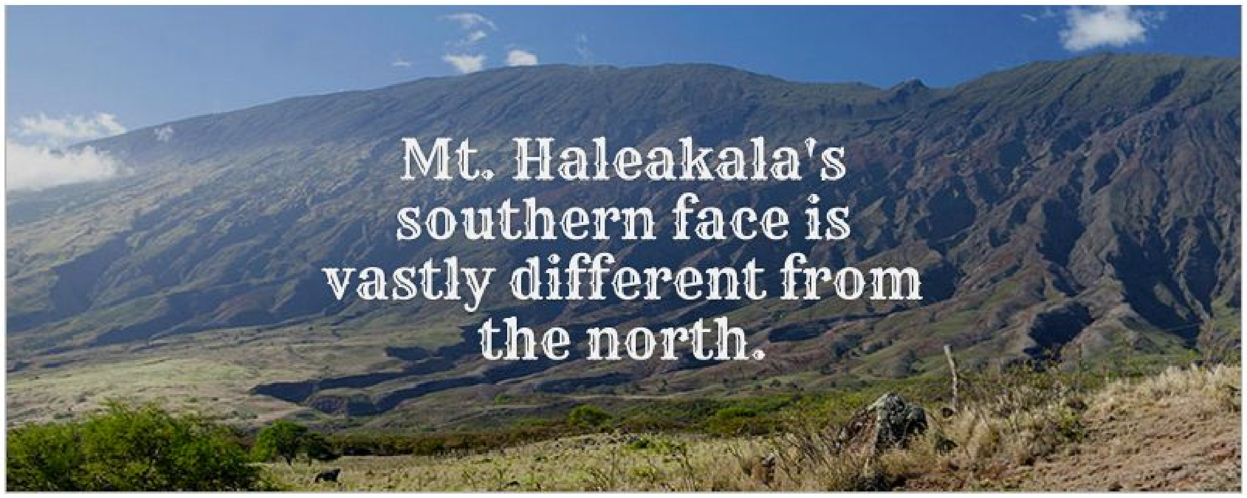
Animating the Panorama & Making the Layout Responsive


Mt. Haleakala's southern face is vastly different from the north.

Exercise Overview

The first thing you'll do in this exercise is animate the panorama background photo, so it pans back and forth. You'll then finish up the layout by making it responsive, so it adapts to any size screen.

Animating the Panorama Background

Right now we have a very wide image in the panorama segment of the page, and we can only see a small portion of the vista. We want to make the background image scroll slowly from left to right. We can't use a transition on this element because we're not changing between two different states. Instead, we have to create a CSS animation.

The most efficient way to get an animation running is to focus on making it work in one browser first, and later, once it's working, add all of the vendor prefixes. Let's start out creating this animation for Chrome/Safari (using the WebKit prefix).

1. Scroll down to the bottom of style.css and add the following code at the very end of the document:

```
@-webkit-keyframes moving-bg {
}
```

2. Define the animation's starting and ending keyframes by adding this bold code:

```
@-webkit-keyframes moving-bg {
        0% {
            background-position: 0 0;
        }
        100% {
            background-position: 100% 0;
        }
    }
```

The above positions are indicated by x, y coordinates. This code means that the starting position keyframe will be at 0 0 (the upper left corner of the image), and the ending position keyframe will be at the point when all of the image (100%) has moved to the right. The x position of the image will change from 0 to 100% while the y position remains constant (at zero), thereby achieving the "panning" effect with our very long, narrow background image.

3. Find the .feature-wrap .panorama class (around line 98). Add the background animation to the class with the following bold code:

```
    .feature-wrap .panorama {
        padding: 10% 0;
        background-image: url(../img/panorama.jpg);
        -webkit-animation: moving-bg 20s linear infinite alternate;
    }
```

4. A breakdown of the line of code you just wrote:
   - **moving-bg** is the name of our animation.
   - **20s** is the length of the animation, 20 seconds.
   - **linear** stands for linear easing, ensuring that the animation moves smoothly.
   - **infinite** means the animation will repeat infinitely.
   - **alternate** means that, once the animation ends, it will repeat in the alternate direction (i.e., go left to right, then right to left) for a seamless look.

5. Save the file and preview index.html in a WebKit-powered browser like Safari or Chrome. The photo behind the Mt. Haleakala text should be moving!
6. Return to your code editor.
7. Our WebKit animation is working, so we can now copy and paste the code and add the vendor prefixes. Copy the **-webkit-animation** line of code you just wrote.
8. Paste it three times below as follows:

```
    .feature-wrap .panorama {
        padding: 10% 0;
        background-image: url(../img/panorama.jpg);
        -webkit-animation: moving-bg 20s linear infinite alternate;
        -webkit-animation: moving-bg 20s linear infinite alternate;
        -webkit-animation: moving-bg 20s linear infinite alternate;
        -webkit-animation: moving-bg 20s linear infinite alternate;
    }
```

9. Change the vendor prefixes as follows, making sure the final one is unprefixed:
-webkit-animation: moving-bg 20s linear infinite alternate;
-moz-animation: moving-bg 20s linear infinite alternate;
-**o**-animation: moving-bg 20s linear infinite alternate;
animation: moving-bg 20s linear infinite alternate;

10. We need to create keyframes for every browser. Scroll to the bottom of style.css.
11. Copy the multiple lines of code for the **@-webkit-keyframes** block of code.
12. Paste the keyframes three times below. You should have four identical versions of the following chunk of code:

```
@-webkit-keyframes moving-bg {
0% {
	background-position: 0 0;
}
100% {
	background-position: 100% 0;
}
}
```

13. Change the vendor prefixes as indicated by the bold code, making sure the last one is unprefixed:

```
@-webkit-keyframes moving-bg {
	***CODE OMITTED TO SAVE SPACE
}
@-moz-keyframes moving-bg {
	***CODE OMITTED TO SAVE SPACE
}
@-o-keyframes moving-bg {
	***CODE OMITTED TO SAVE SPACE
}
@keyframes moving-bg {
	***CODE OMITTED TO SAVE SPACE
}
```

14. Save the file and reload the browser. Take a moment to admire your excellent work! You just created a CSS3 animation from scratch!

Creating Responsive Layouts with CSS Media Queries

Our Hawaii page is looking really good, but if you try resizing the browser window (making it very small, for example), the text doesn't get proportionally smaller. If someone viewed the site on a mobile browser or another small screen, the headlines might cover too much of the photos. We can customize the user experience based on the browser size by using CSS media queries.

1. Return to style.css in your code editor.
2. Type the following code above the first set of keyframes, around line 161:

```
**@media (max-width: 1024px) {
}**
```

This code is a media query that essentially means "for any browsers under 1024 pixels wide, use the following styles." A browser wider than 1024 pixels would never use the styles we're about to code.

3. Add the following bold code:

```
@media (max-width: 1024px) {
        .feature-wrap .photo {
        background-attachment: scroll;
        background-position: center center;
        }
}
```

This overrides the rule we defined in a previous exercise (which set a fixed background position on the photos) because that effect isn't ideal for smaller browsers; viewers would only be able to see a small portion of each photo.

4. Save the file and reload the browser.
5. Resize the browser window so it's large (more than 1024 pixels) and notice that the fixed backgrounds are intact. Then resize the browser to be smaller than 1024 pixels wide, and the backgrounds scroll along with the page, improving the user experience for small browsers.

Next, let's work on making the text sizes responsive to browser size.

6. Return to style.css in your code editor.
7. Add the following bold code to make the photo highlight text resize in small browser windows:

```
@media (max-width: 1024px) {
        .feature-wrap .photo {
                background-attachment: scroll;
                background-position: center center;
        }
        .feature-wrap .text {
                font-size: 2em;
        }
}
```

8. Save the file and reload the browser. Scroll down to the panorama or photo stack. View the browser at a small size and then at a size larger than 1024 pixels wide, noticing the responsive changes in text styles.
9. While the browser is still narrower than 1024 pixels, take note of the panorama animation you created earlier. It's a bit choppy. That's because it's now been scaled down to a small size and doesn't have very far to pan in either direction. Let's make the panorama larger at this size.
10. Return to style.css in your code editor and at the bottom of the media query, add the following bold code:

```
@media (max-width: 1024px) {
        .feature-wrap .photo {
                background-attachment: scroll;
                background-position: center center;
        }
        .feature-wrap .text {
                font-size: 2em;
        }
        .feature-wrap .panorama {
                padding: 15% 0;
        }
}
```

11. Save the file and reload the browser to see the improved panorama at this size. The zooming YouTube video doesn't look as great on small browsers, particularly in the “pre-zoom” state when the video is only 40% of the browser's width. Let's remove the zooming video from small browsers.
12. Switch back to style.css in your code editor.
13. Add the following bold code to remove the zooming feature from browsers smaller than 1024 pixels wide:

```
        .feature-wrap .panorama {
                padding: 15% 0;
        }
        .zoom {
                width: 100%;
        }
}
```

14. Save the file and reload the browser. Make the browser window smaller than 1024 pixels wide and notice that the zoom feature disappears. However, the space around the video needs to be fixed. Also, when you hover over the video, the space around it changes. Let's fix both of these issues.
15. Return to style.css in your code editor.
16. Add the following margins to the .zoom rule you just added:

```
        .zoom {
                width: 100%;
                margin: 0 0 25px 0;
        }
```

17. We'll need to add a new rule to this media query for the .zoom:hover as well.
Add the following bold code:

```
                .zoom {
                        width: 100%;
                        margin: 0 0 25px 0;
                }
                .zoom:hover {
                        margin: 0 0 25px 0;
                }
        }
```

We have essentially overridden the problematic hover property by adding a matching 25-pixel margin on the bottom of the video, so the video will display properly on smaller screens. Let's check that it's working.
18. Save the file and reload the browser. Make sure your browser window is narrower than 1024 pixels wide, then try hovering over the YouTube video to make sure the problem is fixed. Looking good!
19. Let's create more break points with more alternate styles for smaller browsers. Return to style.css in your code editor.
20. Add a new media query after @media (max-width: 1024px), around line 180:

```
        @media (max-width: 850px) {
        }
```

NOTE: The effect we're creating works best on desktops. We will make the page work on mobile devices, but some of the effects are targeted only to desktops. While we could have used a mobile first approach to build this page, we thought it more appropriate to use a desktop first approach because desktop users get most of the effects. In the end it will work on all devices, but the desktop first approach means the media queries will use max instead of min-width and start with large widths and progressively get smaller as you read down the code.
21. This media query will apply only to browsers less than 850 pixels wide. Let's modify the text size and margins at this break point. Add the following bold code:

```
        @media (max-width: 850px) {
                .feature-wrap .text {
                        font-size: 1.5em;
                        margin: 0 5%;
                }
        }
```

22. Save the file and reload the browser. Make the browser narrower than 850 pixels and notice the subtly smaller text sizes and margins. While in the browser at this size, take a look at the panorama photo again. Earlier, in the 1024 pixel media query, we specified that the panorama should have more padding to make the image larger and improve the animation. Let's further increase the padding of the panorama now that we are at 850 pixels or less, to continue to improve the animation's performance.

23. Return to style.css in your code editor and add the following bold code:

```
    @media (max-width: 850px) {
        .feature-wrap .text {
            font-size: 1.5em;
            margin: 0 5%;
        }
        .feature-wrap .panorama {
            padding: 20% 0;
        }
    }
```

24. Save style.css and reload the browser. Make the browser narrower than 850 pixels and notice how the panorama's animation has been improved at this width.

25. We've made good adjustments with media queries so far, but the behavior of the photo stack at this size could use refinement. Return to style.css in your code editor.

26. Add the following bold code to the 850 pixel media query, which will refine margins and center the text in the photo stack:

```
        .feature-wrap .panorama {
            padding: 20% 0;
        }
        .feature-wrap .stack .text {
            width: auto;
            margin: 0 5%;
            font-size: 3.5em;
            text-align: center;
        }
    }
```

27. Save the file and reload the browser. Make the browser narrower than 850 pixels — but wait! The text in the photo stack has strange margins, and it's not centered like we wanted. The problem is specificity: we did not match all the exact class names that we defined earlier.

28. Return to editing style.css in your code editor.

29. Fix the issue by adding the following bold code, paying special attention to the commas at the end of the first two lines:

```
    .feature-wrap .stack .text,
    .feature-wrap .stack .text.right,
    .feature-wrap .stack .text.left {
        width: auto;
        margin: 0 5%;
        font-size: 3.5em;
        text-align: center;
    }
```

30. Save the file and reload the browser, resizing the browser window to be narrower than 850 pixels. Take a moment to admire the new centered text on the photo stack!
31. Return to style.css in your code editor. The final break point in our design will be at 600 pixels. The font size will decrease again, and the callout text, rather than being an absurdly narrow column, will occupy the full width of the browser.
32. Add a new media query directly beneath the closing brace of **@media (max-width: 850px)** around line 197. Type the following bold code:

```
@media (max-width: 600px) {
}
```

33. Add the following bold code to fix the behavior of the callout class:

```
@media (max-width: 600px) {
    .callout {
        float: none;
        margin: 0;
        width: auto;
    }
}
```

34. Save the file and reload the browser. Try making the window narrower and wider than 600 pixels and see how the callout column (near the top of the page) changes responsively!
35. Return to style.css in your code editor.
36. Let’s create one more rule to make the photo stack text even smaller for browsers under 600 pixels. Add the following bold code:

```
@media (max-width: 600px) {
    .callout {
        float: none;
        margin: 0;
        width: auto;
    }
    .feature-wrap .stack .text,
    .feature-wrap .stack .text.right,
    .feature-wrap .stack .text.left {
        font-size: 2.5em;
    }
}
```

37. Save the file and reload the browser. Notice the way the photo stack text changes responsively when you resize the browser window!

The webpage is complete but if you would like to continue to use media queries to add even more refinement to the web-font typography at different break points, continue on to part three.