EXERCISE 18-1. Trying out transitions

In this exercise, we're going to create the rollover and active states for a menu link (FIGURE 18-5) with animated transitions. I've put together a starter document (exercise_18-1.html) for you in the folder this document resides in. The folder also includes a model solution. Be sure you are using an up-to-date desktop browser to view your work (see **Note**).

NOTE

If you're using a touch device for this exercise, you'll miss out on this effect because there is no hover state on touch screens. You may see the hover state with a single tap. Transitions triggered by a click/tap or when the page loads will work on all devices, but they are not covered here.

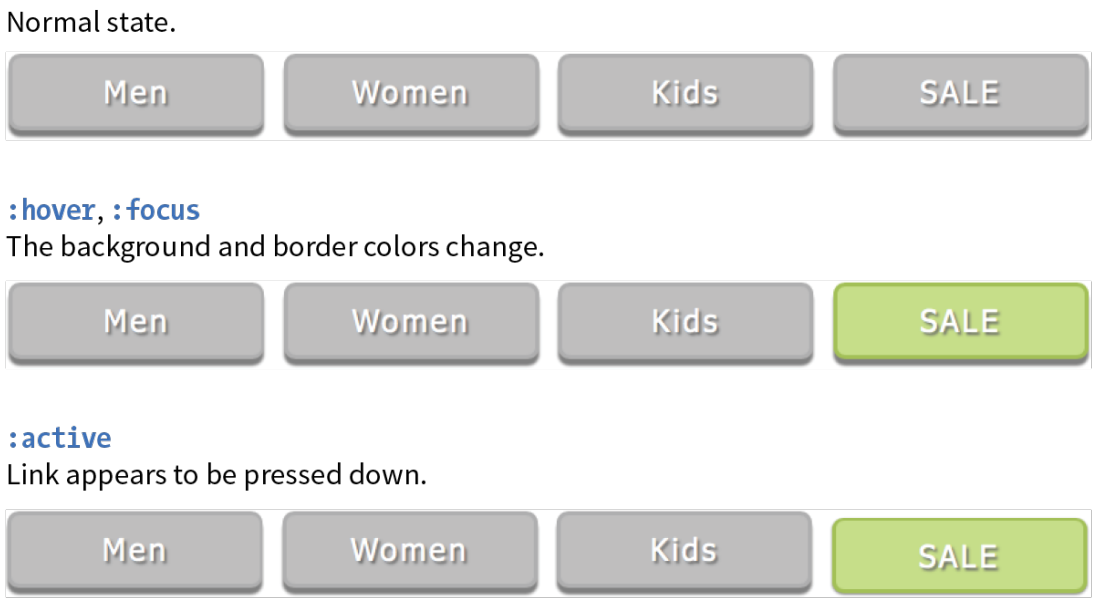

FIGURE 18-5. In this exercise, we'll create transitions between these link states. (Only right-most button is interacted with)

First, take a look at the styles that are already applied. The list has been converted to a horizontal menu with Flexbox. The a element has been set to display as a block element, underlines are turned off, dimensions and padding are applied, and the color, background color, and border are established. I used the box-shadow property to make it look as though the links are floating off the page.

1. Now we'll define the styles for the hover and focus states. When the user puts the pointer over or tabs to the link, make the background color change to green (#c6de89) and the border color change to a darker shade of green (#a3c058).

   ```
   a:hover, a:focus {  background-color: #c6de89;  border-color: #a3c058;}
   ```

2. While the user clicks the link (:active), make it move down by 3 pixels as though it is being pressed. Do this by setting the a element's position to relative and its top position to 0px , and then change the value of the top property for the active state. This moves the link 3 pixels away from the top edge (in other words, down).

   NOTE: Setting the top to 0px in the initial state is for working around a bug that arises when transitioning the top, bottom, left, and right properties.

   ```
   a {  …  position: relative;  top: 0px;}a:active {  top: 3px;}
   ```

3. Logically, if the button were pressed down, there would be less room for the shadow, so we'll reduce the box-shadow distance as well.

```
a:active   top: 3px;  box-shadow: 0 1px 2px rgba(0,0,0,.5);}
```

4. Save the file and give it a try in the browser. The links should turn green and move down when you click or tap them. I'd say it's pretty good just like that. Now we can enhance the experience by adding some smooth transitions.
5. Make the background and border color transition ease in over 0.2 seconds, and see how that changes the experience of using the menu. I'm using the shorthand transition property to keep the code simple. I'm also using the default ease timing function at first so we can omit that value.

   I'm not using any vendor prefixes here because modern browsers don't need them. If you wanted to support mobile browsers released in 2013 and earlier, you could include the -webkit- prefixed version as well, but since this isn't production code, we're fine without it.

```
a {  transition: background-color 0.2s,              border-color 0.2s;}
```

6. Save your document, open it in the browser, and try moving your mouse over the links. Do you agree it feels nicer? Now I'd like you to try some other duration values. See if you can still see the difference with a 0.1s duration. Now try a full second (1s). I think you'll find that 1 second is surprisingly slow. Try setting it to several seconds and trying out various timing-function values (just add them after the duration times). Can you tell the difference? Do you have a preference? When you are done experimenting, set the duration back to 0.2 seconds.
7. Now let's see what happens when we add a transition to the downward motion of the link when it is clicked or tapped. Transition both the top and box-shadow properties because they should move in tandem. Let's start with a 0.2s duration like the others.

```
a {  transition:    background-color 0.2s,    border-color 0.2s,
     top 0.2s,    box-shadow 0.2s;}
```

Save the file, open it in the browser, and try clicking the links. That transition really changes the experience of using the menu, doesn't it? The buttons feel more difficult to "press." Try increasing the duration. Do they feel even more difficult? I find it interesting to see the effect that timing has on the experience of a user interface. It is important to get it right and not make things feel sluggish. I'd say that a very short transition such as 0.1 second—or even no transition at all—would keep these buttons feeling snappy.

8. If you thought increasing the duration made the menu uncomfortable to use, try adding a short 0.5-second delay to the top and box-shadow properties.

```
a {  transition:    background-color 0.2s,    border-color 0.2s,
     top 0.1s 0.5s,    box-shadow 0.1s 0.5s;}
```

I think you'll find that little bit of extra time makes the whole thing feel broken. Timing is everything!

EXERCISE 18-2. Transitioning transforms

In this exercise, we'll make the travel photos in the gallery shown in FIGURE 18-13 grow and spin out to an angle when the user mouses over them—and we'll make it smoooooth with a transition. A starter document (exercise_18-2.html) and all of the images are available in the folder this document resides in.

FIGURE 18-13. Photos get larger and tilt on :hover and :focus . A transition is used to help smooth out the change between states. You can see how it works when you are finished with this exercise (or check it out in the ch18_figures.html page).

1. Open exercise_18-2.html in a text editor, and you will see that there are already styles that arrange the list items horizontally and apply a slight drop shadow. The first thing we'll do is add the transform property for each image.
2. We want the transforms to take effect only when the mouse is over the image or when the image has focus, so the transform property should be applied to the :hover and :focus states. Because I want each image to tilt a little differently, we'll need to write a rule for each one, using its unique ID as the selector. You can save and check your work when you're done.

```
a:hover #img1, a:focus #img1 {  transform: rotate(-3deg);} a:hover #img2,
a:focus #img2 {  transform: rotate(5deg); } a:hover #img3, a:focus #img3 {
transform: rotate(-7deg);}a:hover #img4, a:focus #img4 {  transform:
rotate(2deg);}
```

NOTE

As of this writing, prefixes are still recommended for the transform property, so for production-quality code, the complete rule would look like this:

```
a:hover #img1, a:focus #img1 {  -webkit-transform: rotate(-3deg);  -
ms-transform: rotate(-3deg); /* for IE9 */  transform: rotate(-
3deg);}
```

Because we are checking our work on a modern browser, we can omit the prefixes for this exercise.

3. Now let's make the images a little larger as well, to give visitors a better view. Add scale(1.5) to each of the transform values. Here is the first one; you do the rest:

```
a:hover #img1 {  transform: rotate(-3deg) scale(1.5);}
```

Note that my image files are created at the larger size and then scaled down for the thumbnail view. If we started with small images and scaled them larger, they would look crummy.

4. As long as we are giving the appearance of lifting the photos off the screen, let's make the drop shadow appear to be a little farther away by increasing the offset and blur, and lightening the shade of gray. All images should have the same effect, so add one rule using a:hover img as the selector.

```
a:hover img {  box-shadow: 6px 6px 6px rgba(0,0,0,.3);}
```

Save your file and check it out in a browser. The images should tilt and look larger when you mouse over them. But the action is kind of jarring. Let's fix that with a transition.

5. Add the transition shorthand property to the normal img state (i.e., not on :hover or :focus). The property we want to transition in this case is transform. Set the duration to 0.3 seconds and use the linear timing function.

```
img {  …  transition: transform 0.3s linear;}
```

NOTE

The prefixed transform property should be included in the context of a transition as well, as shown in this fully prefixed declaration:

```
-webkit-transition: -webkit-transform .3s linear;
```

The -ms- prefix is not needed because transitions are not supported by IE9. Those users will see an immediate change to the transformed image without the smooth transition, which is fine.

And that's all there is to it! You can try playing around with different durations and timing functions, or try altering the transforms or their origin points to see what other effects you can come up with.