In this example, we're going to look at the best ways to ensure your animations are smooth, and work with the browser's strengths. Let's open the example HTML and see if we can spot what's gone wrong. In this example we have some boxes of content, when the user hovers their cursor over one of them.

It grows, the font size increases, and the background turns opaque. It's a little painful to see. The animation makes the content around to move and the resizing is jittery. Before we fix it, let's take a look at what's happening in the code. In our HTML, we have the usual demo container and in it, I've added a list of spacer divs.

Each div contains a space themed quote with several paragraphs of text. Let's see how this is being styled. In our stylesheet, stepping through the styles, we see we have a background image and some flex box styling on the demo container. This tells the browser to try to display each of the spacer elements in a grid based on their width of 25%.

The action is happening here toward the end of the document. We have some hover styles, we set the background color and padding on the spacer, we then set the font size on the paragraphs. And lastly we apply a transition property to both the spacer and everything inside it.

The reason the animation is janky for us is due to a couple of factors, the main one, being browser repaints. Back in the demo, we can see that the browser is having to move things around when we hover over the spacer container. It's recalculating positions and redrawing content, and trying to do so 60 times per second.

This is a lot of work on the browser, and if we had more content, could be terrible for performance. There are properties that browsers are good at animating. These tend to be the opacity and transform properties. Generally, if you animate something else it's possible that the browser won't be as efficient.

This means that animating the background color or the font size as we're doing here might not be playing to the browser strengths. Let's update our CSS to make use of opacity and transforms and see if we can make it perform better. Looking at our style sheet, the two properties that are contributing most to the repaint are the padding and the font size.

The padding changes the size of the container making it bigger and pushing other content around. And the font size causes the text to reflow within the box. Let's remove them. Instead, we can add a transform to the spacer on hover. We want it to be bigger on hover, so let's add in a transform and scale and we'll grow it by just a small amount say 1.15.

This will make the result 1.15 times larger on hover. Back in our browser we hover over each box and the result is a lot smoother. The box sizes up as expected, but crucially it's not making the rest of the page adapt. By not forcing the text to reflow, we're cutting back on the amount of work the browser needs to do.

Also since the browser is optimized to animate the transform property, we're working with the browser's strengths, rather than against. Before we finish the topic of browser considerations, I want to show one example that you might run into when animating. We won't be doing any coding on this one, but I've prepared some files so that you can open it in your browser and see what happens.

Here we have an animation where some elements are rotating, each is slightly differently sized and creating a wavy effect. While this animation is itself quite simple, it's using opacity and the use of multiple layers puts quite a strain on the browser. Even though it's using transforms for the rotation, the fans are kicking in pretty quickly here on my machine.

It’s worth keeping in mind when you create animations, whether CSS or HTML is the correct medium. In this example we might want to render this in canvas as that could potentially handle many more layers. When creating animations, it’s good to test the result in a range of browsers and devices and make sure it’s performing as you hope.

As well as testing for performance, it’s important to also test for accessibility, which we’ll discuss next.