Frontend → CSS → Animation → Introduction to animation

Theory: Introduction to animation

18 minutes 0 / 5 problems solved Skip this topic Start practicing

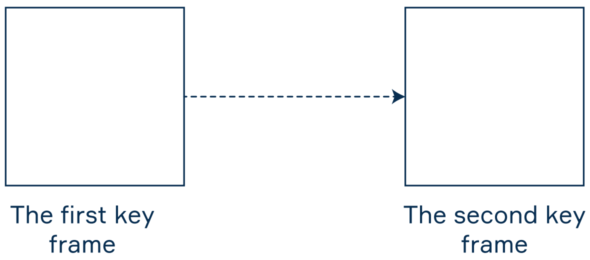
328 users solved this topic. Latest completion was **1 day ago**.

Human brain works in such a way that it pays attention to any movement. Moving elements can improve sites and web applications by making them more interactive for the users. One way to make your web pages more dynamic is adding animation. With CSS, you can create automatically played animation effects that will repeat over a set period of time.

§1. Key frames

In CSS, animation is a smooth transition of an HTML element from one style to another. The animation consists of different frames that gradually replace each other. You can create **key frames**, that is, properties that should be applied to an element at a given time. The calculation of the intermediate values between such frames is performed by the browser.

For example, if you're going to move some element from the left to the right, you'll only need to create two key frames:



§2. The @keyframes rule

Keyframes are the basis of CSS animation. They determine what the animation looks like at each point of the timeline. Key frames are created using the `@keyframes` rule. The syntax of `@keyframes` looks like this:

```
@keyframes animation name {
 0% {
   /* CSS properties */
 }
 100% {
   /* CSS properties */
 }
}
```

Each `@keyframes` consists of the following parts:

- **Animation name:** you can choose any name to describe your animation.
- **Animation steps:** each step can be represented in percentages. 0% represents the initial state of the animation, and a 100% represents its final state; you can create up to 100 steps.
- **CSS properties** can be defined for each step of the timeline.

Consider the following example of animation where the element changes its font size:

```
@keyframes change-font-size {
 0% {
   height: 10%;
 }
 50% {
   height: 30%;
 }
 100% {
   height: 10%;

  }

}
```

Current topic:

Introduction to animation

Topic depends on:

× Absolute units Stage 2

× Relative units Stage 2

Table of contents:

In this example, when the animation is 0% complete, the height of the element is 10%. When the animation is 50% complete, the height is increased to 30%. In the final state, the height is back to 10%.

Instead of 0% and 100%, you can use the keywords `from` and `to`. For instance, the following code does the same thing as the previous:

```
@keyframes change-font-size {
 from {
   height: 10%;
 }
 50% {
   height: 30%;
 }
 to {
   height: 10%;
  }
}
```

> If 0% or 100% are not specified, the user's browser creates them using the initially set values of the animated property.

§3. Applying keyframes to the element

After declaring the `@keyframes` rule, we can apply it to our HTML elements. In order for the animation to start, we need to know two basic CSS properties:

- `animation-name`: the value of this property is the name of the animation as defined in `@keyframes`.
- `animation-duration` is responsible for the duration of the animation. The duration can be specified in seconds (for example, 2s) or milliseconds (200ms).

Let's try to apply the previously created `change-font-size` animation to the `<h1>` element. It will look like this:

```
h1 {
  animation-name: change-font-size;
  animation-duration: 3s;
}
```

Here, our animation will only last 3 seconds. If you want your animation to be repeated infinitely or a certain number of times, the `animation-iteration-count` can help you with that. `animation-iteration-count` indicates how many times an animation cycle is played. You can specify the values of this property:

- `infinite` — the animation is played endlessly.
- any positive number or `0` — the animation is repeated as many times as specified. If the number is not an integer, the animation ends in the middle of the last cycle. A value of `0` triggers the animation instantly. Negative numbers are unacceptable.

> It can be difficult to memorize the names of all the necessary properties. To set your animation, you don't need to write all the necessary properties: it should be enough to know about the general universal property of `animation`. For example, instead of specifying `animation-name`, `animation-duration`, and `animation-iteration-count`, it is enough to write the values for all three properties in random order: `animation: change-font-size 2s 3;`

§4. Crossbrowser

Today, all modern browsers support animation properties. If you want to use animation in earlier browser versions (pre-2017), you need to duplicate the properties and the `@keyframes` rule with the appropriate prefixes: `-webkit-` for versions of Chrome, Safari, Android, `-moz-` for Mozilla Firefox, and `-o-` for Opera.

You often encounter code like this:

```
@-webkit-keyframes font-size-change { /* CSS code */}
@-moz-keyframes font-size-change { /* CSS code */}
@-o-keyframes font-size-change { /* CSS code */}
@keyframes font-size-change { /* CSS code */}

h1 {
  /* -webkit-animation: font-size-change 3s;
  -moz-animation: font-size-change 3s;
  -o-animation: font-size-change 3s;
  animation: font-size-change 3s; */
  -webkit-animation-name: font-size-change;
  -moz-animation-name: font-size-change;
  -o-animation-name: font-size-change;
  animation-name: font-size-change;
  -webkit-animation-duration: 3s;
  -moz-animation-duration: 3s;
  -o-animation-duration: 3s;
  animation-duration: 3s;
}
```

Now you won't be confused by seemingly duplicated code!

§5. Conclusion

CSS animation can be applied to almost all HTML elements. Now you know how to create keyframes, bind the `@keyframes` rule to the elements, and use the universal property of animation. This knowledge will help you create more interactive pages with ease.

Report a typo

37 users liked this theory. **1** didn't like it. **What about you?**

😍 🙂 😐 🙁 😡

Start practicing

Comments (0) Hints (0) Useful links (0) Show discussion