THE MAGIC OF

CHAPTER 6

Transitions

“All you need to know about CSS Transitions”

Before reading this chapter, please read All you need to know about CSS Transitions by Alex MacCaw. This is a fantastic resource and covers much of what we’d like to cover.

Transitions

CSS transitions are the ideal solution for transitioning a CSS property (or set of properties) from one value to another value along some easing path.

Now, together, let’s combine this knowledge with what we’ve learned in the previous chapters on Color and Typography.

Examples

Example 1: Random individual letter fade

One extremely powerful tool at your disposal is `transition-delay`. Transition delays allow you to delay the start of when a transition occurs. In this first example, we set pseudo-random delays for the `opacity` transitions on each letter, creating a reveal effect that is both subtle and attractive.

```
.ex span {
    opacity: 0;
    transition: opacity 1300ms
}

.ex span:nth-child(1)  { transition-delay:  200ms }
.ex span:nth-child(2)  { transition-delay: 1200ms }
.ex span:nth-child(3)  { transition-delay:  800ms }
.ex span:nth-child(4)  { transition-delay:  300ms }
.ex span:nth-child(5)  { transition-delay:  700ms }
.ex span:nth-child(6)  { transition-delay:  600ms }
.ex span:nth-child(7)  { transition-delay:  400ms }
.ex span:nth-child(8)  { transition-delay:  900ms }
.ex span:nth-child(9)  { transition-delay:  700ms }
.ex span:nth-child(10) { transition-delay:   50ms }

.ex:hover span {
    opacity: 1
}
```

HOVER TO SEE TRANSITION

Example 2: Multiple transitions and delays

CSS transitions really shine when they're combined. In this example, we specify two transitions, one for `-webkit-transform` and one for `opacity`. They are written as part of the same `transition` declaration, separated with commas.

```
.ex .title span,
.ex .author span {
    /* ... */
    transition: -webkit-transform 800ms, opacity 800ms
}
```

Since we specified two transitions, we also must specify two transition delays. They are written similarly.

```
.ex .title span:nth-child(1) { transition-delay: 360ms, 300ms }
.ex .title span:nth-child(2) { transition-delay: 420ms, 300ms }
.ex .title span:nth-child(3) { transition-delay: 480ms, 300ms }
/* ... */
```

Together, these transitions and their delays allow us to create a beautiful wave effect with the letters in this example.

```
.ex .title span,
.ex .author span {
    display: inline-block;
    opacity: 0;
    transition: -webkit-transform 800ms, opacity 800ms
}

.ex .title span {
    -webkit-transform: translateZ(0) translateY(-6rem)
}

.ex .author span {
    -webkit-transform: translateZ(0) translateY(6rem)
}

.ex:hover .title span,
.ex:hover .author span {
    opacity: 1;
    transition: -webkit-transform 800ms, opacity 1200ms;
    -webkit-transform: translateZ(0) translateY(0)
}

.ex .title span:nth-child(1) { transition-delay: 360ms, 300ms }
.ex .title span:nth-child(2) { transition-delay: 420ms, 300ms }
.ex .title span:nth-child(3) { transition-delay: 480ms, 300ms }
.ex .title span:nth-child(4) { transition-delay: 540ms, 300ms }
.ex .title span:nth-child(5) { transition-delay: 600ms, 300ms }
```

```
.ex .title span:nth-child(6) { transition-delay: 660ms, 300ms }
.ex .title span:nth-child(7) { transition-delay: 720ms, 300ms }

.ex .author span:nth-child(1)  { transition-delay: 420ms, 0ms }
.ex .author span:nth-child(2)  { transition-delay: 390ms, 0ms }
.ex .author span:nth-child(3)  { transition-delay: 360ms, 0ms }
.ex .author span:nth-child(4)  { transition-delay: 330ms, 0ms }
.ex .author span:nth-child(5)  { transition-delay: 300ms, 0ms }
.ex .author span:nth-child(6)  { transition-delay: 270ms, 0ms }
.ex .author span:nth-child(7)  { transition-delay: 240ms, 0ms }
.ex .author span:nth-child(8)  { transition-delay: 210ms, 0ms }
.ex .author span:nth-child(9)  { transition-delay: 180ms, 0ms }
.ex .author span:nth-child(10) { transition-delay: 150ms, 0ms }
.ex .author span:nth-child(11) { transition-delay: 120ms, 0ms }
.ex .author span:nth-child(12) { transition-delay:  90ms, 0ms }
.ex .author span:nth-child(13) { transition-delay:  60ms, 0ms }
.ex .author span:nth-child(14) { transition-delay:  30ms, 0ms }

.ex:hover .title span,
.ex:hover .author span {
    transition-delay: 0
}
```

Example 3D

And just in case you were wondering: transitions work on 3D transforms too.

```
.ex .letter {
    -webkit-perspective: 20rem
}

.ex .front,
.ex .back {
```

```
    -webkit-backface-visibility: hidden;
    transition: -webkit-transform 800ms
}

.ex .back {
    -webkit-transform: translateZ(0) rotateY(-180deg)
}

.ex:hover .back {
    -webkit-transform: translateZ(0) rotateY(0deg)
}

.ex:hover .front {
    -webkit-transform: translateZ(0) rotateY(180deg)
}

.ex .letter:nth-child(1) span { transition-delay:  200ms }
.ex .letter:nth-child(2) span { transition-delay:  400ms }
.ex .letter:nth-child(3) span { transition-delay:  600ms }
.ex .letter:nth-child(4) span { transition-delay:  800ms }
.ex .letter:nth-child(5) span { transition-delay: 1000ms }
```

Of course, this is just the tip of the iceberg. Colors, gradients, sizes, positions, orientations, etc., can all be transitioned simultaneously. In addition you have delays and custom easing functions right at your fingertips. CSS transitions are so easy to work with. Trust that the browser will do the right thing when you transition a CSS property, because it usually does.

MAGIC

Radio button accordion

Combining HTML state with CSS transitions can make for rich interactions. Here we use the `:checked` pseudo-selector on radio buttons to style elements which follow them. This technique is often referred to as “The Checkbox Hack”, but it could work with just about any element which can hold state via some pseudo-selector (`:checked` , `:focus` , etc.).

There’s a lot of subtlety in this example, but the main idea is this: each accordion baffle has a `label` whose `for` attribute matches the `id` of a radio button. So, three baffles, three radio buttons. These radio buttons are placed in the document just before the baffle contents, so that we can use the adjacent selector `+` to style differently baffles which appear after `:checked` radios.

```
.accordion .baffle {
    height: 0
}

.accordion input[type="radio"]:checked + .baffle {
    height: 10em
}
```

Clicking the baffle label checks the radio button (and unchecks whichever radio button was previously checked) triggering the opening/closing of the appropriate baffle.

And voilà. An accordion with only HTML and CSS.

ACCORDION

> Accordions (from 19th century German Akkordion, from Akkord - “musical chord, concord of sounds”) are a family of box-shaped musical instruments of the bellows-driven free-reed aerophone type, colloquially referred to as a squeezebox. A person who plays the accordion is called an accordionist. The concertina and bandoneón are related; the harmonium and American reed organ are in the same family.

CONSTRUCTION

HISTORY

Transition Inspiration

If you're looking for more inspiration on what is possible with CSS transitions, check out these amazing demos from Codrops:

- Sidebar Transitions
- Page Transitions
- Header Effects
- Creative Link Effects
- Simple Stack Effects
- Animations for Thumbnail Grids
- Loading Effects for Grid Items

jQuery.Transit

For those of you who like a little JS to spice up your CSS, check out rstacruz's jquery.transit. For simple, static transitions (*oxymoron?*), it's overkill. But you may find it really useful if you're doing transitions which have dynamic or user-generated CSS property values.

Further reading

- Can I Use: CSS Transitions
- MDN: Using CSS transitions

Chapter 5