Mastering CSS with Mark Lassoff

Written by Russ Eby and Mark Lassoff

Section 8: CSS Transitions and Animations

In this segment, we're going to be looking at transitions and transformations using CSS. Transformations allow us to change the scale, skew, or rotation of objects within a page.

Let's get started.

The Demo Page

I've created a basic real estate page-- the kind of page you might create advertising a house for sale.

```
<!DOCTYPE html>
<html>

<head>
  <title>Transitions</title>
  <link href="reset.css" rel="stylesheet">
  <link href="https://fonts.googleapis.com/
css?family=Raleway:200" rel="stylesheet">
  <style>
    #container {
      font-family: 'Raleway', sans-serif;
      background-color: #fafafa;
    }
```

```
header,
#thumbs {
   background-color: beige;
   width: 96%;
   margin: 0 auto;
   box-shadow: 2px 2px #eee;
   text-align: center;
}

header p {
   color: red;
   padding-top: 10px;
   margin-bottom: 10px;
}

header h1 {
   font-size: 3em;
}

header h2 {
   font-size: 2em;
   padding-bottom: 10px
}

#feature {
   text-align: center;
}

#feature img {
   width: 70%;
   margin-top: 20px;
   margin-bottom: 10px
}

#thumbs img {
   width: 20%;
   margin-right: 20px;
   margin-left: 20px;
   margin-bottom: 10px;
```

```
        margin-top: 10px;
        opacity: 0.35;
        box-shadow: 2px 2px #ccc;

      }

      #thumbs img:hover {
        opacity: 1.0;
        transition: all 1s;
        transition-delay: 1s;
        transform: rotate(5deg) scale(1.15, 1.15);
      }
    </style>
    <script>
      window.onload = function () {
        const images = document.getElementsByTagName("img");
        for (var x = 1; x < images.length; x++) {
          images[x].addEventListener('click', function () {
            let loc = this.src;
            document.getElementById('featureImage').src = loc;
          });
        }
      }
    </script>
</head>

<body>
  <div id="container">
    <header>
      <!-- Property location information-->
      <p>For Sale</p>
      <h1>129 Maple Ave South</h1>
      <h2>Westport, CT 06880</h2>
      <h2>$699,000</h2>
    </header>
    <div id="feature">
      <img id="featureImage" src="http://placekitten.
com/250/250">
    </div>
```

```
    <div id="thumbs">
      <img src="" alt="">
      <img src="http://placekitten.com/250/250">
      <img src="http://placekitten.com/240/240">
      <img src="http://placekitten.com/250/250">
      <img src="http://placekitten.com/250/250">
      <img src="http://placekitten.com/250/250">
      <img src="http://placekitten.com/250/250">
      <img src="http://placekitten.com/250/250">
      <img src="http://placekitten.com/250/250">
      <img src="http://placekitten.com/250/250">
      <img src="http://placekitten.com/250/250">
      <img src="http://placekitten.com/250/250">
    </div>
  </div>
</body>

</html>
```

Aside from the text, the page is comprised of the main photo and an array of photos underneath. If you point the mouse at an individual picture, it gets more prominent, and then it fades in from translucent to fully visible. If you click on one of the photos in the array, the first picture will be replaced with that photo from the array. (This isn't actually CSS, I used a little JavaScript which I'll show explain).

Tour of the HTML

Let's take a look first at the HTML that's used to put this page together. The HTML is very straight forward.

Containing everything on the page is the #container. It's advisable to use a `<div>` element to include other elements on the page as opposed to using the `<body>` tag.

```
<div id="container">
   ...All Elements
<div>
```

At the top of the page is the `<header>` element which has the "For Sale" text along with the address and the price.

```
<div id="container">
   <header>
      <p>For Sale</p>
      <h1>129 Maple Ave South</h1>
      <h2>Westport, CT 06880</h2>
      <h2>$699,000</h2>
   </header>

   ...All Elements
<div>
```

Next is the page's feature section that has the initial image focal point. When the page loads, it's already displaying image 1.jpg from the set. I just named the image files 1.jpg through 11 .jpg. While not, particularly creative, it does make it easier to iterate through the collection of images.

```
<div id="container">
   ... Header Section

   <div id="feature">
      <img id="featureImage" src="homepics/1.jpg">
   </div>

   ...All Elements
<div>
```

Please note: You should use the `alt` attribute with `img` tags to include some descriptive text for screen readers. Without an `alt` attribute, a screen reader will read the `src` attribute, which won't be as useful.

All of the images are at the bottom are located in a `div` called thumbs, which is short for thumbnails.

```
<div id="container">
   ... Header Section
   ... Feature Image Section

   <div id="thumbs">
      <img src="homepics/2.jpg">
      <img src="homepics/3.jpg">
      <img src="homepics/4.jpg">
      <img src="homepics/5.jpg">
      <img src="homepics/6.jpg">
      <img src="homepics/7.jpg">
      <img src="homepics/8.jpg">
      <img src="homepics/9.jpg">
      <img src="homepics/10.jpg">
      <img src="homepics/11.jpg">
      <img src="homepics/1.jpg">
   </div>
<div>
```

Now we've examined all of the HTML on the page. Easy, right>

CSS

Now we'll take a look at the CSS used to style this page and the transitions and transformations used.

Reset CSS

And as I explained in a previous section. We're using a reset CSS here. The reset CSS sets all CSS options and measurements to zero, so we don't have to worry about the settings of the default stylesheet. The CSS file can be downloaded at Meyer Web.

```
<link href="reset.css" rel="stylesheet">
```

Google Font

I choose fonts from the Google Fonts Web site. Raleway, which is a thin, elegant font, is used as our primary typeface. Since the home advertised is a luxury property, this is fitting.

```
<link href="https://fonts.googleapis.com/
css?family=Raleway:200" rel="stylesheet">
```

CSS

In the `#container` div we'll set some defaults for the entire document.

```
#container {
   font-family: 'Raleway', sans-serif;
   background-color: #fafafa;
}
```

The `<header>` section and the thumbnails have several styles in common, so a compound selector was used so the styles could be applied to both selectors.

```
header, #thumbs {
   background-color: beige;
   width: 96%;
   margin: 0 auto;
   box-shadow: 2px 2px #eee;
   text-align: center;
}
```

Note: `margin: 0 auto`; is used to center all of the content within an element. This is a bit different than using the rule: `text-align: center`.

`margin: 0 auto`; will center content within another `div`. If you have a `div` within a `div`, the interior `div` will be centered within the parent.

Next, we style the `p` element in the header.

```
header p {
   color: red;
   padding-top: 10px;
   margin-bottom: 10px;
}
```

Next, the `h1` and `h2` in the header are styled, which contain the address and price content.

```
header h1 {
   font-size: 3em;
}

header h2 {
   font-size: 2em;
   padding-bottom: 10px
}
```

Within the feature section:

```
#feature{
   text-align: center;
}
```

The feature section img is next.

```
#feature img{
   width: 70%;
   margin-top: 20px;
   margin-bottom: 10px
}
```

Next, we'll style the actual image thumbnails. Note the use of the opacity rule here.

```
#thumbs img{
   width: 20%;
   margin-right: 20px;
   margin-left: 20px;
   margin-bottom: 10px;
   margin-top: 10px;
   opacity: 0.35;
   box-shadow: 2px 2px #ccc;
}
```

Margin Shortcut: instead of calling out each margin side separately we use a shorthand to address each side with a single CSS rule. The generic form is `margin: [top] [right] [bottom] [left];`. In actual use, the rule would be formatted like this: `margin: 10px 20px 10px 20px`

There is also a second short cut s a second short cut style: `margin: [top&bottom] [left&right];` which, in actual use, would look like this: `margin: 10px 20px;` . All three methods style the content exact same way. It's completely up to you which method you want to use.

I mentioned that the set of CSS rules above included an `opacity` rule. For the `opacity` value, a value of `1.0` is entirely opaque; you can't see through it. A value of `0` is altogether transparent; meaning you can't see it at all. Any value between zero and one makes the element translucent.

It is in the `opacity` rule that we see our first CSS transition. The transition occurs when a user hovers their mouse above a thumbnail.

```
#thumbs img:hover {
```

When the hover occurs, we bring the opacity of the element up to 1.0, or fully opaque.

```
    opacity: 1.0;
```

When you view the effect in your web browser, you'll notice there is a transition. We're transitioning the opacity difference in the CSS rules over a one second period:

```
    transition: opacity 1s;
```

And we're going to delay the effect for one full second before it begins:

```
    transition-delay: 1s;
```

There are several transformations we could execute on the elements. Keep in mind that they often don't work well together or work well if there is a click event on the object.

The following CSS transformation rotates an image 5 degrees clockwise.

> Negative numbers would result in counter-clockwise rotation in most browsers.

```
transform: rotate(5deg);
```

The following rule you can observe on our page. Each image thumbnail grows by 15% when you mouse over:

```
  transform: scale(1.15, 1.15);
}
```

> The generic form of the `scale` rule is: `scale([width], [height])`. If you used `scale(2,2)` the elements' width would grow by 200% and height would grow by 200%.

The CSS transformations available are:

- translate
- scale
- rotate
- skew
- scale

These transitions and transformations are designed to be effects, best if they are used subtlety. When overused and/or gratuitous, they can become bothersome for the user.

JavaScript

Let's take a quick peek at the JavaScript we are using to switch out the photo. (If you're not familiar with or interested in JavaScript, feel free to gloss over this section.)

The `window.onload` function is called after the entire document is loaded and the screen is drawn.

```
window.onload = function () {
```

First, we're creating an array of images on the page. Using `getElementByTagName` we'll grab all the `img` elements. and put them in the array called images.

> Yes, it does grab the #featureImage as well, but it'll just put the same image back in so not an issue for this example.

```
const images = document.getElementByTagName("img");
```

We're then going to go through the array of images with a for loop.

```
for (var x = 1; x < images.length; x++) {
```

And we are going to add to each image element an event listener listening for a click.

```
    images[x].addEventListener('click',
```

When an image is clicked, the click listener responds with this anonymous function

```
    function () {
      let loc = this.src;
```

Then we set the src of the featureImg to loc.

```
        document.getElementById('featureImg').src = loc;
    });
  }
}
```

And there you go, a couple of transformations, a couple of transitions and bonus JavaScript. I hope you enjoyed this section.

Submit this

This assignment is really about playing around with these transforms and transitions. Change the numbers up. Put crazy numbers in place. The best way to make art is to put the paint on the canvas in new and unexpected ways. Above all, have fun with this.

1. We weren't able to see what rotate does, so for the first part of the assignment is to create an HTML document with a photo on it and have the image rotate when you hover over it.

2. Try out skew. Using the same HTML document and add a second image and see exactly what happens when you skew the image. Skew takes one or two arguments. skew(xValue) or skew(xValue, yValue). Enter in the degrees you want it to skew. For example `skew(15deg, 15deg).`

Some Questions you can answer by trying:

- Can you skew text? Try adding an `h1` to the top of this document and try to skew it.
- What if you skew an element without hover? Will it start skewed or never be affected?

3. Another popular transform is translate. `transform: translate(xValue, yValue);` This moves an element sideways up or down. Add another image and add a hover to have the image move across the screen. What happens if you don't add a transition time?