CS7026 – CSS3

CSS3 – Transitions & Animations
Fading in the Colour Using CSS Transitions

- Transitions are essentially the simplest type of CSS animation.

- They let you ease the change between two different styles of an element gradually and smoothly, instead of seeing an immediate and abrupt difference in style when an element is hovered, targeted, or has its state otherwise changed.
Let’s have a look at either fading in or fading out a background colour; the “movement” created by the gradually changing colour might direct the viewer’s attention even more effectively than the abrupt change.

We can do this using CSS3 transitions and animations instead of JavaScript.
Animating the Change with Pure CSS

- Transitions can be used here to fade in the colour when the heading is hovered over, and then fade it out again when the hover is removed.

- Animations can be used to either fade in or fade out the colour - or both, in succession - when the heading is targeted.

- Let’s look at both options.
You apply a transition by:

- telling the browser which property you want to gradually change (using `transition-property`)
- how long the change should take in seconds (using `transition-duration`).
- You can optionally add a delay to the start of the transition (`transition-delay`)
- and vary the speed of change over the duration of the transition (`transition-timing-function`).

All of these properties can also be combined into the shorthand `transition` property.
Fading in the Colour Using CSS Transitions

- Add it, and the 3 browser-specific equivalents, to the h2 rule:

```css
h2 {
  clear: left;
  margin: 0 0 -.14em 0;
  color: #414141;
  font-family: Prelude, Helvetica, "Helvetica Neue", Arial, sans-serif;
  font-size: 2.17em;
  font-weight: bold;
  -moz-transition: background-color 3s ease-out;
  -o-transition: background-color 3s ease-out;
  -webkit-transition: background-color 3s ease-out;
  transition: background-color 3s ease-out;
}
```
Fading in the Colour Using CSS Transitions

- Then add a new rule:

```css
h2:hover {
    background-color: hsla(203,78%,36%,.2);
}
```
Fading in the Colour Using CSS Transitions

Here, we’ve told the browsers that any time the `background-color` value of an `h2` element changes, we want it to make that change happen gradually over the course of three seconds.

We’ve also specified a `transition-timing-function` of ease-out so that the animation will slow down slightly at the end.
What Can You Transition?

- Not all properties can be transitioned. The W3C calls those that can “animatable properties” and lists them at [www.w3.org/TR/css3-transitions/#animatable-properties](http://www.w3.org/TR/css3-transitions/#animatable-properties).

- That’s why we use `background-color` in the `h2 :target` rule instead of the `background` shorthand property; `background-color` can be transitioned and `background` can’t.

- (Or at least, it shouldn’t be able to be transitioned; some browsers do allow it.)
Fading in the Colour Using CSS Transitions

- In addition to the `background-color` transition, we can make the left padding added to the highlighted headings transition too, to create the appearance that the text is sliding to the right.

- You can do this by simply writing the padding transition in the same transition property, separated by a comma.
Fading in the Colour Using CSS Transitions

h2 {
  clear: left;
  margin: 0 0 -.14em 0;
  color: #414141;
  font-family: Prelude, Helvetica, “Helvetica Neue”, Arial, sans-serif;
  font-size: 2.17em;
  font-weight: bold;
  -moz-transition: background-color 1s ease-out, padding-left 5s ease-out;
  -o-transition: background-color 1s ease-out, padding-left 5s ease-out;
  -webkit-transition: background-color 1s ease-out, padding-left 5s ease-out;
  transition: background-color 1s ease-out, padding-left 5s ease-out;
}
And add the following to your new h2 : hover rule:

```css
hover{
    padding-left: 10px;
    background-color: hsla(203, 78%, 36%, .2);
    text-shadow: 1px 1px 2px #fff;
}
```
More On the **transition** Property

- The transition property is part of the Transitions module found at [www.w3.org/TR/css3-transitions](http://www.w3.org/TR/css3-transitions).

- It’s shorthand for the transition-property, transition-duration, transition-timing-function, and transition-delay properties.

- It allows you to gradually change from one value of a property to another upon an element’s state change.

- The necessary pieces of a transition are
  - transition-property (to specify the property of the element you want to gradually change) and
  - transition-duration (to specify over how long the change should take—the default is zero if you leave it off).

  The other properties are optional.
More On the *transition* Property

- Multiple properties of an element can be transitioned simultaneously; write each property’s transition in the same transition property, separated by commas.

- You can also use a value of *all* for transition-property to specify that all of the element’s properties should transition.

- Not all properties can be transitioned; see [www.w3.org/TR/css3-transitions/#animatable-properties](http://www.w3.org/TR/css3-transitions/#animatable-properties) for a list of those that can.
More On the `transition` Property

- All of the transition-supporting browsers support transitioning most of these properties; there are various exceptions.

More On the transition Property

- Other than fading in a background colour change, you might want to use transitions for:
  - Making images appear to light up or brighten when hovered (by transitioning opacity)
  - Fading between images that are stacked on top of each other (such as a black and white version that gets swapped with a coloured version, or before and after images); see [http://trentwalton.com/2010/03/30/css3-transition-delay](http://trentwalton.com/2010/03/30/css3-transition-delay) and [http://css3.bradshawenterprises.com](http://css3.bradshawenterprises.com)
More On the **transition** Property

- Gradually making tooltips or informational boxes appear; see [www.zurb.com/playground/drop-in-modals](http://www.zurb.com/playground/drop-in-modals)

- Creating the appearance that something is growing or shrinking (by transitioning its width, height, or transform scale value); see [www.zurb.com/playground/css3-polaroids](http://www.zurb.com/playground/css3-polaroids) and [www.marcofolio.net/css/animated_wicked_css3_3d_bar_chart.html](http://www.marcofolio.net/css/animated_wicked_css3_3d_bar_chart.html)
More On the transition Property


- Creating a moving background image; see https://paulrhayes.com/2009-04/auto-scrolling-parallax-effect-without-javascript/
## transition Browser Support

<table>
<thead>
<tr>
<th></th>
<th>IE</th>
<th>FIREFOX</th>
<th>OPERA</th>
<th>SAFARI</th>
<th>CHROME</th>
</tr>
</thead>
<tbody>
<tr>
<td>No</td>
<td></td>
<td>Yes with -moz-, 4+</td>
<td>Yes with -o-, 10.5+</td>
<td>Yes with -webkit-</td>
<td>Yes with -webkit-</td>
</tr>
</tbody>
</table>
Fading Out the Colour Using CSS Animations

- CSS3 animations can change between as many values as you want.

- They do this by letting you set up a series of keyframes, or points in time within the animation, each with its own set of styles.

- The browser then smoothly transitions between the keyframes in order, gradually changing all the properties included in each one.
Fading Out the Colour Using CSS Animations

- Let’s look at briefly showing the background colour on the current heading and then fading it out.

- It would be great if we could use transitions to do this, as transitions have better browser support than animations.

- Unfortunately, transitions won’t work here, because we need each heading to go from transparent (before it’s hovered over) to blue (when it’s hovered over) to transparent again (a second after its hovered over). That’s three points of change, and transitions can only handle changing between two values.
To create an animation in CSS, you first need to give it a name of your choosing and define what it will do at each keyframe. You do this using an `@keyframes` rule:

```css
@-webkit-keyframes fade {
0% {background: hsla(203,78%,36%,.2);}
100% {background: none;}
}
@-moz-keyframes fade {
0% {background: hsla(203,78%,36%,.2);}
100% {background: none;}
}
@-o-keyframes fade {
0% {background: hsla(203,78%,36%,.2);}
100% {background: none;}
}
keyframes fade {
0% {background: hsla(203,78%,36%,.2);}
100% {background: none;}
}
```
Fading Out the Colour Using CSS Animations

- This assigns a name of “fade” to our animation and specifies two keyframes: one zero percent of the way through the duration (in other words, at the very beginning) and one 100 percent of the way through the duration (in other words, at the very end).

- We could also have used the keywords `from` and `to` in place of 0% and 100% to denote the starting and ending states.
Now that we’ve defined what we want our animation to do, we need to apply it to an element.

We want the animation to occur on targeted h2 elements, so create a new h2:hover rule and place it under the existing one (soon you’ll see why we don’t want to add on to the existing h2:hover rule).
Fading Out the Colour Using CSS Animations

h2:hover{
    padding: 0;
    background: none;
    -webkit-animation-name: fade;
    -webkit-animation-duration: 2s;
    -webkit-animation-iteration-count: 1;
    -webkit-animation-timing-function: ease-in;
}
Fading Out the Colour Using CSS Animations

- This tells the browser that the animation we want to run on this element is named “fade” (using `-webkit-animation-name`).

- We want the animation to take two seconds (the `webkit-animation-duration` value) and run only once (the `webkit-animation-iteration-count` value).

- We’ve also told the browser we want the animation to ease in, making it slightly slower at the beginning than the end.
Fading Out the Colour Using CSS Animations

- You can combine all these properties into the animation shorthand property.

- Combine the -webkit- prefixed properties into the -webkit-animation property, and also add the other prefixed properties and the non-prefixed animation property to the h2:hover rule:

```css
h2:hover{
    padding: 0;
    background: none;
    -webkit-animation: fade 2s ease-in 1;
    -moz-animation: fade 2s ease-in 1;
    -o-animation: fade 2s ease-in 1;
    animation: fade 2s ease-in 1;
}
```
Fading Out the Colour Using CSS Animations

- This second `h2:hover` rule also removes the left padding and background color declared in the first one.

- If we didn’t remove the background, the animation would run once, and then once it was over, it would display the static background color of the h2, popping back to blue suddenly and staying blue.

- We need the heading to have no background so the animation can control the background entirely; once the animation is over, we want the heading to remain transparent.
Fading Out the Colour Using CSS Animations

- Removing the padding, on the other hand, is optional. I’ve chosen to remove it because it doesn’t make sense to have the heading text indented once the background colour has faded away.
Getting More Complex

- You can add as many properties to a keyframe as you want. You simply write them between the curly brackets of the keyframe like you would in any other CSS rule.

- E.g., if you wanted to change the font size as well as the background color:

```css
@-webkit-keyframes fade {
  0% {
    background: hsla(203,78%,36%,.2);
    font-size: 100%;
  }
  100% {
    background: none;
    font-size: 120%;
  }
}
```
Getting More Complex

- It's also possible to assign more than one animation to a single element, so you could break each of the above property changes out into its own animation:

```css
@-webkit-keyframes fade {
  0% {background: hsla(203, 78%, 36%, .2);}
  100% {background: none;}
}

@-webkit-keyframes scaleText {
  0% {font-size: 100%;}
  100% {font-size: 120%;}
}
```
Getting More Complex

- Then declare both animations on one element:

```css
h2:target {
  -webkit-animation-name: fade, scaletext;
  -webkit-animation-duration: 2s;
  -webkit-animation-iteration-count: 1;
  -webkit-animation-timing-function: ease-in;
}
```
Defining the animations separately takes more code, but may make it easier to keep track of what’s happening at which points in complex animations, and it allows you to control the duration, iteration, and other properties of each independently.

Another advantage is that you can reuse each animation on other elements.
Instead of using animation-delay to put off the start of the animation, we can create a delay within the animation itself by adding another keyframe to the animation that keeps the colour the same shade of blue:

```css
@-webkit-keyframes fade {
  0% {background: hsla(203,78%,36%,.2);}
  20% {background: hsla(203,78%,36%,.2);}
  100% {background: none;}
}

@keyframes fade {
  0% {background: hsla(203,78%,36%,.2);}
  20% {background: hsla(203,78%,36%,.2);}
  100% {background: none;}
}
```
Getting More Complex

- Now the animation will start immediately by displaying a blue background, keep showing that background until 20 percent of the way through the animation’s duration (.4 seconds), and then start fading to transparent.
The animation property is part of the Animations module found at www.w3.org/TR/css3-animations.

It’s shorthand for animation-name, animation-duration, animation-timing-function, animation-delay, animation-iteration-count, and animation-direction (in that order).
More on the animation Property

- Before using the above properties to apply an animation to an element, you first name the animation and define what it does using an @keyframes rule.

- The keyframes are multiple points in time through the duration of the animation, indicated with percentages; the keywords from and to correspond to 0% and 100%, respectively.

- Each keyframe contains style rules that should apply at that point in time. The browser gradually changes the styles from one keyframe to the next.
More on the **animation** Property

- Other than fading out a background colour, you might want to use animation for:
  - Pulsing glow on buttons; see [www.zurb.com/playground/radioactive-buttons](http://www.zurb.com/playground/radioactive-buttons)
  - Making elements roll into view; see [www.zurb.com/playground/sliding-vinyl](http://www.zurb.com/playground/sliding-vinyl)
animation Browser Support

<table>
<thead>
<tr>
<th>Property</th>
<th>Firefox</th>
<th>Chrome</th>
<th>Safari</th>
<th>Opera</th>
<th>Mozilla</th>
</tr>
</thead>
<tbody>
<tr>
<td>@keyframes</td>
<td>10.0</td>
<td>4.0 -webkit-</td>
<td>16.0 5.0 -moz-</td>
<td>4.0 -webkit-</td>
<td>15.0 -webkit- 12.1 12.0 -o-</td>
</tr>
<tr>
<td>animation</td>
<td>10.0</td>
<td>4.0 -webkit-</td>
<td>16.0 5.0 -moz-</td>
<td>4.0 -webkit-</td>
<td>15.0 -webkit- 12.1 12.0 -o-</td>
</tr>
</tbody>
</table>