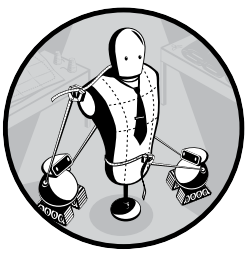
8

BACKGROUND IMAGES



Adding decorative elements to make our websites more visually appealing used to be surprisingly resource- and time-intensive. Even seemingly simple graphical effects, such as two background images on the same element, required a lot of unnecessary markup, and this in turn made pages slower to render and harder to maintain.

CSS3 introduces a number of new and extended properties that are aimed at decorating elements much more simply, and the browser makers have been quick to implement them and to add a number of their own implementations as well. Over the next few chapters, we'll take a look at the new range of features that we can use to prettify our pages, from background images to decorative borders and new color effects.

I'll begin by taking you on a walk through the Backgrounds and Borders Module (*http://www.w3.org/TR/css3-background/*). Because of high demand from web developers, the new properties it brings are already well implemented by browsers. Internet Explorer 9 fully implemented the properties

and changes listed in this chapter, as have—in most cases—all of the other major modern browsers, so unless otherwise stated in the text you can presume widespread support.

As the Backgrounds and Borders Module is quite extensive, I'll break it over two chapters and start with a look at background image properties. Background images have been part of CSS for many years, but unlike previous versions, in CSS3, you can apply multiple images to elements, and you can resize those images on the fly. Just these two new features alone would be enough to please most of us, but the specification goes further to provide more control over image placement and tiling.

Updates to Existing Background Properties

Many of the other CSS3 modules bring new properties and even whole new concepts to CSS, but the strength of the Backgrounds and Borders Module is the way it extends existing properties to make them more powerful and useful. That's not to say this module has no novelties—it certainly does, and I'll come to them shortly. But the subtleties shine, and in this section, I want to talk about the extensions and changes to properties you'll be familiar with from CSS2.1.

background-position

The `background-position` property in CSS2.1 accepts two values: either a keyword for each side of the box (`top`, `right`, and so on), or length or percentage values that set a position relative to the top-left corner of the element to which it's applied. This is okay for many tasks but doesn't really provide the fine control that we desire when laying out pages.

In CSS3, the property now accepts up to four values: you can use keywords to specify a side and then length or percentage values for relative distance from that side. Take a look at this example code:

```
.foo { background-position: right 10em bottom 50%; }
```

The background image on the element `.foo` will be positioned 10em from the right and 50% from the bottom. This positioning would have been very difficult in CSS2.1; you had to know the widths of all the elements involved and that they didn't change.

background-attachment

The way that a background image scrolls in the viewport is determined by the `background-attachment` property. The permitted values in CSS2.1 are `scroll` (the default), which means the image doesn't scroll with the element it's applied to but does scroll with the viewport, and `fixed`, which means the image doesn't scroll with either its element or the viewport.

A new value of `local` is introduced in CSS3; this value allows an image to scroll with both its element and the viewport. This is nigh-impossible to demonstrate in a static book, so I urge you to take a look at example file 8-a on the book's companion website (*http://thebookofcss3.com/*).

The new value is supported in IE9+ and all other major modern desktop browsers. Mobile browsers, however, tend to use different viewport layout mechanisms in which fixed elements don't really work, so you'll likely get unexpected (or, no) behavior in those.

background-repeat

In CSS2.1, the `background-repeat` property accepts one of four possible values: `no-repeat`, `repeat`, `repeat-x`, and `repeat-y`. With these values, you can tile images either horizontally or vertically (or both) across an element, but they don't allow for any finer control than that. CSS3, however, extends the usefulness of the property in two ways: a pair of new properties and a tweak to the syntax.

The first of the new properties is `space`, which sets the background image to repeat across its containing element as many times as possible without clipping the image. All of the repetitions (except the first and last) are then equally spaced, so the image is evenly distributed.

The second is `round`, which likewise sets the background image to repeat as many times as possible without clipping, but instead of equally spacing the repetitions, the images scales so a whole number of images fills the containing element.

To compare the difference between the two, I've put together an example in which a different `background-repeat` value is applied to two elements, using the following code:

```
.space { background-repeat: space; }
.round { background-repeat: round; }
```

Figure 8-1 displays the results. The element on the left is for reference; it has the default `background-repeat` value of `repeat` and shows the behavior you would currently expect. The element in the middle has a value of `space`, and the maximum number of images that can be tiled without clipping or scaling are displayed with empty space between them. Finally, the element on the right has a value of `round`, which calculates the maximum whole number that can fit in the containing element both horizontally and vertically, scaling the image as required.

Internet Explorer 9+ and Chrome are currently the only browsers to implement these keywords correctly. Safari recognizes them but makes them both behave incorrectly, as if `no-repeat` were applied. Firefox ignores them and uses the previous cascaded or inherited value.

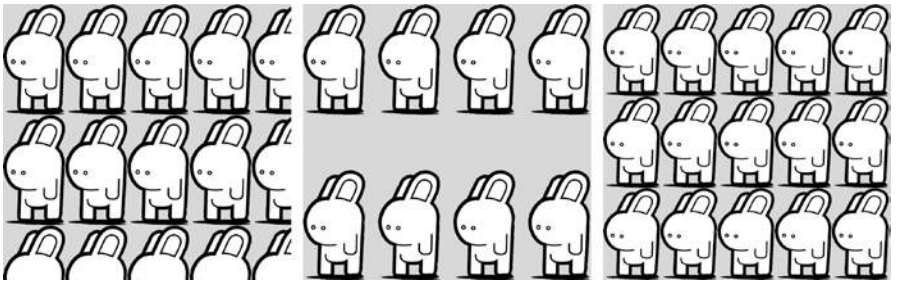


Figure 8-1: `background-repeat` values: `repeat` (left), `space` (center), and `round` (right)[1]

I also mentioned a change to the syntax. You can control tiling on the two different axes independently, as the property now accepts two values. The first value controls tiling on the horizontal axis, the second on the vertical. So if you want a background image to repeat with rounding on the vertical and spacing on the horizontal, you use this code:

```
.foo { background-repeat: round space; }
```

The result is shown in Figure 8-2.

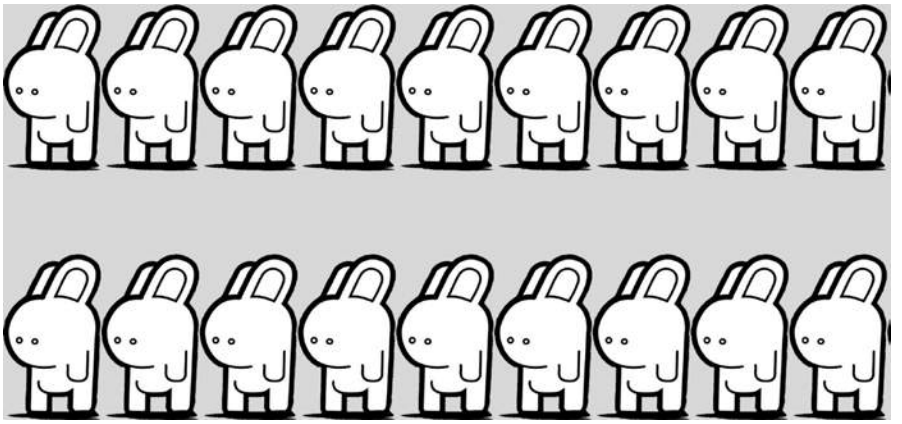


Figure 8-2: Different `background-repeat` values applied to the horizontal and vertical

Multiple Background Images

The first new feature in the Backgrounds and Borders Module isn't a new property but an extension of an existing one—or, rather, several existing ones. Using CSS2.1, you could only apply a single background image to an element, but in CSS3, (almost all of) the `background-*` properties now accept multiple values, so you can add many background images to one element.

1. The bunny image is by Flickr user Andrew Mason (*http://www.flickr.com/photos/a_mason/42744470/*) and is used under license.

To do this, you need just list the values separated by commas. For example, here's the syntax with `background-image`:

```
E { background-image: value, value; }
```

For each background layer you create, you can add appropriate values to all of the relevant `background-*` properties. Here's a real-world example:

```
h2 {
    background-image: url('monkey.svg'), url('landscape.jpg');
    background-position: 95% 85%, 50% 50%;
    background-repeat: no-repeat;
}
```

You can see how this looks in Figure 8-3. The layers are created in reverse order—that is, the first layer in the list becomes the topmost layer, and so on. In my example code, *monkey.svg* is a layer above *landscape.jpg*. The `background-position` property follows the same order: The landscape is positioned at `50%` left and `50%` top (the horizontal and vertical center) of its containing element and the monkey at `95%` left and `85%` top.

Figure 8-3: Two background images on the same element[2]

Note that I've only given `background-repeat` one value; if a property has fewer values than there are background layers, the values will repeat. In this example that means `no-repeat` will be applied to all background layers.

You can use multiple values with the `background` shorthand property; as with the individual properties, you only need to provide a comma-separated list. To get the same result seen in Figure 8-3, I can also use this code:

```
h2 {
    background:
    url('monkey.svg') no-repeat 95% 85%,
    url('landscape.jpg') no-repeat 50% 50%;
}
```

2. The monkey image is by rachelps on openclipart (*https://openclipart.org/detail/2876/cheeky-monkey-by-rachelps*) and is used under license.

I mentioned at the start of this section that almost all background properties can have multiple values. `background-color` is the exception, however, as the color layer will always be stacked below all other background layers. If you want to specify a background color when using the shorthand property, you must place it in the last instance of the comma-separated list. In the case of my example code that would be in the instance with the landscape picture:

```
h2 {
    background:
    url('monkey.svg') no-repeat 95% 85%,
    url('landscape.jpg') no-repeat 50% 50% #000;
}
```

Dynamically Scaled Background Images

A new property to CSS3 is `background-size`. This property, as you can probably guess, allows you to set the size of the background images. Here's the syntax:

```
E { background-size: value; }
```

This property's value can be a pair of lengths or percentages, a single length or percentage, or a keyword. If a pair is used, the syntax is as follows:

```
E { background-size: width height; }
```

To resize a background image to be 100px wide and 200px high, you use:

```
div { background-size: 100px 200px; }
```

The length can be any standard unit of measurement. If you use percentages, the dimension is based on the containing element, *not* the background image. So a width and height of `100%`, for example, will stretch the background image to fill the container. To make the image appear at its natural size, use the `auto` keyword.

If you only specify a single value, that value is considered the width, and the height is then assigned the default value of `auto`. Therefore, these two examples are exactly equivalent:

```
div { background-size: 100px auto; }
div { background-size: 100px; }
```

You can use your newly learned multiple background method with background-size as well. For example, let's revisit Figure 8-3, but repeat the monkey image a few more times, adding different values to the background-position and background-size properties. Here's the code:

```
h2 {
background:
    url('monkey.svg') no-repeat 95% 85%,
    url('monkey.svg') no-repeat 50% 80%,
    url('monkey.svg') no-repeat 10% 100%,
    url('landscape.jpg') no-repeat 50% 50%;
    background-size: auto 80%, auto 15%, auto 50%, auto;
}
```

Figure 8-4 shows this method in action. One monkey has a vertical background-size of 80%, the next 15%, and the last, 50%; in all cases, the horizontal size has been set to auto to keep the image in proportion.

Figure 8-4: Example of multiple resized background images

As well as length values, two keywords are available: contain and cover. The contain keyword sets the image to scale (proportionately) as large as possible, without exceeding either the height or width of the containing element; cover sets the image to scale to the size of either the height or width of the containing element, whichever is larger.

Take a look at the following code to see what I mean:

```
.monkey-1, .monkey-2 {
    background-image: url('monkey.svg');
    background-position: 50% 50%;
}
.monkey-1 { background-size: contain; }
.monkey-2 { background-size: cover; }
```

I used two elements, with classes of `monkey-1` and `monkey-2`, and set different keyword values for `background-size` on each. The result is shown in Figure 8-5.

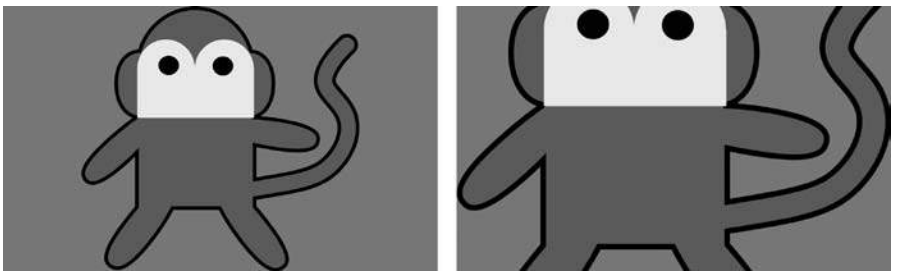


Figure 8-5: `background-size` keywords: `contain` (left) and `cover` (right)

The box on the left has the `contain` keyword value, so the background image fills the box vertically (the shortest length); the box on the right has the `cover` keyword value, so the background image fills the box horizontally (the longest length) and is cropped at the top and bottom.

Background Clip and Origin

In CSS2, the position of a background image is defined relative to the outer limit of its containing element's padding, and any overflow extends underneath its border. CSS3 introduces two new properties that provide more granular control over this placement.

The first property is `background-clip`, which sets the section of the box model that becomes the limit of where the background (either color or image) is displayed. Here's the syntax:

```
E { background-clip: box; }
```

The *box* value can be one of three keywords: `border-box`, `content-box`, or `padding-box`. `border-box`, the default value, displays the background behind the border (you can see it if you use a transparent or semi-opaque border color). A value of `padding-box` displays the background only up to, and not behind, the border. `content-box` means the background stops at the element's padding.

I'll illustrate the difference using the following code:

```
h2 {
    background: url('landscape.jpg') no-repeat 50% 50% #EFEFEF;
    border-width: 20px;
    padding: 20px;
}
h2.brdr { background-clip: border-box; }
h2.pddng { background-clip: padding-box; }
h2.cntnt { background-clip: content-box; }
```

I've used three h2 elements with classes of brdr, pdding, and cntnt, respectively. Figure 8-6 illustrates the difference between the values.

Figure 8-6: Showing the effect of different values on the background-clip property: border-box (left), padding-box (center), and content-box (right)

I've used a semi-opaque border (I'll explain how in Chapter 10) so you can see the image paint beneath it in the box on the left, which has the border-box value. The central box has the padding-box value, and as you can see, the background stops at the limit of the padding. In the box on the right, the value is content-box, so the background does not show behind the padding.

The second property that gives you more granular control is background-origin. Using background-origin, you can set the point where the background is calculated to begin. As I mentioned before, CSS2 background positions are calculated relative to the limit of the padding, but background-origin lets you change that. Here's the syntax:

```
E { background-origin: box; }
```

The *box* value accepts the same keywords as you've just seen in background-clip: border-box, content-box, and padding-box. I'll explain the different results using this code:

```
h2 { background: url('monkey.svg') no-repeat 0 100%;}
h2.brdr { background-origin: border-box; }
h2.cntnt { background-origin: content-box; }
h2.pddng { background-origin: padding-box; }
```

The effects of the different values are illustrated in Figure 8-7. As you can see, the monkey is in a different position in each box because the background-position is calculated relative to a different point in each box (I've added a background grid to make it a little easier to see).

The background-position is always set at 0 100%, which is the bottom left. The point from which the bottom left is measured changes depending on the background-origin value, however. In the first box, the background originates at the limit of the border; in the second, from the limit of the padding; and in the third, from the limit of the content box.

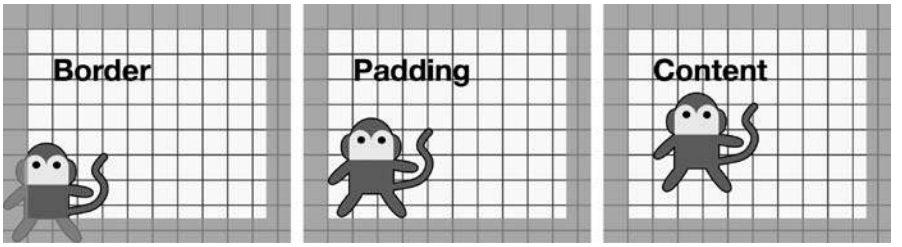


Figure 8-7: The `background-origin` property with values of `border-box` (left), `padding-box` (center), and `content-box` (right)

A couple of things to bear in mind: First, this property has no effect if the `background-position` is set to `fixed`. Second, both `background-clip` and `background-origin` accept multiple values, using the same syntax shown in "Multiple Background Images" on page 88.

Updated Background Shortcut

The `background` shortcut property has been updated to include values for the `background-size`, `background-clip`, and `background-origin` properties. Values for `background-size` should immediately follow those for `background-position` and be separated by a forward slash, like so:

```
E { background: url('bar.png') no-repeat 50% 50% / 50% auto; }
```

In this case, the background image, *bar.png*, will be positioned at the dead center of the element, with a width set to 50% of the element and an automatic height.

For `background-clip` and `background-origin`, if only one box value (`border-box`, `padding-box`, or `content-box`) is present, both properties will be set to that value. If two box values are supplied, the first will be set on `background-origin` and the second on `background-clip`. As an illustration, take this shorthand code:

```
E { background: url('bar.png') no-repeat padding-box content-box; }
```

In this case, the origin of the background image will be the padding box, and the image will be clipped to the content box.

Summary

The new features introduced in this chapter are a big step toward the stated aim of CSS: to separate a page's content from its presentation. More flexibility with background images means fewer required elements to create the effects we want, and the more nonessential markup we can remove from our documents, the easier our pages will be to maintain and the better it will be for semantics.

In this chapter, I've covered only half of what the Backgrounds and Borders module offers, so in the next chapter I'll cover the other half—which, as you can probably guess by the title "Border and Box Effects," relates to borders.

Background Images: Browser Support

	Chrome	Firefox	Safari	IE
`background-position` (edge values)	Yes	Yes	Yes	Yes
`background-attachment`	Yes	Yes	Yes	IE10
`background-repeat` (new values)	Yes	No	No*	Yes
`background-repeat` (two values)	Yes	Yes	Yes	Yes
Multiple background images	Yes	Yes	Yes	Yes
`background-size`	Yes	Yes	Yes	Yes
Updated `background` property	Yes	Yes	Yes	Yes
`background-clip`	Yes	Yes	Yes	Yes
`background-origin`	Yes	Yes	Yes	Yes

* The values are recognized but don't display correctly.