# Using Performant Next-Gen Images in CSS with image-set

[Ollie Williams](https://css-tricks.com/author/olliew/) on

Learn Development at [Frontend Masters](https://frontendmasters.com/?utm_source=css-tricks&utm_medium=website&utm_campaign=css-tricks-monthly-site-sponsor)

The CSS image-set() function has been supported in Chromium-based browsers since 2012 and in Safari since version 6. Support recently landed in [Firefox 88](https://hacks.mozilla.org/2021/04/never-too-late-for-firefox-88/). Let’s dive in and see what we can and can’t do today with image-set().

#### Multiple resolutions of the same image

Here’s what the [CSS spec](https://drafts.csswg.org/css-images-4/" \l "image-set-notation) has to say about image-set():

Delivering the most appropriate image resolution for a user’s device can be a difficult task. Ideally, images should be in the same resolution as the device they’re being viewed in, which can vary between users. However, other factors can factor into the decision of which image to send; for example, if the user is on a slow mobile connection, they may prefer to receive lower-res images rather than waiting for a large proper-res image to load.

It’s basically a CSS background equivalent to the [HTML](https://css-tricks.com/a-guide-to-the-responsive-images-syntax-in-html/" \l "using-srcset) srcset attribute for img tags. By using image-set we can provide multiple resolutions of an image and trust the browser to make the best decision about which one to use. This can be used to specify a value for three different CSS properties: content, cursor, and most useful of all, background-image.

.hero {

background-image: image-set("platypus.png" 1x, "platypus-2x.png" 2x);

}

1x is used to identify the low-res image, while 2x is used to define the high-res image. x is an alias of dppx, which stands for dots per pixel unit.

Chrome/Edge/Opera/Samsung Internet currently require a -webkit- prefix. If you’re using Autoprefixer, this will be handled automatically. Safari no longer requires the prefix but uses an older syntax that requires a url() function to specify the image path. We could also include a regular old background-image: url() to support any browsers that don’t support image-set.

.hero {

/\* Fallback \*/

background-image: url("platypus.png");

/\* Chrome/Edge/Opera/Samsung, Safari will fallback to this as well \*/

background-image: -webkit-image-set(url("platypus.png") 1x, url("platypus-2x.png") 2x);

/\* Standard use \*/

background-image: image-set("platypus.png" 1x, "platypus-2x.png" 2x);

}

Now users on expensive fancy devices will see a super sharp image. Performance will be improved for users on slow connections or with cheaper screens as their browser will automatically request the lower-res image. If you wanted to be sure that the high-res image was used on high-res devices, even on slow connections, you could make use of the min-resolution media query instead of image-set. For more on serving sharp images to high density screens, check out Jake Archibald’s recent post [over on his blog](https://jakearchibald.com/2021/serving-sharp-images-to-high-density-screens/).

That’s pretty cool, but what I really want is to be able to adopt the latest image formats in CSS while still catering for older browsers…

### New image formats

Safari 14 shipped support for WebP. It was the final modern browser to do so which means the image format is now supported everywhere (except Internet Explorer). WebP is useful in that it can make images that are often smaller than (but of the same quality as) JPG, PNG, or GIF.

There’s also a whole bunch of even newer image formats cropping up. [AVIF images](https://css-tricks.com/avif-has-landed/) are shockingly tiny. Chrome, Opera and Samsung Internet have already shipped support for AVIF. It’s already in Firefox behind a flag. This image format isn’t supported by many design tools yet but you can convert images to AVIF using the [Squoosh app](https://squoosh.app/) built by the Chrome team at Google. WebP 2, HEIF and JPEG XL might also make it into browsers eventually. This is all rather exciting, but we want browsers that don’t support these newer formats to get some images. Fortunately image-set() has a syntax for that.

#### Using new image formats by specifying a type

Browser support note: The feature of image-set that I’m about to talk about currently has pretty terrible browser support. Currently it’s only supported in [Firefox 89](https://developer.mozilla.org/en-US/docs/Mozilla/Firefox/Releases/89).

HTML has supported the <picture> element for years now.

<picture>

<source srcset="./kitten.avif" type="image/avif">

<img src="./kitten.jpg" alt="a small kitten">

</picture>

image-set provides the CSS equivalent, allowing for the use of next-gen image formats by specifying the image’s MIME type:

.div1 {

background-image: image-set(

"kitten.avif" type("image/avif"),

"kitten.jpg" type("image/jpeg")

);

}

The next-gen image goes first while the fallback image for older browsers goes second. Only one image will be downloaded. If the browser doesn’t support AVIF it will ignore it and only download the second image you specify. If AVIF is supported, the fallback image is ignored.

In the above example we used an AVIF image and provided a JPEG as a fallback, but the fallback could be any widely supported image format. Here’s an example using a PNG.

.div2 {

background-image: image-set(

"puppy.webp" type("image/webp"),

"puppy.png" type("image/png")

);

}

In Chromium and Safari, specifying the type is not supported yet. That means you can use image-set today only to specify different resolutions of widely-supported image formats but not to add backwards-compatibility when using WebP or AVIF in those browsers. It should be possible to provide both multiple resolutions and multiple image formats, if you are so inclined:

.div2 {

background-image: image-set(

"puppy.webp" type("image/webp") 1x,

"puppy2x.webp" type("image/webp") 2x,

"puppy.png" type("image/png") 1x,

"puppy2x.png" type("image/png") 2x

);

}

Hopefully browser support will improve soon.

#### Using <picture> for backgrounds instead

Maybe you don’t need background-image at all. If you want to use modern image formats, you might be able to use the <picture> element, which has better browser support. If you set the image to position: absolute it’s easy to display other elements on top of it.

As an alternative approach to using position: absolute, CSS grid is another easy way to overlap HTML elements.

https://css-tricks.com/using-performant-next-gen-images-in-css-with-image-set/

# How To Scale and Crop Images with CSS object-fit

[CSS](https://www.digitalocean.com/community/tags/css)

* By Alligator.io

Last Validated onApril 13, 2021 Originally Published onSeptember 3, 2020 176.3k views

### Introduction

You will likely encounter a scenario where you will want to preserve the original aspect ratio when working with images. Preserving the aspect ratio will prevent images from appearing distorted by either being stretched or squished. A common solution for this problem is to use the background-image CSS property. A more modern approach would be to use the object-fit CSS property.

In this article, you will explore the effects of the fill, cover, contain, none, and scale-down values available to the object-fit CSS property and how it can crop and scale images. You will also explore the object-position CSS property and how it can offset images.

## Prerequisites

If you would like to follow along with this article, you will need:

* Understanding [CSS property and values](https://developer.mozilla.org/en-US/docs/Web/CSS/Syntax).
* Using [CSS declarations inline with the style property](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/style).
* A code editor.
* A modern web browser that supports object-fit and object-position.

## Observing the Default Behavior of a Sample Image

Consider the following code used to display a sample image:

<img

src="https://assets.digitalocean.com/articles/alligator/css/object-fit/example-object-fit.jpg"

width="600"

height="337"

style="width: 600px; height: 337px;"

alt="Sample image of a turtle riding on top of an alligator that is swimming in the water - scaled to 600 x 337."

/>

This code will produce the following result in the browser:



This image has an original width of 1200px and a height of 674px. Using img attributes, the width has been set to 600 and 337 - half the original dimensions - preserving the aspect ratio.

Now, consider a situation where the layout expects images to occupy a width of 300px and a height of 337px:

<img

src="https://assets.digitalocean.com/articles/alligator/css/object-fit/example-object-fit.jpg"

width="600"

height="337"

style="width: 300px; height: 337px;"

alt="Sample image of a tutle riding on top of an alligator that is swimming in the water - scaled to 300 x 337."

/>

This code will produce the following result in the browser:



The resulting image no longer preserves the original aspect ratio and appears to be visually “squished”.

## Using object-fit: fill

The fill value is the initial value for object-fit. This value will not preserve the original aspect ratio.

<img

...

style="width: 300px; height: 337px; object-fit: fill;"

...

/>

This code will produce the following result in the browser:



As this is the “initial” value for browser rendering engines, there is no change in appearance from the scaled image. The resulting image still appears squished.

## Using object-fit: cover

The cover value preserves the original aspect ratio, but the image occupies all the available space.

<img

...

style="width: 300px; height: 337px; object-fit: cover;"

...

/>

This code will produce the following result in the browser:



In certain situations, object-fit: cover will result in the image appearing cropped. In this example image, some parts of the original image on the left and right do not appear because they cannot fit within the bounds of the declared width.

## Using object-fit: contain

The contain value preserves the original aspect ratio, but the image is also constrained to not exceed the bounds of the available space.

<img

...

style="width: 300px; height: 337px; object-fit: contain;"

...

/>

This code will produce the following result in the browser:



In certain situations, object-fit: contain will result in the image not filling all the available space. In this example image, there is vertical space above and below the image because the declared height is taller than the scaled-down height.

## Using object-fit: none

The none value does not resize the image at all.

<img

...

style="width: 300px; height: 337px; object-fit: none;"

...

/>

This code will produce the following result in the browser:



In situations where the image is larger than the available space, it will appear cropped. In this example image, some parts of the original image on the left, right, top, and bottom do not appear because they cannot fit within the bounds of the declared width and height.

## Using object-fit: scale-down

The scale-down value will either display an image like contain or none depending on which would result in a smaller image.

<img

...

style="width: 300px; height: 337px; object-fit: scale-down;"

...

/>

This code will produce the following result in the browser:



In this example image, the image has been scaled down to behave like contain.

## Using object-fit and object-position

If the resulting image from object-fit appears cropped, by default the image will appear centered. The [object-position](https://developer.mozilla.org/en-US/docs/Web/CSS/object-position) property can be used to change the point of focus.

Consider the object-fit: cover example from before:



Now let’s change the position of the visible part of the image on the X-axis to reveal the right-most edge of the image:

<img

...

style="width: 300px; height: 337px; object-fit: cover; object-position: 100% 0;"

...

/>

This code will produce the following result in the browser:



In this example image, the turtle has been cropped out of the image.

And finally, let’s observe what happens if the position is specified outside of the bounds of the available space:

<img

...

style="width: 300px; height: 337px; object-fit: cover; object-position: -20% 0;"

...

/>

This code will produce the following result in the browser:



In this example image, the turtle and alligator heads have been cropped out of the image. There is also spacing to make up the 20% of offset on the left of the image.

## Conclusion

In this article, you explored the values available for the object-fit and object-position CSS properties.

object-fit also supports [inherit, initial, and unset](https://www.digitalocean.com/community/tutorials/css-inherit-initial-unset).

Before using object-fit in your project, verify that it is supported in the browsers used by your intended audience by checking the [browser support on Can I Use?](https://caniuse.com/object-fit).

If you’d like to learn more about CSS, check out [our CSS topic page](https://www.digitalocean.com/community/tags/css) for exercises and programming projects.

<https://www.digitalocean.com/community/tutorials/css-cropping-images-object-fit>

# How To Create a Parallax Scrolling Effect with Pure CSS in Chrome

[CSS](https://www.digitalocean.com/community/tags/css)

* By Joshua Bemenderfer

Last Validated onAugust 3, 2020 Originally Published onApril 13, 2017 589.1k views

### Introduction

Modern CSS is a powerful tool you can use to create many advanced User Interface (UI) features. In the past, these features relied on JavaScript libraries.

In this guide, you will set up a few CSS lines to create a scrolling parallax effect on a web page. You will use images from [placekitten.com](http://placekitten.com/) as placeholder background images.

You will have a webpage with a pure CSS scrolling parallax effect once you’ve completed the tutorial.

Warning: This article uses experimental CSS properties that do not work across browsers. This project has been tested and works on Chrome.   
This technique doesn’t work well in Firefox, Safari, and iOS due to some of those browsers’ optimizations.

## Step 1 — Creating a New Project

In this step, use the command line to set up a new project folder and files. To start, open your terminal and create a new project folder.

Type the following command to create the project folder:

mkdir css-parallax

In this case, you called the folder css-parallax. Now, change into the css-parallax folder:

cd css-parallax

Next, create an index.html file in your css-parallax folder with the nano command:

nano index.html

You will put all the HTML for the project in this file.

In the next step, you will start creating the structure of the webpage.

## Step 2 — Setting Up the Application Structure

In this step, you will add the HTML needed to create the structure of the project.

Inside your index.html file add the following code:

css-parallax/index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>CSS Scrolling Parallax</title>

</head>

<body></body>

</html>

This is the basic structure of most webpages that use HTML.

Add the following code inside the <body> tag:

css-parallax/index.html

<body>

...

<main>

<section class="section parallax bg1">

<h1>Cute Kitten</h1>

</section>

<section class="section static">

<h1>Boring</h1>

</section>

<section class="section parallax bg2">

<h1>Fluffy Kitten</h1>

</section>

</main>

...

</body>

This code creates three different sections. Two will have a background image, and one will be a static, plain background.

In the next few steps, you will add the styles for each section using the classes you added in the HTML.

## Step 3 — Creating a CSS File and Adding Initial CSS

In this step, you will create a CSS file. Then you will add in the initial CSS needed to style the website and create the parallax effect.

First, create a styles.css file in your css-parallax folder with the nano command:

nano styles.css

This is where you will put all of the CSS needed to create the parallax scrolling effect.

Next, start with the .wrapper class. Inside your styles.css file add the following code:

css-parallax/styles.css

.wrapper {

height: 100vh;

overflow-x: hidden;

overflow-y: auto;

perspective: 2px;

}

The .wrapper class sets the perspective and scroll properties for the whole page.

The height of the wrapper needs to be set to a fixed value for the effect to work. You can use the viewport unit vh set to 100 to get the full height of the screen’s viewport.

When you scale the images, they will add a horizontal scrollbar to the screen, so you can disable it by adding overflow-x: hidden;. The perspective property simulates the distance from the viewport to the pseudo-elements you will create and transform further down in the CSS.

In the next step, you will add more CSS to style your webpage.

## Step 4 — Adding Styles for the .section Class

In this step, you will add styles to the .section class.

Inside your styles.css file add the following code below the wrapper class:

css-parallax/styles.css

.wrapper {

height: 100vh;

overflow-x: hidden;

perspective: 2px;

}

.section {

position: relative;

height: 100vh;

display: flex;

align-items: center;

justify-content: center;

color: white;

text-shadow: 0 0 5px #000;

}

The .section class defines the size, display, and text properties for the main sections.

Set a position of relative so that the child, .parallax::after can be absolutely positioned relative to the parent element .section.

Each section has a view-height(vh) of 100 to take up the viewport’s full height. This value can be changed and set to whatever height you prefer for each section.

Finally, the remainder CSS properties are used to format and add styling to the text inside each section. It positions the text in the center of each section and adds a color of white.

Next, you will add a pseudo-element and style it to create the parallax effect on two of the sections in your HTML.

## Step 5 — Adding Styles for the .parallax Class

In this step, you will add the styles to the .parallax class.

First, you will add a pseudo-element on the .parallax class to be styled.

Note: You can visit [MDN web docs](https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-elements) to learn more about CSS pseudo-elements.

Add the following code below the .section class:

css-parallax/styles.css

...

.section {

position: relative;

height: 100vh;

display: flex;

align-items: center;

justify-content: center;

color: white;

text-shadow: 0 0 5px #000;

}

.parallax::after {

content: " ";

position: absolute;

top: 0;

right: 0;

bottom: 0;

left: 0;

transform: translateZ(-1px) scale(1.5);

background-size: 100%;

z-index: -1;

}

...

The.parallax class adds an ::after pseudo-element to the background image and provides the transforms needed for the parallax effect.

The pseudo-element is the last child of the element with the class .parallax.

The first half of the code displays and positions the psuedo-element. The transform property moves the pseudo-element back away from the camera on the z-index, then scales it back up to fill the viewport.

Because the pseudo-element is further away, it appears to move more slowly.

In the next step, you will add in the background images and static background style.

## Step 6 — Adding the Images and Background For Each Section

In this step, you will add the final CSS properties to add in the background images and background color of the static section.

First, add a solid background color to the .static section with the following code after the .parallax::after class:

css-parallax/styles.css

...

.parallax::after {

content: " ";

position: absolute;

top: 0;

right: 0;

bottom: 0;

left: 0;

transform: translateZ(-1px) scale(1.5);

background-size: 100%;

z-index: -1;

}

.static {

background: red;

}

...

The .static class adds a background to the static section that does not have an image.

The two sections with the .parallax class also have an extra class that is different for each. Use the .bg1 and .bg2 classes to add the Kitten background images.

Add the following code to the .static class:

css-parallax/styles.css

...

.static {

background: red;

}

.bg1::after {

background-image: url('https://placekitten.com/g/900/700');

}

.bg2::after {

background-image: url('https://placekitten.com/g/800/600');

}

...

The .bg1, .bg2 classes add the respective background images for each section.

The images are from the [placekitten](https://placekitten.com/) website. It is a service for getting pictures of kittens for use as placeholders.

Now that all of the code for the parallax scrolling effect is added, you can link to your styles.css file in your index.html.

## Step 7 — Linking styles.css and Opening index.html in Your Browser

In this step, you will link your styles.css file and open up the project in your browser to see the parallax scrolling effect.

First, add the following code to the <head> tag in the index.html file:

css-parallax/index.html

...

<head>

<meta charset="UTF-8" />

<^>

<link rel="stylesheet" href="styles.css" />

<^>

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>CSS Parallax</title>

</head>

...

Now, you can open your index.html file in your browser:



With that, you have set up a functioning webpage with a scrolling effect. Check out [this GitHub repository](https://github.com/do-community/css-parallax-scrolling-chrome/tree/master) to see the full code.

## Conclusion

In this article, you set up a project with an index.html and styles.css file and now have a functional webpage. You added in the structure of your webpage and created styles for the various sections on the site.

It’s possible to put the images you use or the parallax effect further away so that they move more slowly. You’ll have to change the pixel amount on perspective and the transform properties. If you don’t want a background image to scroll at all, use background-attachment: fixed; instead of perspective/translate/scale.

https://www.digitalocean.com/community/tutorials/css-pure-css-parallax