JavaScript

Pages:
1 **Live Thumbnails: Watch 'em Grow**
2 Animating the images

Live Thumbnails: Watch 'em Grow

by Chris Klimas *22 Feb 2006*

Chris Klimas tries his best to abuse HTML and the English language in equal amounts. He tills the soil at Gimcrack'd and eats breakfast whenever possible.

Page 1

If you ask me, image galleries are boring.

They've been around since forever, and they all work the same way. You have a page full of thumbnails, and when you click one, you get a page with a larger version. What if thumbnails could grow into full images seamlessly right on the same page?

Impossible, you say? Not so, good sir or madam, if we use a little JavaScript. Check out the demo to see yourself.

Let's start with the HTML fragment we'll build our script around:

```
<a href="large.jpeg" class="livethumbnail"> <img
src="thumbnail.jpeg" width="120" height="90" largewidth="480"
largeheight="360" /></a>
```

This is pretty much what a normal gallery page would look like, which is good — People browsing with JavaScript disabled will still have an OK experience — but there are two important differences here.

First, the <a> tag has a class of "livethumbnail". Normally, classes are used to apply CSS styles, but we're going to use this as a way to trigger the growing image effect.

Second, we gave the <img> tag some weird properties called "largewidth" and "largeheight". This isn't your father's HTML 4.01 Transitional! These attributes will pass information back to our JavaScript — in this case, the dimensions of the full-sized image. This has the drawback of making your pages non-standards compliant, but right now it seems to be the best way to do it.

This is all you need to know if all you want to do is use the script. Just paste in the source code and you're good to go. Those curious about how it works can keep reading — though if you're not very comfortable with JavaScript, you should read Thau's basic and advanced tutorials first. The script is fairly short, but there are a lot of tricky things crammed in there.

There are two main tasks ahead of us. First, we've got to attach event handlers to all the links classed as "livethumbnail" on the page. Then, we have to resize the image in response to those clicks.

Here's the first part of the code that attaches handlers:

```
var links = document.getElementsByTagName('a');

for (var i = 0; i < links.length; i++)
    if (links[i].className == 'livethumbnail')
  ...
```

Sadly, there's no way in JavaScript to automatically grab every element on a page with a certain class (though certain frameworks will do this for you), so we have to grab all <a>

elements and check their classes.

Once we've found a link element with the right class, we tunnel to find the image it encloses:

```
var img = links[i].getElementsByTagName('img')[0];
```

Now we add a whole bunch of properties to that image that will help later:

```
img.state = 'small';
img.smallSrc = img.getAttribute('src');
img.smallWidth = parseInt(img.getAttribute('width'));
img.smallHeight = parseInt(img.getAttribute('height'));
img.largeSrc = links[i].getAttribute('href');
img.largeWidth = parseInt(img.getAttribute('largewidth'));
img.largeHeight = parseInt(img.getAttribute('largeheight'));
img.ratio = img.smallHeight / img.smallWidth;
```

JavaScript lets you add new properties to any object at all, so you can stuff information into a page element for future reference. The most interesting of these properties is "ratio", which is the image's ratio of width to height. We'll use it to keep the proportions of the image correct as it grows and shrinks.

Finally, we connect the link to the function that's actually going to scale the image up and down:

```
links[i].onclick = scale;
```

Now for the real work. We start off with some simple stuff:

```
function scale()
{
      var img = this.getElementsByTagName('img')[0];
      img.src = img.smallSrc;
```

This grabs the image inside the <a> tag that was clicked, then changes its source file to the thumbnail version. Regardless of whether we're scaling things up or down, we always want to use the smaller version because it scales much quicker than a full-resolution image.

Next, we start preloading the full image:

```
if (! img.preloaded)
{
      img.preloaded = new Image();
      img.preloaded.src = img.largeSrc;
};
```

We do this by adding yet another property to the element, called "preloaded". This property uses one of the oldest tricks in the book to start loading the full-resolution image while the user's watching the animation.

OK, now let's animate it!

next page»

[an error occurred while processing this directive] [an error occurred while processing this directive]