

## 1. Podstawy HTML, CSS i Sass



## Wyzwania:

- poznasz podstawy HTML i CSS,
  - zaczniesz korzystać ze zmiennych w SCSS,
  - zakodujesz swoją pierwszą stronę i nadasz jej unikalny wygląd.

## Wprowadzenie

W dzisiejszym świecie trudno znaleźć człowieka, który nie korzysta z internetu. Jednak o wiele mniej osób wie, jak skonstruowane są strony internetowe, oraz jak je tworzyć.

Tymczasem nie jest to wiedza tajemna ani tym bardziej niesamowicie skomplikowana. Wręcz przeciwnie! Ten kurs pokaze Ci, jak zacząć swoją przygodę z tworzeniem stron i aplikacji, a także jak później samodzielnie poszerzać swoją wiedzę. Rozpoczniemy jednak od rzeczy najważniejszej, czyli od podstaw.

Bazowymi technologiami służącymi do budowy stron internetowych są **HTML**, **CSS** i **JavaScript**. Odpowiadają one, kolejno, za strukturę strony, jej wygląd i interaktywność. Jeżeli porównalibyśmy stronę internetową do robota, to HTML byłby jego "ciałem" (obudową), CSS – lakierelem, którym jest pokryta, a JavaScript – oprogramowaniem, które jest odpowiedzialne za to, jak robot się zachowuje.



Jak się domyślasz, zanim "pomalujemy" naszego "robotu" oraz napiszemy do niego "oprogramowanie", musimy go zbudować. Dlatego najpierw zapoznajmy się z HTML-em, a nieco później w tym module – z CSS-em! A już za kilka tygodni zacznijmy zajmować się JS-em, czyli logiką działania interaktywnych elementów strony!

## 1.1. Wstęp do HTML



Zacznijmy od odpowiedzi na pytanie: **czym właściwie jest HTML?**

HTML (ang. *HyperText Markup Language*) jest podstawowym językiem do budowy stron internetowych. Porządkuje on treści na stronie i nadaje im znaczenie: za jego pomocą dzielimy treść na sekcje, nagłówki, akapity itd. HTML możesz szybko i prosto zobaczyć na własne oczy:

1. Otwórz dowolną stronę internetową.
  2. Kliknij gdziekolwiek na ekranie prawym przyciskiem myszy i wybierz opcję "Pokaż źródło strony".
  3. Otwórz HTML!

## Czym są tagi?

## Podstawowe tagi HTML

## Ćwiczenie – stosowanie tagów

## Struktura dokumentu HTML

Podsumowanie

## Zadanie: Twoja pierwsza strona

## 1.2. Wstęp do CSS



[Powrót do dashboardu](#)

```

    relationals/p>/strong> with our sponsors. So we try to go beyond good ol' bag inserts and conference stands. We integrate sponsors into the conference experience, starting from lunch
    sponsorships. We have a great sponsor page where you can see all the sponsors and what they do. We also have a sponsor lounge where you can meet them in person. If you're interested in becoming a sponsor, please email us at <a href="mailto:marketing@washdc.org">marketing@washdc.org</a>. Sponsorships are available for different levels, so let's talk options!
    <br><br>Sponsorships are a great way to support the conference and help us keep it running. So let's talk sponsorships. Please get back to us as soon as you
    can. We'd love to tell you all about our sponsorships and how they can benefit your business.
  
```

Na początku może wydawać Ci się on jednym wielkim balaganem, ale nie przejmuj się – w miarę postępów w kursie wszystko będzie coraz bardziej zrozumiałe. Na przykład, powyższy obrazek pokazuje *zminifikowany* kod HTML, czyli taki w którym usunięto znaki nowej linii i wcięcia kodu, o których dowiesz się już niedługo. Dlatego kod HTML pisany przez nas będzie znacznie bardziej czytelny!

Kod HTML jest zapisywany w tekstowych plikach z rozszerzeniem `.html`. Gdy otworzymy taki plik w przeglądarce internetowej, zawarty w nim kod HTML zostanie *zinterpretowany* i zostanie wyświetlona tylko treść bez kodu.

Spójrz na przykład poniżej: jest to bardzo prosta strona internetowa. Po lewej stronie znajduje się jej kod HTML, a po prawej to, co wyświetli przeglądarka:

The screenshot shows the CodePen interface. On the left, under the 'HTML' tab, is the source code:

```

<h1>Hello world!</h1>
<h2>Oto moja pierwsza strona
internetowa.</h2>

```

On the right, under the 'Result' tab, is the rendered output:

Hello world!

Oto moja pierwsza strona  
internetowa.

Below the tabs are buttons for 'Resources', '1x', '0.5x', '0.25x', and 'Rerun'.

Jak widzisz, przeglądarka nie pokazuje całej zawartości pliku HTML, tylko **treści** umieszczone między specjalnymi *wyrażeniami*, zwanyymi **znacznikami** (lub, z angielskiego, *tags*). W naszym przykładzie tagami są na przykład `<h1>` oraz `<h2>`. Spróbuj zmodyfikować treści w powyższym edytorze (np. zmienić powitanie "Hello world" na inną), nacisnąć przycisk "Rerun" w prawym dolnym rogu edytora i zobaczyć, co się stanie!

**Zagnieżdżone edytory kodu**

W trakcie kursu często będziemy używać zagnieżdzonego edytora kodu **CodePen**, takiego jak powyżej. Możesz swobodnie modyfikować jego zawartość, pamiętaj jednak, że są to zmiany tymczasowe i **nie są nigdzie zapisywane**. Dzięki temu możesz od razu eksperymentować z kodem przykładowu, przejmując się, czy coś się zepsuje. Jeśli tak się stanie, wystarczy że odświeżysz stronę, aby ponownie wczytać nasz przykład.

Jeśli zechcesz zapisać swoje zmiany, kliknij w logo CodePen w rogu edytora. Dzięki temu będziesz mieć możliwość zapisania własnej wersji. Z tego względu warto założyć konto w serwisie **CodePen**, aby mieć później dostęp do swoich przykładów.

## Czym są tagi?

Tagi to podstawowe składowe, budujące kod HTML. Mówiąc w skrócie, tag to specjalne słowo kluczowe, umieszczone wewnętrz ostrych nawiasów (znak mniejszości i większości). Nazwa tagu (czyli właśnie wspomniane słowo kluczowe) wskazuje przeglądarce, jakiego rodzaju treść jest zawarta wewnątrz tagu. Innymi słowy, za pomocą tagów możemy przekazać przeglądarce, że dana treść jest np. nagłówkiem, linkiem, obrazkiem lub blokiem tekstu. Tag wraz z treścią stanowi **element HTML**.

Przykładowy element HTML wygląda w ten sposób:

```
<h1>Hello world!</h1>
```

Składa się on, kolejno, z tagu otwierającego, treści i tagu zamk傢cnego.

### Jak zbudowany jest tag?

- Tagi HTML najczęściej występują w parach, np. tak jak w powyższym przykładzie `<h1>` i `</h1>` (tagi nagłówka – ang. *heading*).
- Pierwszy tag w parze to **tag otwierający**, a drugi to **tag zamk傢cący**.
- Tag otoczony jest ostrymi nawiasami.

- Treść elementu (w powyższym przykładzie jest to "Hello World") znajduje się między tagiem otwierającym a zamkającym.
- Tag zamkający wygląda tak samo, jak tag otwierający, z tą różnicą, że po pierwszym ostrym nawiasie ma znak / (slash).

#### Ważne: zamknięcie tagów

Zdecydowana większość tagów występuje w parach: potrzebny jest **tag otwierający** i **tag zamkujący**. Jeżeli tag potrzebuje zamknięcia, nie możesz go pominąć, bo może się to skończyć błędami na stronie.

Zapamiętaj strukturę tagu z otwarciem i zamknięciem:

```
<p>Hello world!</p>
```

Istnieją jednak również tagi samozamykające, których nie trzeba domykać, np. tag <br>, który oznacza złamanie linii, lub tag <img>, który pozwala dodać zdjęcie na stronę. Będziemy zwracać Twoją uwagę, gdy jakiś tag nie będzie wymagał zamknięcia. Jeżeli nie powiemy wyraźnie inaczej, zamkij każdy tag, który stworzysz w ramach tego projektu.

## Podstawowe tagi HTML

Tagów HTML jest całe mnóstwo – poniżej przedstawimy kilka najczęściej używanych.

### Nagłówki

```
<h1>Nagłówek poziomu pierwszego</h1>
<h2>Nagłówek poziomu drugiego</h2>
<h3>Nagłówek poziomu trzeciego</h3>
<h4>Nagłówek poziomu czwartego</h4>
<h5>Nagłówek poziomu piątego</h5>
<h6>Nagłówek poziomu szóstego</h6>
```

### Akapit tekstu

```
<p>To jest akapit tekstu.</p>
```

### Link

```
<a href="https://kodilla.com">Link do strony Kodilla</a>
```

### Obrazek

```

```

### Lista nieuporządkowana

```
<ul>
    <li>Element listy</li>
    <li>Element listy</li>
    <li>Element listy</li>
</ul>
```

Efekt na stronie:

- Element listy
- Element listy
- Element listy

### Lista uporządkowana

```
<ol>
    <li>Element listy</li>
    <li>Element listy</li>
    <li>Element listy</li>
</ol>
```

Efekt na stronie:

1. Element listy

2. Element listy
3. Element listy

Oczywiście nie musisz uczyć się wszystkich tagów i atrybutów HTML na pamięć! Ich listę wraz z opisami i przykładami użycia znajdziesz na stronie [HTML Reference](#).

## Ćwiczenie – stosowanie tagów

Przećwiczmy teraz powyższe tagi, rozbudowując nieco naszą prostą stronę. Postępuj zgodnie ze wskazówkami poniżej, modyfikując kod w lewej części edytora i obserwując efekt w prawej części:

The screenshot shows a live code editor interface. On the left, under 'HTML', the code is:

```
<h1>Hello world!</h1>
<h2>Oto moja pierwsza strona
internetowa.</h2>
```

On the right, under 'Result', the output is:

Hello world!

Oto moja pierwsza strona  
internetowa.

At the bottom, there are buttons for 'Resources', zoom levels (1x, 0.5x, 0.25x), and a 'Rerun' button.

1. Pod nagłówkiem poziomu drugiego (`<h2>`) wstaw **akapit tekstu** z krótkim tekstem o sobie.
2. Pod akapitem tekstu wstaw **link**, prowadzący do dowolnej strony internetowej.
3. Pod linkiem umieść **obrazek**, używając poniższego kodu:

```

```

W tym miejscu zatrzymaj się na moment i przypatrz tagowi `<img>`. Zauważ, że posiada on kilka dodatkowych właściwości, zwanych fachowo **atributami**: `src` (wskażające adres grafiki, która powinna zostać wyświetlona, z angielskiego *source*, czyli źródło), `alt` i `title`. Wiele tagów HTML posiada unikalne dla siebie właściwości (czyli takie, które zadziałają tylko dla konkretnego rodzaju tagu), ale istnieją też właściwości, które możemy nadać każdemu tagowi. O tych drugich będziemy mówić już wkrótce.

4. Poniżej obrazka wstaw **listę nieuporządkowaną** z trzema tagami `<li>`, gdzie umieścisz tytuły filmów, które lubisz.

Ostateczny efekt może wyglądać na przykład tak:

**Pokaż przykładowe rozwiązanie**

Jak widzisz, HTML jest naprawdę prosty! :) Kluczową umiejętnością jest stosowanie właściwych tagów do danych typów treści (nagłówki, akapity tekstu itp.) oraz staranne pilnowanie, by kod był czytelny i uporządkowany. Wtedy na pewno się w nim nie zgubisz!

Za chwilę czeka Cię pierwsze prawdziwe zadanie, które wyślesz swojemu Mentorowi do sprawdzenia. Zanim jednak do niego przejdziemy, musimy wyjaśnić sobie jeszcze jedną rzecz.

Używany przez nas edytor CodePen, w którym pokazujemy fragmenty kodu, wykorzystuje nieco uproszczoną strukturę dokumentu HTML. Już za chwilę będziesz mieć do czynienia z bardziej rozbudowanym edytorem, co oznacza konieczność użycia pełnej struktury. Jak ona wygląda? Już wyjaśniamy.

Każdy dokument HTML ma ścisłe określona **strukturę podstawową**. Struktura to kod początkowy, który w zasadzie powtarza się w każdym pliku HTML. Dopiero w ramach tej struktury dodajemy swoją treść i inne tagi. Struktura wygląda następująco:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>My first website</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
  </body>
</html>
```

Poszczególne tagi naszej podstawowej struktury to:

- Deklaracja `!DOCTYPE`, która określa typ tego dokumentu jako HTML.
- Treść pomiędzy `<html>` i `</html>` to nasz dokument HTML. Ten tag jest obowiązkowy i otacza (*wrapuje*) wszystkie pozostałe tagi. Dobrą praktyką jest dodanie do tego taga atrybutu `lang`, który określa, w jakim języku napisane są treści na stronie (np. `en` - angielski, `pl` - polski).
- Treść pomiędzy `<head>` i `</head>` dostarcza informacji o stronie, które są ważne dla przeglądarki, ale najczęściej niewidoczne dla użytkownika.
- Treść w tagach `<meta>` zawiera konkretne informacje ważne dla przeglądarki. Aby poprawnie wyświetlały się polskie znaki, za pomocą tagu meta dodajemy kodowanie utf-8 (`<meta charset="utf-8">`). Tagi `<meta>` zawsze są zagnieżdżone w `<head></head>`.
- Treść pomiędzy `<title>` i `</title>` to tytuł strony. Widać go na samej górze okna przeglądarki, na zakładce (tab).
- Treść w tagach `<link>` to zazwyczaj podpięcie arkusza stylów – omówimy to dokładniej w kolejnym submodule.
- Treść pomiędzy `<body>` i `</body>` jest tym, co widzisz potem w przeglądarce. Wewnątrz tagu `<body>` wstawiasz całą treść swojej strony – teksty, zdjęcia itp.

Przyjrzyj się jeszcze raz powyższej strukturze i zwróć uwagę, że niektóre elementy umieszczone są wewnętrz innych elementów (np. tag `<meta>` znajduje się wewnętrz `<head>`). Fachowo nazywa się to **zagnieżdżaniem** (ang. *nesting*); mówimy także, że jeden element jest zagnieżdżony (*nested*) w innym. Element zewnętrzny to inaczej rodzic (ang. *parent*), a element zagnieżdżony w nim to dziecko (ang. *child*). Dwa elementy umieszczone na tym samym poziomie zagnieżdżenia w rodzicu są rodzeństwem (ang. *sibling*). Łatwo skojarzyć! :)

Co jest ważne przy zagnieżdżaniu elementów, to to, by pilnować **wcięć**, czyli fachowo *indentacji*. Wcięcia tworzymy za pomocą klawisza Tab. Elementy-dzieci znajdują się dalej od lewej strony ekranu niż elementy-rodzice: każdy nowy poziom zagnieżdżenia to o jedno wcięcie (Tab) więcej. Zwiększa to czytelność kodu i sprawia, że od razu wiadomo, gdzie kończy się jeden tag i zaczyna kolejny.

Od samego początku wdrażaj się do starannego pisania kodu i wstawiania wcięć w prawidłowych miejscach. Dobry i profesjonalny kod poznaje się po tym, że nie tylko działa bez zarzutu, ale również wygląda elegancko. Na to będą również zwracać uwagę rekruterzy, gdy będziesz poszukiwać pracy jako developer!

#### Ważne: struktura sekcji `<head>`

Utrzymuj porządek w sekcji `<head>`. Najpierw umieść w niej tag `<meta>`, a następnie `<title>` i w końcu `<link>`.

## Podsumowanie

HTML (ang. *HyperText Markup Language*) jest językiem służącym do konstruowania struktury stron WWW. Kod HTML zbudowany jest z elementów, na które składają się tagi i treści wewnętrz nich. Tagi informują przeglądarkę, z jakim rodzajem treści ma do czynienia.

#### Terminy do zapamiętania:

- tag (znacznik)
- atrybut
- element HTML

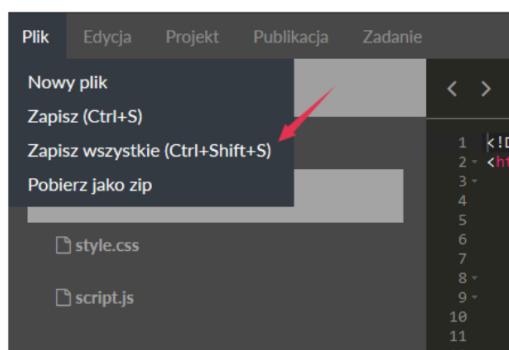
## Zadanie: Twoja pierwsza strona

### Pierwsze spotkanie z edytorem Kodilla

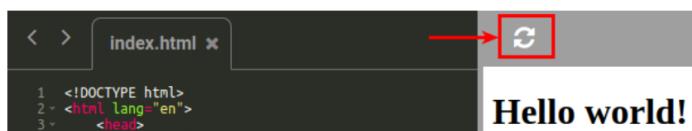
Po przeczytaniu instrukcji zadania kliknij znajdujący się pod nim przycisk. Otworzy się wtedy edytor kodu, w którym wykonasz poniższe polecenia.

Edytor dostarcza Ci gotowe środowisko, gdzie od razu możesz pisać swój kod. Sam edytor jest podzielony na trzy kolumny. W lewej znajduje się lista plików projektu, w środkowej edytujemy pliki, a w prawej widzimy efekty naszych działań. Aby dowiedzieć się więcej o dostępnych opcjach, kliknij link *Jak używać tego edytora?* w lewym dolnym rogu ekranu.

Pamiętaj, że po każdej zmianie należy zapisać wszystkie pliki projektu (*Plik->Zapisz wszystkie*):



Następnie odśwież podgląd (ikonka odświeżania w widoku strony albo Ctrl+R):



Nasze pliki możemy także zapisać i otworzyć lokalnie na komputerze. Wystarczy wybrać opcję *Plik->Pobierz jako zip*. Ciekawą opcją jest także publikacja projektu. Aby zobaczyć projekt na pełnym ekranie (bez widoku edytora), wystarczy wybrać opcję *Publikacja->Udostępnij*, a następnie skopiować link.

Po skończonej pracy wysłesz swój projekt do Mentora, klikając opcję *Wyslij do sprawdzenia* na szarym pasku.

Po sprawdzeniu efektu zadania Mentor skomentuje Twój kod i poinformuje Cię o tym.

### Opis zadania

Zadanie będzie polegało na stworzeniu podstawowej strony w HTML, podążając za instrukcjami poniżej. Strona może być poświęcona Twójemu ulubionemu aktorowi, sportowcowi, filmowi, książce, grze... temat możesz wybrać samodzielnie!

1. Do pliku `index.html` w edytorze wstaw **podstawową strukturę HTML**, omówioną powyżej. Skasuj linię zawierającą tag `<link>`: nie będzie ona nam na razie potrzebna.
2. Dodajmy teraz treść strony pomiędzy tagami `<body>` a `</body>` (jak pamiętasz, to właśnie tam umieszczamy całą zawartość naszej witryny). Zaczniemy od dodania nagłówka pierwszego poziomu z tytułem strony, np. imieniem i nazwiskiem aktora czy tytułem filmu.

3. Stwórz 2 akapity tekstu i nad każdym z nich nagłówek poziomu drugiego.
4. Wypełnij nagłówki i akapity treścią, tak, aby zaprezentować odbiorcy temat Twojej strony.
5. Poniżej dodaj 2 zdjęcia, np. z darmowego serwisu [Pexels](#). Pobierz link do interesującego Cię zdjęcia, klikając na wybranym obrazku prawym przyciskiem myszy i wybierając opcję *Kopij adres obrazu*. Skopiowany adres wstaw do tagu `` na stronie.
- **nawias klamrowy ( { } )** – wstawiamy go zaraz po selektorze. Wewnątrz nawiasów klamrowych umieszczamy wszystkie właściwości CSS dla danego elementu.
- **właściwość** (ang. *property*) – określa, co chcemy zmienić w wyglądzie danego elementu HTML. Po właściwości wstawiamy **drukopiek**. W naszym przykładzie użyliśmy właściwości `color`, odpowiedzialnej za kolor czcionki w elemencie.
- **wartość** (ang. *value*) – określa, w jaki sposób chcemy zmodyfikować daną właściwość. W przykładzie nadaliśmy naszej właściwości wartość `green`, co oznacza, że kolor czcionki zmieni się na zielony.
- **średnik** – wstawiamy go po każdej wartości w CSS.

Jeżeli chcemy przypisać więcej niż jedną właściwość do selektora, robimy to w ten sposób:

```
p {
  color: green;
  font-size: 20px;
  font-weight: bold;
}
```

Zwróć uwagę, że w języku CSS, tak samo, jak w HTML, stosujemy **wcięcia**: każda właściwość wewnętrz nawiasów klamrowych poprzedzona jest Tabem.

Zerknij na przykład poniżej, a potem spróbuj zmodyfikować w nim kod CSS, zastępując słowo `green` słowem `red`:

```
HTML      CSS      LIVE      Result      EDIT ON      CDEPEN
<h1>Hello world!</h1>
<h2>Oto moja pierwsza strona
internetowa.</h2>
<h2>Oto drugi nagłówek na mojej
stronie.</h2>

Resources      1x 0.5x 0.25x      Rerun
```

Świetnie! Teraz w kodzie CSS, w nowej linii, dopisz podobną regułę, która zmieni kolor elementów `h2` na niebieski (`blue`).

Udało się? Znakomicie! A co, gdybyśmy chcieli, żeby tylko pierwszy element `h2` był niebieski? Istnieje kilka sposobów na wybranie w CSS dokładnie tego elementu (lub elementów), które chcemy: omówimy je sobie poniżej.

## Stylowanie po tagu

W powyższym przykładzie zastosowaliśmy w regule CSS selektory dla tagów `<h1>` i `<h2>`. Jeżeli chcemy nadać jakieś style dla konkretnego taga z HTML, wystarczy, że użyjemy jego nazwy jako selektora.

Przykładowa reguła dla elementu `<h1>`:

```
h1 {
  background: black;
}
```

Taka reguła sprawi, że wszystkie nagłówki `<h1>` będą miały czarny kolor tła.

## Stylowanie po klasie

Oprócz stylowania elementów po tagu możemy stylować je wykorzystując ich **klasy**. Jest to najczęściej spotykany typ selektora.

Klasy nadajemy w pliku HTML w poniższy sposób:

```
<h2 class="intro">Treść nagłówka</h2>
```

Klasę możemy nadać każdemu elementowi HTML.

Jak użyć klasy przy stylowaniu? Przykładowo, jeżeli mamy element HTML z atrybutem

`class="banner"`, to możemy go ostylować w następujący sposób:

```
.banner {  
  color: red;  
}
```

Jak widzisz, w CSS regułę z selektorem dla klasy tworzymy poprzez dodanie kropki przed nazwą klasy.

Przy okazji wszystkie inne elementy z tą klasą również otrzymają zdefiniowane w CSS reguły wyglądu.

## Stylowanie po identyfikatorze

Istnieje też selektor dla identyfikatorów (`id`). Co do zasady unikamy tego typu selektorów (wykorzystywane są często przez programistów backendu), jednak każdemu elementowi HTML z `id="name"` możesz przypisać style w CSS za pomocą selektora `#name` (dodajesz w CSS `#` przed nazwą `id`). Pamiętaj, że dany identyfikator może być przypisany tylko jednemu elementowi HTML (w przeciwieństwie do klas).

Przykład:

```
#name {  
  display: none;  
}
```

Taka reguła sprawi, że element z atrybutem `id="name"` zostanie ukryty (nie będzie się wyświetlał).

Oprócz wyżej wymienionych selektorów istnieje ich jeszcze mnóstwo wraz z różnymi kombinacjami. Poznasz je wraz ze swoim rozwojem zawodowym.

Wróć teraz do naszego przykładu:

The screenshot shows the CodePen interface. On the left, the 'HTML' tab displays the following code:

```
<h1>Hello world!</h1>
<h2>Oto moja pierwsza strona
internetowa.</h2>
<h2>Oto drugi nagłówek na mojej
stronie.</h2>
```

On the right, the 'Result' tab shows a blank white page. The top bar includes tabs for 'HTML', 'CSS', and 'LIVE', along with the 'CODEPEN' logo. At the bottom, there are buttons for 'Resources', zoom levels (1x, 0.5x, 0.25x), and a 'Rerun' button.

Pierwszemu nagłówkowi `<h2>` na stronie nadaj klasę `first`, a następnie użij tej klasy do nadania temu elementowi w CSS rozmiaru czcionki (`font-size`) 22px. Drugiemu nagłówkowi `<h2>` nadaj klasę `second`, a następnie ustaw mu wielkość czcionki na 18px, a kolor na pomarańczowy (`orange`).

## Box-model w CSS

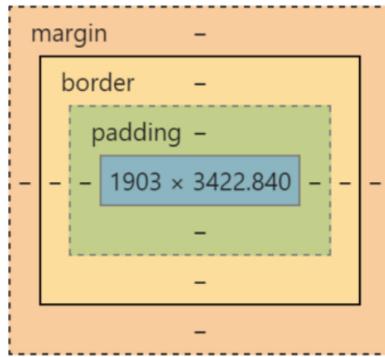
Każdy element HTML swego rodzaju "pudełkiem", a każde takie "pudełko" posiada kilka cech wyglądu, które je charakteryzują.



Wyobraź sobie pudełka umieszczane wewnętrz siebie jak matrioszki. Najmniejszym z nich jest **treść** (`content`), którą chcesz umieścić. Drugie pudełko to **padding**, czyli odległość treści od **ramki** (`border`) będącej kolejnym pudełkiem. Dla ramki też określamy jej grubość. Największym z pudełek jest **margines**, czyli `margin` – wyznacza on, w jakiej odległości od brzegów strony lub innych elementów powinien znajdować się ten element, którym się aktualnie zajmujemy.

Zrozumienie tego ułatwi Ci poniższa grafika pochodząca z narzędzia developerskiego Chrome Developer Tools, będącego częścią przeglądarki Chrome. Niebieskie pole to pudełko z treścią, które ma pewną przestrzeń (`padding`) od ramki, która też ma pewną grubość (`border`). Za pomocą margin określamy odległość innych elementów HTML od bordera.

W tym rozdziale poznajemy m.in. konstrukcję składnego pudełka CSS, której zadaniem jest oddzielenie treści od bordura naszego pudełka z treścią.



Jest to dość subtelna koncepcja, której nie musisz w tej chwili dokładnie rozumieć na poziomie teoretycznym. Ważne, by umieć zastosować ją w praktyce. Poówiczmy więc! Działamy dalej na naszym przykładzie:

The screenshot shows the CodePen interface with the 'LIVE' preview tab selected. The preview area displays the following HTML content:  
`<h1>Hello world!</h1>  
<h2 class="first">Oto moja pierwsza strona internetowa.</h2>  
<h2 class="second">Oto drugi nagłówek na mojej stronie.</h2>`

W kodzie CSS każdemu elementowi `<h2>` nadaj ciągłą ramkę (`border`) o szerokości 1 piksela i w kolorze niebieskim. Ta właściwość będzie wyglądała następująco:

```
border: 1px solid blue;
```

Aby odsunąć zawartość od ramki, nadajmy dodatkowo tym elementom `padding` równy 10px.

Na końcu odsuńmy nasze pudełko od lewej strony ekranu, nadając elementowi o klasie `first` właściwość `margin-left` równą 10px, a elementowi z klasą `second` – `margin-left` o wartości 30px.

Wskutek tych zabiegów, nasza treść powinna wyglądać tak:

The screenshot shows the CodePen interface with the 'LIVE' preview tab selected. The preview area displays the following CSS code:  
`h1 {  
 color: green;  
}  
h2 {  
 border: 1px solid blue;  
 padding: 10px;  
}  
.first {  
 font-size: 22px;  
 margin-left: 10px;  
}  
.second {  
 font-size: 22px;  
 margin-left: 30px;  
}`

## Składowe pudełek

Nasze "pułapkę", w przeciwieństwie do tych prawdziwych, nie muszą mieć wszystkich ścian. Możliwe jest na przykład ustawić tylko jednej części ramki dla elementu HTML. W tym celu do nazwy właściwości CSS (`padding`, `border`, `margin`) dodajemy przyrostek definiujący kierunek:

- `top`, jeśli chcemy ustawić właściwość dla góry elementu HTML;

- `right`, gdy zależy nam na ustawieniach dotyczących prawej strony;
- `bottom` dla właściwości dotyczących dołu;
- `left`, gdy określamy właściwości po lewej.

Zatem chcąc ustawić margines na górze elementu, skorzystamy z właściwości `margin-top`, np. `margin-top: 30px`;

## Wartości "po zegarze"

Wiesz już, jak dodać marginesy tylko z jednej strony elementu. Co w przeciwnym przypadku – gdy Twoim zamiarem jest dodanie marginesu na każdym boku pudełka oprócz góry? Czy trzeba napisać 3 linijki kodu, odpowiadające za właściwości `margin-right`, `margin-bottom` i `margin-left`, zostawiając `margin-top` domyślnie ustawione na 0?

Na szczęście nie – w takim przypadku wykorzystamy właściwość `margin`, która jest skróconym zapisem (*shorthand*) dla czterech powyższych właściwości. Jeśli dla `margin` podamy cztery wartości, np. `margin: 0 10px 20px 30px`, ustawimy inny margines dla każdego brzegu elementu. Skąd jednak wiedzieć, w jakiej kolejności wpisywać wartości? To proste – podajemy je zgodnie z ruchem wskazówek zegara, zaczynając od górnej. Zatem w podanym przykładzie nie mielibyśmy górnego marginesu (jego grubość jest ustawiona na 0px), prawy miałby 10px, dolny 20px, a lewy 30px grubości.

### Dla dociekliwych

Przeglądarki czytają arkusze stylów, które im dostarczamy i je interpretują. To już wiemy. Jak poradzą sobie z takimi zapisami?

```
margin: 10px 5px 10px 5px;
```

Otoż przeglądarka tłumaczy sobie taki zapis na porozbijane, dokładniejsze zapisy, jak w przykładzie poniżej:

```
margin-top: 10px;
margin-right: 5px;
margin-bottom: 10px;
margin-left: 5px;
```

Spostrzegawcze osoby zauważą, że dla górnego i dolnego marginesu wartości są takie same. Również dla bocznych mają one tę samą wartość. Może da się to jeszcze jakaś uproszczyć? Oczywiście!

```
margin: 10px 5px 10px 5px;
```

Powyższy zapis możemy również przedstawić w ten sposób:

```
margin: 10px 5px;
```

Przeglądarka i tak go sobie przetłumaczy tak:

```
margin-top: 10px;
margin-right: 5px;
margin-bottom: 10px;
margin-left: 5px;
```

W podobny sposób zachowuje się np. właściwość `padding`.

Możesz poćwiczyć nadawanie właściwości "po zegarze", definiując inne marginesy dla każdej strony elementów w naszym przykładzie:

The screenshot shows the CodePen interface with the CSS tab selected. The CSS code is as follows:

```
h1 {
  color: green;
}
h2 {
  border: 1px solid blue;
  padding: 10px;
}
.first {
  font-size: 22px;
}
```

The preview window shows the text "Hello world!" in green and "Oto moja pierwsza strona internetowa." in black, enclosed in a blue-bordered box with 10px padding.

## Najczęściej używane style

Jest pewna popularna grupa właściwości CSS, których używa się bardzo często. Są to:

- `color` – określa kolor tekstu w elemencie, np. `color: black;`,
- `border` – określa parametry ramki elementu np. `border: 2px solid #ccc;`, co da nam ramkę o grubości 2px, ciągłą (`solid`), koloru szarego,
- `display` – wyznacza sposób wyświetlania elementu, np. `display: none;`,
- `height` – odpowiada za wysokość podawaną najczęściej w pikselach (`px`) lub procentach (`%`), np. `height: 200px;`,
- `width` – determinuje szerokość, np. `width: 100px;`,
- `margin` – odpowiada za marginesy elementów, np. `margin: 20px;` sprawi, że element będzie miał dwudziestopikselowy margines,
- `padding` – odpowiada za przestrzeń między zawartością elementu a jego ramką, np. `padding: 50px;`,
- `font-family` – określa czcionkę, która zostanie użyta do wyświetlania tekstu wewnątrz elementu, np. `font-family: Arial;`,
- `text-align` – odpowiada za rozmieszczenie tekstu, czyli wyśrodkowanie tekstu lub wyrównanie go do jednej z krawędzi (bądź obu), np. styl `text-align: center;` wyśrodkuje tekst znajdujący się w danym elemencie HTML.

Więcej informacji na temat każdej z tych właściwości oraz wartości, jakie mogą przyjmować, znajdziesz na stronie [CSS Reference](#).

## Podłączenie CSS do HTML i przeciążanie

Podczas pracy w bardziej złożonych edytorech kodu, takich jak ten wykorzystywany przez Kodillę, będziemy musieli wskazać plikom HTML ścieżkę do pliku (lub plików) CSS.

Aby napisane przez nas style zostały odnalezione i zinterpretowane przez przeglądarkę, musimy podłączyć nasz plik `style.css` do pliku `index.html`, tuż przed tagiem zamykającym `</head>`. Robimy to w następujący sposób:

```
<link rel="stylesheet" href="style.css">
```

W powyższym przykładzie za pomocą atrybutu `href` wskazujemy plik, w którym znajduje się nasz kod CSS. Metoda ta zadziała prawidłowo dla pliku `style.css` znajdującego się w tym samym katalogu, co plik HTML, do którego podpinamy style. Jeżeli plik ze stylami byłby głębszy, np. w katalogu o nazwie `stylesheets`, wtedy atrybut `href` wyglądałby następująco: `href="stylesheets/style.css"` (ścieżka względna, bez pełnego adresu pliku typu `C:\file\...`).

Style są czytane (i stosowane) przez przeglądarkę "z góry na dół". Oznacza to, że jeżeli na początku pliku stworzymy regułę nadającą akapitom kolor **czerwony**, a pod koniec pliku stworzymy podobną regułę CSS nadającą im kolor **pomarańczowy**, to w ostateczności nasze akapity będą wyświetlane na **pomarańczowo** (ostatni styl nadpisze poprzednie). Dzieje się tak, ponieważ styl z pomarańczowym kolorem tekstu znajduje się niżej, w związku z czym przeglądarka uzna ją za bardziej aktualny.

Analogicznie jest w przypadku, gdy podpinamy dwa pliki ze stylami.

```
<link rel="stylesheet" href="first-file.css">
<link rel="stylesheet" href="second-file.css">
```

Jeżeli w obu plikach znajdzie się reguła nadająca kolor tekstu akapitom, to przeglądarka użycie reguły z drugiego pliku. Po prostu styl w drugim pliku nadpisze ten zdefiniowany w pierwszym. Aby stworzyć dobry jakościowo kod, należy unikać zbyt dużej liczby nadpisów (przeciążeń) stylów.

## Podsumowanie

CSS (ang. *Cascading Style Sheets* – kaskadowe arkusze stylów) to język służący do nadawania wyglądu elementom HTML. Składa się on z deklaracji wskazujących, które elementy chcemy zmodyfikować, i w jaki sposób.

#### Terminy do zapamiętania:

- reguła
- selektor
- właściwość
- klasa
- id
- box-model (model pudełkowy)

## Zadanie: lifting strony

Użyjemy teraz naszej świeżej wiedzy na temat CSS do ostylowania strony, którą stworzyliśmy w poprzednim submodule. Otwórz tamto zadanie i miej je pod ręką: będziemy z niego kopować treści, które stworzyliśmy.

Na samym końcu, pod poleceniami, znajduje się przycisk, który przeniesie Cię do nowego projektu w edytorze. Wykonaj w nim poniższe instrukcje:

1. W pliku `index.html` wstaw **podstawową strukturę HTML**, omówioną w submodule 01.02. Tym razem nie kasuj tagu `<link>`: będzie nam niezbędny, aby przeglądarka zastosowała nasze style.
2. W pliku `style.css` napisz selektor dla tagu `<body>` i dodaj po nim klamry `{}`.
3. Do tego selektora dodaj właściwości `margin: 0;`.
4. W pliku `index.html`, wewnątrz `<body>` wstaw tag `<div>` o klasie `hero`. Pamiętaj o tagu zamykającym! Wewnątrz tego elementu umieść tag `<h1>` wraz z zawartością, skopiowany z poprzedniego zadania.
5. W pliku `style.css` dodaj selektor dla klasy `hero`, nadaj mu kolor tła (`background`) równy `#222f3e`; oraz kolor czcionki `color` o wartości `#ffffff` (biały). Dodaj też górny i dolny padding o wartości 50px, a na końcu wyśrodkuj tekst za pomocą właściwości `text-align`. Stylowanie tytułu gotowe!
6. Pod tagiem `</div>` utwórz kolejny element: `<section></section>` z klasą `intro`. Wewnątrz niego dodaj element `<div></div>` z klasą `container`. Całość powinna wyglądać tak:

```
<section class="intro">
    <div class="container">

    </div>
</section>
```

7. Wewnątrz containera wstaw dwa nagłówki i dwa akapity z tekstem, również skopiowane z poprzedniego zadania.
8. W pliku `style.css` wszystkim tagom `section` dodaj `padding` górny i dolny o wartości 36px.
9. Stwórz selektor dla elementu o klasie `container` i nadaj mu maksymalną szerokość (`max-width`) o wartości 900px oraz `margin` o wartości `0 auto`, co sprawi, że container będzie wycentrowany w poziomie.
10. Pod zamknięciem tej sekcji dodaj kolejny element `<section></section>` z klasą `gallery`, a wewnątrz niego, tak jak wcześniej, dodaj `<div></div>` z klasą `container`. Wewnątrz containera umieść zdjęcie z poprzedniego zadania.
11. W pliku `style.css` umieść selektor wybierający wszystkie tagi `img`. Dodaj im właściwości `display: inline-block;` oraz `max-width: 48%`; (więcej o właściwości `display` dowiesz się już wkrótce). Dzięki temu zdjęcia będą wyświetlane w jednym rzędzie.
12. Kolejną sekcją będzie `<section></section>` z klasą `info`, również z containerem w środku. Wewnątrz niej znajdzie się lista uporządkowana i jej nagłówek z poprzedniego zadania.
13. W pliku `style.css` nadaj sekcji `info` kolor tła (`background`) o wartości `#c8d6e5`.
14. Ostatnią sekcją będzie `<section></section>` z klasą `links`, jak zwykle z containerem. Wewnątrz niej wstaw listę nieuporządkowaną i jej nagłówek z poprzedniego zadania.
15. W pliku `style.css` nadaj wszystkim linkom kolor `#ee5253`.

Kliknij przycisk niżej, aby przejść do edytora. Po skończonej pracy wyslij efekt do Mentora.

Podgląd zadania

Przejdź do projektu ✓

## 1.3. Zaczynamy projekt!



W poprzednim zadaniu udało Ci się nadać swojej stronie nieco życia za pomocą CSS. Jednak to nie wszystko, co przy tej okazji przećwiczyliśmy: pokazaliśmy Ci także, w jaki sposób podzielić treść strony na logiczne **sekcje**. Do stworzenia tego podziału użyliśmy dwóch nowych tagów: `div` oraz `section`. Zatrzymaj się na moment, by powiedzieć o nich kilka słów, a przy tym poznać nieco historii języka HTML.

### HTML w wersji 5

Podczas nauki na bootcampie będziemy używać standardu HTML5. Być może zdarzyło Ci się zetknąć już wcześniej z web developmentem, czy to podczas prób samodzielnego kodowania, czy podczas interaktywnych kursów w internecie. Bardzo wiele materiałów edukacyjnych wykorzystuje starsze wersje kodu HTML: HTML4 lub XHTML. W tych (i jeszcze wcześniejszych) wersjach dzieliliśmy zawartość strony na sekcje za pomocą tagu `div`. Niezależnie od tego, jaką funkcję pełnił dany element na stronie (nawigacja, nagłówek, główna treść, treść poboczna, pojedynczy artykuł...), był on divem.

Od 2014 roku zalecanym standardem jest HTML5. Jedną ze zmian wprowadzonych w tej wersji jest szereg nowych tagów służących do tego, by w **semantyczny** sposób rozmieszczać elementy na stronie, biorąc pod uwagę ich funkcję. Pojawił się zatem, na przykład, tag `nav` dla nawigacji, `header` dla części nagłówkowej, `section` dla sekcji, `footer` dla stopki, i tak dalej. Oczywiście, używanie dla tych elementów znaczników `div` nie jest błędem, ale pisząc w HTML5 znacznie lepiej jest używać semantycznych tagów, by – między innymi – nasza strona była lepiej pozyjonowana przez wyszukiwarki.

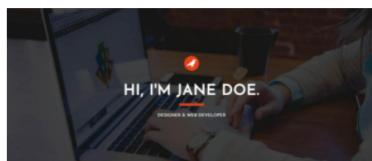
### Przygotowanie do projektu

Znasz już podstawy HTML i CSS, czas więc na nieco ambitniejsze zadanie. Od tej pory wszystkie niezbędne zagadnienia będziemy wyjaśniać sobie na realnych przykładach, których dostarczy nam nasz nowy projekt.

Na samym końcu tego modułu znajdziesz zadanie zatytułowane *Wyslij projekt do Mentora*. Przejdz do niego teraz i kliknij znajdujący się tam przycisk, aby otworzyć edytor. Następnie wróć do submodułu, który czytasz w tej chwili, by zacząć prace nad projektem. Najlepiej, aby submoduł i edytor z projektem były otwarte w dwóch zakładkach w przeglądarce, tak, by można się było między nimi łatwo przełączać.

Edytor odpalony? To zaczynamy!

Jako ostatnie zadanie w tym module stworzysz kompletną stronę na podstawie przygotowanego przez nas szablonu:



Struktura CSS – reguła

Box-model w CSS

Składowe pudełek

Wartości "po zegarze"

Najczęściej używane style

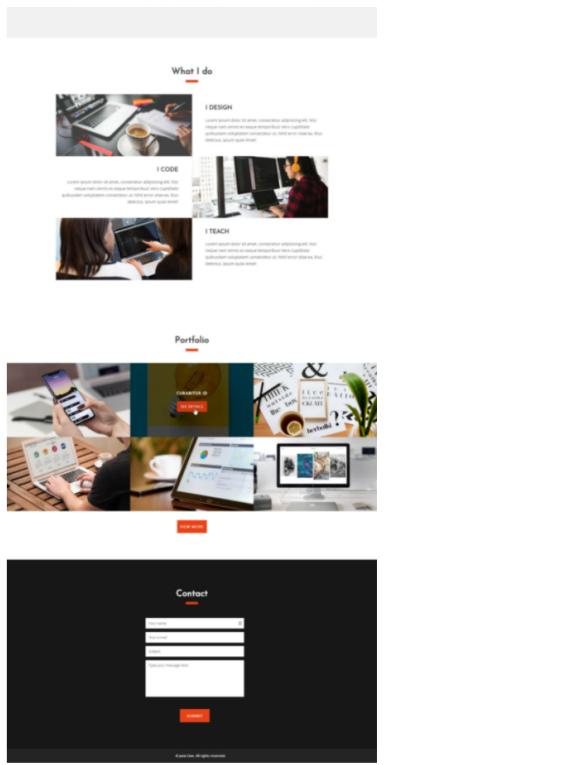
Podłączenie CSS do HTML i przeciążanie

Podsumowanie

Zadanie: lifting strony

Powrót do dashboardu

Napisz do nas



Dzięki temu od razu przećwiczysz w praktyce nowe umiejętności oraz nauczysz się myśleć "projektowo" – analizować wymagania, planować pracę oraz łączyć wiele małych elementów w jedną całość, tak, by harmonijnie ze sobą współgrały. Te umiejętności przydadzą Ci się w dalszej części kursu, kiedy będziesz tworzyć bardziej skomplikowane projekty.

Szablon jest tylko przykładową realizacją gotowej strony. Aby praca była dla Ciebie przyjemniejsza i bardziej inspirująca, otrzymasz dowolność wyboru tematu strony, treści, kolorów i zdjęć.

Nad naszym projektem będziemy pracować dwutorowo. Będziemy tworzyć każdą sekcję po kolej – najpierw umieścimy jej strukturę w pliku HTML, a następnie nadamy odpowiednie style. W ten sposób szybko zobaczysz efekty swojej pracy i zrozumiesz zależność między jednym językiem a drugim.

## SCSS, czyli CSS profesjonalnie

Zanim jednak napiszemy pierwszą linię kodu, poznamy narzędzie ogromnie ułatwiające pracę z CSS.

Jak możesz się domyślić, stylowanie większych stron w czystym CSS może być żmudne i czasochłonne. Wyobraźmy sobie, że w celu zapewnienia spójności wizualnej stosujemy w wielu miejscach ten sam kolor, a później zdecydujemy się go zmienić. Musimy odnaleźć wszystkie miejsca, gdzie ten kolor występuje, a następnie go podmienić.

Dużo prościej byłoby zdefiniować sobie zmienną, np. `color-green`, i wstawić ją jako wartość do każdej reguły, w której używamy tego koloru. Modyfikacja koloru odbywałaby się wtedy tylko w jednym miejscu.

Ponadto, utrzymanie rosnącej ilości kodu CSS przy coraz większym projekcie bywa trudne. Łatwiej byłoby podzielić go na kilka plików, każdy odpowiedzialny np. za jeden komponent tak, by połączyć go w całość. Możemy teoretycznie dodać kilka plików CSS do jednego dokumentu HTML, ma to jednak negatywny wpływ na prędkość ładowania strony.

Odpowiedzią na takie problemy (i wiele innych) są **preprocesory CSS**.

Istnieje kilka preprocesorów – wybór konkretnego z nich zależy zwykle od preferencji developera lub standardów używanych w danej firmie. W bootcampie będziemy używać preprocesora Sass i jego składni SCSS. Rzeczą jasna wykorzystamy tylko niewielką część możliwości, którą oferuje nam Sass – o wiele więcej opoju poznasz w toku dalszej nauki lub w czasie pracy jako developer. Ogrom informacji znajdziesz na [oficjalnej stronie Sass](#).

Aby pisać w SCSS w edytorze Kodilla, kliknij *Plik > Nowy plik* i zapisz go jako `style.scss`. Od tej pory będziesz pisać style **tylko w pliku .scss**. Skompiluje się on automatycznie do pliku `.css`, więc ręczna modyfikacja samego `.css` nie będzie potrzebna (a wręcz może zaszkodzić).

## Analiza projektu graficznego

Plik `style.scss` gotowy? Znakomicie. Idziemy dalej!

Przed przystąpieniem do pracy nad jakimkolwiek projektem należy zastanowić się, z jakich części będzie składać się nasza docelowa strona i jakie sekcje powinniśmy w niej przewidzieć. Nawet jeżeli kodujesz stronę bez gotowego projektu od grafika, dobrą praktyką jest rozrysowanie sobie (nawet na papierze) szkieletu strony (ang. *wireframe, mockup*), aby zobrazować sobie, jakie elementy trzeba będzie przewidzieć w kodzie.

Zerknij jeszcze raz na projekt naszej strony: [link](#).

Jak widzisz, składa się on z siedmiu unikalnych sekcji:

1. nagłówka, zwany często *splash* lub *splash screen*,
2. sekcja powitalna,
3. sekcja – nagłówek z tekstem,
4. sekcja z naprzemiennie występującymi obrazkami i tekstem,
5. sekcja z galerią,
6. sekcja z formularzem kontaktowym,
7. stopka.

Główna struktura strony sama rzuca się w oczy. Sekcja z galerią ma 100% szerokości, a pozostałe mają stałą szerokość, zachowując symetrię rozłożenia elementów.

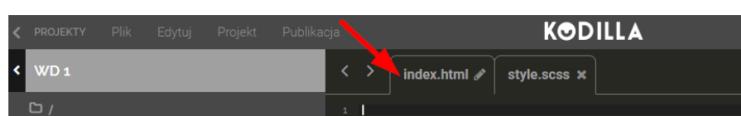
Na poniżej ilustracji zaznaczyliśmy kolejne komponenty naszej strony. Na czarno zaznaczyliśmy podział na sekcje, a żółtym kolorem odpowiednie containery, których użyjemy w celu zachowania symetrii rozłożenia elementów.

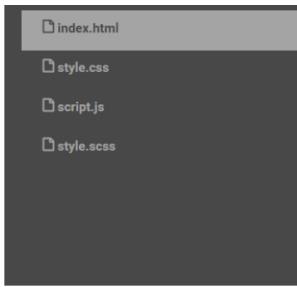


Strukturę każdej sekcji będziemy omawiać szczegółowo przy okazji pisania jej kodu HTML.

## Budujemy strukturę strony

Zaczniemy od zbudowania szkieletu struktury naszej strony. Przejdz do swojego projektu w edytorze i upewnij się, że jest w nim otwarty plik `index.html`:





Jeżeli tak nie jest, kliknij nazwę tego pliku w lewej kolumnie edytora.

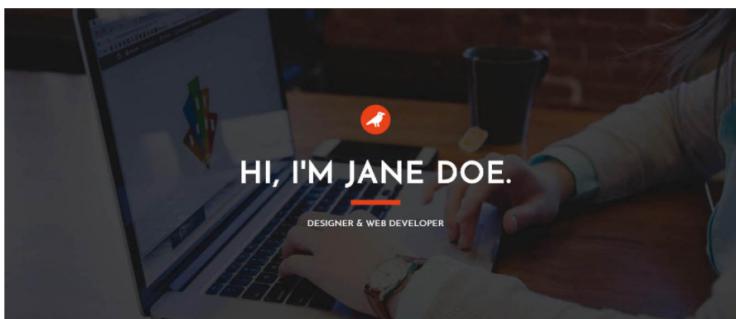
W edytorze, w pliku `index.html`, stwórz dobrze Ci już znaną **strukturę podstawową**, nie zapominając o podpięciu pliku ze stylami:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Project 1</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
  </body>
</html>
```

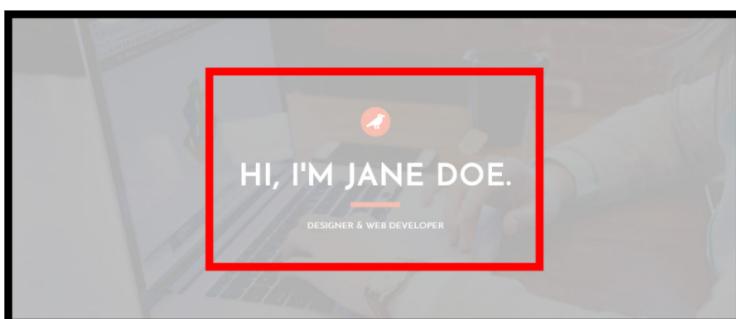
Pamiętaj o wcięciach w kodzie!

## Dodajemy sekcję z nagłówkiem

No dobrze, mamy już szkielet naszej strony, ale na ekranie podglądu nie pokazuje się jeszcze nic. Zaraz to zmienimy! Dodajmy najpierw sekcję z tytułem i podtytułem naszej strony.



Na samym początku przyjrzyjmy się dokładnie budowie tej sekcji i zobaczymy, z jakich elementów się składa, oraz na co będziemy musieli zwrócić uwagę w trakcie kodowania:



1. Cała sekcja (czarne 'pudelko') ma zajmować zawsze 100% wysokości ekranu.
2. Chcemy, aby cała treść była zawsze wycentrowana w pionie i w poziomie – czyli znajdowała się dokładnie na środku sekcji (czerwone "pudelko").
3. Treść powinna składać się z logo (w formie ikony na kolorowym tle), tytułu i podtytułu.

Czy potrafisz wyobrazić sobie, w jaki sposób dębu zagnieżdżanie tagów w pierwszej sekcji naszej strony? Jeżeli tak, to fantastycznie! Napiszemy teraz HTML tej sekcji, a poszczególnym tagom nadamy klasy, aby łatwo było się można do nich odnieść w pliku `.css`.

### O nazywaniu klas

Nazwy klas, których będziemy używać w opisie, są przykładowe. Śmiało możesz zastąpić je własnymi, pamiętając o kilku prostych zasadach:

- w nazwach klas używamy języka angielskiego,
- stosujemy tylko małe litery,
- nazwa klasy powinna być opisowa, czyli zwięźle wskazywać, czym jest dany element.

Te same zasady stosują się także do innych atrybutów, np. identyfikatorów (id).

W poniższych poleceniach nic nie powinno Cię zaskoczyć: bardzo podobne elementy tworzyliśmy w poprzednich zadaniach.

1. W pliku `index.html` w edytorze, wewnątrz tagu `<body>`, dodaj znacznik `<header>` z klasą `splash`. Nie zapomnij o tagu zamykającym i o właściwych wcięciach w kodzie! To będzie nasze główne, czarne "pudełko" – ten tag obejmie wszystkie pozostałe elementy w tej sekcji.

2. Wewnątrz tagu `<header>` dodaj tag `<div>` z klasą `splash-inner`. To będzie czerwone "pudełko" – będzie dzieckiem czarnego "pudełka" i rodzicem dla kolejnych elementów.

3. Wewnątrz tego diva stwórz nagłówek poziomu pierwszego (`<h1>`) z klasą `page-title`. Między tagiem otwierającym a zamykającym umieść swoje imię i nazwisko lub tekst powitalny.

4. Pod tagiem `<h1>` dodaj nagłówek poziomu drugiego z klasą `page-subtitle`. Między tagiem zamykającym a otwierającym wpisz dowolną treść.

Zapisz plik `.html` w edytorze i odśwież podgląd. Efekt powinien być podobny do poniższego:



Zwrót przy okazji uwagę na ciekawą rzeczą: czy coś się zmieni w podglądzie, gdy treść któregoś tagu zapiszesz w sposób pokazany na poniższym obrazku (np. wstawiając między imieniem a nazwiskiem kilka pustych linii za pomocą klawisza Enter)?

```
<h1 class="page-title">Hi, I am Jane  
  
Doe.</h1>
```

Otoż nie: przeglądarka nie zinterpretuje pustych linii w kodzie jako złamania tekstu. Aby złamać linię tekstu na stronie, potrzebny jest samozamykający tag `<br>` (ang. *break*).

### HTML5 a div

Wspomnieliśmy wyżej, że standard HTML5 wprowadził wiele nowych znaczników, które mają odzwierciedlać funkcję danego elementu na stronie. Co jednak w sytuacji, gdy nie jesteśmy w stanie dopasować żadnego z tych znaczników do elementu, który kodujemy? W takim przypadku śmiało możemy użyć elementów generycznych (czyli takich bez znaczenia semantycznego). Jednym z tego typu elementów jest właśnie `<div>`.

Zatem, podsumowując: tam, gdzie element ma wyraźną rolę w strukturze strony (np. nagłówek, stopka, nawigacja), używaj odpowiednich tagów HTML5. Jeżeli element nie ma takiego znaczenia, np. służy tylko jako kontener na inne elementy, możesz wykorzystywać znaczniki generyczne.

Podgląd wyświetla już treść obu nagłówków, ale zaraz zaraz, czy tu czegoś nie brakuje? Nie umieściliśmy do tej pory w HTML nic, co przypominałoby logo z designu. Już to poprawiamy! Możesz zapytać w tym momencie: "No dobrze, ale skąd wziąć taką grafikę"?

Jeszcze całkiem niedawno web developerzy (lub designerzy) musieli mozolnie wycinać wszystkie ikony w programie graficznym, a następnie wstawiać je na stronę w postaci obrazków. Dzisiaj sprawy mają się znacznie prościej dzięki dostępnym w internecie, gotowym do użycia zestawom ikon. W naszym projekcie wykorzystamy zestaw [Font Awesome](#) – jest on bardzo prosty w instalacji, a ikonki przyjemne w stylowaniu. Aby dodać Font Awesome do swojego projektu, wklej poniższy kod:

```
<link rel="stylesheet"  
      href="https://use.fontawesome.com/releases/v5.5.0/css/all.css">
```

w pliku HTML, tuż przed linkiem do pliku stylów (wewnątrz tagu `<head>`).

Ikonę z Font Awesome to nic innego, jak tagi `<i>` z odpowiednią klasą. Aby użyć ikony z tego zestawu, wystarczy wejść na stronę [Font Awesome: Icons](#), kliknąć wybraną ikonkę, skopiować w całości jej tag `<i>`:



i wstawić go do swojego kodu HTML, tuż nad tagiem `<h1>`:

```
<header class="splash">  
  <div class="splash-inner">  
    <i class="fas fa-crow"></i>  
    <h1 class="page-title">Hi, I'm Jane Doe.</h1>  
    <h2 class="page-subtitle">Designer & Web Developer</h2>  
  </div>  
</header>
```

Jeżeli wszystko zostało wykonane poprawnie, wybrana ikona powinna pojawić się nad tytułem w podględzie:



#### Wiele klas w jednym elemencie

Dotychczas wszystkie nasze elementy HTML miały jedną klasę. Kiedy jednak przyjrzyisz się tagowi ikony, np. `<i class="fas fa-crow"></i>`, zauważysz, że posiada on dwie klasy: `fas` i `fa-crow`. Do elementu HTML możesz dodać dowolną liczbę klas, oddzielając je spacją.

Mamy już całą strukturę sekcji Splash! Czas teraz na nadanie pierwszych stylów naszej stronie.

Jeśli masz wątpliwości do powyższego materialu, to - zanim zatwierdzisz - zapytaj na czacie :)

✓ Zapoznate(a)m się!



Po przejściu submodułu "Wstęp do CSS" zapewne bez problemu potrafisz już precyzyjnie wskazać w pliku stylów element o danej klasie lub tagu. Ta wiedza przyda nam się już teraz. Nie martw się, jeżeli ilość informacji wydaje się przytaczająca – wraz z nabywaniem praktyki, style zaczynasz pisać wręcz odruchowo.

Pamiętaj, że cały czas pracujemy na pliku **style.scss**.

Na samym początku zrobimy dwie rzeczy:

- sprawimy, że do wymiarów elementów HTML doliczane będą ich dopełnienia (**padding**) oraz obramowanie (**border**), co ułatwi nam kontrolowanie ich rozmiarów,
- usuniemy domyślne margines nadawany elementowi **<body>** przez przeglądarkę.

Te dwa zabiegi wykonywane są właściwie w każdym projekcie, więc dobrze zapamiętać, jakie deklaracje do nich służą.

#### Domyślne style elementów

Wiele elementów HTML (np. **<body>**, **<p>**, **<a>** itd.) posiada domyślne style, które często musimy nadpisać, aby strona wyglądała tak, jak tego chcemy. W podględzie możesz na przykład zobaczyć, że między tytułem a podtytułem strony jest odstęp – dzieje się tak dlatego, że nagłówki h1-h6 również otrzymują domyślny **margin**.

Ponadto, wdrożymy się do kolejnej dobrej praktyki, czyli dzielenia pliku stylów na uporządkowane, logiczne sekcje. Chcemy, aby style globalne, czyli takie, które mają być stosowane do wielu elementów w całym projekcie, znajdowały się na początku pliku **.scss**, oraz żeby style dla poszczególnych sekcji były od siebie wizualnie odseparowane, tak, aby łatwo można było je zlokalizować w pliku.

Tak więc, w pierwszej linii swojego pliku **.scss** dodaj następującą linię:

```
/* Global */
```

Efekt będzie taki, jak na obrazku poniżej:



```
1  /* Global */
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
```

Jak zauważysz, ten tekst przybierze w edytorze kolor szary. Oznacza to, że nie będzie on interpretowany przez przeglądarkę – będzie to po prostu komentarz do kodu CSS, przeznaczony dla developera.

#### Komentarze w kodzie CSS/SCSS

Komentarze w kodzie są używane przez developerów na przykład wtedy, gdy chcą oni wyjaśnić działanie jakiegoś fragmentu kodu innym programistom pracującym nad tym samym plikiem, lub po prostu, aby podzielić kod na logiczne części. Komentarze są też użyteczne, gdy chcemy czasowo wyłączyć fragment kodu, np. po to, aby wytypić błąd. W slangu developerów nazywa się to "zakomentowaniem kodu" i w CSS/SCSS działa w następujący sposób:

```
p {
  font-size: 14px;
  /* color: red; */
}
```

Właściwość **color** tego selektora jest **zakomentowana**, co oznacza, że nie będzie ona brana pod uwagę podczas wyświetlania strony. Usunięcie znaków komentarza (**/\* \*/**) i tym samym przywrócenie kodu do działania nazywamy **odkomentowaniem**.

Aby sprawić, by do rozmiaru wszystkich elementów HTML wliczał się ich padding i border, do swojego pliku **.scss** dodaj następującą deklarację:

```
*, *::before, *::after {
  box-sizing: border-box;
```

}

Asterisk (gwiazdka) jest selektorem uniwersalnym – w ten sposób w stylach odnosimy się do wszystkich elementów HTML jednocześnie. O `::before` i `::after` będziemy jeszcze mówić, więc na razie nie przejmuj się, jeżeli nie wiesz, do czego służą (na razie możesz zapamiętać, że noszą one nazwę **pseudoelementów**). Efekt w edytorze:

```
1  /* Global */
2 *
3 * *, *::before, *::after {
4 *   box-sizing: border-box;
5 }
6
7
8
9
10
11
```

### Ile dwukropków?

W poprzednich standardach CSS (1 i 2) pseudoelementy zapisywano się z pojedynczym dwukropkiem: `:before` i `:after`. W standardzie CSS3, którego tu używamy, normą jest zapis z podwójnym dwukropkiem: `::before` i `::after`.

Kolejna reguła usunie margines (`margin`) strony, który jest nadawany przez przeglądarkę tagowi `<body>`. Tę regułę już znasz: użyliśmy jej w jednym z poprzednich ćwiczeń. Stwórz selektor dla `body` i nadaj mu `margin: 0;`. Zapisz plik i odśwież podgląd, by zobaczyć zmiany.

## Dodajemy globalne style

Teraz dodamy do projektu jeszcze kilka stylów globalnych.

1. Po pierwsze, jak pamiętasz z analizy projektu graficznego, powtarzającym się elementem w naszym designie jest `container`, który ogranicza szerokość treści w sekcji. Powtarzające się elementy o tych samych właściwościach stylujemy raz, aby uniknąć nadmiarowego kodu i przeciążania pliku ze stylami.

W pliku `.scss` stwórz zatem kolejny selektor – `.container` – i nadaj mu właściwości `max-width: 1140px;`, `margin: 0 auto;`, `padding: 0 20px;`. `max-width` oznacza, że szerokość tego elementu będzie dostosowywać się do szerokości ekranu, ale nigdy nie przekroczy podanej wartości (w tym przypadku 1140px). Właściwość `margin: 0 auto`, jak pamiętasz, centruje nam element poziomo, a `padding: 0 20px;` nadaje odstęp od prawej i lewej krawędzi ekranu.

```
.container {
  max-width: 1140px;
  margin: 0 auto;
  padding: 0 20px;
}
```

2. Po drugie, mówiliśmy również, że tekst w całym projekcie ma być wyśrodkowany (z wyjątkiem treści w jednej z sekcji, dla której dodamy w tym celu osobne style). Dlatego też do selektora `body` dodaj właściwość `text-align: center;`.

3. Oprócz tego zdefiniujmy w selektorze `body` kolor jego tła (`background`) na `#fff`, kolor czcionki (`color`) na `#3d3d3d`, jej rozmiar (`font-size`) np. na `16px`, i interlinię (`line-height`) np. na `1.7` (bez jednostki).

```
body {
  margin: 0;
  text-align: center;
  background: #fff;
  color: #3d3d3d;
  font-size: 16px;
  line-height: 1.7;
}
```

4. Na samym końcu zwróć uwagę na to, że w projekcie każda sekcja ma taki sam górny i dolny `padding`. To również jest styl globalny! Dodaj deklarację, która sprawi, że wszystkie elementy typu `section` będą miały górny i dolny `padding` równy 108px.

## Wybieramy paletę kolorystyczną

Jak wspomnieliśmy, przygotowany przez nas szablon, nad którym pracujemy, jest tylko przykładem. Możesz swobodnie zdefiniować jego wygląd według własnego gustu.

Pamiętaj, że aby Twoja strona dobrze wyglądała, warto dobrać do niej odpowiednie kolory. Na początku wystarczy, że użyjesz "bezpiecznych", pasujących do siebie w różnych konfiguracjach, kolorów przedstawionych na stronie [Flat UI Colors](#).

Wybierz ze strony [Flat UI Colors](#) barwę, która będzie kolorem przewodnim Twojej strony. Wystarczy, że klikniesz wybrany kolor, aby skopiować jego kod.

### Kody kolorów w CSS

Istnieje kilka możliwości definiowania kolorów w CSS. W bootcampie będziemy używać tzw. wartości heksadecymalnych (ang. *hex*) np. `#ff5e57`. Jeżeli chcesz dowiedzieć się więcej, zajrzyj [tutaj](#).

## Tworzymy pierwszą zmienną SCSS

Czas po raz pierwszy skorzystać z potęgi SCSS! Zdefiniujemy nasz wybrany kolor przewodni jako zmienną, aby łatwiej było się nim posługiwać w dalszych pracach. W tym celu na samej górze pliku `.scss` (nad napisanymi do tej pory stylami), wstaw:

```
$color-main: #ff5e57;
```

`$color-main` to nazwa naszej zmiennej (zmienne mogą mieć dowolne nazwy), a `#ff5e57` to jej wartość. Podmień tę wartość na kod wybranego przez siebie koloru.

Już za chwilę wykorzystamy naszą zmienną do pokolorowania kilku elementów w sekcji Splash.

## Stylujemy sekcję Splash

Mamy już wszystko, co potrzebne, by zacząć już na poważnie stylować projekt! Zerknij na design: naszym celem jest sprawienie, by:

- sekcja Splash zajmowała całą wysokość ekranu,
- miała w tle zdjęcie, które zawsze będzie wypełniać ją w całości,
- jej treść była zawsze wyśrodkowana w niej w pionie i w poziomie.

Zaczniemy od dodania do pliku stylów nowej linii z komentarzem:

```
/* Splash */
```

Pod nim umieścimy style przynależące do tej sekcji. W ten sposób będziemy "wydzielać" w pliku kolejne sekcje, które będziemy stylować.

### Ogólne style sekcji Splash

Na początku stwórz w pliku `.scss` selektor dla sekcji Splash. Czy pamiętasz, jak to się robi? Zerknij do kodu HTML i przypomnij sobie, jaką klasę nadaliśmy naszemu znacznikowi `<header>`. Wpisz nazwę tej klasy (poprzedzoną kropką) w nowej linii pliku CSS i otwórz nawias klamrowy:

```
.splash {  
}
```

Wewnętrz tego selektora dodaj właściwość `height` o wartości `100vh`. Tym sposobem sekcja zawsze będzie tak wysoka, jak okno przeglądarki. Następnie dodaj kolejną właściwość: `position: relative;`.

### Pozycje elementów HTML

Właściwość `position` może przyjmować cztery wartości: `static`, `relative`, `absolute` lub `fixed`. `position` determinuje, jak elementy będą pozycjonowane w stosunku do siebie nawzajem i/lub do okna przeglądarki. Zerknij do CSS Reference: Position po więcej wyjaśnień i interaktywne przykłady.

## Zdjęcie w tle

Dodajmy teraz zdjęcie w tle tej sekcji.

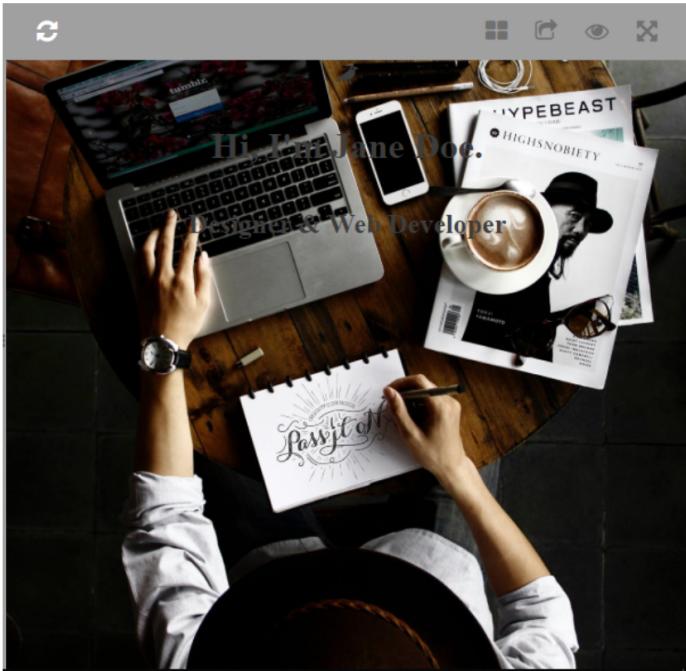
1. Na stronie [Pexels](#) wybierz zdjęcie, które Ci się podoba, kliknij prawym przyciskiem myszy i skopiuj jego adres.

2. Dodaj do selektora `.splash` właściwość `background-image` z wartością `url('link_do_zdjęcia');`. Podmień `link_do_zdjęcia` na skopiony przed chwilą odnośnik do wybranej fotografii.

```
.splash {  
    height: 100vh;  
    position: relative;  
    background-image: url('https://images.pexels.com/photos/57690/pexels-photo-57690.jpeg');  
}
```

5. Jeżeli wszystko zostało zrobione poprawnie, zdjęcie powinno pojawić się w podglądzie. Dodajmy do `.splash` jeszcze kilka właściwości, aby fotografia wyglądała lepiej: `background-position: center center;` oraz `background-size: cover;`. Odśwież podgląd i sprawdź, czy zdjęcie wyświetla się poprawnie. Dla bardziej dynamicznego efektu możesz dodać jeszcze jedną właściwość: `background-attachment: fixed;`. Kiedy dodasz więcej treści na stronę, sprawdź, jak obecność lub brak tej właściwości wpływa na zachowanie sekcji.

Efekt w edytorze:



### Pozycje tła

Obrazek w tle możemy wypożyczyć na kilka różnych sposobów. Przeczytaj [tutaj](#) o dostępnych opcjach i wypróbuj którąś z nich w swoim projekcie.

## Wycentrowanie elementów

Zajmiemy się teraz stylowaniem wnętrza sekcji Splash. Spójrz jeszcze raz na jej strukturę HTML i przypomnij sobie, jakie tagi stworzyliśmy.

Bezpośrednio wewnętrz znacznika z klasą `splash` umieściliśmy tag z klasą `splash-inner`. Nadamy mu teraz niezbędne style: dodaj w pliku CSS selektor dla klasy `splash-inner` i nadaj mu właściwości `position: absolute;`, `width: 100%`, `top: 50%`, `left: 50%` oraz `transform: translate(-50%, -50%);`.

```
.splash-inner {  
    position: absolute;  
    width: 100%;  
    top: 50%;  
    left: 50%;  
    transform: translate(-50%, -50%);  
}
```

Stworzyliśmy właśnie element wyśrodkowany w obu osiach! Warto zapamiętać tę przydatną sztuczkę: jeżeli chcemy, aby element-dziecko był dokładnie na środku elementu-rodzica, zarówno w pionie, jak i w poziomie, to rodzicowi nadajemy `position: relative;`, a dziecku `position: absolute; top: 50%; left: 50%;` oraz `transform: translate(-50%, -50%);`.

## Stylowanie logo

W designie naszej strony logo jest ikoną na tle kolorowego kółka. Sama ikonka jest już na swoim miejscu w kodzie HTML. Jak możemy dobrać się do niej za pomocą CSS?

Theoretycznie moglibyśmy dodać do selektora `<i>` jeszcze jedną klasę i użyć jej do ostylowania ikony, ale pokażemy Ci alternatywny sposób. Ikona znajduje się wewnątrz sekcji `.splash`, jeżeli więc użyjemy w CSS następującego selektora: `.splash i`, będzie on oznaczał: "wybierz wszystkie tagi `i` znajdujące się w elemencie o klasie `splash`".

Aby zatem ostylować logo:

1. Dodaj do swojego pliku ze stylami selektor `.splash i`.

```
.splash i {  
}
```

2. Ustaw jego rozmiar czcionki na `30px`, aby wyregulować wielkość ikony. Następnie dodaj właściwość `background`, a jako jej wartości użij zmiennej swojego przewodniego koloru:

```
.splash i {  
    font-size: 30px;  
    background: $color-main;  
}
```

3. Za pomocą właściwości `width` i `height` (ponownie z wartościami wyrażonymi w pikselach) dostosuj rozmiar całego logo. W naszym przykładzie użyjemy wartości `60px`, ale nie wahaj się wypróbować innych. Ustaw właściwość `line-height` na taką samą wartość, jak ma `height`, aby wycentrować ikonę w pionie.

```
.splash i {  
    font-size: 30px;  
    background: $color-main;  
    width: 60px;  
    height: 60px;  
    line-height: 60px;  
}
```

4. Za pomocą właściwości `color` zmień barwę ikonki (np. na kolor biały, czyli `#fff`). Jak widzisz, tego koloru używamy już kolejny raz, a czeka nas jeszcze trochę stylowania. Stwórzmy nową zmienną (np. `$color-light`) i używajmy jej, zamiast za każdym razem deklarować kolor biały jako `#fff`!

5. Na końcu, jeżeli chcesz, możesz zaokrąglić rogi swojego logo, używając właściwości `border-radius` i wartości wyrażonej w pikselach lub procentach. `border-radius: 50%` sprawi, że element stanie się kolem, pod warunkiem, że jest kwadratem (czyli pod warunkiem, że jego `width` i `height` są identyczne).

```
.splash i {  
    font-size: 30px;  
    background: $color-main;  
    width: 60px;  
    height: 60px;  
    line-height: 60px;  
    color: $color-light;  
    border-radius: 50%;  
}
```

Przykładowy efekt:



## Stylowanie tytułu

Sekcja Splash nabiera już charakteru. Zajmiemy się teraz tytułem i subtytułem, by lepiej je było widać na tle zdjęcia.

1. Zajrzyj do HTML, aby przypomnieć sobie, jaką klasę nadaliśmy tagowi `<h1>` w sekcji Splash. Użyj tej klasy, aby stworzyć selektor dla tytułu w pliku CSS. Nadaj mu `color`, `font-size` i `margin` (w designie użyliśmy białej czcionki rozmiaru 60px oraz marginesu równego `48px 0 0 0`, ale możesz zmodyfikować te parametry według własnego uznania). Ponadto dodaj mu właściwość `position: relative;`.

```
.page-title {  
    color: $color-light;  
    font-size: 60px;  
    margin: 48px 0 0 0;  
}
```

Możesz też zająć się tutaj, wyszukać inne ciekawe efekty tekstu i użyć ich w swoim kodzie.

2. Jak widzisz w projekcie, pod tytułem znajduje się element dekoracyjny w postaci poziomej linii. Stworzymy ją za pomocą wspomnianego już **pseudelementu `::after`**. Pod poprzednim selektorem dodaj kolejny: `.page-title::after`. Nadaj mu następujące właściwości:

```
position: absolute;  
content: '';  
bottom: -24px;  
left: 50%;  
width: 100px;  
height: 10px;  
transform: translateX(-50%);
```

Ponadto, dodaj mu `background` o wartości głównego koloru Twojej strony (użyj zmiennej).

Możesz zmodyfikować ten element według własnych preferencji. Nie zmieniaj ani nie usuwaj tylko właściwości `position` i `content`. Zwróć uwagę na właściwość `transform` – używaliśmy jej już wcześniej do wycentrowania elementu w obu osiach. Tutaj centrujemy nasz element tylko poziomo (tzn. w osi X), stąd wartość `translateX`.

## Słowo o pseudelementach

Pseudelementy `::before` i `::after` często są niesłusznie niedoceniane i pomijane przez developerów. Tymczasem mogą one odciążyć naszą strukturę HTML, gdyż zamiast tworzyć osobny div, możemy do jednego z już istniejących "podpisać" taki pseudelement jako np. dekorację. Tak też uczyniliśmy w naszym projekcie z poziomą kreską pod tytułem.

Zwróć uwagę, że rodzic naszego pseudelementu (czyli element `.page-title`) ma pozycję `relative`, a sam pseudelement – `absolute`. Oznacza to, że pseudelement będzie pozycjonowany względem swojego rodzica, a nie całej strony.

Spójrz na poniższy przykład i przeanalizuj jego HTML i CSS (przyciskając guziki na górnzej

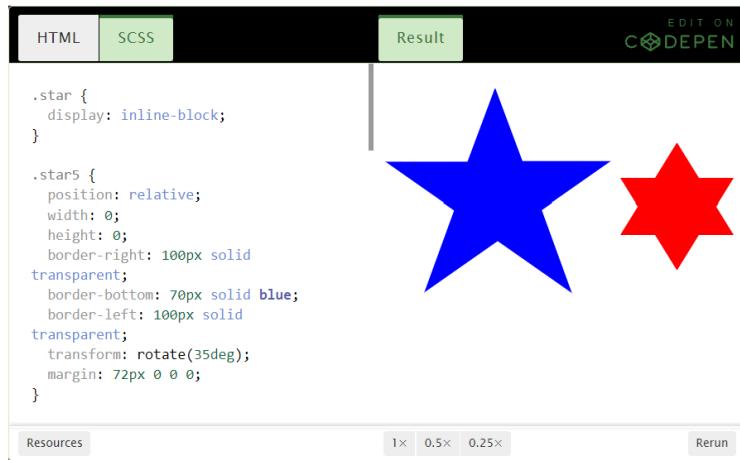
belce edytora CodePen). Zauważ, że jeżeli nie ma innych stylów, które mogłyby nadpisać lub zakłócać to zachowanie, pseudoelement domyślnie ustawi się w lewym górnym rogu swojego rodzica:



The screenshot shows the CodePen editor with the SCSS tab selected. The CSS code defines a parent class with a red background and dimensions, and a child pseudo-element ::after with a yellow background and dimensions, both set to absolute positioning. The result shows a red rectangle with a yellow star shape in its top-left corner.

```
.parent {  
  position: relative;  
  background: red;  
  width: 400px;  
  height: 200px;  
}  
  
.parent::after {  
  position: absolute;  
  content: '';  
  background: yellow;  
  width: 100px;  
  height: 100px;  
}
```

Z pomocą pseudoelementów możemy uzyskać mnóstwo ciekawych efektów, takich jak np. gwiazdki:



The screenshot shows the CodePen editor with the SCSS tab selected. The CSS code defines a star class with inline-block display and a stars class with various border properties and rotation. The result shows two star shapes: a large blue one and a smaller red one side-by-side.

```
.star {  
  display: inline-block;  
}  
  
.stars {  
  position: relative;  
  width: 0;  
  height: 0;  
  border-right: 100px solid transparent;  
  border-bottom: 70px solid blue;  
  border-left: 100px solid transparent;  
  transform: rotate(35deg);  
  margin: 72px 0 0 0;  
}
```

Mogimy też pokazywać pseudoelement po najechaniu myszą na rodzica:



The screenshot shows the CodePen editor with the SCSS tab selected. The CSS code defines a parent class with a red background and dimensions, and a child pseudo-element ::after with a yellow background and dimensions, both set to absolute positioning. A hover selector .parent:hover::after changes the display property to none. The result shows a red rectangle with a yellow star shape in its top-left corner that disappears when the mouse hovers over the red area.

```
.parent {  
  position: relative;  
  background: red;  
  width: 400px;  
  height: 200px;  
}  
  
.parent::after {  
  position: absolute;  
  content: '';  
  display: none;  
  background: yellow;  
  width: 100px;  
  height: 100px;  
}  
  
.parent:hover::after {
```

Innym bardzo interesującym przykładem jest... [czcionka stworzona za pomocą pseudoelementów!](#) Najeżdżając kursem na daną deklarację CSS, podświetlisz odpowiednią część liter.

Ale dość już o pseudoelementach. Wracamy do naszego projektu!

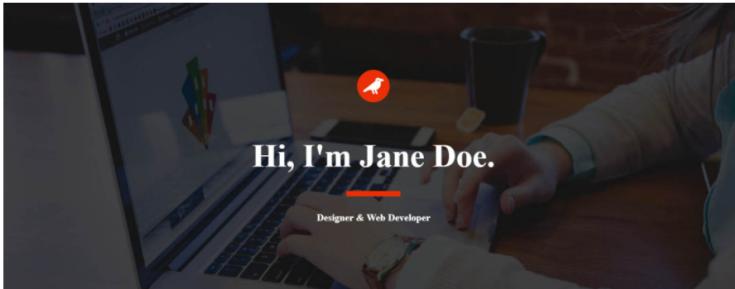
## Stylujemy podtytuł

Ostylowanie podtytułu strony nie powinno sprawić Ci najmniejszych problemów! Stwórz odpowiedni selektor w pliku stylów (przypomnij sobie, jaką klasę ma element `<h2>` w sekcji Splash) i zmień jego kolor, wielkość czcionki i margines. W projekcie zastosowaliśmy biały

kolor, rozmiar 18px i margines równy `48px 0 0 0`. Możesz również zastosować inne efekty tekstu, które znasz.

```
.page-subtitle {  
    color: $color-light;  
    font-size: 18px;  
    margin: 48px 0 0 0;  
}
```

Przykładowy efekt:



Jeżeli masz ochotę wzbogacić swój projekt, wyszukaj na stronie [CSS Reference: Typography](#) interesujące efekty tekstu i użyj ich w swoim kodzie.

## Dodajemy sekcję powitalną (Intro)

Kolejna sekcja w projekcie jest bardzo prosta — jest to krótka treść (cytat, motto) ozdobiona na górze i na dole poziomymi liniami.

Od tego momentu rozwiązania będą ukryte pod przyciskiem "Pokaż odpowiedź". Mocno zachęcamy, by korzystać z nich tylko wtedy, gdy samodzielne próby napisania danego fragmentu kodu okażą się nieskuteczne. Nie spiesz się — daj sobie czas na uważne przeczytanie polecenia i zastanowienia się nad nim.

1. Zaczniemy od struktury HTML. Pod tagiem zamykającym `</header>` w pliku `.html`, zachowując jedną pustą linię odstępu, dodaj tag `<section>` z klasą `intro`.

[Pokaż odpowiedź](#)

2. Wewnątrz tego taga stwórz `div` o klasie `container`. Następnie w środku tego diva dodaj akapit tekstu `<p>` z dowolną treścią (bez żadnej klasy). Jeżeli chcesz, by treść zajmowała więcej niż jedną linię, możesz złamać ją w wybranym miejscu tagiem `<br>`.

[Pokaż odpowiedź](#)

3. Przejdz teraz do pliku CSS i w nowej linii wydziel kolejną sekcję: `/* Intro */`. Następnie dodaj selektor dla sekcji, którą przed chwilą stworzyliśmy (jak pamiętasz, jej klasa nazywa się `intro`) i ustaw jej `background` na biały za pomocą zmiennej.

[Pokaż odpowiedź](#)

4. Teraz będziemy chcieli ostylować tag `<p>` znajdujący się wewnątrz sekcji `intro`. Czy pamiętasz, jak to się robi? Jeżeli nie, wróć do fragmentu, gdzie stylowaliśmy logo w sekcji Splash. Potrzebny Ci selektor mówiący: "wybierz wszystkie znaczniki `<p>` w elemencie o klasie `.intro`".

Utworzonemu selektorowi dodaj `width: 70%;`, `margin: 0 auto;`, `padding: 48px 0;` oraz ustaw wielkość czcionki (`font-size`) na np. 24px. Górną i dolną dekorację stworzymy za pomocą ramek (`border`). Potrzebujemy tylko górnej i dolnej ramki. Niech każda z nich będzie linią ciągłą (`solid`) o grubości `1px` i w wybranym przez Ciebie kolorze. Jeżeli nie pamiętasz, jak zrobić taką ramkę, zajrzyj do submodułu "Wstęp do CSS".

[Pokaż odpowiedź](#)

Przykładowy efekt:

---

I turn tea and cookies into crisp and elegant code.

I experiment and I make the world awesome.

---

---

## Sekcja z nagłówkiem i treścią (About)

---

W kolejnej sekcji umieścisz trochę więcej informacji o sobie. Jak możesz zobaczyć na designie, jej struktura jest prosta: ta sekcja będzie składała się z nagłówka i akapitu z tekstem.

1. Pod zamykającym tagiem poprzedniej sekcji wstaw kolejny znacznik `<section>` z klasą `about`. Wewnątrz dodaj `div` o klasie `container`.
2. W containerze umieść nagłówek drugiego stopnia z klasą `section-title` i dowolną treścią, a pod nagłówkiem – jeden lub dwa akapity dowolnego tekstu (np. kim jesteś, co robisz).

[Pokaż odpowiedź](#)

HTML gotowy, czas na style!

Kiedy przyjrzyzysz się wyglądowi tej i kolejnych sekcji, zauważysz, że pojawił się kolejny wspólny element – tytuły poszczególnych sekcji wyglądają tak samo. Jak już wspomnieliśmy, identyczne elementy staramy się stylować raz, zgodnie z zasadą DRY (Don't Repeat Yourself). Dlatego też style dla nagłówka umieść w CSS w sekcji `/* Global */`, w kolejnej linii.

1. Dodaj selektor dla tego elementu (przypominamy, że ma on klasę `section-title`) oraz nadaj mu `position: relative;`, zdefiniuj rozmiar czcionki (np. `36px`) oraz margines (np. `0 0 60px 0`).

[Pokaż odpowiedź](#)

2. Czy domyślasz się, po co zdefiniowaliśmy temu nagłówkowi `position: relative;`? Tak jest – aby dodać mu dekorację podobną do tego, jaką ma główny tytuł naszej strony! Stwórz selektor `.section-title::after` i dodaj mu takie same właściwości, jakie miał pseudoelement `::after` należący do głównego tytułu. Na końcu w selektorze `.section-title::after` zmień wartość `bottom` na `-12px`, a `width` na `50px`.

[Pokaż odpowiedź](#)

3. Przejdz teraz na koniec pliku CSS, wydziel nową sekcję `/* About */`, dodaj w niej selektor `.about` i zdefiniuj kolor tła (`background`) sekcji według własnego uznania (w projekcie użyty jest kolor `#f0f0f0`).

[Pokaż odpowiedź](#)

Efekt:



Hello world!

*Lorem ipsum dolor sit amet, consectetur adipisicing elit. Reiciendis, autem cupiditate ea necessitatibus.*

Eaque enim sit architecto atque possimus non asperiores rem nesciunt voluptate  
voluptatem, quia quibusdam magnam, sequi sapiente!

## Walidacja kodu HTML

Trzy pełne sekcje strony są gotowe. To dobry moment, aby upewnić się, czy nasz kod jest poprawny.

W tym celu użyjemy validatora dostępnego na stronie W3C. Wejdź na tę stronę: [link](#), przełącz validator na zakładkę "Validate by Direct Input", wklej w okienko całą zawartość swojego pliku HTML i kliknij "Check".

Jeżeli validator pokazał Ci w wynikach błędy (czerwony komunikat typu Error), przeczytaj ich treść i spróbuj poprawić pomyłki w kodzie.

Validator może wyświetlić Ci również ostrzeżenie (żółty komunikat typu Warning). Zwykle oznacza to, że kod nie trzyma się w stu procentach wszystkich rekomendacji, ale nie jest przez to nieprawidłowy. Na przykład validator może poinformować Cię, że w jednej z sekcji brakuje nagłówka ([heading](#)). Nie jest to błąd – dobrą praktyką jest użycie przynajmniej jednego nagłówka w każdej sekcji, ale jeżeli zdarzy się sekcja bez takowego (gdy nie jest tam potrzebny), nic się nie stanie.

Jeśli masz wątpliwości do powyższego materiału, to - zapim zatwierdzisz - zapytaj na czacie :)

✓ Zapoznałeś(a)m sie!

## 1.5. Kolejne sekcje strony



Mamy już stworzone trzy pierwsze sekcje strony i to była ta łatwiejsza część. :-) Jak widzisz na projekcie, kolejne elementy są już nieco bardziej złożone.

Nie przerąża nas to jednak – zakasujemy rekawy i do dzieła!

Dodamy teraz do naszego projektu niestandardowe czcionki oraz stworzymy wspólnie następną sekcję (naprzemienne tekst i obrazki). Dla pozostałych sekcji podamy tylko ogólne instrukcje, a Twoim zadaniem będzie dokończenie ich samodzielnie.

## Lifting czcionki

Bardzo ważnym elementem webdesignu (oprócz kolorów – używaj schematów kolorów!), o który dba każdy frontendowiec, jest czcionka. Aktualnie nasza strona używa najprostszej, domyślnej czcionki. Czas na trochę różnorodności!

W internecie istnieje wiele serwisów (zarówno darmowych, jak i płatnych) oferujących kolekcje czcionek do wykorzystania na stronach. My użyjemy jednej z największych i najpopularniejszych kolekcji tego typu: [Google Fonts](#).

W przykładzie wybierzymy dwa kroje czcionek: jeden dla nagłówków, drugi dla reszty tekstu. Oczywiście można użyć więcej fontów, ale zalecamy umiar – znacznie lepiej czyta się i ogląda oszczędne spośród wizualnie strony.

**1.** Wejdź na stronę [Google Fonts](#) i wybierz dwa kroje czcionek dla swojej strony. Możesz posilić się poniższymi zbiorami przykładowych zestawień dla inspiracji:

- [reliablepsd.com](#)
- [fontpair.co](#)
- [garettcreative.com](#)
- [typewolf.com](#)

W przykładzie zastosujemy kroje *Josefin Sans* i *Open Sans*.

**2.** Wyszukaj odpowiednią czcionkę. Upewnij się, że wybierasz opcję "Latin Extended" w polu "Language". W ten sposób będziesz mieć pewność, że znalezione fonty obsługują polskie znaki.

The screenshot shows the Google Fonts search interface. In the search bar, the word "open" is typed. Below the search bar, there is a button labeled "Latin Extended" which is highlighted with a blue border, indicating it is selected. The results page displays "3 of 999 families" for the "Open Sans" font by Steve Matteson. A preview window shows the text "Almost before we knew it, we had left the ground." in a large, bold, black font. The "Open Sans" font has 10 styles available.

**3.** Kliknij wybraną czcionkę i dodaj wagi (grubości), które Cię interesują (opcja "Select this style"). Po prawej stronie pojawi się lista z Twoimi wyborami.

The screenshot shows the "Selected families" interface. On the left, under the "Review" tab, there is a list of selected font styles: "Josefin Sans Regular 400" and "Open Sans Regular 400". Each entry includes a "Select this style" button and a "Remove" button. On the right, there is a "Selected families" sidebar with tabs for "Review" and "Embed".

**4.** Kiedy już zakończysz selekcję, kliknij zakładkę "Embed", by uzyskać kod do użycia fontów na swojej stronie (jego dokładna treść będzie zależała od wybranych przez Ciebie czcionek):

The screenshot shows the "Selected families" interface with the "Embed" tab selected. It provides instructions: "To embed a font, copy the code into the <head> of your html". Below this, a code editor shows the copied CSS code:  
<link href="https://fonts.googleapis.com/css2?family=Josefin+Sans&family=Open+Sans&display=swap" rel="stylesheet">

#### CSS rules to specify families

```
font-family: 'Josefin Sans', sans-serif;
font-family: 'Open Sans', sans-serif;
```

Skopiuj w całości ten kod i wklej go w swoim pliku HTML, w sekcji `<head>` (tuż przed linkiem do pliku `style.css`).

#### Wagi i style czcionek

Być może znasz już najprostsze deklaracje, których możemy użyć do pogrubienia (czyli zmiany wagi, *weight*) i pochylenia czcionki. Są to:

```
p {
  font-weight: bold;
  font-style: italic;
}
```

Deklaracje te wymuszają na przeglądarce "mechaniczne" pogrubienie czy pochylenie fonta. Niektórzy twórcy udostępniają więcej wag swoich czcionek i/lub ich alternatywny wygląd dla stylu italic. W przypadku Google Fonts możesz dodać te kroje, a następnie odnieść się do nich w stylach w następujący sposób:

```
p {
  font-weight: 100;
}
```

Czas na zdefiniowanie czcionek w pliku stylów, by wreszcie były widoczne w podglądzie. Wróć do okna ze swoimi czcionkami na stronie Google Fonts i odszukaj w nim tę sekcję:

#### CSS rules to specify families

```
font-family: 'Josefin Sans', sans-serif;
font-family: 'Open Sans', sans-serif;
```

Zobaczysz tu deklarację `font-family` dla wybranych fontów; ta deklaracja powie przeglądarce, jakich czcionek należy użyć. Moglibyśmy skopiować te deklaracje i wykorzystać je bezpośrednio, ale postąpimy bardziej profesjonalnie i zapiszemy nasze czcionki w zmiennych.

Na samej górze pliku `.scss` (tam, gdzie pozostałe zmienne) dodaj zmienne `$font-text` i `$font-header` z wybranymi czcionkami, wzorując się na poniższym przykładzie:

```
$color-main: #eb2f06;
$coldarck: #3d3d3d;
$font-text: 'Josefin Sans', sans-serif;
$font-header: 'Open Sans', sans-serif;
```

Następnie do selektora `body` dodaj deklarację `font-family: $font-text`. Podobnie postąp w przypadku czcionki dla nagłówków: selektorom `.page-title` i `.section-title` dodaj deklarację `font-family: $font-header`.

Od razu lepiej!

#### Fallback, czyli alternatywa

Zauważ, że w powyższych deklaracjach, oprócz nazw czcionek, które wybraliśmy, umieściliśmy po przecinku `sans-serif`. Jest to tak zwany fallback, czyli wskazówka dla przeglądarki, jakiego fontu powinna użyć, gdyby pierwsza zadeklarowana czcionka była z jakiegoś powodu niedostępna. W takim

w przypadku zostanie użyta `sans-serif`, czyli podstawowa czcionka bezszeryfowa używana przez przeglądarkę.

Dodawanie fallbacków to dobra praktyka, gdyż daje nam większą kontrolę nad wyglądem naszej strony. Google Fonts ułatwia nam sprawę i od razu sugeruje fallbacki, w przypadku używania czcionek z innych źródeł może być konieczne dodanie "zapasowych" fontów własnoręcznie.

## Sekcja z tekstem i obrazkami (Skills)

Powoli przechodzimy do bardziej złożonych konstrukcji! Jak widzisz w projekcie, sekcja składa się z trzech rzędów, a w każdym rzędzie znajduje się zdjęcie i tekst. Smaczku dodaje fakt, że teksty i zdjęcia ulożone są naprzemienne (raz po prawej, raz po lewej), a chcemy, żeby w HTML wszystkie rzędy miały taką samą strukturę.



### I DESIGN

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Nisi neque nam omnis ex eaque temporibus! Vero cupiditate quibusdam voluptatem consectetur ut. Nihil error vitae ea. Elus delectus. ipsum quia Amet!



### I CODE

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Nisi neque nam omnis ex eaque temporibus! Vero cupiditate quibusdam voluptatem consectetur ut. Nihil error vitae ea. Elus delectus. ipsum quia Amet!



### I TEACH

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Nisi neque nam omnis ex eaque temporibus! Vero cupiditate quibusdam voluptatem consectetur ut. Nihil error vitae ea. Elus delectus. ipsum quia Amet!

- Na początku stwórz szkielet tej sekcji, analogiczny do poprzednich (najpierw tag `<section>` z klasą `skills`, potem `div` z klasą `container`, a w nim nagłówek `<h2>` z odpowiednią klasą i dowolną treścią).

**Pokaż odpowiedź**

2. Mając powyższe, zastanówmy się, jak stworzyć taki układ tekstów i zdjęć, jak na projekcie. Rozrysujmy sobie najpierw podział na rzędy:



Każdy rząd będzie tagiem `<article>` (czarna ramka na powyższym schemacie), wewnątrz którego umieścimy dwa divy: jeden z fotografią (czerwona ramka), a drugi z tekstem (niebieska ramka).

Zanim jednak rozpoczniesz pisanie struktury, zastanów się, jakie treści chcesz zamieścić w tej sekcji. Może to być lista Twoich umiejętności, hobby – co tylko zechcesz. Znajdź trzy pasujące fotografie na stronie [Pexels](#) i skopiuj ich linki – użyjemy ich już niedługo.

3. Pod tagiem `<h2>` w sekcji `skills` dodaj znacznik `<article>` z klasą `skill-left`.

**Pokaż odpowiedź**

To będzie nasz pierwszy rząd.

4. Wewnątrz `<article>` wstaw `<div>` z klasą `skill-image`. W tym divie umieść jeden z przygotowanych obrazków.

**Pokaż odpowiedź**

5. Pod tym divem wstaw kolejny `<div>`, tym razem o klasie `skill-description`. Wewnątrz umieść nagłówek poziomu trzeciego z klasą `skill-title` i dowolną treścią. Będzie to tytuł pierwszego pudełka z tekstem. Pod nagłówkiem wstaw akapit tekstu z dowolną treścią.

**Pokaż odpowiedź**

6. Analogicznie stwórz dwa kolejne znaczniki `<article>` z taką samą strukturą, jeden pod drugim. Podmień w nich zdjęcia i treści. Drugiemu elementowi `<article>` (temu, który na projekcie ma zdjęcie po prawej stronie) podmień klasę `skill-left` na `skill-right`.

Podgląd nie wygląda jeszcze imponująco – obrazki są za duże i źle wypożyczonowane. Zaraz to poprawimy.

7. W pliku `.scss` dodaj "nagłówek" `/* Skills */`. Zajmiemy się najpierw stworzeniem stylów dla wszystkich tagów `<article>`. Jak jednak pamiętasz, część z nich ma klasę `skill-left`, a część `skill-right`. Moglibyśmy rozwiązać ten problem, sięgając po te elementy w stylach za pomocą tagu (`article { ... }`) lub dodać im jeszcze jedną wspólną klasę, ale pokażemy Ci alternatywne rozwiązanie. Wybierzemy w pliku stylów wszystkie elementy, których nazwa klasy zaczyna się od `skill-`. Jak to zrobić? Bardzo prosto:

```
[class^="skill-"] {  
}
```

Takiemu selektorowi nadaj `position: relative;`.

```
/* Skills */  
[class^="skill-"] {  
    position: relative;  
}
```

8. Zajmijmy się teraz stylowaniem obrazków. Przypomnij sobie, jaką klasę miały divy, w które owinięte były w HTML tagi `<img>`, i stwórz dla nich selektor w pliku stylów. Nadaj im `position: absolute;`, `width: 50%;`, `height: 100%` oraz `overflow: hidden;`.

**Pokaż odpowiedź**

9. Odśwież podgląd. Obrazki wyglądają już lepiej, ale fotografie są przycięte i źle wykadrowane. Aby to naprawić, dodaj selektor wybierający wszystkie tagi `<img>` w elementach o klasie `skill-image` i nadaj im `width: 100%;`, `height: 100%` oraz `object-fit: cover;`. W ten sposób dopasowaliśmy obrazki do kontenerów.

**Pokaż odpowiedź**

#### Object-fit

Właściwość `object-fit` jest dość nowym dodatkiem do CSS (tak, ten język ciągle ewoluje!). Pozwala nam na kontrolować, w jaki sposób obrazki dopasowują się do swoich kontenerów. Przez długi czas nie było na to prostego sposobu i developerzy imali się różnych sztuczek, na co odpowiedzią jest właśnie `object-fit`. Na temat samej właściwości przeczytasz [tutaj](#), mniej też na uwadze, że ta właściwość nie jest wspierana przez przeglądarki starsze niż Microsoft Edge (czyli np. Internet Explorer 11 i starsze).

Ponownie odśwież podgląd w edytorze. Jak widzisz, ciągle jednak wszystkie obrazki są z tej samej strony i zasłaniają tekst.

10. Rozwiązaniem pierwszego problemu jest stworzenie selektora mówiącego: "wybierz wszystkie elementy o klasie `skill-image`, znajdujące się wewnątrz elementu z klasą `skill-right`". Z tym poradzisz sobie bez problemu!

Do tego selektora dodaj właściwość `right: 0;`. Oznacza ona, że element-dziecko ustawi się tuż przy prawej krawędzi elementu-rodzica.

[Pokaż odpowiedź](#)

Prawie dobrze! Teraz jeszcze tylko paddingi i wyrównanie tekstu.

**11.** Stwórz selektor wybierający wszystkie elementy z klasą `.skill-description`, znajdujące się wewnątrz elementu z klasą `skill-right`. Dodaj mu `padding: 36px 55% 36px 0;` i za pomocą `text-align` wyrównaj tekst do prawej (`right`). Powtórz proces dla selektora wybierającego wszystkie elementy z klasą `.skill-description`, znajdujące się wewnątrz elementu z klasą `skill-left`: jemu z kolei nadaj `padding: 36px 0 36px 55%;` i wyrównaj tekst do lewej (`left`).

[Pokaż odpowiedź](#)

Pamiętaj, że gdy wartość liczbowa jakiejś właściwości wynosi `0`, nie używamy żadnej jednostki. Zapis np. `0px` jest błędny.

Voilà!

Teraz twoim zadaniem jest ostylowanie nagłówków i tekstów obok obrazków według własnych preferencji. Możesz inspirować się projektem albo użyć własnej wyobraźni.

#### Dla ambitnych

Spraw, by zdjęcia (`<img>`) w tej sekcji powiększały się po najechaniu na nie kursorem. Poczytaj tutaj na temat transformacji typu `scale`, stwórz selektor `.skill-image img:hover` i dodaj mu odpowiednią transformację. Na koniec samemu selektorowi `.skill-image img` dodaj właściwość `transition: all 1s;`, by animacja była płynna. Możesz również przeczytać tutaj więcej na temat właściwości `transition` i zmodyfikować ją według własnych upodobań.

#### Uwaga!

Zwróć uwagę na jedną istotną rzecz. Na grafice, w której proponowaliśmy podział elementów, wskazywaliśmy, że każdy `article` dzieli się na dwie równe części. Części, które są obok siebie. No cóż... to nie do końca prawda.

W rzeczywistości bowiem część z tekstem `.skill-description` tak naprawdę znajduje się... nad obrazkiem. Czyli nad elementem `skill-image`. To, że nie widzimy tekstu bezpośrednio nad obrazkiem jest wyłącznie zasługą użycia właściwości `padding`.

Nie wierzysz? Dodaj na moment do `.skill-description` nowy styl – `background: red`. Szybko zauważysz, że to, o czym przed chwilą powiedzieliśmy, to prawda. Czerwone tło na pewno zasłoni obrazek, który w rzeczywistości jest pod divem `.skill-description`. To duży problem w naszej sytuacji. Nic nie da Ci bowiem reguła `.skill-image img:hover`, jeśli obrazek jest pod divem z tekstem.

Najeżdżając na miejsce, w którym widzimy obrazek, nie najeżdzamy bowiem na obrazek, lecz na `.skill-description`. Aby osiągnąć pożądany efekt musisz więc w jakiś sposób zmusić CSS-a, aby z tych dwóch elementów, które są w jednym miejscu, to właśnie `.skill-image` było wyżej. Pomoże Ci w tym ten artykuł.

Kolejne dwie sekcje projektu, Galeria i Kontakt, przeznaczone są dla chętnych. Są to bardziej skomplikowane zadania, więc zabierz się za nie dopiero, gdy uporasz się z resztą modułu.

## Dla chętnych: sekcja Galeria

#### Portfolio





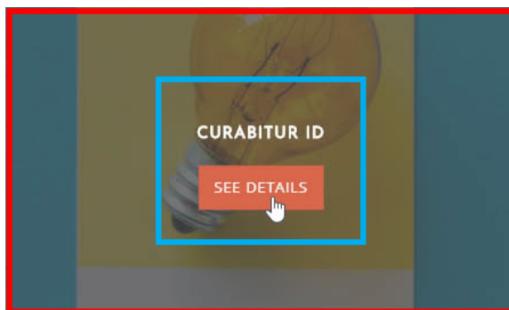
[VIEW MORE](#)

Sekcja Galeria różni się od pozostałych dość istotnym szczegółem – nie używamy w niej diva `container`, ponieważ chcemy, aby treść rozciągała się na całą szerokość okna przeglądarki.

Kilka wskazówek:

**1.** Na samym początku przyjrzyj się dobrze tej sekcji i przeanalizuj jej budowę:

- Pojedynczy "kafelek" w galerii to element to `div`, wewnątrz którego znajduje się obrazek i treść (tytuł i przycisk). Treść jest wycentrowana w swoim rodzicu w obu osiach. Przypomnij sobie, jak robiliśmy taką sztuczkę w sekcji Splash!



- Treść jest widoczna dopiero po najechaniu myszą na "kafelek" ( `hover` ).
- Dodatkowo, po najechaniu myszą pojawia się też półprzezroczysta warstwa (`overlay`). Więcej na jej temat znajdziesz we wskazówce nr 4.

**2.** Aby umieścić trzy elementy w linii, nadaj im `float: left;` oraz `width: 33.3333%;`. Nadaj im również jednakową stałą wysokość, np. 300px.

**Uwaga: elementy z właściwością `float`**

Elementy z właściwością `float` mogą czasami zachowywać się w sposób mało przewidywalny. Mówiąc wprost, to po części działa po prostu błędnie (ku zgromie wielu developerów), dlatego zaczęto prace nad innym rozwiązaniem (o nim w kolejnych modułach).

Problemem floata jest to, że elementy z taką właściwością czasami "uciekają" poza swój kontener. Aby temu przeciwdziałać, korzystamy z właściwości `clear`, a w zasadzie z pewnej gotowej reguły CSS z użyciem tej właściwości:

```
.row::before,  
.row::after {  
  clear: both;  
  content: "";  
  display: table;  
}
```

Ta reguła naprawia działanie floata. Nie będziemy tłumaczyć jej konstrukcji, wielu programistów nawet nie zna jej na pamięć i kopiuje ją z projektu do projektu. Służy ona do wspierania elementów-rodziców, których dzieci mają właściwość `float`.

Jeżeli będziesz używać właściwości `float` do wypożyczonowania elementów w galerii, skopiuj powyższy kod do swojego pliku `.scss` i nadaj klasę `row` na bezpośredniego rodzica tych elementów (np. owiń wszystkie "kafelki" w jednego diva-rodzica i jemu nadaj klasę `row`).

**3.** Jeżeli chcesz wypożycjonować jeden element względem innego, to elementowi-rodzicowi nadaj `position: relative;` a elementowi-dziecku `position: absolute;`, i przesuwaj ten drugi element o wybraną liczbę pikseli za pomocą właściwości `left`, `right`, `top` i `bottom`.

**4.** Aby stworzyć półprzezroczystą warstwę (ang. `overlay`) zakrywającą element (zerknij na design, jak wygląda element, na który najedzie się kursorem myszy), możesz użyć pseudoelementu `::before` lub `::after`. Ustaw mu `width` i `height` na `100%`, `background` na dowolny kolor, a `opacity` na wartość z przedziału między `0` a `1` ( `0` to całkowita przezroczystość, `1` to całkowita nieprzezroczystość), np. `0.5` lub `0.7`. Jeżeli nie pamiętasz, jak pokazywać pseudoelement po najechaniu na element-rodzica, wróć do rozdziału "Słowo o pseudoelementach".

5. Aby zmienić wygląd elementu po najechaniu na niego kursorem, użyj `:hover` w selektorze CSS, np. `.button:hover { }`.

6. Gdy będziesz pracować w wielowarstwowymi elementami o różnych wartościach `position`, może przydać Ci się znajomość właściwości `z-index`.

### Warstwy

W programowaniu zawsze istnieje wiele dróg rozwiązania jednego problemu. Użycie `z-index` jest jedną z możliwości, z której skorzystamy. W przyszłych modułach poznamy coś o wiele lepszego.

Dodatkowa wskazówka: pamiętaj, że `z-index` tworzy warstwy, więc nawet jeśli tego nie widać, tak naprawdę nie mamy tu dwóch kolumn obok siebie, tylko dwie warstwy, jedna na drugiej. Dlatego warstwę z obrazkiem należy wysunąć wyżej, właśnie za pomocą odpowiedniego użycia `z-index`.

## Dla chętnych: sekcja Kontakt

Zakodowanie sekcji Kontakt będzie od początku do końca Twoim zadaniem. Przed przystąpieniem do pracy przerób nasz [kurs samodzielny na temat formularzy](#). Następnie, opierając się na informacjach zdobytych w trakcie tego modułu, dodaj do swojego projektu prosty formularz i nadaj mu odpowiedni wygląd.

W projekcie jako koloru tła tej sekcji użyliśmy `#171717`, ale oczywiście możesz dobrać inną barwę.

## Dodajemy stopkę

To już ostatnia sekcja na naszej stronie! Jej stworzenie nie powinno sprawić Ci absolutnie żadnego problemu. Użyj taga `<footer>` z klasą `page-footer`, w którym zamieść `container` i znacznik akapitu z odpowiednią treścią.

W projekcie tło stopki ma kolor `#242424`.

## Zadanie: wyślij projekt do Mentora

Kiedy Twój projekt będzie gotowy do wysłania, kliknij *Wyślij do sprawdzenia*, aby przekazać go Mentorowi.

Podgląd zadania

Przejdz do projektu ✓

## 1.6. Podsumowanie



Udało Ci się ukończyć moduł wprowadzający w tajniki HTML i CSS – gratulacje! Mamy nadzieję, że praca nad projektem i samodzielne poszukiwanie niektórych odpowiedzi było dla Ciebie przyjemnością.

## Co już potrafisz?

Po przerobieniu tego modułu umiesz już:

- rozróżnić funkcje języków HTML i CSS w budowaniu stron,
- wymienić tagi HTML niezbędne do tego, by strona działała,

- napisać strukturę prostej witryny,
- nadać elementom odpowiednie style, używając jako selektorów klas i tagów,
- stosować złożone selektory CSS,
- zrozumieć zależność między elementem HTML a jego selektorem w CSS.

## Przydatne materiały

Poniżej zamieszczamy linki do materiałów, które przydadzą Ci się w dalszych częściach kursu i w trakcie samodzielnej nauki frontend developmentu.

### Dokumentacje

- [HTML Reference](#)
- [CSS Reference](#)

### Poradniki

- [Wprowadzenie do HTML na stronie MDN](#)
- [Wprowadzenie do CSS na stronie MDN](#)

### Validatory

- [Validator HTML](#)
- [Validator CSS](#)

### Palety kolorów

- [Flat UI Colors](#)
- [Paletton](#)
- [Colormind](#)
- [uiGradients](#)

### Darmowe zdjęcia

- [Pexels](#)
- [Unsplash](#)
- [Pixabay](#)
- [StockSnap](#)

### Inne

- [Can I Use...](#) - na tej stronie możesz sprawdzić, które przeglądarki wspierają poszczególne właściwości/elementy HTML i CSS

Jeśli masz wątpliwości do powyższego materialu, to - zanim zatwierdzisz - zapytaj na czacie :)

✓[Zapoznaj się!](#)