Transitions & Transforms & Animations

What do we use motion for?

- Drawing a user's eyes towards something that they should know is changing
- Easing someone in from one state to another
- Adding some fun and enjoyment to using our app

Transitions

The most common case for wanting motion is to transition a CSS property from one set of values to another. We can do that by telling our element which properties should transition, and for how long:

```
.my-element {
    opacity: 0.8;
    transition: opacity 300ms ease;
}
.my-element:hover {
    opacity: 1;
}
```

Transitions (pt2)

```
.my-element {
    transition: opacity 300ms ease;
}
```

Is the same as:

```
.my-element {
    transition-property: opacity;
    transition-duration: 300ms;
    transition-timing-function: ease;
    transition-delay: 0ms;
}
```

Transition timing functions

Multiple transitions

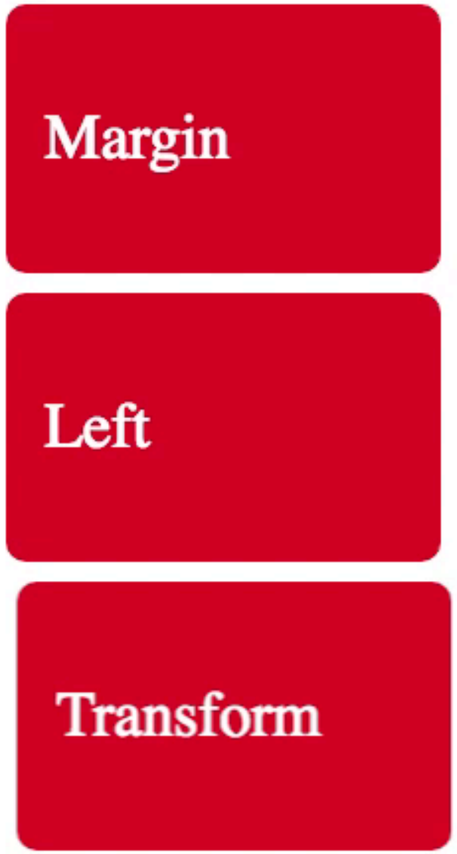
You can comma separate any of the properties, and they'll align with their order:

```
/* Two totally separate transitions */
.my-element {
    transition: opacity 100ms linear, color 100ms linear 100ms;
}

/* Share everything except the delay */
.other-element {
    transition-property: opacity, color;
    transition-duration: 100ms;
    transition-timing-function: linear;
    transition-delay: 0ms, 100ms;
}
```

Transitions on Position

- One of the most common transitions we'll want to do is on the position of something
- We position elements using many properties: padding, margin, height, width, top, left, etc.
- However, we should NEVER transition on these properties, as it's terrible for performance
- Position is being calculated for nearby elements *every frame*



(Codepen)

Transforms

- Fortunately we have transforms, a way of manipulating the visuals of an element without recalculating position
- Move elements on the X / Y / Z axis, scale them larger or smaller, rotate them, and much much more
- Extremely performant, as it uses the GPU and has to calculate much less

Transforms (Example)


```
.element {
    transform: translateX(100px) rotate(360deg) scale(1.2);
    transition: transform 2000ms ease;
}
```

Animations

- Sometimes we want to perform multiple non-linear operations over a set period of time
- We define animations in "keyframes," the steps and order in which they happen
- We then assign them to elements and define how long they should run, and how many times
- These animations trigger either when the element is created, or when a class is added

Animations Example

```
@keyframes four-corners {
  0%, 100% { /* 0s and 2s in */
    transform: translate(0px, 0px);
    background-color: #1abc9c;
  }
  25% { /* .5s in */
    transform: translate(100px, 0px);
    background-color: #3498db;
  }
  50% { /* 1s in */
    transform: translate(100px, 100px);
    background-color: #f1c40f;
  }
  75% { /* 1.5s in */
    transform: translate(0px, 100px);
    background-color: #e74c3c;
  }
}

.block {
  animation: four-corners 2000ms ease 1 500ms;
}
```

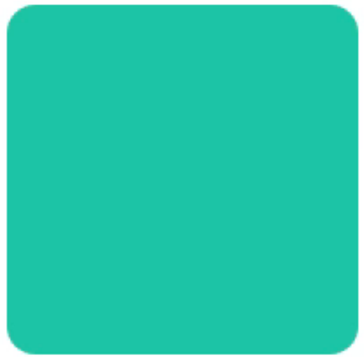
Animations Example (pt2)

```
.block {
  animation: four-corners 2000ms ease 1 500ms;
}
```

Is the same as

```
.block {
    animation-name: four-corners;
    animation-duration: 2000ms;
    animation-timing-function: ease;
    animation-iteration-count: 1; /* can be `infinite` */
    animation-delay: 500ms;
    animation-direction: normal; /* can be `reverse`, `alternate` */
}
```

Animations (Example)



(Codepen)

Resources

- Transitions and Animations - Very thorough overview
- Animista - Library of CSS animations
- Ceaser - Generate custom timing functions graphically