1. Struktura HTML i stylizacja CSS

Wyzwania:

- Zbudujesz pierwszą stronę, zwalidujesz jej poprawność i uruchomisz w swojej przeglądarce
- Podłączysz style i dodasz własny "look & feel"
- Poznasz box-model i umieścisz stronę w prostym layoucie!

Sugestia

Przed rozpoczęciem tego modułu (lub po teorii) warto przejść ten kurs Wprowadzenie do HTML, aby przećwiczyć w praktyce podstawowe zagadnienia, co pomoże przy wykonywaniu zadań.

1.2. Działamy z CSS!

1.1. Wprowadzenie

Front-endowiec… kto to?

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

Ci, którzy posiadają taką wiedzę i potrafią budować takie rozwiązania, mają nie tylko większą świadomość technologiczną, ale także **większe szanse na rynku pracy**. Specjalistów IT wciąż brakuje i nie zanosi się na zmianę tego stanu rzeczy.

Spośród wielu specjalistów IT dobrze zarabiającą grupą są programiści. Pracują oni nad różnymi rozwiązaniami. Część z nich tworzy rozwiązania webowe, a więc te, które można otworzyć w przeglądarce internetowej. Często taka osoba nazywana jest ogólnie *webdeveloperem*.

Jednak w ramach zawodu webdevelopera wymieniane są też specjalizacje (programista to potocznie *developer* lub *hacker*):

- **Front-end Developer** - osoba odpowiedzialna za tworzenie warstwy wizualnej strony/serwisu WWW (na podstawie projektu od grafika) oraz dodawanie pewnych interakcji lub elementów dynamicznych (np. po kliku jakiegoś elementu coś się zmienia na stronie). Tworzy tzw. szablony lub konstruuje pełne strony internetowe, ale mające głównie charakter prezentacyjny. Jednak bardziej zaawansowani front-endowcy potrafią tworzyć także podstawowe rozwiązania funkcjonalne (interakcja z komunikacją z bazą danych). Taka osoba powinna znać m.in. technologie: HTML5, CSS3, SASS, wybrany framework front-end, podstawy JavaScript/jQuery.
- **Back-end Developer** - osoba odpowiedzialna za logikę aplikacji wpływającą na jej zachowania (funkcjonalności) i za konstrukcję bazy danych, w której są przechowywane dane wyświetlane w aplikacji (we front-endzie). Odpowiada ona także za wydajność i bezpieczeństwo rozwiązania. Zestaw potrzebnych technologii może być różny, przykładowo: język back-end - PHP lub Python, baza danych - MySQL lub MongoDB, system dla serwera - Linux lub Windows Server.
- **JavaScript (JS) Developer** - często koduje "na przecięciu" obu powyższych specjalizacji, budując zaawansowane interakcje we front-endzie lub wspierając komunikację pomiędzy front-endem a bazą danych w back-endzie. Zaawansowany JS Developer używa jednego z frameworków (bibliotek), np.: Angular, React, Backbone lub Ember.
- **Full-stack Developer lub webdeveloper** - doświadczona osoba, która łączy w sobie kompetencje front-end, back-end i JavaScript developera, czyli potrafi stworzyć w całości kompletną aplikację.

W tym bootcampie kształcimy osoby zmierzające do uzyskania zawodu *Junior front-end developera*, który jest najlepszym wstępem do rozpoczęcia programowania rozwiązań dla internetu (dziedzina: *web development*).

Rozwiązania internetowe

Webdeveloperzy tworzą różne rozwiązania internetowe, które przyjmują pewne nazewnictwo, często umowne:

- Strony internetowe - mające głównie rolę "wizytówki". Ich cechą jest to, że mają w zasadzie wyłącznie rolę informacyjną. Praktycznie nie posiadają elementów

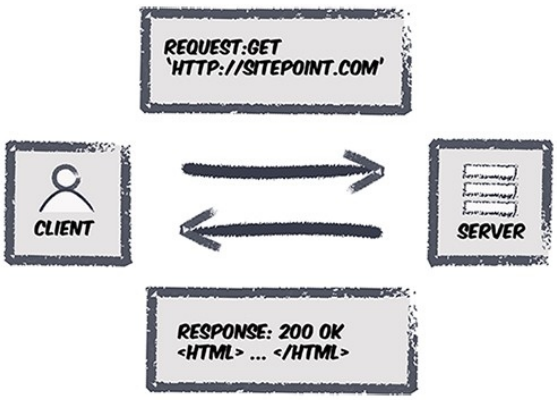
funkcjonalnych.

- Serwisy internetowe - bardziej rozbudowane rozwiązania np. portale społecznościowe, fora, serwisy informacyjne.
- Aplikacje internetowe - rozwiązania, które posiadają pewne funkcjonalności, np. narzędzia online takie jak zarządzanie listą zadań.
- Aplikacje mobilne - programy instalowane na telefonie posiadające zaawansowane funkcjonalności.

W tym bootcampie nauczysz się przygotowywać szablony dla stron internetowych, co jest podstawą pracy front-endowca. Do tego będziemy tworzyć proste aplikacje internetowe posiadające już pewne funkcjonalności. Efekty swojej pracy będzie można otworzyć w przeglądarce i użyć!

Jak działa strona internetowa

Strona internetowa składa się m.in. z plików o rozszerzeniu ".html", np: plik.html. Twój komputer to tzw. klient (ang. client). Kiedy wpiszesz na nim adres w przeglądarce i zatwierdzasz go, do serwera przez internet idzie żądanie (ang. request) otrzymania np. kodu strony internetowej. Jeśli żądanie jest poprawne to serwer odpowie (ang. response) i wyśle z powrotem kod HTML. Wtedy w Twojej przeglądarce pojawi się żądana strona internetowa.



Co to jest HTML

HTML (ang. *HyperText Markup Language*) jest to hipertekstowy język znaczników (*a* tag, *markup*). Dlaczego *hipertekstowy*? Bo można w nim używać linków, które kierują innych plików HTML, czyli można "skakać" pomiędzy stronami :)

HTML jest podstawowym językiem do budowy strony internetowej. Zawiera treść strony i **organizuje strukturę tej treści**. Niektórzy mylnie twierdzą, że za pomocą języka HTML definiujemy wygląd strony, ale tak nie jest. HTML dotyczy struktury informacji na stronie.

Poprzez **określanie struktury** należy rozumieć, że HTML określa typy treści, którą zawiera, jej kolejność i zagnieżdżenia. Jeśli chodzi o typy treści to są to m.in.:

- teksty, paragrafy, znaki końca linii
- linki
- nagłówki
- grafiki, zdjęcia
- kontenery treści (grupujące inne elementy HTML).

Dokument HTML

Kod HTML jest zapisywany w tekstowych plikach z rozszerzeniem html. Może być to np. plik *strona.html* (przed rozszerzeniem może być dowolna nazwa). Gdy otworzymy taki plik w przeglądarce internetowej, zawarty w nim kod HTML zostanie **zinterpretowany** i zostanie wyświetlona tylko treść bez kodu. Dalej dowiesz się co w takim pliku jest kodem, a co treścią.

Witryna internetowa to zbiór takich dokumentów HTML (plików), pomiędzy którymi możemy się poruszać, np. za pomocą linków w tych dokumentach.

Pliki HTML można edytować w zwykłym Notatniku na Twoim komputerze, ale także w innych specjalnych programach, które ułatwiają tę edycję. Mają one możliwość otwarcia całego projektu w formie drzewa plików i katalogów, dostarczają kolorowanie składni czy pilnują formatowania kodu. Jednym z najpopularniejszych jest Sublime Text 3 i jego będziemy używać. Możesz go zainstalować na próbę i spróbować edytować utworzony przez Ciebie plik *index.html* (dowolny tekst w środku).

Notatka

Docelowo pliki witryny internetowej umieszcza się na serwerach, czyli specjalnych komputerach, aby były one dostępne dla internautów pod wybraną domeną. Nauczymy Cię jak to robić w dalszej części kursu.

Tagi HTML i struktura

Głównym elementem kodu HTML jest **tag** (ang. *znacznik*). Jest to specjalne słowo kluczowe umieszczone wewnątrz ostrych nawiasów (znak mniejszości i większości). Budowa tagu wygląda tak:

```
<tag_name>content</tag_name>
```

Nazwa zawarta pomiędzy ostrymi nawiasami określa typ tagu. W uproszczeniu, tagi to kod HTML, a teksty między nimi to treść. Tagi okalają te teksty, co pozwala określić strukturę dokumentu HTML - co jest czym.

Jakie są cechy tagów?

- Tagi HTML najczęściej występują w parach np. i (tagi paragrafu).
- Pierwszy tag w parze to tag otwierający, drugi jest tagiem zamykającym.
- Tag zamykający jest zapisany podobnie jak otwierający, ale dodaje się znak `/` przed jego nazwą (slash, a nie backslash).
- Są też tagi niewymagające zamykającego taga (a nawet nie należy go wstawiać), takie jak
 (złamanie linii, pot. enter).

Struktura dokumentu HTML i przykładowe tagi

Każdy dokument HTML ma ściśle określoną strukturę podstawową. Struktura, czyli kod początkowy, który w zasadzie powtarza się w każdym pliku HTML. Dopiero w ramach tej struktury dodajemy swoją treść i inne tagi. Struktura wygląda tak (to co jest w <body> jest opcjonalne):

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>My first website</title>
  </head>
  <body>
    <h1>My first heading</h1>
    <p>My first paragraph and <a href="https://www.google.pl/">l
    <img src="image.jpg" alt="sand" title="my holiday">
  </body>
</html>
```

Poszczególne elementy (tagi) struktury to:

- Deklaracja `DOCTYPE`, która określa typ tego dokumentu jako HTML.
- Treść pomiędzy <html> i </html> to nasz dokument HTML.

- Treść pomiędzy <head> i </head> dostarcza informacji o stronie, które są ważne dla przeglądarki, **ale najczęściej niewidoczne dla internauty**.
- Treść w tagach <meta> (zawsze zagnieżdżone w <head></head>) zawiera konkretne informacje ważne dla przeglądarki. Aby poprawnie wyświetlały się polskie znaki za pomocą tagu meta dodajemy kodowanie UTF-8 - <meta charset="UTF-8">.
- 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 <script> i </script> to dołączenie skryptów do strony - póki co nie będziemy się tym zajmować.
- Treść pomiędzy <body> i </body> jest tym, **co widzisz potem w przeglądarce**. Tu wstawiasz teksty i zdjęcia dla swojej witryny WWW.... i opcjonalne elementy niżej:
- Treść pomiędzy <h1> i </h1> to nagłówek.
- Treść pomiędzy i utworzy paragraf tekstu.
- Treść pomiędzy i definiuje link, który ma prowadzić pod dany w atrybucie `href` adres.
- Znacznik <img src="file.jpg" alt=""> wstawia plik grafiki na stronę z danego źródła src (ang. *source*).

Zapamiętaj!

W sekcji `head` utrzymuj porządek - najpierw umieszczamy tag <meta>, następnie <title>, <link> i ostatnie <script>.

Ważne!

Zawsze zamykaj tagi!

Wprawdzie coś takiego jak poniżej:

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

zostanie wyświetlone przez przeglądarkę, ale może prowadzić do nieoczekiwanych efektów kiedy zaczniesz rozbudowywać swoją stronę. Dlatego powtórzymy: domykaj tagi:

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

Są także tagi samozamykające się, **których nie trzeba domykać**, np.:

- Tag
, który oznacza złamanie linii. Coś jak "enter".
- Tag <img>, który pozwala dodać zdjęcie na stronę.

- Tag <hr> dodaje linię aby w widoczny sposób rozdzielić kolejne sekcje w dokumencie HTML.

W zasadzie - jeżeli rozumiemy działanie samozamykających się tagów - to ich domykanie byłoby nielogiczne :)

Atrybuty tagów

Tagom (znacznikom) HTML możemy dodawać atrybuty, które określą dodatkowe właściwości tagu. Dodajemy je w następujący sposób:

```
<tag_name attribute1="valueX" attribute2="valueY">content</tag_name>
```

Atrybuty zawsze należy dodawać do tagu otwierającego. Można też dodać kilka atrybutów do jednego tagu, rozdzielając je spacją. Wartość atrybutu podaje się po znaku równości, zawierając ją w cudzysłowie.

Dla różnych tagów da się stosować różne atrybuty, które mogą przyjmować określoną wartość. Np. aby dodać obrazek do dokumentu HTML, należy dodać taki kod:

```
<img src="image.jpg" alt="sand" title="my holiday">
```

Po znaku `<` mamy nazwę taga, czyli `img` (ang. *image*), a następnie atrybut `src`, który wskazuje nazwę pliku grafiki w cudzysłowie. Tuż po `src` znajdują się atrybuty `title` oraz `alt`. `title` powoduje, że po najechaniu kursorem na stronie na tę grafikę, pokaże się napis zawierający wartość tego atrybutu. Natomiast `alt` pozwala nam określić tekst zastępczy dla obrazka, który ma się pojawić, gdy ten nie będzie mógł być wyświetlony - powinien on zwięźle opisywać treść grafiki.

Listę wszystkich tagów/atrybutów znajdziesz na stronie W3schools. To bezcenne źródło informacji o HTML i nie tylko. Jest jak encyklopedia, do której warto co jakiś czas zaglądać, **zamiast uczyć się wszystkiego na pamięć!**

Zagnieżdżanie tagów

Istotnym aspektem struktury dokumentu HTML jest to, że możemy **zagnieżdżać** tagi jeden w drugim:

```
<body>
    <h1>My first heading</h1>
    <p>My first <a href="http://google.pl">paragraph</a>.</p>
    <img src="image.jpg" alt="sand" title="my holiday">
</body>
```

Zauważ, że tag paragrafu <p> jest zagnieżdżony w elemencie <body>, a w nim dodatkowo jest zagnieżdżony link <a href="...">. Tworzy to tzw. strukturę drzewiastą. Tagi zagnieżdżone o poziom niżej warto w kodzie wcinać w prawo (klawiszem TAB), aby struktura dokumentu była bardziej czytelna. Wtedy szybciej widzimy co jest w czym zagnieżdżone. Pamiętaj, że wszystkie dzieci w danym tagu powinny mieć to samo wcięcie.

Stosuje się też nazewnictwo co do relacji tagów wg zagnieżdżenia. Na bazie przykładu z kodu powyżej wygląda to tak:

- child (dziecko) - <p> jest dzieckiem <body>,
- parent (rodzic) - <body> jest rodzicem <p>,
- sibling (rodzeństwo) - <h1> jest rodzeństwem <img> i <p>.

W polskich rozmowach często programiści łamią sobie język:

"Skasuj ten link (<a>)! Który? No tego childa (czyt. czajlda) paragrafu. Nie widzę... No ten zagnieżdżony 2 poziomy w dół, w body i potem w p!".

Nie brzmi to pięknie, ale często tak to wygląda w praktyce. Choć przykład jest dość prosty jak na tak długą dyskusję :)

Pamiętaj także, aby pilnować poprawnego zagnieżdżenia. Poniższy przykład jest nieprawidłowy:

```
<div>
    <p>Hello</p>
<p>coders</p>
</div>
```

Oba Tagi <p> powinny mieć takie samo wcięcie względem . Prawidłowe zagnieżdżenie w tym przypadku będzie wyglądało następująco:

```
<div>
    <p>Hello</p>
    <p>coders</p>
</div>
```

Ważne

Prawidłowe wcięcia są niesamowicie istotne. Gdy już będziesz mieć sporo linii kodu i trafi ci się jakiś błąd, to brak prawidłowego zagnieżdżenia sprawi, że będziesz mieć spore problemy ze znalezieniem go. Dotyczy to zarówno Ciebie, jak i osób, które będą Ci pomagać.

Aby łatwiej zrozumieć **jak poprawnie zagnieżdżać tagi HTML**, wyobraź sobie pudełka jedno w drugim albo matrioszkę. Jedna matrioszka musi znajdować się w drugiej w całości lub zawierać inną w całości. Nie jest możliwe, żeby matrioszka A była trochę w środku, a jednocześnie trochę na zewnątrz matrioszki B. To nielogiczne i łamiące prawa fizyki :)

Tak samo jest z tagami HTML. Dana matrioszka zaczyna się tu `<tag>` i kończy tu `</tag>`!

HTML w wersji 5

Znacznik `<!DOCTYPE html>` na początku dokumentu HTML oznacza, że stosujemy HTML5. Jest to najnowsza i aktualnie stosowana wersja tego języka.

W dużym skrócie: HTML5 dostarcza zestaw nowych tagów wymienionych tutaj. Ich główną rolą jest nadanie dokładniejszego znaczenia poszczególnym elementom w dokumencie HTML. Na przykład, gdy tworzymy pasek nawigacji na stronie, użyjemy tagu `<nav>` zamiast `<div>`, który jest ogólnym znacznikiem sekcji treści. Dzięki temu przeglądarka "wie", że dany obszar to nawigacja strony, a nie stopka bądź inny element.

Najczęściej stosowane znaczniki w HTML5 to: *nav, header, section, aside, footer, article, strong, em*. Te wymienione w zupełności wystarczą.

Zadanie: Pierwsza strona www

Pierwsze spotkanie z edytorem Kodilla

Po przeczytaniu instrukcji zadania kliknij przycisk niżej, wtedy zostanie otwarty edytor z kodem, w którym wykonasz poniższe polecenia.

Pamiętaj, że po każdej zmianie należy zapisać wszystkie pliki projektu (**Plik -> Zapisz wszystkie**, screen), a następnie odświeżyć podgląd (ikonka odświeżania w widoku strony, albo **Ctrl+R**, screen). Nasze kursy dodatkowe maksymalnie upraszczały naukę, a w edytorze projektów przygotowujemy Was do prawdziwej pracy.

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 :) Na końcu wysyłamy zadanie do mentora - opcja **Wyślij do sprawdzenia** na szarym pasku.

Po sprawdzeniu efektu zadania, Mentor skomentuje Twój kod i poinformuje Cię o tym :)

Poniżej zamieszczono instrukcje zadania. Stworzymy pierwszą podstawową stronę w HTML/CSS.

1. Wstaw do index.html strukturę HTML.
2. Dodaj tytuł strony (`<title>`), potem ten sam tytuł umieść w nagłówku pierwszego stopnia w treści strony (największym).
3. Stwórz 3 paragrafy tekstu i nad każdym z nich nagłówek niższego poziomu niż H1.
4. Wypełnij nagłówki i paragrafy treścią (jako strona "O mnie").
5. Dodaj w treści co najmniej 2 działające linki do swoich ulubionych stron, tak aby otwierały się w nowej zakładce, a nie tej samej. W tym celu wewnątrz tagu `<a>` użyj atrybutu `target="_blank"`.
6. Dodaj do swojej strony co najmniej 2 zdjęcia, np. z darmowego serwisu pexels. Pobierz link do interesującego Cię zdjęcia w ten sposób i wstaw go w HTML za pomocą `<img src="...">` jako źródło zdjęcia.
7. Jeśli strona jest gotowa, to skopiuj zawartość pliku *index.html* z edytora zadania, utwórz nowy plik w notatniku, wklej kod i zapisz na pulpicie komputera. Oczywiście nowy plik zapisz jako *index.html*.
8. Zmień dowolny tekst w kodzie strony z poziomu edytora tekstu i zapisz zmiany. Uruchom plik, otworzy się w przeglądarce. Działa? :)
9. Teraz zmieniony plik wgraj tu do walidatora W3C i naciśnij "Check".
10. Zrób screen wyników walidacji za pomocą aplikacji screenShu. Wykonany zrzut ekranu umieść w projekcie w strukturze HTML (edytor) w formie linku prowadzącego do tego zrzutu ekranu (użyj tagu `<a>`). Tak, żeby mentor oglądający Twój projekt mógł sobie kliknąć w ten link i zobaczyć w ten sposób wyniki walida

Podgląd zadania

Przejdź do projektu ✔

1.2. Działamy z CSS!

Sugestia

Przed rozpoczęciem tego modułu (lub po teorii) warto przejść ten kurs Podstawy CSS, aby przećwiczyć w praktyce podstawowe zagadnienia, co pomoże przy wykonywaniu zadań.

Po co nam CSS?

Jak już wiesz, HTML odpowiada za strukturę strony. Jednak wygląd tej strony definiujemy za pomocą pliku CSS (lub wielu plików w przypadku większych witryn).

Przy pomocy stylów CSS możesz zmienić na swojej stronie rzeczy takie, jak np.:

- rozmiar i kolor czcionki,
- grafika w tle,
- obramowanie elementów,
- rozmieszczenie elementów,
- ...a nawet wprowadzisz animacje!

W tej części zajmiemy się zmianą wyglądu elementów HTML, bo na początku do tego najczęściej służą nam arkusze stylów CSS.

Podłączenie CSS do HTML i przeciążanie

Pliki ze stylami CSS podpinamy w nagłówku strony (między tagami <head> a </head>) za pomocą tagu <link>. Jest to tag, który nie wymaga zamykania.

Przykładowe podpięcie do HTML pliku ze stylami wygląda następująco:

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

W powyższym przykładzie za pomocą atrybutu `href` wskazujemy plik, w którym znajdują się utworzone przez nas reguły. 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 jest głębiej, 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 uznaje go za bardziej aktualny.

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

```
1 <link rel="stylesheet" type="text/css" href="first-file.css">
2 <link rel="stylesheet" type="text/css" href="second-file.css">
```

Jeżeli w obu plikach znajdzie się reguła nadająca kolor tekstu akapitom, to przeglądarka użyje 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 nadpisań (przeciążeń) stylów.

Konstrukcja stylu

Przykładowa reguła (styl) CSS wygląda następująco:

```
p {
    color: red;
    font-size: 24px;
}
```

Interpretujemy ją w ten sposób:

```
selector {
    property: value;
    property: value;
}
```

Opis elementów reguły:

- `selector` - określa element HTML, którego dotyczy styl,
- `{ ... }` - klamry, oznaczają blok kodu, wewnątrz nich trzymamy właściwości oraz ich wartości,
- `property` - właściwość, to coś jak cecha wyglądu, np.: kolor, wysokość, szerokość itp.,
- `:` - dwukropek, oddziela właściwość od wartości jaką jej przypisujemy,
- `value` - wartość, określa jaka ta cecha ma być, np. dla koloru określa, że ma być zielony, a dla wysokości, że ma ona wynosić np. 300px itp.,
- `;` - średnik, stawiamy zawsze na końcu linii kodu, jest to uniwersalna praktyka funkcjonująca w bardzo wielu językach programowania.

Popularne selektory

Najczęściej spotykany jest **selektor klasy** przypisanej do elementu. Jeżeli mamy element HTML z atrybutem `class="banner"` to możemy ostylować dany element używając jego klasy jako selektora. Przy okazji wszystkie inne elementy z tą klasą również otrzymają zdefiniowane w CSS reguły wyglądu.

W CSS regułę z selektorem dla klasy tworzymy poprzez dodanie kropki przed nazwą klasy. Przykładowy styl stworzony z selektorem klasy:

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

Taki styl zadziała np. na `<p class="banner">` - nada paragrafowi w HTML czerwony kolor tekstu.

Popularne są też **selektory dla tagu**. W tym przypadku nie stosujemy żadnego przedrostka w CSS. Jeżeli chcemy nadać jakieś style dla konkretnego tagu z HTML, wystarczy, że użyjemy jako selektora jego nazwy.

Przykładowa reguła dla elementu `<h1>`:

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

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

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

Przykład:

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

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 :)

Dla niecierpliwych

Spis wszystkich funkcjonujących selektorów znajdziesz tutaj

Frequently Used Styles :)

Jest pewna popularna grupa właściwości CSS, których używa się bardzo często. Abyś nie musiał błądzić po setkach właściwości, poniżej wymieniamy najpopularniejsze:

- **color** - określa kolor tekstu w elemencie np. `color: black`,
- **background** - za jego pomocą możemy określić właściwość tła np. kolor `background: red`,
- **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: 'Open sans'`,
- **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 elemencie HTML (kontener).

Listę wszystkich właściwości oraz ich wartości znajdziesz na stronie W3schools. To bezcenne źródło informacji. Polecamy także odwiedzić CSSreference - jest to skarbnica wiedzy o właściwościach i wartościach CSS wraz z przykładami.

Kolory w CSS

Jest kilka sposobów na zdefiniowanie koloru (np. kolor czcionki lub tła). Stosowane są w zależności od potrzeb i upodobań programistów.

Przykłady najpopularniejszych form zapisu kolorów:

- **nazwy koloru w języku angielskim** - blue, green, black, white (link do pełnej listy);
- **heksadecymalny (hex)** - czyli kod koloru zapisany w systemie szesnastkowym, przykładowe kolory to: `#000000` (czarny), `#00FF00` (kolor limonki), `#FFFFFF` (biały). Można także zapisać kolor w ten sposób `#FFF` (biały) gdzie każdy znak finalnie się "podwaja". Nie musisz znać sposobu jak "obliczyć" dany kolor, wystarczy Ci tzw. *color picker* do generowania oczekiwanego koloru (przykładowy na Kodilla, ale możesz wygooglować w sieci);

- **RGB** - jest to wypadkowa 3 kolorów (red, green, blue). Zapisuje się go w ten sposób: np. `rgb(255, 0, 0)` daje kolor czerwony, a `rgb(255, 255, 255)` - kolor biały. Każda z wartości po przecinku to liczba z zakresu 0-255. Podobnie jak dla hex, programiści najczęściej generują kolory w RGB za pomocą *color pickerów*;
- **RGBa** - działa analogicznie jak RGB, ale ma jeszcze dodatkowy (czwarty) parametr odpowiadający za przeźroczystość, np. `rgba(255, 0, 0, 1)` wyświetli kolor czerwony, a `rgba(255, 255, 255, 0.5)` da nam kolor biały, ale półprzezroczysty. Ostatnia wartość może przyjmować wartości od 0 (całkiem przeźroczysty) do 1 (zupełnie nieprzezroczysty). Wartości podajemy jako ułamek z kropką (nie przecinkiem).

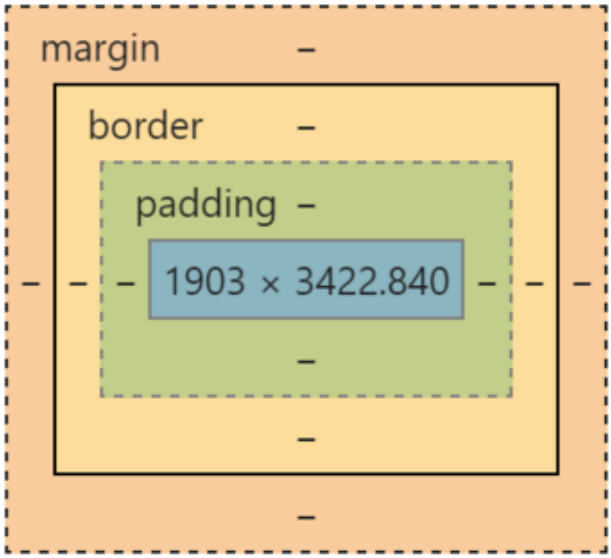
Najczęściej używa się kolorów zapisanych heksadecymalnie, głównie ze względu na prosty zapis. Minusem tego zapisu jest to, że nie można ustawić w nim przeźroczystości. Dlatego, jeśli jest nam ona potrzebna, sięgamy po zapis **RGBa**.

Box-model w CSS

Box-model to założenie, że każdy umieszczony na stronie element HTML jest swego rodzaju pudełkiem (kontenerem), a każde takie pudełko posiada kilka cech wyglądu, które go charakteryzują.

Wyobraź sobie pudełka umieszczane wewnątrz 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 deweloperskiego *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`). A za pomocą `margin` określamy odległość innych elementów HTML od bordera naszego pudełka z treścią.



Składowe pudełek

Nasze "pudełka", w przeciwieństwie do tych prawdziwych, nie muszą mieć wszystkich ścian. Możliwe jest na przykład ustawienie 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 ramkę na górze elementu, skorzystamy z właściwości `border-top`.

Wartości "po zegarze"

Wiesz już, jak zrobić ramkę tylko na górze elementu. Co w przeciwnym przypadku - gdy Twoim zamiarem jest zrobienie ramki na każdym boku pudełka oprócz góry? Czy trzeba napisać 3 linijki kodu odpowiadające za właściwości `border-right`, `border-bottom` i `border-left`, zostawiając `border-top` domyślnie ustawione na `0`?

Na szczęście NIE - w takim przypadku wykorzystamy znów właściwość `border`. Jeśli dla `border` podamy **cztery wartości**, np. `border: 0 1px 2px 3px`, ustawimy inną grubość ramki 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 górnej ramki (jej grubość jest ustawiona na `0px`), prawa miałaby `1px`, dolna `2px`, a lewa `3px` grubości.

Dla dociekliwych

Przeglądarki czytają arkusze stylów które im dostarczamy i je interpretują. To już wiemy. Ale jak sobie radzą z takimi zapisami jak ten?

```
border: 10px 5px 10px 5px;
```

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

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

Spostrzegawcze osoby zauważą, że dla górnej i dolnej ramki wartości są takie same. Również dla bocznych mają one tą samą wartość. Może da się to jeszcze jakoś uprościć? Oczywiście, że się da :)

Ten zapis:

```
border: 10px 5px 10px 5px;
```

Możemy również przedstawić w ten sposób:

```
border: 10px 5px
```

A przeglądarka i tak go sobie przetłumaczy tak:

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

Jest wiele właściwości, które zachowują się w ten sposób np. `margin` czy `padding`.

Dobre praktyki z CSS - walidacja kodu, debugowanie

Kiedy skończymy pisać reguły w CSS, warto sprawdzić ich poprawność. W tym celu odwiedzamy stronę css-validator.org, na której mamy możliwość sprawdzenia naszego kodu. W walidatorze możemy podać adres naszej witryny, wrzucić gotowy plik albo wkleić nasz kod.

Dobrym zwyczajem jest także debugowanie kodu. W tym celu musimy otworzyć narzędzie dla developerów, klikając na interesującym nas elemencie prawym przyciskiem myszy i wybierając opcję "Zbadaj element". Po lewej stronie Inspektora widzimy nasz kod HTML, prawa strona to reguły CSS. Za pomocą tego narzędzia możemy zobaczyć jakie reguły działają na dany element.

Zadanie: Lifting strony

W tym zadaniu będziesz szlifować swój projekt rozpoczęty w poprzedniej sekcji.

1. Zacznij od skopiowania kodu z poprzedniego zadania i wklejenia go do tego projektu w pliku *index.html*.
2. Podepnij plik *style.css*.
3. W pliku *style.css* napisz selektor dla tagu `<body>` i dodaj po nim klamry `{}`.
4. Wewnątrz tak utworzonej reguły (między klamrami `{}`) za pomocą właściwości `background-color` nadaj tło dla całej strony o kolorze np. `#2ecc71`.
5. Przejdź na chwilę do pliku *index.html*. Wewnątrz elementu `<body>` utwórz tag `<div>` z klasą `content"` (`class="content"`), a jego zamykający odpowiednik (`</div>`) umieść tuż przed tagiem `</body>`. W ten sposób cała treść strony znajdzie się wewnątrz elementu div. Będzie to nasz kontener :)
6. Wróć do pliku *style.css* i utwórz w nim regułę dla świeżo dodanego elementu, sięgając po niego za pomocą jego klasy.
7. Wewnątrz nowej reguły ogranicz szerokość naszego kontenera (czyli wymieniony wcześniej znacznik div z klasą `content` do przechowywania innych elementów) do `500px`.
8. Dodatkowo nadaj mu margines, dzięki któremu będzie wyśrodkowany w poziomie (`margin: 0 auto;` więcej na temat tej praktyki dowiesz się w kolejnym submodule).
9. To nie wszystkie reguły dla naszego kontenera :) Teraz nadaj mu białe tło. Dla `background-color` użyj wartości: `white`, `rgb(255,255,255)` lub `#FFF`.
10. Teraz dodaj `padding` o wartości `20px`.
11. Stwórz kolejną regułę, tym razem dla tagu `<h1>`, a następnie wyśrodkuj w niej tek za pomocą właściwości `text-align`.
12. Otwórz plik *index.html*.

13. Zaraz pod tagiem `<body>`, przed `<div class="content">`, wstaw nagłówek strony. W tym celu stwórz tag `<div>` z klasą `header`. Pamiętaj o zamknięciu go.
14. Wewnątrz tego tagu utwórz tag `<h1>` i umieść w nim tekst witający na Twojej stronie.
15. Zauważ, że zdjęcia mogą wychodzić poza nasz div o klasie `content`. `width: 100%` dla tagu `<img>` to deklaracja, która rozwiąże nasz problem. Zdjęcia będą zajmować teraz 100% szerokości swojego rodzica.
16. Następnym krokiem będzie stworzenie stopki. Pod zamknięciem elementu o klasie `content` otwórz tag `<div>` o klasie `footer`. Wewnątrz niego umieść paragraf z tekstem "© [Twoje imię i nazwisko], 2018".
17. Brawo! Utworzyłeś szkielet HTML dla prostego layoutu strony. Czas dodać nagłówkowi i stopce trochę życia, korzystając z CSS.
18. Przejdź do pliku *style.css*. Korzystając z selektora `body`, ustaw margines całej zawartości strony na `0`.
19. Następnym krokiem będzie ostylowanie nagłówka strony. Utwórz dla niego selektor, korzystając z nadanej mu klasy. Wewnątrz tego selektora nadaj mu właściwości: centrowanie tekstu, `padding` o wartości `30px 0`, kolor tła: `#f39c12` oraz biały kolor tekstu.
20. W tym kroku zajmiesz się ostylowaniem stopki. Znów utwórz selektor, korzystając z klasy, którą nadałeś stopce. Wpisz w nim właściwości: kolor tła `#7f8c8d`, zmiana całego tekstu na wielkie litery (zrobisz to za pomocą `text-transform: uppercase`) oraz pogrubienie czcionki.
21. Aby na Twojej stronie zagościła stopka z prawdziwego zdarzenia, zajmij się ostylowaniem zawartego w niej paragrafu tekstu. Utwórz dla niego selektor i spraw, aby marginesy wynosiły 0, a `padding` wynosił `30px` na górze, `0` po prawej, `15px` na dole i `20px` od lewej strony.

Kliknij przycisk niżej, aby przejść do edytora.

Podgląd zadania

Przejdź do projektu ✔

1.3. Prosta strona

Sugestia

Przed rozpoczęciem tego modułu (lub po teorii) warto przejść ten kurs Semantyka HTML5, aby przećwiczyć w praktyce podstawowe zagadnienia, co pomoże przy wykonywaniu zadań.

Pamiętaj, że trening czyni mistrza! Teraz, gdy znasz już podstawy posługiwania się zarówno językiem HTML, jak i CSS, przeprowadzimy Cię przez proces tworzenia strony "O mnie". Potraktuj tę sekcję **jako jedno duże zadanie**. Poniższe zapisy wykonaj po przeczytaniu całej sekcji.

Uwaga: w poprzednim zadaniu stosowaliśmy głównie tagi div. W poniższym będziemy starać się używać jak najwięcej tagów HTML5.

Projektowanie

Projekt strony WWW może powstawać m.in. na 2 poniższe sposoby:

- Dostajesz projekt graficzny od grafika i na jego podstawie tworzysz szablon html/css.
- Tworzysz stronę dla siebie bez projektowania jej w Photoshopie, najpierw tworzysz jej ogólny zarys (prototyp) i na tej podstawie samą stronę.

My w tej sekcji zajmiemy się na razie drugim podejściem.

Przed każdym projektem należy się zastanowić, jakie elementy chcesz mieć na swojej stronie oraz jakie treści chcesz na niej umieścić. W każdym przypadku (prawie zawsze), przy tworzeniu strony używamy nagłówka i stopki. To możemy uznać za standard.

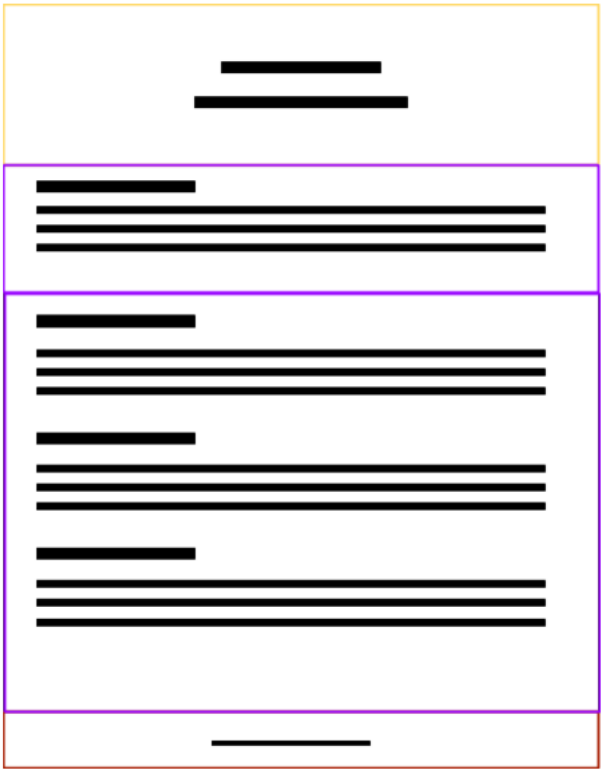
Zastanówmy się nad treścią. Dobrze byłoby się przedstawić i napisać kilka zdań na swój temat. To będzie nasza **pierwsza sekcja**.

W **drugiej sekcji** umieścisz pozostałe informacje pod kolejnymi nagłówkami, wypiszesz więcej ciekawostek na swój temat. Pochwalisz się swoim hobby, swoją wymarzoną pracą, ulubioną muzyką czy filmami.

Oczywiście możesz umieścić inne informacje niż te, które sugerujemy, najważniejsze by zachować założoną strukturę HTML :)

Dobrą praktyką jest rozrysowanie sobie projektu. Wystarczy kartka papieru i długopis. Dzięki takiej wizualizacji łatwiej jest oszacować to, jakich elementów potrzebujemy do osiągnięcia celu oraz jak rozlokować przestrzeń strony. Wielu webdeveloperów robi tak przed rozpoczęciem pracy z kodem.

Poniżej zobaczysz przykładowy szkic tego, co planujemy zbudować. Żółtym kolorem jest zaznaczony **nagłówek**, fioletowym podział na **sekcje**, bordowego użyliśmy dla **stopki**, a dla **tekstów** zastosowaliśmy grubsze czarne kreski.



Czasami spotkasz się z określeniem takiego rysunku jako: makiety lub prototypu (ang. *mockup, wireframe*). Jest to nic innego, jak wstępna wizualizacja układu strony. Jak już mówiliśmy, do rysowania prototypu można użyć kartki lub specjalnego programu online do tego celu, np. Moqups. Posiada on zestaw wielu gotowych elementów, za pomocą których możesz szybko stworzyć prototyp swojej strony.

Prototyp ma tę zaletę, że zanim zakodujesz stronę, możesz kilka razy zastanowić się czy czegoś nie zmienić. Pozwala to oszczędzić czas na dokonywanie takich modyfikacji na zakodowanym szablonie HTML/CSS.

Struktura HTML

Rozpoczniemy od stworzenia podstawowej struktury dokumentu zgodnej ze standardami HTML5 oraz podpięcia pliku ze stylami. Wspominaliśmy już o niej.

Następnie użyjmy tagów wprowadzonych w HTML5, takich jak:

- **header** dla nagłówka,
- **section** dla sekcji zawierających tekst (aby je odróżnić, dodaj im dwie różne klasy),
- **footer**, by stworzyć stopkę na dole strony. Pamiętaj o stworzeniu dwóch sekcji.

Odwzorowanie w kodzie HTML struktury rozrysowanej na makiecie powinno więc wyglądać tak:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>My website title</title>
        <link rel="stylesheet" type="text/css" href="style.css">
    </head>
    <body>
        <header></header>
        <main class="container">
            <section class="about"></section>
            <section class="interests"></section>
        </main>
        <footer></footer>
    </body>
</html>
```

Równocześnie stworzymy w pliku CSS selektor wybierający `body` i nadamy mu margines wynoszący `0` (dla wartości zero nie trzeba przypisywać jednostki, np. px). Sprawi to, że nie będzie domyślnego marginesu jaki przyjmuje body.

Wygląda to tak:

```
body {
    margin: 0;
}
```

Najprostszy layout

Co jeśli chcesz, aby całość była wyśrodkowana w kolumnie, jak na większości stron? Czyli nasze pudełka będą miały stałą szerokość, a marginesy po bokach będą także o równej szerokości, która będzie ustalana automatycznie zależnie od szerokości przeglądarki.

Wystarczy, że wszystkie nasze pudełka umieścimy w jednym głównym pudełku (kontenerze), czyli w tagu `<main>` o klasie `container`. Następnie w CSS dla tego kontenera (selektor dla klasy) należy dać styl `margin: 0 auto`, który dla całego kontenera zawartości po jego prawej i lewej nada automatyczny margines (dzięki wartości auto). Teraz ustalamy maksymalną szerokość (`max-width`) tego kontenera np. na `700px` (bez tego automatyczne marginesy po bokach nie zadziałają bo element będzie miał cały czas 100% domyślnej szerokości).

Możesz też nadać wyśrodkowanemu kontenerowi w CSS biały kolor tła (`#FFF`), a całej stronie body kolor szary (`#CCC`). Wtedy wizualnie kontener będzie wyraźnie oddzielony.

Po stworzeniu szkieletu i prostego layout strony czas dokładnie zająć się każdym z elementów po kolei.

Nagłówek

Nagłówek (header) znajduje się w górnej części strony, zatem jest pierwszym jej elementem, jaki widzi odwiedzający. Użyjemy go do powitania naszych gości :)

W nagłówku strony umieść paragraf z powitaniem (tag `<p>`) oraz swoje imię, które zamkniesz w tagu `<h1>`.

Kod HTML nagłówka powinien więc wyglądać tak:

```
<header>
    <p>Hello, my name is</p>
    <h1>Tomek</h1>
</header>
```

Przejdź do pliku CSS. Najpierw za pomocą selektora wybierającego nagłówek nadaj mu padding równy `30px`. Korzystając np. z darmowego z serwisu pexels, znajdź zdjęcie, które umieścisz w tle nagłówka strony za pomocą właściwości `background-image: url(link go grafiki)`.

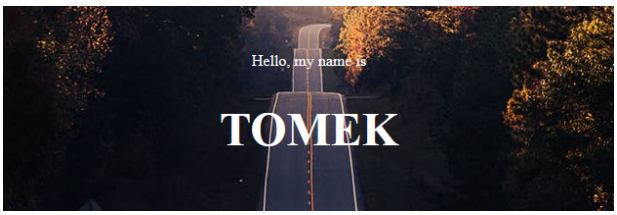
Następnie stwórz selektor, za pomocą którego ostylujesz tylko **nagłówek h1 będący dzieckiem elementu <header>** i pogrub tekst w nim zawarty oraz spraw, by wszystkie litery napisu były wielkie (podpowiedź: przyda Ci się do tego właściwość text-transform).

Powiększ tekst nagłówka `h1` do rozmiaru `48px`. Kolejny selektor niech wybiera zarówno `h1`, jak i `p`. Ustaw taki kolor tekstu, aby wyróżniał się na tle wybranego przez Ciebie zdjęcia. Niech tekst będzie wyśrodkowany.

Wzorcowe style:

```
header {
    padding: 30px;
    background-image: url(https://static.pexels.com/photos/...);
}
header h1 {
    font-weight: bold;
    text-transform: uppercase;
    font-size: 48px;
}
header p, header h1 {
    text-align: center;
    color: white;
}
```

Efekt w przeglądarce:



Kaskadowość stylów, dziedziczenie, selektory

Ostateczne formatowanie strony jest zależne od wielu czynników. Istnieje hierarchia reguł CSS, która decyduje jak finalnie będzie wyglądał formatowany element. Jeżeli dany element ma przypisaną więcej niż jedną regułę, np. dla koloru tekstu, to zastosowanie będzie mieć ta, która jest wyżej w hierarchii.

Od najważniejszych:

1. styl lokalny (inline),
2. rozciąganie stylu (span),
3. wydzielone bloki (div),
4. wewnętrzny arkusz stylów,
5. import stylów do wewnętrznego arkusza,
6. zewnętrzny arkusz stylów,
7. import stylów do zewnętrznego arkusza,
8. atrybuty prezentacyjne HTML - np. color="...", width="..." i inne).

Wyobraźmy sobie sytuację, że w pliku style.css umieścimy:

```
div {
    color: red;
}
```

a nasz HTML będzie wyglądał:

```
<div style="color: green">content</div>
```

Według kaskadowości stylów nasz div będzie miał kolor zielony, gdyż styl inline ma wyższy priorytet ważności i pierwszeństwo w modyfikowaniu koloru czcionki w podanym divie.

Kolejnym czynnikiem wpływającym na końcowy styl elementów jest dziedziczenie. Każdy element html posiada swoich “rodziców" od których otrzymuje style i "dzieci", którym je narzuca. Trudne? Wyobraźmy sobie sytuację:

```
<body>
    <div>
        <p>Content</p>
    </div>
</body>
```

Kiedy do elementu `<body>` dodamy właściwość `color: green`, to tekst diva, który jest w środku `<body>`, będzie zielony. Odziedziczy on kolor po swoim rodzicu.

Duży wpływ na finalne stylowanie strony ma stopień ważności selektorów.

Od najważniejszych:

- P1 - selektory identyfikatora: `#name { color: #000; }`,
- P2 - selektory atrybutu, klasy i pseudoklasy `.name { color: #000; }`,

- P3 - selektory pseudoelementu i elementu `p { color: #000; }`,

Sekcje

Zajmijmy się teraz sekcjami. Umieścisz w nich kilka zdań o sobie - czym się zajmujesz, ile masz lat - wpiszesz to, na co masz ochotę.

W pierwszej sekcji utwórz nagłówek `<h2>` informujący, o czym będzie tekst znajdujący się poniżej. Stwórz paragraf i umieść w nim te informacje.

```
<h2>Who am I?</h2>
<p>A few words about me.</p>
```

Druga sekcja to rozwinięcie informacji o Tobie - dodaj tam nieuporządkowaną listę (tag `<ul>`) z Twoimi hobby oraz napisz w paragrafie kilka słów o swoich umiejętnościach. Niech lista oraz paragraf będą poprzedzone nagłówkami h2.

```
<h2>I have different interests:</h2>
<ul>
    <li>programming</li>
    <li>...</li>
    <li>and many others!</li>
</ul>
<h2>More about me:</h2>
<p>I have a lot of skills!</p>
```

Teraz nadajemy style dla kontenera. Używając selektora `.container`, dodajemy mu maksymalną szerokość (`max-width`) `700px`, szerokość `100%` oraz `margin: 0 auto;`.

Mamy już zawartość i kontener, teraz napisz selektor wybierający elementy `<section>`. Nadaj im `padding` wynoszący po `20%` z prawej i lewej strony.

Warto, aby sekcje różniły się od siebie stylistycznie. Korzystając z nadanych obu sekcjom różnych klas, możesz na przykład użyć innego koloru nagłówków `h2` dla pierwszej sekcji niż dla drugiej.

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.

Możesz także wybrać także inny schemat kolorów (ang. color scheme) za pomocą tego narzędzia. W górnej części narzędzia możesz wybrać schemat 1-, 2-, 3-, 4-kolorowy. Takie bezpieczne zestawy pozwalają bez kompetencji graficznych w miarę dobrze dobrać kolory, aby przyjemnie się komponowały.

Gdy już wybierzesz kolory, jakich chcesz użyć na swojej stronie, utwórz kolejne dwa selektory. Najpierw dla nagłówków `h2` będących potomkiem elementu o klasie nadanej pierwszej sekcji (tutaj: `about`) i ustaw w nim właściwości: margines górny `60px`, rozmiar czcionki `28px` i jeden z wybranych kolorów czcionki. Drugi selektor niech wybiera nagłówki `h2` w drugiej sekcji. Wpisz w nim właściwości: margines `40px` od góry i drugi z wybranych kolorów czcionki.

Kod obu sekcji powinien teraz wyglądać tak:

```
<section class="about">
    <h2>Who am I?</h2>
    <p>I'm 20 years old and I'm studying at a polytechnic. I hav
</section>
<section class="interests">
    <h2>I'm interested in:</h2>
    <ul>
        <li>new technologies</li>
        <li>sport</li>
        <li>aquaristics</li>
        <li>modern art</li>
    </ul>
    <h2>More about me:</h2>
    <p>I have been practicing karate for 5 years and earning pri
    <p>Recently I have started to learn web development. My goal
</section>
```

A dotyczące ich (oraz kontenera) style CSS:

```css
.container {
    max-width: 700px;
    width: 100%;
    margin: 0 auto;
}
section {
    padding: 0 20%;
}
.about h2 {
    margin-top: 60px;
    font-size: 28px;
    color: #FFBA08;
}
.interests h2 {
    margin-top: 40px;
    color: #3F88C5;
}
```

Twoje sekcje powinny teraz wyglądać podobnie, ale możesz wybrać inne kolory i wielkości w pikselach:

Stronie możesz nadać dowolny wygląd, tematykę. Możesz także dodać zdjęcia.

Stopka

W stopce zwykle umieszcza się dane kontaktowe, linki do profilów w mediach społecznościowych, informacje o stronie, a często także mapę jej zawartości w przypadku większych portali.

Umieść w niej paragraf ze swoim imieniem i nazwiskiem oraz tekstem "You will find me on:", umieszczając za nim linki do mediów społecznościowych wraz z ich nazwami (np. You will find me on: Facebook | Twitter - gdzie w formie napisów Facebook oraz Twitter będą ukryte linki do Twoich profilów na tych stronach). Jeśli w fazie tworzenia strony nie chcesz podawać linków, niech wartość atrybutu href tagu a wynosi "#".

```
<p>You will find me on: <a href="#">Facebook</a> | <a href="#">Twitte
```

W pliku CSS stwórz selektor wybierający stopkę. Nadaj jej właściwości: górny margines `60px` (by odsunąć ją od reszty strony), `padding: 30px;` oraz dowolny kolor tła.

Następnie napisz selektor, który będzie dotyczył paragrafu tekstu umieszczonego w stopce. Nadaj w nim właściwości odpowiadające za wycentrowanie tekstu w poziomie i ustawienie koloru tekstu.

Ostatni selektor niech dotyczy linków umieszczonych w tym paragrafie. Spraw, by zniknęły podkreślenia (wykorzystaj do tego właściwość text-decoration), a tekst był pogrubiony i miał wyróżniający się kolor.

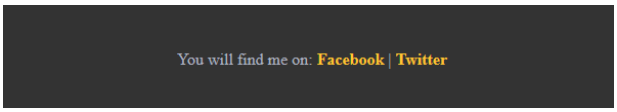
Struktura HTML stopki powinna wyglądać więc tak:

```
<footer>
    <p>You will find me on: <a href="#">Facebook</a> | <a href="
</footer>
```

A dotyczący jej CSS:

```
footer {
    margin-top: 60px;
    padding: 30px;
    background: #333333;
}
footer p {
    text-align: center;
    color: #B0B5C4;
}
footer a {
    text-decoration: none;
    color: #FFC532;
    font-weight: bold;
}
```

Po zinterpretowaniu kodu przez przeglądarkę stopka powinna wyglądać tak, ale możesz skorygować to pod swój gust:



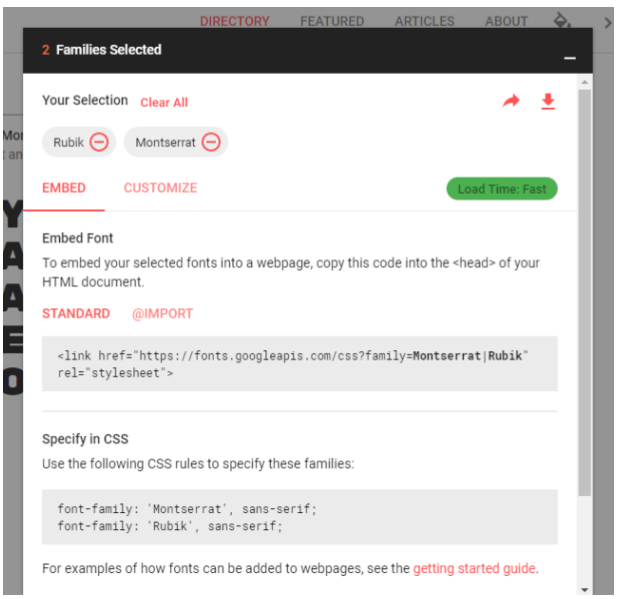
Lifting czcionki

Bardzo ważnym elementem webdesignu (oprócz kolorów - używaj schematów kolorów!), o który dba każdy front-endowiec, jest czcionka.

Na swojej stronie możesz użyć dowolnej czcionki dostępnej w serwisach typu Google Fonts, Font Squirrel lub Typekit. W przypadku Google Fonts wystarczy, że dodasz do niej link w części <head> w pliku HTML. Mniej poprawną alternatywą tego rozwiązania jest umieszczenie w CSS `@import` - czcionki wtedy ładują się wolniej i nie mamy pewności, czy zadziałają na stronie przed załadowaniem się treści.

Strona Google Fonts oferuje całą kolekcję darmowych czcionek. Znajdziesz tam także deklaracje, które należy umieścić w pliku HTML/CSS, by użyć wybranego kroju czcionki.

Dla przykładu zastosujemy dwie czcionki: Montserrat oraz Rubik. Odnajdźmy je na stronie Google Fonts i kliknijmy ikonę plusa przy każdej z nich, by dodać je do naszego zestawu. Następnie kliknijmy pasek, który pojwił się u dołu strony, by mieć dostęp do dodatkowych opcji oraz do kodu naszych czcionek.



W zakładce "Customize" obu czcionkom wybierzmy dwa style: "regular 400" i "bold 700". Wróćmy następnie do zakładki "Embed", by uzyskać kod, dzięki któremu będziemy mogli użyć fontów na naszej stronie. Przed podlinkowanymi stylami w sekcji `head` umieśćmy:

```
<link href="https://fonts.googleapis.com/css?family=Montserrat:4
```

Wagi i style czcionek

Być może znasz już najprostsze deklaracje, których możemy użyć do pogrubienia (czyli zmiany wagi) 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 zaznaczyć wybrane opcje w zakładce "Customize", a następnie odnieść się do nich w stylach w następujący sposób:

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

Niech podstawową czcionką na stronie będzie `Rubik`, z kolei `Montserrat` zastosujmy do nagłówków `h1` oraz `h2`. W tym celu w selektorze body dodaj właściwość: `font-family: 'Rubik', sans-serif;`.

Następnie napisz selektor zbiorowy wybierający wszystkie nagłówki `h1` i `h2` na stronie. Umieść w nim właściwość: `font-family: 'Montserrat', sans-serif;`. Od razu lepiej!

Fallback, czyli alternatywa

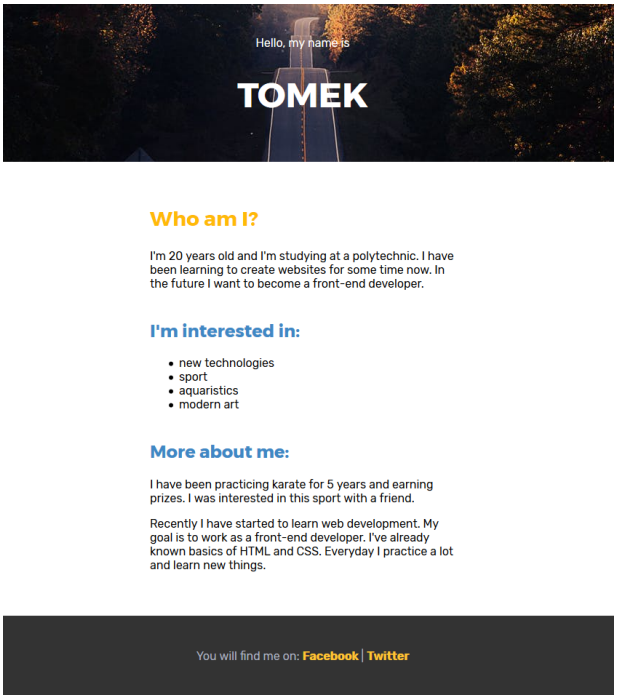
Zauważ, że w powyższych deklaracjach, oprócz nazw czcionek, które wybraliśmy, umieściliśmy po przecinku `sans-serif`. Jest to tak zwany fallback, czyli wskazówka dla przeglądarki, jakiego fontu powinna użyć, gdyby pierwsza zadeklarowana czcionka była z jakiegoś powodu niedostępna. Dodawanie fallbacków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

W swoim projekcie wykorzystaj te same kroje czcionek lub znajdź inne.

Spróbuj także wykorzystać inne właściwości CSS, o których nie wspomnieliśmy w kursie, a jakie uda Ci się znaleźć na Mozilla Developer Network, a także cssreference.io.

Zadanie: Projekt

1. Zbuduj stronę "O mnie" według instrukcji z całej sekcji wyżej.
2. Twoim celem jest zbudowanie strony mniej więcej o takim wyglądzie jak niżej.



Podgląd zadania

Przejdź do projektu ✔

Regulamin

Polityka prywatności