TRANSITION AND TRANSFORMATIONS WITH CSS

This post will introduce you to CSS transitions and CSS transforms: the CSS power couple. When used together, these properties allow you to create simple animations and add valuable interaction and visual feedback for your users.

Just remember when adding any kind of movement to your project to keep it simple, subtle, and consistent. The movement you create should convey meaning, always enhancing, not distracting from the interaction for your users.

So what are transforms and transitions? At their most basic level, transforms move or change the appearance of an element, while transitions make the element smoothly and gradually change from one state to another.

CSS transitions: an introduction

Let's start with CSS transitions. Transitions are the grease in the wheel of CSS transforms. Without a transition, an element being transformed would change abruptly from one state to another. By applying a transition you can control the change, making it smooth and gradual.

Hover below:

In this post I'll be using transitions in conjunction with transforms. However, transitions can also be used elsewhere where elements change from one style to another (e.g., when a button changes color on hover).

There are two properties that are required in order for the transition to take effect:

1. `transition-property`
2. `transition-duration`

Note: Transition Shorthand

Each transition property can be defined individually, but for cleaner and faster code, it's recommended that you use the transition shorthand.

Here's the full shorthand sequence. Again, the first two properties are required.

```
div {
  transition: [property] [duration] [timing-function] [delay];
}
```

transition-property *(required)*

The `transition-property` specifies the CSS property where the transition will be applied. You may apply a transition to an individual property (e.g., `background-color` or `tranform`) or to all properties in the rule-set (i.e., `all`).

CSS syntax examples for transition-property

```
div {
  transition-property: all;
```

```
 transition-property: transform;
}
```

transition-duration *(required)*

The transition-duration property specifies the time span of the transition. You can specify in seconds or milliseconds.

CSS syntax example for transition-duration

```
 div {
  transition-duration: 3s;
 }
```

Shorthand example for transition-duration

```
div {
 transition: all 3s;
}
```

transition-timing *(optional)*

The transition-timing-function property allows you to define the speed of the transition over the duration. The default timing is ease, which starts out slow, quickly speeds up, and then slows down at the end. The other timing options are: linear, ease, ease-in, ease-out, and ease-in-out.

Here's an example of the different timing options (used with the transform: translate property):

For more advanced timing options, you can define a custom timing function with a cubic-bezier.

CSS syntax example for transition-timing-function

```
div {
 transition-timing-function: ease-in-out;
}
```

Shorthand example for transition-timing-function

```
div {
 transition: all 3s ease-in-out;
}
```

transition-delay *(optional)*

The transition-delay property allows you to specify when the transform will start. By default, the transition starts as soon as it is triggered (e.g., on mouse hover). However, if you want to transition to start after it is triggered you can use the transition delay property.

Shorthand example for transition-delay

```
div {
  transition: all 3s 1s;
}
```

A negative value will start the transition immediately, but part way through the transition process.

CSS transforms: an introduction

Now that we reviewed how to make smooth and gradual transitions, let's look at CSS transforms - how to make an element change from one state to another. With the CSS transform property you can rotate, move, skew, and scale elements. (This post will only cover 2D transforms, but stay tuned for future blog posts on 3D transforms.)

Transforms are triggered when an element changes states, such as on mouse-hover or mouse-click. The examples in this post will demonstrate transforms on mouse-hover.

For simplicity, I'll only be using the unprefixed versions in my examples. However, you may want to include prefixes to ensure it works in modern browsers.

scale

The `scale` value allows you to increase or decrease the size of an element.

For example, the value `2` would transform the size to be 2 times its original size. The value `0.5` would transform the size to be half its original size.

You can scale an element by setting parameters for the width (X-axis) or height (Y-axis). For example, `transform: scaleX(2)`.

Or, use the `scale()` shorthand to scale both axes at the same time: `transform: scale(2);`. Or define them independently of each other: `transform: scale(2, 4);`

CSS syntax example for scale

Don't forget to add a transition! Without applying transition, the element would abruptly change sizes. Add the transition to the parent selector (not the hover selector). To make the transition smooth on both hover-over/hover-off.

```
div {
  transition: transform 1s;
}

div:hover {
  transform: scale(2);
}
```

rotate

With the `rotate` value, the element rotates clockwise or counterclockwise by a specified number of degrees. A positive value, such as `90deg`, rotates the element clockwise, while a negative value, such as `-90deg`, rotates it counterclockwise.

You can rotate more than a full rotation with numbers over than 360, such as `1080deg`, for three full rotations.

CSS syntax example for rotate

```
div {
  transition: transform 1s;
}

div:hover {
  transform: rotate(1080deg);
}
```

translate

The `translate` value moves an element left/right and up/down. The movement is based on the parameters given for the X (horizontal) Y (vertical) axes.

A positive X value moves the element to the right, while a negative X moves the element to the left. A positive Y value moves the element downwards and a negative Y value, upwards.

In this example, the element will move 20 pixels to the right and 20 pixels down.

CSS syntax example for translate

```
div {
  transition: transform 1s;
}

div:hover {
  transform: translate(20px, 20px);
}
```

skew

With the `skew` value, the element skews (or tilts) one direction or the other based on the values given for the X and Y axes.

A positive X value tilts the element left, while a negative X value tilts it right. A positive Y value tilts the element down, and a negative Y value tilts is up. Or use a shorthand to include both X and Y properties:

CSS syntax examples for skew

```
div {
  transform: skewX(25deg);
  transform: skewY(10deg);
  transform: skew(25deg, 10deg);
}
div {
  transition: transform 1s;
}
```

```
div:hover {
  transform: skewX(-20px);
}
```

Note: Skewing an element will also skew all of the children inside of the element as well. If you need to maintain the original angle of a child element, you can use the opposite value of skew to bring it back.

transform-origin

The transform-origin property is separate from the transform property but works in tandem with it. It allows you to specify the location origin of the transform. By default, the origin is in the center of the element.

For example, if you are using the transform: rotate property but want it to rotate not from the center, but from the top left corner, you'd use the value 0% 0% or left top. For the bottom right corner, you would use 0% 100% or right bottom, etc.

Make sure to add the transform-origin property to the parent element, not with the transform property in the hover selector.

```
div {
  transform-origin: left top;
  transition: transform 1s;
}

div:hover {
  transform: rotate(720deg);
}
```

Combining transforms

You can combine multiple transforms by using the transform shorthand or the matrix method.

Shorthand for transform-origin

The transform shorthand allows you to string the various transform methods into one property.

```
div {
  transform: rotate(90deg) scale(2) translateY(-50%) translateX(50%);
}
```