JavaScript

Live Thumbnails: Watch 'em Grow

Page 2 — Animating the images

Here's how we start animating our live thumbnails:

```
var interval = window.setInterval(scaleStep, 10);
return false;
```

The `setInterval` function is a lot like `setTimeout`, only you don't have to keep invoking it on every step of your animation.

The `scaleStep` function is nested inside the scale function. This is a nifty trick called a closure. There are a lot of weird things you can do with closures, but here we're just taking advantage of being able to refer to variables defined in `scale()`. Note that there's a separate state for each instance of a closure — this means that you can scale up several images at once without the functions stepping on each other's feet.

Here's how `scaleStep` starts:

```
function scaleStep()
{
      var step = 10;
       var width = parseInt(img.getAttribute('width'));
       var height = parseInt(img.getAttribute('height'));
```

Notice that we're using the `img` variable we set up in the scale function. The `step` variable controls how quickly the image will resize, and the next two lines figure out what the image's dimensions are right now. We have to throw in a `parseInt()` to make sure the variable is treated as a number, not a string.

I'm going to show you how we handle growing the image. Shrinking it works exactly the same way, but with minus signs instead of plus signs.

```
if (img.state == 'small')
{
      width += step;
      height += Math.floor(step * img.ratio);
       img.setAttribute('width', width);
       img.setAttribute('height', height);
```

We take advantage of the properties we stuffed into the image to keep its proportions intact. `Math.floor()` makes sure we're setting the height to a round number, since there's no such thing as 120.6 pixels.

Finally, we have to figure out when we're done:

```
if (width > img.largeWidth - step)
{
       img.setAttribute('width', img.largeWidth);
       img.setAttribute('height', img.largeHeight);
      img.setAttribute('src', img.largeSrc);
       window.clearInterval(interval);
     img.state = 'large';
};
```

If we're close enough to the full-sized dimensions, we snap the image to the correct size and swap the image to its full-resolution version. If we're lucky, the full-sized image will already be loaded by the time this happens. If not, then the image will stay blocky for a moment, then turn full-resolution once it finishes loading. You can make the step variable smaller to compensate for this, but it's an inexact science.

Finally, we stop the animation and change the image's state to large, so we know to scale it down the next time it's clicked.

That's it! Give it a spin with your next set of digital photos and keep your audience awake.

Did you love this article? Did you hate it? Think you can do better? Send us your Feedback. Feedback submitted here will be considered for publication on Webmonkey or Wired News, so if you don't want us to print your comments, please say so in your email.