


Technologies ▾

References & Guides ▾

Feedback ▾

Sign in 

 Search

Changing background styles using CSS

[Previous](#)

[Overview: Styling boxes](#)

[Next](#)

In CSS you can do a lot to style the background behind your content. We've already looked at some simple uses, such as basic background colors and images; in this article we'll tell the whole story, and look at some advanced features like multiple background images and color gradients.

Prerequisites: HTML basics (study [Introduction to HTML](#)), and an idea of How CSS works (study [Introduction to CSS](#).)

Objective: To learn about styling element backgrounds in detail.

What exactly is a background?

The background of an element is the area that sits underneath an element's content, padding, and border. By default this is the case anyway — in newer browsers you can alter the area the

background takes up, using the `background-clip` property (see the CSS box model article `background-clip` coverage for more details.)

The background doesn't sit underneath the margin — the margin doesn't count as part of the element's area, but rather the area outside the element.

There are many other different properties you can use to manipulate the element's background, some of which we have already seen a number of times in our course already:

- `background-color`: Sets a solid color for the background.
- `background-image`: Specifies a background image to appear in the background of the element. This can be a static file, or a generated gradient.
- `background-position`: Specifies the position that the background should appear inside the element background.
- `background-repeat`: Specifies whether the background should be repeated (tiled) or not.
- `background-attachment`: Specifies the behaviour of an element's background when its content scrolls, e.g. does it scroll with the content, or is it fixed?
- `background`: Shorthand for specifying the above five properties on one line.
- `background-size`: Allows a background image to be resized dynamically

Note: Most of these features have really good browser support, except for the last two, from which the support is a bit more limited (Internet Explorer 9+, Safari 7+, Android 4.4+), and background gradients (Internet Explorer 9+).

Throughout the rest of the article, we'll look at how to use these features in detail.

The basics: color, image, position, repeat [↗](#)

Let's explore a simple example to show how the four most basic properties work — you'll use these ones all the time when dealing with backgrounds.

Background color [↗](#)

You'll use the `background-color` property very often:

- For a start, the default background color of most elements is not white (as you might expect) but transparent — therefore if you set an element's background color to something interesting, but want its child elements to be white, you'll have to set that explicitly.
- In addition, it is important to set a background color as a fallback. Background colors are supported pretty much everywhere, whereas more exotic features such as background gradients are supported only in newer browsers, plus a background image might fail to load for some reason. It is therefore a good idea to set a basic background color as well as specifying such features, so the element's content is readable no matter what.

Let's start by building up an example. We'll begin with some simple HTML:

```
1 | <p>Exciting box!</p>
```

Let's give it a background color:

```
1 | p {  
2 |   font-family: sans-serif;  
3 |   padding: 20px;  
4 |   /* background properties */  
5 |   background-color: yellow;  
6 | }
```

The result is as follows:



Exciting box!

Background image 

The `background-image` property specifies a background image to display in the background of an element. The simplest usage of this property involves using the `url()` function — which takes the path to an image as a parameter — to fetch a static image file to insert. Let's add a background image to the above example:

```
1 | p {  
2 |   font-family: sans-serif;  
3 |   padding: 20px;  
4 |   /* background properties */  
5 |   background-color: yellow;  
6 |   background-image: url(https://mdn.mozillademos.org/files/13026/fire-ball-ic  
7 | }
```

The result is as follows:

Exciting box!

This doesn't look great at the moment — images are repeated horizontally and vertically by default. We can fix this using `background-repeat`, which we'll look at next.

Note: Pretty much any image format that HTML images support is also supported in CSS background images, including SVG.

One important thing to bear in mind is that since background images are set using CSS and appear in the background of content, they will be invisible to assistive technologies like screen readers. They are not content images — they are just for decoration — if you want to include an image on your page that is part of the content, then you should do so with an `` element.

Background repeat [↗](#)


`background-repeat` allows you to specify how the background image is repeated. The default value is `repeat` which, as you saw above, makes the image keep repeating until the whole element background is filled. This isn't what we want in this case (although it might be in some cases, e.g. `repeating-background.html`). Other common and widely supported values are:

- `no-repeat`: The image will not repeat at all: it will only be shown once.
- `repeat-x`: The image will repeat horizontally all the way across the background.
- `repeat-y`: The image will repeat vertically all the way down the background.
- `repeat`: The image will repeat both vertically and horizontally.

Let's fix our example:

```
1 p {  
2   font-family: sans-serif;  
3   padding: 20px;  
4   /* background properties */  
5   background-color: yellow;  
6   background-image: url(https://mdn.mozillademos.org/files/13026/fire-ball-ic  
7   background-repeat: no-repeat;  
8 }
```

The result is as follows:



Exciting box!

This is certainly looking better, but the positioning of the image is off — it is currently overlapping the text. We need to position it in a better place.

Background position

`background-position` allows you to position your background image wherever you want inside the background. Generally the property will take two values separated by a space, which

specify the horizontal (x) and vertical (y) coordinates of the image. The top left corner of the image is the origin — (0,0). Think of the background as a graph, with the x coordinate going across from left to right, and the y coordinate going from top to bottom.

The property can accept many different value types; the most common ones you'll use are:

- Absolute values like pixels — for example `background-position: 200px 25px`.
- Relative values like rems — for example `background-position: 20rem 2.5rem`.
- Percentages — for example `background-position: 90% 25%`.
- Keywords — for example `background-position: right center`. These two values are intuitive, and can take values of `left`, `center`, `right`, and `top`, `center`, `bottom`, respectively.

You should note that you can mix and match these values, for example `background-position: 99% center`. Also note that if you only specify one value, that value will be assumed to be the horizontal value, and the vertical value will default to `center`.

Let's fix up our example:

```
1  p {  
2    font-family: sans-serif;  
3    padding: 20px;  
4    /* background properties */  
5    background-color: yellow;  
6    background-image: url(https://mdn.mozillademos.org/files/13026/fire-ball-ic  
7    background-repeat: no-repeat;  
8    background-position: 99% center;  
9  }
```

The result is as follows:

Exciting box!

Background image: gradients [↗](#)

There are another set of available values for `background-image` — color gradients, which are smooth color transitions across a background. Dynamically generated gradients were introduced as a feature a little while ago, because the use of gradients in web design was so popular, but implementing gradients using background images is rather inflexible. There are two types of gradient available at the moment — linear gradients (that go from one line straight across to another) and radial gradients (which radiate out from a single point).

In this article we'll only focus on the first type. The second one is a bit more complex, and less commonly used. You can find more information about both in the links at the end of the article.

A linear gradient is created by passing in a `linear-gradient()` function as the value of a `background-image` property. At a minimum, the function needs to take three parameters separated by commas — the direction the gradient should be going in across the background, the color at the beginning, and the color at the end. For example:

```
1 | background-image: linear-gradient(to bottom, orange, yellow);
```

This gradient would run from top to bottom, starting as orange at the top, and transitioning smoothly to yellow at the bottom. You can use keywords to specify the direction (`to bottom`, `to right`, `to bottom right`, etc.), or degree values (`0deg` is equivalent to `top`, `90deg` is equivalent to `to right`, up to `360deg`, which would be equivalent to `to top` again).

You can also specify other points along the gradient where the color is defined — these are called **color stops**, and the browser will work out the color gradients between each set of color stops. So for example:

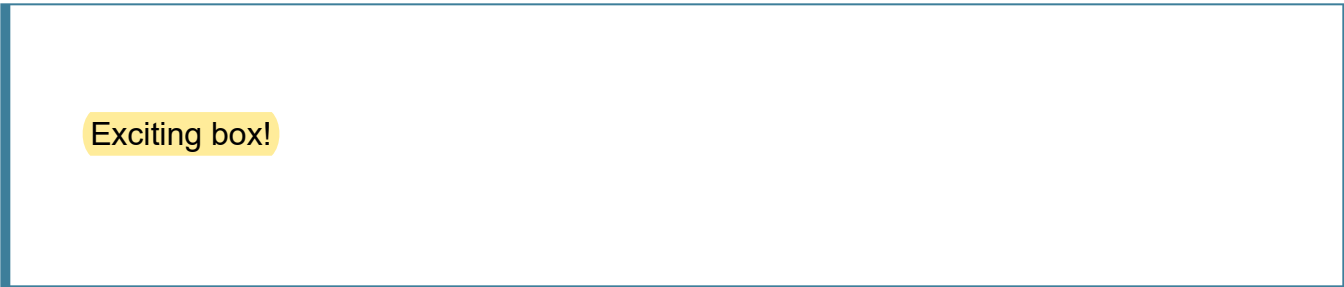
```
1 | background-image: linear-gradient(to bottom, yellow, orange 40%, yellow);
```

This gradient would run from top to bottom, starting at yellow, radiating to orange 40% of the way down the element, then back to yellow at 100%. You can specify as many color stops as you like, and you can also specify the position of those stops using other units, such as `rem`s, `px`, etc.

Let's add a linear gradient to our example:

```
1 p {  
2   font-family: sans-serif;  
3   padding: 20px;  
4   /* background properties */  
5   background-color: yellow;  
6   background-image: linear-gradient(to bottom, yellow, #ddd000 50%, orange);  
7 }
```

The result is as follows:



Exciting box!

Note that you can also set a repeating linear gradient using the `repeating-linear-gradient()` function. This works in exactly the same way, except that the pattern you set is repeated over and over again until the edge of the background. For example:

```
1 background-image: repeating-linear-gradient(to right, yellow, orange 25px, ye
```

This would create a gradient that gradients from yellow to orange and back again every 50 pixels along the gradient.

Background attachment [↗](#)

Another option we have available for backgrounds is specifying how they scroll when the content scrolls. This is controlled using the `background-attachment` property, which can take the following values:

- `scroll`: Causes the element's background to scroll when the page is scrolled. If the element content is scrolled, the background does not move. In effect, the background is fixed to the same position on the page, so it scrolls as the page scrolls.
- `fixed`: Causes an element's background to be fixed to the viewport, so that it doesn't scroll when the page or element content is scrolled. It will always remain in the same position on the screen.
- `local`: This value was added later on (it is only supported in Internet Explorer 9+, whereas the others are supported in IE4+) because the `scroll` value is rather confusing and doesn't really do what you want in many cases. The `local` value fixes the background to the element it is set on, so when you scroll the element, the background scrolls with it.

The `background-attachment` property only has an effect when there is content to scroll, so we've made a demo to demonstrate the differences between the three values — have a look at [background-attachment.html](#) (also [see the source code here](#)).

Background shorthand [↗](#)

It is possible to declare all of the property values discussed above (and some others, in newer browsers) in a single line, using the `background` property. To do this, you use the same values as you did in the individual properties, but you put them all on one line separated by spaces, like so:

1 | `background: yellow linear-gradient(to bottom, orange, yellow) no-repeat left`

Any properties not specified in the shorthand will be assigned default values. You can check out the [background reference page](#) to find out what those defaults are, and what other properties can be included in the `background` shorthand.

Let's go back to our exciting box example, and replace the existing properties with shorthand. We can replace all of these properties:

```
1 background-color: yellow;
2 background-image: linear-gradient(to bottom, yellow, #ddd000 50%, orange);
3 background-repeat: no-repeat;
4 background-position: 99% center;
```

With this:

```
1 background: yellow linear-gradient(to bottom, yellow, #ddd000 50%, orange) no
```

The final code looks like so:

```
1 p {
2     font-family: sans-serif;
3     padding: 20px;
4     /* background properties */
5     background: yellow linear-gradient(to bottom, yellow, #ddd000 50%, orange)
6 }
```

The result is as follows:

Exciting box!

Multiple backgrounds [↗](#)

Fairly recently (since Internet Explorer 9) we've had the ability to attach multiple backgrounds to a single element. This is a good thing, as multiple backgrounds are very useful. You separate

your different background definitions with commas:

```
1 background: url(image.png) no-repeat 99% center,  
2             url(background-tile.png),  
3             linear-gradient(to bottom, yellow, #ddd000 50%, orange);  
4 background-color: yellow;
```

And the backgrounds are stacked on top of one another with the first appearing at the top, then the second below it, then the third, etc. This is possibly not what you were expecting, so take care. Also note that we've put the fallback background color into a separate property declaration, because trying to include it in the multiple backgrounds seems to break things.

You can also put multiple values into longhand background-* properties, for example:

```
1 background-image: url(image.png), url(background-tile.png);  
2 background-repeat: no-repeat, repeat;
```

Although you are probably better off using background shorthand — not only is it quicker to write, but it is easier to see which background properties apply to which background.

Previous to multiple backgrounds being available, web developers had to put multiple <div> elements around their boxes and then apply a separate background image to each:

```
1 <div class="background1">  
2   <div class="background2">  
3     <div class="background3">  
4       <p>My content</p>  
5     </div>  
6   </div>  
7 </div>
```

This worked, but it resulted in pretty messy hard-to-read markup. You might still have to do this if you need to support old versions of Internet Explorer.

Let's put multiple backgrounds onto our exciting box example — we want to see the gradient and the fireball, both at the same time:

```
1 | p {  
2 |   font-family: sans-serif;  
3 |   padding: 20px;  
4 |   /* background properties */  
5 |   background-color: yellow;  
6 |   background: url(https://mdn.mozillademos.org/files/13026/fire-ball-icon.png  
7 |               linear-gradient(to bottom, yellow, #dddd00 50%, orange);  
8 | }
```

This gives us the following result:

Exciting box!

Note: You can find the completed example on Github (see the source code too).

Background size [↗](#)

As we mentioned earlier, there is a property available — `background-size` — which allows you to dynamically alter the size of a background image so that it fits better into your design. This is very useful in many ways, for example automatically correcting the size of icons that aren't uploaded correctly. Just bear in mind that this isn't supported by Internet Explorer versions lower than 9, so you can't rely on it if you need to support older browsers. For each `background-image`, you need to include two background size values — one for the horizontal dimension, and one for the vertical:

```
1 | background-image: url(myimage.png);  
2 | background-size: 16px 16px;
```

You can use all the length units you'd expect, to be able to specify the values you want — px, percentages, rems, etc.

You can see a good example of `background-size` in use in [Including icons on links](#).

Active learning: Playing with backgrounds [🔗](#)

This active learning session also has no right answers — here we want you to play with backgrounds and have some fun before you move on. You could:

- Set a different background color.
- Include a different background image — find an absolute path to an image to use inside the `url()` function.
- Apply a background gradient.
- Apply multiple backgrounds.
- Include a `background-size` property to alter the size of your background images.

If you are following this as part of a class or study group, your teacher or mentor might also set some additional tasks for you to complete.

If you make a mistake, you can always reset it using the *Reset* button.

HTML Input

```
<p>Exciting box!</p>
```

CSS Input

```
p {  
  font-family: sans-serif;  
  padding: 20px;  
  /* background properties */  
  background-color: yellow;  
  background:  
url(https://mdn.mozillademos.org/files/13026/fi  
re-ball-icon.png) no-repeat 99% center,  
          linear-gradient(to bottom,  
yellow 49%, #ddd000 49%, #ddd000 50%, orange); 1
```

Output

Exciting box!

Summary [↗](#)

This article should have taught you most of what you'll ever need to know about styling element backgrounds. Hopefully you'll have had some fun along the way too! In the next article we'll play with borders, and look at how to style those.

See also [↗](#)

- [CSS Linear Gradients](#)
- [CSS3 Radial Gradients](#)

[Previous](#)[Overview: Styling boxes](#)[Next](#)

In this module [↗](#)

- Box model recap
 - Backgrounds
 - Borders
 - Styling tables
 - Advanced box effects
 - Creating fancy letterheaded paper
 - A cool looking box
-

