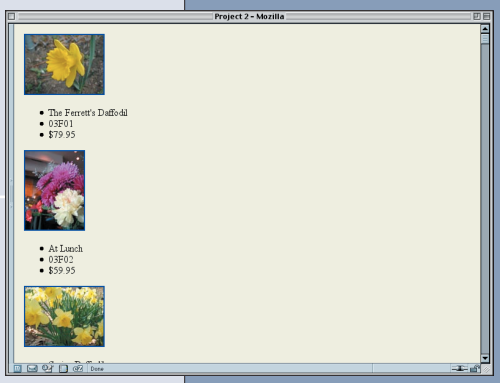
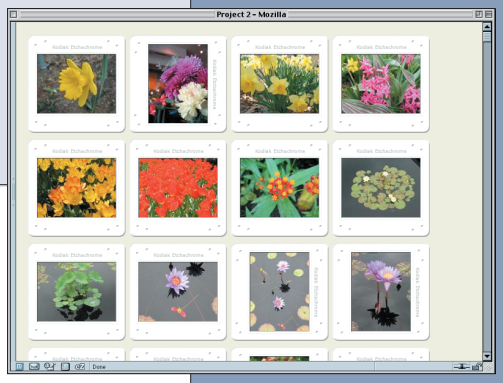
2

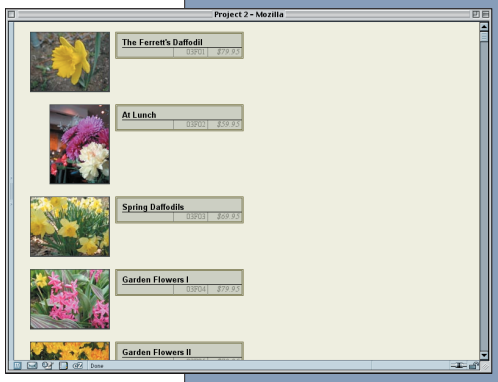
Styling a Photo Collection

All photographs are there to remind us of what we forget. In this—as in other ways—they are the opposite of paintings. Paintings record what the painter remembers. Because each one of us forgets different things, a photo more than a painting may change its meaning according to who is looking at it.

—John Berger







Although not everyone puts his or her photographs online, such collections are an interesting layout challenge. Each photo and its associated information forms a small, self-contained unit that nevertheless has to be laid out with respect to the other photographs on the page. In a way, they're like portals, except with each "box" in this portal leading to more information about a photo instead of to the latest headlines or sports scores.

Photo collections are also reminiscent of another, far more common layout challenge: that of a catalog of products for sale via an e-commerce site. In fact, sometimes the photos themselves can be products for sale, which is the assumption we'll be making in this project.

Project Goals

In this project, we're looking for ways to present a collection of photographs for sale. Our client has given us the following requirements:

- We need to have three different possible presentations: a Contact Sheet view for the artist to check what's available and to show off to his peers, a Gallery view for users to be able to see all the offerings, and a more detailed Catalog view to allow for ordering.
- In the Gallery and Contact Sheet views, as many photographs as possible should appear "above the fold" and without requiring a horizontal scroll, no matter the browser window size. It is acceptable to show only the photo and its title in this view. However, the pictures should arrange themselves into a regular grid.
- In the Catalog view, every photograph should be presented along with its title, catalog number, and price. Scrolling is not a problem in this view.
- The same markup should drive all three views because our client doesn't want to pay for a dynamic site and therefore wants the page markup to be produced only once.

For this project, we're only working on the photo collection piece of the layout, so we don't have to worry about anything but that piece. We will assume that the layout will go into a main central column in a larger layout, but that doesn't really change anything for this project.

Due to the constraints of the project, particularly those of the Gallery and Contact Sheet views, we won't be able to use tables to lay out these photos. Why not? Because of the request to get as many pictures as possible "above the fold" (that is, into the browser window at page load).

So, instead of tables, we'll need to float the pictures and their information for those two "compact" views. Floating them will allow us to get as many pictures in each "row" as will fit in the browser window. In other words, a user with an 800×600 browser window might get four images per row, while a 1280×1024 user will get six or seven. Using floats allows for this kind of "flow" behavior, whereas using tables does not. As an added bonus, we can set up the floats so that each one is the same width. This will ensure that they lay themselves out in a grid-like fashion.

Preparation

Download the files for Project 2 from this book's Web site. If you're planning to play along at home, load the file `ch02proj.html` into the editing program of your choice. This is the file you'll be editing, saving, and reloading as the chapter progresses.

See the Introduction for instructions on how to download files from the Web site.

Laying the Groundwork

The first thing we ought to do is take a look at the markup with which we'll be working. Here are the first two sets of images and information in the document:

```
<div class="pic ls"><a href="orig/img01.jpg" class="tn"><img
  src="tn/thumb01.jpg" alt=""></a><ul>
<li class="title">The Ferrett's Daffodil</li>
<li class="catno">03F01</li>
<li class="price">$79.95</li>
</ul></div>
<div class="pic pt"><a href="orig/img02.jpg" class="tn"><img
  src="tn/thumb02.jpg" alt=""></a><ul>
<li class="title">At Lunch</li>
<li class="catno">03F02</li>
<li class="price">$59.95</li>
</ul></div>
```

That's a lot of classes, and we ought to find out what they all mean. Fortunately, we have a style guide.

- `pic` marks any `div` that contains a picture and its associated information. This helps keep these `div`s separate from any others that might be used.
- `ls` means the picture has landscape orientation (it's wider than it is tall), whereas `pt` refers to a portrait orientation (taller than wide).
- `tn` marks a link as being the hyperlink that's wrapped around the thumbnail image.
- `title` marks the picture's title, `catno` its catalog number, and `price`… okay, that one is fairly obvious.

The really important classes are `ls` and `pt`, as we'll see soon enough, but all of them will come in handy. For example, we'll use those classes to set the height and width of the images. These aren't expressed in the HTML, as you can see. We know that our landscape thumbnails are 128 pixels wide by 96 pixels tall (with the portraits being 96×128), but we'll have to say so in the CSS before the project is done.

Whitespace Blues

Take a close look at the markup: Notice how the `ul` element has been moved right up against the link before it, and the `div`'s closing tag is after it. This was done to avoid certain whitespace-triggered bugs in older versions of Explorer. It's unfortunate, but whitespace is still significant to some older browsers, and sometimes adding or subtracting whitespace can fix mysterious layout problems.

To get started, let's add some basic body and footer styles. For the `body` element, we'll just add a light tan background and some margins. We know we'll be using `float` a lot, and we want the footer to show up after the images, so we'll `clear` it. These actions are illustrated in Figure 2.1.

```
<style type="text/css">
body {background: #EED; margin: 1em;}
div#footer {clear: both; padding-top: 3em;
  font: 85% Verdana, sans-serif;}
</style>
```

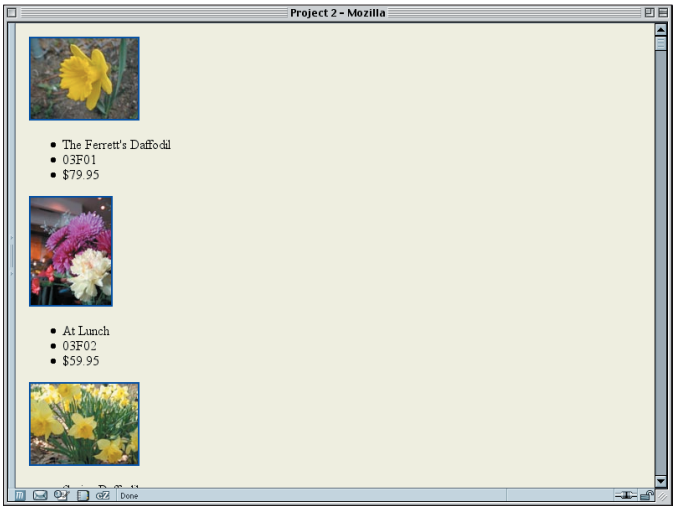


FIGURE 2.1
Taking the first steps.

These styles won't change throughout the rest of the project, so we won't have to worry about them again. With that out of the way, let's get down to business!

CREATING THE CONTACT SHEET VIEW

At this stage, we can see the document structure itself: images followed by unordered lists. What we're after at this point is the creation of a "contact sheet" layout, in which the images are all laid out in a grid. This will let us see as many images as possible at once.

Floating Away

Since we aren't using a table, the obvious solution is to float the images. We know the images are no more than 128 pixels wide or tall, so we'll make our `div`s 128×128 and give them a white background and black border. Speaking of borders, we want to get rid of the blue link-border around the images as well.

Border Incidents

Not all browsers will put a border around a linked image, but some will, so it's a good idea to explicitly get rid of the border. It won't hurt in browsers that didn't have a border in the first place.

```
div#footer {clear: both; padding-top: 3em;
  font: 85% Verdana, sans-serif;}
div.pic {float: left; height: 128px; width: 128px;
  background: white;
  border: 1px solid black;}
div.pic img {border: none;}
</style>
```

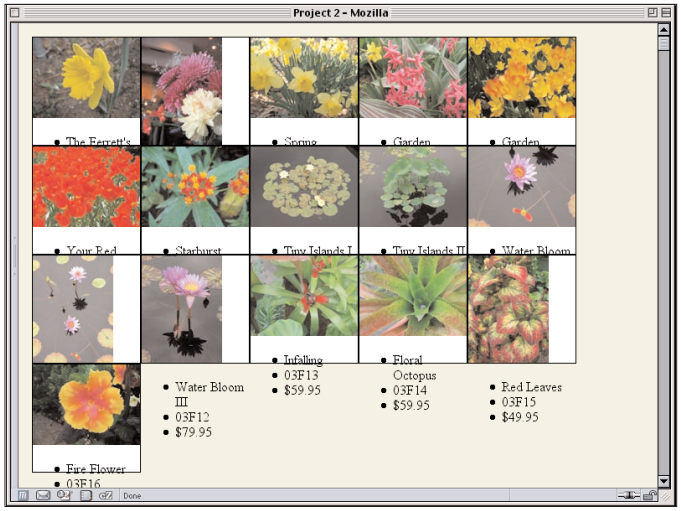


FIGURE 2.2

Floating the picture `divs`.

We've already taken a huge step toward our contact sheet, but there's an obvious problem—the lists! Because we forced the `div`s to be a specific height, there isn't enough room for the lists, so they're spilling out of the `div`s and generally messing up our layout. We need to get rid of them for now, so we'll remove them from the display routines.

```
div.pic img {border: none;}
div.pic ul {display: none;}
</style>
```

This will keep the information from being displayed at all. We'll bring back the lists in future changes to the styles, but for now we'll just get rid of them.

Spacing and Centering

Thanks to the floating, we now have our images forming a grid, but it feels a little too confined. Let's spread out the pictures a bit by adding margins to the floats.

```
div.pic {float: left; height: 128px; width: 128px;
  margin: 5px 3px; background: white;
  border: 1px solid black;}
```

Explorer Watch

In Explorer for Windows, the lists will expand the `div`s instead of sticking out of them, leading to a very different but equally bad layout. This is a bug in Explorer 5+ that treats `height` as if it were `min-height` (which, ironically, Explorer doesn't support).

Because margin floats don't collapse together, the actual spacing between two floats sitting next to each other will be 6 pixels (`3px` + `3px`), and there will be 10 pixels between a float and the one below or above it. This can be adjusted to whatever spacing one prefers, of course. What we have so far is shown in Figure 2.3.

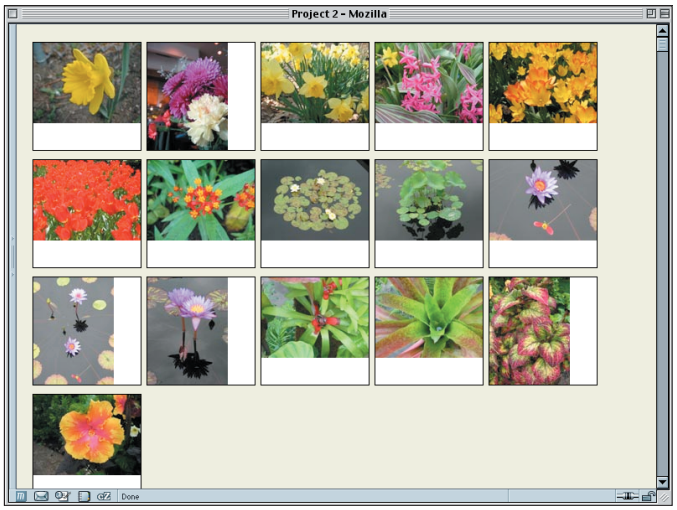


FIGURE 2.3
Spreading the thumbnails apart.

It's getting better and better, but the images look kind of weird as they are now, either up against the top of the box or against the left edge. They'd look much better centered within each box.

To do this, we should first define the size of the images. We can do this with two simple (and very similar) rules.

```
div.pic img {border: none;}
div.ls img {height: 96px; width: 128px;}
div.pt img {height: 128px; width: 96px;}
div.pic ul {display: none;}
```

All we've done here is express what we already knew to be true but the browser didn't: that landscape (`ls`) images are 96 pixels tall by 128 pixels wide and portrait (`pt`) images are the other way round.

Recall that our `div`s have been defined as 128×128 pixels. Now all we need to center the images are some margins on the images themselves. The difference between 128 and 96 is 32, and half of that is 16. Therefore, landscape images will need `16px` of top and bottom margin, and portraits will need `16px` of left and right margin. The result is shown in Figure 2.4.

```
div.ls img {height: 96px; width: 128px; margin: 16px 0;}
div.pt img {height: 128px; width: 96px; margin: 0 16px;}
```

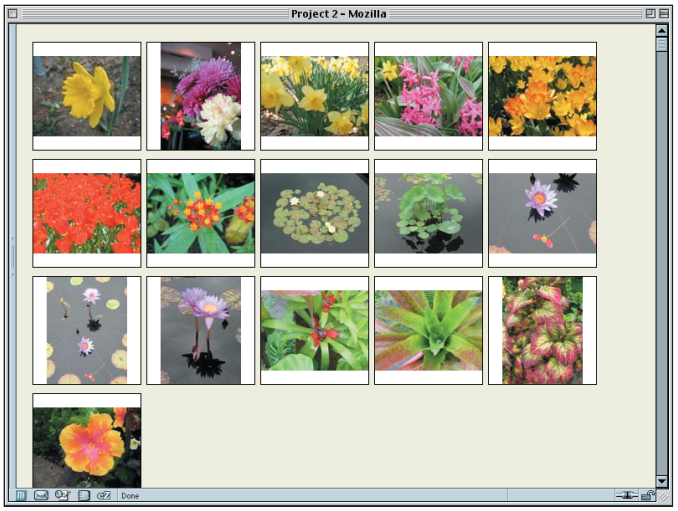


FIGURE 2.4

Centering the thumbnails.

Sliding into Style

That's already a decent layout, but let's take it a little further. Let's add some extra padding to the `div`s so that the thumbnails have white all the way around. We'll stick with our powers-of-two motif and add 16 pixels of padding.

```
div.pic {float: left; height: 128px; width: 128px;
  padding: 16px; margin: 5px 3px; background: white;
  border: 1px solid black;}
```

At this point, the contact sheet is beginning to resemble a collection of 35mm slides, so let's run with that idea. First we'll add a border back onto the images, one that makes them look like they've been inset into a slide frame.

```
div.pic img {border: 1px solid;
 border-color: #444 #AAA #AAA #444;}
```

These changes give all the images a dark gray top and left border, with a light gray right and bottom border. That's a decent enough inset effect.

However, there's a subtle imbalance that's been introduced by this rule. The images plus the borders are now longer than 128×96 pixels and vice versa; now they're 130×98 pixels because the borders are added to the height and width of the images. To address this, we'll need to alter the height and width declarations for the `div`s as well as the padding.

```
div.pic {float: left; height: 130px; width: 130px;
  padding: 15px; margin: 5px 3px; background: white;
  border: 1px solid black;}
```

Another Approach

Instead of the technique shown, we could have used a single color and the border style inset, but there's a drawback to this approach: Browsers are allowed to modify the colors for `inset` (as well as `outset`, `groove`, and `ridge`) however they like. Predictably, they all do it differently. Thus, in cases in which the shading of a border is important, it's better to make the border solid and set the colors the way *you* want them.

With these small changes, we've restored the balance we had before. To finish the effect, let's create an outset effect for the border of the `div`s. All we need there is the same colors we used for the inset effect, except reversed.

```
div.pic {float: left; height: 130px; width: 130px;
  padding: 15px; margin: 5px 3px; background: white;
  border: 1px solid; border-color: #AAA #444 #444 #AAA;}
```

Note that we removed the keyword `black` from the `border` declaration. It isn't needed any longer thanks to the `border-color` declaration, so to save on file size, it was taken out. We can see the result in Figure 2.5.

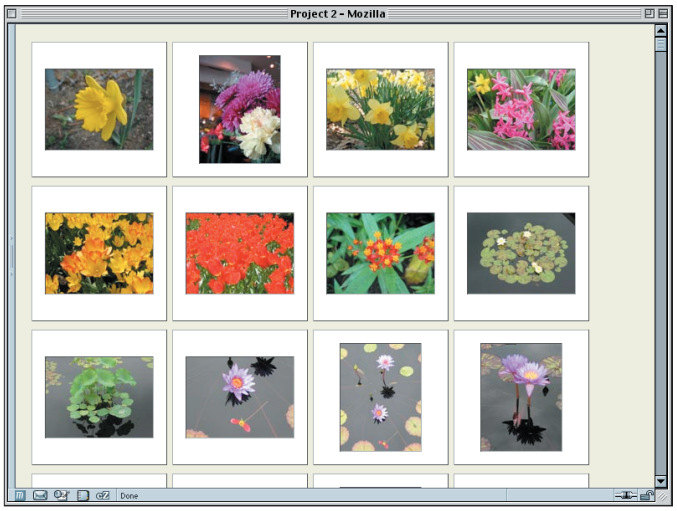


FIGURE 2.5

A sheet of slides, or something very much like it.

That's pretty close to looking like 35mm slides, but we can take the effect even further. Instead of relying on background color and borders, we can remove those and use a background image instead.

How? We already know the dimensions of the `div`s once all is said and done: They're 162×162 pixels. For example, landscape slides have the following dimensions along the horizontal axis:

128px `img` width + 2px `img` border + 30px `div` padding + 2px `div` border = 162 pixels total

Since we're going to be removing the borders on the `div`s, we can knock those off, leaving us with 160×160 pixels. Therefore, we'll make the background images for the `div`s that size. Because we have two kinds of images (portrait and landscape), we'll need two different backgrounds. We'll call them `frame-pt.gif` and `frame-ls.gif`. Whether they're created by scanning actual 35mm slide frames or by creating facsimiles in Photoshop is irrelevant. All we need is something that looks the part.

You can find images named `frame-ls.gif` and `frame-pt.gif` in the project files available on the book's companion site.

Once we've created the background images we need, all that remains is to add them to the styles. We'll start by applying the same image to all the `div`s, as well as removing our border styles.

```
div.pic {float: left; height: 130px; width: 130px;
  padding: 15px; margin: 5px 3px;
  background: url(frame-ls.gif) center no-repeat;}
```

This is great for the landscape thumbnails, but it will look weird when applied to the portrait images. So we just substitute the image we want used for portrait thumbnails with a short rule.

```
div.pic {float: left; height: 130px; width: 130px;
  padding: 15px; margin: 5px 3px;
  background: url(frame-ls.gif) center no-repeat;}
div.pt {background-image: url(frame-pt.gif);}
div.pic img {border: 1px solid; border-color: #444 #AAA #AAA #444;}
```

This will override the background image value while leaving the other keywords (`center no-repeat`) alone, making the declaration `background-image: url(frame-pt.gif)` functionally equivalent to declaring `background: url(frame-pt.gif) center no-repeat`. Figure 2.6 illustrates the result.

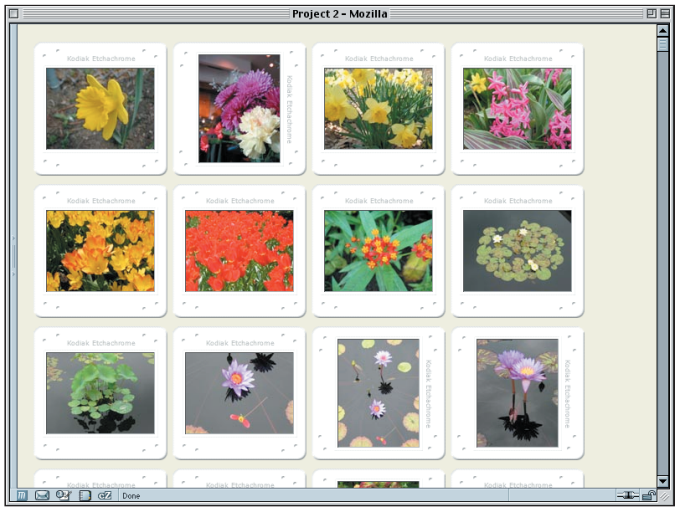


FIGURE 2.6

Now it really looks like a bunch of slides!

That's a pretty nifty effect, eh? The great thing is that you can substitute the "frame" background with a better one later on just by updating the image files.

Listing 2.1 shows the slide-collection style sheet we've created in its entirety.

This might be a good time to save the work you've done in a separate file because the next section will remove many of these styles even as it adds new ones.

Listing 2.1 The Complete "Slides" Style Sheet

```
body {background: #EED; margin: 1em;}
div#footer {clear: both; padding-top: 3em;
  font: 85% Verdana, sans-serif;}
div.pic {float: left; height: 130px; width: 130px;
  padding: 15px; margin: 5px 3px;}
  background: url(frame-ls.gif) center no-repeat;}
div.pt {background-image: url(frame-pt.gif);}
div.pic img {border: 1px solid; border-color: #444 #AAA #AAA #444;}
div.ls img {height: 96px; width: 128px; margin: 16px 0;}
div.pt img {height: 128px; width: 96px; margin: 0 16px;}
div.pic ul {display: none;}
```

Creating the Gallery View

As interesting as the preceding style sheet may be, the drawback is that it doesn't show us what photos are called. This might be a perfect presentation for the artist himself, who probably knows what they're all called and just wants to see everything in a compact form. For visitors, though, it won't be nearly as useful. Therefore, let's turn our slides into a gallery of photos, each one captioned with its title.

Removing the Slide Styles

To clear the field, so to speak, we'll want to drop the styles that place the background images into place (see Figure 2.7). This leaves us with the style sheet in Listing 2.2.

Listing 2.2 The Reduced Style Sheet

```
body {background: #EED; margin: 1em;}
div#footer {clear: both; padding-top: 3em;
  font: 85% Verdana, sans-serif;}
div.pic {float: left; height: 130px; width: 130px;
  padding: 15px; margin: 5px 3px;}
div.pic img {border: 1px solid; border-color: #444 #AAA #AAA #444;}
div.ls img {height: 96px; width: 128px; margin: 16px 0;}
div.pt img {height: 128px; width: 96px; margin: 0 16px;}
div.pic ul {display: none;}
```

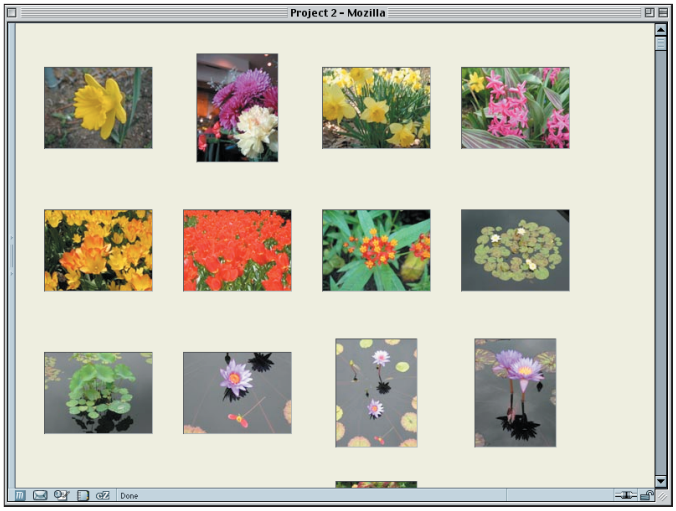


FIGURE 2.7

Stripping out the slide frames.

With this style sheet, we're actually most of the way to where we want to be. The next step is to reveal the titles.

Titular Revelations

To make this happen, we'll need to unhide the `ul` elements and hide the parts we don't want to see at this point. The unhiding part is simple enough: We remove `display: none;` from the `div.pic ul` rule. Of course, that leaves us with an empty declaration block:

```
div.pic ul {}
```

This is legal, even if it's kind of useless: The elements will be selected, but no styles will be applied. So let's fill in the blank. We'll set the padding and margins for the `ul` as well as its font styles.

```
div.pic ul {margin: 0.25em 0 0; padding: 0;
  font: bold small Arial, Verdana, sans-serif;}
```

Now the list will be visible again—all of it. Since we only want the titles to appear, we'll hide the other list items, as shown in Figure 2.8.

```
div.pic ul {margin: 0.25em 0 0; padding: 0;
  font: bold small Arial, Verdana, sans-serif;}
li.catno, li.price {display: none;}
</style>
```

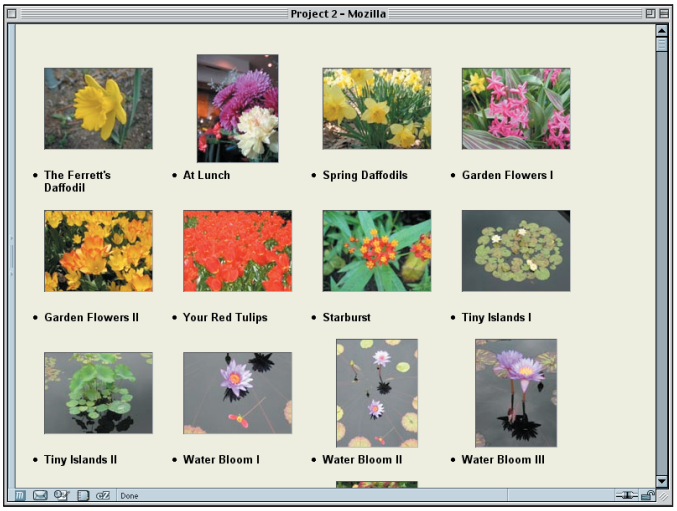


FIGURE 2.8

Revealing the title while hiding the rest.

Not bad, but there's obviously still some work to do. We'll want to get rid of the bullets since they only serve to distract and look ugly. We could remove them using the property `list-style`, but on the other hand we could just change the `li` element so that it's no longer a list item. This would work great, except that IE/Win preserves the bullets, so we'll actually want to do both. While we're at it, we'll center the text.

```
div.pic ul {margin: 0.25em 0 0; padding: 0;
  font: bold small Arial, Verdana, sans-serif;}
li.title {display: block; list-style: none; text-align: center;}
li.catno, li.price {display: none;}
```

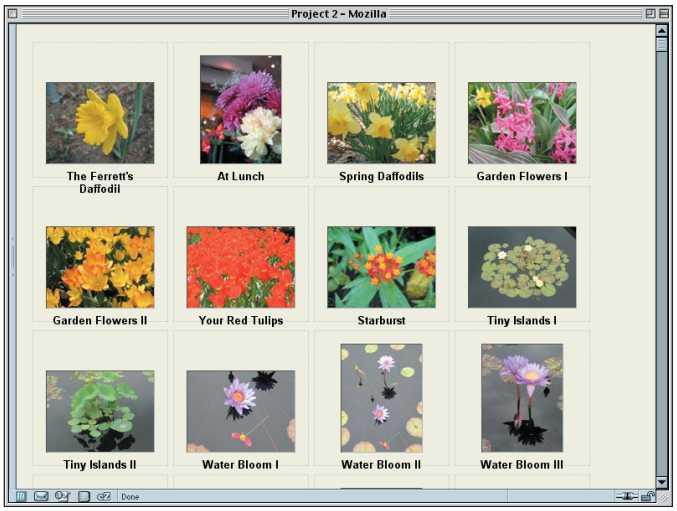
Now we'll move the titles up close to the images. If you remember, we added margins to the images while creating the "slides" style sheet in the preceding section. That margin is what is pushing the titles away from the landscape images.

Note that I say the *landscape* images. The portrait images have no top or bottom margins. Thus, we really only need to alter the margins on landscape images. Also note something interesting in Figure 2.8: The first lines of the captions in each row all line up. That's an effect worth preserving, so rather than just removing the bottom margin on landscape images, we're going to shift the entire margin to the top.

```
div.ls img {height: 96px; width: 128px; margin: 32px 0 0;}
```

So far so good, but there's something we haven't considered yet: the height of the `div`s. They're still locked at `130px` tall plus some padding. All of that together means the captions are in danger of spilling out of the `div`s. If we add a subtle border to the `div`s, we can see what's happening without detracting too much from the overall progress we've made, as shown in Figure 2.9.

```
div.pic {float: left; height: 130px; width: 130px;
  padding: 15px; margin: 5px 3px; border: 1px dotted silver;}
```



Explorer Watch

As before, the `height` bugs in Explorer for Windows will lead to a different result than that shown in Figure 2.9.

FIGURE 2.9

We're almost there, but there are some excesses.

Cleaning Up

At this stage, all we really need to do is increase the height of the `div`s so that the text has room to fit. Why bother? One reason is that, if we don't, the layout will be broken in Internet Explorer for Windows. Another reason is that, if a long title appears above a portrait image, the chance of overlap is fairly high. A third reason is that it just seems untidy to have elements intentionally sticking out of their parent elements (for the moment, anyway; we'll actually take advantage of that behavior in a later phase of the project).

So what we'll do is remove the top and bottom padding of the `div`s and increase their `height`. Oh, and remove that dotted border while we're at it.

```
div.pic {float: left; height: 190px; width: 130px;
  padding: 0 15px; margin: 5px 3px;}
```

With this change, we've arrived at our gallery style sheet, provided in Listing 2.3 and displayed in Figure 2.10.

This might be a good time to save the work you've done in a separate file because the next section will effectively start over, tossing out most of what we've done so far.

Listing 2.3 The Gallery Style Sheet

```
body {background: #EED; margin: 1em;}
div#footer {clear: both; padding-top: 3em;
  font: 85% Verdana, sans-serif;}
div.pic {float: left; height: 190px; width: 130px;
  padding: 0 15px; margin: 5px 3px;}
div.pic img {border: 1px solid; border-color: #444 #AAA #AAA #444;}
div.ls img {height: 96px; width: 128px; margin: 32px 0 0;}
div.pt img {height: 128px; width: 96px; margin: 0 16px;}
div.pic ul {margin: 0.25em 0 0; padding: 0;
  font: bold small Arial, Verdana, sans-serif;}
li.title {display: block; text-align: center;}
li.catno, li.price {display: none;}
```

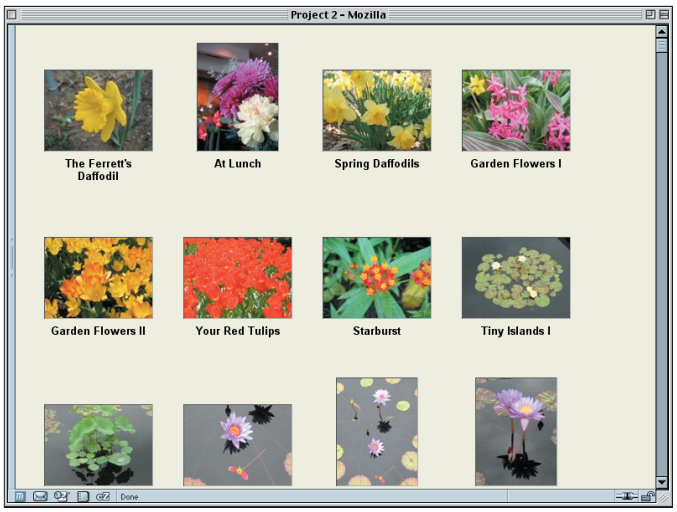


FIGURE 2.10
A gallery of garden delights.

Height and Tables

A value of 190px was chosen to give enough room for the captions to be placed even if they go to three lines, but there's a potential flaw lurking there. Suppose the user resizes the text or a caption goes to five or six lines. If either happens, the layout will effectively be broken, with no real way to fix it.

This is an inescapable limitation of floated elements. In effect, every element is an island unto itself: Its dimensions are completely free of association with any other element. Put another way, the only elements in all of Web design that will size themselves to match their neighbors are table cells.

This means that there are times when a table is the best layout for a photo gallery. If you have a situation in which your captions are of unpredictable length, or if you just want each picture to occupy a box as tall as the tallest one in the row, you'll need to use a table. You can, of course, style the contents of that table using CSS and many of the techniques we've explored here.

Another situation in which using a table makes sense is if you want a certain number of images per row, no more or less. It's possible to hack the document structure to do this using CSS, of course. For example, consider this skeleton:

```
<div class="row">
  <div class="pic ls">...</div>
  <div class="pic pt">...</div>
  <div class="pic ls">...</div>
  <div class="pic ls">...</div>
</div>
```

By grouping every four pictures together, we can make sure there are ever only four pictures per row. But why would we bother, when we could just as easily use a table? The markup we just proposed is practically a table as it is. Consider this:

```
<tr>
  <td class="pic ls">...</td>
  <td class="pic pt">...</td>
  <td class="pic ls">...</td>
  <td class="pic ls">...</td>
</tr>
```

The table markup, in this case, is easier, requires fewer characters, and will easily limit the number of pictures per row. That's one of the things tables do.

Remember, though, that our original purpose was to allow the pictures to "flow," with each row containing as many images as possible, no matter how wide or narrow the window becomes. Tables can't do that, but floats can. So, as always, choose the tool most appropriate for the job and use it well.

Tableless Table Layout!

There actually is an alternative to both floats and tables: Use the markup shown here and assign table `display` values to the `div`s. We could, for example, declare `div.row {display: table-row;}` and `div.row div {display: table-cell;}`. That would create a table-like structure out of non-table elements, and it would even work in current browsers—except, of course, in Explorer.

Creating the Catalog View

Now that we've seen how to create flowing grids of pictures, let's consider a different approach. This time, we'll set up a vertical listing of images, next to each of which will be placed the title, catalog number, and price of the image. To do this, we'll toss out nearly all of the styles we had before and start over. The only things we'll keep are the body and footer styles, as shown in Listing 2.4. We're basically back to where we were at the beginning of the project (see Figure 2.1 for an illustration).

Listing 2.4 Starting Nearly from Scratch

```
body {background: #EED; margin: 1em;}
div#footer {clear: both; padding-top: 3em;
  font: 85% Verdana, sans-serif;}
```

Floating Again

To keep the layout from getting too outrageously long, we want to put the title and other information next to each image instead of beneath each one. Getting text next to an image is easy: You float the image. Except in this case, we're going to float the link that contains the image.

```
div#footer {clear: both; padding-top: 3em;
  font: 85% Verdana, sans-serif;}
div.pic a.tn {float: left;}
</style>
```

Because the link is floated, the image will come along with it, just as text will come along with a floated `div`. In both cases, the content gets placed into the float.

We'll also add a little styling to the `div`s, just a small margin and a border to help us see how the layout is progressing. The result is shown in Figure 2.11.

```
div#footer {clear: both; padding-top: 3em;
  font: 85% Verdana, sans-serif;}
div.pic {margin: 10px; padding: 0; border: 1px dotted gray;}
div.pic a.tn {float: left;}
```

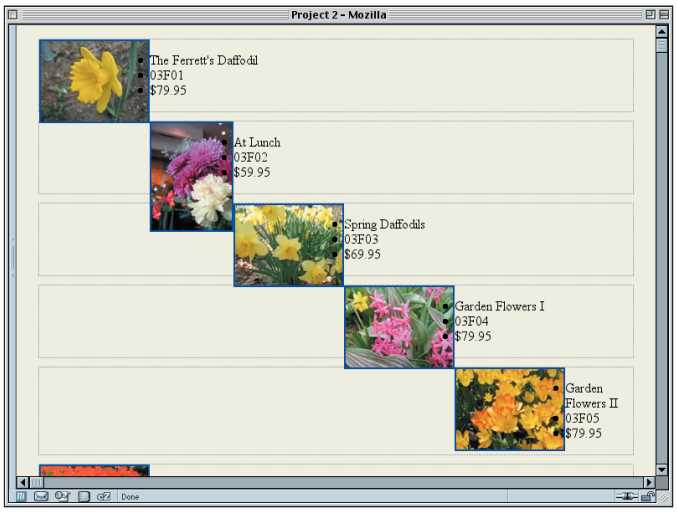


Figure 2.11
Floating can be dangerous if you aren't careful.

Yikes! If we'd meant to do that, then great, no problem; but that's not at *all* what we want for this layout.

What happened? Exactly what we asked to have happen. When you float something, it's removed from the normal flow, which means it doesn't participate in the height of any of its ancestors. Therefore, every `div` is as tall as its normal-flow content (in this case, the information list).

Furthermore, floats start from where they would fall if they were in the normal flow, and then they float to the side. In this case, we're floating left. So the second linked thumbnail started floating from the top of its `div` and headed to the left. Before reaching the left edge of the layout, it encountered another float (the first link), so it stopped and placed itself to the right of that other float. The same thing happened to the third, the fourth, and so on, each one floating over and stopping to the right of the link that came before it.

There are a number of ways we could fix this. If we wanted each picture's `div` to "stretch around" the floated links, we could also float the `div`s. In current browsers (and CSS2.1), a floated element will stretch to enclose any floated descendants. However, we don't actually want that here because the dotted borders are going to be removed later. They're really only for diagnostic purposes.

So, instead, we clear each `div`. Since the floats are all to the left, we can simply clear to the left. We'll also set the width of the `div`s because we'll want it to be restrained later on.

```
div.pic {margin: 10px; padding: 0; border: 1px dotted gray;
  clear: left; width: 350px;}
```

This will push each `div` below any leftward float that comes before it in the layout—like floated links, for example—as shown in Figure 2.12.

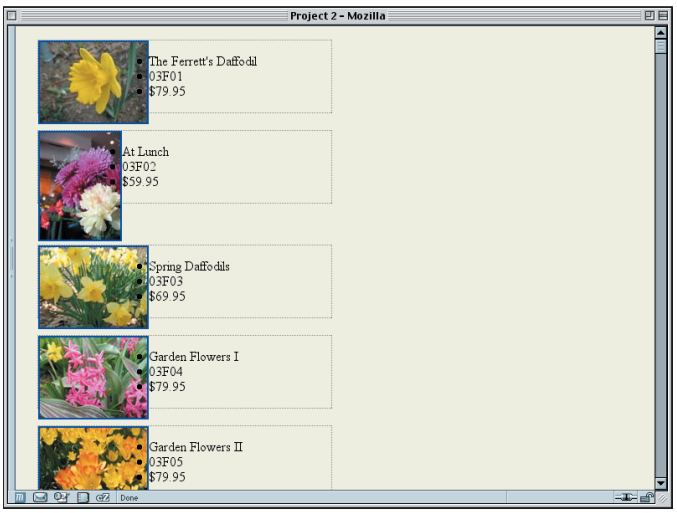


FIGURE 2.12

Clearing away the float problems.

How's That Again?

That explanation might seem confusing; if so, try reading it slowly a few times. While you do, think about what happens if you float a bunch of images with no text near them. The same basic thing is happening here, except the text is making things more complicated.

Alignment and Placement

In looking at Figure 2.12, we can see one thing that needs to be fixed: The portrait image doesn't line up with the others. In a table, the image would probably be in its own cell, and we could just right align it, but we aren't afforded the same luxury here. Fortunately, we can just fiddle with the floated link's margin to get the effect we want. We'll just set its width to match the image it contains and set the left margin to make up the difference between its width and the width of landscape links.

```
div.pic a.tn {float: left;}
div.pt a.tn {width: 96px; margin-left: 32px;}
</style>
```

Since we've explicitly set the width of portrait links, we may as well go ahead and do it for landscape links. It won't hurt anything, and it might come in handy in the future.

```
div.pt a.tn {width: 96px; margin-left: 32px;}
div.ls a.tn {width: 128px;}
</style>
```

Since we're working on the images, let's replace those ugly blue borders with something a little less garish.

```
div.ls a.tn {width: 128px;}
a.tn img {border: 1px solid #333; border-width: 1px 2px 2px 1px;}
</style>
```

The setting of a thicker right and bottom border will lend a barely noticeable "drop shadow" effect to each image. This can be seen on some of the images in Figure 2.13.

FIGURE 2.13

Lining up and shadowing the images with borders.

If you look closely at Figure 2.13, you'll notice two things. The first is that the text and the image borders are overlapping just a bit. That's because, thanks to the borders we just added, the images are now wider than the floating links that contain them. We should increase their widths to compensate. Actually, let's bump them up to be slightly wider than the images, just for the heck of it.

```
div.pt a.tn {width: 100px; margin-left: 32px;}
div.ls a.tn {width: 132px;}
```

The second thing you'll notice is that the list bullets are overlapping the images. This happens due to the way floats work and the way bullets are placed—and it doesn't happen in all browsers. If we wanted to keep these as a bulleted list, we might have problem, but we're going to take them away, so it's kind of irrelevant. In fact, let's do that now.

```
a.tn img {border: 1px solid #333; border-width: 1px 2px 2px 1px;}
div.pic li {list-style: none;}
</style>
```

That's good for preventing overlaps, but it still leaves the text right up against the floated links. Let's move the text over a little bit by using a combination of margins and padding. We'll add in a left border just so we can see where the list's border ends up. This will have the result shown in Figure 2.14.

```
a.tn img {border: 1px solid #333; border-width: 1px 2px 2px 1px;}
div.pic ul {margin: 0 0 0 140px; padding: 0 0 0 0.5em;
  border-left: 1px solid;}
div.pic li {list-style: none;}
```

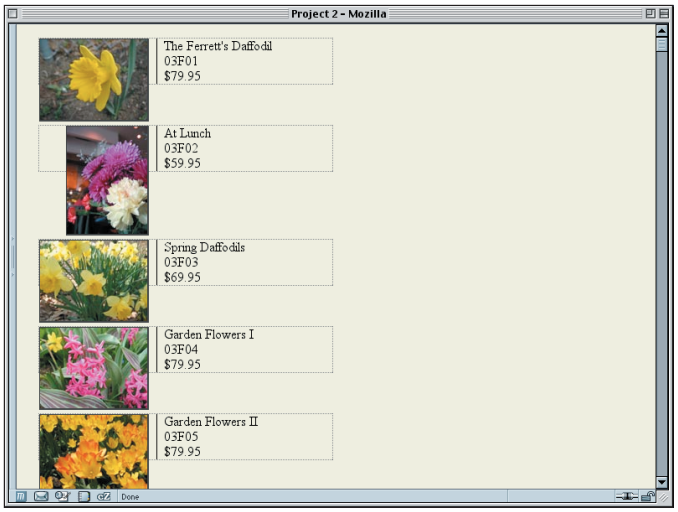


Figure 2.14

Margins and padding help separate the text from the linked images.

Bullet Placement

Bullets are placed in relation to the left content edge of the list item; no matter where that edge lands, the bullet ends up a short distance away. Because the left edges of the list items are right up against the floated images, the bullets end up placed on top of the images.

How did that work? The 140-pixel left margin is actually "sliding beneath" the floated link, extended all the way to the left edge of the `div`. If you look at the second `div` in Figure 2.14, you can see the `div` continuing on to the right of the link. In fact, that happens in all the `div`s, but only with the portrait links is it really obvious.

Improving the Listings

Now that the images and links are all lined up and the lists have been placed so that they're separated from the images a bit, let's turn our attention to the text inside those lists. The first thing we want to do is get rid of those borders since they're starting to get in the way. Thus, the `div.pic` rule should now look like this:

```
div.pic {margin: 10px; padding: 0;
  clear: left; width: 350px;}
```

Similarly, the `div.pic ul` rule should be altered to read as follows:

```
div.pic ul {margin: 0 0 0 140px; padding: 0 0 0 0.5em;}
```

With that done, let's make the titles stand out. Setting them in a bold, sans-serif font would be nice, as would a bottom border. Also, they ought to be shifted downward a bit from the top of the floated links (say, half an em).

```
div.pic li {list-style: none;}
div.pic li.title {font: bold small Arial, Verdana, sans-serif;
  padding-top: 0.5em; border-bottom: 1px solid;}
</style>
```

We've made the font size `small` here because sans-serif fonts look better when they're a little smaller than normal. Just to keep a sense of parallelism, let's make all of the list items' fonts small.

```
div.pic li {list-style: none; font-size: small;}
```

Now for the catalog number and price. These aren't as important as the title, so let's fade them into the background by using a darker shade of tan. We'll also right-align them and italicize the price to give it a little extra emphasis (see Figure 2.15).

```
div.pic li.title {font: bold small Arial, Verdana, sans-serif;
  padding-top: 0.5em; border-bottom: 1px solid;}
div.pic li.catno {color: #776; text-align: right;}
div.pic li.price {color: #776; text-align: right;
  font-style: italic;}
</style>
```

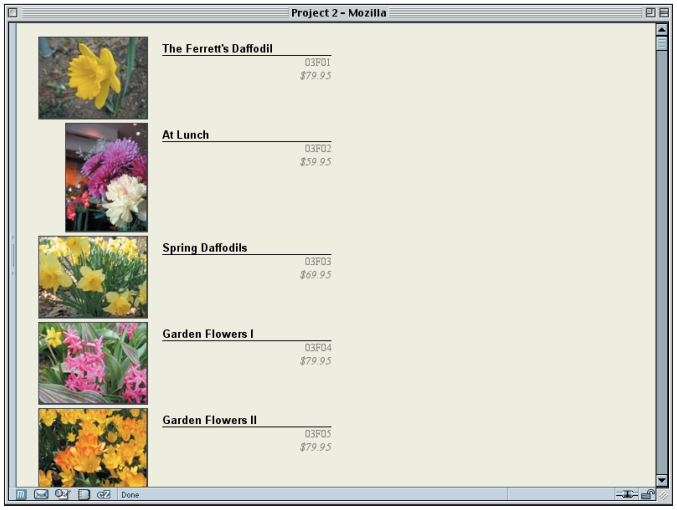


FIGURE 2.15

The text looks a lot better thanks to some alignment and color.

We could just stop here, but let's keep going. It would be kind of cool if the catalog number and price could sit side by side, wouldn't it? Well, maybe not cool, but more elegant perhaps. At any rate, let's do it. And let's do it without adding any floats to the layout.

To make this work, we're going to have to pull the price upward by the height of one line. To make *that* work, we need to make sure the line heights are equal across browsers and that the catalog number won't be in the way. First we'll regularize the list item heights by explicitly defining a `line-height` and zeroing out the margin and padding.

```
div.pic li {list-style: none; font-size: small;
  line-height: 1.2em; margin: 0; padding: 0;}
```

Now we'll move the catalog number aside. We'll do this by giving it a right margin of 4.5em.

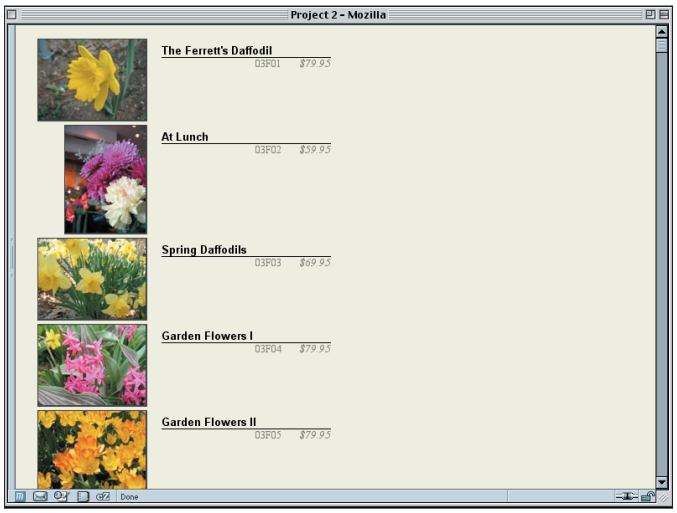
```
div.pic li.catno {color: #776; text-align: right;
  margin-right: 4.5em;}
```

Now all we need to do is slide the price up into the space we just created for it. Remember the `line-height` value? We'll just set a negative top margin equal to that value, and the text will drop into place, as shown in Figure 2.16.

```
div.pic li.price {color: #776; text-align: right;
  font-style: italic; margin: -1.2em 0 0 0;}
```

FIGURE 2.16

Placing text next to text through the magic of margins.



Boxing the Information

Explorer Watch

Almost none of the rest of this project will work in Explorer 5.x for Windows, which doesn't understand auto margins. The first part of this section does work quite nicely in IE6, however, as long as you're in standards mode.

Let's take another step toward advanced layout by putting a vertical border to the left of the catalog number, another one between the number and price, and then enclosing the whole set of text information in a box.

To get the borders where we want them, we'll apply very similar styles to the catalog and price list items. The basic idea is to set a width for each one and then push them over as far to the right as needed. We'll do this by setting the left margin to be `auto` and the rest to be zero. So, for the price, we'll declare:

```
div.pic li.price {color: #776; text-align: right;
  font-style: italic; margin: -1.2em 0 0 auto;
  width: 4em; border-left: 1px solid;}
```

Then we'll replace the catalog number's `margin-right` with a `margin` declaration and give it the same width and border as we gave the price. You can see this in Figure 2.17.

```
div.pic li.catno {color: #776; text-align: right;
  margin: 0 4.5em 0 auto;
  width: 4em; border-left: 1px solid;}
```

Now to add the box around all the text information—here's where things get a little tricky. If we just add a border and background to the `ul` element, IE6 for Windows will get hopelessly muddled and start making elements disappear for no good reason. So, rather than confuse the poor dear, we'll just hide the offending styles from its sight. Here's how:

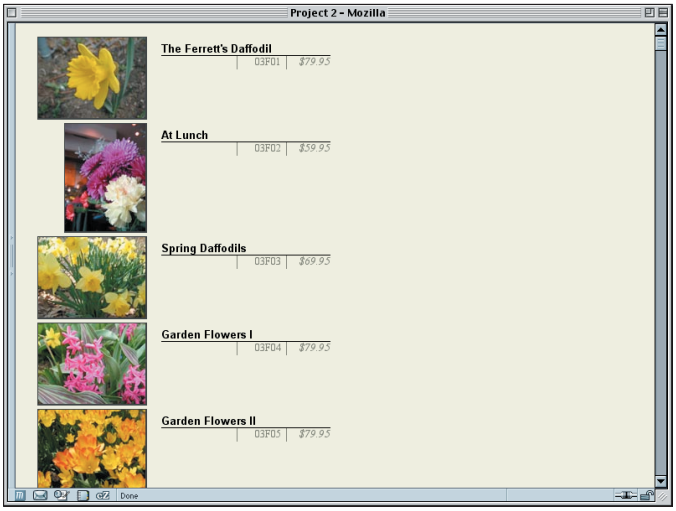


FIGURE 2.17

Placing borders using more margin magic.

```
div.pic ul {margin: 0 0 0 140px; padding: 0 0 0 0.5em;}
html>body div.pic ul {background: #CCB; border: 3px double #552;}
div.pic li {list-style: none; font-size: small;
  line-height: 1.2em; margin: 0; padding: 0;}
```

The first part of the selector, the `html>body` part, is what hides the rule from IE/Win. It's a child selector, which is perfectly valid CSS2 that IE/Win just doesn't understand. More capable browsers do, so they can see and apply those styles.

With the addition of the borders, we need to move the price over to the left just a bit so that it isn't right up against the double border. We'll do that with a quick change of its right margin.

```
div.pic li.price {color: #776; text-align: right;
  font-style: italic; margin: -1.2em 3px 0 auto;
  width: 4em; border-left: 1px solid;}
```

Finally, let's add some separation between each listing in our Catalog view. We have two choices: We can add a bottom margin to the floated links or to the images inside those links. Adding a top margin to the `div`s won't work because of the way `clear` is defined. To avoid a few obscure bugs in Explorer, we'll place the margin on the image.

```
a.tn img {border: 1px solid #333; border-width: 1px 2px 2px 1px;
  margin: 0 0 1em;}
```

With that, we've arrived at the style sheet shown in Listing 2.5 and illustrated in Figure 2.18.

Clear Rules

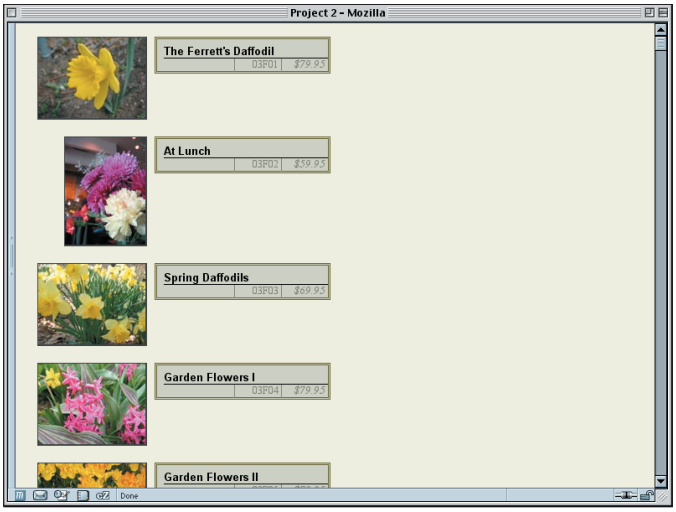
`clear` works by increasing the top margin of the cleared element until its outer top border edge is just below the bottom edge of previous floating elements. Thus, if we tried to set the top margin on the `div`s, our setting would be overridden by the clearing.

Listing 2.5 The Complete "Catalog" Style Sheet

```
body {background: #EED; margin: 1em;}
div#footer {clear: both; padding-top: 3em;
  font: 85% Verdana, sans-serif;}
div.pic {margin: 10px; padding: 0;
  clear: left; width: 350px;}
div.pic a.tn {float: left;}
div.pt a.tn {width: 100px; margin-left: 32px;}
div.ls a.tn {width: 132px;}
a.tn img {border: 1px solid #333; border-width: 1px 2px 2px 1px;
  margin: 0 0 1em;}
div.pic ul {margin: 0 0 0 140px; padding: 0 0 0 0.5em;}
html>body div.pic ul {background: #CCB; border: 3px double #552;}
div.pic li {list-style: none; font-size: small;
  line-height: 1.2em; margin: 0; padding: 0;}
div.pic li.title {font: bold small Arial, Verdana, sans-serif;
  padding-top: 0.5em; border-bottom: 1px solid;}
div.pic li.catno {color: #776; text-align: right;
  margin: 0 4.5em 0 auto;
  width: 4em; border-left: 1px solid;}
div.pic li.price {color: #776; text-align: right;
  font-style: italic; margin: -1.2em 3px 0 auto;
  width: 4em; border-left: 1px solid;}
```

FIGURE 2.18

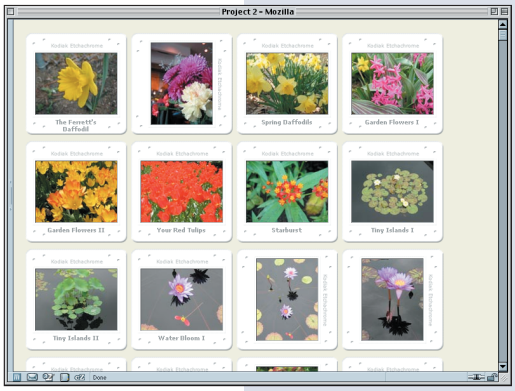
The information presented in catalog style.



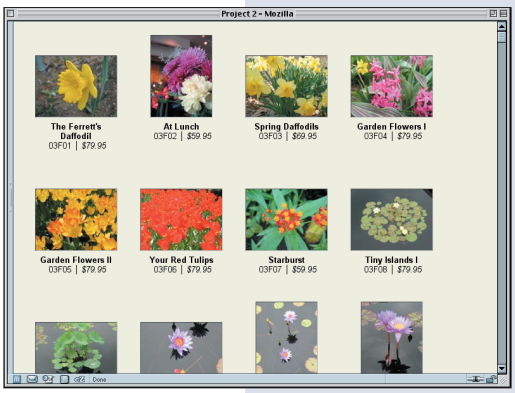
Branching Out

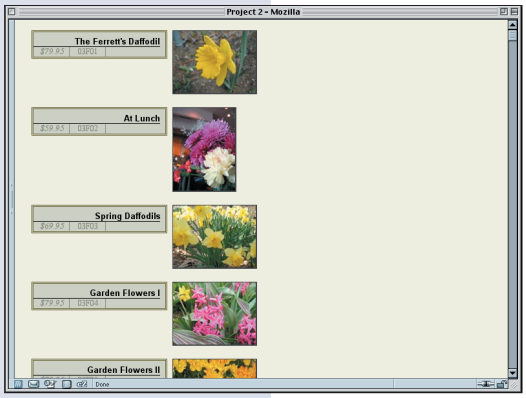
Try re-creating the following changes to the work in this project.

1. In the Contact Sheet view, try placing the title right below the picture, looking as if it were written onto the lower part of the slide frame. To do this, you'll need to remove the bottom margin from the images without throwing off the overall layout. Note that this will not be possible for portrait images since CSS is not able to rotate text, so you'll need to constrain your styles accordingly. Note also that long titles might flow off the slide, so a property like `overflow` might be useful.



2. In the Gallery view, add the catalog number and price back into the layout, but put them next to each other instead of one on top of the other. This will enhance the Gallery view without significantly changing the layout. Remember that a little extra height may be needed.





3. In the Catalog view, try flipping things around so that the picture is on the right and all of the textual information is on the left. This will mean more than changing the float's direction: You will also need to adjust the layout of the list items.