3. Tniemy layout od grafika!

Wyzwania:

- Dowiesz się, czym jest layout i jak jest budowany za pomocą HTML i CSS.
- Poznasz metody i narzędzia wykorzystywane przy budowaniu stron internetowych na podstawie projektu graficznego.
- Na podstawie dostarczonego przez nas projektu graficznego zbudujesz od początku do końca nowocześnie wyglądającą stronę internetową.

Skompletowanie materiałów

3.1. Proces "cięcia"

Jak tworzony jest wygląd?

Proces tworzenia strony internetowej najczęściej (lecz nie zawsze) rozpoczyna się od otrzymania od grafika projektu, na podstawie którego tworzymy nasz kod. Projekt ten najczęściej ma postać pliku PSD lub PNG. PSD jest to rozszerzenie, którego używa program **Adobe Photoshop**, czyli najpopularniejsze obecnie narzędzie pracy dla grafików.

Istnieje też darmowa alternatywa. To popularny program GIMP lub Pixelmator działający na systemie OS X. Jest też program online Pixlr. Problematyczny jest jednak fakt, że praca w tych programach diametralnie różni się od siebie, więc zrobienie czegoś w Photoshopie wymaga zupełnie innych narzędzi niż w GIMP-ie.

Ponadto pliki, do których eksportowane są projekty w obu programach, nie są ze sobą stuprocentowo kompatybilne i GIMP lub Pixelmator, pomimo tego że otworzy plik PSD, może gubić niektóre efekty lub nieprawidłowo wyświetlać poszczególne fonty. Jeśli nie posiadasz Photoshopa albo praca w GIMP-ie (lub innym narzędziu) nie weszła Ci w krew, zawsze możesz poprosić grafika o wersję *.png*, którą otworzy każdy program.

Projekt strony w formacie *.psd* składa się z warstw, z których możemy odczytać wszystkie niezbędne dla nas informacje (zdjęcia, tekst, kolory, wymiary). Obecnie proces tworzenia strony wygląda tak, jak opisujemy to poniżej.

Czynności, którymi się zajmujemy, nazywa się zwyczajowo cięciem, co wywodzi się ze starego sposobu budowania stron. Poszczególne elementy były z pliku .psd wycinane i wstawiane do strony jako obrazki. Obecnie proces ten -wraz z rozwojem CSS, w którym głównie tworzy się oprawę graficzną - uległ jednak ewolucji.

Jeśli jednak musisz wyciągnąć coś z PSD, polecamy narzędzie, które niedawno pojawiło się na rynku - Avocode. Pozwala ono wyciąć interesujące nas elementy i ma także opcje kolaboracji.

Projekt graficzny

Otrzymujemy plik od grafika, np. w formacie *.png*. Naszym zadaniem będzie odczytanie wszystkich istotnych informacji niezbędnych do zbudowania strony na jego podstawie. Należy zmierzyć szerokość kontenerów, rozmiary fontów, szerokość kolumn.

Znajdziemy też w nim najczęściej potrzebne zdjęcia oraz ikony. Konieczne będzie odpowiednie wycięcie tych elementów i zapisanie ich w formatach obsługiwanych przez przeglądarki. Polega to na wyeksportowaniu fotografii oraz elementów wektorowych (ikon elementów, które mają być animowane) do formatu *.png* lub *.jpg*. **Oczywiście o dostarczenie odpowiednio wydzielonych grafik, których potrzebujemy, też możemy poprosić grafika** projektującego stronę. Nie należy bać się pracy z grafikami. Nie gryzą :)

Cięcie...

Na podstawie zebranych informacji budujemy stronę statyczną. To tzw. tworzenie szablonu HTML/CSS. Ten krok pod kątem stosowanej metodyki nie odbiega zbytnio od tworzenia dotychczasowych projektów. **Tutaj podobnie zaczynamy od szkieletu HTML, dodajemy właściwości CSS** oraz tworzymy możliwości interakcji za pomocą kodu JavaScript.

Różnica tkwi jednak w tym, że teraz budujemy dużo większy projekt, dlatego też musimy to zrobić w sposób przemyślany. Kluczem jest w tym przypadku dążenie do ułatwiania sobie pracy. Oglądasz projekt i zastanawiasz się, jak powinna wyglądać struktura HTML z punktu widzenia semantycznego, a także w jaki sposób umieścić odpowiednie właściwości CSS w blokach, które możemy wykorzystać w różnych miejscach.

Musimy też zastanowić się nad tym, czego w pliku *.psd* nie ma, czyli w jaki sposób powinna wyglądać interakcja z użytkownikiem, jak wyglądać będzie animacja oraz tzw. look & feel określający jak strona wygląda i jak się zachowuje. Graficy często przedstawiają to za pomocą dodatkowych warstw w pliku przedstawiających część przycisków w stanie z hoverem, nieaktywnych, odwiedzonych itp.

Alternatywne sposoby projektowania

Proces opisany powyżej nie jest oczywiście jedyną słuszną metodologią i nie jest on pozbawiony wad. Ze względu na stale rosnącą liczbę urządzeń o różnych rozmiarach ekranów nie jesteśmy w stanie zapewnić, że strona będzie wyglądać identycznie na wszystkich urządzeniach.

Pewnym rozwiązaniem jest tworzenie kilku widoków dostosowanych do urządzeń o różnych rozmiarach, co jednak wymaga dodatkowej pracy. **Ponadto w programach graficznych nie sposób ująć wszystkich możliwości**, jakie otwierają przed nami nowoczesne przeglądarki, takich jak np. zaawansowane animacje czy bardziej skomplikowane aplikacje stworzone przy użyciu JavaScript. Dlatego też obecnie metoda "klasyczna", jaką opisujemy, ulega pewnej erozji. Nie oznacza to jednak, że wyszła całkiem z użycia.

Innym podejściem jest budowanie na podstawie prototypów, czyli tak zwany prototype driven development. W takiej sytuacji omijamy całkowicie fazę projektu graficznego na rzecz stworzenia prostej strony z minimalną liczbą funkcji i następnie dostosowywanie jej do wymagań klienta, np. poprzez dodawanie look&feel (kolory, czcionki).

Podejście takie wiąże się często z tym, że zamiast wysokiej jakości projektów graficznych tworzony jest tzw. podręcznik stylu (ang. *style guide*) definiujący poszczególne komponenty strony oraz ich ogólny design. Tworzymy następnie stronę od podstaw, najpierw od ogólnego schematu działania (często na papierze) przez schematy bardziej szczegółowe aż do prototypu działającego w przeglądarce.

Z pomocą przychodzą tutaj gotowe biblioteki komponentów, takie jak **Bootstrap** (omawiany w kolejnych modułach) czy **Foundation**, za pomocą których możemy szybko zbudować działający prototyp. Następnie przychodzi czas na fazę testów. Niektórzy ze stosujących to podejście tworzą więcej niż jeden prototyp i następnie przeprowadzają testy porównawcze. Często też przeprowadzana jest więcej niż jedna faza testów, by ostatecznie "wyszlifować" projekt. Po fazie testów projekt oddaje się już do klienta.

Grafika czy CSS?

Z definicji grafika w formacie *.png* czy *.jpg* ma określoną rozdzielczość i - pomimo tego że możemy ją w pewnym zakresie skalować - nie zapewnimy jej prawidłowego wyglądu na wszystkich ekranach. Dlatego też obecnie minimalizuje się ilość elementów interfejsu w formacie grafiki na rzecz tych stworzonych za pomocą CSS. Przyciski zbudowane z obrazków odchodzą w niepamięć na rzecz rozwiązań bardziej elastycznych. **Mówimy tu o oprawie graficznej, a nie o treści w postaci zdjęć.**

Nie znaczy to jednak, że powinniśmy zrezygnować całkowicie ze stosowania grafik. Wręcz przeciwnie - odpowiednie grafiki stanowią doskonałe uzupełnienie treści strony. Logo raczej trudno wyobrazić sobie jako CSS, to wciąż grafika.

Jednak odchodzenie od grafiki (jako oprawy wizualnej strony) na rzecz CSS ma też jeszcze jedną konsekwencję: graficy w założeniu powinni wiedzieć o tym, co można, a czego nie można uzyskać przy pomocy CSS. Stąd też niezwykle istotna jest komunikacja pomiędzy grafikiem a front-endowcem. Możliwości CSS powinny stanowić ramy, wewnątrz których będzie poruszał się grafik tworzący stronę internetową.

To Photoshop or not to Photoshop?

To pytanie, które zadaje sobie prawdopodobnie każdy początkujący front-endowiec. I, jak to w przypadku takich pytań bywa, nie ma na nie jednoznacznej odpowiedzi. Znajomość jego podstawowych funkcji będzie konieczna, gdy pracujemy z grafikiem w sposób przedstawiony na początku rozdziału, czyli naszym zadaniem jest pocięcie pliku *.psd*.

Nie musimy jednak do tego celu znać zaawansowanych funkcji manipulowania obrazem, wystarczy, że posiadamy pewną swobodę w poruszaniu się po warstwach, umiemy odczytać wymiary poszczególnych elementów layoutu oraz tekstu lub zoptymalizować zdjęcia na potrzeby strony internetowej. Inną sprawą jest, że zdarzają się osoby łączące zawód designera i front-end developera. Trzeba przyznać, że w pewnym zakresie funkcje te się pokrywają. Na razie wystarczy nam Avocode lub Pixlr.

Ważne też, że jeśli chcemy umieścić na stronie zdjęcia, to należy zadbać o ich optymalizację poprzez rozmiar **(nie wsadzamy na stronę zdjęć 2000x1000 pikseli i dajemy im rozmiar 200x100)**. Dodatkowo plikom *jpg* możemy ustawiać jakość od 0 do 100. Często 60-70 to wystarczająca wartość. To wpływa finalnie na "wagę" tych plików i czas wgrywania się strony. Nikt nie lubi jeśli trwa to za długo. Wniosek jest taki, że podstawa znajomości narzędzi graficznych się przydaje.

Skompletowanie materiałów

Jednymi z ważniejszych elementów graficznych strony, które musimy uzyskać od projektanta, są: logo, grafika sekcji powitalnej, ikony. Mają one duży wpływ zarówno na wygląd strony, jak i wydajność (szybkość z jaką strona wyświetla się na komputerze użytkownika) - im większe, cięższe zdjęcie, tym dłużej będzie ono pobierane z serwera.

Dlatego też musimy dobrze zastanowić się przy wyborze formatu zdjęć, których będziemy używać. Programy graficzne dają nam wiele możliwości, jednak dwie najpopularniejsze opcje to format JPEG (albo JPG) oraz PNG.

- **JPG** to format stosujący stratny algorytm kompresji (zapisując zdjęcie, pomija część kolorów). Zdjęcie jest przez to gorszej jakości (szczególnie widoczne w przypadku tekstu czy ostrych krawędzi), ale zajmuje mniej miejsca.
- **PNG** to format bezstratnej kompresji danych (żadne piksele nie są pomijane podczas zapisu), przez co zdjęcia są dużo lepszej jakości. Dodatkową zaletą tego formatu jest możliwość zapisu zdjęcia z przezroczystym tłem, co przydaje się szczególnie przy tworzeniu ikon albo loga strony.

Ze względu na powyżej wymienione cechy wspomnianych formatów najczęściej stosuje się następującą zasadę: dla wszelkiego rodzaju zdjęć, szczególnie tych o dużym rozmiarze i z małą ilością detali, używamy formatu JPG, a zdjęcia mniejsze, pełne ostrych krawędzi albo te, które muszą mieć przezroczyste tło, zapisujemy w formacie PNG.

Na koniec warto wspomnieć o technologii, która zyskuje coraz większą popularność wśród webdeveloperów, czyli **SVG**. Obraz w tym formacie jest niejako rysowany przez przeglądarkę na podstawie kodu XML. Sprawia to, że uzyskany obrazy jest świetnej jakości (we wszystkich rozmiarach) oraz daje duże możliwości w kontrolowaniu wyglądu czy zachowania grafiki z poziomu CSS lub JS. Format SVG najczęściej wykorzystuje się do tworzenia ikon (także tych animowanych) oraz wizualizacji danych.

Weryfikacja i przekazanie

Jako developerzy potrzebujemy grafiki, na której jest przedstawiona strona, najlepiej wraz z niektórymi elementami w stanach typu `:hover` lub `:active`. Jeżeli na stronie są alerty, to oczywiście grafik też musi je umieścić na grafice w odpowiednim miejscu, tak żebyśmy wiedzieli, gdzie mają być w danych sytuacjach i jak wyglądać. Często wydaje się, że pewne rzeczy są "logiczne" i to bywa największą zgubą oraz przyczyną wielu poprawek. Oczywiście w mniejszych projektach developer może, a nawet powinien, posługiwać się intuicją. Jednak w bardziej skomplikowanych projektach wręcz wymagamy od grafików konkretów.

Niestety często jest to bagatelizowane przez designerów. Nie ma się co dziwić - rzadko kiedy zdają sobie sprawę ze złożoności projektu oraz funkcjonalności, mając jedynie ogólny pogląd przedstawiony przez klienta jako podstawę do wykonania zadania. Oczywiście pliki graficzne, prócz głównych poglądowych, których potrzebujemy zawsze, to grafiki użyte jako tło (jeżeli nie da się tego odtworzyć w CSS), logo (o ile nie dostarcza nam go klient) oraz dodatkowe materiały. Tutaj może być różnie, bo część jest elementami graficznymi (od grafika), a część, jak np. zdjęcia pracowników, pozyskujemy jako treść od klienta. Warto sobie omówić na początku kto co dostarcza i czego oraz kiedy możemy się spodziewać.

Materiały dobrze jest trzymać odpowiednio skatalogowane gdzieś w chmurze. Dysk od googla (Google Drive) doskonale sprawdza się w takich przypadkach. Nie tracimy wtedy czasu na szukanie materiałów po mailach, bo ktoś kiedyś w jednym z 200 maili wysłał nam, jak ma ten przycisk wyglądać. Takie sytuacje się zdarzają i są niesamowicie irytujące.

Podsumowując, materiały najlepiej trzymać na wspólnym dysku w odpowiednich katalogach w formie odpowiednich dokumentów. W mailach często coś ucieka naszej uwadze :)

Określenie wsparcia przeglądarek

Na rynku są różne przeglądarki internetowe, najpopularniejsze to: Google Chrome, Firefox, Internet Explorer (teraz nowa od MS -> Microsoft Edge), Safari. Każda z nich ma swoją aktualną stabilną wersję. Jednak ludzie nie zawsze mają najnowszą wersję u siebie na komputerze.

I pojawia się następujący problem: dana strona internetowa może działać (wyglądać) dokładnie jak w projekcie graficznym, a np. w Internet Explorer 8 będzie wyglądać zupełnie inaczej. Bo taka przeglądarka w starszej wersji nie obsługuje np. danej właściwości CSS.

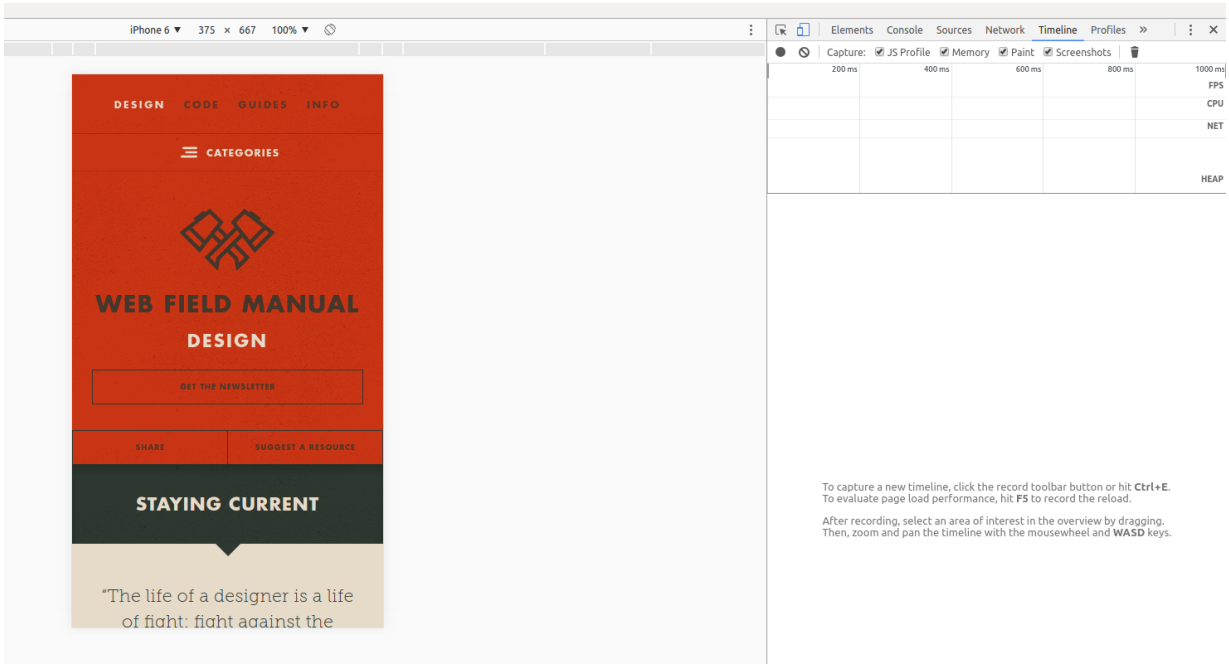
Zazwyczaj problematyczne są starsze wersje Internet Explorera i Safari. Dlatego, podejmując nowe zlecenie, ustala się z klientem, **które przeglądarki mają być obsługiwane**. Sprawdza się na tej stronie obecnie używane przeglądarki i podejmuje się decyzje, których wersji przeglądarek już nie obsługujemy (czyli jaki procent ruchu nie będzie obsługiwany).

Dodatkowo jeśli chcemy dodać na stronę eleganckie gradienty, okrągłe ramki itp., to warto zajrzeć na Can I Use i sprawdzić, czy są one obsługiwane w danej wersji przeglądarki.

Nierzadko też spotkamy się z przypadkiem, gdy strona dobrze prezentująca się w Google Chrome nie będzie równie dobrze wyglądać np. w Safari. Wszystko to przekłada się na dodatkowy nakład pracy. Dlatego też dobrze jest ustalić z klientem (lub samym sobą) ile jesteśmy w stanie tego czasu zainwestować i jakie wersje przeglądarek powinny być wspierane. Można to pytanie sformułować w taki sposób: "czy warto zainwestować X czasu, aby zyskać dodatkowy Y% użytkowników?".

Zawsze trzeba pamiętać o przetestowaniu swojej strony na różnych przeglądarkach. Idealnie byłoby też sprawdzić ją na różnych systemach operacyjnych, tabletach i smartfonach, co w praktyce jest oczywiście niezwykle trudne. W warunkach małego projektu wystarczy zobaczyć stronę na popularnych przeglądarkach: Google Chrome, Firefox, Internet Explorer, Safari. Możesz w tym celu użyć serwisu Browser Stack w darmowym planie.

Wyświetlanie na przeglądarkach mobilnych można zasymulować poprzez odpowiednie zmniejszenie okna. Doskonałym narzędziem do testowania kompatybilności strony z ekranami o zróżnicowanej rozdzielczości jest Chrome Developer Tools. Pozwala ono na symulację nie tylko rozmiarów ekranu, ale też przewijania strony na tablecie czy działania na powolnym łączu mobilnym. Pamiętajmy jednak o tym, że jest to tylko symulacja, która nie odda wszystkich niuansów działania strony na innych urządzeniach (więcej o tym w module o RWD).



Jednak nie dajmy wejść sobie na głowę. W przypadku nieszczęsnego IE przesadą byłoby zakładanie kompatybilności z wersjami wcześniejszymi niż 10. Zawsze można negocjować, dać klientowi ciekawsze i sprawniejsze rozwiązanie w zamian za ograniczenie do nowszych wersji przeglądarek.

Pixel perfect?

Kolejną przeszkodą do pokonania na drodze od projektu graficznego do gotowej strony jest pojawiająca się wątpliwość dotycząca tego, do jakiego stopnia szczegółowości odwzorowywać projekt graficzny na stronie internetowej, czyli pytanie o tzw. pixel perfect. W wolnym tłumaczeniu pojęcie pixel perfect oznacza, że projekt graficzny został odwzorowany z dokładnością do pojedynczego piksela. Perfekcja odwzorowania świadczy o wysokiej jakości wykonanej pracy. Jak łatwo się domyślić, odwzorowanie wszystkiego z dokładnością co do piksela bywa trudne lub praktycznie niemożliwe i również wymaga poświęcenia temu odpowiedniego czasu w komunikacji ze zleceniodawcą.

Spróbujmy wyobrazić sobie, że ktoś otwiera naszą stronę na czarno-białym czytniku Kindle, a ktoś inny na 60-calowym telewizorze z UltraHD. Nie ma technicznej możliwości, żeby to wyglądało identycznie. Rozważając zagadnienie perfekcyjnego odwzorowania projektu graficznego, trzeba wziąć pod uwagę skalę projektu i jego koszty oraz przede wszystkim oczekiwania klienta.

Wypełniacze

Zdarza się, że we wczesnej fazie projektu nie mamy jeszcze do dyspozycji pełnej treści, jaka docelowo znajdzie się na stronie, dlatego też warto jest mieć pewien zasób tzw. "wypełniaczy". W przypadku tekstu jest to używane od wieków *Lorem Ipsum*. Tekst jest pozbawiony sensu, lecz zachowuje równowagę w liczbie liter, dzięki czemu łatwiej zobaczyć wygląd tekstu bez skupiania się na jego treści. Generator tekstu Lorem Ipsum dostępny jest pod adresem www.lipsum.com. Dostępne są też generatory "wypełniaczy" elementów HTML: http://html-ipsum.com/. Aby przetestować wygląd tekstu z polskimi znakami, możemy tez skorzystać z tekstu Pana Tadeusza http://lipsum.pl/index.php. W tytułach dobrze jest wstawić pangramy, czyli krótkie zdania zawierające dużo polskich znaków. Fonty używane w tytułach są często bardziej wymyślne i nie mają wszystkich polskich znaków.

Przejdźmy teraz do zdjęć. Poznaliśmy już serwis pexels, gdzie można znaleźć darmowe zdjęcia. Innymi popularnymi serwisami z darmowymi zdjęciami są m.in. Unsplash oraz stocksnap.io

Często potrzebujemy szybko dodać zdjęcia o konkretnych wymiarach, które można umieścić w layoucie strony zamiast np. bannerów reklamowych. Wówczas, zamiast pobierać zdjęcie i wycinać je do odpowiedniego rozmiaru, możemy skorzystać z kilku serwisów, które zwracają obrazki w dowolnych wymiarach, wykorzystując parametry umieszczane w wywołaniu HTTP.

Korzystamy z nich, dodając poniższy adres do atrybutu "src", jak w przykładzie poniżej:

```
<img src="http://placehold.it/350x150" alt="">
```

W tym wypadku dostaniemy obrazek z serwisu placehold.it o wymiarach 350 na 150 pikseli. Spróbuj otworzyć link w przeglądarce albo dodać go do projektu! Aby nieco urozmaicić swoje wypełniacze, mamy też serwisy zwracające np. zdjęcia:

```
<img src="http://lorempixel.com/300/300/" alt="">
```

Logika działania strony

Po zbudowaniu frontu przychodzi czas na przygotowanie tego, co się dzieje się za kulisami, po stronie serwera. To jest tzw. back-end. Z reguły w przypadku dużych projektów szablon HTML/CSS przekażesz do innej osoby (lub całego zespołu) zajmującej się back-endem. Jednak nie zawsze tak jest. W przypadku małych zleceń, np. strony firmy, bloga lub sklepu internetowego popularnym rozwiązaniem będzie podłączenie front-endu do systemu zarządzania treścią, tzw. CMS-a (Content Management System).

Popularnymi CMS-ami są np. Wordpress, Joomla! lub Drupal. Dzięki nim klienci mogą modyfikować zawartość swojej strony przy pomocy przyjaznego interfejsu bez konieczności każdorazowego zaglądania do kodu, nie mówiąc już o dodaniu możliwości obsługi np. sprzedaży czy dodawania wpisów na blogu. Warto mieć na

uwadze ten etap tworząc front-end. Trzeba upewnić się, że strona będzie działać zgodnie z założeniem w sytuacji, gdy klient będzie modyfikował jej zawartość. Jednak tymi zagadnieniami nie będziemy się zajmować.

Testy i poprawki

Po zbudowaniu strony czas oddać ją klientowi. Mówimy tutaj o tzw. testach akceptacyjnych (UAT, czyli User Acceptance Testing), które w przypadku dużych stron są niejednokrotnie złożonym procesem. Niemal zawsze okazuje się, że klient będzie chciał wnieść poprawki do projektu. O tym etapie też warto pamiętać, gdyż wówczas okazuje się, że musimy stworzoną przez nas stronę zmodyfikować.

Aby uniknąć późniejszej katastrofy, dobrze jest od samego początku zadbać o modułowość naszego kodu: stworzenie jasnych bloków kodu o wyraźnym podziale ze względu na pełnione funkcje, tak aby modyfikacja jednej części nie powodowała awarii reszty :)

Dokładny klient zainteresuje się także sprawdzeniem jak działa strona w uzgodnionych wersjach przeglądarek, więc monitoruj to na bieżąco. Sprawdzanie tylko na samym końcu może spowodować, że stronę trzeba zakodować właściwie od nowa...

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

✔Zapoznałe(a)m się!

3.2. Budowa layoutu

Sugestia

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

Zanim zaczniemy przekuwać projekty graficzne na strony internetowe, musimy przyswoić sobie pewien zasób wiedzy na temat samego projektowania strony oraz dobrych praktyk z tym związanych.

Tworząc stronę od zera, front-endowcy wielokrotnie napotykają te same trudności dotyczące rozmieszczenia treści na stronie oraz ich wyglądu. Z racji tego, że nie są to nowe problemy, istnieją dla nich pewne sprawdzone rozwiązania, czyli tzw. wzorce (patterns). Takim wzorcem może być np. umieszczenie treści w trzech kolumnach, które na urządzeniach mobilnych wyświetlają się jedna pod drugą, czy też klikalne logo, które prowadzi do strony głównej.

Niektórzy przyjmują stanowisko, że na skutek stosowania wzorców wiele stron wygląda obecnie bardzo podobnie, przez co jest coraz mniej miejsca na nowatorskie rozwiązania i eksperymenty. Można powiedzieć, że jest to pewnego rodzaju spektrum, na którego jednym końcu znajdują się sprawdzone i skuteczne, lecz powtarzalne, wzorce, a na drugim znajdziemy układy mniej oczywiste i eksperymentalne. Naszym zadaniem będzie wyważenie obu tych podejść.

Przykłady układów

Poniżej przedstawiamy kilka różnych layoutów stron. Być może część z nich jest Ci już znana. Skupimy się teraz na sposobie, w jaki są one zorganizowane graficznie, by odnaleźć powtarzające się elementy:

- Time Magazine - trzy kolumny
  Strona magazynu Time, pomimo pewnego natłoku treści, ma wyraźny podział na trzy kolumny wraz z menu znajdującym się na górze. Nawiązuje w ten sposób do sposobu, w jaki składane są tradycyjne gazety.
- Dogma - minimalizm dwóch kolumn
  Na przeciwległym biegunie znajduje się minimalistyczna strona agencji Dogma przedstawiająca jej portfolio. Poszczególne projekty tworzą galerię w dwóch kolumnach, a każda część galerii prowadzi do strony zawierającej opis projektu.
- Nice Portfolio - Stały sidebar
  Stały sidebar umieszczony po lewej stronie dokumentu zawiera menu, zaś cała zawartość umieszczona jest po prawej. Warto również przejrzeć strony, które się tam znajdują!
- Avocode 2015 - one/single page layout
  Strony z informacjami pojawiającymi się w miarę ich przewijania stały się hitem i można uznać je za standard wśród nowoczesnych layoutów. W takich stronach li zazwyczaj nie przekierowują nas na nowe podstrony, tylko na sekcje w obrębie tej samej strony. Często też w miarę przewijania wyświetlane są użytkownikowi

animacje. W powyższym przykładzie technika ta znalazła zastosowanie do stworzenia animowanej infografiki przedstawiającej trendy w projektowaniu graficznym na potrzeby Internetu.

Dobre praktyki komponowania layoutu

Co do zasady, **front-endowiec nie jest projektantem**, lecz pewna fundamentalna wiedza z zakresu projektowania graficznego jest przydatna, jeśli nie niezbędna. Dzięki niej możemy lepiej porozumieć się z grafikiem, a także samemu rozpoznać dobre (lub niekoniecznie dobre) rozwiązania i wykorzystać tę wiedzę do sprawniejszego budowania strony. Poniżej znajduje się lista dobrych praktyk związanych z tworzeniem layoutu strony od podstaw.

KISS - Keep it simple, stupid!

Jedną z najważniejszych zasad, które powinniśmy sobie przyswoić, jest dążenie do prostoty. Skomplikowany interfejs jest trudny w użytkowaniu i zniechęca użytkowników (i klientów!) do korzystania z naszej strony. Ponadto prostszy HTML i CSS powoduje, że strona wczytuje się w przeglądarce szybciej, nie skazując użytkowników na czekanie.

Korzystaj ze znanych wzorców.

Omówiliśmy już zalety korzystania ze sprawdzonych wzorców. To, że znajdują się one na wielu stronach, wynika z tego, że przetrwały one próbę czasu. Możemy założyć, że jeżeli coś sprawdza się na stronie z milionami użytkowników, sprawdzi się też na mniejszą skalę.

Ustal wizualną hierarchię.

Istotną sprawą jest to, aby uwaga użytkownika była skupiona na tym, co jest najważniejsze. W kontekście layoutu ma to znaczenie zwłaszcza przez ustalenie kolejności i rozmiarów poszczególnych treści, ich rozmiarów, kolorów, odstępów pomiędzy elementami.

Nie każ użytkownikom myśleć.

Jest to założenie podobne do pierwszego, jednak nie do końca. Twój layout powinien być zrozumiały dla użytkownika, czyli funkcja poszczególnych jego elementów powinna być jasna od momentu ich ujrzenia. Ponadto użytkownik powinien być w stanie znaleźć bez problemu potrzebne mu funkcje i treści.

Zadbaj o czytelność tekstu.

Twój tekst powinien mieć dostatecznie duży rozmiar zapewniający czytelność na każdym ekranie. Ponadto interlinie oraz odstępy między literami powinny być dostatecznie duże, by tekst oraz litery nie najeżdżały na siebie.

Spraw by Twój tekst oddychał! Nie żałuj odstępów i marginesów, by zapewnić każdemu elementowi odpowiednią przestrzeń. Pamiętaj też o odpowiednim kontraście pomiędzy tłem a tekstem. Projektując stronę, trzeba pamiętać o starszych użytkownikach ze słabszym wzrokiem. Słabszy kontrast ogranicza czytelność tekstu przy wyświetlaniu na mniejszych ekranach. Także wtedy gdy na wyświetlacz świeci słońce lub gdy tekst wyświetlany jest na starszym monitorze.

Konsorcjum W3C ustanowiło pewne minimalne standardy w tej kwestii: minimalny kontrast pomiędzy tekstem a tłem powinien wynosić 4,5 : 1.

Możemy sprawdzić kontrast np. za pomocą tego narzędzia.

Właściwości rozmieszczenia

Layout strony możemy kształtować, nadając elementom HTML naszej strony odpowiednie właściwości w pliku CSS, które ustalą pozycję naszej treści na stronie. Do takich stylów, które determinują układ strony, należą m.in. następujące właściwości CSS:

- **Display** - sposób wyświetlenia elementu HTML.
- **Position** - pozycja elementu HTML.
- **Float** - opływanie elementu HTML przez inne elementy.

Poprzez przypisanie odpowiednich wartości tym właściwościom będziemy mogli precyzyjnie określić, w którym miejscu ma znaleźć się dany element oraz w jakiej relacji ma on pozostawać względem pozostałych.

Ze względu na mnogość urządzeń, na których obecnie wyświetlane są strony internetowe, zapewnienie odpowiedniego wyświetlania nabiera kolosalnego znaczenia. Nowoczesnym sposobem na rozwiązanie tego problemu jest tzw. Flexbox, z którym zapoznasz się w kolejnym module.

Czym jest grid

Znamy już zasady tworzenia layoutu, teraz dowiemy się, jak zbudować za pomocą poznanych wcześniej właściwości przydatne narzędzie w jego budowie. Mowa o tzw. **gridzie**, czyli siatce wierszy i kolumn, wewnątrz której będziemy mogli rozmieszczać elementy na stronie. Jest to sprawdzone rozwiązanie mające wielu zwolenników. Wiele stron jest zbudowanych w oparciu o siatkę 12 **kolumn**, wewnątrz których możemy umiejscowić nasze elementy, tak jak na ilustracji poniżej:

Czemu akurat 12? Taka liczba kolumn daje dużą swobodę w rozmieszczaniu: możemy ustalić, że element po lewej będzie zajmować 1/3 widoku, a pozostała część 2/3, definiując szerokość jednej jako 8 kolumn, a drugiej jako 4. Dwunastokolumnowy layout dzieli się na różne sposoby: 2, 3, 4 itp.

Jak się to stosuje? Stworzymy klasy `col-X`, gdzie X to będzie szerokość, jaką chcemy nadać interesującemu nas elementowi. Na przykład, nadając elementowi klasę `col-3`, sprawimy, że będzie on zajmował szerokość trzech kolumn, tj. 3/12, czyli 25% szerokości (tak, jak niebieskie elementy w przykładzie powyżej), `.col-4` - czterech itd.

Proces budowy własnego gridu

Poniżej omówimy dokładnie sposób tworzenia własnego gridu. Nie musisz wykonywać go teraz. Na końcu submodułu będą zadania, w których to zrobisz.

Zacznijmy od ustalenia wartości właściwości `box-sizing` na `border-box` dla wszystkich elementów. W ten sposób do wymiarów poszczególnych elementów doliczane będą ich dopełnienia (`padding`) oraz obramowanie (`border`). Robimy to za pomocą uniwersalnego selektora:

```
* {
    box-sizing: border-box;
}
```

Kolejnym krokiem będzie zdefiniowanie wiersza (row).

Nasza treść będzie w ten sposób zorganizowana w poziome wiersze, które następnie będą zawierać kolumny. Dzięki temu możemy umieścić kilka elementów w kolumnach jeden pod drugim i upewnimy się, że nie będą ze sobą kolidować.

Jako że kolumny będą opierały się na floatach, musimy mieć pewność, że nie będą uciekały ze swoich wierszy. Wykorzystamy w tym celu tzw. clearfix hack. Jest to pewnego rodzaju trik, który zapewni prawidłowe wyświetlanie zawartości. Należy do tego stworzyć klasę `row` wraz z pseudoelementami `::before` oraz `::after`, a następnie dodać tam następujący kod:

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

Rozwiązanie takie uchodzi za niezbyt eleganckie z tego względu, że jest to trik wykorzystujący właściwości wyświetlania tabeli. Jednak osiągamy w ten sposób nasz cel i clearfix przyjął się jako pewnego rodzaju obejście problemu.

W odpowiedzi na potrzebę prawidłowego wyświetlania elementów na stronach powstał flexbox (o którym więcej napiszemy w dalszej części kursu), jednak z racji tego, że nie jest on wspierany przez stare przeglądarki wciąż mające duży udział w rynku, warto poznać i takie rozwiązania, jak clearfix. Zasady jego działania są dość skomplikowane, jednak nie musimy się na razie w nie zagłębiać. Istotne jest, że robi to, czego od niego oczekujemy.

Istnieje także możliwość zagnieżdżania w sobie kolumn/wierszy. Wówczas szerokość elementu będzie procentową wartością szerokości elementu rodzica, a nie całej strony. Pamiętaj, że jeśli chcemy zagnieździć kilka kolumn w jednej, nigdy nie robimy tego bezpośrednio, tylko w kolumnie umieszczamy wiersz, w którym tworzymy kolumny. Struktura HTML wygląda więc tak:

```
<div class="col-8">
    <div class="row">
        <div class="col-4">A column with a width of 1/3 of the parent
        <div class="col-4">A column with a width of 1/3 of the parent
        <div class="col-2">A column with a width of 1/6 of the parent
        <div class="col-2">A column with a width of 1/6 of the parent
    </div>
</div>
```

Tą samą zasadą kierujemy się, zagnieżdżając w sobie wiersze - umieszczamy wiersz, w nim kolumnę, a dopiero wtedy kolejne wiersze.

Budowa kolumn

Przejdźmy do stworzenia stylów odpowiedzialnych za poszczególne kolumny naszego layoutu. Jest to najbardziej skomplikowane zagadnienie ze względu na to, że istnieje co najmniej kilka metod ich pozycjonowania, a ponadto trzeba wziąć pod uwagę wyświetlanie na różnych szerokościach ekranu. Teraz skupimy się na podstawowym gridzie, zaś aspekt responsywności poznamy lepiej w module temu poświęconym.

Chcemy, aby wszystkie elementy znajdowały się w jednym rzędzie, niezależnie od tego, ile zajmują miejsca. Dlatego też dodamy do wszystkich klas `.col` właściwość `float: left`, oraz minimalną wysokość. Aby przypisać to do wszystkich klas skorzystamy tutaj z selektora atrybutu.

Pozwala on nam wybrać wszystkie elementy HTML, które posiadają dany atrybut lub jego określoną wartość. Przykładowo, stosując `a[target]`, wybieramy wszystkie elementy `a` posiadające atrybut `[target]` o dowolnej wartości.

Jeżeli dodamy tam sformułowanie `a[target="_blank"]`, wybierzemy tylko te linki, które otwierają się w nowej karcie (mają atrybut `target="_blank"`). Dodając gwiazdkę `*` przed znakiem równości, wybierzemy wszystkie te elementy, które mają dany atrybut, a jego wartość zawiera w sobie określony ciąg znaków. W przykładzie poniżej wybieramy wszystkie elementy, które w nazwie klasy mają `col-`:

```
[class*='col-'] {
    float: left;
    Min-height: 1px;
}
```

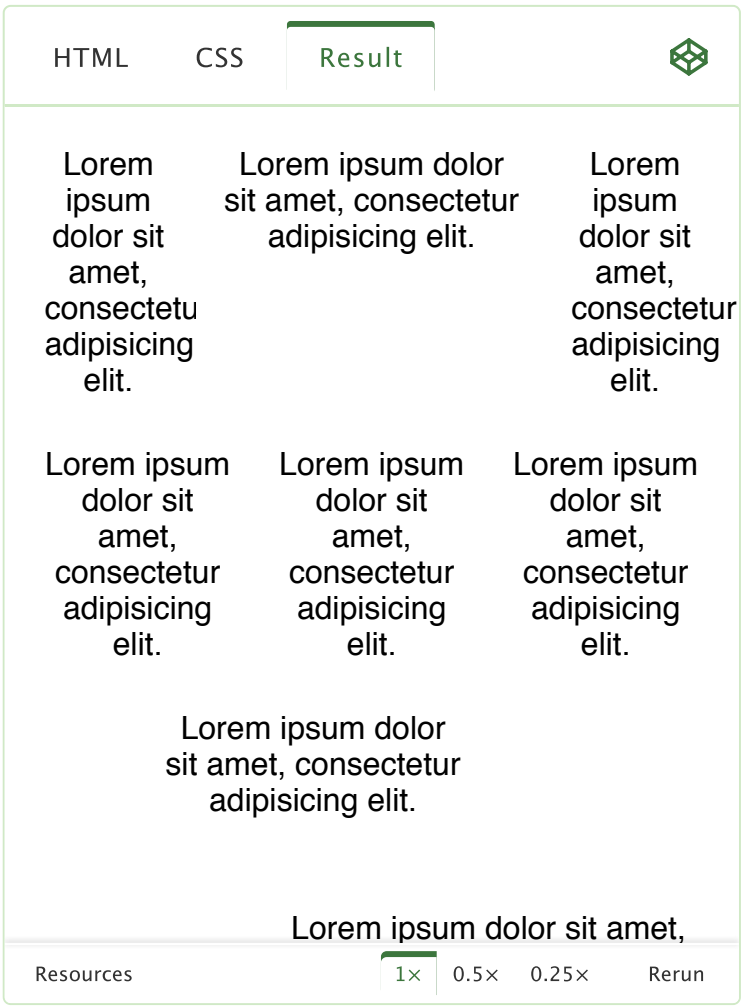
Następnie przypiszemy do poszczególnych wymiarów kolumn odpowiednie szerokości. Jeżeli mamy 12 kolumn, zaś nasz kontener zajmuje 100% szerokości, to pojedyncza kolumna będzie mieć 100 / 12 = 8.33% szerokości. Natomiast jeżeli chcemy, aby element rozciągał się np. na dwie kolumny, mnożymy 8,33% * 2 = 16,66% itd. Dlatego też teraz przypiszemy do każdej klasy odpowiednią szerokość:

```
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}
```

Oczywiście zawsze można dodać dodatkową klasę `col` i jej przypisać wszystkie wspólne właściwości. Wówczas wystarczy ją dodać do kolumny w ten sposób:

```
<div class="col col-2"></div>
```

Nasz grid jest gotowy! Teraz możemy przypisywać naszym elementom szerokość wyrażoną w liczbie kolumn, jak w przykładzie poniżej:



Co w sytuacji, gdy chcemy przesunąć elementy, np. żeby miały margines po lewej stronie równy dwóm kolumnom? Wystarczy w tym celu stworzyć nową grupę klas i zamiast szerokości `width` przypisać im te same wartości właściwości `margin-left`. Omijamy naturalnie przesunięcie o 12 kolumn, gdyż wówczas element będzie usunięty poza widok strony. Dodajemy też jedną klasę .col-offset-0, by móc wyzerować margines z lewej strony.

```
.col-offset-0 {margin-left: 0;}
.col-offset-1 {margin-left: 8.33%;}
.col-offset-2 {margin-left: 16.66%;}
.col-offset-3 {margin-left: 25%;}
.col-offset-4 {margin-left: 33.33%;}
.col-offset-5 {margin-left: 41.66%;}
.col-offset-6 {margin-left: 50%;}
.col-offset-7 {margin-left: 58.33%;}
.col-offset-8 {margin-left: 66.66%;}
.col-offset-9 {margin-left: 75%;}
.col-offset-10 {margin-left: 83.33%;}
.col-offset-11 {margin-left: 91.66%;}
```

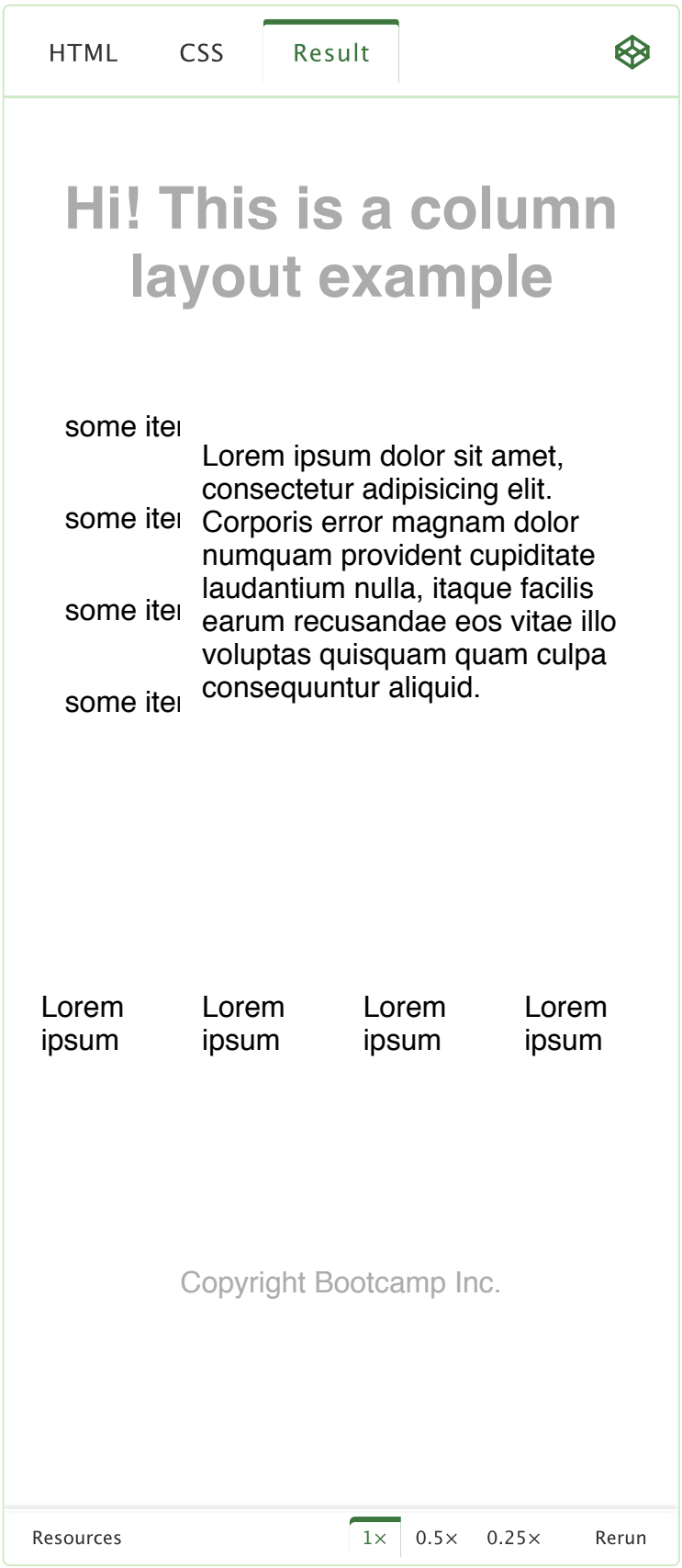
Teraz możemy nie tylko ustalać szerokość elementów, ale także ich rozmieszczenie w poziomie. W ten sposób za pomocą przypisywania klas możemy precyzyjnie określić położenie treści na stronie. Ze względu na swoją prostotę, takie rozwiązanie stało się bardzo popularne i można je zaobserwować na wielu stronach.

Aby określić odstępy pomiędzy zawartością kolumn możemy zdefiniować właściwość `padding`. W ten sposób zawartość naszych kolumn nie będzie przylegać do ich krawędzi, co jest oczywiście pożądane. Przychodzi nam tutaj z pomocą stworzony już przez nas selektor atrybutu:

```
[class*="col-"]{
    float: left;
    min-height: 1px;
    padding: 12px;
}
```

Grid w praktyce

Kiedy zadbaliśmy o prawidłowe działanie naszych kolumn, możemy stworzone przez nas klasy zastosować w praktyce do opracowania całego layoutu strony. Spójrz na przykład znajdujący się poniżej:



Wykorzystaliśmy tam napisane przez nas klasy do poukładania layoutu strony. Najpierw stworzyliśmy jeden `<div>` z klasą `container`, wewnątrz którego będzie znajdować się cała treść. W ten sposób wyśrodkowaliśmy zawartość i stworzyliśmy

marginesy po lewej i prawej stronie.

Teraz możemy dodawać treść. Nagłówek powinien zajmować całą szerokość strony, więc stworzyliśmy `<div>` z klasą `row`, a samemu nagłówkowi nadaliśmy klasę `col-12` - to nadaje mu szerokość 100%.

Poniżej stworzyliśmy dwie sekcje: po lewej i po prawej stronie. Jedna z nich zajmuje ¼ szerokości, a druga ¾ . Aby osiągnąć ten efekt nadaliśmy odpowiednio klasy `col-3` oraz `col-9` dla poszczególnych sekcji.

Poniżej widać jeszcze inne rozwiązanie, również często stosowane na wielu stronach w sekcjach przedstawiających np. poszczególne usługi. W tym wypadku zastosowaliśmy cztery elementy z klasą `col-3`. Poniżej tej sekcji znajduje się stopka - tutaj analogicznie do nagłówka nadaliśmy jej klasę `col-12`.

Na marginesie warto też powiedzieć, że stworzyliśmy tutaj zestaw stylów, których możemy używać wielokrotnie (tzw. reusable styles). Jest to kolejny dobry zwyczaj, który warto praktykować, tworząc kod CSS. W sytuacji gdy pracujemy nad złożonym projektem, możemy nasz układ kolumnowy zastosować w wielu miejscach.

I to nie tylko w jednym projekcie! Dobry front-endowiec wraz z czasem i doświadczeniem tworzy swoją kolekcję fragmentów kodu, które możemy wykorzystywać ponownie, są to tzw. snippets.

Front-endowcy często dzielą się swoimi rozwiązaniami, więc polecamy szukanie snippetów w Internecie, jak i umieszczanie tam swoich rozwiązań, by ułatwić pracę innym. Przykładowe kolekcje znajdują się np. na stronie css-tricks. Niezastąpiona jest też biblioteka wzorców na znanym nam już serwisie Codepen.

Podsumowując:

Ogólnie staraj się trzymać następujący układ:

```
<!-- HTML5 tag with our class, for example features, about, jumbotron
<section class="about">
    <div class="container">
        <!-- clearfix - row must be the parent of the columns! -->
        <div class="row">
                    <div class="col-6"></div>
                    <div class="col-6"></div>
            </div>
    </div>
</section>
```

Zadanie: Budujemy grid

1. W pliku *style.css* nadaj wszystkim elementom `box-sizing: border-box`.
2. Następnie stwórz klasę dla kontenera (np. `container`) i nadaj jej szerokość 100%, dodatkowo maksymalną szerokość w pikselach (np. `900px`), oraz wyśrodkowanie za pomocą `margin: 0 auto;`.
3. Stwórz następnie klasę `row`, na której zastosujemy `clearfix`. Utwórz selektor wybierający równocześnie pseudolementy `row::before` oraz `row::after` i ustaw dla nich poniższe style:

```
content: "";
display: table;
clear: both;
```

4. Teraz zajmijmy się stworzeniem wspólnych właściwości dla wszystkich kolumn. Skorzystaj z selektora atrybutu `[class*="col-"]` i dodaj do niego własności `float: left;`, `min-height: 1px;` oraz `padding` (wartość ustal samodzielnie).
5. Czas stworzyć kolumny. Stwórz klasy `col-X`, wstawiając w miejsce X liczby od 1 do 12, oraz dodaj do nich właściwość `width` wraz z kolejnymi wartościami:
    - **1** - 8.33%,
    - **2** - 16.66%,
    - **3** - 25%,
    - **4** - 33.33%,
    - **5** - 41.66%,
    - **6** - 50%,

- **7** - 58.33%,
- **8** - 66.66%,
- **9** - 75%,
- **10** - 83.33%,
- **11** - 91.66%,
- **12** - 100%.

6. Dodajmy teraz możliwość przesuwania kolumn. Stwórz klasy `col-offset-X`, dodając jako X - podobnie jak w poprzednim poleceniu - liczby od 1 do 11, oraz przypisz im wartości `margin-left` odpowiadające szerokościom kolumn.
7. Czas wykorzystać nasz grid! Przechodzimy do pliku *index.html*. Dodaj jeden element `<div>` i przypisz mu stworzoną już klasę `container`. Wewnątrz kontenera dodaj element `<h1>` z dowolnym tekstem.
8. Poniżej nagłówka dodaj przynajmniej dwa rzędy za pomocą klasy `row`. Stwórz w każdym z nich kilka elementów `<div>` oraz przypisz im klasy `col-X`. Mogą one mieć dowolne wartości oraz zawartość. Poeksperymentuj z różnymi szerokościami!
9. Po dodaniu layoutu naszej strony na zakończenie wstaw stopkę i również skorzystaj ze stworzonych klas, by miała ona szerokość 100%.

Najważniejsze w tym zadaniu jest to żeby wiedzieć jak skonstruowany jest grid i jak można go zrobić samodzielnie. Polecamy wypełnienie kolumn za pomocą *Lorem Ipsum* i eksperymentowanie ze zmianą szerokości kolumn poprzez zmianę klas określających ich szerokość :)

Podgląd zadania

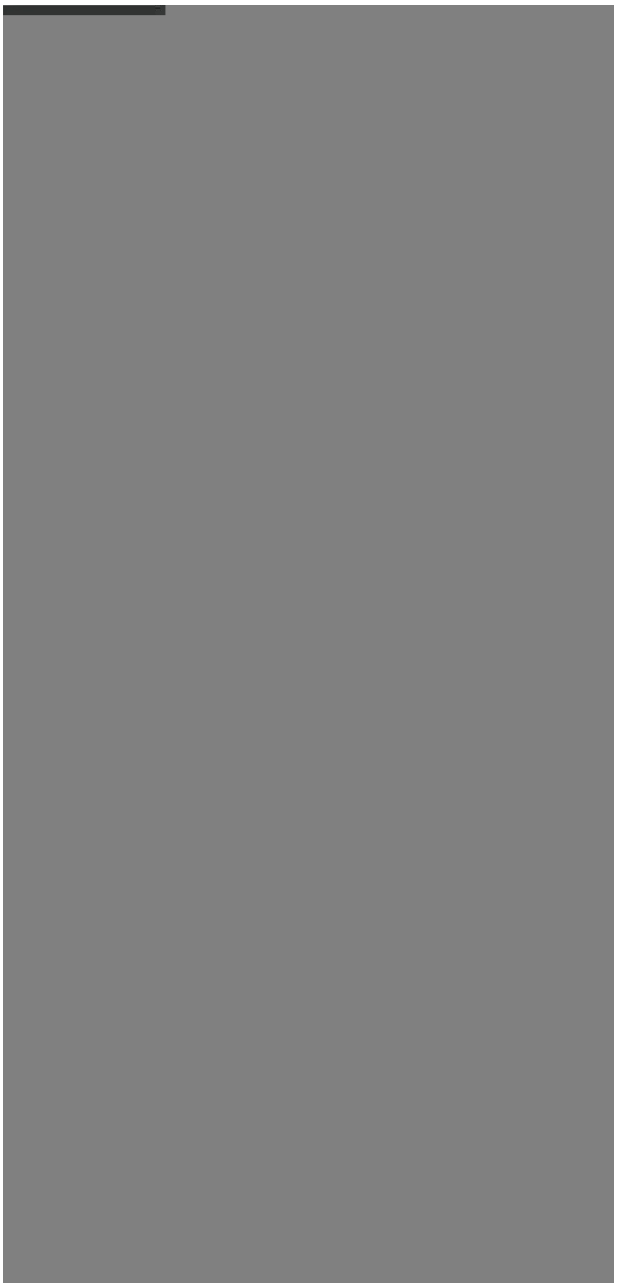
Przejdź do projektu ✔

3.3. Tniemy projekt na zlecenie!

Tniemy projekt na zlecenie!

Wielu programistów demonizuje pracę z klientami, powołując się na ich brak zdecydowania, nierealne wymagania, ciągłe zmiany założeń, nieznajomość rynku czy technologii, trudności w kontakcie. Tak wcale nie musi być! Współpraca powinna przebiegać na zasadzie partnerstwa. Pamiętaj, że jeśli klient uzna, że dobrze wykonujesz swoją pracę, jest duże prawdopodobieństwo, że wróci do Ciebie z kolejnymi projektami, a może nawet poleci Cię swoim znajomym. Poczta pantoflowa jest Twoim wielkim sprzymierzeńcem w szukaniu nowych zleceń :)

W tym submodule przedstawimy proces tworzenia strony w kontekście konkretnego zlecenia. Poniżej znajduje się link do pliku z projektem graficznym strony, nad którą będziemy pracować:



Wymagania klienta

Niezależnie, czy samodzielnie szukasz zleceń i kontaktujesz się z klientem jako freelancer, czy też pracujesz w firmie tworzącej strony internetowe i masz nad sobą managera projektu, który się tym zajmuje, warto trzymać się pewnych zasad dotyczących współpracy z klientem (z którym łącznikiem jest w firmie manager).

Zazwyczaj wszystko zaczyna się od projektu graficznego.

Następnie klient przekazuje wymagania dotyczące realizacji projektu - układ stron, oczekiwane interakcje między stroną a użytkownikiem, jak również kwestie formalne, takie jak oczekiwany termin wykonania zadania. Klient zawsze opowiada o tym, czego on potrzebuje i co go najbardziej obchodzi. W naszym interesie jest, żeby dopytać o szczegóły techniczne typu urządzenia mobilne lub co ma się dziać w konkretnych sytuacjach.

Jeśli ustalanie wymagań prowadzone jest drogą telefoniczną lub na żywo, spisuj wszystkie ustalenia i po rozmowie wyślij klientowi e-maila z podsumowaniem i zapytaniem, czy dobrze rozumiesz jego oczekiwania.

Ustalenie założeń

Gdy znasz już wymagania klienta, zaproponuj mu stawkę, jakiej oczekujesz za wykonanie projektu. Zapoznaj się z dostarczonym plikiem przedstawiającym layout. Zastanów się, czy wszystkie jego elementy są możliwe do wykonania. Czy wykonalne jest zakodowanie ich za pomocą CSS? Miej na uwadze, że klient zazwyczaj dysponuje niewielką wiedzą z zakresu programowania, więc może nie zdawać sobie sprawy z wielu potencjalnych problemów wynikających z jego wymagań - Twoim zadaniem jest poinformować go o tym i zaproponować odpowiednie rozwiązania.

Rozpoczęcie prac

Gdy poznamy już wymagania klienta i ustalimy z nim, jakie technologie zastosujemy w projekcie, najwyższy czas na najprzyjemniejszą stronę wykonywania zlecenia: **programowanie**.

To trzeci tydzień naszego bootcampa, dlatego zwiększamy poziom trudności i nasze polecenia będą wymagać od Ciebie więcej samodzielnej pracy niż dotychczas. Daje Ci też większą dowolność - sam wybierz tematykę projektu oraz tekst i grafiki, których

użyjesz na stronie.

W ramach przygotowań przed kodowaniem musimy najpierw przeanalizować układ strony i rozkład elementów. Patrząc na projekt, który otrzymaliśmy od klienta, dzielimy go na sekcje i elementy.

Wymieńmy sobie główne elementy oraz sekcje strony w kolejności, w jakiej występują:

1. nagłówek
2. sekcja powitalna,
3. sekcja z trzema elementami w jednej linii,
4. sekcja - nagłówek z tekstem,
5. sekcja z galerią,
6. sekcja z czterema elementami w jednej linii, w których mamy pracowników klienta,
7. sekcja promocyjna - złożona ze zdjęcia, nagłówka i tekstu,
8. sekcja z serią liczb w jednej linii opisujących firmę,
9. stopka z dodatkowymi informacjami.

Główna struktura strony sama rzuca się w oczy. Niektóre sekcje mają 100% szerokości, inne mają stałą szerokość, zachowując symetrię rozłożenia elementów.

Gdy rozrysujemy sobie siatkę strony, będzie ona wyglądała następująco:

Na czarno zaznaczyliśmy podział na sekcje, a żółtym kolorem odpowiednie kontenery, których użyjemy w celu zachowania symetrii rozłożenia elementów. W większości przypadków sekcje nie mają tła, więc będziemy mogli użyć na nich klasy pomocniczej. Jednak niektóre sekcje mają tło, dlatego w ich przypadku stworzymy dodatkowy element wewnątrz każdej z sekcji, który będzie naszym kontenerem.

Nagłówek

Struktura

Do stworzenia nagłówka użyjemy tagu <header>. Zgodnie z projektem ma mieć ciemny kolor tła i być rozciągnięty na całą szerokość. Jednak żeby zachować symetrię strony, musimy umieścić jego zawartość w kontenerze. Wykorzystamy do tego tag <nav>, któremu przypiszemy klasę container. Będzie to nasza klasa pomocnicza, kt uporządkuje treść na stronie. Wewnątrz niego umieścimy element <img>, za pomocą

którego wyświetlimy logo klienta. Zaraz za nim umieścimy naszą nawigację w postaci listy linków do innych podstron. Użyjemy do tego tagu `<ul>`, w którego trzech elementach (`<li>`) umieścimy linki, korzystając z elementu `<a>`.

Struktura nagłówka strony będzie wyglądać następująco:

```
<header>
    <nav class="container">
        <img src="logo.png">
        <ul>
            <li><a href="#">Link</a></li>
            <li><a href="#">Link</a></li>
            <li><a href="#">Link</a></li>
        </ul>
    </nav>
</header>
```

Wygląd

Zajmijmy się wyglądem nagłówka. Załóżmy, że nie otrzymaliśmy od grafika schematu kolorów dla górnej belki. Jak sobie poradzić z tym problemem? Możemy pobrać je sami z przesłanej grafiki. Jeżeli otworzysz ją w swojej przeglądarce, możesz użyć jednego z wielu colorpickerów dostępnych jako rozszerzenia dla przeglądarek, np. wtyczki do Google Chrome. Instalacja i obsługa jest banalnie prosta. Dzięki temu narzędziu dowiadujemy się, że kolor tła nagłówka to `#313234` i właśnie tego koloru użyjemy. Przy okazji nadajemy mu jeszcze wysokość `50px`.

Style globalne

Da się także zauważyć, że zawartość naszej strony jest nieco ściśnięta (odstaje od krawędzi okna przeglądarki). Wynika to z marginesu, jaki domyślnie ma nadany element `<body>`, trzeba więc będzie go wyzerować, korzystając z odpowiedniego selektora CSS.

Zajmiemy się teraz klasą pomocniczą `container`, którą mamy przypisaną do `<nav>`, jeszcze nie raz nam się przyda, ustalając jej zachowanie, uporządkujemy praktycznie całą stronę. Jest to klasa służąca do zarządzania rozmieszczeniem, nie nadajemy jej żadnych cech wizualnych typu kolor tła czy ramki. Jedyne co zrobimy, to nadanie odpowiedniej bazowej szerokości (`100%`) oraz ograniczenie maksymalnej szerokości - standardowa dla monitorów to `1200px` (tutaj musimy użyć zarówno właściwości `width` jak i `max-width`) oraz wyśrodkowanie za pomocą `margin: 0 auto;`.

Nawigacja

Zanim zajmiemy się logotypem, przesuniemy nawigację na prawo i przy okazji uporządkujemy jej elementy. Może się zdarzyć, że na stronie będzie kilka elementów `<ul>`, więc żeby nie dodawać kolejnej - zbędnej klasy - użyjemy selektora `header>nav ul`. Jest on bardzo bezpiecznym rozwiązaniem.

Naturalnie nadajemy naszemu `<ul>` margines, `padding` równy 0, `list-style-type: none;` (żeby pozbyć się domyślnych wartości) oraz właściwość `float: right;`. I tu pojawia się niespodzianka. Nasza lista nagle ucieka ze swojego kontenera. Uporamy się z tym, używając omawianej już nie raz metody `clearfix` na naszym `nav`. Wujek google pomoże :)

Linki

Ostatnim elementem nawigacji są odnośniki, czyli nasze elementy `<a>` wewnątrz `<li>`. Kopiujemy i rozbudowujemy ostatnio użyty selektor do postaci `header>nav ul>li>a`. Dobrą praktyką jest tworzenie takich przycisków w sposób, który sprawi, że naturalnie rozsuną one elementy listy `<li>`. Na tych przyciskach będą różne efekty typu `:hover`, więc żeby to przyzwoicie wyglądało, to właśnie im nadamy odpowiednie właściwości.

Żeby mieć absolutną pewność, że tekst odnośnika zawsze będzie na środku, nadajemy mu wysokość linii równą wysokości nagłówka, czyli 50px. Żeby przyciski do siebie nie przylegały, nadajemy im też `padding` o wartości `0 20px` (potrzebujemy tylko bocznych paddingów). I teraz — żeby zachowywały się właściwie — dodajemy jeszcze wyświetlanie blokowe.

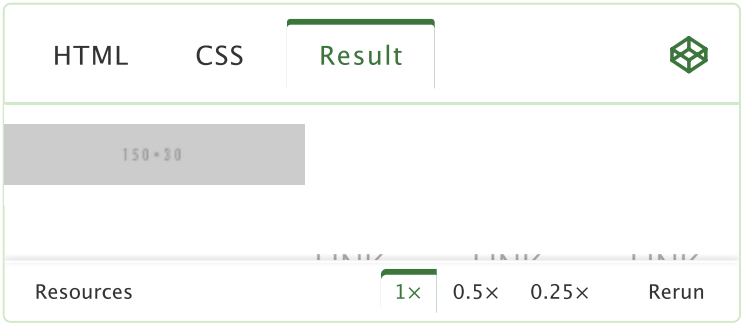
To jeszcze nie koniec właściwości dla odnośników. Wypadałoby zrobić coś z podkreśleniem i kolorem. Naturalnym rozwiązaniem będzie `color: #fff;` (już pewnie pamiętasz, że to kolor biały), `text-decoration: none;` (pozbywamy się podkreślenia) oraz `text-transform: uppercase;`. Warto dodać jakiś efekt po najechaniu myszką (hover). Nie dostaliśmy wytycznych w tym zakresie, więc intuicyjnie używamy tego samego koloru, który ma `header`, tyle że delikatnie rozjaśnionego.

Zostaje nam czcionka linków. Domyślna nie jest zbyt piękna. Możemy śmiało zaimportować Open Sans i podpiąć ją w stylach dla body.

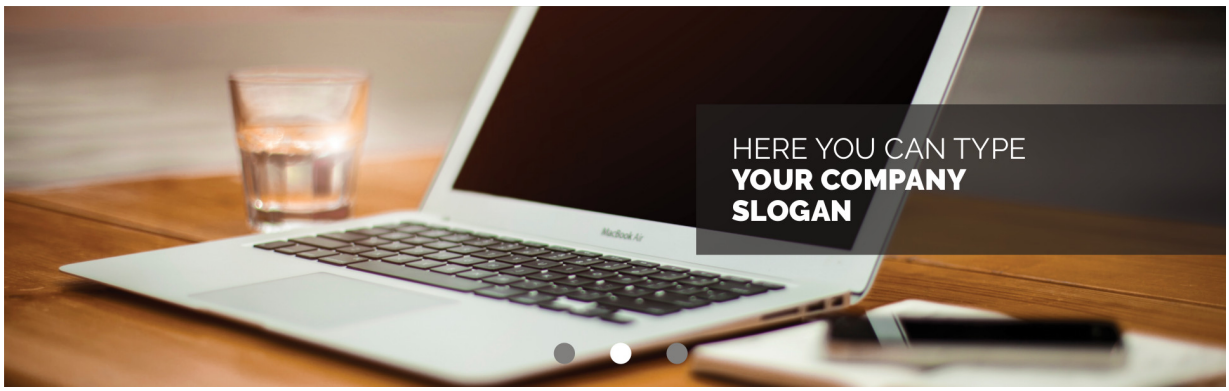
Logo

Przyszedł czas na logo. Zakładamy, że jeszcze go nie dostaliśmy od grafika lub klienta. W takim przypadku możemy wstawić jakiś tekst lub użyć wypełniacza. Żeby nie modyfikować później nawigacji, użyjemy podlinkowanego wyżej wypełniacza o rozmiarach 150 na 30 pikseli. Dodajmy odpowiedni znacznik (`<img>`) z `src="//placehold.it/150x30"`. Zdjęcie przykleja się do górnej granicy. Żeby było w jednej linii, tworzymy regułę z selektorem `header>nav img` w której przypisujemy grafice `padding: 10px 0;`

Efekt dotychczasowych prac:



Sekcja powitalna



W tej sekcji umieścimy tzw. hero (w Internecie na pewno spotkasz się także ze stosowaną zamiennie nazwą jumbotron).

Utwórz sekcję (`section`) o klasie `hero`, a następnie, korzystając z darmowych serwisów (np. www.unsplash.com), znajdź zdjęcie, które będzie odgrywało rolę elementu przyciągającego wzrok.

Kolejnym krokiem jest zagnieżdżenie wewnątrz elementu `div` o klasie `slogan`. Będzie on zawierał slogan firmy. Nadaj mu półprzezroczyste tło (rgba), `float: right`, a także ustal maksymalną szerokość (np. na 800 pikseli). Wewnątrz tego elementu umieść nagłówek `<h1>` z dowolnym tekstem.

Im bardziej upodobnimy ten element do projektu, który dostaliśmy, tym lepiej.

Sekcja korzyści

Tutaj przyda nam się wcześniej utworzony grid. Możesz go śmiało skopiować do tego projektu.

Co my tu mamy? Oczywiście sekcję, w której znajduje się dodatkowo kontener, żeby elementy były w odpowiednim miejscu. A wewnątrz tego kontenera trzy kolumny.

Aż się prosi, żeby użyć naszego gridu. Wstawiamy więc wiersz (row), a w nim trzy divy z odpowiednimi klasami.

Każda kolumna zawiera ikonę, mały nagłówek oraz trochę tekstu. Ikon możemy użyć z zestawu, jaki daje nam do dyspozycji Font Awesome. W poprzednich projektach już je wykorzystywaliśmy. Należy je wstawić i nadać im odpowiedni rozmiar.

Do tekstów możemy użyć dowolnych wypełniaczy, które były przedstawione wcześniej w tym module.

Wszystkie sekcje z białym tłem mają pewną przestrzeń między elementami zawartymi w środku a sąsiednimi sekcjami. Dodatkowo wszystkie takie sekcje mają w sobie kontenery umieszczające zawartość w jednej kolumnie. Możemy wykorzystać ten schemat w konstrukcji, by dodać w sprytny sposób powietrza w sekcjach, które mają tę charakterystykę.

Korzystając z tej własności, tworzymy selektor przeznaczony właśnie dla takich przypadków `section>.container` i do niego przypiszemy styl `margin: 50px auto;`. Dzięki temu wszystkie kontenery w section będą miały margines górny i dolny, ale i tak będą wyśrodkowane.

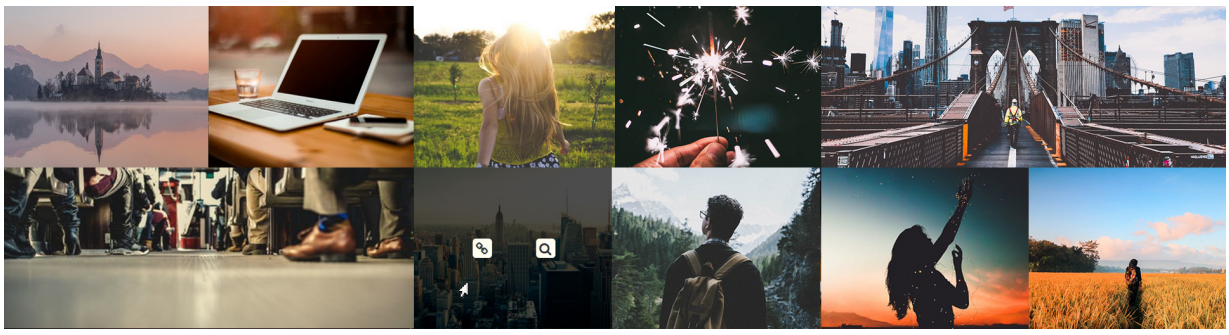
Gdy się uważniej przyjrzymy, to zauważymy, że tego typu sekcje też mają zawsze wyśrodkowany tekst. Wynika z tego, że do tego samego selektora możemy śmiało dopisać wyśrodkowanie tekstu.

Pod sekcją z ikonami znajduje się również sekcja z nagłówkiem i tekstem. Biorąc pod uwagę, że stworzyliśmy kilka globalnych stylów, wykonanie takiej sekcji powinno zaj mniej niż 5 minut :).

Galeria

Przyjrzyjmy się teraz galerii. Wykonaliśmy podobny komponent w poprzednich modułach, więc możemy skorzystać z naszego fragmentu kodu, by zbudować galerię. Zajmuje ona 100% strony. I znowu aż się prosi, żeby użyć naszego gridu, prawda?

Na grafice poniżej zobaczysz jeden element galerii przedstawiający, jak powinien on wyglądać przy najechaniu na niego kursorem. Znajdują się tam dwie ikony.



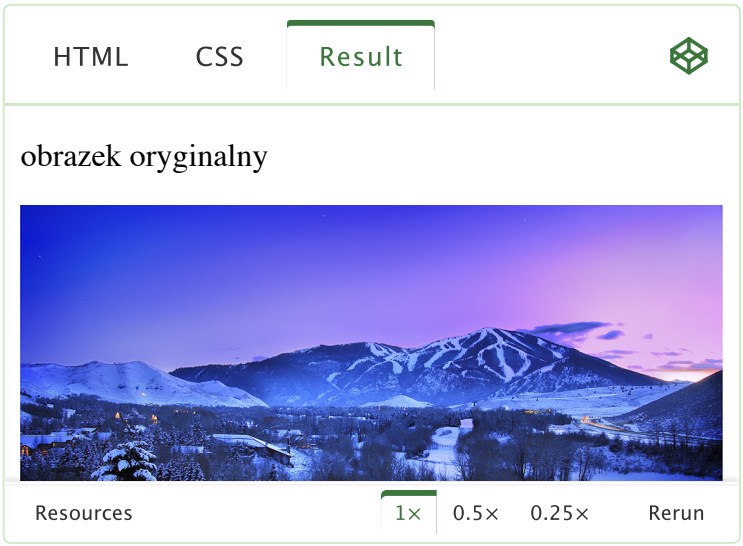
Moglibyśmy posłużyć się do tego celu pseudoelementami `::before` oraz `::after`, jest jednak z tym pewien problem: nie działają one w przypadku elementu `<img>`.

Możemy to rozwiązać, modyfikując nieco nasz markup: każdy element galerii będzie kolumną z gridu, wewnątrz niej znajdować się będą ikony pojawiające się dopiero przy hoverze (podobnie jak w przypadku dwupoziomowego menu) oraz właściwa grafika. Same ikony też będą w elemencie, który będzie swoistą warstwą z przezroczystym tłem. Ambitnym polecamy pokombinować z wysuwaniem się warstwy przy wykorzystaniu `position: absolute;` oraz `transition`.

Przykładowa struktura HTML takiego elementu galerii będzie następująca:

```
<div class="col-X">
    <div class="layer">
        <i class="fa fa-camera-retro fa-lg"></i>
        <i class="fa fa-search fa-lg"></i>
    </div>
    <img src="https://placekitten.com/200/200" alt="">
</div>
```

Aby nasze zdjęcie poprawnie wypeniało całą przestrzeń komórki gridu, musimy je odpowiednio przeskalować. W tym celu użyjmy właściwości object-fit i nadajmy mu wartość `cover`. Zobacz ten zabieg na przykładzie poniżej:



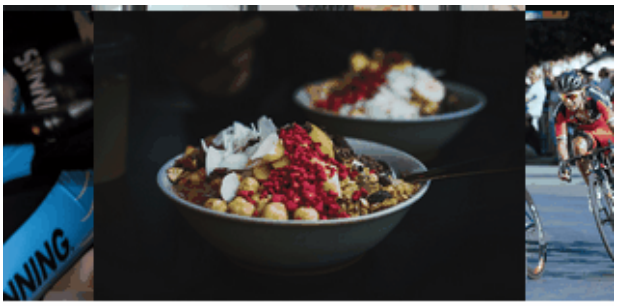
Nasza galeria powinna przypominać tą umieszczoną poniżej. Pamiętajmy jednak o `transition`, które sprawi, że przejście będzie płynne, efektowne i przyjazne dla oka.

Podpowiedzi

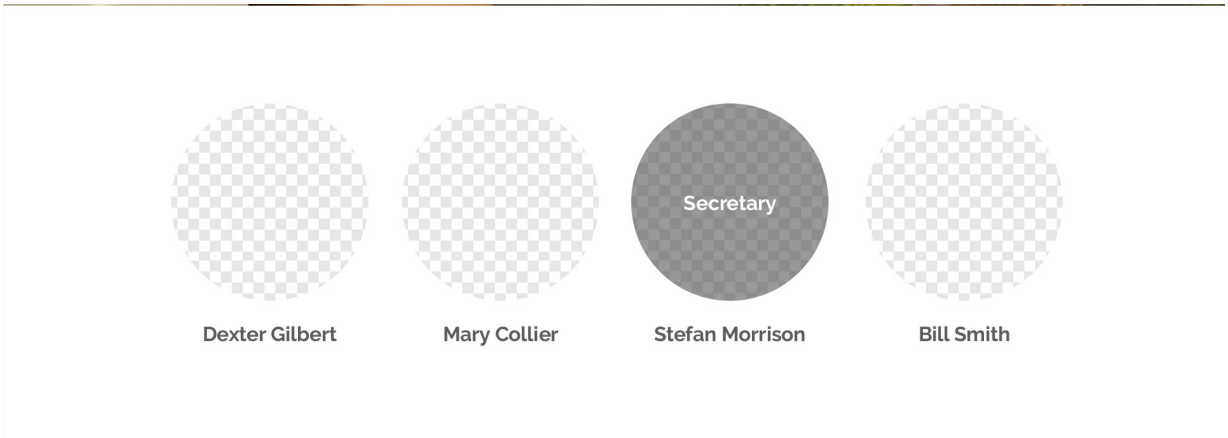
1. Aby ustawić przezroczyste tło, należy użyć `rgba(0,0,0, 0.5)`. Jeżeli użyjesz `opacity`, to ikony wewnątrz też odziedziczą przezroczystość.
2. Alternatywą dla umieszczania zdjęć w tagu `<img>` jest użycie tagu `<div>` i dodanie zdjęcia jako tła za pomocą (`background-image`).

Dla chętnych

Spróbuj wykonać jakiś niestandardowy efekt, który przyciągnie uwagę osoby przeglądającej Twoją stronę. Jeśli nie masz pomysłu, możesz skorzystać z grafiki poniżej. Przed rozpoczęciem zadania dla chętnych możesz zapoznać się z naszym kursem dodatkowym o animacjach.



Zespół



Strona nowoczesnego przedsiębiorstwa nie może obejść się bez sekcji przedstawiającej skład zespołu.

Jak widać na projekcie, składa się ona z czterech równych kolumn, wewnątrz których znajduje się okrągły obrazek wraz z podpisem.

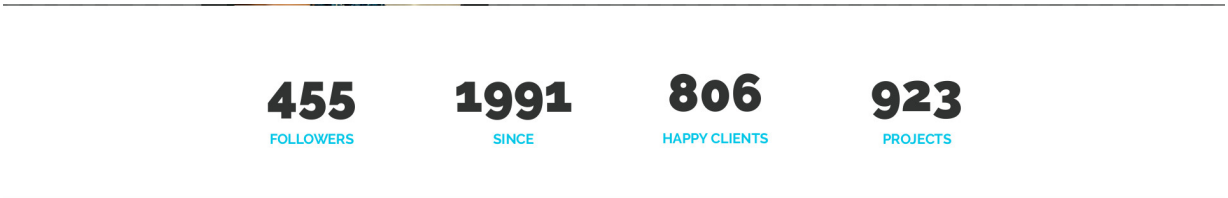
Do stworzenia okrągłych zdjęć wykorzystamy znaną nam już właściwość `border-radius` o wartości 50%. Oprócz tego konieczne będzie również dodanie podpisu do zdjęć.

Dzięki stworzonym wcześniej globalnym stylom oraz gridowi i ta sekcja nie powinna przysporzyć żadnego problemu :).

Sekcja z obrazkiem

Ponownie mamy do czynienia z sekcją rozciągniętą na całą szerokość strony. Grafik nie umieścił tutaj żadnego szczególnego tła. Możemy wiec wstawić jakieś zdjęcie lub zostawić je białe. W środku jest kontener, a wewnątrz niego niespodzianka! Znowu grid :) Dwa elementy: obrazek po lewej zajmuje około ⅓ dostępnej przestrzeni, a ⅔ jest zajęte przez tekst po prawej. Aby dokonać odpowiedniego podziału, wystarczy skorzystać z odpowiednich klas stworzonych w gridzie: ⅓ to inaczej szerokość 4 kolumn, ⅔ to 8.

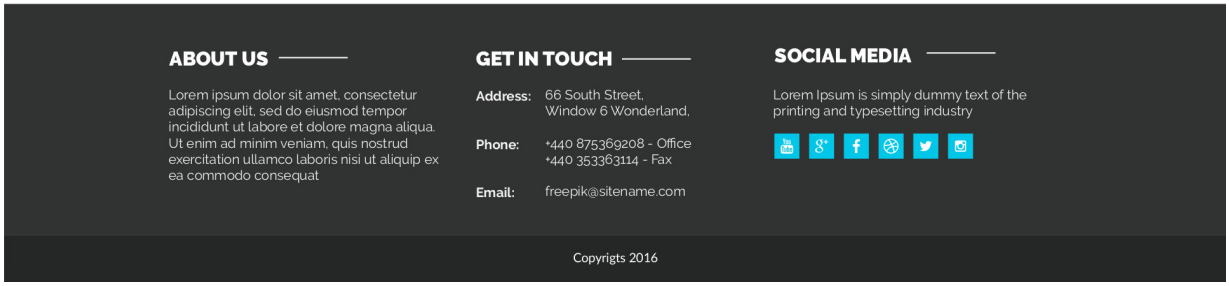
Kilka liczb



Również jedna z bardzo modnych obecnie sekcji. Każda firma chwali się, ile wypiła kubków kawy czy ile zjadła kawałków pizzy.

Ta sekcja nie powinna stanowić większego problemu: znajduje się w nich kilka statystyk dotyczących naszego przedsięwzięcia wraz z podpisami. Stworzenie jej będzie polegać ponownie na wykorzystaniu layoutu kolumnowego i dodaniu odpowiedniego kroju tekstu.

Stopka



Sprawy nieco komplikują się w przypadku stopki. Gdy przyjrzysz się środkowej kolumnie, to zobaczysz, że znajdują się tam dwie części, jedna po lewej, a druga po prawej stronie.

Jednak nie martw się: aby to zrobić, wystarczy zagnieździć wewnątrz tej kolumny drugi grid - w tym celu wystarczy po prostu dodać `<div>` z klasą `row` wewnątrz tej kolumny, tak jak omawialiśmy w poprzednim module, a następnie wewnątrz wiersza umieścić odpowiednie kolumny. W sumie środkowa kolumna stopki sama w sobie będzie zawierała trzy wiersze (`row`), a każdy z nich będzie miał jeszcze dwie kolumny.

Idealne miejsce, żeby poćwiczyć zagnieżdżenie gridu w gridzie :)

Zakończenie

Czas przyjrzeć się naszemu gotowemu projektowi. Na tym etapie można wyłapać dużo błędów, które warto poprawić przed oddaniem strony do klienta.

Sprawdzamy, czy odstępy są zgodne z tym, co zostało zaprojektowane. Czy kolory są takie jak na projekcie, zarówno po najechaniu kursorem, jak i w normalnym stanie? Czy czcionka jest odpowiednio dobrana i przekształcona we właściwych miejscach? Zobacz też, jak wyświetla się strona na różnych przeglądarkach.

Zadanie: Budowa strony na podstawie projektu

Zbuduj stronę podążając śladami autora submodułu :)

Podgląd zadania

Przejdź do projektu ✔

3.4. Projekt dla ambitnych

Osiągniecie wysokiej jakości kodu czy poczucia estetyki przy tworzeniu stron internetowych, wymaga ciągłej praktyki. Właśnie dla tego dajemy Ci możliwość wykonania dodatkowego, samodzielnego, projektu.

Jeżeli nie chcesz podjąć się próby samodzielnego działania to po prostu zaznacz ten submoduł jako wykonany i nie będzie miał on żadnego wpływu na przebieg Twojego szkolenia. Jednak gorąco zachęcam do podjęcia się tego wyzwania jako pierwszej próby samodzielnego działania :)

Tylko dla ambitnych

Gdy już zostaniesz developerem, nadal będziesz spotykać się z różnego rodzaju projektami które mogą być dla ciebie problematyczne. Jakoś wtedy trzeba sobie radzić. Ten projekt jest do samodzielnego wykonania. Można się przy nim posiłkować jedynie wsparciem społeczności którą zebraliśmy na czacie. Może się zdarzyć że akurat mentor pomoże ci rozwiązać jakiś konkretny problem jednak co do zasady to w przypadku dodatkowego projektu nie mają takiego obowiązku.

Im więcej praktyki i samodzielnego kodowania tym lepiej. Dodatkową korzyścią jest możliwość pochwalenia się nimi w ramach portfolio.

Jak się do tego zabrać?

Znasz już projekty Kodilla, to za ich pomocą wykonywaliśmy poprzednie zadania. Jako osoba biorąca udział w kursie masz możliwość tworzenia własnych projektów na stronie https://kodilla.com/projects. Tam możesz swobodnie tworzyć nowe projekty. Utwórz jeden na potrzeby tego zadania i w nim trzymaj swój kod :)

Masz dwie opcje na zakodowanie szablonu. Możesz skorzystać z przygotowanej przez nas grafiki i dobrze znanych Tobie narzędzi, albo użyć programu graficznego - Photoshop lub Avocode.

OPCJA 1

Poniżej znajdziesz link do grafiki do pobrania w formacie `.png`:
Link do grafiki

OPCJA 2

Jeśli posiadasz Avocode lub Photoshop udostępniamy także plik `.psd`:
Link do pliku .psd

Linki do poprzedniej wersji zadania dodatkowego 3.4:
Link do grafiki
Link do pliku .psd

Nie znasz ani Photoshopa, ani Avocode? Nie ma problemu! Przygotowaliśmy specjalnie dla Ciebie poradniki jak zakodować szablon używając któregoś z wyżej wymienionych programów.
Poradnik Avocode
Poradnik Photoshop

Twoim zadaniem jest:

1. Utworzenie nowego projektu.
2. Zakodowanie layoutu na podstawie grafiki.
3. Uzupełnienie treści, mogą być sztuczne (wypełniacze).
4. Zachowanie dobrych praktyk przedstawionych w dotychczasowych treściach.

Jeżeli nie masz czasu to pamiętaj, że zawsze możesz później wrócić i go udoskonalać lub dokończyć wraz ze zdobywaniem wiedzy przekazywanej w kolejnych modułach. Póki co, skup się na dobrych praktykach żeby weszły Ci w krew :)

Powodzenia!

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

✔Zapoznałe(a)m się!

Regulamin

Polityka prywatności