HTML: Create scrolling effects

Put a bit of movement into HTML pages – **Ben Everard** demonstrates the power of Skrollr using an omelette recipe.

Our expert

Ben Everard left his job as an IT consultant to spend two years in Tanzania installing Ubuntu-based systems in schools. Now he's putting his skills to use in the roiling cauldron of discovery that is LXF Towers.

HTML is possibly the greatest innovation in text since the printing press. It's simple to write, doesn't care about language and can be displayed on almost any computer made in the past two decades. However, it does lack a little pizazz. Fortunately, though, it does enable us to build our own pizazz using CSS and JavaScript. In this tutorial, we're going to look at using the Skrollr JavaScript library to create various scrolling effects. This is where things animate themselves in some way as you scroll down a page.

To start with, you'll need a copy of the Skrollr library, which you can download from GitHub here: **http://bit.ly/1cTr8cX**), or you'll find it on the **LXFDVD**. The first job on the list is unzipping the file, and taking a look at the **index.html** file to see a whole host of effects that are possible (this file's also online at **http://prinzhorn.github.io/skrollr**).

To add these effects to a website, you just need to add a few lines to the HTML:

```
<meta name="viewport" content="width=device-width,
initial-scale=1, user-scalable=no">
```

This goes inside the **<head></head>** tags.

```
<div id="skrollr-body">
 </div>
```

All the contents of your website needs to go inside this div that sits inside the **<body></body>** tags.

```
<script type="text/javascript" src="dist/skrollr.min.js">
</script>
<script type="text/javascript">var s = skrollr.init();</script>
```

These two lines load and initialise the Skrollr script. You can add all of these to a HTML file manually, or you'll find a template in the Skrollr file called **shim.html**.

The one final thing to do before we start is add our stylesheet. This CSS file just contains a few styles that we'll use in this example. The fixed-positioning style sheet adds some pieces that Skrollr needs.

```
<link href="examples/fixed-positioning.css" rel="stylesheet"
type="text/css" />
```

Data properties

With all this done, we can start adding content. Our page won't really be like a page of text at all, but more like an animation that the viewers can control the speed of. Each element in the animation needs to be inside **<div></div>** tags. We then assign these tags data properties that tell Skrollr how to animate them. We're going to create a simple HTML page that teaches the reader how to make an omelette. The first div we'll add is just an introduction.

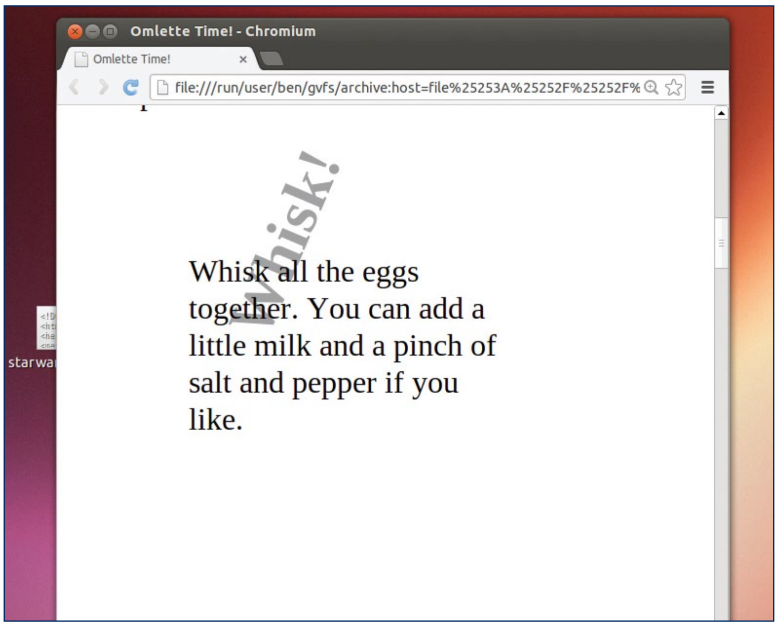
```
<div data-0="width:50%;height:50%;left:20%;top:25%;
opacity:1" data-200="opacity:0">
 <h1>Lets make an Omelette</h1>
 You'll need a knob of butter, two eggs and some cheese.
<br> Scroll down to get started
 </div>
```

Don't forget to add it inside the **skrollr-body** div. Skrollr uses data-x properties where x is any number from one upwards that corresponds to the number of pixels the user scrolls down. Whenever you change a property between two data-x properties, Skrollr will interpolate to change the div between them. In this case, we set the position at data-0, but the only thing that changes is the opacity, so Skrollr gradually changes that between these two data points.

Now we'll add our second div:

```
<div data-200="width:50%;height:50%;left:20%;top:25%;
opacity:0" data-250="opacity:1;color:rgb(0,0,0)" data-
400="color:rgb(255,0,0);opactiy:1" data-500="opacity:0" >
  Pop your frying pan on the heat with the butter and let it
heat up.
</div>
```

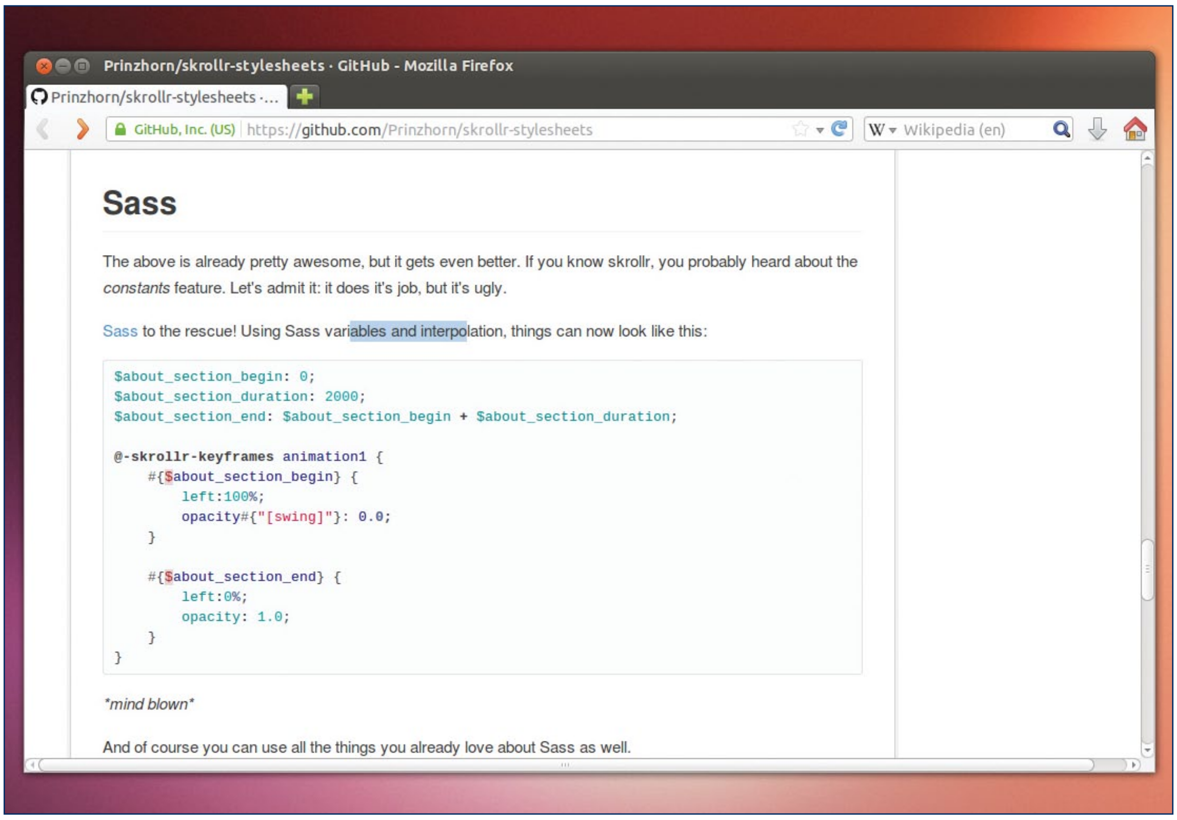
Here you can see a few things. First, you can have as many of the data points as you like. You should also see how we've animated the colour between two points. You'll notice that we've included the opacity property in every data point, even though it doesn't change between point 250 and 400. This is because we want it to animate between 200 and 250, then again between 400 and 500. In this case, we have to keep it in


Omlette Time! - Chromium
Omlette Time!
file:///run/user/ben/gvfs/archive:host=file%25253A%25252F%25252F%
starwa

Whisk!

Whisk all the eggs together. You can add a little milk and a pinch of salt and pepper if you like.

› **We make no guarantees that this method will produce a decent omelette.**


Prinzhorn/skrollr-stylesheets · GitHub - Mozilla Firefox

https://github.com/Prinzhorn/skrollr-stylesheets

Sass

The above is already pretty awesome, but it gets even better. If you know skrollr, you probably heard about the *constants* feature. Let's admit it: it does it's job, but it's ugly.

Sass to the rescue! Using Sass variables and interpolation, things can now look like this:

```
$about_section_begin: 0;
$about_section_duration: 2000;
$about_section_end: $about_section_begin + $about_section_duration;

@-skrollr-keyframes animation1 {
    #{$about_section_begin} {
        left:100%;
        opacity#{"[swing]"}: 0.0;
    }

    #{$about_section_end} {
        left:0%;
        opacity: 1.0;
    }
}
```

mind blown

And of course you can use all the things you already love about Sass as well.

❯ There are only a few websites that take advantage of Skollr, but it's likely to become more prevalent as designers learn to harness its power.

to ensure that the text stays opaque:

```
<div data-500="opacity:0;left:20%;top:10%;transform:rotate(0
deg);" data-699="opacity:1;transform:rotate(720deg);" data-
750="opacity:0">
<h1>Whisk!</h1>
</div>
<div data-500="width:50%;height:50%;left:20%;top:25%;opac
ity:0" data-550="opacity:1" data-700="opacity:1" data-
750="opacity:0" >
Whisk all the eggs together. You can add a little milk and a
pinch of salt and pepper if you like.
</div>
```

As you can see. There's nothing to stop you having two or more elements on the screen at any one time. Here, we've used the **transform:rotate(...)** property to spin the text:

```
<div data-750="opacity:0;" data-800="left:10%;top[cubic]:10%
;opacity:1" data-1000="left:90%;top:90%;opacity:1" data-
1050="opacity:0"> <h1>Pour</h1> </div>

<div data-750="width:50%;height:50%;left:20%;top:25%;opac
ity:0" data-800="opacity:1" data-1000="opacity:1" data-
1050="opacity:0" >
            Pour the eggs into the frying pan.
</div>
```

The key part to this section is the **top[cubic]** property at data point 750 in the first div. As we've said, Skrollr will interpolate between the various data points we supply it with. Usually it uses linear interpolation, but it doesn't have to. You can add other options in square brackets after the property. Quadratic, cubic, swing and bounce are the most useful here.

```
<div class="header" data-1050="opacity:0;left:30%;top:20%"
data-1100="opacity:1;left:40%;top:25%" data-
1150="left:55%;top:35%" data-1200="left:50%;top:45%"
data-1250="left:30%;top:50%" data-1300="left:20%;top:45%"
data-1450="left:15%;top:35%" data-1500="opacity:1;left:20%;
top:25%" data-1550="opacity:0;left:30%;top:20%">
<h1>Stir</h1></div>
```

This section very roughly animates a stirring motion. If you want to get started with the more advanced elements of Skrollr, try to convert this into a perfect circle motion. The method isn't particularly obvious , so take a look at the **circular_motion.html** file in examples for details of how to do it. In fact, now you know the basics, you may want to take a look through all the example files to both see how things can be done and get a bit of inspiration.

There is more to our omelette recipe, but you'll have to take a look at the file for yourself as we've run out of space. It shouldn't be hard to understand. As with all graphical methods, the trick with parallax scrolling is moderation. When used well it can enhance the user experience of a site and make it easier to understand what's going on. When used badly, it just creates a confusing mess. This example may be closer to the latter than the former, but we created it to teach you how to code, not make omelettes. If you want to get some inspiration from sites that use these techniques, check out the following: **http://everylastdrop.co.uk** and **www.cabletv.com/the-walking-dead**. LXF

Add some Sass

We've covered quite a bit of what Skrollr does, and enough to create your own websites, but there's more. Take a look at the examples for plenty of, well, examples. Perhaps the most useful thing that we haven't covered is Skrollr style sheets. These allow you to split up the animation and the contents of the page, and also to reuse code more efficiently. Not only do they allow you to split your code animations into CSS files, but you can use it with Sass (Syntactically Awesome StyleSheets) that significantly eases the creation of complex animations, and allows you to make adjustments without having to re-recreate the entire timing system. More details on Skrollr stylesheets can be found on the GitHub page here: **http://bit.ly/16oHFiZ**.