Backgrounds and borders

- Background colors
- Background images
  - Controlling background-repeat
  - Sizing the background image
    - Playing with background-size
  - Positioning the background image
    - Playing with background-position
- Gradient backgrounds
- Multiple background images
  - Playing with multiple background images
- Background attachment
- Using the background shorthand property
- Accessibility considerations with backgrounds
- Borders
  - Playing with borders
- Rounded corners
  - Playing with border radius
- Summary

In this lesson, we will take a look at some of the creative things you can do with CSS backgrounds and borders. From adding gradients, background images, and rounded corners, backgrounds and borders are the answer to a lot of styling questions in CSS.

Prerequisites:	HTML basics (study Basic HTML syntax), CSS Values and units, CSS Sizing.

Learning outcomes:	• Basic background styling — colors and images. • Background image size, repeat, position, and attachment. • Background gradients — general concept and linear gradients (radial, conic, and repeating gradients are more advanced; in-depth knowledge is not required at this stage.) • Accessibility considerations of backgrounds — ensure good contrast. • Border basics — width, style, color, and border shorthand. Border radius for rounded corners.

Background colors

The background-color property defines the background color on any element in CSS. The property accepts any valid <color>. A `background-color` extends underneath the content and padding box of the element.

In the example below, we have used various color values to add a background color to the box, a heading, and a span element.

Try editing the example and swapping out the specified colors with any available <color> values.

```
<div class="box">
  <h2>Background Colors</h2>
  <p>Try changing the background <span>colors</span>.</p>
</div>
```

```
.box {
  padding: 0.3em;
  background-color: #567895;
}

h2 {
  background-color: black;
  color: white;
}
span {
  background-color: rgb(255 255 255 / 50%);
```

```
}
```

Background images

The background-image property enables the display of an image in the background of an element. In the example below, we have two boxes — one has a background image that is larger than the box (balloons.jpg). The other has a small image of a single star (star.png).

This example demonstrates two things about background images. By default, the large image is not scaled down to fit the box, so we only see a small corner of it, whereas the small image is tiled to fill the box.

```
<div class="wrapper">
  <div class="box a"></div>
  <div class="box b"></div>
</div>
```

```
.wrapper {
  display: flex;
}

.box {
  width: 200px;
  height: 80px;
  padding: 0.5em;
  border: 1px solid #cccccc;
  margin: 20px;
}

.a {
  background-image: url("https://mdn.github.io/shared-assets/images/examples/balloons.jpg");
}

.b {
  background-image: url("https://mdn.github.io/shared-assets/images/examples/star.png");
}
```

If you specify a background color in addition to a background image then the image displays on top of the color. Try adding a `background-color` property to the example above to see that in action.

Controlling background-repeat

The background-repeat property is used to control the tiling behavior of images. The available values are:

- `no-repeat` — stop the background from repeating altogether.
- `repeat-x` — repeat horizontally.
- `repeat-y` — repeat vertically.
- `repeat` — the default; repeat in both directions.
- `space` — repeat as many times as possible, adding space between the images if there is extra space available.
- `round` — similar to `space`, but stretches the images to fill any extra space

Try these values out in the example below. We have set the value to `no-repeat` so you will only see one star. Try out the different values to see what their effects are.

```
<div class="box"></div>
```

```
.box {
  width: 200px;
  height: 80px;
  padding: 0.5em;
  border: 1px solid #cccccc;
  margin: 20px;
}
```

```
.box {
  background-image: url("https://mdn.github.io/shared-assets/images/examples/star.png");
  background-repeat: no-repeat;
}
```

Sizing the background image

The *balloons.jpg* image used in the initial background image example is a large image that was cropped due to being larger than the element it is a background of. In this case, we can use the background-size property to size the image to fit inside the background.

`background-size` can take two length or percentage values to specify the image's size on the horizontal and vertical directions, or the following keywords:

- `cover` — the browser will make the image just large enough so that it completely covers the box area while still retaining its aspect ratio. In this case, part of the image is likely to end up outside the box.
- `contain` — the browser will make the image the right size to fit inside the box. In this case, you may end up with gaps on either side or on the top and bottom of the image, if the aspect ratio of the image is different from that of the box.

Playing with `background-size`

In the example below, the *balloons.jpg* image has length units set to make it fit inside the box. You can see this has distorted the image.

Try the following:

- Change the length units used to modify the size of the background.
- Remove the length units and see what happens when you use `background-size: cover` or `background-size: contain`.
- Size the image smaller than the box, then change the value of `background-repeat` to repeat the image.

```
<div class="box"></div>
```

```
.box {
  width: 500px;
  height: 100px;
  padding: 0.5em;
  border: 1px solid #cccccc;
  margin: 10px;
}
```

```
.box {
  background-image: url("https://mdn.github.io/shared-assets/images/examples/balloons.jpg");
  background-repeat: no-repeat;
  background-size: 80px 10em;
}
```

Positioning the background image

The background-position property allows you to choose the position in which the background image appears on the box it is applied to. This uses a coordinate system in which the top-left-hand corner of the box is `(0,0)`, and the box is positioned along the horizontal (`x`) and vertical (`y`) axes.

> **NOTE:** The default `background-position` value is `(0,0)`.

The most common `background-position` values take two individual values — a horizontal value followed by a vertical value. You can use keywords such as `top` and `right` (look up the others on the background-position page):

```
.box {
  background-image: url("image.png");
  background-repeat: no-repeat;
  background-position: top center;
}
```

You can also use lengths and percentages:

```
.box {
  background-image: url("image.png");
  background-repeat: no-repeat;
  background-position: 20px 10%;
}
```

You can also mix keyword values with lengths or percentages, in which case the first value refers to the horizontal position and the second refers to the vertical position. For example:

```
.box {
  background-image: url("image.png");
  background-repeat: no-repeat;
  background-position: 20px top;
}
```

Finally, you can also use a 4-value syntax to indicate a distance from certain edges of the box. Each pair of values represents the edge of box to offset from, and the size of the offset from that edge. In the snippet below, we are positioning the background `20px` from the `top` and `10px` from the `right`:

```
.box {
  background-image: url("image.png");
  background-repeat: no-repeat;
  background-position: top 20px right 10px;
}
```

Playing with `background-position`

Use the example below to play around with these values and move the star around inside the box:

```
<div class="box"></div>
```

```
.box {
  width: 500px;
  height: 80px;
  padding: 0.5em;
  border: 1px solid #cccccc;
  margin: 20px;
}
```

```
.box {
  background-image: url("https://mdn.github.io/shared-assets/images/examples/star.png");
  background-repeat: no-repeat;
  background-position: 120px 1em;
}
```

> **NOTE:** The shorthand `background-position` is used instead of background-position-x and background-position-y, which allow you to set the different axis position values individually.

Gradient backgrounds

A gradient — when used for a background — acts just like an image and is also set by using the background-image property.

You can find out about the different types of gradient value and things you can do with them on the MDN page for the gradient data type.

Try some different gradient values in the example below. Initially, we have a linear gradient that is stretched over the whole first box, and a radial gradient with a set size, repeating over the second box.

```
<div class="wrapper">
  <div class="box a"></div>
  <div class="box b"></div>
</div>
```

```
.wrapper {
  display: flex;
}

.box {
  width: 400px;
  height: 80px;
  padding: 0.5em;
  border: 1px solid #cccccc;
  margin: 20px;
}

.a {
  background-image: linear-gradient(
    105deg,
    rgb(0 249 255 / 100%) 39%,
    rgb(51 56 57 / 100%) 96%
  );
}

.b {
  background-image: radial-gradient(
    circle,
    rgb(0 249 255 / 100%) 39%,
    rgb(51 56 57 / 100%) 96%
  );
  background-size: 100px 50px;
}
```

> **NOTE:** A fun way to play with gradients is to use one of the many CSS gradient generators available on the web, such as CSSGradient.io. You can create a gradient and copy and paste the source code that generates it.

Multiple background images

It is also possible to specify multiple background images in a single declaration. You do this by specifying multiple `background-image` values separated by commas.

When you do this you may end up with background images overlapping each other. The backgrounds will layer with the last listed background image at the bottom of the stack, and each previous image stacking on top of the one that follows it in the code.

> **NOTE:** Gradients can be happily mixed with regular background images.

The other `background-*` properties can also have comma-separated values in the same way as `background-image`:

```
background-image:
  url("image1.png"), url("image2.png"), url("image3.png"), url("image4.png");
background-repeat: no-repeat, repeat-x, repeat;
background-position:
  10px 20px,
  top right;
```

Each value of the different properties will match up to the values in the same position in the other properties. Above, for example, `image1`'s `background-repeat` value will be `no-repeat`. However, what happens when different properties have different numbers of values? The answer is that the smaller numbers of values will cycle — in the above example there are four background images but only two `background-position` values. The first two position values will be applied to the first two images, then they will cycle back around again — `image3` will be given the first position value, and `image4` will be given the second position value.

Playing with multiple background images

Let's play. The example below includes two background images. Try editing the example as follows:

- To demonstrate the stacking order, try switching which background image comes first in the list.
- Add some other `background-*` properties to change the position, size, or repeat value of the images.
- Try adding a gradient in as a third `background-image`.

```
<div class="wrapper">
  <div class="box"></div>
</div>
```

```
.wrapper {
  display: flex;
}

.box {
  width: 500px;
  height: 80px;
  padding: 0.5em;
  border: 1px solid #cccccc;
  margin: 20px;
}

.box {
  background-image:
    url("https://mdn.github.io/shared-assets/images/examples/star.png"),
    url("https://mdn.github.io/shared-assets/images/examples/big-star.png");
}
```

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`: 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:

```
<section>
  <article class="scroll">
    <p>
      <code>background-attachment: scroll</code> causes the element's background
      to be fixed to the page, so that it scrolls when the page is scrolled. If
      the element content is scrolled, the background does not move.
    </p>

    <pre></pre>
  </article>

  <article class="fixed">
    <p>
      <code>background-attachment: fixed</code> 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.
    </p>

    <pre></pre>
  </article>

  <article class="local">
    <p>
      <code>background-attachment: local</code> causes an element's background
      to be fixed to the actual element itself. When the page is scrolled, the
      element's background will move along with it only if the element does so.
      When the element's content is scrolled, the background will scroll along
      with it.
    </p>

    <pre></pre>
  </article>
</section>
```

```
html,
body {
  margin: 0;
  padding: 0;
}
```

```
h1 {
  margin-top: 0;
}

body {
  padding: 1em;
}

html {
  background-color: yellow;
  font-family: sans-serif;
}

body {
  height: 2000px;
}

p {
  padding: 10px;
  color: white;
  background: rgb(0 0 0 / 0.3);
}

section {
  display: flex;
  gap: 10px;
}

article {
  flex: 1;
  height: 300px;
  background-color: rgb(0 0 0 / 0.5);
  background-image: url("https://mdn.github.io/shared-assets/images/examples/grapefruit-
slice.jpg");
  background-size: 400px 400px;
  background-repeat: no-repeat;
  background-position: top center;
  padding: 1%;
  overflow: auto;
```

```
}

article pre {
  height: 800px;
}

.fixed {
  background-attachment: fixed;
}

.scroll {
  background-attachment: scroll;
}

.local {
  background-attachment: local;
}
```

Try scrolling the whole of the embedded example and then the individual containers, and observe the differences in behavior of the container backgrounds.

Using the background shorthand property

You will often see backgrounds specified using the background shorthand property, which lets you set all of the different properties at once.

If using multiple backgrounds, you need to specify all of the properties for the first background, then add your next background after a comma. In the example below we have a gradient with a size and position, then an image background with `no-repeat` and a position, then a color.

There are a few rules that need to be followed when writing background image shorthand values, for example:

- A `background-color` may only be specified after the final comma.
- The value of `background-size` may only be included immediately after `background-position`, separated with the `/` character, like this: `center/80%`.

Take a look at the MDN page for background to learn more about the syntax.

```
<div class="box"></div>
```

```
.box {
  width: 500px;
  height: 300px;
  padding: 0.5em;
  background:
    linear-gradient(
        105deg,
        rgb(255 255 255 / 20%) 39%,
        rgb(51 56 57 / 100%) 96%
      )
      center center / 400px 200px no-repeat,
    url("https://mdn.github.io/shared-assets/images/examples/big-star.png")
      center no-repeat,
    rebeccapurple;
}
```

Accessibility considerations with backgrounds

When placing text on top of a background image or color, you should take care that you have enough contrast for the text to be legible for your visitors. When specifying an image with text content on top of it, you should also specify a `background-color` that will allow the text to be legible if the image does not load.

Screen readers cannot parse background images; therefore, they should be purely decorative. Any important content should be part of the HTML page and not contained in a background.

Borders

When learning about the box model, we discovered how borders affect the size of our box. In this lesson, we will look at how to use borders creatively.

Typically, when we add borders to an element with CSS we use the border shorthand property to set the color, width, and style of the border on all four sides of a box in one declaration:

```
.box {
  border: 1px solid black;
}
```

Or we can target one edge of the box, for example:

```
.box {
  border-top: 1px solid black;
}
```

The individual properties include the border-width, border-style, and border-color shorthand properties:

```
.box {
  border-width: 1px;
  border-style: solid;
  border-color: black;
}
```

There are also longhand properties for width, style, and color for each of the four sides:

```
.box {
  border-top-width: 1px;
  border-top-style: solid;
  border-top-color: black;
}
```

> **NOTE:** These top, right, bottom, and left border properties also have mapped *logical* border properties that relate to the writing mode of the document (e.g., left-to-right or right-to-left text, or top-to-bottom). You can read about these in handling different text directions.

Playing with borders

There are a variety of styles that you can use for borders. In the example below, we have used two different border styles for the box and two different border styles for the heading. Play with the border style, width, and color to see how borders work.

```
<div class="box">
  <h2>Borders</h2>
  <p>Try changing the borders.</p>
</div>
```

```
* {
  padding: 0.2em;
}
.box {
  width: 500px;
  background-color: #567895;
  border: 5px solid #0b385f;
  border-bottom-style: dashed;
  color: white;
}

h2 {
  border-top: 2px dotted rebeccapurple;
  border-bottom: 1em double rgb(24 163 78);
}
```

Rounded corners

You can add rounded corners to a box using the border-radius property and associated longhands that relate to each corner of the box. Two lengths or percentages may be used as a value, the first value defining the horizontal radius, and the second the vertical radius. In a lot of cases, you will only pass in one value, which will be used for both.

For example, to give all four corners of a box a `10px` radius:

```
.box {
  border-radius: 10px;
}
```

Or to make the top right corner have a horizontal radius of `1em`, and a vertical radius of `10%`:

```
.box {
  border-top-right-radius: 1em 10%;
}
```

> **NOTE:** Like with the border properties above, these border-radius properties also have mapped *logical* border-radius properties.

Playing with border radius

We have set all four corners in the example below and then changed the values for the top right corner to make it different. You can play with the values to change the corners. Take a look at the property page for border-radius to see the available syntax options. The border-radius generator can be used to output rounded corner values for you.

```
<div class="box">
  <h2>Borders</h2>
  <p>Try changing the borders.</p>
</div>
```

```
.box {
  width: 500px;
  height: 110px;
  padding: 0.5em;
  border: 10px solid rebeccapurple;
  border-radius: 1em;
  border-top-right-radius: 10% 30%;
}
```

Summary

You can see that there is quite a lot to adding a background or a border to a box. Explore the different property pages if you want to find out more about any of the features discussed here. Almost every page on MDN has examples for you to play with to enhance your knowledge.

In the next article, we'll give you some tests that you can use to check how well you've understood and retained the information we've provided on background and border styling.

□□□ #3
□ □□□ □□□ 2026-10-31 03:27:53 UTC
□ cocobook-sync □□□ 2026-11-04 17:32:54 UTC