Graphics and animations in CSS

In this tutorial you will learn how to use CSS to transform HTML structure elements, perform transitions and animations, and how to use filters and gradients.

Spis treści

Introduction to CSS animation 1/1

1.1. Introduction to animation

This course aims to teach you how to create CSS animations and graphic effects while maintaining good practices and compatibility with various browsers.

We will get to know deeply the principles of operation of CSS transitions, 2D and 3D transformation, and also an extensive lesson about filters and gradients.

A large part of the course are also CSS animations in which we show from scratch how to build great animations without a lot of work..

WORKS Just go to the next task:)

```
<!DOCTYPE html>
<html>
<head>
   <title>Fiters and prefixes</title>
   <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
   <div class="loader">
      <div class="one"></div>
      <div class="two"></div>
      <div class="three"></div>
   </div>
</body>
</html>
```

```
* {
  margin: 0;
  padding: 0;
  border: 0;
  box-sizing: border-box;
}

html,
body {
  height: 100%;
}

body {
  background-color: #353D3E;
}

.loader {
  width: 200px;
  height: 200px;
  position: absolute;
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  margin: auto;
}
.loader > div {
  top: 0;
  left: 0;
  right: 0;
  bottom: 0;
  margin: auto;
  transform: rotate(45deg);
  border-top-color: #55ee70;
  animation-delay: .25s;
  border-radius: 50%;
  position: absolute;
  animation: spin 2s infinite;
}
.loader .one {
  width: 200px;
  height: 200px;
  transform: rotate(45deg);
  border: 20px solid transparent;
  border-top-color: #55ee70;
  animation-delay: 0s;
}
.loader .two {
  width: 133.33333px;
  height: 133.33333px;
  border: 20px solid rgba(255, 255, 255, 0.5);
  border-top-color: #55ee70;
  animation-delay: 0.25s;
}
.loader .three {
  width: 66.66667px;
  height: 66.66667px;
  border: 20px solid rgba(255, 255, 255, 0.5);
  border-top-color: #55ee70;
  animation-delay: .5s;
}

@keyframes spin {
  50%,
            100% {
```

```
    transform: rotate(405deg);
  }
}
```

2.1. What will change the transition - transition-property

In this part of the course we will learn how to use transitions - it is a simple way to animate changes in various CSS properties.

Let's start with transition-property. This is the basic property of transitions that determines which of the changed CSS values should be animated.

We can see the perfect example of operation in the preview window. If we set several properties in our element and we want all of them to be changed during transformation, we can omit the transition-property property.

However, if we want only the selected properties to be changed, then we must add them as values for transition-property (values separated by commas), eg:

transition-property: width, height.

However, before we see the smooth effect of our transition, we need to learn one more property - transition-duration, which we will discuss in the next exercise :)

WORKS

1. 1. For the element in the .practice class, add the transition-property property.
2. 2. transition-property should have the following properties: width, height, margin-left.

HINT

transition-property should contain: width, height, margin-left.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Transition basics</title>
    <link rel="stylesheet" type="text/css"
href="style.css">
  </head>
  <body>
    <div class="hero">
      <h1>Transition basics</h1>
    </div>
    <div class="container">
      <div class="example">
        <h2>Example:</h2>
        <p>With <b>transition-property</b> for
margin, color and background.</p>
        <div></div>
        <p>With <b>transition-property</b>
only for margin (other changes apply instantly,
<b>transition-duration</b> is ignored for
them).</p>
        <div></div>
        <p>Without any <b>transition-
property</b>.</p>
        <div></div>
      </div>
      <div class="exercise">
        <h2>Exercise:</h2>
        <div class="practice" >
          <p>Change me!</p>
        </div>
      </div>
    </div>
  </body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=Ope
n+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/13.jpg') center;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
```

```
    text-align: justify;
    background-color: white;
    padding: 20px;
    box-shadow: 2px 2px 10px #000;
    text-align: center;
}
.example,
.exercise {
    padding-bottom: 20px;
}
.example > div {
    height: 50px;
    width: 50px;
    background-color: #e74c3c;
    border-radius: 8px;
    transition-duration: 5s;
}
.example > div:nth-of-type(1) {
    transition-property: margin-left, width,
background-color;
}
.example > div:nth-of-type(2) {
    transition-property: margin-left;
}
.example:hover > div {
    background-color: blue;
    margin-left: 50%;
    width: 100px;
}


/*EXERCISE*/

.practice {
    border-radius: 5px;
    width: 80px;
    height: 80px;
    background-color: #16a085;
}
.practice > p {
    color: #fff;
    padding-top: 15px;
}
.exercise:hover > .practice {
    width: 100px;
    height: 100px;
    margin-left: 50%;
    background-color: orange;
}


/*EXERCISE*/

.practice {
    border-radius: 5px;
    width: 80px;
    height: 80px;
    background-color: #16a085;
    transition-property: width, height, margin-left;
}
.practice > p {
    color: #fff;
    padding-top: 15px;
}
.exercise:hover > .practice {
    width: 100px;
    height: 100px;
    margin-left: 50%;
    background-color: orange;
}
```

2.2. Duration of transitions - transition-duration

In the previous exercise, we got to know transition-property - now let's take a transition-duration. The value given for this property is simply the time in which the whole transformation takes place.

This time is usually determined in seconds, so if we specify transition-duration: 0.5s, this transformation will take place in half a second. We can also use milliseconds (and here, instead of 0.5s, we would write 500ms).

Sounds a bit complicated? It's just a semblance, hover, for example, in the preview window and see for yourself how easy it is!!

WORKS

1. 1. For the element in the .practice class, add the transition-duration property
2. 2. transition-duration should be 2s..

HINT

The code for our selector for .practice should contain the following properties: transition-duration: 2s; transition-property: width, height, margin-left;

```
<!DOCTYPE html>
<html>
  <head>
    <title>Transition basics</title>
    <link rel="stylesheet" type="text/css"
href="style.css">
  </head>
  <body>
    <div class="hero">
      <h1>Transition basics</h1>
    </div>
    <div class="container">
      <div class="example">
        <h2>Example:</h2>
        <p>With <b>transition-property</b> for
margin, color and background.</p>
        <div></div>
        <p>With <b>transition-property</b>
only for margin (other changes apply instantly,
<b>transition-duration</b> is ignored for
them).</p>
        <div></div>
        <p>Without any <b>transition-
property</b>.</p>
        <div></div>
      </div>
      <div class="exercise">
        <h2>Exercise:</h2>
        <div class="practice" >
          <p>Change me!</p>
        </div>
      </div>
    </div>
  </body>
</html>

@import
url(https://fonts.googleapis.com/css?family=Ope
n+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/13.jpg') center;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  text-align: justify;
  background-color: white;
  padding: 20px;
  box-shadow: 2px 2px 10px #000;
  text-align: center;
}
.example,
.exercise {
```

```
   padding-bottom: 20px;
}
.example > div {
   height: 50px;
   width: 50px;
   background-color: #e74c3c;
   border-radius: 8px;
   transition-duration: 5s;
}
.example > div:nth-of-type(1) {
   transition-property: margin-left, width,
background-color;
}
.example > div:nth-of-type(2) {
   transition-property: margin-left;
}
.example:hover > div {
   background-color: blue;
   margin-left: 50%;
   width: 100px;
}

/*EXERCISE*/

.practice {
   border-radius: 5px;
   width: 80px;
   height: 80px;
   background-color: #16a085;
   transition-property: width, height, margin-left;
}
.practice > p {
   color: #fff;
   padding-top: 15px;
}
.exercise:hover > .practice {
   width: 100px;
   height: 100px;
   margin-left: 50%;
   background-color: orange;
}

/*EXERCISE*/

.practice {
   border-radius: 5px;
   width: 80px;
   height: 80px;
   background-color: #16a085;
   transition-property: width, height, margin-left;
   transition-duration: 2s;
}
.practice > p {
   color: #fff;
   padding-top: 15px;
}
.exercise:hover > .practice {
   width: 100px;
   height: 100px;
   margin-left: 50%;
   background-color: orange;
}
```

2.3. The tempo of transition, ie transition-timing-function

Another interesting feature to be mentioned is the transition-timing-function. Its task is to determine the speed of change occurrence at given stages (eg start, middle and end). This property does not affect the length of transition (duration) in any way.

Let's move on to a more thorough understanding of the properties. Its default value is ease (quick start and slow end), the other popular properties are:
linear (constant tempo)
ease-in-out (slow start and end)
cubic-bezier (x, x, x, x) (where x is a number from 0 to 1) (any function created by us - this value is not so often used, but it's good to know that we have such a possibility).

WORKS

1. Create a rule with the .exercise: hover> .practice-linear selector and give it the transition-timing-function: linear property.
2. Next, create a rule with the .exercise: hover> .practice-ease selector.
3. Give it the transition-timing-function property: ease-in; .

HINT

The correct code for this exercise is as follows:

```
.exercise:hover > .practice-linear {
transition-timing-function: linear;
}
.exercise:hover > .practice-ease {
transition-timing-function: ease-in;
}
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>transition-property and transition-duration</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <div class="hero">
      <h1>transition-timing-function</h1>
    </div>
    <div class="container">
      <div class="example">
        <h2>Example:</h2>
        <p><b>ease</b> timing function.</p>
        <div></div>
        <p><b>linear</b> timing function.</p>
        <div></div>
        <p><b>ease-in-out</b> timing function.</p>
        <div></div>
        <p><b>cubic-bezier(x,x,x,x)</b>(where <b>x</b> is number between 0 and 1) timing function.</p>
        <div></div>
      </div>
      <div class="exercise">
        <h2>Exercise:</h2>
        <div class="practice-linear"></div>
        <div class="practice-ease"></div>
      </div>
    </div>
  </body>
</html>
```

```
@import url(https://fonts.googleapis.com/css?family=Open+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/13.jpg') center;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
```

```
   margin: 20px auto;
   text-align: justify;
   background-color: white;
   padding: 20px;
   box-shadow: 2px 2px 10px #000;
   text-align: center;
}
.example,
.exercise {
   padding-bottom: 20px;
}
.example > div {
   height: 50px;
   width: 50px;
   background-color: #e74c3c;
   border-radius: 8px;
   transition-duration: 2s;
}
.example:hover > div {
   margin-left: 50%;
}
.example > div:nth-of-type(2) {
   transition-timing-function: linear;
}
.example > div:nth-of-type(3) {
   transition-timing-function: ease-in-out;
}
.example > div:nth-of-type(4) {
   transition-timing-function: cubic-
bezier(1,0.5,1,0.5);
}
.example:hover > div {
  margin-left: 40%;
}

/*EXERCISE*/

.exercise > div {
   border-radius: 5px;
   margin-top: 10px;
   width: 80px;
   height: 80px;
   background-color: #16a085;
   transition-property: width, height, margin-left;
   transition-duration: 2s;
}
.exercise:hover > div {
   margin-left: 40%;
   width: 100px;
   height: 100px;
}

/*EXERCISE*/

.exercise > div {
   border-radius: 5px;
   margin-top: 10px;
   width: 80px;
   height: 80px;
   background-color: #16a085;
   transition-property: width, height, margin-left;
   transition-duration: 2s;
}
.exercise:hover > div {
   margin-left: 40%;
   width: 100px;
   height: 100px;
}

.exercise:hover > .practice-linear
{
  transition-timing-function: linear;
}

.exercise:hover > .practice-ease
{
   transition-timing-function: ease-in;
}
```

2.4. Delay transition - transition-delay

The final property of the transitions we will learn will be transition-delay.

As the name suggests, this is a delay in the transformation. In addition, it is important that the transformation delay time does not affect the length of the transformation in any way.

The units we use to determine the transition-delay are seconds, so an example: transition-delay: 3s will delay the start of the transformation by 3 seconds.

You will see more examples when you hover in the preview window in the example part..

WORKS

1. Check the contents of the style.css file, you will find three rules with no properties in it.
2. Set the first element of the transition delay time to half a second.
3. The second one second.
4. The third three-quarters of a second.

HINT

Rules should have successive properties of transition-delay properties:
0.5s
1s
0.75s

```
<!DOCTYPE html>
<html>
  <head>
    <title>transition-property and transition-duration</title>
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>
  <body>
    <div class="hero">
      <h1>transition-delay</h1>
    </div>
    <div class="container">
      <h2>Example:</h2>
      <div class="example">
        <p>default (0s delay) value</p>
        <div></div>
        <p>1s delay</p>
        <div></div>
        <p>2s delay</p>
        <div></div>
      </div>
      <h2>Exercise:</h2>
      <div class="exercise">
        <div></div>
        <div></div>
        <div></div>
      </div>
    </div>
  </body>
</html>
```

```
@import url(https://fonts.googleapis.com/css?family=Open+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/13.jpg') center;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  text-align: justify;
  background-color: white;
  padding: 20px;
  box-shadow: 2px 2px 10px #000;
  text-align: center;
}
.example,
.exercise {
  padding-bottom: 20px;
```

```
}
.example > div {
   height: 50px;
   width: 50px;
   background-color: #e74c3c;
   border-radius: 8px;
   transition-duration: 2s;
}
.example > div:nth-of-type(2) {
  transition-delay: 1s;
}
.example > div:nth-of-type(3) {
  transition-delay: 2s;
}
.example:hover > div {
  margin-left: 40%;
}

/*EXERCISE*/

.exercise > div {
   border-radius: 5px;
   margin-top: 10px;
   width: 80px;
   height: 80px;
   background-color: #16a085;
   transition-property: width, height, margin-left;
   transition-duration: 2s;
}
.exercise:hover > div {
   margin-left: 40%;
}
.exercise:hover > div:nth-of-type(1) {

}
.exercise:hover > div:nth-of-type(2) {

}
.exercise:hover > div:nth-of-type(3) {

}

/*EXERCISE*/

.exercise > div {
   border-radius: 5px;
   margin-top: 10px;
   width: 80px;
   height: 80px;
   background-color: #16a085;
   transition-property: width, height, margin-left;
   transition-duration: 2s;
}
.exercise:hover > div {
   margin-left: 40%;
}
.exercise:hover > div:nth-of-type(1) {
transition-delay: 0.5s;
}
.exercise:hover > div:nth-of-type(2) {
transition-delay: 1s;
}
.exercise:hover > div:nth-of-type(3) {
transition-delay: 0.75s;
}
```

2.5. All in one - transition

You already know what are the properties associated with changes, so let's try to group them now with one transition property.

We need the transition-property and transition-duration for the basic operation of the property, so collecting everything in one property we get:
transition: all 1s;

A more complex property could look like this:
transition: all 1s linear 2s

When creating such properties, we must take the following pattern:
transition: property duration timing-function delay

Let's go a step further and let's create an even more extensive property:
margin-left 1s, color 1.5s, width 2s

The transition property simply speeds up coding - it's easier to write one short property with several values than a few long properties, right?:)

WORKS

1. 1. In the rule for the .exercise: hover> .practice rule, create the transition property.
2. 2. And give it margin-left 1s linear.
3. 3. After the decimal place: background-color 1s ease-in-out.

HINT

The entire property should look like this:
transition: margin-left 1s linear, background-color 1s ease-in-out

```
<!DOCTYPE html>
<html>
  <head>
    <title>Grouping transition properties</title>
    <link rel="stylesheet" type="text/css"
href="style.css">
  </head>
  <body>
    <div class="hero">
      <h1>Grouping transition properties</h1>
    </div>
    <div class="container">
      <h2>Example:</h2>
      <div class="example">
        <p>Element using single
<b>transition</b> property:</p>
        <div></div>
      </div>
      <h2>Exercise:</h2>
      <div class="exercise">
        <div class="practice"></div>
      </div>
    </div>
  </body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=Ope
n+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/13.jpg') center;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  text-align: justify;
  background-color: white;
  padding: 20px;
  box-shadow: 2px 2px 10px #000;
  text-align: center;
}
.example,
.exercise {
```

```
    padding-bottom: 20px;
}
.example > div {
    height: 50px;
    width: 50px;
    background-color: #e74c3c;
    border-radius: 8px;
    transition: all 1s ease-in 0.5s;
}
.example:hover > div {
  margin-left: 40%;
}

/*EXERCISE*/

.exercise > div {
    border-radius: 5px;
    margin-top: 10px;
    width: 80px;
    height: 80px;
    background-color: #16a085;
}
.exercise:hover > div {
    margin-left: 40%;
    background-color: red;
}
.exercise:hover > .practice {

}

/*EXERCISE*/

.exercise > div {
    border-radius: 5px;
    margin-top: 10px;
    width: 80px;
    height: 80px;
    background-color: #16a085;
}
.exercise:hover > div {
    margin-left: 40%;
    background-color: red;
}
.exercise:hover > .practice {
    transition: margin-left 1s linear, background-
color 1s ease-in-out;
}
```

2.6. Button with special effects

Since we have already learned the basics of CSS transformation properties, we can move to practice.

We will start slowly and along with the progress of the course, we will create more and more advanced animations and transformations.

At the beginning, we will give the button element a unique effect after hovering the cursor and then determine which properties should change and at what pace.

WORKS

1. 1. For the element with the classmractice-button that is in the state: hover, add the padding property: 30px.
2. 2. For the same element in the state: hover, add another property, this time: font-size with a value of 120%.
3. 3. For the practice-button element, add a property that specifies the conversion length to 0.5s.
4. 4. Next, add the transition-property property and give it its padding, font-size properties.
5. 5. Hover over the button and see what changes are taking place!!

HINT

The length of the conversion is determined by the transition-duration property.
The second property of the practice-button should look like the following: transition-property: padding, font-size

```
<!DOCTYPE html>
<html>
  <head>
    <title>Button with sepcial effects</title>
    <link rel="stylesheet" type="text/css"
href="style.css">
  </head>
  <body>
    <div class="hero">
      <h1>Button with special effects</h1>
    </div>
    <div class="container">
      <div class="example">
        <h2>Example:</h2>
        <button>button 1</button>
        <button>button 2</button>
        <button>button 3</button>
      </div>
      <div class="exercise">
        <h2>Exercise:</h2>
        <button class="practice-button">Put
mouse over me!</button>
      </div>
    </div>
  </body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=Ope
n+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/13.jpg') center;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  background-color: white;
  padding: 20px;
  box-shadow: 2px 2px 10px #000;
  text-align: center;
}
.example,
.exercise {
  padding-bottom: 20px;
```

```
}
.example button {
   padding: 20px;
   border: none;
   outline: none;
   background-color: #2980b9;
   color: #fff;
   text-transform: uppercase;
   cursor: pointer;
   transition-duration: 0.5s;
}
.example button:nth-of-type(1):hover {
   font-size: 120%;
}
.example button:nth-of-type(2):hover {
   background-color: green;
}
.example button:nth-of-type(3):hover {
   padding: 25px;
   font-size: 120%;
   background-color: green;
}

/*EXERCISE*/

.practice-button {
   cursor: pointer;
   padding: 20px;
   outline: none;
   border: 2px solid #8e44ad;
   text-transform: uppercase;
   color: #8e44ad;
   font-weight: bold;
   background-color: transparent;
}
.practice-button:hover {
   background-color: #8e44ad;
   color: #fff;
}


/*EXERCISE*/

.practice-button {
   cursor: pointer;
   padding: 20px;
   outline: none;
   border: 2px solid #8e44ad;
   text-transform: uppercase;
   color: #8e44ad;
   font-weight: bold;
   background-color: transparent;
   transition-duration: 0.5s;
   transition-property: padding, font-size;
}
.practice-button:hover {
   background-color: #8e44ad;
   color: #fff;
   padding: 30px;
   font-size: 120%;
}
```

3.1. translate()

Since you already know the basics of going through CSS, it's time to get to know the properties that can be combined with them.

We will start by discussing the values of the transform property with a graceful name of translate ().

translate (x, y) is used to move the element according to the given parameters in brackets (x - for the X axis, horizontal axis and y - for the Y axis, vertical axis).

For example, let's say we want to move an element 100px to the right and 100px down - our code could look like this: transform: translate (100px, 100px)

Remember that the negative values on the X axis will move the element to the left and the negative values on the Y axis up.

WORKS

1. 1. In the selector for .practice-button, add the transition property with the value .5s all.
2. 2. Create the .practice-button: hover selector and add the transform property in it with translate value (30px, 20px).

HINT

The right code for the .practice-button with pseudo-class: hover looks like this:

```
.practice-button:hover {
transform: translate(30px, 20px);
}
```

```
<!DOCTYPE html>
<html>

<head>
  <title>translate()</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>

<body>
  <div class="hero">
    <h1>translate()</h1>
  </div>
  <div class="container">

    <div class="example">
      <h2>Example:</h2>
      <p>Hover your mouse over the button and
see the <b>translate()</b> property
working!</p>
      <button>button</button>
    </div>

    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <button class="practice-button">Practice
button</button>
    </div>

  </div>
</body>

</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=Ope
n+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/4.jpg') bottom;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  background-color: white;
```

```
    padding: 20px;
    box-shadow: 2px 2px 10px #000;
    text-align: center;
}
.example,
.exercise {
    padding-bottom: 20px;
}
.example button {
    padding: 20px;
    margin: 10px;
    border: none;
    outline: none;
    background-color:#d35400;
    width: 40%;
    color: #fff;
    text-transform: uppercase;
    cursor: pointer;
    transition: transform 2s;
}
.example button:hover {
    background-color: #e67e22;
    transform: translate(100px, 100px);
}

/*EXERCISE*/

.practice-button {
    border: none;
    padding: 20px;
    outline: none;
    background-color: #f39c12;
    color: #fff;
    text-transform: uppercase;
    transition: .5s all;
}

.practice-button:hover
{
    transform: translate(30px, 20px);
}

/*EXERCISE*/

.practice-button {
    border: none;
    padding: 20px;
    outline: none;
    background-color: #f39c12;
    color: #fff;
    text-transform: uppercase;
    transition: .5s all;
}

.practice-button:hover
{
    transform: translate(30px, 20px);
}
```

3.2. rotate()

Kolejną wartością właściwości transform, którą poznamy będzie rotate.

wygląda następująco: transform: rotate(Xdeg) (gdzie X to liczba stopni o które obracamy element, a deg, to skrót od angielskiego **degree**, czyli stopień).

Przykładowa właściwość może wyglądać następująco:
transform: rotate(90deg)
Obróci ona obiekt o 90 stopni w kierunku ruchu wskazówek zegara.

Dopuszczalnymi liczbami są również liczby ujemne np:
transform: rotate(-45deg)
Tak zbudowana właściwość obróci element o 45 stopni w kierunku przeciwnym do ruchu wskazówek zegara.

WORKS

1. Elementowi o klasie clockwise przypisz właściwość nadającą rotację o 5 stopni .
2. Elementowi o klasie counter-clockwisenadaj właściwość tranform o wartości rotate(-5deg).

HINT

Element .clockwise powinien posiadać właściwość transform: rotate(5deg).

```
<!DOCTYPE html>
<html>
<head>
  <title>rotate()</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>rotate()</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>Hover your mouse over the button
and see <b>rotate()</b> working!</p>
```

Jak pewnie domyślasz się, powoduje ona obrót elementu o ilość stopni podaną w parametrze.

Budowa tej wartości

```
        <button>THE button</button>
      </div>
      <div class="exercise">
        <h2>Exercise:</h2>
        <p>Practice!</p>
        <div class="clockwise">clockwise-
rotation</div>
        <div class="counter-
clockwise">counter-clockwise-rotation</div>
      </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/4.jpg')
bottom;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  background-color: white;
  padding: 20px;
  box-shadow: 2px 2px 10px #000;
  text-align: center;
```

```
}
.example,
.exercise {
   padding-bottom: 20px;
}
.example button {
   padding: 20px;
   margin: 10px;
   border: none;
   outline: none;
   border: 2px solid grey;
   width: 40%;
   color: grey;
   background-color: #fff;
   text-transform: uppercase;
   cursor: pointer;
   transition: transform 1s;
}
.example button:hover {
   background-color: grey;
   color: #fff;
   transform: rotate(10deg);
}

/*EXERCISE*/

.exercise div {
        color: #fff;
        width: 300px;
        padding: 10px;
        margin: 30px auto;
        background-color: #e74c3c;
}

/*EXERCISE*/

.exercise div {
        color: #fff;
        width: 300px;
        padding: 10px;
        margin: 30px auto;
        background-color: #e74c3c;
}

.clockwise
{
   transform: rotate(5deg);
}

.counter-clockwise
{
   transform: rotate(-5deg);
}
```

3.3. scale()

Since you already know how to move and rotate objects, let's get to scaling them.

To do this, we will use the transform: scale (x) property (where x is a non-negative number).

Suppose we want to enlarge our item more than twice, so we'll use the properties:
transform: scale (2.5)

The important information is that if you scale the object down (let's say you want to reduce it by half), it's worth knowing that although it will decrease to the expected size, it will take the same amount of space (there will be an empty space around it that will not occupy other elements).

WORKS

1. 1. For a scale-big element, give it a double magnification property.
2. 2. For a scale-small element, add a property that will be halved.
3. 3. Check how the space around the reduced element behaves.

HINT

The scale-small element should have the property:
transform: scale(0.5)

```
<!DOCTYPE html>
<html>
<head>
  <title>scale()</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>scale()</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>Put your mouse over button and
see <b>scale()</b> working!</p>
      <button>button</button>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="scale-big">scale-big</div>
      <div class="scale-small">scale-
small</div>
    </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/4.jpg')
bottom;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  background-color: white;
  padding: 20px;
  box-shadow: 2px 2px 10px #000;
  text-align: center;
}
.example,
.exercise {
  padding-bottom: 20px;
}
.example button {
```

```
    padding: 20px;
    margin: 10px;
    border: none;
    outline: none;
    background-color: grey;
    width: 40%;
    color: #fff;
    text-transform: uppercase;
    cursor: pointer;
    transition: transform 1s;
}
.example button:nth-of-type(1):hover {
    background-color: #95a5a6;
    color: #fff;
    transform: scale(1.2);
}
.example button:nth-of-type(2):hover {
    background-color: #95a5a6;
    color: #fff;
    transform: scale(0.8);
}

/*EXERCISE*/

.exercise div {
        color: #fff;
        width: 50%;
        padding: 10px;
        margin: 30px auto;
        background-color: #e74c3c;
}

/*EXERCISE*/

.exercise div {
        color: #fff;
        width: 50%;
        padding: 10px;
        margin: 30px auto;
        background-color: #e74c3c;
}

.scale-big
{
    transform: scale(2);
}

.scale-small
{
    transform: scale(0.5);
}
```

3.4. skewX()

skewX () skews the element along the X (horizontal) axis.

As in the case of rotating, the element is skewed using steps.

For example, if you want to skew a 20-degree element along the X-axis, we'll use the following property:
transform: skewX (20deg)

WORKS

1. 1. For a skew-light element, add a property that skews the element along the X axis by 20 degrees.
2. 2. The skew-medium element gives the skewX property a value of 50deg.
3. 3. Add a skewX property of 70 degrees to the skew-hard element.
4. 4. Compare the elements and check the difference.

HINT

Elements with skew-light, skew-medium and skew-hard classes should have properties as appropriate:
transform: skewX(20deg)
transform: skewX(50deg)
transform: skewX(70deg)

```
<!DOCTYPE html>
<html>
<head>
  <title>skewX()</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>skewX()</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>Put your mouse over button and
see <b>skewX()</b> working!</p>
      <button>button</button>
      <button>button</button>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="skew-light">skew-
light</div>
      <div class="skew-medium">skew-
medium</div>
      <div class="skew-hard">skew-
hard</div>
    </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/4.jpg')
bottom;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  background-color: white;
  padding: 20px;
  box-shadow: 2px 2px 10px #000;
  text-align: center;
}
.example,
.exercise {
  padding-bottom: 20px;
}
.example button {
  padding: 20px;
```

```
   margin: 10px;                                    }
   border: none;
   outline: none;
   background-color: #27ae60;
   width: 40%;
   color: #fff;
   text-transform: uppercase;
   cursor: pointer;
   transition: transform 1s;
}
.example button:hover {
        background-color: #2ecc71;
        color: #fff;
}
.example button:nth-of-type(1):hover {
   transform: skewX(20deg);
}
.example button:nth-of-type(2):hover {
   transform: skewX(-20deg);
}

/*EXERCISE*/

.exercise div {
        color: #fff;
        width: 50%;
        padding: 10px;
        margin: 30px auto;
        background-color: #e74c3c;
}

/*EXERCISE*/

.exercise div {
        color: #fff;
        width: 50%;
        padding: 10px;
        margin: 30px auto;
        background-color: #e74c3c;
}

.skew-light
{
   transform: skewX(20deg);
}

.skew-medium
{
   transform: skewX(50deg);
}

.skew-hard
{
   transform: skewX(70deg);
```

3.5. skewY()

skewY is a fraternal property of skewX. And how else - it skews the object, but in this case along the Y axis.

And as we already know - the units we use to define the skew are degrees (deg).

To make it happen in the lesson, let's go to the example:
transform: skewY (50deg)
The property thus constructed will skew the element 50 degrees along the axis **Y**.

WORKS

1. 1. For the element with the skew-light class, add the property transform: skewY () with the value 5deg.
2. 2. The skew-hard element. Specify the skewY property with the value -15deg.
3. 3. Compare the elements and check the difference!

HINT

The code for the .skew-hard element should look like this:

```
.skew-hard {
transform: skewY(-15deg);
}
```

```
<!DOCTYPE html>
<html>

<head>
  <title>skewY()</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>

<body>
  <div class="hero">
    <h1>skewY()</h1>
  </div>
  <div class="container">

    <div class="example">
      <h2>Example:</h2>
      <p>Put your mouse over button and
see <b>skewY()</b> working!</p>
      <button>button</button>
    </div>

    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="skew-light">skew-
light</div>
      <div class="skew-hard">skew-
hard</div>
    </div>

  </div>
</body>

</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/4.jpg')
bottom;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  background-color: white;
  padding: 20px;
  box-shadow: 2px 2px 10px #000;
  text-align: center;
}
.example,
```

```
.exercise {                                                    }
   padding-bottom: 20px;
}
.example button {
   padding: 20px;
   margin: 10px;
   border: none;
   outline: none;
   background-color: #27ae60;
   width: 40%;
   color: #fff;
   text-transform: uppercase;
   cursor: pointer;
   transition: transform 1s;
}
.example button:hover {
        background-color: #2ecc71;
        color: #fff;
}
.example button:nth-of-type(1):hover {
   transform: skewY(20deg);
}
.example button:nth-of-type(2):hover {
   transform: skewY(-15deg);
}

/*EXERCISE*/

.exercise div {
        color: #fff;
        width: 50%;
        padding: 10px;
        margin: 30px auto;
        background-color: #e74c3c;
}

/*EXERCISE*/

.exercise div {
        color: #fff;
        width: 50%;
        padding: 10px;
        margin: 30px auto;
        background-color: #e74c3c;
}

.skew-light
{
   transform: skewY(5deg);
}

.skew-hard
{
   transform: skewY(-15deg);
```

3.6. matrix()

matrix () combines all other properties into one, slightly more complex one.

The property structure is as follows:

matrix (scaleX, skewY, skewX, scaleY, translateX, translateY)

So let's say that we want to give the element the following properties:

• slightly bend the element relative to the X axis and the Y axis
• move the element by 30px along the X axis
• and move the element by 10px on the Y axis
We could do it using four properties, or one.
So why bother yourself with life if the desired effect is achieved thanks to:

matrix(1, 0.03, 0.03, 1, 30, 10)

WORKS

1. 1. Create a rule for the .exercise-pseudo-class selector: hover.
2. 2. Give it the property transform.
3. 3. And give it the matrix value (0.8, 0.1, 0.1, 0.7, 30, 30).
4. 4. Hover over the element and check how the changes take place.

HINT

The correct code for the .exercise-element selector: hover should look like this:
.exercise-element:hover {
transform: matrix(0.8, 0.1, 0.1, 0.7, 30, 30);
}

```
<!DOCTYPE html>
<html>
<head>
  <title>matrix()</title>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <div class="hero">
    <h1>matrix()</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>Put your mouse over button and see <b>matrix()</b> working!</p>
      <button>button</button>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="exercise-element">exercise element</div>
    </div>
  </div>
</body>
</html>
```

```
@import url(https://fonts.googleapis.com/css?family=Open+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/4.jpg') bottom;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  background-color: white;
  padding: 20px;
  box-shadow: 2px 2px 10px #000;
  text-align: center;
}
```

```
.example,
.exercise {
   padding-bottom: 20px;
}
.example button {
   padding: 20px;
   margin: 10px;
   border: none;
   outline: none;
   background-color: #27ae60;
   width: 40%;
   color: #fff;
   text-transform: uppercase;
   cursor: pointer;
   transition: transform 1s;
}
.example button:hover {
        background-color: #2ecc71;
        color: #fff;
        transform: matrix(1.1, 0.03, 0.03, 1.2,
30, 10);
}

/*EXERCISE*/

.exercise-element {
        color: #fff;
        width: 50%;
        padding: 10px;
        margin: 30px auto;
        background-color: #e74c3c;
        text-transform: uppercase;
        transition-duration: 2s;
}

/*EXERCISE*/

.exercise-element {
        color: #fff;
        width: 50%;
        padding: 10px;
        margin: 30px auto;
        background-color: #e74c3c;
        text-transform: uppercase;
        transition-duration: 2s;
}

.exercise-element:hover
{
   transform: matrix(0.8, 0.1, 0.1, 0.7, 30, 30);
}
```

3.7. transform-origin

As you probably noticed, all transformations occur in relation to the center of the object.

This is because its "source point" (meaning the default value of transform-origin) is in a 50% 50% coordinate.

The position of this point consists of two values (three possible, but we will tell about it in 3D transformations) - the value of the X axis and the value of the Y axis.

The coordinate for the X axis can have the following values:
left
center
right
%

The coordinate for the Y axis can have the following values:
top
center
bottom
%

Sounds a bit confusing? Take a good look at how the sample code might look like:
transform-origin: 50% 0%

In this way, we moved the source point of the element to the half-width of its upper edge.

WORKS

1. 1. Create a rule with the .new-origin: hover selector.
2. 2. Give it the rotateY property (360deg).
3. 3. Next, hover over the element and see what the transformation looks like.
4. 4. Add the transform-origin property: 0 0.
5. 5. Check again how the transformation works

HINT

This transform-origin value moved the source point to the upper-left corner of the element..

```
<!DOCTYPE html>
<html>
<head>
  <title>matrix()</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>matrix()</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>Put your mouse over example area
and see <b>transform-origin</b>
working!</p>
      <button>button with default
origin</button>
      <button>button with changed
origin</button>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="new-origin">exercise
element</div>
    </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
```

```
    font-size: 200%;
    text-shadow: 1px 1px 1px #000;
}
.hero {
    background: url('/static/lessons/4.jpg')
bottom;
    background-size: cover;
    min-height: 150px;
    box-shadow: 1px 1px 3px #000;
}
.container {
    width: 80%;
    margin: 20px auto;
    background-color: white;
    padding: 20px;
    box-shadow: 2px 2px 10px #000;
    text-align: center;
}
.example,
.exercise {
    padding-bottom: 20px;
}
.example button {
    padding: 20px;
    margin: 10px;
    border: none;
    outline: none;
    background-color: #27ae60;
    width: 40%;
    color: #fff;
    text-transform: uppercase;
    cursor: pointer;
    transition: transform 1s;
}
.example button:hover {
        background-color: #2ecc71;
        color: #fff;
}
.example:hover button {
        transform: rotateX(360deg);
}
.example button:nth-of-type(2) {
        transform-origin: 0% 0%;
}

/*EXERCISE*/

.new-origin {
        color: #fff;
        width: 50%;
        padding: 10px;
        margin: 30px auto;
        background-color: #16a085;
        text-transform: uppercase;
        transition-duration: 2s;
}

/*EXERCISE*/

.new-origin {
        color: #fff;
        width: 50%;
        padding: 10px;
        margin: 30px auto;
        background-color: #16a085;
        text-transform: uppercase;
        transition-duration: 2s;
}

.new-origin:hover
{
    transform: rotateY(360deg);
    transform-origin: 0 0;
}
```

4.1. perspective

At the very beginning it should be mentioned how to make 3D transformations possible, perspective is required.

We can give it through two CSS properties.

The first one is transform: perspective.

It gives the chosen element "depth", which serves as a perspective.

Another property is: perspective: Xpx (where X is a number) is given to the parent element and also applies to the elements contained in it.

The units used to specify these properties are pixels (px).

The higher the value, the perspective of the perspective decreases, and the element "moves away" from us.

WORKS

1. 1. Select the transform: perspective (800px) rotateY property (60deg) for the element selector with the big-perspective class.
2. 2. Add an 800px perspective to the element selector of the complex-perspective class using the perspective property.
3. 3. Select the transform: perspective (200px) rotateY property (60deg.) For the small-perspectived element selector.

HINT

You should add the following properties:

transform: perspective(800px) rotateY(60deg)

perspective: 800px

transform: perspective(200px) rotateY(60deg)

```
<!DOCTYPE html>
<html>
<head>
  <title>Perspective</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>Perspective</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>This is element using
<b>perspective: 800px</b>.</p>
      <div class="global-perspective">
        <div>1</div>
        <div>2</div>
        <div>3</div>
      </div>
      <p>This is element using <b>transform:
perspective(800px)</b>.</p>
      <div class="element-perspective">
        <div>1</div>
        <div>2</div>
        <div>3</div>
      </div>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="big-perspective">big-
perspective</div>
      <div class="complex-
perspective">complex-perspective</div>
      <div class="small-perspective">small-
perspective</div>
    </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
```

```
}
h1 {
   margin: 0;
   color: #fff;
   text-align: center;
   padding: 50px;
   font-size: 200%;
   text-shadow: 1px 1px 1px #000;
}
.hero {
   background: url('/static/lessons/4.jpg')
bottom;
   background-size: cover;
   min-height: 150px;
   box-shadow: 1px 1px 3px #000;
}
.container {
   width: 80%;
   margin: 20px auto;
   background-color: white;
   padding: 20px;
   box-shadow: 2px 2px 10px #000;
   text-align: center;
}
.example,
.exercise {
   padding-bottom: 20px;
}
.global-perspective, .element-perspective {
        margin-top: 20px;
}
.global-perspective {
        perspective: 800px;
}
.global-perspective div {
        transform: rotateY(70deg);
}
.global-perspective div, .element-perspective
div {
        background: red;
        width: 40%;
        height: 30px;
        margin: 5px auto;
}
.element-perspective div {
        transform: perspective(800px)
rotateY(70deg);
}

/*EXERCISE*/

.exercise > div {
        width: 50%;
        margin: 10px auto;
        color: #fff;
        background-color: #9b59b6;
}
.small-perspective {

}
.big-perspective {

}
.complex-perspective {
   transform: rotateY(60deg);
}

/*EXERCISE*/

.exercise > div {
        width: 50%;
        margin: 10px auto;
        color: #fff;
        background-color: #9b59b6;
}
.small-perspective {
   transform: perspective(200px)
rotateY(60deg);
}
.big-perspective {
   transform: perspective(800px)
rotateY(60deg);
}
.complex-perspective {
   transform: rotateY(60deg);
   perspective: 800px;
}
```

4.2. translate3d()

translate3d (x, y, z) defines the displacement of an object in a three-dimensional space. The first two values define the displacement in the X axis (horizontal) and the Y axis (vertical), the third in the Z axis.

For example, the property might look like:
transform: translate3d (10%, 10%, 200px)

The result will be an object shift by 10% on the Y axis, 10% on the X axis and 200px on the Z axis. (Units used for Z axis displacement can only be pixel values).

WORKS

1. 1. Create a rule with the selector .exercise-element: hover.
2. 2. Add the transform: perspective (800px) rotateX (45deg) translate3d (10%, 10%, -100px) property in it.
3. 3. Hover over the element and check how its position changes!

HINT

The correct code for the element .exercise-element with pseudo-class: hover looks like this:
.exercise-element:hover {
transform: perspective(800px) rotateX(45deg) translate3d(10%, 10%, -100px);
}

```
<!DOCTYPE html>
<html>
<head>
  <title>translate3d()</title>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <div class="hero">
    <h1>translate3d()</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>Put your cursor over elements!</p>
                    <div></div>
                    <div></div>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="exercise-element"></div>
    </div>
  </div>
</body>
</html>
```

```
@import url(https://fonts.googleapis.com/css?family=Open+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/4.jpg') bottom;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  background-color: white;
  padding: 20px;
  box-shadow: 2px 2px 10px #000;
  text-align: center;
}
.example,
.exercise {
  padding-bottom: 20px;
}
.example div {
        width: 100px;
        height: 100px;
```

```
        background-color: red;
        margin: 10px auto;
        transition-duration: 1s;
}
.example div:nth-of-type(1):hover {
        transform: perspective(500px)
translate3d(10px, 10px, -50px);
}
.example div:nth-of-type(2):hover {
        transform: perspective(500px)
translate3d(5px, 5px, 10px);
}

/*EXERCISE*/

.exercise-element {
   width: 180px;
   height: 180px;
   margin: auto;
   background: blue;
   transform: perspective(800px)
rotateX(45deg);
   transition-duration: 2s;
}

/*EXERCISE*/

.exercise-element {
   width: 180px;
   height: 180px;
   margin: auto;
   background: blue;
   transform: perspective(800px)
rotateX(45deg);
   transition-duration: 2s;
}

.exercise-element:hover
{
   transform: perspective(800px)
rotateX(45deg) translate3d(10%, 10%, -
100px);
}
```

4.3. scale3d()

scale3d (x, y, z) is another property with a similar behavior to its predecessor operating in two-dimensional space.

In this case, we scale the elements in three dimensions.

The scale3d (x, y, z) property scales the element in sequence:

x in the X axis
y in the Y axis
z in the Z axis

The property accepts numbers ranging from 0 to infinity.
For numbers smaller than 1, the element will be reduced. If the number is zero, the element will disappear, and for positive numbers greater than 1 will be increased.

Theoretically, the property also takes negative numbers, but the element will not be scaled then.

WORKS

1. 1. Create a rule with the .practice: hover selector and give it the transform property.
2. 2. Make the object scale in the direction of the X axis by 1.7, the Y axis by 1.1 and in the direction of the Z axis at all.
3. 3. Hover over the item and check for changes.

HINT

The property should look like this: transform: scale3d(1.7, 1.1, 1)

```
<!DOCTYPE html>
<html>
<head>
  <title>scale3d()</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>scale3d()</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>Put your cursor over elements!</p>
                    <div></div>
                    <div></div>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>

      <div class="practice"></div>
    </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/4.jpg')
bottom;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  background-color: white;
  padding: 20px;
  box-shadow: 2px 2px 10px #000;
  text-align: center;
}
.example,
.exercise {
```

```
   padding-bottom: 20px;
}
.example div {
	width: 100px;
	height: 100px;
	background-color: #1abc9c;
	border: 2px solid #fff;
	margin: 10px auto;
	transition-duration: 1s;
}
.example div:nth-of-type(1):hover {
	transform: scale3d(1.5, 1.5, 1);
}
.example div:nth-of-type(2):hover {
	transform: scale3d(1.5, 0.8, 1);
}

/*EXERCISE*/

.practice {
   width: 140px;
   height: 140px;
   margin: auto;
   background: blue;
   transition-duration: 2s;
}

/*EXERCISE*/

.practice {
   width: 140px;
   height: 140px;
   margin: auto;
   background: blue;
   transition-duration: 2s;
}

.practice:hover
{
   transform: scale3d(1.7, 1.1, 1);
}
```

4.4. rotate3d()

rotate 3d () is a property that you will definitely like. With its help, you will rotate the element relative to the three axes at the same time.

The construction of the property is as follows:
transform: rotate3d (x, y, z, deg)

Where further values mean:
x - position of the x vector
y - position of the vector y
z - the position of the vector z
deg - amount of rotation in degrees (possible negative values, then the element rotates in the counterclockwise direction)

So if the space between the vectors is the same regardless of the value (eg all vectors have the value 5, or all have the value 100), the animation will be displayed in the same way.

The disproportion of the values of particular vectors will have an obvious influence on the differences in rotation.

The example property may look like this:
transform: rotate3d(1,2,1, 180deg)

WORKS

- Create a rule with the .practice: hover selector.
- Add the transform: rotate3d (1,2,3,360deg) property in it.
- Hover over the element and check what change is taking place!

HINT

The correct selector that you need to create in this task looks as follows:

```
.practice:hover {
transform: rotate3d(1, 2, 3, 360deg);
}
```

```
<!DOCTYPE html>
<html>
<head>
   <title>rotate3d()</title>
   <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
   <div class="hero">
      <h1>rotate3d()</h1>
   </div>
   <div class="container">
      <div class="example">
         <h2>Example:</h2>
         <p>Put your cursor over elements!</p>
                              <div></div>
                              <div></div>
      </div>
      <div class="exercise">
         <h2>Exercise:</h2>
         <p>Practice!</p>
         <div class="practice"></div>
      </div>
   </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
   box-sizing: border-box;
}
body {
   padding: 0;
   margin: 0;
   background-color: #2980b9;
   font-family: 'Open Sans', sans-serif;
}
h1 {
   margin: 0;
   color: #fff;
   text-align: center;
   padding: 50px;
   font-size: 200%;
   text-shadow: 1px 1px 1px #000;
}
.hero {
   background: url('/static/lessons/4.jpg')
bottom;
   background-size: cover;
   min-height: 150px;
   box-shadow: 1px 1px 3px #000;
}
.container {
   width: 80%;
   margin: 20px auto;
   background-color: white;
   padding: 20px;
   box-shadow: 2px 2px 10px #000;
```

```
   text-align: center;
}
.example,
.exercise {
   padding-bottom: 20px;
}
.example div {
        width: 100px;
        height: 100px;
        background-color: #1abc9c;
        border: 2px solid #fff;
        margin: 10px auto;
        transition-duration: 1s;
}
.example div:nth-of-type(1):hover {
        transform: rotate3d(2, 3, 4, 180deg);
}
.example div:nth-of-type(2):hover {
        transform: rotate3d(1, 2, 1, 180deg);
}

/*EXERCISE*/

.practice {
   width: 140px;
   height: 140px;
   margin: auto;
   background: blue;
   transition-duration: 2s;
}

/*EXERCISE*/

.practice {
   width: 140px;
   height: 140px;
   margin: auto;
   background: blue;
   transition-duration: 2s;
}

.practice:hover
{
   transform: rotate3d(1, 2, 3, 360deg);
}
```

4.5. backface-visibility

backface-visiblity is a property that hides the contents of an element when we can see its back face.

backface-visibility has only two properties:
visible (the back of the element is visible)
hidden (the back of the element is hidden and the element becomes black)

For example, the property may look like this:
backface-visibility: hidden

WORKS

1. 1. Create a rule with the .practice: hover selector.
2. 2. And give it the property transform: rotateY (360deg).
3. 3. Hover over the element and check the appearance of the change.
4. 4. Then add the property backface-visibility: hidden ;.
5. 5. Check what changes have taken place!

HINT

The correct code looks as follows:

```
.practice:hover {
transform: rotateY(360deg);
backface-visibility: hidden;
}
```

```
<!DOCTYPE html>
<html>
<head>
  <title>backface-visibility</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>backface-visibility</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>Put your cursor over elements!</p>
                    <div>element with
default backface-visibility</div>
                    <div>element with
changed backface-visibilty value</div>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="practice"></div>
    </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/4.jpg')
bottom;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  background-color: white;
  padding: 20px;
  box-shadow: 2px 2px 10px #000;
  text-align: center;
}
.example,
.exercise {
  padding-bottom: 20px;
}
```

```
.example div {
	width: 100px;
	padding: 10px;
	background-color: #1abc9c;
	border: 2px solid #fff;
	margin: 10px auto;
	transition-duration: 1s;
	text-transform: uppercase;
	color: #fff;
}
.example div:hover {
	transform: rotateY(180deg);
}
.example div:nth-of-type(2) {
	backface-visibility: hidden;
}
/*EXERCISE*/

.practice {
  width: 140px;
  height: 140px;
  margin: auto;
  background: blue;
  transition-duration: 1s;
}

/*EXERCISE*/

.practice {
  width: 140px;
  height: 140px;
  margin: auto;
  background: blue;
  transition-duration: 1s;
}

.practice:hover
{
  transform: rotateY(360deg);
  backface-visibility: hidden;
}
```

5.1. The basics of keyframes

We should start the lessons about CSS animations from the very basics. Undoubtedly, the basis is the required properties that will allow us to put animation in motion as well as the construction itself.

We start the declaration of animation with @keyframes and then add its name and open the brackets.

For example:
@keyframes mojaAnimacja {...}

Within this declaration, we place keywords (keywords) that act as steps. Possible keywords are: from and meaning as follows: from, to (start and end of the animation).

Possible keywords are also percentages. An example of such a construction can be found in the style.css and preview file.

An example for an animation that uses from to, which will move an element from left to right (only by 200px) may look like:

```
@keyframes animacja {
from {left: 0}
to {left: 200px}
}
```

WORKS

1. 1. In the case-element selector, we already have a ready-made property, animation so we'll only do the creation of keyframes.
2. 2. Create the selector rule @keyframes and name it buttonColor.
3. 3. Create in it the keyword from and place in it background-color: # f39c12.
4. 4. Then create keyword to and put in it background-color: blue.

HINT

The @keyframes rule should look like this:

```
@keyframes buttonColor{
from {
background-color: #f39c12;
}
to {
background-color: blue;
}
}
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>Keyframes basics</title>
    <link rel="stylesheet" type="text/css"
href="style.css">
  </head>
  <body>
    <div class="hero">
      <h1>keyframes basics</h1>
    </div>
    <div class="container">

      <div class="example">
        <h2>Example:</h2>
        <p>This is animated button created
by percentage keyframes!</p>
        <button>button</button>
      </div>

      <div class="exercise">
        <h2>Exercise:</h2>
        <p>Create your own animation!</p>
        <button class="practice-
button">Animate me!</button>
      </div>

    </div>
  </body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
```

```
   margin: 0;
   color: #fff;
   text-align: center;
   padding: 50px;
   font-size: 200%;
   text-shadow: 1px 1px 1px #000;
}
.hero {
   background: url('/static/lessons/16.jpg')
center;
   background-size: cover;
   min-height: 150px;
   box-shadow: 1px 1px 3px #000;
}
.container {
   width: 80%;
   margin: 20px auto;
   background-color: white;
   padding: 20px;
   box-shadow: 2px 2px 10px #000;
   text-align: center;
}
.example,
.exercise {
   padding-bottom: 20px;
}
.example button {
   padding: 20px;
   margin: 10px;
   border: none;
   outline: none;
   background-color: #27ae60;
   width: 40%;
   color: #fff;
   text-transform: uppercase;
   cursor: pointer;
   animation: buttonPulse 1s infinite;
}
.example button:hover {
   background-color: #2ecc71;
}
@keyframes buttonPulse {
   0% {
      transform: scale(1);
   }
   50% {
      transform: scale(1.05);
   }
   100% {
      transform: scale(1);
   }
}

/*EXERCISE*/
```

```
.practice-button {
   border: none;
   padding: 20px;
   outline: none;
   background-color: #f39c12;
   color: #fff;
   animation: buttonColor 2s infinite;
}

/*EXERCISE*/

.practice-button {
   border: none;
   padding: 20px;
   outline: none;
   background-color: #f39c12;
   color: #fff;
   animation: buttonColor 2s infinite;
}

@keyframes buttonColor
{
   from
   {
      background-color: #f39c12;
   }
   to
   {
      background-color: blue;
   }
}
```

5.2. name, duration - construction of properties

Since you already know the basics of building the keyframes rule, we can go on to learn about the construction of animation properties - it is necessary to set the animation in motion.

animation requires two properties to work. The first is the name of the animation and the second time it is performed. Of course, we can add other values to it, which are responsible for, for example, the number of repetitions, but we will tell about it later :)

An example property is as follows:
animation: myAnimacja 1s

animation can be broken down into single properties: animation-name and animation-duration. However, a single animation property will save us writing code and group the necessary properties, just like the previous "shortcut" we met, namely transition.

WORKS

1. 1. Since you already know how to create keyframes, let's practice creating properties that support them.
2. 2. Create a practice-button element selector in the hover state
3. 3. Give it the animation property.
4. 4. And add values that support animation called exercise and give it 1s duration.

HINT

The selector should look like this:
.practice-button:hover{...}

```
<!DOCTYPE html>
<html>
<head>
  <title>Animation basics</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>Animation basics</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>This is animated button created by
percentage keyframes!</p>
      <button>button</button>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Create your own animation!</p>
      <button class="practice-
button">Animate me!</button>
    </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/16.jpg')
center;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  background-color: white;
```

```
    padding: 20px;
    box-shadow: 2px 2px 10px #000;
    text-align: center;
}
.example,
.exercise {
    padding-bottom: 20px;
}
.example button {
    padding: 20px;
    margin: 10px;
    border: none;
    outline: none;
    background-color: #27ae60;
    width: 40%;
    color: #fff;
    text-transform: uppercase;
    cursor: pointer;
    animation: buttonPulse 1s infinite;
}
.example button:hover {
    background-color: #2ecc71;
}
@keyframes buttonPulse {
    0% {
        transform: scale(1);
    }
    50% {
        transform: scale(1.05);
    }
    100% {
        transform: scale(1);
    }
}

/*EXERCISE*/

.practice-button {
    border: none;
    text-transform: uppercase;
    width: 40%;
    padding: 20px;
    outline: none;
    background-color: #f39c12;
    color: #fff;
    cursor: pointer;
}

@keyframes exercise {
        from {transform: rotateY(0deg)}
        to {transform: rotateY(360deg)}
}

/*EXERCISE*/
```

```
.practice-button {
    border: none;
    text-transform: uppercase;
    width: 40%;
    padding: 20px;
    outline: none;
    background-color: #f39c12;
    color: #fff;
    cursor: pointer;
}

.practice-button:hover
{
    animation: exercise 1s;
}

@keyframes exercise {
        from {transform: rotateY(0deg)}
        to {transform: rotateY(360deg)}
}
```

5.3. Animation pace - animation-timing-function

I bet you remember the lesson about transition-timing-function. If so, you probably guess that the action of the animation-timing-function is identical. If, however, you have forgotten, I will quickly remind you of the most important assumptions.

The animation-timing-function property determines the rate at which the animation steps are run (start, middle, end), but it does not affect the duration of the animation. We can assume that it is a kind of synchronization function.

Examples of properties are:
ease (default value - slow start, quick center and slow end)
linear (constant tempo)
ease-in-out (slow start and end)
cubic-bezier (x, x, x, x) (where x is a number from 0 to 1) (any function created by us)
steps (x) (where x is the number of steps) (breaks the animation into steps in the given amount)

You can see examples with your own eyes in the preview and look at the differences between them!

WORKS

1. 1. We already have a ready-made, active animation, but we want to change it a bit through animation-timing-function, so let's get to work!
2. 2. Create the .practice-steps selector and give it the synchronization function that will break the animation into 6 steps.
3. 3. Next, create a selector for the practice-linear element.
4. 4. And give it an animation-timing-function property with a linear value.
5. 5. In the preview window, see how much this property changes!

HINT

The first selector should contain a property:
animation-timing-function: steps(6)

```
<!DOCTYPE html>
<html>
  <head>
    <title>animation-timing-function</title>
    <link rel="stylesheet" type="text/css"
href="style.css">
  </head>
  <body>
    <div class="hero">
      <h1>animation-timing-function</h1>
    </div>
    <div class="container">

      <div class="example">
        <h2>Example:</h2>
        <p><b>ease</b> (default) timing
function.</p>
        <div></div>
        <p><b>linear</b> timing
function.</p>
        <div></div>
        <p><b>ease-in-out</b> timing
function.</p>
        <div></div>
        <p><b>cubic-
bezier(x,x,x,x)</b>(where <b>x</b> is number
between 0 and 1) timing function.</p>
        <div></div>
        <p><b>steps(x)</b>(where <b>x</b>
is a number of steps) timing function</p>
        <div></div>
      </div>

      <div class="exercise">
        <h2>Exercise:</h2>
        <div class="practice-steps">Element
using steps timing function</div>
        <div class="practice-linear">Element
using linear timing function</div>
      </div>

    </div>
  </body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/16.jpg')
center;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  background-color: white;
  padding: 20px;
  box-shadow: 2px 2px 10px #000;
  text-align: center;
}
.example,
.exercise {
  padding-bottom: 20px;
}
.example div {
  width: 100px;
  height: 50px;
  margin: auto;
  background-color: #16a085;
  animation: pulse 2s infinite;
}
.example div:nth-of-type(2) {
  animation-timing-function: linear;
}
.example div:nth-of-type(3) {
  animation-timing-function: ease-in-out;
}
.example div:nth-of-type(4) {
  animation-timing-function: cubic-bezie(0.7,
0.3, 0.4, 0.6);
}
.example div:nth-of-type(4) {
  animation-timing-function: steps(4);
}
@keyframes pulse {
  0% {
    transform: scale(1);
  }
  50% {
    transform: scale(1.2);
  }
  100% {
    transform: scale(1);
  }
}

/*EXERCISE*/

.exercise div {
  background-color: #e67e22;
  color: #fff;
  font-weight: bold;
  margin-top: 15px;
  max-width: 200px;
  padding: 5px;
  animation: marginLeft 3s infinite;
}
@keyframes marginLeft {
  0% {
    margin-left: 0;
  }
  50% {
    margin-left: 20%;
  }
  100% {
    margin-left: 0;
  }
}




/*EXERCISE*/

.exercise div {
  background-color: #e67e22;
  color: #fff;
  font-weight: bold;
  margin-top: 15px;
  max-width: 200px;
  padding: 5px;
  animation: marginLeft 3s infinite;
}
```

```
@keyframes marginLeft {
   0% {
      margin-left: 0;
   }
   50% {
      margin-left: 20%;
   }
   100% {
      margin-left: 0;
   }
}

.practice-steps
{
   animation-timing-function: steps(6);
}

.practice-linear
{
   animation-timing-function: linear;
}
```

5.4. We delay the animation - animation-delay

In this lesson, we will learn the animation-delay property. Yes, you are right - we have already learned the twin property during a lesson connected with transitions.

As a quick reminder, I will mention the effect of this property (in the end it is always better to consolidate the newly learned knowledge).

animation-delay sets the delay in animation and does not affect the speed of its execution.

After all, if we delay all the animations by a multiple of their execution time, then after the first loop the difference in time will no longer be visible - that is why it is worth to delay at least one of the animations by 1.5 times its duration instead of by 1 or 2 times of execution :)

An example of a property that delays the animation by one second may be as follows:
animation-delay: 1s

WORKS

1. 1. Create a selector for the element with a delay-short class in the element with the exercise class and give it a delay of animation by 2s
2. 2. Next, create an element selector of the delay-medium class contained in the .exercise element and give it animation-delay: 3s.
3. 3. Finally, create a selector for the element with a delay-long class in the element with the exercise class and specify the delay for its animation by 4s.

HINT

The correct selector for the .delay-short element looks like this:
.exercise .delay-short {
animation-delay: 2s;
}

```
<!DOCTYPE html>
<html>
  <head>
    <title>animation-timing-function</title>
    <link rel="stylesheet" type="text/css"
href="style.css">
  </head>
  <body>
    <div class="hero">
      <h1>animation-timing-function</h1>
    </div>
    <div class="container">
      <div class="example">
        <h2>Example:</h2>
        <p>2s delay</p>
        <div></div>
        <p>4s delay</p>
        <div></div>
        <p>6s delay</p>
        <div></div>
      </div>
      <div class="exercise">
        <h2>Exercise:</h2>
        <p>2s delay</p>
        <button class="delay-
short"></button>
        <p>3s delay</p>
        <button class="delay-
medium"></button>
        <p>4s delay</p>
        <button class="delay-
long"></button>
      </div>
    </div>
  </body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
```

```
    margin: 0;
    color: #fff;
    text-align: center;
    padding: 50px;
    font-size: 200%;
    text-shadow: 1px 1px 1px #000;
}
.hero {
    background: url('/static/lessons/16.jpg')
center;
    background-size: cover;
    min-height: 150px;
    box-shadow: 1px 1px 3px #000;
}
.container {
    width: 80%;
    margin: 20px auto;
    background-color: white;
    padding: 20px;
    box-shadow: 2px 2px 10px #000;
    text-align: center;
}
@keyframes pulse {
    0% {
        transform: scale(1);
    }
    50% {
        transform: scale(1.2);
    }
    100% {
        transform: scale(1);
    }
}
.example,
.exercise {
    padding-bottom: 20px;
}
.example div {
    width: 100px;
    height: 50px;
    margin: auto;
    background-color: #16a085;
    animation: pulse 2s infinite;
}
.example div {
    animation-delay: 2s;
}
.example div:nth-of-type(2) {
    animation-delay: 4s;
}
.example div:nth-of-type(3) {
    animation-delay: 6s;
}
```

```
/*EXERCISE*/
@keyframes marginLeft {
    0% {
        margin-left: 0;
    }
    50% {
        margin-left: 20%;
    }
    100% {
        margin-left: 0;
    }
}
.exercise button {
    background-color: #00ff48;
    border: none;
    width: 20px;
    height: 20px;
    color: #fff;
    font-weight: bold;
    margin-top: 15px;
    max-width: 200px;
    padding: 5px;
    animation: marginLeft ease 2s infinite;
}

/*EXERCISE*/
@keyframes marginLeft {
    0% {
        margin-left: 0;
    }
    50% {
        margin-left: 20%;
    }
    100% {
        margin-left: 0;
    }
}
.exercise button {
    background-color: #00ff48;
    border: none;
    width: 20px;
    height: 20px;
    color: #fff;
    font-weight: bold;
    margin-top: 15px;
    max-width: 200px;
    padding: 5px;
    animation: marginLeft ease 2s infinite;
}

.exercise .delay-short
{
    animation-delay: 2s;
}
```

```
.exercise .delay-medium
{
   animation-delay: 3s;
}
.exercise .delay-long
{
   animation-delay: 4s;
}
```

5.5. We count the repetition - animation-iteration-count

Nothing can last forever ...

Or maybe? We will check this using the animation-iteration-count property. As the name indicates, this property determines the number of repetitions of the animation.

The default value of animation-iteration-count is the number 1. If the default value is a number, then going further with this logic we can conclude that the example value of this property is also simply a number.

So if we want to do the animation five times then the property will look like this:
animation-iteration-count: 5

Well, what if, however, we want the animation not to stop after a certain number of repetitions? In this case, we will use the infinite value, and the property will look like this:
animation-iteration-count: infinite

WORKS

1. 1. First, assign a property for the element selector with the class few-repeats to specify the number of repetitions of the animation.
2. 2. Let the animation repeat 4 times.
3. 3. Enter the animation-iteration-count property into the infinite-animation element selector. Set its value so that the animation repeats itself infinitely many times.

HINT

Second The property should look like this:
animation-iteration-count: infinite

```
<!DOCTYPE html>
<html>
  <head>
    <title>animation-timing-function</title>
    <link rel="stylesheet" type="text/css"
href="style.css">
  </head>
  <body>
    <div class="hero">
      <h1>animation-timing-function</h1>
    </div>
    <div class="container">

      <div class="example">
        <h2>Example:</h2>
        <p>It repeats 3 times.</p>
        <div></div>
      </div>

      <div class="exercise">
        <h2>Exercise:</h2>
        <div class="few-repeats"></div>
        <div class="infinite-
animation"></div>
      </div>

    </div>
  </body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/16.jpg')
center;
  background-size: cover;
```

```
    min-height: 150px;
    box-shadow: 1px 1px 3px #000;
}
.container {
    width: 80%;
    margin: 20px auto;
    background-color: white;
    padding: 20px;
    box-shadow: 2px 2px 10px #000;
    text-align: center;
}
.example,
.exercise {
    padding-bottom: 20px;
}
.example div {
    width: 100px;
    height: 100px;
    border-radius: 50%;
    margin: auto;
    background-color: #16a085;
    animation: pulse 2s;
    animation-iteration-count: 3;
}
@keyframes pulse {
    0% {
        transform: scale(1);
    }
    50% {
        transform: scale(1.2);
    }
    100% {
        transform: scale(1);
    }
}

/*EXERCISE*/

.exercise div {
    background-color: #e67e22;
    color: #fff;
    font-weight: bold;
    margin-top: 15px;
    height: 90px;
            width: 90px;
            margin-left: auto;
            margin-right: auto;
    animation-name: pulse;
    animation-duration: 2s;
}

/*EXERCISE*/

.exercise div {
    background-color: #e67e22;
    color: #fff;
    font-weight: bold;
    margin-top: 15px;
    height: 90px;
            width: 90px;
            margin-left: auto;
            margin-right: auto;
    animation-name: pulse;
    animation-duration: 2s;
}

.few-repeats
{
    animation-iteration-count: 4;
}

.infinite-animation
{
    animation-iteration-count: infinite;
}
```

5.6. Your first animation

We have already learned the most important animation properties in CSS - it's time to put it into practice.

In the starting code we have an animation that suggests loading - it looks nice, but it's a bit too empty - let's do something with it.

Let's add a second animated element in the middle of the present - this time smaller and rotating the other way - thanks to this our animation will be more interesting.

WORKS

1. 1. Create the selector .spinner: before and add the following properties in it: absolute position, content: "", width and height 60px, box-sizing: border-box, border with 2px solid transparent properties and right and left border with 2px properties solid black.
2. 2. In the same selector, add border-radius: 50%, transparent background, left top with 50% value, transform with translate value (-50%, -50%) and the animation property with the value 1s linear spin-counterclockwise infinite.
3. 3. Let's get to create our animation - create @keyframes with the name spin-counterclockwise. For 50% progress, assign transform: translate (-50%, -50%) rotate (-180deg), and for progress 100% the same, only with the value -360deg instead of -180deg.

HINT

translate (-50%, -50%) is needed in the transform property in combination with toporaz left with a value of 50%, so that our pseudo-element: before with the absolute position is located in the center of the main element.

```
<!DOCTYPE html>
<html>
  <head>
    <title>animation-timing-function</title>
    <link rel="stylesheet" type="text/css"
href="style.css">
  </head>
  <body>
    <div class="hero">
      <h1>animation-timing-function</h1>
    </div>
    <div class="container">
      <div class="exercise">
        <h2>My first spinner</h2>
        <div class="spinner">

        </div>
      </div>
    </div>
  </body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/16.jpg')
center;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
```

```
    width: 80%;
    margin: 20px auto;
    background-color: white;
    padding: 20px;
    box-shadow: 2px 2px 10px #000;
    text-align: center;
}
.example,
.exercise {
    padding-bottom: 20px;
}
.exercise {
        min-height: 500px;
}

/*EXERCISE*/
.spinner {
        display: block;
        margin: 100px auto 0;
        position: relative;
        box-sizing: border-box;
        width: 100px;
        height: 100px;
        border: 3px solid transparent;
        border-left: 3px solid black;
        border-right: 3px solid black;
        border-radius: 50%;
        background: transparent;
        animation: 2s linear spin infinite;
}
@keyframes spin {
        50% {
                transform: rotate(180deg);
        }
        100% {
                transform: rotate(360deg);
        }
}

/*EXERCISE*/
.spinner {
        display: block;
        margin: 100px auto 0;
        position: relative;
        box-sizing: border-box;
        width: 100px;
        height: 100px;
        border: 3px solid transparent;
        border-left: 3px solid black;
        border-right: 3px solid black;
        border-radius: 50%;
        background: transparent;
        animation: 2s linear spin infinite;
}
@keyframes spin {
        50% {
                transform: rotate(180deg);
        }
        100% {
                transform: rotate(360deg);
        }
}

.spinner:before
{
    content: "";
    height: 60px;
    width: 60px;
    box-sizing: border-box;
    border: 2px solid transparent;
    border-right: 2px solid black;
    border-left: 2px solid black;
    border-radius: 50% top left;
    transform: translate(-50%, -50%);
    animation: 1s linear spin-counterclockwise
infinite;

}

@keyframes spin-counterclockwise
{
    50%
    {
        transform: translate(-50%, -50%) rotete(-
180deg);
    }
    100%
    {
        transform: translate(-360deg, -180deg)
rotete(-180deg);
    }
}
```

6.1. About filters and prefixes

In this lesson, we will discuss the behavior of the filter property.

Its task is to give selected effects (defined as its properties) to elements.

For example:
filter: blur (10px) (we will give the element a blur)

Extremely important information is that prefixes will be needed for proper operation of this property on different browsers.

Prefixes are specific prefixes that we add before properties to allow proper operation on a given type of browsers.

The existing prefixes are:
-webkit- (Chrome, Opera)
-moz- (Firefox)
-o- (older versions of the Opera)
-ms- (Internet Explorer)

In addition, it is worth emphasizing that only some of the properties need prefixes (usually newer ones), and browsers along with their development begin to support properties without the need to add prefixes.

In our case, we'll use -webkit-, and the example property may look like this:
-webkit-filter: blur(20px)

WORKS

1. 1. Go to the style.css file.
2. 2. The element with the old-photo class has the filter: sepia (100%) property, also add the prefixed property for it.
3. 3. Another item with the filter property you need to give the prefix value to is new-photo - in this case you must add a prefix for the contrast filter..

HINT

W elemencie o klasie old-photo powinny znajdować się następujące właściwości:
filter: sepia(100%)
-webkit-filter: sepia(100%)

The new-photo element should contain:
filter: contrast(150%)
-webkit-filter: contrast(150%)

```
<!DOCTYPE html>
<html>
<head>
  <title>Fiters and prefixes</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>Filters and prefixes</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>This is element with prefix value
:</p>
                        <div></div>
                        <p>This is element
without prefixed value:</p>
                        <div></div>
                        <small>NOTE: If you
are <b>opera</b> or <b>chrome</b> user
second element will not have filter.</small>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="new-photo"></div>
      <div class="old-photo"></div>
    </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
```

```
   margin: 0;
   background-color: #2980b9;
   font-family: 'Open Sans', sans-serif;
}
h1 {
   margin: 0;
   color: #fff;
   text-align: center;
   padding: 50px;
   font-size: 200%;
   text-shadow: 1px 1px 1px #000;
}
.hero {
   background: url('/static/lessons/4.jpg')
bottom;
   background-size: cover;
   min-height: 150px;
   box-shadow: 1px 1px 3px #000;
}
.container {
   width: 80%;
   margin: 20px auto;
   background-color: white;
   padding: 20px;
   box-shadow: 2px 2px 10px #000;
   text-align: center;
}
.example,
.exercise {
   padding-bottom: 20px;
}
.example div {
        width: 150px;
        padding: 10px;
        border: 1px solid #34495e;
        margin: 10px auto;
        height: 200px;
        background: url('/static/lessons/1.jpg')
center;
        background-size: cover;
}
.example div:nth-of-type(1) {
        -webkit-filter: blur(2px);
        filter: blur(2px);
}
.example div:nth-of-type(2) {
        filter: blur(2px);
}

/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/5.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}
.old-photo {
        filter: sepia(100%);

}
.new-photo {
        filter: contrast(150%);

}

/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/5.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}
.old-photo {
        -webkit-filter: sepia(100%);

}
.new-photo {
        -webkit-filter: contrast(150%);

}
```

6.2. blur()

blur (Xpx) (where X is a number) gives the element a blur according to the number of pixels given in the declaration.

The only supported unit are pixels, and the higher their value, the more blur we give the element.

For example, to give a large blur the property we will construct as follows:
filter: blur(15px)

WORKS

1. 1. For the light-blur element, add the filter: blur (2px) property.
2. 2. Then add the same property, this time with the prefix -webkit-.
3. 3. In the rule for the element with the strong-blur class, the filter: blur (20px) property already exists.
4. 4. Add a prefix property to the rule

HINT

You should add properties to the light-blur element:
filter: blur(2px)
-webkit-filter: blur(2px)

And to the rule.strong-blur you should add a property:
-webkit-filter: blur(20px)

```
<!DOCTYPE html>
<html>
<head>
  <title>blur()</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>blur()</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>This is element with high
<b>blur</b> value:</p>
                    <div></div>
                    <p>This is element
with low <b>blur</b> value:</p>
                    <div></div>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="light-blur"></div>
      <div class="strong-blur"></div>
    </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/4.jpg')
bottom;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  background-color: white;
  padding: 20px;
  box-shadow: 2px 2px 10px #000;
  text-align: center;
```

```
}
.example,
.exercise {
   padding-bottom: 20px;
}
.example div {
        width: 150px;
        padding: 10px;
        border: 1px solid #34495e;
        margin: 10px auto;
        height: 200px;
        background:
url('/static/lessons/19.jpg') center;
        background-size: cover;
}
.example div:nth-of-type(1) {
        -webkit-filter: blur(12px);
        filter: blur(12px);
}
.example div:nth-of-type(2) {
        filter: blur(2px);
        -webkit-filter: blur(2px);
}

/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/6.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}
.strong-blur {
   filter: blur(20px);
}

/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/6.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}

.light-blur
{
   -webkit-filter: blur(2px);
}

.strong-blur {
   -webkit-filter: blur(20px);
}
```

6.3. brightness()

brightness () is responsible for brightening and darkening an element.

The property accepts percentages (%) in the range from 0 to infinity (theoretically, because at a higher value the element will become completely white and there will be no more changes). At the value 0, the element will be completely black.

For example, let's say that we want the element to be slightly dimmed, so let's use the following property:
filter: brightness(75%)

WORKS

1. 1. For the element with bright class, add the filter: brightness (150%) property.
2. 2. Also create the same property with the -webkit-prefix.
3. 3. For the dark class element, add the twin filter property for the existing one and give it a prefix -webkit-.

HINT

In the rule for the bright element you should add properties:
filter: brightness(150%)
-webkit-filter: brightness(150%)

In the rule for the dark element you should add a property:
-webkit-filter: brightness(50%)

```
<!DOCTYPE html>
<html>
<head>
  <title>brightness()</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>brightness()</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>This is element with a high
<b>brightness</b> value :</p>
      <div></div>
      <p>This is element
with a low <b>brightness</b> value:</p>
      <div></div>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="bright"></div>
      <div class="dark"></div>
    </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/4.jpg')
bottom;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  background-color: white;
  padding: 20px;
```

```
   box-shadow: 2px 2px 10px #000;
   text-align: center;
}
.example,
.exercise {
   padding-bottom: 20px;
}
.example div {
        width: 150px;
        padding: 10px;
        border: 1px solid #34495e;
        margin: 10px auto;
        height: 200px;
        background:
url('/static/lessons/19.jpg') center;
        background-size: cover;
}
.example div:nth-of-type(1) {
        -webkit-filter: brightness(150%);
        filter: brightness(150%);
}
.example div:nth-of-type(2) {
        filter: brightness(50%);
        -webkit-filter: brightness(50%);
}

/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/6.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}
.dark {
   filter: brightness(50%);
}

/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/6.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}
```

```
.bright
{
   -webkit-filter: brightness(150%);
}

.dark {
   -webkit-filter: brightness(50%);
}
```

6.4. contrast()

contrast () is used to set the contrast of an element.

Accepted units are percentages (%) in the range from 0 (completely black / gray element) to infinity (again theoretical infinity, because at a certain value the changes will cease to occur).

So if we want to change the contrast in our element, we will use the following property:
filter: contrast (250%)

In addition, check the examples for other values in the section **Example**.

WORKS

1. 1. Add the filter: contrast (150%) property to the element with the strong-contrast class.
2. 2. Also add a property with the -webkit-prefix.
3. 3. In the .light-contrast element selector, add the filter: contrast (80%) property.
4. 4. Also add the prefixed property.

HINT

In the element with the strong-contrast class, you should add the following properties:
filter: contrast (150%)
-webkit-filter: contrast (150%)

And in the light-contrast element:
filter: contrast (80%)
-webkit-filter: contrast(80%)

```
<!DOCTYPE html>
<html>
<head>
  <title>contrast()</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>contrast()</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>This is element with high
<b>contrast</b> value :</p>
                        <div></div>
                        <p>This is element
with low <b>contrast</b> value:</p>
                        <div></div>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="strong-contrast"></div>
      <div class="light-contrast"></div>
    </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/4.jpg')
bottom;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  background-color: white;
```

```
   padding: 20px;
   box-shadow: 2px 2px 10px #000;
   text-align: center;
}
.example,
.exercise {
   padding-bottom: 20px;
}
.example div {
        width: 150px;
        padding: 10px;
        border: 1px solid #34495e;
        margin: 10px auto;
        height: 200px;
        background:
url('/static/lessons/19.jpg') center;
        background-size: cover;
}
.example div:nth-of-type(1) {
        -webkit-filter: contrast(250%);
        filter: contrast(250%);
}
.example div:nth-of-type(2) {
        filter: contrast(50%);
        -webkit-filter: contrast(50%);
}

/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/6.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}

/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/6.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}

.strong-contrast
{
   -webkit-filter: contrast(150%);
}

.light-contrast
{
   -webkit-filter: contrast(80%);
}
```

6.5. grayscale()

grayscale () is responsible for giving the element a gray.

The units we use to determine this property are percentages (%) in the range of 0 to 100%.

At the value of 0, the picture does not change, and the higher the value, the more we desaturate the picture (we get rid of the colors).

For example, if we want our image to be black and white we will use the following property:
filter: grayscale (100%)

WORKS

1. 1. For an element with an old-photo class, give a property that will completely remove the color image.
2. 2. Also, remember to add the property with the prefix.
3. 3. For the new-photo element, set the filter: grayscale (30%) property.
4. 4. Also add a property with a prefix -webkit-.

HINT

For an element with an old-photo class, you should add the following properties:
filter: grayscale (100%)
-webkit-filter: grayscale (100%)

And for the new-photo element:
filter: greyscale (30%)
-webkit-filter: greyscale(30%)

```
<!DOCTYPE html>
<html>
<head>
  <title>grayscale()</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>grayscale()</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>This is element with a high
<b>grayscale</b> value :</p>
                    <div></div>
                    <p>This is element
with a low <b>grayscale</b> value:</p>
                    <div></div>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="old-photo"></div>
      <div class="new-photo"></div>
    </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/4.jpg')
bottom;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
```

```
   margin: 20px auto;
   background-color: white;
   padding: 20px;
   box-shadow: 2px 2px 10px #000;
   text-align: center;
}
.example,
.exercise {
   padding-bottom: 20px;
}
.example div {
        width: 150px;
        padding: 10px;
        border: 1px solid #34495e;
        margin: 10px auto;
        height: 200px;
        background:
url('/static/lessons/19.jpg') center;
        background-size: cover;
}
.example div:nth-of-type(1) {
        -webkit-filter: grayscale(100%);
        filter: grayscale(100%);
}
.example div:nth-of-type(2) {
        filter: grayscale(10%);
        -webkit-filter: grayscale(10%);
}

/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/7.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}

/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/7.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}
```

```
.old-photo
{
   -webkit-filter: grayscale(100%);
}

.new-photo
{
   -webkit-filter: grayscale(30%);
}
```

6.6. invert()

invert () is used to invert colors.

The property accepts percent units (%) in the range of 0 to 100%, where zero means no changes, and 100% is completely inverted colors.

An example of the rule partially reversing the colors may look as follows:
filter: invert(50%)

WORKS

1. 1. For a full-invert element, give the property a completely inverting color.
2. 2. Also add a property with -webkit prefix.
3. 3. To the rule with the selector .partial-invert, you copy the filter: invert (30%) property.
4. 4. Also add the prefixed property.

HINT

For a full-invert element, you should add the following properties:
filter: invert (100%)
-webkit-filter: invert (100%)

And for a partial-invert element:
filter: invert (30%)
-webkit-filter: invert(30%)

```
<!DOCTYPE html>
<html>
<head>
  <title>invert()</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>invert()</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>This is element with high
<b>invert</b> value :</p>
                        <div></div>
                        <p>This is element
with low <b>invert</b> value:</p>
                        <div></div>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="full-invert"></div>
      <div class="partial-invert"></div>
    </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/4.jpg')
bottom;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  background-color: white;
  padding: 20px;
  box-shadow: 2px 2px 10px #000;
  text-align: center;
}
```

```
.example,
.exercise {
   padding-bottom: 20px;
}
.example div {
        width: 150px;
        padding: 10px;
        border: 1px solid #34495e;
        margin: 10px auto;
        height: 200px;
        background:
url('/static/lessons/19.jpg') center;
        background-size: cover;
}
.example div:nth-of-type(1) {
        -webkit-filter: invert(100%);
        filter: invert(100%);
}
.example div:nth-of-type(2) {
        filter: invert(10%);
        -webkit-filter: invert(10%);
}

/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/6.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}

/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/6.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}

.full-invert
{
   -webkit-filter: invert(100%);
}
```

```
.partial-invert
{
   -webkit-filter: invert(30%);
}
```

6.7. saturate()

saturate () is responsible for the saturation of the element. As a quick reminder - saturation is the color saturation.

saturate () is another property taking percentages (%) in the range from 0 to infinity (And yes, you guessed it ... Again this is the theoretical infinity because at some value the element will stop changing.)

At the value of 0, we get a colorless element. 100% is the default value above which we will begin to saturate the element with colors. (The property works best between 0 - 200%.)

So if we want to saturate the element with colors, we can use the following property:
filter: saturate (150%)

WORKS

1. 1. For the element with the color-image class, add the filter: saturate (200%) property.
2. 2. Also add the prefixed property
3. 3. For the gloomy-image element, add the filter: saturate (60%) property.
4. 4. Also add a property with a prefix -webkit-.

HINT

The .colorful-image class selector should contain the following properties:

filter: saturate (200%);
-webkit-filter: saturate(200%);

```
<!DOCTYPE html>
<html>
<head>
  <title>saturate()</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>saturate()</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>This is element with high
<b>saturate</b> value :</p>
                        <div></div>
                        <p>This is element
with low <b>saturate</b> value:</p>
                        <div></div>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="colorful-image"></div>
      <div class="gloomy-image"></div>
    </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/4.jpg')
bottom;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
```

```
   margin: 20px auto;
   background-color: white;
   padding: 20px;
   box-shadow: 2px 2px 10px #000;
   text-align: center;
}
.example,
.exercise {
   padding-bottom: 20px;
}
.example div {
        width: 150px;
        padding: 10px;
        border: 1px solid #34495e;
        margin: 10px auto;
        height: 200px;
        background:
url('/static/lessons/19.jpg') center;
        background-size: cover;
}
.example div:nth-of-type(1) {
        -webkit-filter: saturate(130%);
        filter: saturate(130%);
}
.example div:nth-of-type(2) {
        filter: saturate(40%);
        -webkit-filter: saturate(40%);
}

/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/7.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}

/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/7.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}
```

```
.colorful-image
{
   -webkit-filter: saturate(200%);
}

.gloomy-image
{
   -webkit-filter: saturate(60%);
}
```

6.8. sepia()

sepia () is a property that allows you to give a sepia effect to an element.

sepia () assumes percent units (%) in the range of 0 to 100%, where 0 is the default value, and higher values intensify the sepia effect.

So let's say that we want to give the sepia effect to our element to make it look older. We will use the following property:
filter: sepia(100%)

WORKS

1. 1. Assign the filter: sepia (100%) property to the element with the old-photo class.
2. 2. Also create a property with a prefix.
3. 3. Add the filter: sepia (20%) property to the new-photo selector.
4. 4. Remember to add the value with the prefix -webkit-.

HINT

The .old-photo class selector should contain the following properties:

filter: sepia (100%);
-webkit-filter: sepia(100%);

```
<!DOCTYPE html>
<html>
<head>
  <title>sepia()</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>sepia()</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>This is element with a high
<b>sepia</b> value :</p>
                        <div></div>
                        <p>This is element
with a low <b>sepia</b> value:</p>
                        <div></div>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="old-photo"></div>
      <div class="new-photo"></div>
    </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/4.jpg')
bottom;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  background-color: white;
  padding: 20px;
  box-shadow: 2px 2px 10px #000;
  text-align: center;
}
.example,
.exercise {
```

```
   padding-bottom: 20px;
}
.example div {
        width: 150px;
        padding: 10px;
        border: 1px solid #34495e;
        margin: 10px auto;
        height: 200px;
        background:
url('/static/lessons/19.jpg') center;
        background-size: cover;
}
.example div:nth-of-type(1) {
        -webkit-filter: sepia(120%);
        filter: sepia(120%);
}
.example div:nth-of-type(2) {
        filter: sepia(10%);
        -webkit-filter: sepia(10%);
}

/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/6.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}

/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/6.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}

.old-photo
{
   -webkit-filter: sepia(100%);
}

.new-photo
{
   -webkit-filter: sepia(20%);
}
```

6.9. hue-rotate()

hue-rotate is used to change the colors of the object according to the color wheel.

Property as its determining units takes degrees (deg) in the range from 0 to 360deg (we can also use negative numbers).

0 is the default value, and any other will cause changes.

An example property can look like this: filter: hue-rotate (180deg)

WORKS

1. 1. In the rule with the selector .small-rotation, set the filter: hue-rotate (50deg) property.
2. 2. Also add a property with a prefix.
3. 3. In the rule with the big-rotation selector, set the filter: hue-rotate (200deg) property.
4. 4. Also create a property with a prefix -webkit-.

HINT

The .small-rotation class selector should contain the following properties:

filter: hue-rotate(50deg);
-webkit-filter: hue-rotate(50deg);

```
<!DOCTYPE html>
<html>
<head>
  <title>hue-rotate()</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>hue-rotate()</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>This is element with a high <b>hue-
rotate</b> value :</p>
                    <div></div>
                    <p>This is element
with a low <b>hue-rotate</b> value:</p>
                    <div></div>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="small-rotation"></div>
      <div class="big-rotation"></div>
    </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/4.jpg')
bottom;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
.container {
  width: 80%;
  margin: 20px auto;
  background-color: white;
  padding: 20px;
  box-shadow: 2px 2px 10px #000;
  text-align: center;
}
```

```
.example,
.exercise {
   padding-bottom: 20px;
}
.example div {
        width: 150px;
        padding: 10px;
        border: 1px solid #34495e;
        margin: 10px auto;
        height: 200px;
        background:
url('/static/lessons/19.jpg') center;
        background-size: cover;
}
.example div:nth-of-type(1) {
        -webkit-filter: hue-rotate(120deg);
        filter: hue-rotate(120deg);
}
.example div:nth-of-type(2) {
        filter: hue-rotate(10deg);
        -webkit-filter: hue-rotate(10deg);
}

/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/7.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}


/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/7.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}

.small-rotation
{
   -webkit-filter: hue-rotate(50deg);
}

.big-rotation
{
   -webkit-filter: hue-rotate(200deg);
}
```

6.10. How to use filters

Since you already know so many filters, you're probably wondering how you can use them effectively.

Let's start with the fact that, like most properties, those related to filters can also be grouped.

For example, if we want to give a sepia effect and lightly blur the element we will use the following property:
filter: blur (1px) sepia (100%) (we do not use separating characters)

Filters are ideal for creating universal reusable rules.

Let's say we want to give some elements data filters. So we create classes with the same properties for filters and give them to selected elements in HTML.

WORKS

1. 1. Create a rule with the .multiple-values selector.
2. 2. Add the filter: brightness (80%) contrast (120%) grayscale property in it (80%).
3. 3. Also add a property with a prefix -webkit-.

HINT

The correct code for this exercise looks as follows:

```
.multiple-values {
filter: brightness(80%) contrast(120%) grayscale(80%);
-webkit-filter: brightness(80%) contrast(120%) grayscale(80%);
}
```

```
<!DOCTYPE html>
<html>
<head>
  <title>Combining filters</title>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <div class="hero">
    <h1>Combining filters</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>This is element with a single filter value :</p>
                    <div></div>
                    <p>This is element with multiple values:</p>
                    <div></div>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="multiple-values"></div>
    </div>
  </div>
</body>
</html>
```

```
@import url(https://fonts.googleapis.com/css?family=Open+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #2980b9;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
  text-shadow: 1px 1px 1px #000;
}
.hero {
  background: url('/static/lessons/4.jpg') bottom;
  background-size: cover;
  min-height: 150px;
  box-shadow: 1px 1px 3px #000;
}
```

```
.container {
   width: 80%;
   margin: 20px auto;
   background-color: white;
   padding: 20px;
   box-shadow: 2px 2px 10px #000;
   text-align: center;
}
.example,
.exercise {
   padding-bottom: 20px;
}
.example div {
        width: 150px;
        padding: 10px;
        border: 1px solid #34495e;
        margin: 10px auto;
        height: 200px;
        background:
url('/static/lessons/19.jpg') center;
        background-size: cover;
}
.example div:nth-of-type(1) {
        -webkit-filter: hue-rotate(10deg);
        filter: hue-rotate(10deg);
}
.example div:nth-of-type(2) {
        filter: hue-rotate(120deg)
contrast(150%);
        -webkit-filter: hue-rotate(120deg)
contrast(150%);
}

/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/7.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}

/*EXERCISE*/

.exercise div {
   width: 50%;
   background: url('/static/lessons/7.jpg')
center;
   background-size: cover;
   height: 200px;
   margin: 15px auto;
}

.multiple-values
{
   -webkit-filter: brightness(80%)
contrast(120%) grayscale(80%);
}
```

7.1. linear-gradient

linear-gradient () is one of the values of the background property.

For the basic operation it needs two values: the initial color and the final color.

For example:
background: linear-gradient (red, green)

Slightly more complex property due to which we will create a three-color gradient:
background: linear-gradient (red, green, yellow)

By default, the gradient moves from top to bottom. However, we can control this direction.

For example:
background: linear-gradient (left, blue, green) (gradient from left to right)

background: linear-gradient (bottom right, blue, green) (gradient from bottom-right to upper-left corner)

We can also use the steps:
background: linear-gradient (260deg, green, blue) (gradient from top to bottom)
background: linear-gradient (190deg, green, blue) (gradient from right to left)

NOTE: For gradients, we also have to add prefixes, but only one -webkit - is enough.

An example property with a prefix:
background: -webkit-linear-gradient(red, blue)

WORKS

1. 1. In the rule for the gradient-container element, create a linear gradient.
2. 2. It has to move from left to right.
3. 3. Its initial color is to be green, which will turn blue until it turns red in the end.
4. 4. Also add a property with a prefix.
5. 5. If the task seems too difficult, take a look at the hints.

HINT

In the rule with the .gradient-container selector, you should add the property:

background: linear-gradient(left, green, blue, red)

background: -webkit-linear-gradient(left, green, blue, red)

```
<!DOCTYPE html>
<html>
<head>
  <title>Gradients</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>Gradients</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>Example gradient:</p>
                    <div></div>
                    <p>Second example
gradient:</p>
                    <div></div>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="gradient-container"></div>
    </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
```

```
   box-sizing: border-box;
}
body {
   padding: 0;
   margin: 0;
   background-color: #05D857;
   font-family: 'Open Sans', sans-serif;
}
h1 {
   margin: 0;
   color: #fff;
   text-align: center;
   padding: 50px;
   font-size: 200%;
   text-shadow: 1px 1px 0px #000;
}
.hero {
   background: url('/static/lessons/7.jpg')
0/cover;
   background-size: cover;
   min-height: 150px;
   box-shadow: 1px 1px 3px #000;
}
.container {
   width: 80%;
   margin: 20px auto;
   background-color: white;
   padding: 20px;
   box-shadow: 2px 2px 10px #000;
   text-align: center;
}
.example,
.exercise {
   padding-bottom: 20px;
}
.example div {
        width: 250px;
        padding: 10px;
        box-shadow: 0px 0px 10px #34495e;
        margin: 10px auto;
        height: 200px;
        background-size: cover;
}
.example div:nth-of-type(1) {
        background: linear-gradient(260deg,
#55ee70, #55acee)
}
.example div:nth-of-type(2) {
        background: linear-gradient(60deg,
green, #00fc48)
}

/*EXERCISE*/

.exercise div {
   width: 250px;
   background-size: cover;
        box-shadow: 0px 0px 10px #34495e;
        height: 200px;
   margin: 20px auto;
}

/*EXERCISE*/

.exercise div {
   width: 250px;
   background-size: cover;
        box-shadow: 0px 0px 10px #34495e;
        height: 200px;
   margin: 20px auto;
}

.gradient-container
{
   background: -webkit-linear-gradient(left,
green, blue, red);
}
```

7.2. radial-gradient

radial-gradient () is a brotherly linear-gradient value. However, they are not twin brothers ...

A radial gradient (radial) will be created using a radial-gradient.

If you find it difficult to imagine such a gradient, think about a stone thrown into the water and imagine how waves travel from one point (you can also just see in the preview).

For proper operation we need two values: the initial and final color. Of course, we can expand the property, for example:
background: radial-gradient (red, blue, green, yellow)

In addition, you should know that the shape of our gradient is an ellipse, but we can change this default value to a circle:
background: radial-gradient (circle, red, blue)

If there are not enough possibilities for you, then we can always change the position of the point from which the gradient is "coming out":
backgrund: radial-gradient (at x% y%, red, green)

The x% and y% values correspond to the shift in the X axis and the Y axis. At this keyword, meaning that successive values mean the place from which the gradient "propagates".

So if our property looks like this:
background: radial-gradient (at 100% 100%, red, green)
The center of the gradient will be in the lower right corner.

WORKS

1. 1. In the rule with the .radial-gradient-container select the background property: radial-gradient(at 100% 100%, yellow, blue).
2. 2. Also create a property with a prefix -webkit-.

HINT

The radial gradient code looks as follows:

background: radial-gradient(at 100% 100%, yellow, blue);

```
<!DOCTYPE html>
<html>
<head>
  <title>Gradients</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>Gradients</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>Example gradient:</p>
                    <div></div>
                    <p>Second example
gradient:</p>
                    <div></div>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="radial-gradient-
container"></div>
    </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #05D857;
  font-family: 'Open Sans', sans-serif;
```

```
}
h1 {
   margin: 0;
   color: #fff;
   text-align: center;
   padding: 50px;
   font-size: 200%;
   text-shadow: 1px 1px 0px #000;
}
.hero {
   background: url('/static/lessons/7.jpg')
0/cover;
   background-size: cover;
   min-height: 150px;
   box-shadow: 1px 1px 3px #000;
}
.container {
   width: 80%;
   margin: 20px auto;
   background-color: white;
   padding: 20px;
   box-shadow: 2px 2px 10px #000;
   text-align: center;
}
.example,
.exercise {
   padding-bottom: 20px;
}
.example div {
        width: 250px;
        padding: 10px;
        box-shadow: 0px 0px 10px #34495e;
        margin: 10px auto;
        height: 200px;
        background-size: cover;
}
.example div:nth-of-type(1) {
        background: radial-gradient(100%
100%, #55ee70, #55acee);
        background: -webkit-radial-
gradient(100% 100%, #55ee70, #55acee);
}
.example div:nth-of-type(2) {
        background: radial-gradient(100%
100%, #00ffff, #00fa48);
        background: -webkit-radial-
gradient(100% 100%, #00ffff, #00fa48);
}

/*EXERCISE*/
```

```
.exercise div {
   width: 250px;
   background-size: cover;
        box-shadow: 0px 0px 10px #34495e;
        height: 200px;
   margin: 20px auto;
}

/*EXERCISE*/

.exercise div {
   width: 250px;
   background-size: cover;
        box-shadow: 0px 0px 10px #34495e;
        height: 200px;
   margin: 20px auto;
}

.radial-gradient-container
{
   background: radial-gradient(at 100% 100%,
yellow, blue);
   background: -webkit-radial-gradient(at
100% 100%, yellow, blue);
}
```

7.3. repeating-linear-gradient

repeating-linear-gradient allows us to create a repeatable pattern using a gradient.

repeating-linear-gradient is simply an "improved" version of linear-gradient properties.

As an improved version, it gives us quite interesting possibilities. For example:
background: repeating-linear-gradient (-45deg, green 0, green 5%, yellow 0, yellow 10%)

It creates a repeating green-yellow-blue pattern with sharp edges and visible borders between colors (check in preview).

If you are curious about how this property works, take a look carefully:

-45deg declares the direction in which the gradient is moving
green 0 sets the beginning of a green color
green 5% up to 5% of the element's width
yellow 0 resets the position of the yellow bar
yellow 10% makes the yellow color take up to 10% of the element's width

Then the sequence is repeated until the entire available space is full and we get a repeating gradient.

WORKS

1. 1. In the selector with the .linear-gradient-container rule, create the background property.
2. 2. And give it a repeating-linear-gradient value.
3. 3. Complete the value as follows: left, white 10%, orange 20%, white 30%.
4. 4. Also add a property with a prefix -webkit-.

HINT

The whole thing should look like this:

background: repeating-linear-gradient(left, white 10%, orange 20%, white 30%)

```
<!DOCTYPE html>
<html>
<head>
  <title>Gradients</title>
  <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
  <div class="hero">
    <h1>Gradients</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>Example gradient:</p>
                        <div></div>
                        <p>Second example
gradient:</p>
                        <div></div>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="linear-gradient-
container"></div>
    </div>
  </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #05D857;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
```

```
    padding: 50px;
    font-size: 200%;
    text-shadow: 1px 1px 0px #000;
}
.hero {
    background: url('/static/lessons/7.jpg')
0/cover;
    background-size: cover;
    min-height: 150px;
    box-shadow: 1px 1px 3px #000;
}
.container {
    width: 80%;
    margin: 20px auto;
    background-color: white;
    padding: 20px;
    box-shadow: 2px 2px 10px #000;
    text-align: center;
}
.example,
.exercise {
    padding-bottom: 20px;
}
.example div {
        width: 250px;
        padding: 10px;
        box-shadow: 0px 0px 10px #34495e;
        margin: 10px auto;
        height: 200px;
        background-size: cover;
}
.example div:nth-of-type(1) {
        background: repeating-linear-
gradient(-45deg, #55ee70 40%, #55acee 50%);
        background: -webkit-repeating-linear-
gradient(-45deg, #55ee70 40%, #55acee 50%);
}
.example div:nth-of-type(2) {
        background: repeating-linear-
gradient(90deg, #00ff48 80%, #52ba65 100%);
        background: -webkit-repeating-linear-
gradient(90deg, #00ff48 80%, #52ba65 100%);
}

/*EXERCISE*/

.exercise div {
    width: 250px;
    background-size: cover;
        box-shadow: 0px 0px 10px #34495e;
        height: 200px;
    margin: 20px auto;
}

/*EXERCISE*/

.exercise div {
    width: 250px;
    background-size: cover;
        box-shadow: 0px 0px 10px #34495e;
        height: 200px;
    margin: 20px auto;
}

.linear-gradient-container
{
    background: repeating-linear-gradient(left,
white 10%, orange 20%, white 30%);
    background: -webkit-repeating-linear-
gradient(left, white 10%, orange 20%, white
30%);
}
```

7.4. repeating-radial-gradient

repeating-radial-gradient is a specific extension of the radial-gradient property with the possibility of multiple gradient execution.

When creating properties, we can use the previous values used to create a radial gradient, but we also declare the spread of colors.

What should such a property look like? Here we have an example:

background: repeating-radial-gradient (circle, green 10px, yellow 25px, blue 35px)

The principle of operation is identical to the repeating-linear-gradient properties.

WORKS

1. 1. For the element rule with the radial-gradient-container class, add the background property.
2. 2. Assign the value of repeating-radial-gradient (circle, white 0, white 15px, black 0, black 30px) in this property.
3. 3. Also, remember to add the same property with the prefix -webkit-.

HINT

The code for a repeating radial gradient looks as follows:

background: repeating-radial-gradient (circle, white 0, white 15px, black 0, black 30px);

Do not forget to add the second version with the prefix!

```
<!DOCTYPE html>
<html>
<head>
   <title>Gradients</title>
   <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
   <div class="hero">
      <h1>Gradients</h1>
   </div>
   <div class="container">
      <div class="example">
         <h2>Example:</h2>
         <p>Example gradient:</p>
                              <div></div>
                              <p>Second example
gradient:</p>
                              <div></div>
      </div>
      <div class="exercise">
         <h2>Exercise:</h2>
         <p>Practice!</p>
         <div class="radial-gradient-
container"></div>
      </div>
   </div>
</body>
</html>
```

```
@import
url(https://fonts.googleapis.com/css?family=O
pen+Sans);
* {
   box-sizing: border-box;
}
body {
   padding: 0;
   margin: 0;
   background-color: #05D857;
   font-family: 'Open Sans', sans-serif;
}
h1 {
   margin: 0;
   color: #fff;
   text-align: center;
   padding: 50px;
   font-size: 200%;
   text-shadow: 1px 1px 0px #000;
}
.hero {
   background: url('/static/lessons/7.jpg')
0/cover;
   background-size: cover;
   min-height: 150px;
```

```
   box-shadow: 1px 1px 3px #000;
}
.container {
   width: 80%;
   margin: 20px auto;
   background-color: white;
   padding: 20px;
   box-shadow: 2px 2px 10px #000;
   text-align: center;
}
.example,
.exercise {
   padding-bottom: 20px;
}
.example div {
        width: 250px;
        padding: 10px;
        box-shadow: 0px 0px 10px #34495e;
        margin: 10px auto;
        height: 200px;
        background-size: cover;
}
.example div:nth-of-type(1) {
        background: repeating-radial-
gradient(circle, #55ee70 40%, #55acee 50%);
        background: -webkit-repeating-radial-
gradient(circle, #55ee70 40%, #55acee 50%);
}
.example div:nth-of-type(2) {
        background: repeating-radial-
gradient(circle, #00ff48 80%, #52ba65 100%);
        background: -webkit-repeating-radial-
gradient(circle, #00ff48 80%, #52ba65 100%);
}

/*EXERCISE*/

.exercise div {
   width: 250px;
   background-size: cover;
        box-shadow: 0px 0px 10px #34495e;
        height: 200px;
   margin: 20px auto;
}

/*EXERCISE*/

.exercise div {
   width: 250px;
   background-size: cover;
        box-shadow: 0px 0px 10px #34495e;
        height: 200px;
   margin: 20px auto;
}

.radial-gradient-container
{
   background: repeating-radial-
gradient(circle, white 0, white 15px, black 0,
black 30px);
   background: -webkit-repeating-radial-
gradient(circle, white 0, white 15px, black 0,
black 30px);
}
```

7.5. Complex gradients

Since you already know the basic and slightly more complex properties of gradients, we can practice their creation.

Let's take on something that seems to be complicated, but the effect will be obtained without much difficulty thanks to the known properties concerning gradients.

You probably remember a variety of background pages in the form of grids or circles - both of these effects can be obtained with simple gradients, without the need for an additional page load the next picture.

Now we will deal with the creation of this second version using circles.

WORKS

1. 1. Since we want to create a gradient based on circles, we will need a radial gradient.
2. 2. So, create a rule for the element, radial-gradient-container.
3. 3. Create a background property in it.
4. 4. And give it a radial-gradient value (circle, # 000 0, # 55ee70 50%, # 00ff48 0, # 000 100%).
5. 5. Also create a property with the -webkit prefix.
6. 6. Finally, add the background-size property with the value of 15px.

HINT

Your code for the .radial-gradient-container should look like:

```
.radial-gradient-container {
background: radial-gradient (circle, # 000 0, # 55ee70 50%, # 00ff48 0, # 000 100%);
background: -webkit-radial-gradient(circle, #000 0, #55ee70 50%, #00ff48 0, #000 100%);
background-size: 15px;
}
```

```
<!DOCTYPE html>
<html>
<head>
  <title>Gradients</title>
  <link rel="stylesheet" type="text/css" href="style.css">
</head>
<body>
  <div class="hero">
    <h1>Gradients</h1>
  </div>
  <div class="container">
    <div class="example">
      <h2>Example:</h2>
      <p>Example gradient:</p>
                        <div></div>
                        <p>Second example gradient:</p>
                        <div></div>
    </div>
    <div class="exercise">
      <h2>Exercise:</h2>
      <p>Practice!</p>
      <div class="radial-gradient-container"></div>
    </div>
  </div>
</body>
</html>
```

```
@import url(https://fonts.googleapis.com/css?family=Open+Sans);
* {
  box-sizing: border-box;
}
body {
  padding: 0;
  margin: 0;
  background-color: #05D857;
  font-family: 'Open Sans', sans-serif;
}
h1 {
  margin: 0;
  color: #fff;
  text-align: center;
  padding: 50px;
  font-size: 200%;
```

```
   text-shadow: 1px 1px 0px #000;
}
.hero {
   background: url('/static/lessons/7.jpg')
0/cover;
   background-size: cover;
   min-height: 150px;
   box-shadow: 1px 1px 3px #000;
}
.container {
   width: 80%;
   margin: 20px auto;
   background-color: white;
   padding: 20px;
   box-shadow: 2px 2px 10px #000;
   text-align: center;
}
.example,
.exercise {
   padding-bottom: 20px;
}
.example div {
        width: 250px;
        padding: 10px;
        box-shadow: 0px 0px 10px #34495e;
        margin: 10px auto;
        height: 200px;
        background-size: cover;
}
.example div:nth-of-type(1) {
        background: repeating-radial-
gradient(circle, #55ee70 40%, #55acee 50%);
        background: -webkit-repeating-radial-
gradient(circle, #55ee70 40%, #55acee 50%);
}
.example div:nth-of-type(2) {
        background: repeating-radial-
gradient(circle, #00ff48 80%, #52ba65 100%);
        background: -webkit-repeating-radial-
gradient(circle, #00ff48 80%, #52ba65 100%);
}

/*EXERCISE*/

.exercise div {
   width: 250px;
        box-shadow: 0px 0px 10px #34495e;
        height: 200px;
   margin: 20px auto;
}
```


```
/*EXERCISE*/

.exercise div {
   width: 250px;
        box-shadow: 0px 0px 10px #34495e;
        height: 200px;
   margin: 20px auto;
}
```


```
.radial-gradient-container
{
   background: radial-gradient(circle, #000 0,
#55ee70 50%, #00ff48 0, #000 100%);
   background: -webkit-radial-gradient(circle,
#000 0, #55ee70 50%, #00ff48 0, #000 100%);
   background-size: 15px;
}
```

8.1. We start - basic styles.

In this part of the course we will focus on building an animated charging icon consisting of two interpenetrating rings.

It's a bit unusual, but this time we will not sacrifice the whole exercise to build an HTML structure - why? This time we just need one HTML element - all the rest will be created using CSS and animation.

However, before we get to animate our element, we need to give it the required look and put it in the right place on the page.

Let's move to coding!

WORKS

1. 1. Let's start by creating one element with the class circle and we can go to styling.
2. 2. Let's get to the styling - create a group selector for <html> and <body> and assign it a height of 100%, then create a selector for <body> and assign the property overflow: hidden in it.
3. 3. Let's consider the circle element's styling, give it an absolute position, top and left with 50% value, transform with translate value (-50%, -50%) and height and width of 400px.

HINT

The transform: translate property (-50%, -50%) in combination with top and left with a value of 50% and the absolute position centers the element vertically and horizontally.

```
<!DOCTYPE html>
<html>
<head>
   <title>Custom animated preloader</title>
   <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
```

EXERCISE

```
<!DOCTYPE html>
<html>
<head>
   <title>Custom animated preloader</title>
   <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
   <div class="circle"></div>
</body>
</html>

html, body
{
 height: 100%;
}

body
{
 overflow: hidden;
}

.circle
{
 position: absolute;
 top: 50%;
 left: 50%;
 transform: translate(-50%, -50%);
 width: 400px;
 height: 400px;
}
```

8.2. Creating rings

Now we will focus on pseudo-elements, which will soon become our rings.

In the next exercises, we will gradually develop the created elements, and at the end we will add animation to them.

WORKS

1. 1. Create a group selector for the .circle: before and circle: after pseudo-elements, and the following properties: content: "", border-radius of 50%, absolute position, top and left with the value 0, height and width 100%, and transform -origin with center value.

HINT

The transform-origin property specifies the point against which the actions from the transform property are executed.

EXERCISE

```
html, body
{
  height: 100%;
}

body
{
  overflow: hidden;
}

.circle
{
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    width: 400px;
    height: 400px;
}

.circle:before
{
    position: absolute;
    content: "";
    top: 0;
    left: 0;
    height: 100%;
    width: 100%;
    border-radius: 50%;
    transform-origin: center;
}

.circle:after
{
    position: absolute;
    content: "";
    top: 0;
    left: 0;
    height: 100%;
    width: 100%;
    border-radius: 50%;
    transform-origin: center;
}
```

8.3. We style rings!

In this exercise, we will give our rings with pseudo-elements the right styles that will make them look really spectacular.

Due to the fact that the required properties are quite long, we have prepared two ready-made code fragments for you, based on the box-shadow properties.

The box-shadow property for the .circle: before pseudo-element:

box-shadow: inset 0 25px 0 rgba(0, 250, 250, 0.6), inset 25px 0 0 rgba(0, 200, 200, 0.6), inset 0 -25px 0 rgba(0, 150, 200, 0.6), inset -25px 0 0 rgba(0, 200, 250, 0.6);

Właściwość box-shadow dla pseudoelementu .circle:after:
box-shadow: inset 0 25px 0 rgba(250, 250, 0, 0.6), inset 25px 0 0 rgba(250, 200, 0, 0.6), inset 0 -25px 0 rgba(250, 150, 0, 0.6), inset -25px 0 0 rgba(250, 100, 0, 0.6);

WORKS

1. 1. Create separate selectors for .circle: before and .circle: after.
2. 2. In the selected selectors, add the appropriate properties listed above.
3. 3. Pseudo-element: before for .circlenad also have animation property with counterclockwise value 1.5s -0.5s linear infinite
4. 4. Pseudo-element: after for .circlenad the same animation property as: before, but replace the name of counterclockwise animation with clockwise.

HINT

The box-shadow property specifies the shadow applied to the element.

EXERCISE

```
html, body
{
  height: 100%;
}

body
{
  overflow: hidden;
}

.circle
{
   position: absolute;
   top: 50%;
   left: 50%;
   transform: translate(-50%, -50%);
   width: 400px;
   height: 400px;
}

.circle:before
{
   position: absolute;
   content: "";
   top: 0;
   left: 0;
   height: 100%;
   width: 100%;
   border-radius: 50%;
   transform-origin: center;
   animation: counterclockwise 1.5s -0.5s
linear infinite;
}

.circle:before
{
   box-shadow: inset 0 25px 0 rgba(0, 250,
250, 0.6), inset 25px 0 0 rgba(0, 200, 200, 0.6),
inset 0 -25px 0 rgba(0, 150, 200, 0.6), inset -
25px 0 0 rgba(0, 200, 250, 0.6);
}

.circle:after
{
   position: absolute;
   content: "";
   top: 0;
   left: 0;
   height: 100%;
   width: 100%;
   border-radius: 50%;
   transform-origin: center;
```

```
    animation: clockwise 1.5s -0.5s linear
infinite;
}

.circle:after
{
    box-shadow: inset 0 25px 0 rgba(250, 250,
0, 0.6), inset 25px 0 0 rgba(250, 200, 0, 0.6),
inset 0 -25px 0 rgba(250, 150, 0, 0.6), inset -
25px 0 0 rgba(250, 100, 0, 0.6);
}
```

8.4. Time for animations!

We have already created our rings, they just lack ... life :)

In this exercise, we will put together previously created elements in motion and give them properties that will make the element we write look much better:)

WORKS

1. 1. Create @keyframes with the name clockwise and place the following properties in it - for progress 0%, assign a transform with the value rotateZ (0deg) scaleX (1) scaleY (1). For 50% progress, assign the transform with the value rotateZ (180deg) scaleX (0.80) scaleY (0.95), and for 100% of the progress assign the transform: rotateZ (360deg) scaleX (1) scaleY (1).
2. 2. Then create @keyframes with the name counterclockwise and assign him transform for progress 0%: rotateZ (0deg) scaleX (1) scaleY (1), for progress 50% transform: rotateZ (-180deg) scaleX (0.95) scaleY (0.80) and for 100% progress, give it a transform with the value of rotateZ (-360deg) scaleX (1) scaleY (1).

HINT

The correct code for the first @keyframes should look as follows:@keyframes clockwise {
0% {
transform: rotateZ(0deg) scaleX(1) scaleY(1);
}
50% {
transform: rotateZ(180deg) scaleX(0.80) scaleY(0.95);
}
100% {
transform: rotateZ(360deg) scaleX(1) scaleY(1);
}
}

EXERCISE

```
<!DOCTYPE html>
<html>
<head>
   <title>Custom animated preloader</title>
   <link rel="stylesheet" type="text/css"
href="style.css">
</head>
<body>
   <div class="circle"></div>
</body>
</html>

html, body
{
 height: 100%;
}

body
{
 overflow: hidden;
}

.circle
{
   position: absolute;
   top: 50%;
   left: 50%;
   transform: translate(-50%, -50%);
   width: 400px;
   height: 400px;
}

.circle:before
{
   position: absolute;
   content: "";
   top: 0;
   left: 0;
   height: 100%;
   width: 100%;
   border-radius: 50%;
   transform-origin: center;
   animation: counterclockwise 1.5s -0.5s
linear infinite;
}

.circle:before
{
```

```
   box-shadow: inset 0 25px 0 rgba(0, 250,
250, 0.6), inset 25px 0 0 rgba(0, 200, 200, 0.6),
inset 0 -25px 0 rgba(0, 150, 200, 0.6), inset -
25px 0 0 rgba(0, 200, 250, 0.6);
}

.circle:after
{
   position: absolute;
   content: "";
   top: 0;
   left: 0;
   height: 100%;
   width: 100%;
   border-radius: 50%;
   transform-origin: center;
   animation: clockwise 1.5s -0.5s linear
infinite;
}

.circle:after
{
   box-shadow: inset 0 25px 0 rgba(250, 250,
0, 0.6), inset 25px 0 0 rgba(250, 200, 0, 0.6),
inset 0 -25px 0 rgba(250, 150, 0, 0.6), inset -
25px 0 0 rgba(250, 100, 0, 0.6);
}

@keyframes clockwise
{
   0%
   {
      transform: rotateZ(180deg) scaleX(1)
scaleY(1);
   }

   50%
   {
      transform: rotateZ(180deg) scaleX(0.80)
scaleY(0.95);
   }

   100%
   {
      transform: rotareZ(360deg) scaleX(1)
slaceY(1);
   }
}

@keyframes counterclockwise
{
   0%
   {
      transform: rotateZ(0deg) scaleX(1)
scaleY(1);
   }

   50%
   {
      transform: rotateZ(-180deg) scaleX(0.95)
scaleY(0.85);
   }

   100%
   {
      transform: rotareZ(-360deg) scaleX(1)
slaceY(1);
   }
}
```