Styling the Photos & Text

Exercise Overview

Throughout the exercises in this book, you will code an image-rich webpage and learn how to take advantage of the awesome native CSS capabilities of modern browsers. You'll build an elegant, responsive design where all content—including images and videos—resize in accordance with browser size. You will also explore techniques for using CSS animated transitions that zoom, pan, and scroll. Along the way, you'll use media queries to fine tune the content so it displays beautifully in any size browser, from a mobile device to a wide-screen display. In this exercise you'll get started by styling sections that have photo backgrounds which remain fixed on-screen while their text scrolls over top.

Getting Started

Before we get coding, let's look at a finished version of the webpage we're about to create and check out all of its great new CSS features.

1. Open a web browser. To see all the effects built into this page you'll need to use Chrome, Safari, Firefox, or Internet Explorer 10+. Internet Explorer 9 and older do not support all the features.
2. To open a file, hit Command–O (Mac) or Control–O (Windows).
3. Navigate into Desktop > Class Files > Responsive CSS3 Scrolling Effects > Hawaii Done and double-click on index.html to open it. We want to do a quick survey of all the highlights of the finished webpage.
4. First, make sure your browser window is as large as possible so you're looking at the desktop design.
5. Scroll through the site, and take note of the following:
    - The full-size background images don't scroll along with the page, as is customary, but have a special static or "sticky" behavior.
    - The Mt. Haleakala image displays as a moving panorama.
    - Web fonts are fully integrated and the title fonts have a subtle shadow.
    - The YouTube video zooms to full-size when you mouse over it.
    - Most exciting of all, when you resize the browser window, note the responsive behavior of all the text and image elements. Much of the coding we'll be doing will focus on ensuring that the page looks great in any size browser. These are the CSS features you'll soon see how to implement!
6. Let's get to work! Launch your preferred code editor (such as Sublime Text, Atom, Dreamweaver, etc.).
7. Open the following file: Desktop > Class Files > Responsive CSS3 Scrolling Effects > Hawaii > index.html (Be sure to go into the Hawaii folder, not one of the Done folders.) If you're using a code editor that allows you to open an entire project folder, such as Sublime Text, we suggest you open the Hawaii folder.
8. This file (index.html) is contains the text for the Hawaii webpage and some of the basic layout. On line 8 note the viewport meta tag. <meta name="viewport" content="width=device-width, initial-scale=1"> We added this code to save you some work, but it's important to know that this tag is required for responsive pages to display properly on mobile devices.
9. Let's check this page out in a web browser so we can see what work we need to do. Open a web browser and hit Command–O (Mac) or Control–O (Windows).
10. Navigate to Responsive CSS3 Scrolling Effects > Hawaii and double–click on index.html to open it.
11. Resize the window, making it narrower and wider. Notice how the main text column has a max-width of 1000 pixels. This prevents the lines from getting too long to comfortably read, but allows for it to get narrower on small screens. When made narrower, the text already flows responsively because elements like the callout (pull quote) have a percentage-based width.

At this point the page is devoid of images, but there are text-based placeholders where we are going to make the photo panorama and the photo stack.

Coding the Image Styles

Because we want the images to do various things (such as animate or stay fixed as we scroll) we'll create div tags for them instead of using img tags.

1. Switch back to index.html in your code editor.
2. Around line 17 add the following bold code:

```
<div class="page-layout">
    <div class="feature-wrap">

    </div>
<div class="main-wrap">
```

By default, divs are 100% of the width of their parent element. We do want this div to be full-width, so that's all set, but we also want the div to have a 2-pixel border at the top. We will have to add this in the CSS file.

3. Save index.html.
4. Open the following file: Hawaii > css > style.css
5. Take a brief look at the style.css file and notice that some basic CSS styles have already been defined to save you some work.
6. Let's create a feature-wrap rule to create 2-pixel borders at the top and bottom of the div. Find the .main-wrap rule (around line 22) and add the following bold code underneath it:

```
.main-wrap {
    max-width: 1000px;
    margin: 45px auto;
    padding: 0 15px;
    font-family: 'Vollkorn', Georgia, serif;
}
.feature-wrap {
    border-top: 2px solid #181818;
    border-bottom: 2px solid #181818;
}
```

7. Save style.css.
8. Preview index.html in a web browser. Look very carefully for the dark border at the very top of the page. This will look much better once we get a picture inside.

NOTE: We recommend leaving the page open in the browser while you work so you can simply reload it to see each change.

9. Switch to index.html in your code editor.

10. Let's take a moment to think about how we'll code the page. We will have four feature-wrap divs, each containing photo highlights and text descriptions. Although the photos will share some of the same styles, each feature photo will have slight variations on the theme. It will be useful to lay out each separate photo feature as its own self-contained module—or div— that sits inside the feature-wrap div. Let's start by adding the div we'll use for the lead photo and header information.
11. Around line 18, inside the feature-wrap div, add a new div for photo feature headings, like so:

```
<div class="feature-wrap">
    <div class="photo lead">

    </div>
</div>
```

Note that we'll use the photo class to share styles with all feature photos but the second class (lead) will be used to vary the style from feature to feature.
12. Next, let's add some variously sized header text that will overlay the photograph in the newly created div. Type the following bold code:

```
<div class="photo lead">
    <h3>Dan Rodney | May 2</h3>
    <h1>The Islands of Hawaii</h1>
    <h2>Diverse and Majestic</h2>
</div>
```

13. Save the file.
14. Switch to style.css.
15. Scroll down to the very bottom of the file and add the following style:

```
.feature-wrap .lead {
padding: 10% 0 30% 0;
}
```

Because we want this site to be a really responsive webpage, the padding values will often be percentages rather than absolute values.
NOTE: The order of four padding values goes clockwise starting at the top, or if you prefer a mnemonic: Top, Right, Bottom, Left = TRBL (Trouble).
16. Save style.css.
17. Reload index.html in your web browser. Notice the large headings, and try resizing the browser window in various ways to see how the 10% top padding and 30% bottom padding behave responsively to the changing window size.
18. Return to style.css in your code editor.
19. Add a background image to the lead style by adding the bold code as follows:

```
.feature-wrap .lead {
padding: 10% 0 30% 0;
```

background-image: url(../img/lead.jpg);
}

20. Save the file and reload the browser. The background image is there, but the photo is very large and we can only see a small portion of the top-left corner. Resize the page to see that the background image doesn't change size, it's just being cropped. We want to see the whole lead background image and also have it resize responsively when we change the page size. In fact, we want all of our images to behave in this manner, so we should create a rule for the photo class.
21. Return to style.css and add the following bold code directly above the .feature-wrap .lead rule:

.feature-wrap .photo {
background-repeat: no-repeat;
background-size: cover;
}

22. Here's how the new style breaks down:
• background-repeat: no-repeat ensures that the image doesn't repeat (or tile) in an unsightly way.
• background-size: cover is a great new feature of CSS3 that tells the image to always display 100% of its width and 100% of its height, no matter the window size. This way we'll always have the full photo showing!
23. Save the file and reload the browser to see the changes.
• To see how the background-size: cover feature works, resize the page to different proportions.
• Scroll up and down on the page. Currently the background image scrolls with the page, but we want to make it stay static as we scroll.
24. Return to style.css in your code editor.
25. Add the following bold code:

.feature-wrap .photo {
background-repeat: no-repeat;
background-size: cover;
background-attachment: fixed;
}

26. Save the file and reload the browser. Try scrolling now, and see the new, fancier "static" behavior of the background image! You may notice that the cover setting is no longer working the same way. In wide windows it still works as expected, but when the window is narrower the photo becomes cropped instead of scaling to fit.

There is a known conflict between **background-attachment: fixed** and **background-size: cover**. This happens because fixed backgrounds are essentially removed from the DOM and are fixed relative to the window, whereas background-size: cover is positioned relative to the

containing element. Later on we'll refine this behavior at specific screen sizes using CSS media queries.

Styling the Header Text

Our headings incorporate custom fonts from Google Fonts. We've already loaded the fonts to save you some work. If you've never used Google Fonts on your own before, you can find out more at fonts.google.com

1. Return to style.css in your code editor.
2. Let's work on styling the header text. Add a new style below the .feature-wrap style (around line 32) as shown below:

```
.lead h1 {
    font-family: 'Rye', cursive;
    font-weight: 400;
    font-size: 3.6em;
    line-height: 1em;
    text-transform: uppercase;
}
```

3. Save style.css and reload the browser to see your changes. What a snazzy typeface —and it's a true web font, too! Unlike an image containing text, you are able to select this text, copy it, and most importantly, it will turn up in search engine results.
4. Return to style.css in your code editor.
5. The heading looks a bit too close to the left-hand margin. Add the following code to add responsive margins to the top, right, and left of the h1 heading:

```
.lead h1 {
    font-family: 'Rye', cursive;
    font-weight: 400;
    font-size: 3.6em;
    line-height: 1em;
    text-transform: uppercase;
    margin: 1% 10% 0 10%;
}
```

6. Save the file and reload the browser. Great, the spacing around the heading is much more balanced! Let's fix up the other headings to look just as good.
7. Return to style.css in your code editor.
8. Add the following two new styles beneath the .lead h1 style (around line 40):

```
.lead h2 {
font-family: 'Quicksand', sans-serif;
font-size: 1.6em;
line-height: 1em;
```

```
text-transform: uppercase;
margin: 1% 10% 0 10%;

}
.lead h3 {
font-family: 'Quicksand', sans-serif;
font-weight: 400;
font-size: 1.05em;
text-transform: uppercase;
margin: 0 10%;
}
```

NOTE: The h3 style has only two margins defined. **The first value** applies to the top and bottom margins, and the second value applies to the **left** and **right** margins. It's a quicker way of writing **margin: 0 10% 0 10%** and it looks cleaner too!

9. Save the file and reload the browser. Check out those new text styles! They're all aligned nicely. Again, all of the margins are percentage-based, so if you try resizing the window to different sizes and proportions, the spacing and margins will change in subtle ways, ensuring that the site looks beautiful in any browser. Our header section is looking good for now. Let's move on to coding and styling the photo panorama.

Styling the Photo Panorama

1. The photo panorama will be another full-width feature-wrap section. Switch to index.html in your code editor.
2. Scroll down until you find the words photo panorama (around line 48).
3. Delete the words photo panorama.
4. In the empty space, type the following:

```
<div class="feature-wrap">
    <div class="photo panorama">

    </div>
</div>
```

5. Save the file.
6. Switch to style.css.
7. Add the following style at the very bottom of the file:

```
.feature-wrap .panorama {
    padding: 10% 0;
    background-image: url(../img/panorama.jpg);
}
```

Remember that the .panorama is also inheriting all the features of the .photo style that we defined earlier (like background-size), so we only have to specify the things that will be different about the panorama section.
8. Save the file and preview index.html in the browser. Scroll down to see that the panorama photo is looking good! Notice it has inherited the cover property and the special static scrolling behavior from the .photo style.
9. Later on we will add a panoramic scrolling animation, but we're not going to worry about that yet. First let's work on the text that will overlay the panorama. Switch to index.html in your code editor.
10. Create a new div by typing the following around line 50:

```
<div class="photo panorama">
    <div class="text">

    </div>
</div>
```

11. This div will contain the panorama text. Type the following:

```
<div class="text">
    Mt. Haleakala's southern face is vastly different from the north.
</div>
```

12. Save the file.
13. Switch to style.css.
14. Create a new style at the very bottom of the document:

```
.feature-wrap .text {
    font-family: 'Rye', sans-serif;
    font-size: 3em;
    line-height: 1.1em;
    color: #f1f1f1;
    text-align: center;
    margin: 0 25%;
    text-shadow: 0 0 4px rgba(0,0,0, 0.5);
}
```

The text styling exists inside the .feature-wrap module because we only want this style to be used in the featured photograph areas.
15. The text-shadow property is a great new feature of CSS3. It will make our text easier to read as it floats over the images. Here is how that line of code breaks down:
• **text-shadow: 0 0** These two numbers are the x-offset and y-offset (setting them both to zero will position the shadow directly behind the text).
• **4px** This number controls the blur size of the shadow. A larger number will create a larger shadow.

• **rgba(0,0,0, 0.5)** The letters stand for red, green, blue, alpha (alpha means transparency). Setting all the RGB values to zero (as we just did) makes the shadow color black. The alpha works differently, on a scale from 0.0 to 1.0, where 0.0 is transparent and 1.0 is opaque. Because we set the value to 0.5, the text shadow will be 50% transparent.
16. Save the file and reload the browser. Check out the new style on the panorama text! The text shadow makes the blurb easier to read in light areas, and the percentage-based responsive margins keep the text perfectly centered when you resize the browser window. The panorama section is looking good for now, so let's move on to coding another one of the site's photo highlights.

Styling the Photo Stack

The next photo section we need to code is a photo stack: three large photos that will each occupy the entire screen as the viewer scrolls down the page.
1. Switch to index.html in your code editor.
2. Scroll down until you see the words photo stack around line 76. Delete them.
3. In the empty space, type the following:

```
<div class="feature-wrap">
    <div class="photo stack">
        <div class="text">
        Hawaiian Luau: Good food & hula dancers!
        </div>
    </div>
</div>
```

4. Great! Our photo stack has three pictures, though. Because we want the second and third sections of the photo stack to behave like the one we just created, copy everything you just typed.
5. Paste the code two times below. You should end up with a total of three chunks of identical code.
6. Add the captions as shown below in bold:

```
<div class="feature-wrap">
    <div class="photo stack">
        <div class="text">
            Hawaiian Luau: Good food & hula dancers!
        </div>
    </div>
</div>
<div class="feature-wrap">
    <div class="photo stack">
        <div class="text">
            You're going to need a boat to get here!
```

```
                </div>
        </div>
</div>
<div class="feature-wrap">
        <div class="photo stack">
                <div class="text">
                        Watch the sun rise from above. You're standing outside
                        looking down at the clouds!
                </div>
        </div>
</div>
```

7. Save the file and reload the browser. Scroll down to see the three blocks of text we just typed against the white background, with the dark div borders above and below. They're very close together, so let's add some padding.
8. Switch to style.css in your code editor.
9. Add a new style for the photo stack around line 102, directly above the **.feature-wrap .text** style. Type the following:

```
.feature-wrap .stack {
        padding: 55% 0;
}
```

10. Save the file and reload the browser. Now each caption has plenty of space around it —each photo stack div is 110% of the browser window's height because there is 55% padding on the top and 55% padding on the bottom. This is exactly what we want in order to make each photo take up the entire screen as the viewer scrolls down the page.
11. We have spots for three different stack photos, but we don't yet have a way to style them uniquely. Let's fix that. Switch to index.html in your code editor.
12. Make the changes shown below to differentiate each of the photo stack divs:

```
<div class="feature-wrap">
        <div class="photo stack stack-one">
                <div class="text">
                        Hawaiian Luau: Good food & hula dancers!
                </div>
        </div>
</div>
<div class="feature-wrap">
        <div class="photo stack stack-two">
                <div class="text">
                        You're going to need a boat to get here!
                </div>
        </div>
</div>
```

```
<div class="feature-wrap">
    <div class="photo stack stack-three">
        <div class="text">
            Watch the sunrise from above the clouds. You're standing outside
            looking down at the clouds!
        </div>
    </div>
</div>
```

13. Save the file.
14. Switch to style.css.
15. Type the following style for stack-one directly below the .feature-wrap .stack style, around line 105:

```
.feature-wrap .stack-one {
    background-image: url(../img/luau.jpg);
}
```

16. Copy what you just typed.
17. Paste it twice below (for a total of three chunks of identical code) as follows:

```
.feature-wrap .stack-one {
    background-image: url(../img/luau.jpg);
}
.feature-wrap .stack-one {
    background-image: url(../img/luau.jpg);
}
.feature-wrap .stack-one {
    background-image: url(../img/luau.jpg);
}
```

18. Change the photo filenames and the numbers of the stacks as shown below:

```
.feature-wrap .stack-one {
    background-image: url(../img/luau.jpg);
}
.feature-wrap .stack-two {
    background-image: url(../img/coast.jpg);
}
.feature-wrap .stack-three {background-image: url(../img/clouds.jpg);
}
```

19. Save the file and reload the browser. The photo stack should be looking very good. The images are static and the text scrolls as we move down the page. Maybe it’s a bit boring to keep all the text centered, though. Let’s change up the text alignment by creating classes that will let us align some of the text blurbs to the left and align others to the right.

20. Switch to index.html in your code editor.
21. Find the following code and make the changes in bold:

```
<div class="feature-wrap">
        <div class="photo stack stack-one">
                <div class="text left">
                        Hawaiian Luau: Good food & hula dancers!
                </div>
        </div>
</div>
<div class="feature-wrap">
        <div class="photo stack stack-two">
                <div class="text right">
                        You're going to need a boat to get here!
                </div>
        </div>
</div>
<div class="feature-wrap">
        <div class="photo stack stack-three">
                <div class="text left">
                        Watch the sun rise from above the clouds. You're standing outside
                        looking down at the clouds!
                </div>
        </div>
</div>
```

22. Save the file.
23. Switch to style.css.
24. Scroll to the bottom of the file and add the following rule to make sure that the photo stack text is a bit narrower, only 35% of the width of the parent div:

```
.feature-wrap .stack .text {
        width: 35%;
}
```

25. Below that, add a rule for left aligned text with a small left-hand margin:

```
.feature-wrap .stack .text.left {
        text-align: left;
        margin: 0 0 0 10%;
}
```

26. Almost done styling this text! Add the right aligned style as follows:

```
.feature-wrap .stack .text.left {
        text-align: left;
```

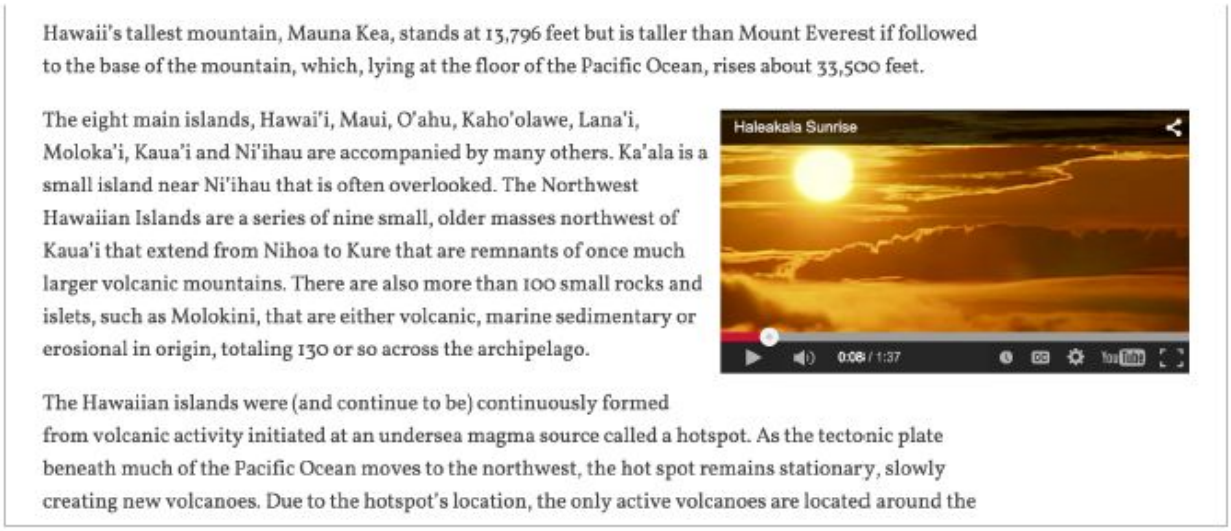
```
        margin: 0 0 0 10%;
}

.feature-wrap .stack .text.right {
        text-align: right;
        margin: 0 0 0 50%;
}
```

27. Save the file and reload the browser. Scroll down to check out the improvements to the photo stack! The text looks much more sophisticated with the latest changes. We've done the basic styling on the three major feature-wrap sections of the site.

Creating an Animated CSS Transition for a YouTube Video

Exercise Preview



Exercise Overview

In this exercise you'll add a YouTube video that enlarges when you mouse over it. You'll use a CSS transition so it enlarges smoothly, with no JavaScript required!

Positioning & Sizing the YouTube Video

1. In index.html, around line 107 add the following bold code to create a home for the YouTube video:

<p>
Hawaii's tallest mountain, Mauna Kea, stands at 13,796 feet but is taller than Mount Everest if followed to the base of the mountain, which, lying at the floor of the Pacific Ocean, rises about 33,500 feet.

```
</p>
        <div class="zoom">
            <div class="video-wrap">

            </div>
        </div>
<p>
```

The eight main islands, Hawai'i, Maui, O'ahu, Kaho'olawe, Lana'i, Moloka'i, Kaua'i and Ni'ihau are accompanied by many others.

2. Save the file.
3. Open style.css (in the css folder).
4. Scroll down to the very bottom of the file and add the following style:

```
.zoom {
    float: right;
    width: 40%;
    margin: 7px 20px 15px 7px;
    padding: 5px;
    border: 1px solid #000;
}
```

5. Save the file and reload the browser. Scroll down until you see the small, empty rectangle, the video's future home. We won't need the border and padding for the final design but it's helpful to have them here to gauge the size and positioning of the video wrapper as you work. Try changing the proportions of the browser window and see how the video div resizes automatically to occupy 40% of the browser's width.

It's pretty easy to tell a div to resize to a certain percentage of a browser's width, like we just did. Unfortunately, it will be trickier to make the box's height resize in proportion to the browser window while maintaining the video's 16:9 aspect ratio — this is because there is no height unit that will take into account the page's width.

The solution to getting the box to resize in height proportionally as we change the page's width is to use some percentage-based padding to hold the box open! Percentage-based padding will use the width of the box as the value of its percentage. Let's add that now.
6. Return to style.css in your code editor.
7. At the very bottom of the file add the following style:

```
.video-wrap {
    padding-top: 56.25%;
}
```

This means that whatever the width of the box, we want the height to be 56.25% of that width. How did we come up with that number? The aspect ratio of the video is 16:9, and 9/16 = 56.25%.
8. Save the file and reload the browser. While looking at the video box (towards the bottom of the page), try changing the proportions of the browser window. This is perfect! The box maintains the 16:9 aspect ratio as it resizes.
9. Return to your code editor.
10. Open Hawaii > snippets > youtube-iframe.html This is the standard embed code that YouTube gives you when you click on Share. This will allow us to embed their video into our page.
11. Select all the code in the file.
12. Copy the code.
13. Close youtube-iframe.html.
14. Switch to index.html in your code editor.
15. Paste the code you just copied into the video-wrap div (around line 109):

```
<div class="zoom">
    <div class="video-wrap">
        <iframe width="560" height="315"
        src="http://www.youtube.com/embed/
        G0K9Ht0N_as?rel=0" frameborder="0" allowfullscreen></iframe>
    </div>
</div>
```

16. Save the file and reload the browser. The YouTube video is appearing on the page, but it still needs some work. We need to position the video over the padding in the box, and make sure that the width and height of the video corresponds to the size of the box. Let's use absolute positioning to get it where we want it!
17. Switch to style.css in your code editor.
18. Towards the bottom of the file, add the following code shown in bold:

```
.video-wrap {
    padding-top: 56.25%;
    position: relative;
}
```

We're going to use absolute positioning to position the iframe inside our .video-wrap. Therefore we need to declare a position (relative or absolute) on the nearest parent element that we wish to use as a positioning anchor. If we don't declare a position on the nearest relative parent (in this case the .video-wrap), the body tag will be used as the positioning anchor. Now the iframe inside the wrapper can use absolute positioning relative to the wrapper itself.

19. At the very bottom of the document, add a rule for the iframe that'll hold the video:

```
.video-wrap iframe {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
}
```

This style specifies that the iframe (which is a child of the .video-wrap) is using absolute positioning to be placed in the upper left corner of the wrapper.

20. Save the file and reload the browser. Looking excellent! The YouTube video should be sitting over the top of the padding. Even better, since we told it to take up 100% of the width and height of the available space, it should resize perfectly if you play around with changing the proportions and size of the browser window. Now we have a fully responsive, easily resizing video.

21. Return to style.css in your code editor.

22. We no longer need the border and padding on the rule for the zoom div, so delete those two properties. When you're done, the rule for .zoom should read as follows:

```
.zoom {
    float: right;
    width: 40%;
    margin: 7px 20px 15px 7px;
}
```

Adding an Animated CSS Transition on Hover

1. We also want our video to actually zoom when a visitor hovers over it! Let's make that style now. Add the following code directly below the zoom rule, around line 139, as shown below:

```
.zoom:hover {
    width: 100%;
    margin: 15px 0;
}
```

This new class ensures that when a visitor mouses over (hovers over) the video, it will expand to be 100% of the width of the browser.

2. Save the file and reload the browser. Great! Our hover style should be working! However, it is rather jumpy as it switches from occupying 40% to 100% of the browser's width. Let's add an animation that makes this move less jerky.
3. Return to style.css in your code editor.
4. Let's implement another cool new CSS3 feature: the ability to animate transitions between two states. We can specify that whenever any property in our zoom class undergoes a change, that change should be animated. Add the following bold code:

```
.zoom {
        float: right;
        width: 40%;
        margin: 7px 20px 15px 7px;
        transition: all 0.4s ease-in-out;
}
```

This rule says to transition all of the properties (in our case, that will include changes to the width and margins) over the course of 0.4 seconds. The ease-in-out option slows the animation at the beginning and end.
5. In the future, the line of code that we just entered will be all that's needed to make the animation work in any browser. Unfortunately, at this point in time we need to add all the vendor prefixes to ensure it functions correctly in all browsers. Copy the line of code you just wrote and paste it three times as shown below:

```
.zoom {
        float: right;
        width: 40%;
        margin: 7px 20px 15px 7px;
        transition: all 0.4s ease-in-out;
        transition: all 0.4s ease-in-out;
        transition: all 0.4s ease-in-out;
        transition: all 0.4s ease-in-out;
}
```

6. Add the vendor prefixes as shown below in bold:
-webkit-transition: all 0.4s ease-in-out;
-moz-transition: all 0.4s ease-in-out;
-o-transition: all 0.4s ease-in-out;
transition: all 0.4s ease-in-out;

The prefixes you just typed refer to, in order: WebKit (which powers Safari, Chrome, and most mobile browsers), Mozilla, and Opera. You should always include the unprefixed property—the one that future implementations of browsers will use—at the end of the list.
7. Save the file and preview index.html in the web browser. Try hovering over the video to see its amazing CSS3-powered zoom animation! How incredible that we can make this animation just using CSS, no Flash or JavaScript required.

Internet Explorer Compatibility
IE 10 and later support the CSS (unprefixed) transition property. All earlier versions of IE will ignore the transition, prefixed or not. So viewers using IE 9 can enjoy the site but will not see the animation.