

```
/* Styling for printed page only */
@media print {
 .faqs div p {
 display: block;
 }
}
```

It works because the @media print{} block defines a set of styles that apply to the printed page only. Inside its curly braces, I've put a copy of the style rule that makes the answer invisible on the screen. But I changed display:none to display:block, which will make all the answers visible. But again, it applies only on the printed page because it's inside the @media print{...} block.

Q: Can I get a copy of the entire FAQs page's code?

A: Sure, here's all the code, including the code from Chapter 2 and the FAQ above. If you're using a layout, you may want to put all the code that's in the <body>...</body> tags of this page between the <article>...</article> tags of your layout. If you have multiple FAQs page, or you just prefer to keep all your style rules in one place, you can move all the styles rules from the internal style sheet to your external style sheet.

```
<!DOCTYPE html>
<html>
<head>
 <title>FAQs</title>
 <!-- JQuery code to make iOS devices trigger :hover on a tap -->
 <script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.0/jquery.min.js"></script>
 <script type="text/javascript">
 $(document).ready(function () {
 if ((navigator.userAgent.match(/iPhone/i)) || (navigator.userAgent.match(/iPod/i))
|| (navigator.userAgent.match(/iPad/i))) {
 $(".faqs div").click(function () {});
 }
 });
 </script>
 <style type="text/css">
 .faqs div:first-letter, .faqs p:first-letter {
 font: bold 1.5em Arial Black, Gadget, sans-serif;
 color: navy;
 text-shadow: 2px 2px #ccc;
 }
 /* Style of mouse pointer over qa */
 .faqs div {
 cursor: help;
 }
 /* Hide the paragraph within each qa div */
 .faqs div p {
 display: none;
 }
 /* Show the paragraph when hovering on the qa div */
 .faqs div:hover p {
 display: block;
 margin-left: 2em;
 background-color: #cef;
 margin-top: 0;
 }
 /* Styling for printed page only */
 @media print {
 /* Male all answers visible on printed page */
 .faqs div p {
 display: block;
 }
 }
 </style>
</head>
<body>
 <h1>FAQs</h1>
 <p>Here are the answers to frequently asked questions. Touch your mouse pointer to
any question, or tap
 the question on a touchcreen, to see the answer.</p>
 <div class="faqs">
 <div>
 <!-- Start a qa-->
 Q. This is the first question.
 <p>A. This is the first answer.</p>
 </div><!-- End a qa -->
 <div><!--Start a qa -->
 Q. This is the second question.
 <p>A. This is the second answer.</p>
 </div><!-- End a qa -->
 </div>
 <!-- End FAQs -->
</body>
</html>
```

Q: Can I get all the code for the smaller gallery with the images down the left side?

A: Yes, here it is below. As with other examples in this lesson, I've put all the code into a single page. In real life, you'd likely put most of the code that's in the body of this page between the <article>...</article> tags in a page that has a layout. The style rules can go into an external style sheet if you prefer, or if you intend to have multiple thumbnail gallery pages and want to share the style rules across multiple pages.

```
<!DOCTYPE html>
<html>
<head>
 <title>Thumbnail Gallery</title>
<!-- jQuery code to help with :hover on iOS devices -->
 <script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.0/jquery.min.js"></script>
 <script type="text/javascript">
 $(document).ready(function () {
 if ((navigator.userAgent.match(/iPhone/i)) || (navigator.userAgent.match(/iPod/i))
|| (navigator.userAgent.match(/iPad/i))) {
 $(".gallery div").click(function () {});
 }
 });
 </script>
 <style type="text/css">
 /* Entire gallery */
 .gallery {
 position: relative;
 }
 /* Smaller divs with two img tags each */
 .gallery div {
 padding: 2px;
 width: 120px;
 }
 /* First image in each smaller div */
 .gallery div img:nth-child(1) {
 width: 100px;
 }
 /* Second image in each smaller div */
 .gallery div img:nth-child(2) {
 position: absolute;
 top: 10px;
 left: 120px;
 z-index: 10;
 visibility: hidden;
 }
 /* Hover on any smaller div */
 .gallery div:hover img:nth-child(2) {
 visibility: visible;
 }
 </style>
</head>
<body>
 <h1>Thumbnail gallery</h1>
 <p>Point to or tap any small image to see a larger version.</p>
 <div class="gallery">
 <div>
 <img src="pix2/flower01.png" alt="">
 <img src="pix2/flower01.png" alt="">
 </div>
 <div>
 <img src="pix2/flower02.png" alt="">
 <img src="pix2/flower02.png" alt="">
 </div>
 <div>
 <img src="pix2/flower03.png" alt="">
 <img src="pix2/flower03.png" alt="">
 </div>
 <div>
 <img src="pix2/flower04.png" alt="">
 <img src="pix2/flower04.png" alt="">
 </div>
 <div>
 <img src="pix2/flower05.png" alt="">
 <img src="pix2/flower05.png" alt="">
 </div>
 </div><!-- End gallery -->
 <p>This is regular paragraph text after the gallery.</p>
</body>
</html>
```

Q: How about the code for the larger thumbnail gallery. Can I get all of that?

A: Sure. And as with the other examples in this lesson, you can put the content that's in the body of the page between the <article>...</article> tags of a page that has a layout and move the style rules to an external style sheet if you prefer to store them there. Anyway, here's all the code:

```
<!DOCTYPE html>
<html>
<head>
 <title>Thumbnail Gallery</title>
 <!-- jQuery code to help with :hover on iOS devices -->
 <script src="//ajax.googleapis.com/ajax/libs/jquery/1.8.0/jquery.min.js"></script>
 <script type="text/javascript">
 $(document).ready(function () {
 if ((navigator.userAgent.match(/iPhone/i)) || (navigator.userAgent.match(/iPod/i))
|| (navigator.userAgent.match(/iPad/i))) {
 $("table.thumbs td").click(function () {});
 }
 });
 </script>
 <style type="text/css">
 /* Center the table */
 table.thumbs {
```

```
margin: 0 auto;
}
/* Style for each table cell */
table.thumbs td {
position: relative;
padding: 0 2px;
}
/* Small image in each cell */
table.thumbs img.small {
width: 100px;
}
/* Large images in left two columns */
table.thumbs img.left {
position: absolute;
top: 50%;
left: 50%;
z-index: 10;
visibility: hidden;
}
/* Large images in right two columns */
table.thumbs img.right {
position: absolute;
top: 50%;
right: 50%;
z-index: 10;
visibility: hidden;
}
/* Hover on any table cell */
table.thumbs td:hover img.left,
table.thumbs td:hover img.right {
visibility: visible;
}
```

```
</style>
</head>
<body>
 <h1>Thumbnail gallery</h1>
 <p>Point to or tap any small image to see a larger version.</p>
 <table class="thumbs">
```

```
<tr>
<td>
<img src="pix2/flower01.png" alt="" class="small">
<img src="pix2/flower01.png" alt="" class="left"></td>
<td>
<img src="pix2/flower02.png" alt="" class="small">
<img src="pix2/flower02.png" alt="" class="left"></td>
<td>
<img src="pix2/flower03.png" alt="" class="small">
<img src="pix2/flower03.png" alt="" class="right"></td>
<td>
<img src="pix2/flower04.png" alt="" class="small">
<img src="pix2/flower04.png" alt="" class="right"></td>
</tr>
<tr>
<td>
<img src="pix2/flower05.png" alt="" class="small">
<img src="pix2/flower05.png" alt="" class="left"></td>
<td>
<img src="pix2/flower06.png" alt="" class="small">
<img src="pix2/flower06.png" alt="" class="left"></td>
<td>
<img src="pix2/flower07.png" alt="" class="small">
<img src="pix2/flower07.png" alt="" class="right"></td>
<td>
<img src="pix2/flower08.png" alt="" class="small">
<img src="pix2/flower08.png" alt="" class="right"></td>
</tr>
<tr>
<td>
<img src="pix2/flower09.png" alt="" class="small">
<img src="pix2/flower09.png" alt="" class="left"></td>
<td>
<img src="pix2/flower10.png" alt="" class="small">
<img src="pix2/flower10.png" alt="" class="left"></td>
<td>
<img src="pix2/flower11.png" alt="" class="small">
```

```
 <img src="pix2/flower11.png" alt="" class="right"></td>
 <td>
 <img src="pix2/flower12.png" alt="" class="small">
 <img src="pix2/flower12.png" alt="" class="right"></td>
 </tr>
 </table>
 <p>This is regular paragraph text after the gallery.</p>
</body>
</html>
```

Assignment

Below is a sample page that contains some CSS and HTML. A div with the class name captionpic contains a graphic image and another div containing some descriptive text.

```
<!DOCTYPE html>
<html>
<head>
 <title>Captioned Picture</title>
 <style type="text/css">
 /* Applies to any div tag with class="captionpic" */
 div.captionpic {
 position: relative;
 width: 50%;
 margin: 1em auto;
 border: solid 1px silver;
 border-radius: 10px;
 }
 /* Image inside the captionpic div */
 .captionpic img {
 width: 100%;
 display: block;
 border-radius: 10px;
 }
 /* Caption text inside captionpic */
 .captionpic div {
 position: absolute;
 top: 0;
 right: 0;
 width: 38%;
 padding: 0 2%;
 border-radius: 0 10px 10px 0;
 height: 100%;
 background: #ddd;
 font: 10pt/14pt Arial, Helvetica, Sans-serif;
 overflow:auto;
 opacity: .1;
 }
 </style>
</head>
<body>
 <!-- Div contains an image and another div that contains caption text -->
 <div class="captionpic">
 <img src="pix2/raptor.png" alt="F-22 Raptor" />
 <div>
 <h2>F22-Raptor</h2>
 <p>The Lockheed Martin/Boeing F-22 Raptor is a single-seat, twin-engine fifth-
generation supermaneuverable fighter aircraft that uses stealth technology.</p>
 </div><!-- End caption text -->
 </div><!-- End captionpic div -->
</body>
</html>
```

When viewed in a Web browser (assuming the page has access to the pix2 folder), the image and caption text appear on the page. The caption text is faint to let the user know it's there, and to serve as a hint that if they do something, it will be more readable.

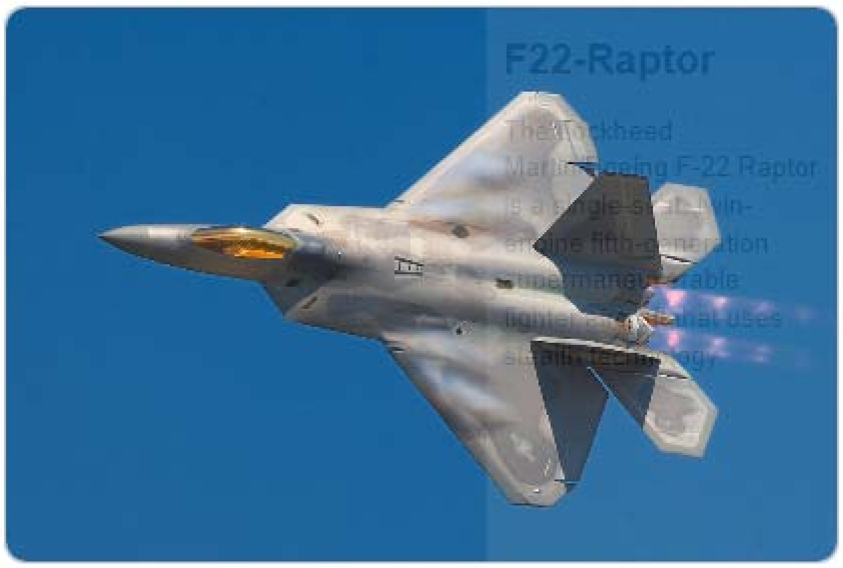
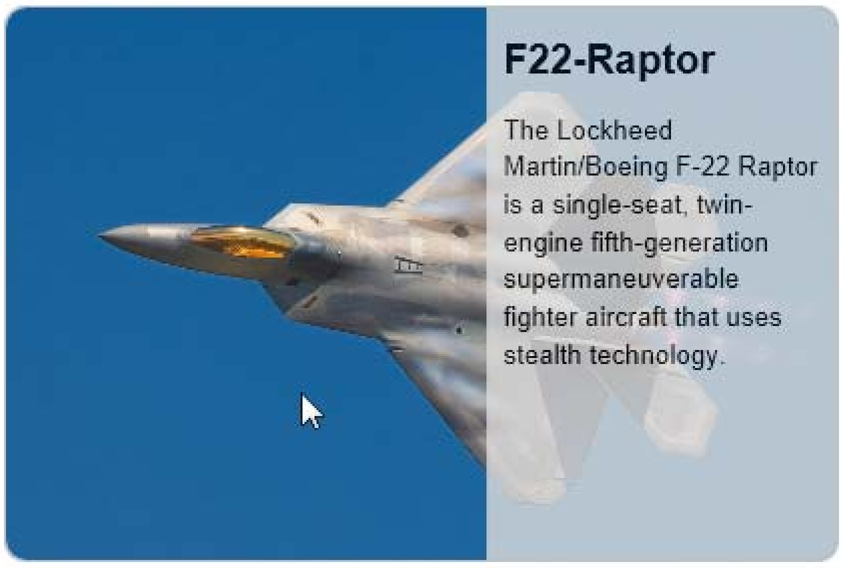

F22-Raptor
The Lockheed Martin/Boeing F-22 Raptor is a single-seat, twin-engine fifth-generation supermaneuverable fighter aircraft that uses stealth technology.

Image with barely visible caption on right

Unfortunately, right now, pointing to the image in the page has no effect because there's no style rule in the page to make the text more readable. Your challenge for this assignment is to add some CSS to the page that increases the opacity of the smaller div to .8 when the user places the mouse pointer anywhere within the captionpic div. Try to do it without peeking at the instructions first. If you do it right, the caption text will be much more opaque when the mouse pointer is on the image, as below.


F22-Raptor
The Lockheed Martin/Boeing F-22 Raptor is a single-seat, twin-engine fifth-generation supermaneuverable fighter aircraft that uses stealth technology.

Mouse pointer on captionpic div

Here are the steps, in case you can't get it to work on your own.

Here are the steps, in case you can't get it to work on your own.

Close

First you'll need to create the page in your HTML5 Intermediate folder. I'll refer to it as *captionpic.htm* but you can actually name it anything you like, so long as it has a .htm or .html extension. You can copy and paste all the code shown in the lesson to that page. Then you'll need to add the following style rule to the internal style sheet:

```
/* Caption text inside captionpic */
.captionpic:hover div {
 opacity: .8;
}
```

It's always a good idea to place the :hover style rule that makes the item visible right below the style rule that makes it invisible. But make sure you start within the <style>...</style> tags that define the internal style sheet as below.

```
/* Caption text inside captionpic */
 .captionpic div {
 position: absolute;
 top: 0;
 right: 0;
 width: 38%;
 padding: 0 2%;
 border-radius: 0 10px 10px 0;
 height: 100%;
 background: #ddd;
 font: 10pt/14pt Arial, Helvetica, Sans-serif;
 overflow:auto;
 opacity: .1;
 }
 /* Caption text inside captionpic */
 .captionpic:hover div {
 opacity: .8;
 }
 </style>
</head>
<body>
```

Save the page after making the change. Then make sure you refresh or reload the page in the browser. Move the mouse pointer onto and off of the image to ensure that your style rule works.