**CSS Transitions**

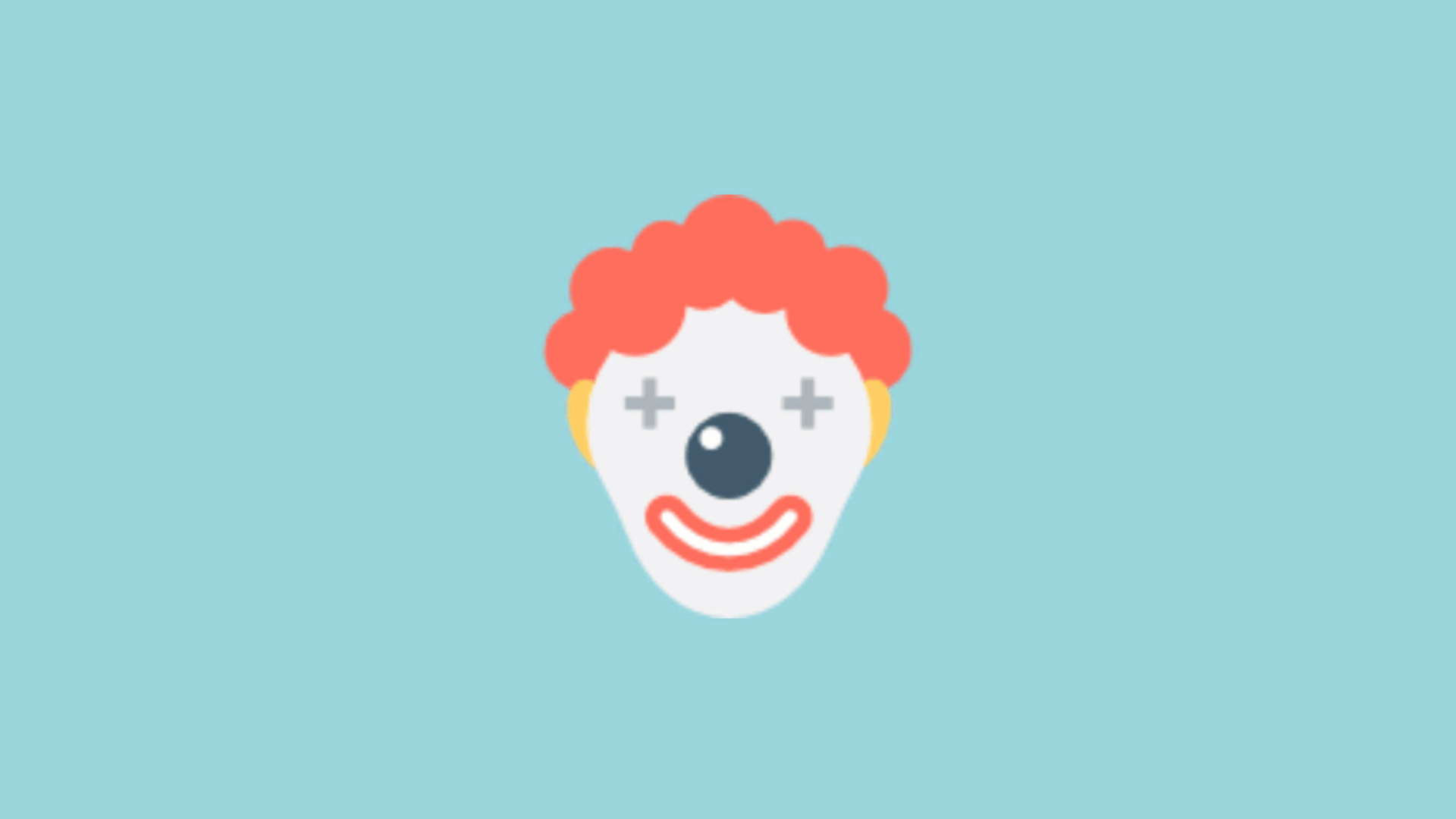
We have previously looked at CSS Animations as one of three ways of bringing animations to life. In this tutorial we are going to look at CSS Transitions.

According to MDN:

"**CSS transitions** provide a way to control animation speed when changing CSS properties. Instead of having property changes take effect immediately, you can cause the changes in a property to take place over a period of time. For example, if you change the color of an element from white to black, usually the change is instantaneous. With CSS transitions enabled, changes occur at time intervals that follow an acceleration curve, all of which can be customized."

CSS Animations, which we looked at previously, are a great tool for animating your page. However, many of the animations you see on your page will be reactions to what the user is doing. These animations are accomplished with CSS Transitions that we will be looking at in this tutorial. So whether it is a link changing colors when you hover over it, text growing when it has focus or a number of user interactions.

To help you get acquainted with CSS Transitions we are going to build the following animation. Yup, it is a super scary clown animation. Notice, that the animation isn't initiated until the mouse hovers over the animation. This isn't like CSS Animations that seemed to start or be initiated upon page load. CSS Transition animations fire based on user interaction:



When you hover over the clown image, notice what happens. The image smoothly scales up and rotates when your mouse cursor is over it. It then smoothly scales and rotates back to its original state when your mouse cursor goes elsewhere. All of this is made possible thanks to the magic of CSS transitions, and in the following sections, we are going to learn the basics of how to use them.

Let's go!

Creating an Abrupt Transition

Just like with CSS animations earlier, the way we are going to learn about CSS transitions is simple. We are going to dive head first and just use them. The first thing we'll need to do is create a new HTML document and add the following things into it ([codepen](http://codepen.io/" \t "_blank) is an easy coding environment to work in):

1. <!DOCTYPE html>
2. <html>
4. <head>
5. <title>CSS Transitions!</title>
7. <style>
8. #container {
9. width: 100%;
10. height: 250px;
11. background-color: #9ad3de;
13. display: flex;
14. align-items: center;
15. justify-content: center;
16. }
17. </style>
18. </head>
20. <body>
21. <div id="container">
22. <img id="hexagon" src="https://github.com/robgmerrill/online-course-images/blob/master/clown.png?raw=true"/>
23. </div>
24. </body>
26. </html>

After you've added all these lines of HTML and CSS, preview this document in your browser. If everything works correctly, you'll see an *almost* recreation of the example you hovered over in the previous section:



This is like our first frightening animation except that our clown doesn't do anything when you hover over it. This is because we haven't added our animation yet.

It is an *almost* recreation because hovering over our smiling clown does nothing. That's because we haven't added the CSS responsible for that yet. Before we do that, let's take a moment to see what we are dealing with. The HTML and CSS you just added has nothing interesting going on, and the main detail you should notice is the markup used for displaying our clown image. It looks as follows:

1. <div id="container">
2. <img id="hexagon" src="https://github.com/robgmerrill/online-course-images/blob/master/clown.png?raw=true"/>
3. </div>

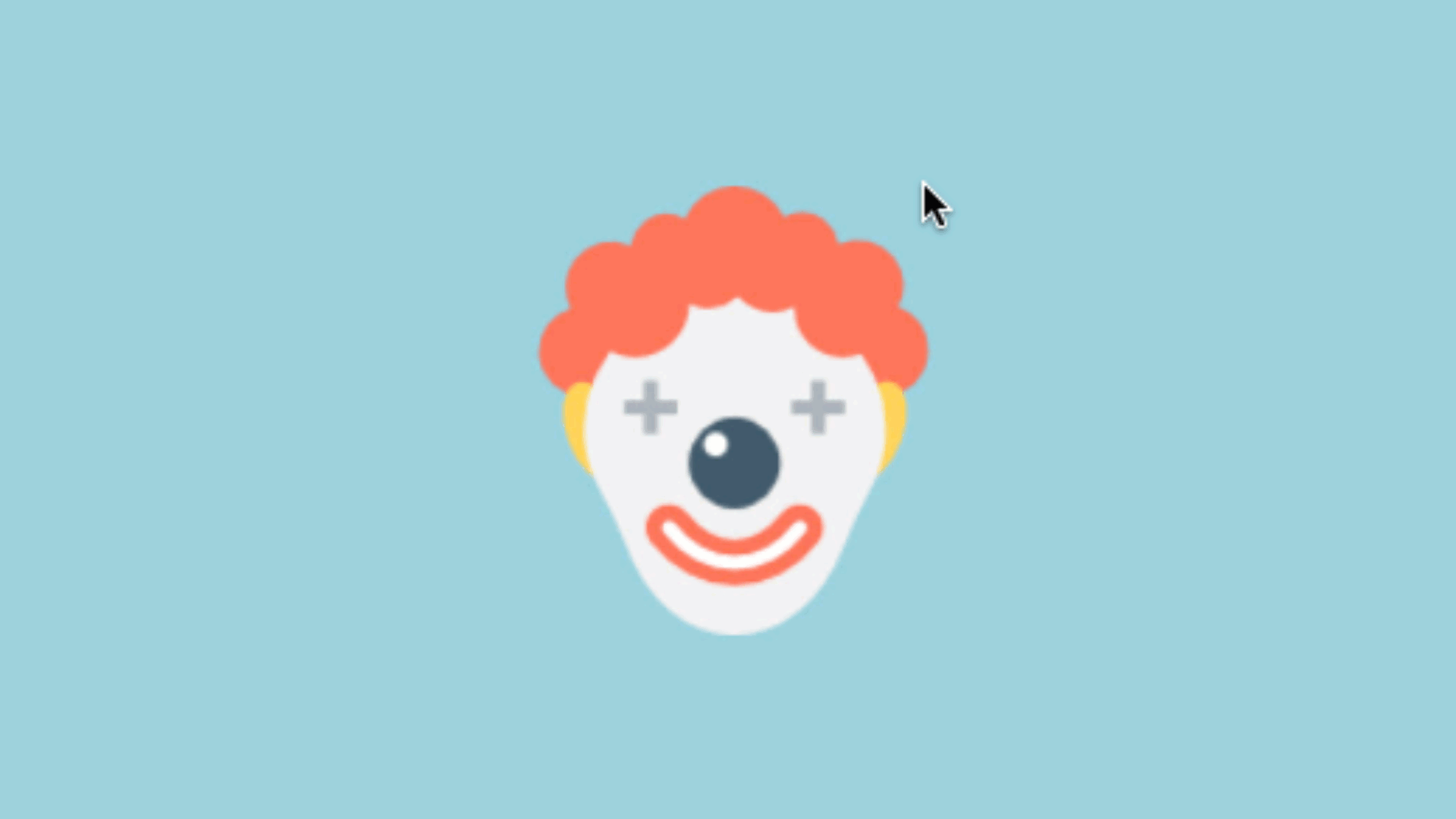
We have a div with the id of container. Within that div we have an img tag and a src attribute pointing to where an image of a clown is hosted.

Now, let's add some CSS to to make our animation animate!

Our animation will make our clown image get larger and rotate when a user hovers over it. To accomplish this we will add the following CSS in our style block:

1. #hexagon:hover {
2. transform: scale3d(1.2, 1.2, 1) rotate(45deg);
3. }

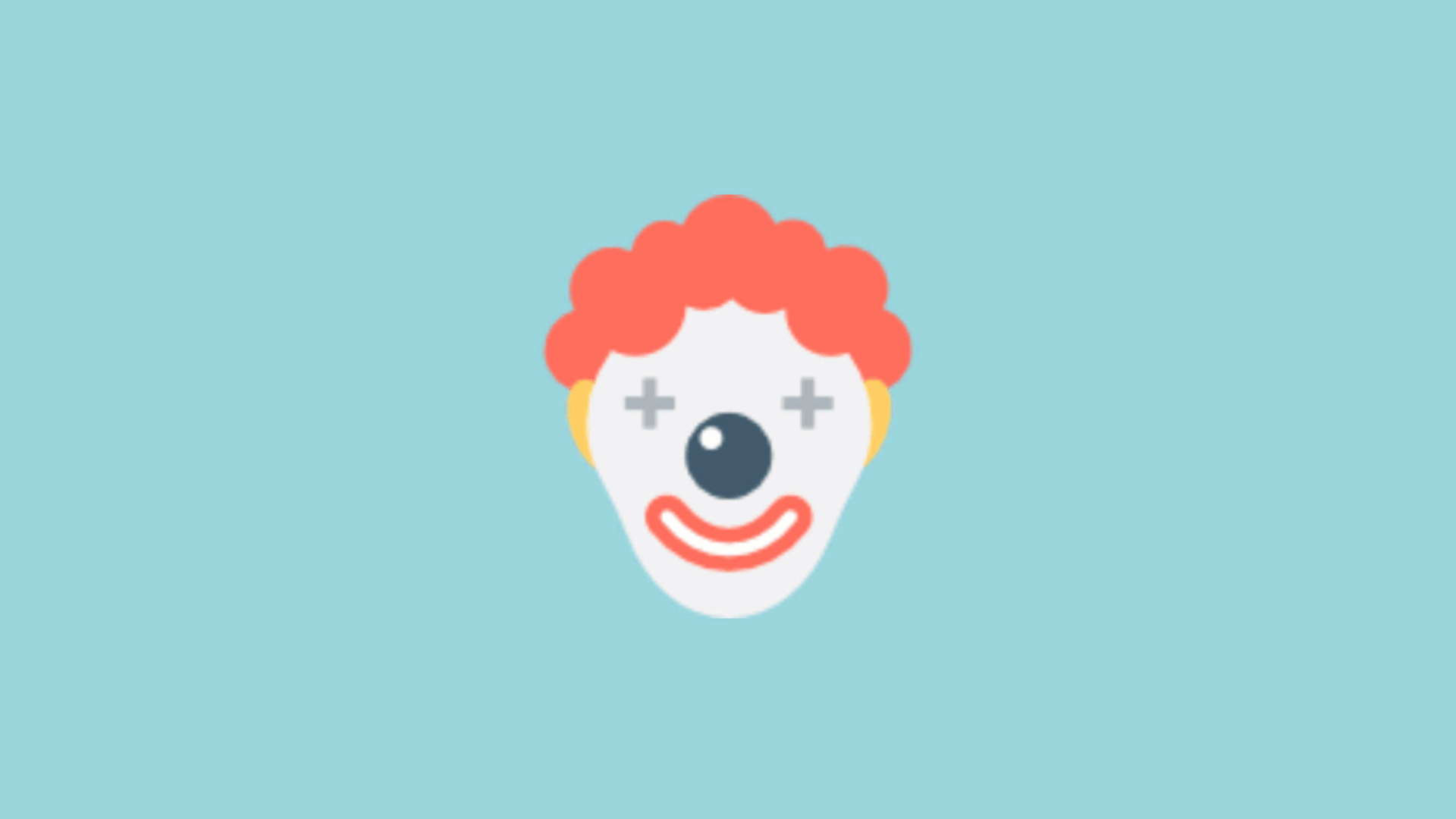
All we are doing is specifying a style rule that activates on hover (thanks to the [:hover](https://developer.mozilla.org/en-US/docs/Web/CSS/:hover) pseudoselector). We set the transform property and call the **scale3d** and **rotate**functions that are responsible for scaling and rotating our hexagon image. Once you've added this CSS, go ahead and preview this page in your browser and hover over the clown image. When you hover over it, you'll see that the image scales and rotates as expected. The only problem is that the scaling and rotating is not smooth and animated. The change is abrupt and jerky!



To fix this, we are going to add a (you guessed it!) transition. Anywhere inside your style block, add the following style rule that contains the transition property:

1. #clown {
2. transition-property: transform;
3. transition-duration: .1s;
4. }

Once you have added this style rule take a look at the results! Much smoother!



Here is What Happened

The secret that made transition work so smoothly is the well named transition property.

The way our transition property does its thing is pretty simple. **It animates property changes**. In order for it to do that, you need to specify just two things:

1. CSS Property: Specifies the name or names of the CSS properties to which transitions should be applied. Only properties listed here are animated during transitions; changes to all other properties occur instantaneously as usual.
2. Duration: Specifies the duration over which transitions should occur. You can specify a single duration that applies to all properties during the transition, or multiple values to allow each property to transition over a different period of time.

You can see how these things map to our transition that we have tucked away inside the #clown style rule:

1. transition: transform .1s;

We have our transition property, it is listening for changes to the transform property, and it runs for **.1** seconds. The result of this transition working can be seen when we hover over the hexagon image and change the transform property's value:

1. #clown:hover {
2. transform: scale3d(1.2, 1.2, 1) rotate(45deg);
3. }

Our **scale3d** value goes from the default (1, 1, 1) to (1.2, 1.2, 1). Feel free to try out other values! Our **rotate** value goes from the default 0deg to 45deg. The CSS transition takes care of figuring out those intermediate, interpolated values to create the smooth animation you see over .1 seconds.

The Shorthand Version

What we've seen so far is the transition sub-properties. To specify a basic transition using just the shorthand version, you can set the transition property as follows:

1. #hexagon {
2. transition: transform .1s;
3. }

What we have just done is learn the basics of how to define a simple CSS transition. Just knowing how to define a CSS transition by specifying the property to listen to and the transition duration will take you pretty far. But, that's not good enough! For creating the kinds of more realistic animations that your UIs deserve and your users expect, there is actually a whole lot more for us to cover. We'll do all of that in subsequent tutorials.

Great job!

Extra Reading

**WHICH PROPERTIES YOU CAN ANIMATE USING CSS ANIMATIONS**

A lot! They are the same you can animate using CSS Transitions, too.

Here’s the full list:

**Property**

background

background-color

background-position

background-size

border

border-color

border-width

border-bottom

border-bottom-color

border-bottom-left-radius

border-bottom-right-radius

border-bottom-width

border-left

border-left-color

border-left-width

border-radius

border-right

border-right-color

border-right-width

border-spacing

border-top

border-top-color

border-top-left-radius

border-top-right-radius

border-top-width

bottom

box-shadow

caret-color

clip

color

column-count

column-gap

column-rule

column-rule-color

column-rule-width

column-width

columns

content

filter

flex

flex-basis

flex-grow

flex-shrink

font

font-size

font-size-adjust

font-stretch

font-weight

grid-area

grid-auto-columns

grid-auto-flow

grid-auto-rows

grid-column-end

grid-column-gap

grid-column-start

grid-column

grid-gap

grid-row-end

grid-row-gap

grid-row-start

grid-row

grid-template-areas

grid-template-columns

grid-template-rows

grid-template

grid

height

left

letter-spacing

line-height

margin

margin-bottom

margin-left

margin-right

margin-top

max-height

max-width

min-height

min-width

opacity

order

outline

outline-color

outline-offset

outline-width

padding

padding-bottom

padding-left

padding-right

padding-top

perspective

perspective-origin

quotes

right

tab-size

text-decoration

text-decoration-color

text-indent

text-shadow

top

transform.

vertical-align

visibility

width

word-spacing

z-index