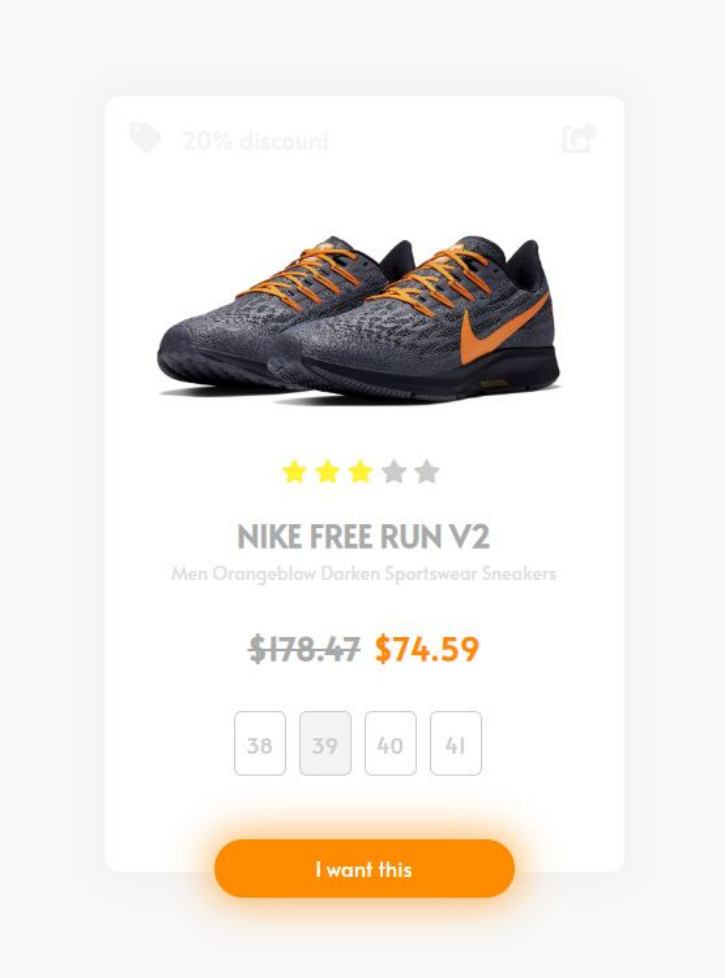
Tytuł: Refaktoryzacja karty produktu (HTML5/CSS3 & Angular 8)

Autor: Jakub Lebiedziński

Kontekst: Implementacja dwóch bliźniaczych elementów aplikacji internetowej (dwóch komponentów) napisanych w HTML w wersji 5 i CSS w wersji 3 przy użyciu framework'u Angular w wersji 8 i refaktoryzacja jednego z nich w celu łatwiejszego przeprowadzenia w przyszłości zmian kontentu i stylistyki karty.

Link do repozytorium: https://github.com/lebkovy/Refactoring

W moim doświadczeniu wygenerowałem dwa komponenty w Angularze w celu implementacji karty produktu szeroko rozpowszechnionej na świecie, gdyż występuje w prawie każdym sklepie internetowym. Na kartę zawierają się: ikony akcji umieszczone w górnej części karty, zdjęcie przedstawiające produkt (w tym wypadku parę sportowych butów), nazwę i krótki opis produktu, ocenę użytkowników w formie skali 1-5, cenę i ewentualny rabat, możliwość wyboru rozmiaru oraz przycisk akcji do potencjalnego rozpoczęcia procesu zakupu ww. produktu. Poniżej widok karty.



Kod HTML przed refaktoryzacją

```html
<div style="position: relative; border-radius: 12px; box-shadow: 0px 0px 30px 5px rgba(185, 185, 185, 0.1); border: 1px solid #f4f4f4; background-color: white; padding: 40px; height: 600px; width: 400px; margin-left: auto; margin-right: auto; position: relative">
    <div style="position: absolute; top: 20px; left: 20px; width: 300px;">
        <i class="fas fa-tag" style="color: #f4f4f4; font-size: 24px; display: inline-block"></i>
        <div style="display: inline-block; margin-left: 15px; color: #f4f4f4; font-size: 18px;">20% discount</div>
    </div>
    <i class="fas fa-share-square"
       style="position: absolute; top: 20px; right: 20px; color: #f4f4f4; font-size: 24px; cursor: pointer"></i>
    <img alt="Sneaker" src="../../assets/sneaker.png"
         style="height: 40%; display: block;width: auto; margin-top: 30px;">
    <div style="margin-top: 0px; width: 100%; text-align: center">
        <div class="fas fa-star" style="color: #FFF334; margin-right: 5px; font-size: 18px;"></div>
        <div class="fas fa-star" style="color: #FFF334; margin-right: 5px; font-size: 18px;"></div>
        <div class="fas fa-star" style="color: #FFF334; margin-right: 5px; font-size: 18px;"></div>
        <div class="fas fa-star" style="color: #ccc; margin-right: 5px; font-size: 18px;"></div>
        <div class="fas fa-star" style="color: #ccc; margin-right: 5px; font-size: 18px;"></div>
        <div style="color: #aaa; font-weight: 600; font-size: 24px; width: 100%; text-align: center; margin-top: 20px;">
            NIKE FREE RUN V2
        </div>
        <div style="font-size: 14px; color: #dfdfdf; margin-bottom: 30px;">
            Men Orangeblow Darken Sportswear Sneakers
        </div>
        <div style="width: 100%; text-align: center; display: inline-block">
            <div style="color: #aaa; font-weight: 600; text-decoration: line-through; margin-right: 5px; font-size: 24px; display: inline-block">
                $178.47
            </div>
            <div style="color: darkorange; font-weight: 600; font-size: 24px; margin-left: 5px; display: inline-block">
                $74.59
            </div>
        </div>
        <div style="margin-top: 30px; margin-left: auto; margin-right: auto;">
            <div style="margin-right: 10px; cursor: pointer; border: 1px solid #ccc; border-radius: 6px; color: #ccc; text-align: center; width: 40px; height: 50px; padding: 5px 0; line-height: 40px; display: inline-block;">
                38
            </div>
            <div style="background-color: #f4f4f4; margin-right: 10px; cursor: pointer; border: 1px solid #ccc; border-radius: 6px; color: #ccc; text-align: center; width: 40px; height: 50px; padding: 5px 0; line-height: 40px; display: inline-block;">
                39
            </div>
            <div style="margin-right: 10px; cursor: pointer; border: 1px solid #ccc; border-radius: 6px; color: #ccc; text-align: center; width: 40px; height: 50px; padding: 5px 0; line-height: 40px; display: inline-block;">
                40
            </div>
            <div style="margin-right: 10px; cursor: pointer; border: 1px solid #ccc; border-radius: 6px; color: #ccc; text-align: center; width: 40px; height: 50px; padding: 5px 0; line-height: 40px; display: inline-block;">
                41
            </div>
        </div>
    </div>
    <div style="position: absolute; left: 0; right: 0; bottom: -20px; margin-left: auto; box-shadow: 0px 0px 30px 5px rgba(255, 140, 0, 0.6); margin-right: auto; width: 230px; cursor: pointer; height: 45px; background-color: darkorange; color: white; text-align: center; line-height: 45px; border-radius: 32px;">
        I want this
    </div>
</div>

```

Kod HTML po refaktoryzacji

```html
<div class="app-card">
    <div class="icons">
        <div class="discount-div">
            <i class="fas fa-tag"></i>
            <div class="discount-number">{{discount}}% discount</div>
        </div>
        <i class="fas fa-share-square"></i>
    </div>
    <div class="image"></div>
    <div class="stars">
        <div *ngFor="let star of stars">
            <i (click)="rateProduct(star)" [class.star-yellow]="star <= rate" class="fas fa-star">
</i>
        </div>
    </div>
    <div class="heading">
        <div class="title">
            {{productTitle}}
        </div>
        <div class="subtitle">
            {{productSubtitle}}
        </div>
    </div>
    <div class="price">
        <div class="old-price">
            {{oldPrice | currency}}
        </div>
        <div class="new-price">
            {{newPrice | currency}}
        </div>
    </div>
    <div class="sizes">
        <div *ngFor="let size of sizes">
            <div (click)=selectSize(size) [class.size-box-selected]="size === selectedSize"
                 class="size-box">{{size}}</div>
        </div>
    </div>
    <div class="button">
        I want this
    </div>
</div>

```

Proces refaktoryzacji

1. W pierwszy etapie oddzieliłem style elementów i umieściłem je w oddzielnym pliku z rozszerzeniem ***.scss** *(Można wyraźnie zobaczyć różnice w ilości turkusowego koloru odpowiedzialnego za style)*

2. W drugim etapie teksty statyczne (na biało) przypisałem do wcześniej utworzonych zmiennych w pliku ***.ts** *(W przyszłości zmiany nie będą wymagać ingerencji w kod HTML)*

3. W trzecim etapie pozbyłem się redundancji kodu tj. części powtarzającego się kodu. Tam gdzie można użyłem dyrektywy angularowej ***ngFor** i zamiast kilka razy powtarzać ten sam kod, umieściłem go raz i uruchomiłem pętle.

4. W czwartym etapie stworzyłem zmienne w pliku ze stylami, aby łatwiej móc w przyszłości szybko zmienić kolorystykę i wielkość niektórych elementów.

5. W piątym etapie utworzyłem oddzielny plik ze stylami, w którym umieściłem w formie kilku zmiennych paletę kolorów wykorzystywanych przy implementacji designu karty.

6. W szóstym etapie zaimplementowałem technologię ***flex-box*** w celu sprawniejszego pozycjonowania elementów HTML-owych.

Istotne zmiany w kodzie HTML-owym:

- **Linijka 5.** - zmienna discount przechowuje liczbę, którą łatwo można zmienić
- **Linijka 10.** - implementacji pętli do renderowania pięciu gwiazdek do ocen
- **Linijka 12. i 33.** - przechwycenie eventu click w celu wystawienia oceny. Użycie class injection w zależności od spełnionego warunku
- **Linijka 25. i 28.**- dodanie do wyświetlanego tekstu formatowania na walutę z oznaczeniem dolara

Istotne zmiany w kodzie CSS-owym:

- **Linijka 96.** - zamiast używać wielkich liter w HTML-u, dodałem atrybut text-transform: uppercase
- **Linijki 1. - 15.** - import palety kolorów i implementacja zmiennych
- **Linijki 131. - 133.** - implementacja technologii flex-box
- Zagnieżdżone klasy potrafią lepiej zrozumieć strukturę pliku HTML-owego i ułatwiają ewentualną edycję styli *(podobnie w przypadku zmiennych)*
- **Linijki 57. - 62.** - zamiast umieszczać obraz za pomocą znacznika ***<img>*** zastosowałem manewr z tłem dla kontenera i ustawiłem wielkość tła, aby pokrywało całą dostępną powierzchnię pozbywając się tym samym obowiązkiem skalowania obrazu.
- **Linijki 109. - 111.** - współdzielone atrybuty, aby nie musieć powielać kodu

Kod CSS-owy dostępny pod tym linkiem: [> Plik CSS po refaktoryzacji <]

Kod TS przed refaktoryzacją:

```ts
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'card',
  templateUrl: './card.component.html',
  styleUrls: ['./card.component.scss']
})
export class CardComponent implements OnInit {

  constructor() { }

  ngOnInit() {
  }

}

```

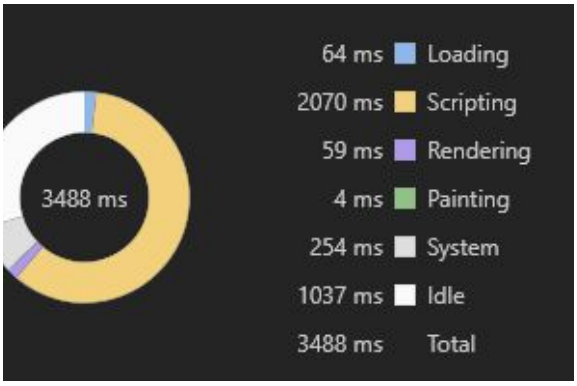
Kod TS po refaktoryzacji:

```ts
import { Component } from '@angular/core';

@Component({
    selector: 'better-card',
    templateUrl: './better-card.component.html',
    styleUrls: ['./better-card.component.scss']
})
export class BetterCardComponent {
    discount = 20;
    stars = this.returnArray(5);
    rate = 3;
    productTitle = 'Nike free run v2';
    productSubtitle = 'Men Orangeblow Darken Sportswear Sneakers';
    oldPrice = 178.47;
    newPrice = 74.59;
    sizes = [38, 39, 41, 42];
    selectedSize = 39;

    returnArray(length: number): Array<any> {
        return Array.from({ length }, (v, k) => k + 1);
    }

    selectSize(size: number) {
        this.selectedSize = size;
    }

    rateProduct(rate: number) {
        this.rate = rate;
    }
}

```

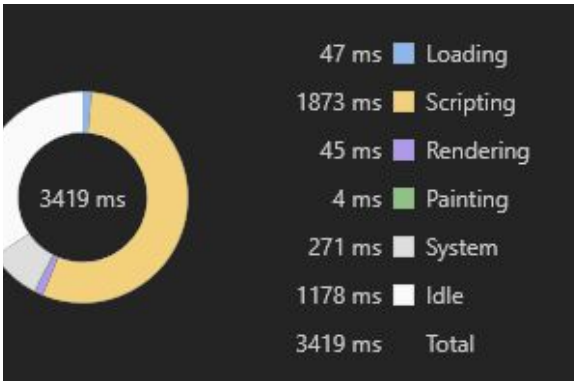
Istotne zmiany w kodzie TS-owym:

- **Linijki 9. 17.** - deklaracja zmiennych “trzymająca” dane nt. produktu. Po zmianie karta dynamicznie zmienia swój kontent
- **Linijka 19.** - implementacja funkcji zwracającej tablicę potrzebną do wykonania wcześniej wspomnianej pętli
- **Linijka 23.** - implementacja funkcji wyboru rozmiaru
- **Linijka 27.** - implementacja funkcji wystawienia oceny

Optymalizacja i szybkość kodu (statystyki)

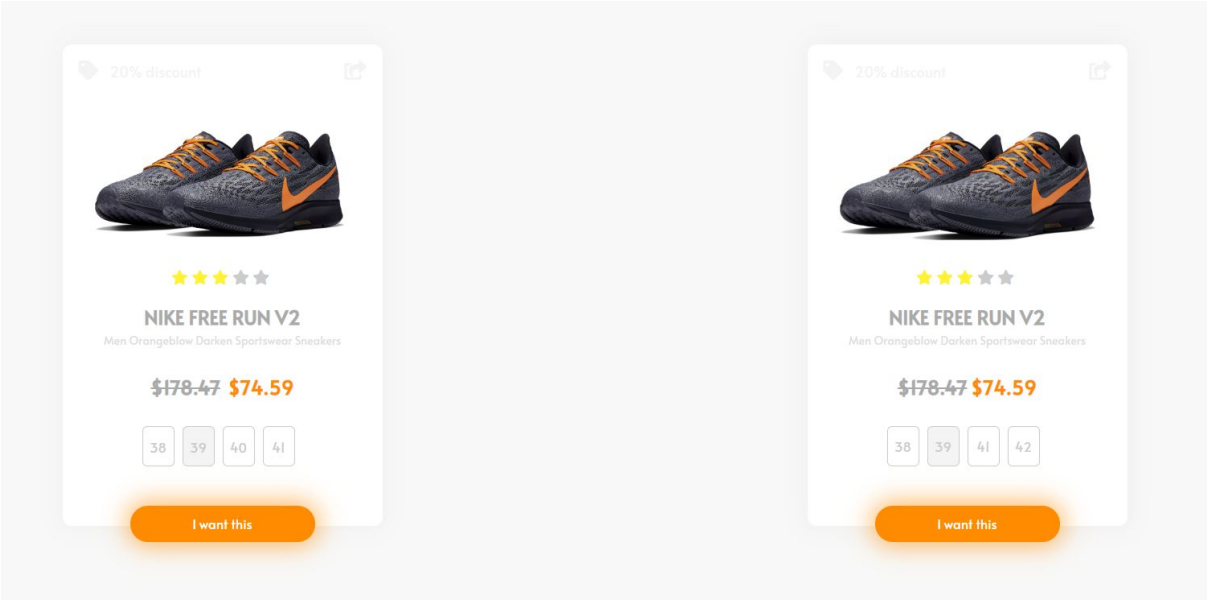


Przed refaktoryzacją



Po refaktoryzacji

Choć różnice nie są zbyt duże to wciąż zauważalne. Przy większych projektach różnica może być zauważalna przy wczytywaniu strony toteż warto dbać o poprawny, czysty i podatny na zmiany kod, aby wyeliminować błędy ludzkie oraz spowolnienie przy ładowaniu.



Podsumowanie refaktoryzacji

Choć powyższe zestawienie dwóch kart zakodowanych w dwojaki sposób (przed i po refaktoryzacji) nie mówi nam nic to zagłębienie się w kod mówi nam sporo o jakości i czystości kodu, w którym poziom trudności w wykonaniu jakichkolwiek zmian różni się diametralnie. Kod po refaktoryzacji mamy uporządkowany i podzielony na moduły, a wszelkie zmiany kontentu i stylistyki wymagają jedynie zmiany kilku zmiennych - co za tym idzie, nie trzeba przeszukiwać całego bloku kodu w poszukiwaniu jednej wartości, którą chcemy zmienić. Style umieszczone są w dwóch plikach (paleta barw ze zmiennymi i główny plik do wszystkich klas), kod HTML został znacznie zredukowany i pozbawiony styli (zostały referencje do ww. klas), a zmienne statyczne tj. tytuł, opis, ocena etc. przechowywane są w zmiennych zadeklarowanych w oddzielnym pliku.