# Grafika i animacje w CSS

W tym kursie dowiesz się, jak wykorzystać CSS do przekształcenia elementów struktury HTML, wykonania przejść i animacji oraz jak używać filtrów oraz gradientów.

## Spis treści

**Wprowadzenie do animacji CSS**

1/1

# 1.1. Wprowadzenie do animacji

Ten kurs ma na celu nauczyć Cię jak tworzyć animacje CSS oraz efekty graficzne zachowując dobre praktyki i kompatybilność z różnymi przeglądarkami.

Poznamy dogłębnie zasady działania przejść w CSS, transformacji 2D oraz 3D, nie zabraknie także rozbudowanej lekcji o filtrach i gradientach.

Dużą część kursu stanowią także animacje CSS, w których od podstaw pokazujemy, jak budować świetne animacje bez dużego nakładu pracy.

**ZADANIA**Po prostu przejdź do kolejnego zadania :)

```
<!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. Co zmieni przejście - transition-property

W tej części kursu nauczymy się korzystać z przejść (ang. *transition*) - jest to prosty sposób na animowanie zmian w różnych właściwościach CSS.

Zacznijmy od transition-property. Jest to podstawowa właściwość przejść określająca, które ze zmienionych wartości CSS mają być animowane.

Idealny przykład działania możemy zobaczyć w oknie podglądu. Jeśli ustalamy kilka właściwości w naszym elemencie i chcemy, aby wszystkie z nich zostały zmienione w trakcie transformacji, możemy pominąć właściwość transition-property.

Jeśli jednak chcemy, żeby tylko wybrane właściwości podlegały zmianie, to musimy dodać je jako wartości dla transition-property (wartości oddzielamy przecinkami) np:

transition-property: width, height.

Zanim jednak zobaczymy płynny efekt naszego przejścia, musimy poznać jeszcze jedną właściwość - transition-duration, o której powiemy sobie w następnym ćwiczeniu :)

**ZADANIA**

1. Dla elementu o klasie .practice dodaj właściwość transition-property.
2. transition-property powinno mieć następujące właściwości: width, height, margin-left.

**PODPOWIEDŹ**

transition-property powinno zawierać: 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=Open+Sa
ns);
* {
  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. Czas trwania przejść - transition-duration

W poprzednim ćwiczeniu poznaliśmy transition-property - teraz weźmy pod lupę transition-duration. Wartość podana dla tej właściwości to po prostu czas, w jakim ma odbyć się cała przemiana.

Czas ten najczęściej określamy w **sekundach**, więc jeśli określimy transition-duration: 0.5s to przemiana wykona się w pół sekundy. Możemy także użyć milisekund (i tu, zamiast 0.5s napisalibyśmy 500ms).

Brzmi trochę skomplikowanie? To tylko pozory, najedź na przykład w oknie podglądu i zobacz na własne oczy jakie to proste!

**ZADANIA**

1. Dla elementu o klasie .practice dodaj właściwość transition-duration
2. transition-duration powinno mieć wartość 2s.

**PODPOWIEDŹ**

Kod naszego selektora dla .practice powinien zawierać następujące właściwości:
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=Open+Sa
ns);
* {
  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. Tempo przejścia czyli transition-timing-function

Kolejną ciekawą właściwością, o której należy wspomnieć, jest transition-timing-function. Jej zadaniem jest określenie szybkości zachodzenia zmian na danych etapach (np. start, środek i koniec). Właściwość ta nie wpływa w żaden sposób na długość wykonywania przemian (transition-duration).

Przejdźmy więc do dokładniejszego poznania właściwości. Domyślną jej wartością jest ease(szybki start i zakończenie z powolnym środkiem), pozostałe popularne właściwości to:
linear (stałe tempo)
ease-in-out (powolny start i koniec)
cubic-bezier(x,x,x,x)(gdzie x to liczba od **0**do **1**) (dowolnie stworzona przez nas funkcja - ta wartość nie jest tak często używana, ale dobrze wiedzieć, że mamy taką możliwość).

**ZADANIA**

1. Utwórz regułę z selektorem .exercise:hover > .practice-linear i nadaj mu właściwość transition-timing-function: linear.
2. Następnie utwórz regułę z selektorem .exercise:hover > .practice-ease.
3. Nadaj mu właściwość transition-timing-function: ease-in; .

**PODPOWIEDŹ**

Poprawny kod do tego ćwiczenia wygląda następująco:

```
.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. Opóźnienie przejścia - transition-delay

Ostatnią właściwością dotyczącą przejść, którą poznamy, będzie transition-delay.

Jak sama nazwa wskazuje, jest to opóźnienie zachodzenia przemian. Dodatkowo istotne jest, że czas opóźnienia transformacji nie wpływa w żaden sposób na długość jej samej.

Jednostki, jakie używamy do określania transition-delay to sekundy, więc przykładowe: transition-delay: 3s opóźni start wykonania przemiany o 3 sekundy.

Więcej przykładów zobaczysz po najechaniu na okno podglądu w części **example**.

**ZADANIA**

1. Sprawdź zawartość pliku *style.css*, znajdziesz w nim trzy reguły bez właściwości.
2. Pierwszemu elementowi ustaw czas opóźnienia przejścia na pół sekundy.
3. Drugiemu jedną sekundę.
4. Trzeciemu trzy czwarte sekundy.

**PODPOWIEDŹ**

Reguły powinny mieć kolejne warotści właściwości transition-delay:
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. Wszystko w jednym - transition

Wiesz już jakie istnieją właściwości związane z przemianami, więc spróbujmy je teraz zgrupować za pomocą jednej właściwości transition.

Do podstawowego działania właściwości potrzebujemy: transition-property oraz transition-duration, więc zbierając wszystko w jednej właściwości otrzymamy:
transition: all 1s;

Bardziej złożona właściwość mogłaby wyglądać następująco:
transition: all 1s linear 2s

Podczas tworzenia takich właściwości musimy przyjąć następujący wzorzec:
transition: property duration timing-function delay

Pójdźmy krok dalej i stwórzmy jeszcze bardziej rozbudowaną właściwość:
margin-left 1s, color 1.5s, width 2s

Właściwość transition po prostu przyspiesza nam kodowanie - łatwiej jest napisać jedną, krótką właściwość z kilkoma wartościami niż kilka długich właściwości, prawda? :)

**ZADANIA**

1. W regule o selektorze .exercise:hover > .practice utwórz właściwość transition.
2. I nadaj jej margin-left 1s linear.
3. Po przecinku umieść: background-color 1s ease-in-out.

**PODPOWIEDŹ**

Cała właściwość powinna wyglądać następująco:
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=Open+Sa
ns);
* {
  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 z efektami specjalnymi

Skoro poznaliśmy już podstawy właściwości przemian CSS, możemy przejść do praktyki.

Zaczniemy powoli i wraz z postępami w kursie, będziemy tworzyć coraz bardziej zaawansowane animacje i transformacje.

Na początek zajmiemy się nadaniem elementowi button unikalnego efektu po najechaniu kursorem po czym określimy, które właściwości mają się zmieniać i w jakim tempie.

**ZADANIA**

1. Dla elementu o klasiepractice-buttonbędącego w stanie :hover dodaj właściwość padding: 30px.
2. Dla tego samego elementu w stanie :hoverdodaj kolejną właściwość, tym razem: font-size o wartości 120%.
3. Dla elementu o klasie practice-button dodaj właściwość określającą długość przemiany na 0.5s.
4. Następnie dodaj właściwość transition-property i nadaj jej właściwości padding, font-size.
5. Najedź na guzik i zobacz jakie przemiany zachodzą!

**PODPOWIEDŹ**

Długość przemiany ustalimy poprzez właściwość: transition-duration.
Druga właściwość elementu o klasie practice-button powinna wyglądać następujaco: 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=Open+Sa
ns);
* {
  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()

Skoro znasz już podstawy przejść w CSS, pora na poznanie właściwości, które można z nimi połączyć.

Zaczniemy od omówienia wartości właściwości transform o wdzięcznej nazwie translate().

translate(x,y) służy przemieszczeniu elementu zgodnie z podanymi parametrami w nawiasie (x - dla osi X, poziomej oraz y - dla osi Y, pionowej).

Dla przykładu powiedzmy, że chcemy przemieścić element o 100px w prawo i o 100px w dół - nasz kod mógłby wyglądać następująco:
transform: translate(100px, 100px)

Pamiętaj, że wartości ujemne na osi X przemieszczą element w lewo, a wartości ujemne na osi Y do góry.

**ZADANIA**

1. W selektorze dla .practice-button dodaj właściwość transition o wartości .5s all.
2. Utwórz selektor .practice-button:hover i dodaj w nim właściwość transform o wartości translate(30px, 20px).

**PODPOWIEDŹ**

Właściwy kod dla .practice-button z pseudoklasą :hover wygląda następująco:

```
.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=Open+Sa
ns);
* {
  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.

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

Budowa tej wartości 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.

**ZADANIA**

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

**PODPOWIEDŹ**

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>
      <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=Open+Sa
ns);
* {
  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()

Skoro wiesz już jak przemieszczać i obracać obiekty, przejdźmy do skalowania ich.

W tym celu użyjemy właściwości transform: scale(x) (gdzie x to liczba nieujemna).

Załóżmy, że chcemy powiększyć nasz element ponad dwukrotnie, więc użyjemy właściwości:
transform: scale(2.5)

Ważną informacją jest, że jeśli skalujesz obiekt w dół (powiedzmy, że chcesz zmniejszyć go o połowę), to warto wiedzieć, że mimo iż zmniejszy się do oczekiwanego rozmiaru, to będzie zabierał tyle samo miejsca (pozostanie wokół niego pusta przestrzeń, której nie zajmą inne elementy).

**ZADANIA**

1. Dla elementu o klasie scale-big nadaj właściwość powiększającą go dwukrotnie.
2. Dla elementu o klasie scale-small dodaj właściwość, która go pomniejszy o połowę.
3. Sprawdź jak zachowuje się przestrzeń wokół zmniejszonego elementu.

**PODPOWIEDŹ**

Element scale-small powinien posiadać właściwość:
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>
      <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=Open+Sa
ns);
* {
  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() przekrzywia element wzdłuż osi **X**(poziomej).

Tak samo jak i podczas nadawania rotacji, element przekrzywiamy używając **stopni**.

Przykładowo chcąc przekrzywić element o **20**stopni wzdłuż osi **X** użyjemy następującej właściwości:
transform: skewX(20deg)

**ZADANIA**

1. Dla elementu o klasie skew-light dodaj właściwość, która przekrzywi element wzdłuż osi **X** o 20 stopni.
2. Elementowi skew-medium nadaj właściwość skewX o wartości 50deg.
3. Elementowi skew-hard dodaj właściwość skewX o wartości 70 stopni.
4. Porównaj elementy i sprawdź różnicę.

**PODPOWIEDŹ**

Elementy o klasach skew-light, skew-medium i skew-hard powinny mieć odpowiednio właściwości:
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=Open+Sa
ns);
* {
  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 jest bratnią właściwością dla skewX. I jakżeby inaczej - przekrzywia ona obiekt, ale w tym przypadku wzdłuż osi **Y**.

I jak już wiemy - jednostki jakich używamy do określenia przekrzywienia to stopnie (**deg**).

Aby w lekcji stało się zadość, przejdźmy do przykładu:
transform: skewY(50deg)
Tak zbudowana właściwość przekrzywi element o **50** stopni wzdłuż osi **Y**.

**ZADANIA**

1. Dla elementu o klasie skew-light dodaj właściwość transform: skewY() o wartości 5deg.
2. Elementowi skew-hard .nadaj właściwość skewY o wartości -15deg.
3. Porównaj elementy i sprawdź różnicę!

**PODPOWIEDŹ**

Kod dla elementu .skew-hard powinien wyglądać następująco:

```
.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>
      <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=Open+Sa
ns);
* {
  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() łączy wszystkie pozostałe właściwości w jedną, nieco bardziej złożoną.

Struktura właściwości wygląda następująco:

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

Powiedzmy więc, że chcemy nadać elementowi następujące właściwości:

- lekko przekrzywić element względem osi X i osi Y
- przesunąć element o 30px po osi X
- oraz przesunąć element o 10px po osi Y

Moglibyśmy zrobić to przy użyciu czterech właściwości, lub jednej.
Po co więc utrudniać sobie życie jeśli oczekiwany efekt osiągniemy dzięki:

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

**ZADANIA**

1. Utwórz regułę dla selektora .exercise-element z pseudoklasą :hover.
2. Nadaj mu właściwość transform.
3. I nadaj jej wartość matrix(0.8, 0.1, 0.1, 0.7, 30, 30).
4. Najedź na element i sprawdź jak zachodzą zmiany.

**PODPOWIEDŹ**

Poprawny kod dla selektora .exercise-element:hover powinien wyglądać następująco:

```
.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+Sa
ns);
* {
  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

Jak zapewne zauważyliście, wszystkie transformacje zachodzą względem środka obiektu.

Dzieje sie tak, ponieważ jego "punkt źródłowy" (czyli domyślna wartość transform-origin) znajduje się w miejscu o współrzędnych 50% 50%.

Na położenie tego punktu składają się dwie wartości (możliwe trzy, ale o tym powiemy w transformacjach 3D)- wartość osi X oraz wartość osi Y.

Współrzędna dla osi **X** może przyjmować następujące wartości:
left
center
right
%

Współrzędna dla osi **Y** może przyjmować następujące wartości:
top
center
bottom
%

Brzmi trochę zagmatwanie? Spokojnie zobacz jak może wyglądać przykładowy kod:
transform-origin: 50% 0%

W ten sposób przemieściliśmy punkt źródłowy elementu do połowy szerokości jego górnej krawędzi.

**ZADANIA**

1. Utwórz regułę z selektorem .new-origin:hover.
2. Nadaj jej właściwość rotateY(360deg).
3. Następnie najedź na element i sprawdź jak wygląda przemiana.
4. Dodaj właściwość transform-origin: 0 0.
5. Sprawdź ponownie jak wykonuje się przemiana

**PODPOWIEDŹ**

Ta wartość transform-origin przeniosła punkt źródłowy do lewego górnego rogu elementu.

```
<!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=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;
}
.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

Na sam początek należy nadmienić jak sprawić by transformacje 3D były możliwe, wymagana jest perspektywa.

Nadać ją możemy poprzez dwie właściwości CSS.

Pierwszą jest transform: perspective.

Nadaje ona wybranemu elementowi "głębię", która służy za perspektywę.

Kolejna właściwość to: perspective: Xpx(gdzie X to liczba) nadawana jest elementowi rodzica i dotyczy także elementów w nim zawartych.

Jednostki używane do określania tych właściwościach to pixele (**px**).

Im większa wartość tym wrażenie perspektywy zmniejsza się, a element "oddala" się od nas.

**ZADANIA**

1. Do seleketora elementu o klasie big-perspective nadaj właściwość transform: perspective(800px) rotateY(60deg).
2. Do selektora elementu o klasie complex-perspective nadaj perspektywę o wielkości 800px przy użyciu właściwości perspective.
3. Do selektora elementu small-perspectivedopisz właściwość transform: perspective(200px) rotateY(60deg).

**PODPOWIEDŹ**

Powinieneś dodać następujące właściwości:

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=Open+Sa
ns);
* {
  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) definiuje przemieszczenie obiektu w **trójwymiarowej** przestrzeni. Dwie pierwsze wartości określają przemieszczenie w osi **X** (pozioma) oraz osi **Y** (pionowa), trzecia w osi **Z**.

Przykładowo właściwość może wyglądać następująco:
transform: translate3d(10%, 10%, 200px)

Efektem będzie przesunięcie obiektu o 10% na osi **Y**, 10% na osi **X** oraz 200px na osi **Z**. (Jednostki używane dla przemieszczenia na osi **Z** mogą być tylko wartościami w pixelach).

**ZADANIA**

1. Utwórz regułę z selektorem .exercise-element:hover.
2. Dodaj w niej właściwość transform: perspective(800px) rotateX(45deg) translate3d(10%, 10%, -100px).
3. Najedź na element i sprawdź jak zmienia się jego położenie!

**PODPOWIEDŹ**

Poprawny kod dla elementu .exercise-elementz pseudoklasą :hover wygląda następująco:

```
.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) jest kolejną właściwością o podobnym działaniu do swojej poprzedniczki działającej w przestrzeni dwuwymiarowej.

W tym wypadku skalujemy elementy w trzech wymiarach.

Właściwość scale3d(x, y, z) skaluje element kolejno:

x w osi **X**
y w osi **Y**
z w osi **Z**

Właściwość przyjmuje liczby z zakresu od 0 do nieskończoności.
W przypadku liczb mniejszych niż **1** element zostanie zmniejszony. Jeśli liczba będzie zerem to element zniknie, a w przypadku liczb dodatnich większych niż **1** zostanie powiększony.

Teoretycznie właściwość przyjmuje również liczby ujemnie, ale element nie zostanie wtedy skalowany.

**ZADANIA**

1. Utwórz regułę z selektorem .practice:hoveri nadaj jej właściwość transform.
2. Spraw aby obiekt skalował się w kierunku osi **X** o 1.7, osi **Y** o 1.1 a w kierunku osi **Z** wcale.
3. Najedź kursorem na element i sprawdź zachodzące zmiany.

**PODPOWIEDŹ**

Właściwość powinna wyglądać następująco:
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=Open+Sa
ns);
* {
  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() to właściwość, którą zdecydowanie polubicie. Przy jej pomocy obrócicie element względem trzech osi **jednocześnie**.

Budowa właściwości wygląda następująco:
transform: rotate3d(x, y, z, deg)

Gdzie kolejne wartości oznaczają:
x - pozycję wektora **x**
y - pozycję wektora **y**
z - pozycję wektora **z**
deg - wielkość rotacji w stopniach (możliwe wartości ujemne, wtedy element obraca się w kierunku przeciwnym do ruchu wskazówek zegara)

Jeśli więc przestrzeń między wektorami będzie taka sama niezależnie od wartości (np wszystkie wektory mają wartość 5, lub wszystkie mają wartość 100) animacja wyświetli się tak samo.

Dysproporcja wartości poszczególnych wektorów wpłynie w najwidoczniejszym stopniu na różnice w rotacji.

Przykładowa właściwość może wyglądać następująco:
transform: rotate3d(1,2,1, 180deg)

**ZADANIA**

- Utwórz regułę z selektorem .practice:hover.
- Dodaj w niej właściwość transform: rotate3d(1,2,3,360deg).
- Najedź na element i sprawdź jaka przemiana zachodzi!

**PODPOWIEDŹ**

Poprawny selektor, który trzeba utworzyć w tym zadaniu wygląda następująco:

```
.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=Open+Sa
ns);
* {
  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 to właściwość ukrywająca zawartość elementu w momencie gdy możemy zobaczyć jego tylną ściankę.

backface-visibility posiada tylko dwie właściwości:
visible (tył elementu jest widoczny)
hidden (tył elementu zostaje ukryty a element staje się czarny)

Przykładowo więc właściwość może wyglądać następująco:
backface-visibility: hidden

**ZADANIA**

1. Utwórz regułę o selektorze .practice:hover.
2. I nadaj jej właściwość transform: rotateY(360deg).
3. Najedź na element i sprawdź wygląd przemiany.
4. Następnie dodaj właściwość backface-visibility: hidden;.
5. Sprawdź jakie zmiany zaszły!

**PODPOWIEDŹ**

Poprawny kod wygląda następująco:

```
.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=Open+Sa
ns);
* {
  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. Podstawy keyframes

Lekcje o animacjach CSS najlepiej zacznijmy od samych podstaw. Niewątpliwie podstawą są wymagane właściwości, które pozwolą nam wprawić w ruch animację, jak i sama jej budowa.

Deklarację animacji zaczynamy od @keyframes po czym dodajemy jej nazwę i otwieramy nawiasy.

Przykładowo:
@keyframes mojaAnimacja {...}

Wewnątrz takiej deklaracji umieszczamy słowa kluczowe (ang, *keywords*), które pełnią rolę kroków.
Możliwe *keywords* to: from oraz tooznaczając e następująco: od, do (start i koniec animacji).

Możliwe *keywords* to również procenty. Przykład takiej konstrukcji możesz znaleźć w pliku *style.css* i podglądzie.

Przykład dla animacji używającej from to, która będzie przemieszczała element z lewej do prawej (tylko o 200px) może wyglądać następująco:

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

**ZADANIA**

1. W selektorze elementu o klasie practice-button mamy już gotową właściwość,animation więc zajmiemy się tylko utworzeniem keyframes.
2. Utwórz więc selektor regułę @keyframes i nazwij ją buttonColor.
3. Utwórz w niej keyword from i umieść w nim background-color: #f39c12.
4. Następnie utwórz keyword to i umieść w nim background-color: blue.

**PODPOWIEDŹ**

Reguła @keyframes powinna wyglądać następująco:

```
@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=Open+Sa
ns);
* {
  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 - budowa właściwości

Skoro już znasz podstawy budowania reguły keyframes, możemy przejść do poznania budowy właściwości animation - jest ona niezbędna do wprawienia animacji w ruch.

animation wymaga dwóch właściwości, aby zadziałać. Pierwszą jest nazwa animacji, a drugą czas jej wykonania. Oczywiście możemy dopisywać do niej inne wartości, które odpowiadają za np. ilość powtórzeń, ale o tym opowiemy sobie później :)

Przykładowa właściwość wygląda następująco:
animation: mojaAnimacja 1s

animation możemy rozbić na pojedyńcze właściwości: animation-name oraz animation-duration. Jednak pojedyncza właściwość animation zaoszczędzi nam pisania kodu i zgrupuje potrzebne właściwości, podobnie jak poprzedni "skrót", który poznaliśmy, czyli transition.

**ZADANIA**

1. Skoro wiesz już jak tworzyć **keyframes**, poćwiczmy tworzenie właściwości, które je obsługują.
2. Utwórz selektor elementu o klasie practice-button w stanie :hover
3. Nadaj mu właściwość animation.
4. I dodaj wartości obsługujące animację o nazwie exercise i nadaj jej 1s czasu trwania.

**PODPOWIEDŹ**

Selektor powinien wyglądać następująco:
.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=Open+Sa
ns);
* {
  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. Tempo animacji - animation-timing-function

Założę się, że pamiętasz lekcję o transition-timing-function. Jeśli tak to pewnie domyślasz się, że działanie omawianej właściwości animation-timing-function jest identyczne. Jeśli jednak zdarzyło Ci się zapomnieć, to szybko Ci przypomnę najważniejsze założenia.

Właściwość animation-timing-functionokreśla w jakim tempie wykonywać dane etapy animacji (start, środek, koniec), jednak nie wpływa na czas trwania animacji. Możemy przyjąć, że jest to swoista **funkcja synchronizacji**.

Przykładowe właściwości to:
ease (wartość domyślna - powolny start, szybki środek i powolny koniec)
linear (stałe tempo)
ease-in-out (powolny start i koniec)
cubic-bezier(x,x,x,x)(gdzie x to liczba od **0**do **1**) (dowolnie stworzona przez nas funkcja)
steps(x) (gdzie x to liczba kroków) (rozbija animację na kroki w podanej ilości)

Przykłady możesz zobaczyć na własne oczy w podglądzie i przyjrzeć się różnicom między nimi!

**ZADANIA**

1. Mamy już gotową, działającą animację, jednak chcemy trochę ją zmienić poprzez animation-timing-function, więc bierzmy się do roboty!
2. Utwórz selektor .practice-steps i nadaj mu funkcję synchronizacji, która rozbije animację na 6 kroków.
3. Następnie utwórz selektor dla elementu o klasie practice-linear.
4. I nadaj mu właściwość animation-timing-function o wartości linear.
5. W oknie podglądu zobacz jak wiele zmienia ta właściwość!

**PODPOWIEDŹ**

Pierwszy selektor powinien zawierać właściwość:
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=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/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. Opóźniamy animację - animation-delay

Podczas tej lekcji poznamy właściwość animation-delay. Tak, masz rację - poznaliśmy już bliźniaczą właściwość podczas lekcji związanej z przejściami.

W roli szybkiego przypomnienia wspomnę o działaniu tej właściwości (w końcu zawsze lepiej utrwalać nowo poznaną wiedzę).

animation-delay ustala opóźnienie w wykonywaniu animacji i nie wpływa na szybkość jej wykonywania.

Mimo wszystko, jeśli opóźnimy wszystkie animacje o wielokrotność czasu ich wykonania, to po pierwszej pętli nie będzie już widoczna ta różnica w czasie - dlatego warto przynajmniej jedną z animacji opóźnić np. o 1.5 czasu jej wykonywania zamiast o 1 lub 2 czasy wykonywania :)

Przykładowa właściwość opóźniająca wykonanie animacji o jedną sekundę może wyglądać następująco:
animation-delay: 1s

**ZADANIA**

1. Utwórz selektor dla elementu o klasie delay-short znajdującego się w elemencie z klasą exercise i nadaj mu opóźnienie animacji o 2s
2. Następnie utwórz selektor elementu o klasie delay-medium zawartego w elemencie .exercise i nadaj mu animation-delay: 3s.
3. Na koniec utwórz selektor dla elementu o klasie delay-long znajdującego się w elemencie z klasą exercise i określ opóźnienie dla jego animacji o 4s.

**PODPOWIEDŹ**

Poprawny selektor dla elementu .delay-shortwygląda następująco:

```
.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=Open+Sa
ns);
* {
  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. Liczymy powtórzenia - animation-iteration-count

Nic nie może przecież wiecznie trwać...

A może jednak? Sprawdzimy to przy użyciu właściwości animation-iteration-count. Jak nazwa wskazuje właściwość ta określa ilość powtórzeń animacji.

Domyślną wartością animation-iteration-count jest liczba 1. Skoro domyślną wartością jest liczba, to idąc dalej tą logiką możemy wywnioskować, że przykładowa wartość tej właściwości to również po prostu liczba.

Jeśli więc chcemy wykonać animację pięć razy to właściwość będzie wyglądała następująco:
animation-iteration-count: 5

No dobrze, a co jeśli jednak chcemy, żeby animacja nie zatrzymywała się po określonej ilości powtórzeń? W tym wypadku użyjemy wartości infinite, a właściwość będzie wyglądała następująco:
animation-iteration-count: infinite

**ZADANIA**

1. Na początek do selektora elementu o klasie few-repeats przypisz właściwość określającą ilość powtórzeń animacji.
2. Niech animacja powtórzy się 4 razy .
3. Do selektora elementu infinite-animationprzypisz właściwość animation-iteration-count. Ustal jej wartość tak aby animacja powtarzała się nieskończenie wiele razy.

**PODPOWIEDŹ**

Druga właściwość powinna wyglądać następująco:
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=Open+Sa
ns);
* {
  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. Twoja pierwsza animacja

Poznaliśmy już najważniejsze właściwości animacji w CSS - pora wykorzystać to w praktyce.

W kodzie początkowym mamy animację sugerującą ładowanie - wygląda ładnie, ale jest nieco zbyt pusta - zróbmy coś z tym.

Dodajmy drugi animowany element w środku obecnego - tym razem mniejszy i obracający się w drugą stronę - dzięki temu nasza animacja będzie ciekawsza.

**ZADANIA**

1. Utwórz selektor .spinner:before oraz dodaj w nim następujące właściwości - pozycję absolutną, content: "", szerokość i wysokość 60px, box-sizing: border-box, border o właściwościach 2px solid transparent oraz prawy i lewy border o właściwościach 2px solid black.
2. W tym samym selektorze dodaj jeszcze border-radius: 50%, przezroczyste tło, toporaz left o wartości 50%, transform o wartości translate(-50%, -50%) oraz właściwość animation o wartości 1s linear spin-counterclockwise infinite.
3. Przejdźmy do tworzenia naszej animacji - utwórz @keyframes o nazwie spin-counterclockwise. Dla postępu 50% przypisz transform: translate(-50%, -50%) rotate(-180deg), a dla postępu 100% to samo, tylko z wartością -360deg zamiast -180deg.

**PODPOWIEDŹ**

translate(-50%, -50%) jest nam potrzebne w właściwości transform w połączeniu z toporaz left o wartości 50%, aby nasz pseudoelement :before z pozycją absolutną był położony na środku głównego elementu.

```
<!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=Open+Sa
ns);
* {
  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. O filtrach i prefiksach

Podczas tej lekcji omówimy działanie właściwości filter.

Jej zadaniem jest nadawanie wybranych efektów (określonych jako jej właściwości) elementom.

Przykładowo:
filter: blur(10px) (nadamy elementowi rozmycie)

Niesamowicie ważną informacją jest to, że do poprawnego działania tej właściwości na różnych przeglądarkach potrzebne będą **prefiksy**.

Prefiksy to swoiste przedrostki, które dodajemy przed właściwościami dla umożliwienia poprawnego działania na danym typie przeglądarek.

Istniejące prefiksy to:
-webkit- (Chrome, Opera)
-moz- (Firefox)
-o- (starsze wersje Opery)
-ms- (Internet Explorer)

Dodatkowo warto podkreślić, że tylko **niektóre**właściwości potrzebują prefiksów (zwykle te nowsze), a przeglądarki wraz z ich rozwojem zaczynają obsługiwać właściwości bez potrzeby dodawania prefiksów.

W naszym wypadku używać będziemy -webkit-, a przykładowa właściwość może wyglądać następująco:
-webkit-filter: blur(20px)

**ZADANIA**

1. Przejdź do pliku *style.css*.
2. Element o klasie old-photo posiada właściwość filter: sepia(100%), dodaj dla niego również właściwość prefiksowaną.
3. Kolejny element, posiadający właściwość filter, któremu musisz nadać wartość prefiksowaną to new-photo - w tym przypadku musisz dodać prefix dla filtru od kontrastu..

**PODPOWIEDŹ**

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

Element o klasie new-photo powinien zawierać:
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=Open+Sa
ns);
* {
  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)(gdzie X to liczba) nadaje elementowi rozmycie zgodnie z liczbą pixeli podanych w deklaracji.

Jedyną obsługiwaną jednostką są pixele, a im wyższa ich wartość, tym większe rozmycie nadamy elementowi.

Przykładowo, aby nadać duże rozmycie właściwość skonstruujemy następująco:
filter: blur(15px)

**ZADANIA**

1. Dla elementu o klasie light-blur dodaj właściwość filter: blur(2px).
2. Następnie dodaj taką samą właściwość, tym razem z prefiksem -webkit-.
3. W regule dla elementu o klasie strong-bluristnieje już właściwość filter: blur(20px).
4. Do reguły dopisz właściwość prefiksowaną

**PODPOWIEDŹ**

W elemencie o klasie light-blur powinieneś dodać właściwości:
filter: blur(2px)
-webkit-filter: blur(2px)

A do reguły .strong-blur powinieneś dodać właściwość:
-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=Open+Sa
ns);
* {
  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() odpowiada za rozjaśnianie oraz przyciemnianie elementu.

Właściwość przyjmuje wartości **procentowe**(%) w przedziale od 0 do nieskończoności (teoretycznie, ponieważ przy większej wartości element stanie się całkowicie biały i nie zajdzie więcej zmian). Przy wartości 0 element będzie całkowicie czarny.

Dla przykładu powiedzmy, że chcemy aby element był lekko przyciemniony, użyjmy więc następującej właściwości:
filter: brightness(75%)

**ZADANIA**

1. Dla elementu o klasie bright dodaj właściwość filter: brightness(150%).
2. Utwórz też taką samą właściwość z prefiksem -webkit-.
3. Dla elementu o klasie dark dodaj bliźniaczą właściwość filter dla już istniejącej i nadaj jej prefiks -webkit-.

**PODPOWIEDŹ**

W regule dla elementu bright powinieneś dodać właściwości:
filter: brightness(150%)
-webkit-filter: brightness(150%)

W regule dla elementu dark powinieneś dodać właściwość:
-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=Open+Sa
ns);
* {
  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() służy do ustawiana kontrastu elementu.

Przyjmowane jednostki to **procenty**(%) w zakresie od 0 (całkowicie czarny/ szary element) do nieskończoności (znów teoretycznej nieskończoności, ponieważ przy pewnej wartości zmiany przestaną zachodzić).

Więc jeśli chcemy zmienić kontrast w naszym elemencie użyjemy następującej właściwości: filter: contrast(250%)

Dodatkowo sprawdź przykłady dla innych wartości w sekcji **Example**.

**ZADANIA**

1. Do elementu o klasie strong-contrast dodaj właściwość filter: contrast(150%).
2. Dodaj również właściwość z prefiksem -webkit-.
3. Do selektora elementu .light-contrastdodaj właściwość filter: contrast(80%).
4. Dodaj również właściwość prefiksowaną.

**PODPOWIEDŹ**

W elemencie o klasie strong-contrastpowinieneś dodać następujące właściwości:
filter: contrast(150%)
-webkit-filter: contrast(150%)

A w elemencie o klasie light-contrast:
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=Open+Sa
ns);
* {
  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() odpowiada za nadanie elementowi szarości.

Jednostki jakich używamy do określenia tej właściwości to **procenty**(%) w zakresie od 0 do 100%.

Przy wartości 0 obrazek nie zmienia się, a im większa wartość tym bardziej desaturujemy obrazek (pozbywamy się kolorów).

Przykładowo jeśli chcemy aby nasz obrazek był czarno-biały użyjemy następującej właściwości:
filter: grayscale(100%)

**ZADANIA**

1. Dla elementu o klasie old-photo nadaj właściwość, która całkowicie pozbawi zdjęcie kolorów.
2. Pamiętaj także o dodaniu właściwości z prefiksem.
3. Dla elementu new-photo przypisz właściwość filter: grayscale(30%).
4. Dodaj też właściwość z prefiksem -webkit-.

**PODPOWIEDŹ**

Dla elementu o klasie old-photo powinieneś dodać następujące właściwości:
filter: grayscale(100%)
-webkit-filter: grayscale(100%)

A dla elementu new-photo:
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=Open+Sa
ns);
* {
  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() służy do odwracania kolorów.

Właściwość przyjmuje jednostki **procentowe**(%) w zakresie od 0 do 100%, gdzie zero to brak jakichkolwiek zmian, a 100% to całkowicie odwrócone kolory.

Przykładowa reguła częściowo odwracająca kolory może wyglądać następująco:
filter: invert(50%)

**ZADANIA**

1. Dla elementu o klasie full-invert nadaj właściwość całkowicie odwracającą kolory.
2. Dodaj też właściwość z prefiksem -webkit-.
3. Do reguły z selektorem .partial-invertprzypisz właściwość filter: invert(30%).
4. Dodaj też właściwość prefiksowaną.

**PODPOWIEDŹ**

Dla elementu o klasie full-invert powinieneś dodać następujące właściwości:
filter: invert(100%)
-webkit-filter: invert(100%)

A dla elementu partial-invert:
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=Open+Sa
ns);
* {
  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() odpowiada za saturację elementu. W roli szybkiego przypomnienia - saturacja to nasycenie kolorem.

saturate() jest kolejną właściwością przyjmującą wartości **procentowe**(%) w przedziale od 0 do nieskończoności (I tak, zgadłeś... Znów jest to teoretyczna nieskończoność bo przy pewnej wartości element przestanie się zmieniać).

Przy wartości 0 otrzymamy element pozbawiony kolorów. 100% to wartość domyślna, powyżej której zaczniemy nasycać element kolorami. (Właściwość najlepiej funkcjonuje między wartościami 0 - 200%.)

Więc jeśli chcemy nasycić element kolorami możemy użyć następującej właściwości:
filter: saturate(150%)

**ZADANIA**

1. Dla elementu o klasie colorful-image dodaj właściwość filter: saturate(200%).
2. Dodaj także właściwość prefiksowaną
3. Dla elementu gloomy-image dodaj właściwość filter: saturate(60%).
4. Dodaj także właściwość z prefiksem -webkit-.

**PODPOWIEDŹ**

Selektor klasy .colorful-image powinien zawierać następujące właściwości:

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=Open+Sa
ns);
* {
  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() jest właściwością umożliwiającą nadanie efektu sepii elementowi.

sepia() przyjmuje jednostki **procentowe**(%) w zakresie od 0 do 100%, gdzie 0 to wartość domyślna, a wartości większe nasilają efekt sepii.

Powiedzmy więc, że chcemy nadać efekt sepii naszemu elementowi, żeby wyglądał starzej. Użyjemy następującej właściwości: filter: sepia(100%)

**ZADANIA**

1. Do elementu o klasie old-photo przypisz właściwość filter: sepia(100%).
2. Utwórz też właściwość z prefiksem.
3. Do selektora new-photo dodaj właściwość filter: sepia(20%).
4. Pamiętaj o dodaniu wartości z prefiksem -webkit-.

**PODPOWIEDŹ**

Selektor klasy .old-photo powinien zawierać następujące właściwości:

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=Open+Sa
ns);
* {
  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 służy do zmiany kolorów obiektu zgodnie z kołem kolorów.

Właściwość jako swoje jednostki określające przyjmuje **stopnie**(deg) w zakresie od 0 do 360deg (możemy również używać liczb ujemnych).

0 jest wartością domyślną, a każda inna spowoduje zmiany.

Przykładowa właściwość może wyglądać następująco:
filter: hue-rotate(180deg)

**ZADANIA**

1. W regule z selektorem .small-rotationnadaj właściwość filter: hue-rotate(50deg).
2. Dodaj też właściwość z prefiksem.
3. W regule z selektorem big-rotation nadaj właściwość filter: hue-rotate(200deg).
4. Utwórz także właściwość z prefiksem -webkit-.

**PODPOWIEDŹ**

Selektor klasy .small-rotation powinien zawierać następujące właściwości:

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=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(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. Jak używać filtrów

Skoro znasz już tyle filtrów, pewnie zastanawiasz się jak można efektywnie ich używać.

Zacznijmy od tego, że jak większość właściwości, te związane z filtrami również możemy grupować.

Przykładowo jeśli chcemy nadać efekt sepii i lekko rozmyć element użyjemy następującej właściwości:
filter: blur(1px) sepia(100%) (nie używamy znaków rozdzielających)

Filtry idealnie nadają się do tworzenia uniwersalnych reguł wielokrotnego użytku.

Powiedzmy, że chcemy nadać kilku elementom dane filtry. Tworzymy więc klasy z samymi właściwościami dla filtrów i nadajemy je wybranym elementom w HTML.

**ZADANIA**

1. Utwórz regułę z selektorem .multiple-values.
2. Dodaj w niej właściwość filter: brightness(80%) contrast(120%) grayscale(80%).
3. Dodaj też właściwość z prefiksem -webkit-.

**PODPOWIEDŹ**

Poprawny kod dla tego ćwiczenia wygląda następująco:

```
.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() jest jedną z wartości właściwości background.

Do podstawowego działania potrzebuje dwóch wartości: koloru początkowego i koloru finalnego.

Przykładowo:
background: linear-gradient(red, green)

Nieco bardziej rozbudowana właściwość dzięki której utworzymy trójkolorowy gradient:
background: linear-gradient(red, green, yellow)

Domyślnie gradient przemieszcza się z góry na dół. Jednak możemy kontrolować ten kierunek.

Przykładowo:
background: linear-gradient(left, blue, green) (gradient od lewej do prawej)

background: linear-gradient(bottom right, blue, green) (gradient od prawego dolnego rogu do lewego górnego rogu)

Możemy również używać stopni:
background: linear-gradient(260deg, green, blue) (gradient z góry na dół)
background: linear-gradient(190deg, green, blue) (gradient od prawej do lewej)

**UWAGA:** Dla gradientów również musimy dodawać **prefiksy**, jednak wystarczy tylko jeden- -webkit-.

Przykładowa właściwość z prefiksem:
background: -webkit-linear-gradient(red, blue)

**ZADANIA**

1. W regule dla elementu gradient-containerutwórz gradient liniowy.
2. Ma on przemieszczać się od lewej do prawej.
3. Jego kolorem początkowym ma być zielony, który przejdzie w niebieski aż na końcu zamieni się w czerwony.
4. Dodaj również właściwość z prefiksem.
5. Jeśli zadanie wydaje ci się zbyt trudne zajrzyj do podpowiedzi.

**PODPOWIEDŹ**

W regule o selektorze .gradient-containerpowinieneś dodać właściwośći:

```
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=Open+Sa
ns);
* {
  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() jest bratnią wartością linear-gradient. Nie są to jednak bracia bliźniacy...

Przy pomocy radial-gradient utworzymy gradient **radialny** (promieniowy).

Jeśli ciężko ci wyobrazić sobie taki gradient pomyśl o kamieniu rzuconym w wodę i wyobraź sobie jak z jednego punktu rozchodzą się fale (możesz też po prostu zobaczyć w podglądzie).

Do prawidłowego działania potrzebujemy dwóch wartości: koloru początkowego i finalnego. Oczywiście możemy jednak rozbudowywać właściwość np:
background: radial-gradient(red, blue, green, yellow)

Dodatkowo warto wiedzieć, że kształt naszego gradientu to **elipsa**, jednak możemy zmienić tę domyślną wartość na okrąg:
background: radial-gradient(circle, red, blue)

Jeśli tyle możliwości wam nie wystarcza, to zawsze możemy jeszcze zmienić pozycję punktu, z którego "wydobywa" się gradient:
backgrund: radial-gradient(at x% y%, red, green)

Wartości **x%** i **y%** odpowiadają przesunięciu w osi **X** oraz osi **Y**. at to słowo kluczowe, oznaczające że kolejne wartości oznaczają miejsce od którego "rozchodzi" się gradient.

Więc jeśli nasza właściwość będzie wyglądała następująco:
background: radial-gradient(at 100% 100%, red, green)
Środek gradientu znajdzie się w prawym dolnym rogu.

**ZADANIA**

1. W regule o selektorze .radial-gradient-container utwórz właściwość background: radial-gradient(at 100% 100%, yellow, blue).
2. Utwórz również właściwość z prefiksem -webkit-.

**PODPOWIEDŹ**

Kod gradientu promieniowego wygląda następująco:

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=Open+Sa
ns);
* {
  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 umożliwia nam tworzenie **powtarzalnego** wzoru przy użyciu gradientu.

repeating-linear-gradient to po prostu "ulepszona" wersja właściwości linear-gradient.

Jako ulepszona wersja daje nam całkiem ciekawe możliwości. Przykładowo:
background: repeating-linear-gradient(-45deg, green 0, green 5%, yellow 0, yellow 10%)

Utworzy powtarzający się zielono-żółto-niebieski wzór z ostrymi krawędziami i widocznymi granicami między kolorami (sprawdź w podglądzie).

Jeśli jesteś ciekaw w jaki sposób działa ta właściwość przypatrz się uważnie:

-45deg deklaruje kierunek w jakim przemieszcza się gradient
green 0 ustala rozpoczęcie się zielonego koloru
green 5% trwającego do 5% szerokości elementu
yellow 0 zeruje pozycję żółtego paska
yellow 10% sprawia, że żółta barwa zajmuje miejsce do 10% szerokości elementu

Następnie sekwencja powtarza się, aż do zapełnienia całego dostępnego miejsca, a my otrzymujemy powtarzalny gradient.

**ZADANIA**

1. W selektorze o regule .linear-gradient-container utwórz właściwość background.
2. I nadaj jej wartość repeating-linear-gradient.
3. Wartość uzupełnij następująco: left, white 10%, orange 20%, white 30%.
4. Dodaj też właściwość z prefiksem -webkit-.

**PODPOWIEDŹ**

Cała właściwość powinna wyglądać następująco:
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=Open+Sa
ns);
* {
  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 to swoiste rozszerzenie właściwości radial-gradient o możliwość wielokrotnego wykonania gradientu.

Podczas tworzenia właściwości możemy używać poprzednich wartości stosowanych do tworzenia gradientu radialnego, ale dodatkowo deklarujemy rozprzestrzenianie się kolorów.

Jak powinna wyglądać taka właściwość? Tutaj mamy przykład:

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

Zasada działania jest identyczna jak właściwości repeating-linear-gradient.

**ZADANIA**

1. Dla reguły elementu o klasie radial-gradient-container dopisz właściwość background.
2. W tej właściwości przypisz wartość repeating-radial-gradient(circle, white 0, white 15px, black 0, black 30px).
3. Pamiętaj też o dodaniu tej samej właściwości z prefiksem -webkit-.

**PODPOWIEDŹ**

Kod powtarzającego się gradientu promieniowego wygląda następująco:

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

Nie zapomnij o dodaniu drugiej wersji z prefixem!

```
<!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+Sa
ns);
* {
  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. Złożone gradienty

Skoro znasz już podstawowe i te nieco bardziej złożone właściwości gradientów możemy poćwiczyć ich tworzenie.

Zabierzmy się za coś, co wydaje się być skomplikowane, jednak efekt otrzymamy bez większego trudu dzięki poznanym właściwościom dotyczącym gradientów.

Zapewne pamiętasz różnorakie tła stron w postaci siatek lub okręgów - oba te efekty możemy uzyskać za pomocą prostych gradientów, bez potrzeby dodatkowego obciążana strony kolejnym obrazkiem.

Teraz zajmiemy się tworzeniem tej drugiej wersji używającej okręgów.

**ZADANIA**

1. Skoro chcemy utworzyć gradient na bazie okręgów będziemy potrzebować gradientu radialnego.
2. Utwórz więc na początek regułę dla elementu, radial-gradient-container.
3. Utwórz w niej właściwość background.
4. I nadaj jej wartość radial-gradient(circle, #000 0, #55ee70 50%, #00ff48 0, #000 100%).
5. Utwórz też właściwość z prefiksem -webkit-.
6. Na koniec dodaj właściwość background-size o wartości 15px.

**PODPOWIEDŹ**

Twój kod dla elementu .radial-gradient-container powinien wyglądać następująco:

```
.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. Zaczynamy - podstawowe style.

W tej części kursu skupimy się na zbudowaniu animowanej ikony ładowania składającej się z dwóch przenikających się pierścieni.

To trochę niecodzienne, ale tym razem nie będziemy poświęcać całego ćwiczenia na budowę struktury HTML - dlaczego? Tym razem potrzebujemy po prostu tylko jednego elementu HTML - cała reszta zostanie utworzona za pomocą CSS i animacji.

Zanim jednak przejdziemy do animowania naszego elementu, musimy nadać mu wymagany wygląd oraz umieścić go w odpowiednim miejscu na stronie.

Przejdźmy do kodowania!

**ZADANIA**

1. Zacznijmy od utworzenia jednego elementu  z klasą circle i będziemy mogli przejść do stylowania.
2. Przejdźmy do stylowania - utwórz selektor grupowy dla <html> i <body> i przypisz w nim wysokość 100%, a następnie utwórz selektor dla <body> i przypisz w nim właściwość overflow: hidden.
3. Weźmy się za stylowanie elementu o klasie circle nadaj mu pozycję absolutną, top oraz left o wartości 50%, transform o wartości translate(-50%, -50%) oraz wysokość i szerokość 400px.

**PODPOWIEDŹ**

Właściwość transform: translate(-50%, -50%) w połączeniu z top oraz left o wartości 50% i pozycją absolutną centruje element w pionie i w poziomie.

```
<!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. Tworzenie pierścieni

Teraz zajmiemy się ostylowaniem pseudoelementów, które niedługo staną się naszymi pierścieniami.

W kolejnych ćwiczeniach będziemy sukcesywnie rozwijać stworzone elementy, a na samym końcu dodamy do nich animację.

**ZADANIA**

1. Utwórz selektor grupowy dla pseudoelementów .circle:before i circle :after, a w następujące właściwości: content: "", border-radius o wartości 50%, pozycję absolutną, top oraz left o wartości 0, wysokość i szerokość 100% oraz transform-origin o wartości center.

**PODPOWIEDŹ**

Właściwość transform-origin określa punkt względem którego wykonywane są działania z właściwości transform.

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. Stylujemy pierścienie!

W tym ćwiczeniu nadamy naszym pierścieniom z pseudoelementów odpowiednie style, dzięki którym będą wyglądały naprawdę widowiskowo.

Z uwagi na to, że wymagane właściwości są dosyć długie, przygotowaliśmy dla Ciebie dwa gotowe fragmenty kodu, opierające się na właściwości box-shadow.

Właściwość box-shadow dla pseudoelementu .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);

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);

**ZADANIA**

1. Utwórz oddzielne selektory dla .circle:before oraz .circle:after.
2. W utworzonych selektorach dodaj odpowiednie właściwości podane powyżej.
3. Pseudoelementowi :before dla .circlen adaj również właściwość animation o wartości counterclockwise 1.5s -0.5s linear infinite
4. Pseudoelementowi :after dla .circlenad aj taką samą właściwość animation jak :before, ale zamień nazwę animacji counterclockwise na clockwis e.

**PODPOWIEDŹ**

Właściwość box-shadow określa cień nałożony na 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. Pora na animacje!

Utworzyliśmy już nasze pierścienie, brakuje im tylko... życia :)

W tym ćwiczeniu wprawimy stworzone wcześniej elementy w ruch i nadamy im właściwości, które sprawią, że napisany przez nas element będzie prezentował się o wiele lepiej :)

**ZADANIA**

1. Utwórz @keyframes o nazwie clockwise i umieść w niej następujące właściwości - dla postępu 0%, przypisz transform o wartości rotateZ(0deg) scaleX(1) scaleY(1). Dla postępu 50% przypisz transform o wartości rotateZ(180deg) scaleX(0.80) scaleY(0.95), a dla 100% postępu przypisz transform: rotateZ(360deg) scaleX(1) scaleY(1).
2. Następnie stwórz @keyframes o nazwie counterclockwise i dla postępu 0% przypisz mu transform: rotateZ(0deg) scaleX(1) scaleY(1), dla postępu 50% transform: rotateZ(-180deg) scaleX(0.95) scaleY(0.80), a dla postępu 100% nadaj mu transform o wartości rotateZ(-360deg) scaleX(1) scaleY(1).

**PODPOWIEDŹ**

Poprawny kod dla pierwszego @keyframespowinien wyglądać następująco:@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);
    }
}
```