Backgrounds

- Background color
- Background images
  - Setting a background-image with the url CSS function
  - CSS gradient backgrounds
- Repeating background images
- Background position
- Background Size
- Background attachment
- Background origin
- Background clip
- Multiple backgrounds
- The background shorthand

Behind every CSS box is a specialized layer called the background layer. CSS provides a variety of ways to make meaningful changes to it–including allowing multiple backgrounds.

Background layers are furthest from the user, rendered behind the contents of a box starting from its `padding-box` region. This enables the background layer to not overlap with borders at all.

> {% Codepen { user: 'argyleink', id: 'BaLedvd', theme: 'dark', height: 700, tab: 'result' } %}

Background color

{% BrowserCompat 'css.properties.background-color' %}

One of the simplest effects you can apply to a background layer is setting the color. The initial value of `background-color` is `transparent`, which allows the contents of a parent to be visible. A valid color set on a background layer sits behind other things painted on that element.

{% Codepen { user: 'web-dot-dev', id: 'GRvqQZZ', theme: 'dark', height: 850, tab: 'result' } %}

Background images

{% BrowserCompat 'css.properties.background-image' %}

On top of the `background-color` layer, you can add a background image, using the `background-image` property. A `background-image` accepts the following:

- An image URL or data URI using the `url` CSS function.
- An image dynamically created by a gradient CSS function.

Setting a background-image with the `url` CSS function

{% Codepen { user: 'web-dot-dev', id: 'JjJNNro', theme: 'dark', height: 525, tab: 'result' } %}

CSS gradient backgrounds

Several gradient CSS functions exist to allow you to generate a background-image, when passed two or more colors.

Regardless of which gradient function is used, the resulting image is intrinsically sized to match the amount of space available.

Demo showing example of applying a background-image using gradient functions:

{% Codepen { user: 'web-dot-dev', id: 'oNeLZWa', theme: 'dark', height: 600, tab: 'result' } %}

Repeating background images

{% BrowserCompat 'css.properties.background-repeat' %}

By default, background images repeat horizontally and vertically to fill the entire space of the background layer.

Change this by using the `background-repeat` property with one of the following values:

- `repeat`: The image repeats within the space available, cropping as necessary.
- `round`: The image repeats horizontally and vertically to fit as many instances into the space available, without cropping, compressing, or stretching it.
- `space`: The image repeats horizontally and vertically to fit as many instances within the space available without cropping—spacing out instances of the image as needed. Repeating images touch the edges of the space a background layer occupies, with white space evenly distributed between them.

The `background-repeat` property allows you to set the behavior for the x and y axis independently. The first parameter sets the horizontal repeat behavior, and the second parameter sets the vertical repeat behavior.

If you use a single value, it will be applied to both the horizontal and vertical repeats.

The shorthand also has convenient one-word options to make your intent clearer.

The value `repeat-x` repeats an image only horizontally; this is equivalent to `repeat no-repeat`.

The following demo demonstrates these capabilities of the `background-repeat` property:

> {% Codepen { user: 'web-dot-dev', id: 'KKvMmjb', theme: 'dark', height: 1100, tab: 'result' } %}

Background position

{% BrowserCompat 'css.properties.background-position' %}

You may have noticed when some images on the Web are styled with a `background-repeat: no-repeat` declaration, such images are displayed top left of their container.

The initial position of background images is top left. The `background-position` property allows you to change this behavior by offsetting the image position.

As with `background-repeat`, the `background-position` property allows you to position images along the x and y axis independently with two values by default.

When CSS lengths and percentages are used, the first parameter corresponds to the horizontal axis while the second parameter corresponds to the vertical axis.

When keywords are only used the order of the keywords does not matter:

{% Compare 'better' %}

```
background-position: left 50%;
```

{% endCompare %}

{% Compare 'better' %}

```
background-position: top left;
```

{% endCompare %}

{% Compare 'better' %}

```
background-position: left top;
```

{% CompareCaption %}

Order does not matter for keywords associated with different axes of position.

{% endCompareCaption %}

{% endCompare %}

{% Compare 'worse' %}

```
   background-position: 50% left;
```

{% CompareCaption %}

When CSS values are used alongside keywords, the order matters. The first value represents the horizontal axis and the second the vertical axis. {% endCompareCaption %}

{% endCompare%}

{% Compare 'worse' %}

```
   background-position: left right;
```

{% CompareCaption %}

You cannot use keywords associated with the same axis simultaneously.

{% endCompareCaption %}

{% endCompare %}

The `background-position` property also has a convenient one value shorthand; the omitted value resolves to `50%`. Here's an example that demonstrates this using the keywords the `background-position` property accepts:

> {% Codepen { user: 'web-dot-dev', id: 'YzxWQqx', theme: 'dark', height: 950, tab: 'result' } %}

In addition to its default two parameter form and one parameter form; the `background-position` property also accepts up to four parameters;

When three or four parameters are used, a CSS length or percentage must be preceded by the `top`, `left`, `right`, or `bottom` keywords in order for the browser to calculate which edge of the CSS box the offset should originate from.

When three parameters are used, a CSS length or value can be the second or third parameter with the other two being keywords; the keyword it succeeds will be used to determine the edge the CSS length or value corresponds to being the offset of. The offset of the other keyword specified is set to 0.

{% Compare 'better' %}

```
background-position: bottom 88% right;
```

{% endCompare %}

{% Compare 'better' %}

```
background-position: right bottom 88%;
```

{% endCompare %}

{% Compare 'worse' %}

```
background-position: 88% bottom right;
```

{% CompareCaption %} CSS length value must be preceded by the `top`, `right`, `bottom`, or `left` keywords when using three or more parameters. {% endCompareCaption %} {% endCompare %}

{% Compare 'better' %}

```
background-position: bottom 88% right 33%;
```

{% endCompare %}

{% Compare 'better' %}

```
background-position: right 33% bottom 88%;
```

{% endCompare %}

{% Compare 'worse' %}

```
background-position: 88% 33% bottom left;
```

{% CompareCaption%} CSS length value must be preceded by the `top`, `right`, `bottom`, or `left` keywords when using three or more parameters. {% endCompareCaption %} {% endCompare %}

If `background-position: top left 20%`is applied to a CSS background image, the image is placed at the top of the box, the `20%` value represents a 20% offset from the left of the box (on the x axis).

If `background-position: top 20% left` is applied to a CSS background image, the 20% value represents a 20% offset from the top of the CSS box (on the y axis), and the image is placed at the left of the box.

When four parameters are used, the two keywords are paired with two values corresponding to an offset against the origins of each keyword specified. If `background-position: bottom 20% right 30%` is applied to a background-image, the background-image is positioned 20% from the bottom, and 30% from the right of the CSS box.

The following demo demonstrates this behavior:

{% Codepen { user: 'web-dot-dev', id: 'porbwrM', theme: 'dark', tab: 'result' } %}

Here are more examples of using the `background-position` property using a mix of CSS and keyword values:

{% Aside %} To learn more about the nuances related to positioning backgrounds, check out `background-position` on MDN. {% endAside %}

Background Size

{% BrowserCompat 'css.properties.background-size' %}

The `background-size` property sets the size of background images; By default background images are sized based on their intrinsic (actual) width, height, and aspect ratio.

The `background-size` property uses CSS length and percentage values or specific keywords. The property accepts up to two parameters corresponding to allowing you to change width and height of a background independently.

The `background-size` property accepts the following keywords:

- `auto`: When used independently, the background image is sized based on its intrinsic width and height; when `auto` is used alongside another CSS value for the width (first parameter) or height (second parameter), the dimension set to `auto` is sized as needed to maintain the natural aspect ratio of the image. This is the default behavior of the `background-size` property.
- `cover`: Covers the entire area of the background layer. This may mean the image is stretched or cropped.
- `contain`: Sizes the image to fill the space without stretching or cropping. As a result, empty space can remain that will cause the image to repeat, unless `background-repeat` is set to `no-repeat`.

The latter 2 are intended to be used in a standalone fashion without another parameter.

The following demo demonstrates these keywords in action:

Demo demonstrating applying these keywords to `background-size`:

> {% Codepen { user: 'web-dot-dev', id: 'YzxWQYY', theme: 'dark', height: 700, tab: 'result' } %}

Background attachment

{% BrowserCompat 'css.properties.background-attachment' %}

The `background-attachment` property enables you to modify the fixed position behavior of background images (images part of a background layer) once the layer is visible on a screen.

It accepts 3 keywords: `scroll`, `fixed`, and `local`.

The default behavior of the `background-attachment` property is the initial value of `scroll`. When more space is needed, the images move with that space within the background layer determined by the bounds of the CSS box.

> {% Codepen { user: 'web-dot-dev', id: 'mdwwzOe', theme: 'dark', height: 1000, tab: 'result' } %}

Using the value `fixed` fixes the position of background images to the viewport.

Once the space of the background layer images originally takes up needs to be scrolled (or rendered) offscreen, images within the background layer stay fixed in the original position the background layer enabled them to be until the entire layer is scrolled off screen by the viewport.

{% Codepen { user: 'web-dot-dev', id: 'MWoozvN', theme: 'dark', height: 1000, tab: 'result' } %}

The `local` keyword enables the position of background images to be fixed relative to the element's contents. Background images now move along the space they occupy as that space renders inside and outside the bounds of the CSS box (usually due to scrolling, 2D, or 3D transformations).

{% Codepen { user: 'web-dot-dev', id: 'WNExZmK', theme: 'dark', height: 1000, tab: 'result' } %}

Background origin

{% BrowserCompat 'css.properties.background-origin' %}

The `background-origin` property enables you to modify the area of backgrounds associated with a particular box. The values the property accepts correspond to the `border-box` , `padding-box`, and `content-box` regions of a box .

Try these options out using the following demo:

{% Codepen { user: 'web-dot-dev', id: 'ExvyXeZ', theme: 'dark', height: 650, tab: 'result' } %}

Background clip

{% BrowserCompat 'css.properties.background-clip' %}

The `background-clip` property controls what is visually seen from a background layer regardless of the bounds created by the `background-origin` property.

Like `background-origin` the regions that can be specified are `border-box`, `padding-box`, and `content-box` corresponding to where a CSS background layer can be rendered. When these keywords are used, any rendering of the background further than the region specified will be cropped or clipped.

{% Codepen { user: 'web-dot-dev', id: 'vYJKZba', theme: 'dark', height: 650, tab: 'result' } %}

The `background-clip` property also accepts a `text` keyword that clips the background to be no further than the text within the content box. For this effect to be evident in the actual text within a CSS box, the text must be partially or completely transparent.

A relatively new property, at the time of this writing, Chrome and most browsers require the `-webkit-` prefix to use this property.

{% BrowserCompat 'css.properties.background-clip' %}

> {% Codepen { user: 'web-dot-dev', id: 'qBjjweL', theme: 'dark', height: 650, tab: 'result,css' } %}

{% Aside 'gotchas' %} It should be also noted this property is not compatible with `background-clip: text` being simultaneously set on a CSS box. {% endAside %}

Multiple backgrounds

As mentioned at the beginning of the module, the background layer allows multiple sublayers to be defined. For brevity, I'll refer to these sublayers as backgrounds.

Multiple backgrounds are defined top to bottom; The first background is the closest to the user, while the last background is the furthest from the user.

The only background defined or the last layer is designated the *final background layer* by the browser. Only this layer is allowed to assign a `background-color`.

{% Aside %} With the unpredictability of the web in mind, a background-image may fail to load. Setting a background-color on the final layer ensures good contrast for text and so on if important background layers fail to load. {% endAside %}

Multiple layers can be individually configured using most background-related CSS properties that are comma separated, as demonstrated in the code snippet and live demo below.

```
background-image: url("https://assets.codepen.io/7518/pngaaa.com-1272986.png"),
    url("https://assets.codepen.io/7518/blob.svg"),
    linear-gradient(hsl(191deg, 40%, 86%), hsla(191deg, 96%, 96%, 0.5));
  background-size: contain, cover, auto;
  background-repeat: no-repeat, no-repeat, no-repeat;
  background-position: 50% 50%, 10% 50%, 0% 0%;
  background-origin: padding-box, border-box, content-box;
```

> {% Codepen { user: 'web-dot-dev', id: 'dyRzQBz', theme: 'dark', height: 650, tab: 'css' } %}

The background shorthand

To make it easier to style the background layer of a box-especially when multiple background layers are desired–there is a shorthand that follows the following specific pattern:

```
background:
  <background-image>
  <background-position> / <background-size>?
  <background-repeat>
  <background-attachment>
  <background-origin>
  <background-clip>
  <background-color>?
```

{% Aside 'gotchas' %} It's important to note that any background properties omitted in the shorthand are set to their initial values. {% endAside %}

Order is important in the shorthand form of declaring multiple backgrounds. The position and size values must both be provided, separated by a slash (`/`). Declaring the origin and clip behavior in the correct order allows you take advantage of setting keywords that are valid for both simultaneously

The following declaration clips the background, and originates it from the content box:

```
background: url("https://assets.codepen.io/7518/blob.svg") 50%
      50% / contain no-repeat content-box;
```

With these shorthand semantics in mind, the previous background-related declarations of the code snippet could be rewritten to be following:

```
background: url("https://assets.codepen.io/7518/pngaaa.com-1272986.png") 50% 50%/contain no-repeat padding-box, url("https://assets.codepen.io/7518/blob.svg") 10% 50% / cover border-box, linear-gradient(hsl(191deg, 40%, 86%), hsla(191deg, 96%, 96%, 0.5) ) 0% 0% / cover no-repeat content-box ;
```

{% Assessment 'backgrounds' %}

□□□ #2
□ □□□ □□□ 2026-10-31 22:45:12 UTC
□ □□□ □□□ 2026-10-31 22:45:14 UTC