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
    - 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

Artykuł

Zadanie 4

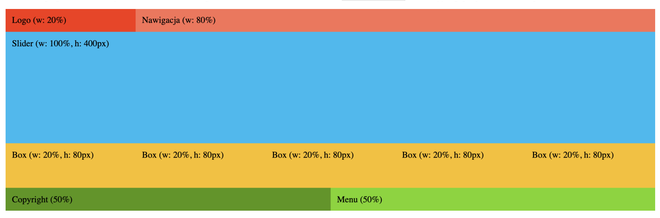
Idź w górę Poprzedni Następny

Zadanie 4 skupia się na Gridzie. Na początku mieliśmy stworzyć własnego Grida o układzie 10-kolumnowym. Jak najbardziej możemy tutaj skorzystać z siatki którą tworzyliśmy podczas zajęć. Wystarczy zmienić wartość zmiennej `$columns` na `10`, a także ustawić szerokość `1200px` dla klasy `container`.

Kopiuj

```scss
$columns: 10;
$column-base-width: 100% / $columns;

.container {
  width: 100%;
  max-width: 1200px;
  margin: 0 auto;
  padding: 0 15px;

  .row {
    display: flex;
    flex-wrap: wrap;

    [class*=col-] {
      min-height: 1px;
      width: $column-base-width;
      padding: 12px;
    }

    @for $i from 1 through $columns {
      .col-#{$i} {
        width: $column-base-width * $i;
      }
    }
  }
}
```

W kolejnym kroku powiedziano aby odwzorować w klasie `task-04` następującą strukturę:



Na początek zajmijmy się strukturą HTML. Zanim zaczniemy tworzyć kolumny potrzebujemy dwóch klas: `container` i `row`.

1. W pierwszym rzędzie mamy układ 20% / 80%. Potrzebujemy użyć więc klas: `col-2` i `col-8` (2/10 i 8/10).
2. Cały drugi rząd zajmuje "slider". Dlatego tworzymy klasę `row` a w niej `col-10` (10/10).
3. Trzeci rząd to pięć boksów po 20% szerokości każdy. Dlatego tworzymy pięć elementów `div` z klasami `col-2` (2/10).
4. W czwartym rzędzie mamy układ 50% / 50%, co dają nam klasy `col-5` użyte dwukrotnie.

Będziemy musieli również pokolorować wszystkie kolumny, dlatego dodajmy do nich klasy:

- Logo: `logo`
- Nawigacja: `navigation`
- Slider: `slider`
- Box: `box`
- Copyright: `copyright`
- Menu: `menu`

Kopiuj

```html
<section class="task-04">
  <div class="container">
    <div class="row">
      <div class="col-2 logo">
        Logo (w: 20%)
      </div>
      <div class="col-8 navigation">
        Nawigacja (w: 80%)
      </div>
    </div>
    <div class="row">
      <div class="col-10 slider">Slider (w: 100%, h: 400px)</div>
    </div>
    <div class="row">
      <div class="col-2 box">Box (w: 20%, h: 80px)</div>
      <div class="col-2 box">Box (w: 20%, h: 80px)</div>
      <div class="col-2 box">Box (w: 20%, h: 80px)</div>
      <div class="col-2 box">Box (w: 20%, h: 80px)</div>
      <div class="col-2 box">Box (w: 20%, h: 80px)</div>
    </div>
    <div class="row">
      <div class="col-5 copyright">Copyright (50%)</div>
      <div class="col-5 menu">Menu (50%)</div>
    </div>
  </div>
</section>
```

Do kolorowania boksów mieliśmy użyć mapy z zadnia drugiego. Wszystkie pliki importowane są do `main.scss` dlatego nie musimy tutaj robić dodatkowego importu pliku `task02` - mamy już dostęp do zmiennej `$colors`.

Jednak żeby wyciągnąć z niej wartości musimy użyć metody `map.get`, dlatego użyjemy moduły `sass:map`. Zadanie mówi też o tym, że niektóre kolory mają być jaśniejsze a inne ciemniejsze. Służy do tego metoda `color.scale` z modułu `sass:color`:

Kopiuj

```scss
// Na początku pliku
@use 'sass:map';
@use 'sass:color';

// ... grid

.logo {
  background-color: map.get($colors, red);
}

.navigation {
  background-color: color.scale(map.get($colors, red), $lightness: 30%);
}

.slider {
  background-color: map.get($colors, blue);
  height: 200px;
}

.box {
  background-color: map.get($colors, gold);
  height: 80px;
}

.copyright {
  background-color: color.scale(map.get($colors, green), $lightness: -30%
}

.menu {
  background-color: map.get($colors, green);
}
```

Aby rozjaśnić dany kolor, podajemy w metodzie `color.scale` parametr `$lightness` w przedziale 0% - 100%. Aby go przyciemnić: -100% - 0%.

Pełne rozwiązanie

Kopiuj

```scss
@use 'sass:map';
@use 'sass:color';

$columns: 10;
$column-base-width: 100% / $columns;

.container {
  width: 100%;
  max-width: 1200px;
  margin: 0 auto;
  padding: 0 15px;

  .row {
    display: flex;
    flex-wrap: wrap;

    [class*=col-] {
      min-height: 1px;
      width: $column-base-width;
      padding: 12px;
    }

    @for $i from 1 through $columns {
      .col-#{$i} {
        width: $column-base-width * $i;
      }
    }
  }
}


.logo {
  background-color: map.get($colors, red);
}

.navigation {
  background-color: color.scale(map.get($colors, red), $lightness: 30%);
}

.slider {
  background-color: map.get($colors, blue);
  height: 200px;
}

.box {
  background-color: map.get($colors, gold);
  height: 80px;
}

.copyright {
  background-color: color.scale(map.get($colors, green), $lightness: -30%
}

.menu {
  background-color: map.get($colors, green);
}
```

Jak oceniasz artykuł?