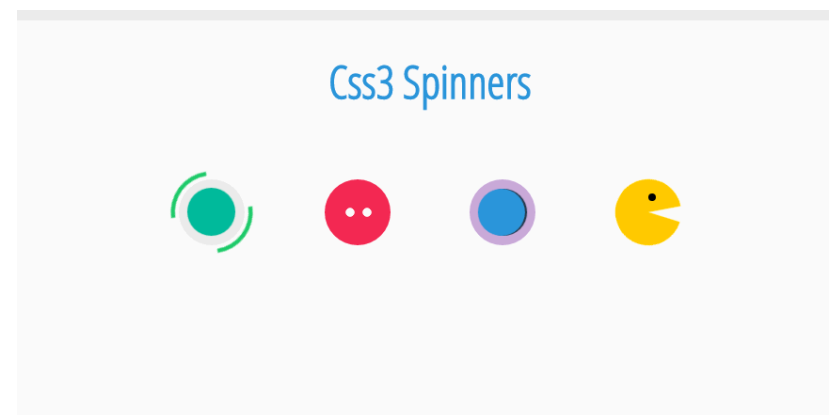


# Lesson - 34

Animation



# Lesson Plan

- HW Review
- Animations

# Animation

CSS allows animation of HTML elements without using JavaScript or Flash!

---

## Css3 Spinners



# Animation

**CSS animations** make it possible to animate transitions from one CSS style configuration to another. Animations consist of two components, a style describing the CSS animation and a set of keyframes that indicate the start and end states of the animation's style, as well as possible intermediate waypoints.

There are three key advantages to CSS animations over traditional script-driven animation techniques:

1. They're easy to use for simple animations; you can create them without even having to know JavaScript.
2. The animations run well, even under moderate system load. Simple animations can often perform poorly in JavaScript (unless they're well made). The rendering engine can use frame-skipping and other techniques to keep the performance as smooth as possible.
3. Letting the browser control the animation sequence lets the browser optimize performance and efficiency by, for example, reducing the update frequency of animations running in tabs that aren't currently visible.
4. An animation lets an element gradually change from one style to another.
5. You can change as many CSS properties you want, as many times you want.
6. To use CSS animation, you must first specify some keyframes for the animation.
7. Keyframes hold what styles the element will have at certain times.




# CSS Animation Properties

The following table lists the @keyframes rule and all the CSS animation properties:

Property	Description
<u>@keyframes</u>	Specifies the animation code
<u>animation</u>	A shorthand property for setting all the animation properties
<u>animation-delay</u>	Specifies a delay for the start of an animation
<u>animation-direction</u>	Specifies whether an animation should be played forwards, backwards or in alternate cycles
<u>animation-duration</u>	Specifies how long time an animation should take to complete one cycle
<u>animation-fill-mode</u>	Specifies a style for the element when the animation is not playing (before it starts, after it ends, or both)
<u>animation-iteration-count</u>	Specifies the number of times an animation should be played
<u>animation-name</u>	Specifies the name of the @keyframes animation
<u>animation-play-state</u>	Specifies whether the animation is running or paused
<u>animation-timing-function</u>	Specifies the speed curve of the animation

# Browser support

The numbers in the table specify the first browser version that fully supports the property.

Property					
@keyframes	43.0	10.0	16.0	9.0	30.0
animation-name	43.0	10.0	16.0	9.0	30.0
animation-duration	43.0	10.0	16.0	9.0	30.0
animation-delay	43.0	10.0	16.0	9.0	30.0
animation-iteration-count	43.0	10.0	16.0	9.0	30.0
animation-direction	43.0	10.0	16.0	9.0	30.0
animation-timing-function	43.0	10.0	16.0	9.0	30.0
animation-fill-mode	43.0	10.0	16.0	9.0	30.0
animation	43.0	10.0	16.0	9.0	30.0

# The @keyframe rule

When you specify CSS styles inside the `@keyframes` rule, the animation will gradually change from the current style to the new style at certain times.

To get an animation to work, you must bind the animation to an element.

The following example binds the "example" animation to the `<div>` element. The animation will last for 4 seconds, and it will gradually change the background-color of the `<div>` element from "red" to "yellow":

```
/* The animation code */
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

# The @keyframe rule

**Note:** The `animation-duration` property defines how long time an animation should take to complete. If the `animation-duration` property is not specified, no animation will occur, because the default value is 0s (0 seconds).

In the example above we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).

It is also possible to use percent. By using percent, you can add as many style changes as you like.

The following example will change the background-color of the `<div>` element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

```
/* The animation code */
@keyframes example {
  0%   {background-color: red;}
  25%  {background-color: yellow;}
  50%  {background-color: blue;}
  100% {background-color: green;}
}


/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```



# Delay the animation

The **animation-delay** property specifies a delay for the start of an animation.

The following example has a 2 seconds delay before starting the animation:




```
div {  
  width: 100px;  
  height: 100px;  
  position: relative;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
  animation-delay: 2s;  
}
```

# Delay the animation

Negative values are also allowed. If using negative values, the animation will start as if it had already been playing for  $N$  seconds.

In the following example, the animation will start as if it had already been playing for 2 seconds:



```
div {  
  width: 100px;  
  height: 100px;  
  position: relative;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
  animation-delay: -2s;  
}
```

# Number of times animation run

The `animation-iteration-count` property specifies the number of times an animation should run.

The following example will run the animation 3 times before it stops:



```
div {  
  width: 100px;  
  height: 100px;  
  position: relative;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
  animation-iteration-count: 3;  
}
```

# Number of times animation run

The following example uses the value "infinite" to make the animation continue for ever:



```
div {  
  width: 100px;  
  height: 100px;  
  position: relative;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
  animation-iteration-count: infinite;  
}
```

# Run Animation in Reverse Direction or Alternate Cycles

The **animation-direction** property specifies whether an animation should be played forwards, backwards or in alternate cycles.

The animation-direction property can have the following values:

- **normal** - The animation is played as normal (forwards). This is default
- **reverse** - The animation is played in reverse direction (backwards)
- **alternate** - The animation is played forwards first, then backwards
- **alternate-reverse** - The animation is played backwards first, then forwards

The following example will run the animation in reverse direction (backwards):

```
div {  
  width: 100px;  
  height: 100px;  
  position: relative;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
  animation-direction: reverse;  
}
```


# Specify the Speed Curve of the Animation

The **animation-timing-function** property specifies the speed curve of the animation.

The animation-timing-function property can have the following values:

- **ease** - Specifies an animation with a slow start, then fast, then end slowly (this is default)
- **linear** - Specifies an animation with the same speed from start to end
- **ease-in** - Specifies an animation with a slow start
- **ease-out** - Specifies an animation with a slow end
- **ease-in-out** - Specifies an animation with a slow start and end
- **cubic-bezier(n,n,n,n)** - Lets you define your own values in a cubic-bezier function

The following example shows the some of the different speed curves that can be used:



```
#div1 {animation-timing-function: linear;}
#div2 {animation-timing-function: ease;}
#div3 {animation-timing-function: ease-in;}
#div4 {animation-timing-function: ease-out;}
#div5 {animation-timing-function: ease-in-out;}
```

# Animation Shorthand Property

The example below uses six of the animation properties:

```
div {  
  animation-name: example;  
  animation-duration: 5s;  
  animation-timing-function: linear;  
  animation-delay: 2s;  
  animation-iteration-count: infinite;  
  animation-direction: alternate;  
}
```

The same animation effect as above can be achieved by using the shorthand `animation` property:

```
div {  
  animation: example 5s linear 2s infinite alternate;  
}
```

# Home work

Visit this link to find lesson homework:

<https://docs.google.com/document/d/1DjHA57xV3AiSd-lycKfBWCg5V9YR69e6hfdj07MuMo8/edit?usp=sharing>



# Learning Resources

## 1. Animations

1. <https://yalantis.com/blog/web-animation-technologies-and-tools/>
2. Example of simple animations: <https://www.30secondsofcode.org/css/t/animation/p/1>
3. [https://www.w3schools.com/css/css3\\_animations.asp](https://www.w3schools.com/css/css3_animations.asp)
4. Array.reduce [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Array/Reduce](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce)

## 2. Webpack Source Map: <https://webpack.js.org/configuration/devtool/>

3. Source Map: <https://blog.teamtreehouse.com/introduction-source-maps>