CSS animation

CSS animation is feature of CSS3 which allow animating transitions from one CSS style to another with a use of animation properties applied to a certain html element and a keyframe, which defines the animation itself.

In the following paragraph I am going to describe the main principle of creating basic CSS animations by providing an example for each of the transition properties in action.

Although not every property can be animated so in the following links there is a list of properties that can be animated:
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_animated_properties
http://leaverou.github.io/animatable/

How to create basic css animations

In order to create an animation sequence the element that needs to be animated should be styled with animation property or its sub-properties. There are 8 animation properties which I am going to show in action (for all examples I followed a tutorial available at https://codeburst.io/a-guide-to-css-animation-part-1-8777f5beb1f8).

All the examples are accessible at http://obosa.uk/cm/css-animation/index.html

For each example I am going to animate a div element.

First of all, we need to create a keyframe rule to define our animation. Generally it takes this form:

```
@keyframe<animation-name>{
keyframe selector { css styles }
}
```

Where **animation name** is the name of the animation to which we will refer when applying it to a html-element; **keyframe selector** can either be in percentages (e.g. 0%, 25%, 50%) or a keyword: to, from It selects the period of animation time. Inside the brackets of keyframe selector we specify the **style** that will apply to the element in selected period of animation running time.

Below there is an example of a keyframe that defines the 360deg spin of an element.

```
@keyframes spin {
  from{
    transform: rotate(0deg);
  }
  to{
    transform: rotate(360deg);
  }
}
```

In order to apply this animation to an actual element we need to style it with animation sub-properties (or with an animation property shortcut).
animation-name and **animation-duration** are mandatory properties.

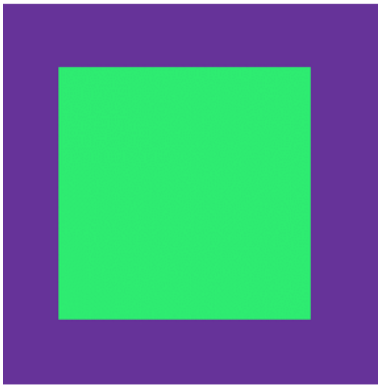
```
div{
  animation-name: spin;
  animation-duration: 2s;
}
```

animation-name

The value defines the animation that should be applied to this element.

animation-duration

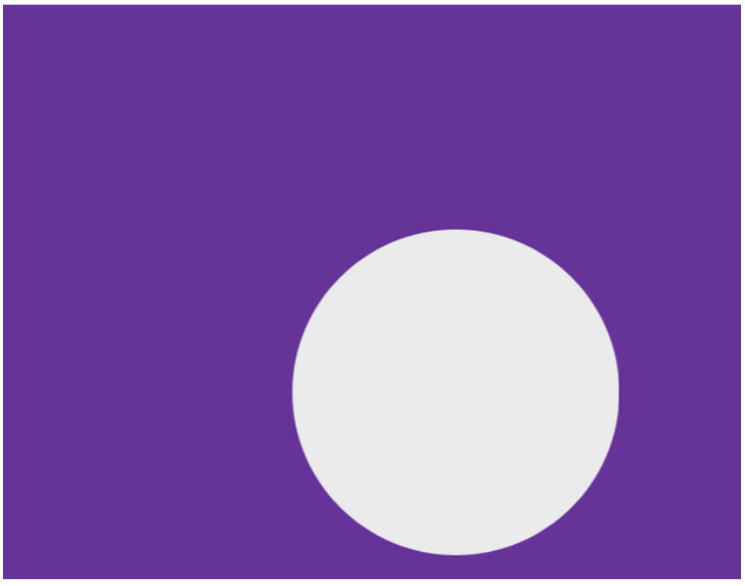
The value defines the animation running time. Could take any *value in seconds* (s) or *milliseconds*(ms), *infinite*, *initial* or *inherit*.



Above you can see the recreated result of defined animation of a spin.
In the spinning example I used to-from keywords as keyframe selector, but we can also use percentages as in the example below.

```
@keyframes squarePath {
0%, 100% {
  transform: translate(40%, 40%);
}
25% {
  transform: translate(-40%, 40%);
}
50% {
  transform: translate(-40%, -40%);
}
75% {
  transform: translate(40%, -40%);
}
}
```

Here I change the position of a div by setting translate with values of x- and y-axis coordinates to a transform property. As in the previous example of a spin, I add **animation-name: squarePath;** and **animation-duration:2s;** to the div. You can see the result I tried to recreate with a GIF below.



animation-iteration-count

The value defines the number of times we want the animation to repeat. Takes the numeral value or the keyword *infinite*.

animation-timing-property

The value defines the speed characteristics of the animation. It can take such values:

- **ease**

Is the default value for animation-timing-property. Starts and ends slowly, but rapids in the middle.

To see this property in action, please check out the live example at http://obosa.uk/cm/css-animation/index.html

- **ease-in**

Starts slowly, but increases the speed to the end.

- **ease-out**

Reduces the speed to the end.

- **ease-in-out**

Starts and ends slowly, rapids in the middle, but the transitions are more smooth then in **ease**.

- **ease-linear**

The same speed through the running time.

- **cubic-bezier**

Allows to define the speed manually. It takes four numeric values from 0 to 1 **cubic-bezier(x_1, y_1, x_2, y_2)**. This values define the speed in four fragments of animation running. Although by setting negative values we can create a bounce effect.

- **steps**

Provides an option to split the animation into equal steps. It can either take one numeral value that sets the number of steps or an optional value for direction: *start* or *end*.

animation-play-state

This property can take either *paused* value to stop the animation from running or *running* value which is the default value.

To show you this property in action I set the **animation-play-state** to *running* for a checkbox in :checked state which controls the animation of the spinning square that I showed earlier. So the animation will be paused by default and will start when the checkbox will be checked.

animation-delay

Defines the amount of time either in seconds(s) or milliseconds(ms) before the animation starts.

animation-fill-mode

Specifies the behaviour of the element when the animation is not running (before, after or both). It can take four values:

- **none**

The default value. The animation will not apply any style to the element before or after the executing.

- **forwards**

The element will retain the style set in the last applied keyframe.

- **backwards**

The element will retain the style set in the first keyframe.

- **both**

The animation will follow the rules for both forwards and backwards, extending the animation properties in both directions.

animation-direction

Allows to define the direction of the animation. It can take four values:

- **alternate**

The animation direction alternates on each iteration.

- **alternate-reverse**

The animation direction alternates on each iteration, starting with reverse.

- **normal**

The default.

- **reverse**

Animation starts running in reverse.

animation shortcut

As many CSS properties which have sub-properties, animation property has a shortcut.
So this

```
div{
animation-name: spin;
animation-duration: .5s;
animation-timing-function: ease-out;
animation-delay: 1s;
animation-iteration-count: infinite;
animation-direction: normal;
animation-fill-mode: none;
animation-play-state: running;
}
```

is equivalent to this

```
div{
animation: spin .5s ease-out 1s infinite normal none running;
}
```

CSS animation vs JS animation

Although JavaScript allows to create more complex animations, there are still cases in which CSS animation is more efficient.
The key advantages of CSS animation over scripts are:

1. Animations run well, even in moderate system load.
2. Animations still available if the user disables JavaScript in their browser.
3. The browser has control over animation sequence so it can optimize performance and efficiency.

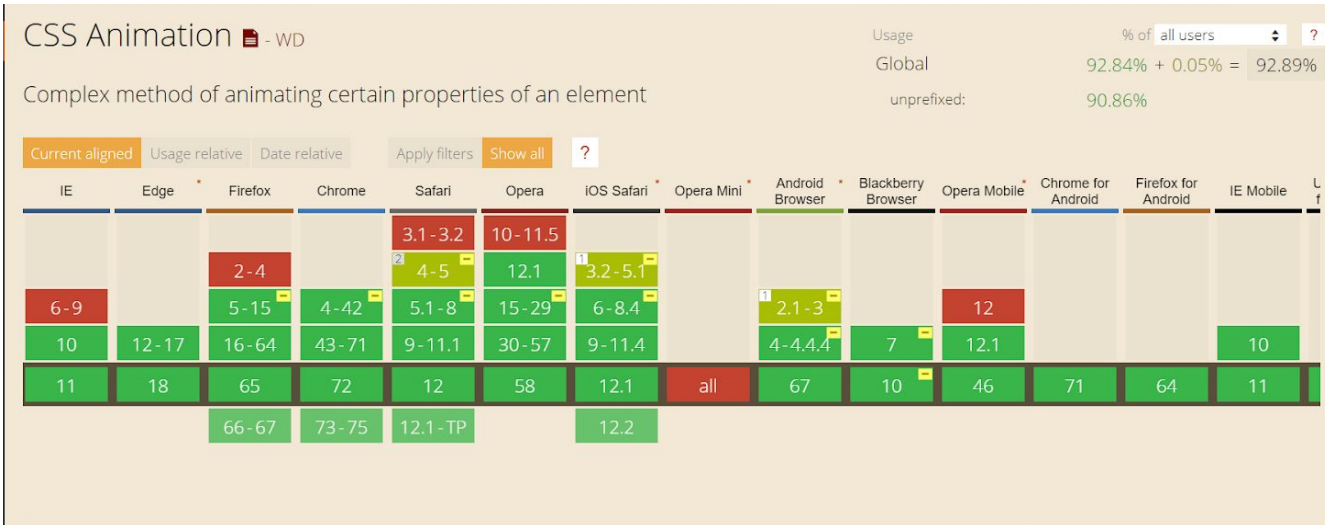
However, CSS animation are still limited and here is a list of examples we can't do in CSS yet:

- animate along a curve;
- animate bounce or rough ease(basically more control over element's movement style);
- use different eases to properties in the keyframe; ease effect applies to the whole keyframe;
- physics-based motion;
- animate attributes;
- animate the scroll position;
- directional rotation(like "animate to exactly 270 degrees in the shortest direction, clockwise or counter-clockwise").

Generally CSS animation is much better solution for simple animations like changing the position of the element or basic rotation and show better performance than JavaScript. Although for more complex and realistic animations with movement scenarios CSS animation is inapplicable.

Browser support

Below you can see a table from https://caniuse.com/#feat=css-animation,css-transitions which shows browser support of CSS animation.
All popular browser support CSS animations except for their old versions and all versions of Opera Mini browser. Generally, 92.84% of browsers support CSS animation.



Sources

https://codeburst.io/a-guide-to-css-animation-part-1-8777f5beb1f8
https://caniuse.com/#feat=css-animation,css-transitions
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Animations/Using_CSS_animations
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_animated_properties
http://leaverou.github.io/animatable/
https://www.tutorialrepublic.com/css-tutorial/css3-animations.php
https://www.html5rocks.com/en/tutorials/speed/high-performance-animations/