

Moje Kursy

Career Lab Front End  
sygnatura: WAR\_CFE\_D\_06

JavaScript Developer: React  
sygnatura: WRO\_FER\_S\_24

0. Prework

1. Zaawansowany HTML i CSS

2. Sass i RWD

- Praca samodzielna - Podstawy NodeJS i npm

Dzień 1 - Konfiguracja środowiska i Sass

Dzień 1 - Praca domowa

Dzień 2 - Sass i RWD

Dzień 2 - Praca domowa

Dzień 3 - RWD

Dzień 3 - Praca domowa

Dzień 4 - RWD i Animacje

Dzień 4 - Praca domowa

Dzień 5 - Warsztat

Egzamin próbny

Egzamin próbny - rozwiązanie

3. Praca własna 1

4. JavaScript

5. ES6 i React Podstawy

6. Praca własna 2

7. React

Scrum Lab Front End  
sygnatura: WRO\_SFE\_S\_33

Kursy > WRO\_FER\_S\_24 > Nauka > Sass i RWD > Egzamin próbny - rozwiązanie > Egzamin próbny - rozwiązanie > Zadanie 3

Artykuł

## Zadanie 3

Idź w górę

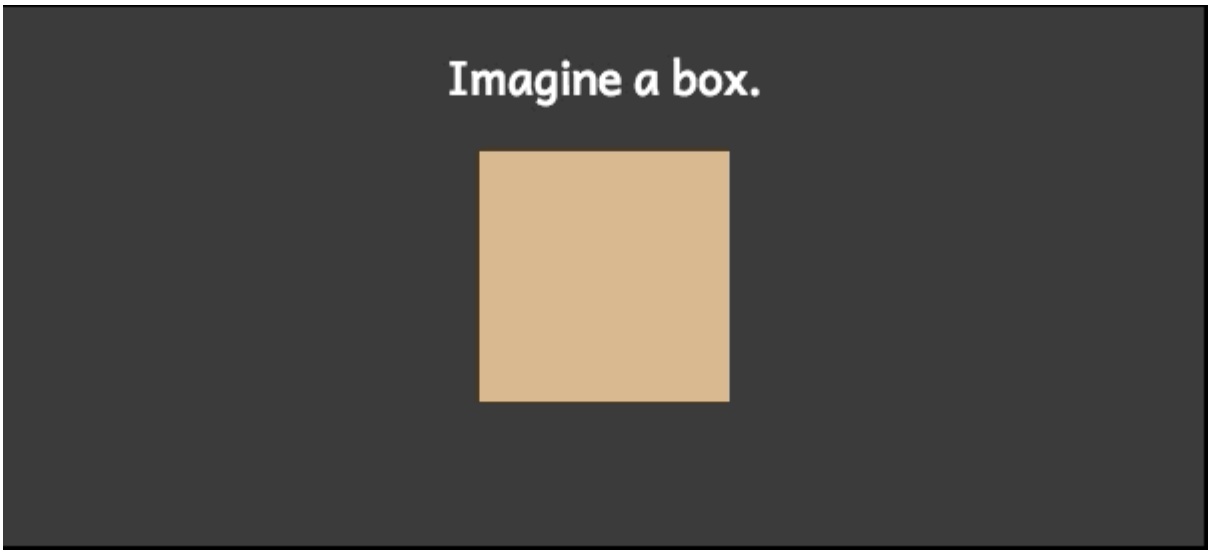
Poprzedni

Następny

Zadanie trzecie polegało na stworzeniu `mixina` tworzącego strzałki we wszystkich kierunkach. Mixin miał przyjmować trzy parametry:

- kolor,
- kierunek,
- rozmiar.

Na tej podstawie powinna powstać strzałka. Jak stworzyć strzałkę w CSS? Dzięki specyfice ich ramek (`border`). Schodzą się one w narożnikach elementów pod kątem 45°. Aby stworzyć strzałkę skierowaną w górę, wystarczy stworzyć `border-bottom` o konkretnym kolorze i wymiarze, a także `border-left` i `border-right` które będą miały ustawiony kolor `transparent` a także połowę szerokości `border-bottom`.



oryg. <https://www.labnol.org/code/19659-pure-css-triangles>

```
1 | @mixin arrow($color, $direction, $size) {
2 |   width: $size;
3 |   height: 0;
4 |   border-bottom: $size solid $color;
5 |   border-left: $size / 2 solid transparent;
6 |   border-right: $size / 2 solid transparent;
7 | }
```

Kopiuuj

W naszym mixinie stworzyliśmy jak na razie tylko jeden wariant strzałki - skierowaną do góry. Aby reagować na wartość zmiennej `$direction` potrzebujemy skorzystać z:

- instrukcji warunkowej `@if`
- właściwości `transform` i metody `rotate` aby odpowiednio obracać strzałki

Na koniec naszego mixina dodamy:

```
1 | @if $direction == bottom {
2 |   transform: rotate(180deg);
3 | } @else if $direction == left {
4 |   transform: rotate(270deg);
5 | } @else if $direction == right {
6 |   transform: rotate(90deg);
7 | }
```

Kopiuuj

Pozostało tylko przetestować jego działanie na stworzonych wcześniej klasach:

```
1 | .arrow-test {
2 |   &-blue-top-20px {
3 |     @include arrow(blue, top, 20px);
4 |   }
5 |
6 |   &-red-left-25px {
7 |     @include arrow(red, left, 25px);
8 |   }
9 |
10 |   &-green-right-60px {
11 |     @include arrow(green, right, 60px);
12 |   }
13 |
14 |   &-orange-bottom-100px {
15 |     @include arrow(orange, bottom, 100px);
16 |   }
17 | }
```

Kopiuuj

### Pełne rozwiązanie

```
1 | @mixin arrow($color, $direction, $size) {
2 |   width: $size;
3 |   height: 0;
4 |   border-bottom: $size solid $color;
5 |   border-left: $size / 2 solid transparent;
6 |   border-right: $size / 2 solid transparent;
7 |
8 |   @if $direction == bottom {
9 |     transform: rotate(180deg);
10 |   } @else if $direction == left {
11 |     transform: rotate(270deg);
12 |   } @else if $direction == right {
13 |     transform: rotate(90deg);
14 |   }
15 | }
16 |
17 | .arrow-test {
18 |   &-blue-top-20px {
19 |     @include arrow(blue, top, 20px);
20 |   }
21 |
22 |   &-red-left-25px {
23 |     @include arrow(red, left, 25px);
24 |   }
25 |
26 |   &-green-right-60px {
27 |     @include arrow(green, right, 60px);
28 |   }
29 |
30 |   &-orange-bottom-100px {
31 |     @include arrow(orange, bottom, 100px);
32 |   }
33 | }
```

Kopiuuj

Jak oceniasz artykuł ?

