learnable-content / css-animations Public

lesson3.2 Code

This branch is 5 commits ahead, 4 commits behind master.

6

03/start
CSS_Animations_hando...
README.md

About

No description, website, or topics provided.

Readme
Activity
8 stars
7 watching
9 forks
Report repository

Releases

No releases published

Packages

No packages published

Contributors 2

spxds
donovanh Donovan Hutchinson

Languages

● CSS 93.2% ● HTML 6.8%

README.md

We'll start with a simple example to show how the animation property can be used to show an element on page load. We'll also see a simpler way to write our keyframes.

We have a simple webpage containing a box and some text. When this page loads, the box shows up straight away. Let's see how to use animation to make it fade in instead. You can imagine this might be useful for websites or apps when introducing new content.

Create a set of keyframes:

```
.demo span {
          animation: pop-in .5s 1.4s cubic-bezier(0,.8,.4,1.25) forwards;
  display: inline-block;
          transform: scale(1);
}

@keyframes pop-in {
  0% {
    opacity: 0;
    -webkit-transform: scale(.5);
            transform: scale(.5);
  }
  40% {
    opacity: 1;
  }
  100% {
    opacity: 1;
    -webkit-transform: scale(1);
            transform: scale(1);
  }
}
```

Opening our browser and refresh, we see the animation. Now you may have noticed something here. The box is visible for half a second before the animation starts. This is not ideal. Let's do something about this.

Back in our CSS, let's change the initial state of the element - by default it's visible. Let's instead set the opacity to 0, so that when the page first loads this element is not visible:

```
.demo span {
  opacity: 0;
}
```

We're using opacity to hide the object by default. There may be situations where keyframes aren't supported. Since we're relying on these keyframes to show the content, we might find ourselves in trouble. Well, later I'm going to revisit this and talk about how we can use JavaScript to detect a browser's capability and ensure that the keyframe animation is only applied where it will work.

For now though, let's look at how we can use keyframes for more complex animation. In this example we'll look at how the percentage syntax of keyframes works. To show this, we'll build an animation that uses a series of keyframes. We'll also show how several animations can work together to introduce an element.

```css
@keyframes show-background {
  0% {
    -webkit-transform: translate(-110%, 95%);
            transform: translate(-110%, 95%);
  }
  50% {
    -webkit-transform: translate(0, 95%);
            transform: translate(0, 95%);
  }
  100% {
    -webkit-transform: none;
            transform: none;
  }
}

.demo:before {
  -webkit-animation: show-background 1s .5s cubic-bezier(0,.9,.3,1) forwards;
          animation: show-background 1s .5s cubic-bezier(0,.9,.3,1) forwards;
  background: #3991AE;
  bottom: 0;
  content: "";
  left: 0;
  position: absolute;
  right: 0;
  top: 0;
  -webkit-transform: translate(-110%, 95%);
          transform: translate(-110%, 95%);
}
```

This animation will slide across from the left, and then move up into place.

I'm sure you could think of many other ways to apply this keyframe. Now that we've covered the basics of keyframes and the animation property, let's step back and take a look at timing functions.