CSS M7-8 Notes

CSS3 transitions allow us to change from one property value to another over a given duration.
transition-property - specifies the property to be transitioned
transition-duration - specifies the duration over which transitions should occur
transition-timing-function - specifies how the pace of the transition changes over its duration
transition-delay - specifies a delay (in seconds) for the transition effect
In the example below, we set the transition property to transform, with a duration of 5 seconds, and with an ease-in timing function that specifies a transition effect with a slow start.
transition: transform 5s ease-in;
In the example below, the div element has width and height of 50px, with a green background. We specified a transition effect for the width property, with a duration of 3 seconds:
The CSS will look like this:

```
div {
  width: 50px;
  height: 50px;
  background: #32CD32;
  transition: width 3s;
}
div:hover {
  width: 250px;
}
```

The transition-timing-function property specifies the speed curve of the transition effect.
It can have the following values:
ease - the animation starts slowly, then accelerates quickly.
ease-in - starts slowly, then accelerates, and stops abruptly.
ease-out - starts quickly, but decelerates to a stop.
ease-in-out - similar to ease, but with more subtle acceleration and deceleration.
linear - constant speed throughout the animation; often best for color or opacity changes.
Finally, we have cubic-bezier(), which allows you to define your own values in the cubic-bezier function. Possible values are numeric values from 0 to 1.
transition-timing-function: cubic-bezier(0,0,1,1);
CSS3 transforms allow you to translate, rotate, scale, and skew elements.

A transformation is an effect that lets an element change shape, size, and position. CSS3 supports 2D and 3D transformations. Let's take a look at the rotate transformation:

```
div {
  width: 200px;
  height: 100px;
  margin-top: 30px;
  background-color: #32CD32;
}
```

As previously mentioned, using a positive value will rotate an element clockwise, and using a negative value will rotate the element counter-clockwise.

```
div.positive {
  width: 200px;
  height: 100px;
  margin-top: 30px;
  background-color: #32CD32;
  transform: rotate(10deg);
}
div.negative {
  width: 200px;
  height: 100px;
  margin-top: 30px;
  background-color: #32CD32;
  transform: rotate(-10deg);
}
```

The transform-origin property allows you to change the position of transformed elements. The default value for the property is 50% 50%, which corresponds to the center of the element.

In the example below, we use the transform-origin property together with transform-rotate. The origin of the x-axis (horizontal) is set to 25% from the left. The origin for the y-axis (vertical) is set to 75% from above.

The CSS:

```
div.empty-div {
  position: relative;
  height: 100px;
  width: 100px;
  margin: 30px;
  padding: 10px;
```

```
  border: 1px solid black;
}
div.green-div {
  padding: 50px;
  position: absolute;
  background-color: #8bc34a;
  border: 1px solid white;
  transform: rotate(15deg);
  transform-origin: 25% 75%;
}
```

The translate() method moves an element from its current position (according to the parameters given for the x-axis and the y-axis). Positive values will push an element down and to the right of its default position, while negative values will pull an element up and to the left of its default position.

In this example below, the div element is moved 100px to the right and 50px down:

```
div {
  padding: 50px;
  position: absolute;
  background-color: #32CD32;
  transform:translate(100px, 50px);
}
```

The skew() method skews an element along the x-axis and the y-axis by the given angles.

The following example skews the element by 30 degrees along the X-axis:

```
transform: skew(30deg);
```

The scale() method increases or decreases the size of an element, according to the parameters given for the width and height. 1 stands for the original size, 2 for twice the original size, and so on.

In the example below, we decreased the first div by the factor 0.7 both horizontally and vertically, and increased the second div by a factor of 1.5 horizontally and vertically.

```
div.first {
  width: 200px;
  height: 100px;
  background-color: #8BC34A;
  transform: scale(0.7, 0.7);
  color:white;
}
```

```
div.second {
  margin: 60px;
  width: 200px;
  height: 100px;
  background-color: #8bc34a;
  transform: scale(1.5,1.5);
  color:white;
}
```