5. Skrótem… Bootstrap

Wyzwania:

- Poznasz, czym są gotowe biblioteki komponentów i dlaczego warto ich używać.
- Dowiesz się, czym jest Twitter Bootstrap i jak go stosować.
- Poznasz komponenty wchodzące w skład Bootstrapa i dzięki temu szybciej zbudujesz swoją stronę.

Kodowanie z gotowców — druga strona medalu

5.1. Gotowce

Czy zawsze trzeba pisać od początku?

Kiedy już stworzysz kilka stron internetowych, stwierdzisz z pewnością, że używasz wielokrotnie tych samych fragmentów kodu. Analogie kodu w różnych projektach we front-endzie mogą dotyczyć między innymi:

- **układu strony** — wiele z nich się powtarza,
- **górnej nawigacji i stopki** — ogólne konwencje ich wyglądu można zliczyć na palcach jednej ręki :)
- **obsługi RWD** — kod responsywnego grida jest prawie zawsze taki sam,
- **innych komponentów CSS** — m.in. takich jak: header, tabele, formularze, listy; wszystkie one też często mają podobny kod.

Czy zatem developer zawsze jest skazany na wymyślanie koła na nowo? Może jednak warto zwrócić uwagę na koszty czy czas poświęcony na projekt i zamiast robić coś 10 razy dłużej, poszukać czasami dróg na skróty?

Skoro pewne fragmenty kodu są częściowo powtarzalne, to może warto wykorzystać te analogie? Możesz np. kopiować kawałki kodu z projektu do projektu i potem lekko je dostosowywać. To pierwszy krok do zaoszczędzenia sobie czasu.

Jednak dużo efektywniej byłoby zebrać takie powtarzalne fragmenty w coś w rodzaju biblioteki z "gotowcami", którą można wykorzystywać wielokrotnie, prawda? Na szczęście nie musisz się tym zajmować, zostało to już zrobione za Ciebie! Zbiór takich bibliotek to tzw. **framework**.

Co to jest framework programistyczny?

Framework to coś w rodzaju szkieletu lub środowiska do tworzenia strony internetowej lub aplikacji. Poprzez swoje założenia narzuca stosującej go osobie pewien sposób budowania rozwiązań i stosowanie uznanych praktyk.

Oprócz ogólnego podejścia do programowania, framework często dostarcza zestaw komponentów i bibliotek, które są czymś w rodzaju gotowców. Frameworki dla serwisów internetowych mają najczęściej postać iluś plików, które są "podłączane" do tworzonego projektu, aby móc wykorzystywać jego możliwości.

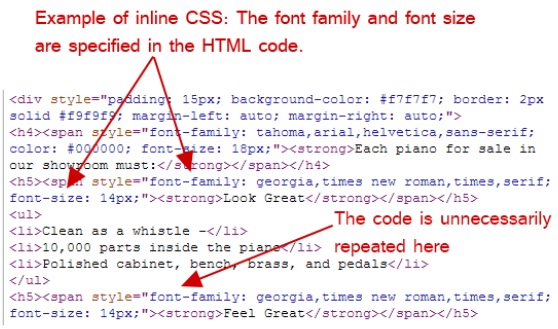
Programista prawie zawsze modyfikuje niektóre z rozwiązań frameworka, aby dostosować tworzony projekt pod specyficzne rozwiązania (np. klienta), których ten uniwersalny szkielet nie uwzględnia.

My skupimy się na frameworkach HTML/CSS, które przyspieszą naszą pracę we front-endzie.

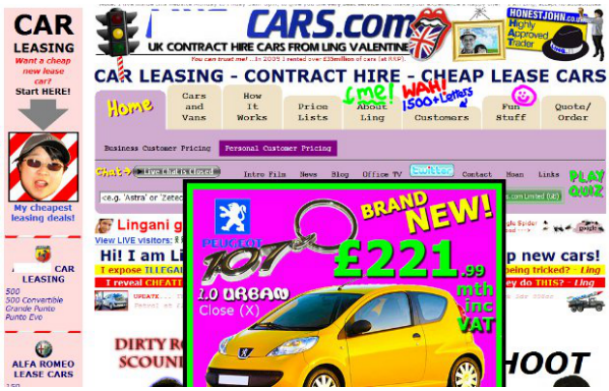
Geneza frameworków CSS

Pierwsza dekada po roku 2000 upłynęła pod znakiem sporego bałaganu w sposobie kodowania front-endu. Mieszano kod HTML z CSS (style inline), tudzież budowano layouty na tabelach.

Po części wynikało to z dosyć pobieżnego, a nawet protekcjonalnego, traktowania przez koderów warstwy front-end w aplikacjach. Programiści uważali (i niektórzy uważają tak do dziś, my się z tym nie zgadzamy), że prawdziwe programowanie jest w back-endzie. Może i back-end jest relatywnie trudniejszy, ale **nie oznacza to, że kod CSS czy JavaScript ma być tworzony byle jak.**



Oprócz standardów w jakości kodu brakowało także standardów w kwestii wyglądu stron www czy ich komponentów. Było to także powiązane z brakiem badań nad ich użytecznością. Te badania pojawiły się w późniejszym czasie i ich wyniki dostarczyły wniosków, że warto stosować konwencje wizualne na stronach www, aby użytkownik na każdej nowej witrynie nie musiał zgadywać, jak jej użyć.



Dlatego społeczność front-endowców zaczęła tworzyć frameworki CSS (zestawy plików CSS z gotowymi klasami), które miały wprowadzać standardy i konwencje przy tworzeniu szablonów HTML/CSS. Celem było również ułatwienie sobie pracy (wykluczenie pisania w kółko tych samych rzeczy), dlatego frameworki szybko zdobyły popularność.

Dodatkowo designerzy zaczęli tworzyć projekty aplikacji internetowych, korzystając z rozwiązań, które były zawarte we frameworkach CSS. W ten sposób doszło do pewnej homogenizacji (ujednolicenia, upodobnienia) web designu. **Strony www zaczęły stawać się do siebie coraz bardziej podobne.**

Dzięki takiej homogenizacji konwencje stały się popularne (np. logo prawie zawsze znajdziemy w lewym górnym rogu) i internautom zaczęło się łatwiej korzystać ze stron, które ich używały. Niektórzy jednak podnoszą kwestię, że strony straciły na oryginalności. Coś za coś. Albo "oryginalne", ale nieużyteczne, albo podobne, ale łatwiejsze w użyciu. Postawiono na to drugie.

Podsumowując, **framework CSS to środowisko do budowania front-endu z zestawem gotowych rozwiązań HTML oraz CSS**. Dzięki nim łatwo i szybko możemy generować elementy stron internetowych, tj. nagłówki z nawigacją, formularze czy różnego rodzaju przyciski. Przy ich pomocy możemy także tworzyć układ naszej strony wykorzystując responsywny grid wbudowany już w dany framework.

Dzięki frameworkom nie musimy pisać zestawu deklaracji CSS dla elementu, który chcemy np. wyśrodkować. Często wystarczy nadać mu odpowiednią klasę zawartą we frameworku. W ten sposób oszczędzamy sobie dużo "pracy u podstaw" i startujemy z gotowymi rozwiązaniami, które dodatkowo możemy swobodnie modyfikować.

Lista popularnych frameworków

Początkowo frameworki CSS tworzono w domowym zaciszu, ale tematem zainteresowały się firmy, które zaczęły tworzyć takie zestawy gotowców dla swoich potrzeb. Następnie zaczęły one udostępniać takie rozwiązania za darmo innym podmiotom, co miało też wymiar PR-owy :)

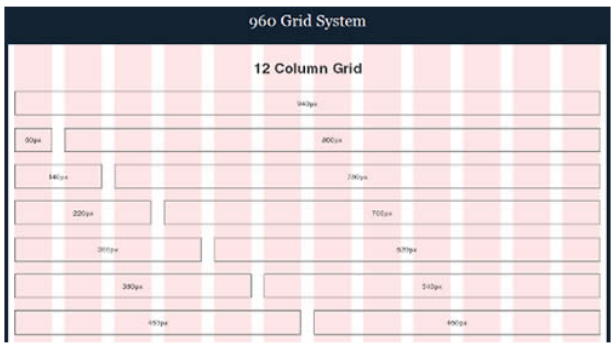
Niekwestionowanym liderem wśród frameworków CSS jest od dawna **Twitter Bootstrap**. Stworzony przez pracowników Twittera na wewnętrzne potrzeby firmy został udostępniony jako projekt open source i zyskał wielką popularność. Oprócz biblioteki CSS zawiera on także zestaw skryptów w JavaScript, dzięki czemu jest niemal kompletnym narzędziem.

Inne popularne frameworki to **Zurb Foundation**, **Pure CSS** czy **Semantic UI**. Wybór pomiędzy nimi to głównie kwestia preferencji. Każdy z nich ma nieco inną filozofię i założenia, co przekłada się też na skalę i stopień zaawansowania biblioteki.

Zurb Foundation

Najpoważniejszym konkurentem Bootstrapa jest Foundation.
Foundation, podobnie jak Bootstrap, oferuje grid oraz bibliotekę komponentów front-endowych, z których możemy korzystać. Nie ma jednak aż tak silnie zdefiniowanych kwestii wizualnych, co z jednej strony pozwala nam na utworzenie bardziej unikalnego wyglądu, ale z drugiej — wymaga jednak większego nakładu pracy.

Foundation ma pewną przewagę nad Bootstrapem w tej kwestii, że domyślnie umożliwia walidację formularzy (sprawdzanie poprawności wprowadzonych danych wpisanych do formularza na stronie). Innym ciekawym rozwiązaniem jest tzw. interchange, który umożliwia wczytywanie zawartości strony w zależności od tego, z jakiej przeglądarki korzysta użytkownik.



Pure CSS

Zupełnie inne podejście proponuje Pure CSS. W tym wypadku zamiast rozbudowanego frameworku dostajemy lekki plik CSS, w którym zawarty jest zestaw klas. W jego skład wchodzą często stosowane rozwiązania, takie jak responsywny grid, domyślne style formularzy, przyciski etc. W ten sposób dostajemy lekką bibliotekę, nienarzucającą swojego wyglądu.

Niewątpliwą zaletą Pure CSS jest jego niewielki rozmiar, co dobrze wpływa na prędkość ładowania strony. Ponadto framework ten nie narzuca zbyt wielu rozwiązań. Działa to jednak w dwie strony: to, czego nie ma w bibliotece, będzie wymagało stworzenia od zera, co oczywiście wymaga więcej pracy.

Semantic UI

Ciekawym rozwiązaniem jest również Semantic UI. Założeniem tego frameworku było to, że miał on być czytelny i zrozumiały dla programisty. Podobnie jak w Bootstrapie, mamy tutaj rozbudowaną bibliotekę różnorodnych komponentów o mocno zdefiniowanym wyglądzie. Ciekawym rozwiązaniem jest również możliwość wczytania tylko poszczególnych komponentów. Dzięki temu możemy z frameworka wykorzystać tylko to, czego potrzebujemy. Nie musimy każdorazowo ładować całego dużego pliku biblioteki.

Dlaczego Twitter Bootstrap?

Jeśli dopiero zaczynasz swoją przygodę z programowaniem front-end i chcesz zacząć używać jakiegoś frameworka CSS, naszym zdaniem najlepszym wyborem jest zdecydowanie **Twitter Bootstrap**. Wybór podyktowany jest następującymi powodami:

- Oparcie zawartych w nim rozwiązań na podstawie ogromnej liczby odwiedzin serwisu Twitter. Możemy być pewni, że wypracowane tam konwencje są potwierdzone licznymi badaniami na dużych grupach odwiedzających.
- Największa społeczność, która stale pomaga usprawniać ten framework, co sprawia, że wszelkie błędy są szybko usuwane, a tempo rozwoju zadowalające. Dodatkowo istnieje wiele forów, na których szybko otrzymamy pomoc w razie problemów z używaniem tego frameworka.
- Istnieje masa dodatków do Bootstrapa w postaci różnych motywów wyglądu strony, czy bazy snippetów dla modyfikacji komponentów. Są one tworzone przez społeczność.

Z czasem można zacząć korzystać także z innych frameworków CSS. Najpierw warto jednak poznać najpopularniejszy, aby lepiej zdefiniować swoje potrzeby co do innych wyborów.

Popularne biblioteki

Wyjaśnijmy pewne różnice pomiędzy pojęciami framework a biblioteka, gdyż nie są to pojęcia równoznaczne. Przyjęło się, że różnica sprowadza się najczęściej do tego, kto kontroluje kogo.

W przypadku **biblioteki** jest to tak, jakbyśmy dostali zestaw narzędzi (w tym wypadku jest to zestaw fragmentów kodu). To, co z nim zrobimy, zależy od nas, i mamy w tym zakresie pełną dowolność. W przypadku programowania stron będzie to po prostu fragment kodu, np. zestaw funkcji czy klas, które możemy przywoływać wtedy, kiedy nam to potrzebne. Biblioteka z reguły ma jedno konkretnie ustalone zadanie i zajmuje się przede wszystkim rozwiązaniem jednego problemu.

Framework natomiast jest dużo bardziej rozbudowany. Ma za zadanie umożliwić programiście stworzenie całej aplikacji od wczesnego prototypu do finalnego produktu. Wymaga to najczęściej pewnego określonego podejścia — twórcy frameworku najczęściej podejmują określone decyzje na temat metody tworzenia aplikacji, np. decydując się na podejście mobile-first lub graceful degradation. Dlatego też gdy decydujemy się na framework, jesteśmy ograniczeni przez sposób, w jaki został on zbudowany. Bywa tak, że framework korzysta z kilku bibliotek.

Bootstrap jest frameworkiem, i to przede wszystkim na nich skupiamy się w niniejszym rozdziale. Warto jednak poznać popularne biblioteki, z których korzystamy przy tworzeniu stron internetowych.

Reset

Najprostszą, a zarazem jedną z najpopularniejszych bibliotek CSS, jest CSS Reset. Jest to zaledwie kilkanaście linijek CSS, których zadaniem jest zniwelowanie domyślnych reguł CSS, aplikowanych przez przeglądarkę. Potrzeba taka podyktowana jest głównie tym, że każda przeglądarka ma nieco inne style domyślne, co może powodować różnice w wyświetlaniu elementów w różnych przeglądarkach. Resetując je, dostajemy jakby czystą kartkę, na której następnie możemy definiować swoje reguły od zera.

Normalize

Nieco bardziej rozbudowaną alternatywą jest normalize.css. Jego zadaniem jest zapewnienie spójności wyświetlania wszystkich elementów na różnych przeglądarkach. Autor tej biblioteki stoi na stanowisku, że nie ma potrzeby wymazywać wszystkich domyślnych stylów przeglądarek, a jedynie te, które powodują problemy. Dzięki temu otrzymujemy bardziej jednolite wyświetlanie wszystkich elementów na najpopularniejszych przeglądarkach. Z tej biblioteki korzysta m.in. Bootstrap.

Animate.css

Inną popularną biblioteką jest animate.css. Razem z tym plikiem dostajemy pokaźną kolekcję efektownych animacji CSS. By je dodać, wystarczy dopisać odpowiednie klasy do elementu. W ten prosty sposób możemy nieco ożywić naszą stronę, dodając jedynie lekki plik i kilka klas.

Kodowanie z gotowców — druga strona medalu

Korzystanie z frameworków czy bibliotek może wydawać się bardzo kuszące, szczególnie na początku przygody z kodowaniem. Dzięki nim wystarczy podstawowa wiedza o HTML i CSS, żeby stworzyć całkiem zaawansowane i profesjonalnie wyglądające strony internetowe.

Problem polega na tym, że zaczynanie przygody z kodowaniem od frameworka powoduje, że uczący się nie zrozumie jego działania "pod spodem". Według niektórych opinii takie osoby są "klepaczami", którzy tylko "składają klocki" z frameworka, ale tak naprawdę nie rozumieją, jak to działa.

Czasem taka osoba przychodzi na rekrutację na front-endowca i okazuje się, że nie potrafi pociąć prostej strony bez pomocy frameworka. **To nie świadczy o niej najlepiej**. Dlatego w toku nauki zastosowaliśmy taką, a nie inną kolejność modułów.

Do tego **nigdy nie jest tak**, że wykorzystując np. Bootstrapa w projekcie, korzystamy tylko z jego oryginalnych klas z CSS! Bootstrap jest uniwersalny, ale nie aż tak. Z powodu unikalnych wizualnie elementów w danym projekcie graficznym zawsze trzeba coś w tych klasach CSS dodać, zmodyfikować, lub dodać swoje reguły wyglądu.

Dla kogoś, kto nie ma większej świadomości CSS (poza frameworkiem), może to być bardzo trudne do wykonania. Pamiętaj, że nic nie zastąpi solidnej znajomości technologii webowych i zawsze należy być świadomym, jak działają narzędzia, których używamy. A więc zanim zaczniemy ułatwiać sobie życie gotowcami, napiszmy kilka tysięcy linijek czystego kodu!

Wady używania frameworka

- Niezalecane dla osób, które nie miały do czynienia z czystym kodem.
- Jeśli mamy bardzo oryginalny projekt (często niezgodny z konwencjami, co jest wadą), użycie np. Bootstrapa wymaga nadpisania lub zmodyfikowania wielu istniejących reguł CSS. Czasem użycie frameworka wręcz nie ma sensu.

Zalety

- Oszczędność czasu i energii poświęconych kodowaniu.
- Łatwe tworzenie stron responsywnych.
- Bezproblemowe działanie w większości popularnych przeglądarek internetowych.
- Sugerowanie użycia wizualnej konwencji dla elementów kodowanej strony internetowej.
- Wsparcie społeczności w przypadku popularnego frameworka.

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

✔Zapoznałe(a)m się!

5.2. Jak używać Bootstrapa?

Bootstrap, jak już powiedzieliśmy, jest dość rozbudowanym frameworkiem CSS. Duża liczba komponentów idzie w jego przypadku w parze z drobiazgowo określonym wyglądem, przez co możemy powiedzieć, że wraz z Bootstrapem dostajemy również

miarę określony *look&feel*. Jest on przez sceptyków określany czasami jako "Bootstrap look".

Jednak możemy dzięki niemu szybko zbudować gotowy prototyp strony i jej docelową wersję. Jedyne, co trzeba zrobić, to dodać go do strony. Do tego nic nie stoi na przeszkodzie, aby go modyfikować pod swoje potrzeby.

Dokumentacja Bootstrapa

Każdy szanujący się framework ma swoją stronę, na której umieszczona jest jego dokumentacja. Dokumentacja to podstawowe źródło informacji dotyczących zarówno instalacji, jak i użytkowania danego narzędzia.

Dokumentację Twitter Bootstrapa znajdziesz pod adresem: https://getbootstrap.com/docs/3.3/.
Jest ona wykonana w bardzo czytelny sposób, jak przystało na framework, który ma pomagać tworzyć czytelne i ładne szablony HTML/CSS.

Wersje Bootstrapa

Niedawno został opublikowany Bootstrap w wersji 4, jednak w tym kursie będziemy korzystać z wersji 3.3. O przyczynach przeczytasz pod koniec tego submodułu.

Główne sekcje dokumentacji (menu górne):

- **Getting started** — informacje związane głównie z podłączeniem Bootstrapa do naszego projektu.
- **CSS** — informacje związane z głównymi założeniami frameworka i jego kluczowymi elementami, takimi jak: korzystanie z grida responsywnego, rozmiary czcionek (typografia), elementy HTML (m.in. obrazki, tabele, formularze, przyciski), klasy pomocnicze i kwestie związane z RWD.
- **Components** — biblioteka gotowych komponentów CSS, które możemy wstawić na naszą stronę, np. menu nawigacji, grupy przycisków i inne. Każdy komponent posiada opis kodu potrzebnego do jego zastosowania, a także wyjaśnienie, jak działa i jak go modyfikować.
- **JavaScript** — analogiczna biblioteka komponentów wykorzystujących JavaScript, takich jak pop-upy wewnętrzne (tzw. modale) czy karuzela slajdów do headera.
- **Customize** — narzędzie do generowania pliku Bootstrapa z własnymi wartościami wizualnymi dla komponentów.

- **Themes** (po prawej w menu) — przykładowe szablony z użyciem Bootstrapa. Pamiętaj jednak, że takich szablonów można znaleźć w Internecie wiele, z czego duża część jest dostępna za darmo :)

Samodzielne zapoznanie się z tą stroną może być dla wielu osób wystarczające, aby zacząć używać Bootstrapa. Jednak przeprowadzimy Cię przez większość sekcji z tej dokumentacji, aby rozwiać możliwe wątpliwości dotyczące działania tego frameworka :)

Struktura informacji na temat dokumentacji kolejnych fameworków CSS, które będziesz poznawać, prawdopodobnie będzie analogiczna, więc będzie Ci je łatwiej wypróbować.

Jak dodać Bootstrapa do strony

Istnieje kilka sposobów dodania Bootstrapa do własnego projektu. Jeden z nich polega na pobraniu zestawu plików CSS, JS i fontów ze strony Bootstrapa (Getting started / Download Bootstrap). Następnie musimy rozpakować i umieścić pobrany folder w naszym projekcie, w głównym katalogu, i dodać w pliku HTML odpowiednie linki do plików bootstrap.min.css (w head) oraz opcjonalnie *bootstrap.min.js* (przed `</body>`).

Innym, prostszym rozwiązaniem jest dołączenie pliku Bootstrapa ze źródła zewnętrznego, który jest udostępniany przez tzw. **CDN** czyli **Content Delivery Network**. CDN jest siecią serwerów, na których przechowywane są często używane (m.in. na stronach internetowych) pliki.

W tym podejściu nie trzeba ich ściągać. Wystarczy zalinkować odpowiednie źródła:

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrc
```

Jak zauważysz, plik w powyższym linku ma w nazwie również słowo min. Oznacza to, że link wskazuje na plik *zminifikowany*, z którego zostało usunięte wszelkie formatowanie (wcięcia, łamanie linii) w celu wydajniejszego wczytywania strony.

Są też inne metody podłączenia Bootstrapa, ale dwie powyższe będą wystarczające na długi czas Twojej przygody z front-endem.

Podepnij Bootstrapa z CDN w swoim przykładowym pliku HTML (np. na Twoim komputerze) i wstaw w body taki kod:

```
<button type="button" class="btn btn-primary">Primary</button>
```

Otwórz podgląd. Zobaczysz ładnie ostylowany za pomocą gotowych stylów z Bootstrapa przycisk. Po raz pierwszy użyliśmy tego frameworka CSS!

Lokalnie czy CDN

Zastanówmy się, czy jest różnica pomiędzy trzymaniem plików Bootstrapa (bądź innych bibliotek) na własnym serwerze (po ściągnięciu ich ze strony frameworka) a linkowaniem ich z CDN.

Naszym zdaniem warto linkować CDN, ponieważ:

- wiele stron używa plików bibliotek z takich źródeł i tym samym jest duże prawdopodobieństwo, że plik jest gdzieś już na komputerze internauty (*pamięć cache*),
- serwery z CDN mają zazwyczaj mniejszą awaryjność niż te, których my sami (programiści) używamy w ramach hostingu.

O zalecie takiego linkowania w przypadku projektów edukacyjnych nie trzeba wspominać. Zamiast wgrywać framework na serwer, wystarczy dodać jedną linijkę w `<head>`.

Sporym wyjątkiem od takiej praktyki są strony dużych instytucji, których polityka bezpieczeństwa wymusza trzymanie wszystkich plików witryny na własnych serwerach. Co prawda ciężko o zagrożenie dla bezpieczeństwa ze strony takich front-endowych plików, ale korporacje wolą dmuchać na zimne.

Kolejnym powodem trzymania pliku Bootstrapa u siebie może być konieczność jego zmodyfikowania, czego na CDN nie możemy zrobić. Jednak zanim zdecydujesz się na modyfikację arkuszy bootstrapowych, przeczytaj kolejną sekcję.

Modyfikacja czy oddzielny CSS?

Niemal w 100% przypadków jest tak, że pomimo zastosowania Bootstrapa, projekt graficzny lub wymagania klienta bądź własne wymuszają zmianę domyślnego wyglądu elementów na bazie tego frameworka.

Można trzymać plik Bootstrapa u siebie i go modyfikować. Naszym zdaniem lepszą praktyką jest jednak pozostawienie tego pliku bez zmian i nadpisywanie jego styli za pomocą klas we własnym, osobnym pliku CSS.

Przykładowo, Bootstrap narzuca, że główne przyciski są niebieskie. Wtedy możemy:

1. utworzyć plik style.css i podłączyć go w <head> po linku do pliku Bootstrapa, **nie przed**,
2. dodać do HTML kod głównego przycisku z Bootstrapa,
3. następnie w podglądzie, za pomocą inspektora Chrome, sprawdzamy, jaka jest nazwa klasy z Bootstrapa odpowiadająca za ten przycisk,
4. wstawiamy do *style.css* regułę z taką klasą i zmieniamy kolor tła tego przycisku na inny kolor :)
5. zadziała ostatni załączony plik CSS z sekcji head, którego reguła o takim samym selektorze przeciąży (nadpisze) tę z Bootstrapa.

Tym sposobem nadal używamy frameworka, ale niektóre elementy dostosowujemy pod siebie :)

Takie podejście, że nie modyfikujemy pliku Bootstrapa, ma jeszcze następujące zalety:

- znając ten framework CSS prawie na pamięć, nie musimy przeglądać, co w nim zmieniono — wystarczy, że w oddzielnym (pewnie mniejszym) pliku CSS, sprawdzamy tylko to, co zmodyfikowano,
- jeśli pojawia się nowa wersja Bootstrapa, można śmiało podmienić framework na nowszą edycję bez obaw o zepsucie wyglądu strony. W przypadku zmodyfikowanego framewworka, zastąienie go nowszą wersją oznacza sporo pracy z ponowną zmianą styli.

Która wersja?

Każdy system, oprogramowanie, framework lub biblioteka są stale rozwijane przez swoich twórców. Po wprowadzeniu iluś ulepszeń nadaje się takiemu bytowi numerek, np. 3.1. Zmiana liczby przed kropką to zazwyczaj większe wydanie kolejnej wersji frameworka. Liczba po kropce to kolejne podwersje z poprawkami błędów lub małymi ulepszeniami.

Po wejściu na http://getbootstrap.com/ możemy sprawdzić aktualną wersję Bootstrapa (current). Polecamy używanie tych aktualnych i stabilnych.

Oczywiście często zdarza się, że kusi nas użycie wersji oznaczonej jako "beta". Możemy wówczas przetestować jej działanie w testowej wersji projektu i sprawdzić nowe opcje. Odradzamy jednak stosowanie wersji beta w projektach klienckich, ponieważ w trakcie może okazać się, że wersja ta jest niekompatybilna z założeniami klienta i musimy pisać kod od nowa.

W naszym kursie będziemy używać wersji 3.3. Wynika to z faktu, że większość projektów z którymi spotkasz w najbliższym czasie jest oparta o dotychczasową wersję 3, a nie o najnowszą wersję 4 (została wydana 18.01.2018). Upewnij się, że czytasz dokumentację właściwej wersji, ponieważ o ile nie zmieniły się założenia na których oparty jest Bootstrap, to kod nowej wersji nie jest kompatybilny ze starą.

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

✔Zapoznałe(a)m się!

5.3. Klasy pomocnicze Bootstrapa

Wprowadzenie

Najprostszym sposobem, w jaki możemy skorzystać z Bootstrapa, jest dodanie do różnych elementów prostych klas pomocniczych. Możemy w ten sposób po prostu podłączyć arkusz CSS oraz wykorzystać zdefiniowane tam gotowe reguły.

Grid

Jednym z najważniejszych komponentów, jakie dostajemy z Bootstrapem, jest grid. Dzięki temu nie musimy każdorazowo przepisywać go na nowo.

Bootstrap dostarcza nam bardzo rozbudowany system, który został wielokrotnie przetestowany i sprawdzony na wielu urządzeniach. Po dodaniu Bootstrapa nie musimy pisać ani jednej linijki CSS w celu utworzenia gridu, po prostu działa on poprzez dodawanie odpowiednich klas.

Zasady stosowania są podobne jak w przypadku systemu, który dotychczas budowaliśmy od zera. Wszystkie elementy umieszczamy w kontenerze (opcjonalnie), a następnie wierszu (koniecznie). Robimy to za pomocą klas `container` oraz `row`, tak jak w przykładzie poniżej:

```
<div class="container">
    <div class="row">
        <div class="col-md-4">Hello World!</div>
        <div class="col-md-4">Hello World!</div>
        <div class="col-md-4">Hello World!</div>
    </div>
</div>
```

Mamy również do dyspozycji klasę `container-fluid`. Różni się ona od zwykłego kontenera tym, że zamiast stałej szerokości dostajemy kontener zajmujący 100% dostępnej szerokości.

Podobnie jak w przypadku gridu stosowanego w poprzednim rozdziale, mamy do dyspozycji system kolumn, za pomocą którego możemy nadawać elementom szerokość i modyfikować ją na różnych urządzeniach. Bootstrap stosuje 4 breakpointy, każdemu z nich przypisany jest odpowiedni prefiks:

- telefony (poniżej 768px) `.col-xs-`
- małe urządzenia (>768px) `.col-sm-`
- średnie urządzenia (>992px) `.col-md-`
- duże urządzenia (>1200px) `.col-lg-`

Dokumentacja grida

Dostajemy również do dyspozycji cały zestaw klas do nadawania przesunięcia. Stosujemy tutaj podobne nazewnictwo jak poprzednio, dodając jedynie -offset:

- `.col-xs-offset`
- `.col-sm-offset`
- `.col-md-offset`
- `.col-lg-offset`

Na poniższym przykładzie zobaczmy, jak możemy z tego skorzystać:

Przykładowe wykorzystanie gridu

Dodaliśmy tam po jednym kontenerze, pierwszy o stałej szerokości, a drugi o szerokości 100%. Następnie dodane zostały wiersze oraz kolumny. Aby nadać offset całemu wierszowi, wystarczy dodać klasę do pierwszej kolumny. Zobacz też, jak wyświetlają się kolumny na małym ekranie. Jak widać, stworzyliśmy cały layout przy pomocy zaledwie kilku linijek HTML!

Typografia

Bootstrap daje nam do dyspozycji dużo pomocniczych klas, za pomocą których możemy kontrolować wygląd naszego tekstu.

Wszystkie poziomy nagłówków od `<h1>` do `<h6>` mają zdefiniowane style, można też wykorzystać ich wygląd dla innych elementów dodając po prostu klasę `h1`.

Możemy również dodawać mniejszy podtytuł do nagłówków, dodając w nich element `<small>`:

```
<div class="container">
    <h1>Example text <small>Smaller text</small></h1>
    <h2>Example text <small>Smaller text</small></h2>
    <h3>Example text <small>Smaller text</small></h3>
    <h4>Example text <small>Smaller text</small></h4>
    <h5>Example text <small>Smaller text</small></h5>
    <h6>Example text <small>Smaller text</small></h6>
</div>
```

Ponadto dostajemy też do dyspozycji kilka klas, za pomocą których możemy modyfikować wyrównanie tekstu:

```
<p class="text-left">Alignment to left</p>
<p class="text-center">Centering</p>
<p class="text-right">Alignment to right</p>


<p class="text-justify">Text justify</p>
<p class="text-nowrap">Text without wrapping line</p>
```

Możemy też skorzystać z klas, by przekształcić tekst:

```
<p class="text-lowercase">small letters</p>
<p class="text-uppercase">CAPITAL LETTERS</p>
<p class="text-capitalize">Every Word Starts With Capital Letter</p>
```

HTML Result EDIT ON

```
<div class="container">
    <h1>Example text <small>Smaller text</small></h1>
    <h2>Example text <small>Smaller text</small></h2>
    <h3>Example text <small>Smaller text</small></h3>
    <h4>Example text <small>Smaller text</small></h4>
    <h5>Example text <small>Smaller text</small></h5>
    <h6>Example text <small>Smaller text</small></h6>
    <br>
<p class="text-left">Alignment to left</p> <p class="text-center">Centering</p> <p class="text-right">Alignment to right</p> <p class="text-justify">Text justify</p> <p class="text-nowrap">Text without wrapping line</p>
  <p class="text-lowercase">small letters</p> <p class="text-uppercase">CAPITAL LETTERS</p> <p class="text-capitalize">Every Word Starts With Capital Letter</p>
```

Example text

Smaller text

Example text Smaller text

Example text Smaller text

Example text Smaller text

Example text Smaller text

Example text Smaller text

Alignment to left

Centering

Alignment to right

Text justify

Text without wrapping line

small letters

CAPITAL LETTERS

Every Word Starts With Capital Letter

Resources 1× 0.5× 0.25× Rerun

Tabele

Możemy szybko budować tabele za pomocą gotowych bootstrapowych klas. Wystarczy dodać do elementu <table> klasę "table". Początkowo może wydawać się to dziwnym rozwiązaniem, jednak zważywszy na to, że z tego elementu korzystają różne pluginy i inne rozwiązania, twórcy Bootstrapa zdecydowali się przypisać style klasy zamiast do elementu:

```
<div class="container">
  <h2>Simple table</h2>
  <p>Using the table class we can add basic styles to the table</p>
  <table class="table">
    <thead>
      <tr>
        <th>Name</th>
        <th>Surname</th>
        <th>e-mail</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>Jan</td>
        <td>Kowalski</td>
        <td>jan@example.com</td>
      </tr>
      <tr>
        <td>Sebastian</td>
        <td>Gałecki</td>
        <td>sebastian@example.com</td>
      </tr>
      <tr>
        <td>Mateusz</td>
        <td>Nowak</td>
        <td>mati@example.com</td>
      </tr>
    </tbody>
  </table>
</div>
```

Możemy też modyfikować wygląd tabeli — za pomocą dodania klasy `table-striped`, by zastosować naprzemienne kolory tła wierszy oraz za pomocą klasy `table-bordered`, by dodać obramowanie. Innym ciekawym efektem jest np. `klasa table-hover`, która podświetli rzędy, na które najechał użytkownik. Możemy korzystać z tych rozwiązań dowolnie, mieszając je ze sobą, tak jak w przykładzie poniżej:

HTML

```
<div class="container">
  <h2>Table</h2>
  <p>The .table class adds basic
CSS rules, which are responsible
for the look. In addition, we can
use additional classes responsible
for the look, such as table-
striped, table-bordered and table-
hover:</p>
  <table class="table table-
striped table-bordered table-
hover">
    <thead>
      <tr>
        <th>Name</th>
        <th>Surname</th>
        <th>Email</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>Jan</td>
        <td>Kowalski</td>
      <td>jan@example.com</td>
      </tr>
      <tr>
        <td>Janusz</td>
        <td>Zawadzki</td>

<td>janusz@example.com</td>
      </tr>
      <tr>
        <td>Anna</td>
```

Result

EDIT ON

Table

The .table class adds basic CSS rules, which are responsible for the look. In addition, we can use additional classes responsible for the look, such as table-striped, table-bordered and table-hover:

Name	Surname	Email
Jan	Kowalski	jan@example.com
Janusz	Zawadzki	janusz@example.com
Anna	Zawadzka	anna@example.com

Resources 1× 0.5× 0.25× Rerun

Formularze

Formularze pozwalają nam na wprowadzanie danych przez użytkownika, np. nazwiska, hasła, tekstu. Wszystkie elementy HTML, które wchodzą w skład formularza (nie tylko korzystające z Bootstrapa), powinny znaleźć się wewnątrz elementu `<form>`.

By dodać odpowiednie rodzaje pól (`<input>`) do formularza, dodajemy do tagów `<input>` odpowiedni atrybut, jak na przykładzie poniżej:

```
<form>
  e-mail:<br>
  <input type="email" name="email" placeholder="example-adress@exampl
  Imię:<br>
  <input type="text" name="name" placeholder="Jan"><br><br>
  <input type="submit" value="Submit">
</form>
```

Wyjątkiem jest tutaj element przeznaczony do wprowadzania dłuższego tekstu, jakim jest `<textarea>`.

```
<textarea name="InputMessage" id="InputMessage" class="form-control"
```

Najczęściej używane rodzaje elementu input to:

- **text** — dla danych tekstowych, np. imienia i nazwiska
- **password** — dla podawania hasła
- **submit** — dla przycisku wysyłającego formularz
- **radio** — pozwalają użytkownikowi wybrać jedną opcję z kilku dostępnych
- **checkbox** — pozwalają wybrać użytkownikowi dowolną liczbę opcji
- **button** — dla przycisków.

Aby elementy formularza zostały prawidłowo opisane, każdy `<input>` powinien mieć przypisany atrybut `name="nazwa-pola"`. Następnie możemy dodać opis do pól formularza (do inputów) za pomocą tagu `<label for="element_formularza">`. Wartość atrybutu "for" powinna być równa atrybutowi id, jaki nadaliśmy opisywanemu elementowi.

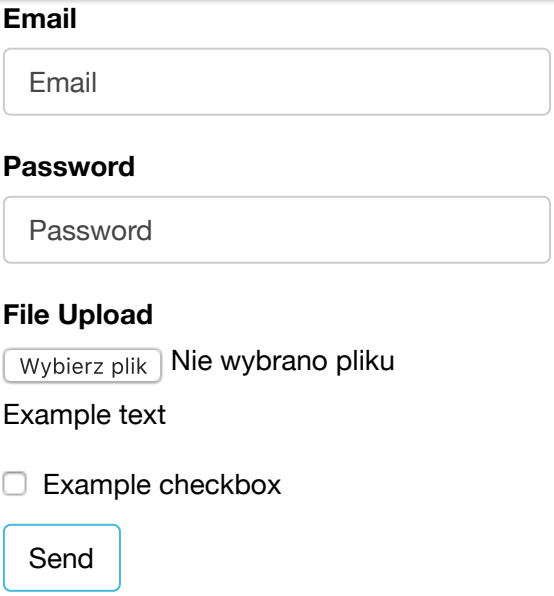
Gdy korzystamy z Bootstrapa, możemy pogrupować elementy `<input>` wraz z ich opisami `<label>` w jeden div z klasą form-group, by nadać im elegancki wygląd.

HTML

```
<div class="container">
<form>
  <div class="form-group">
    <label
for="exampleInputEmail1">Email</label
    <input type="email"
class="form-control"
id="exampleInputEmail1"
placeholder="Email">
  </div>
  <div class="form-group">
    <label
for="exampleInputPassword1">Password
    <input type="password"
class="form-control"
id="exampleInputPassword1"
placeholder="Password">
  </div>
  <div class="form-group">
    <label
for="exampleInputFile">File
Upload</label>
    <input type="file"
id="exampleInputFile">
    <p class="help-block">Example
text</p>
  </div>
  <div class="checkbox">
    <label>
      <input
type="checkbox">Example checkbox
    </label>
```

Result

EDIT ON



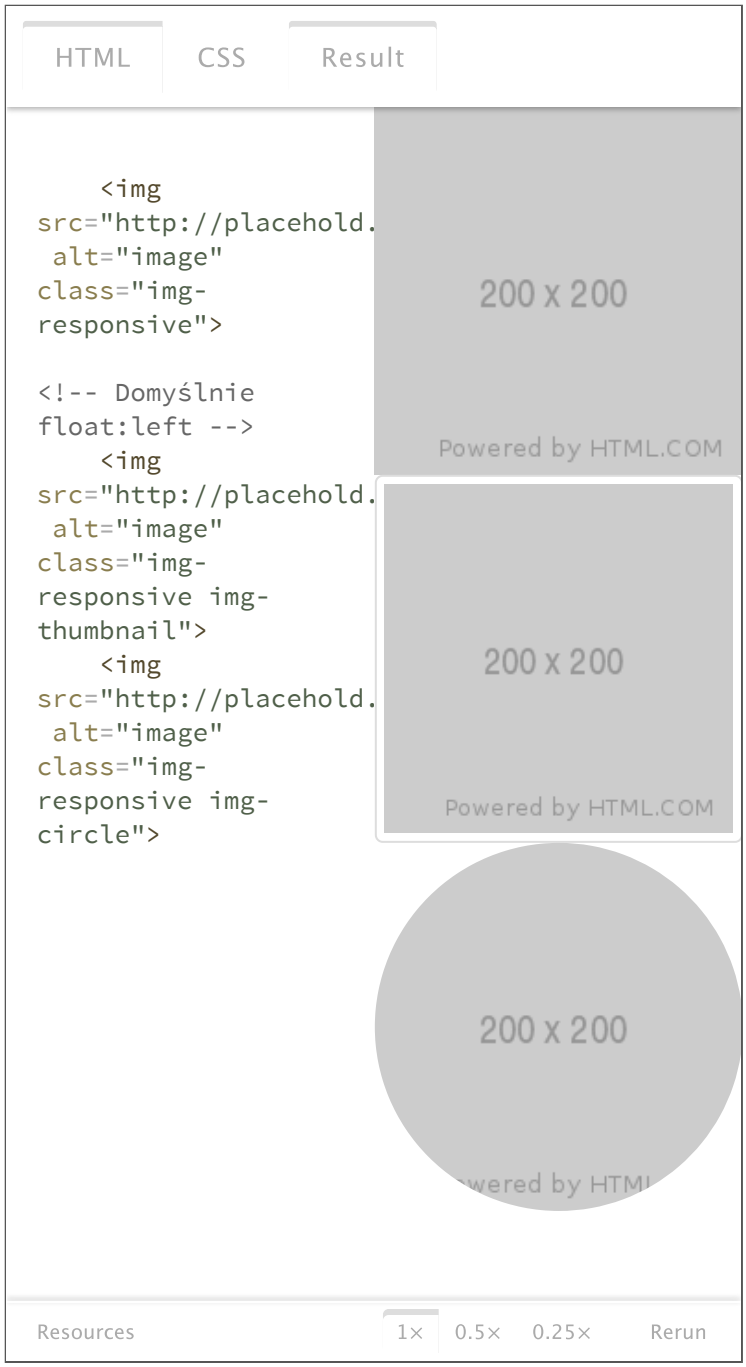
Resources 1× 0.5× 0.25× Rerun

By jeszcze precyzyjniej kontrolować wygląd formularzy, możemy skorzystać z klas grida przy dodawaniu formularza. Aby skorzystać z takiej możliwości, wystarczy dodać klasę `form-horizontal` do elementu `<form>`, co przypisze mu wszystkie właściwości odpowiedzialne za grid.

Grafiki

Bootstrap sam zadba o to, by nasze obrazki były responsywne. Wystarczy, że dodamy do obrazka klasę `img-responsive`. Znalazła tu zastosowanie poznana wcześniej technika nadawania `max-width: 100%;` oraz `height: auto;`, by obrazek dostosowywał się do elementu, w którym jest zawarty.

Ponadto mamy też do dyspozycji klasy `img-thumbnail` oraz `img-circle`, które pozwalają nam modyfikować wygląd obrazków. Klasa `img-circle` sprawi, że nasz obrazek będzie okrągły, natomiast `img-thumbnail` tworzy ramkę.



Buttony

Skorzystaliśmy już z klas Bootstrapa by dodać do strony przycisk, jednak to dopiero początek.

Podstawowy wygląd przycisku tworzymy za pomocą klasy `btn` (zawiera ona style wspólne dla wszystkich przycisków). Następnie możemy dodać przyciskom klasy odpowiedzialne za wygląd w zależności od tego, za co są odpowiedzialne:

- `btn-default` — domyślny wygląd przycisku, niczym się nie wyróżnia,
- `btn-primary` — koloruje przycisk na niebiesko, wyróżniając go na tle innych, wskazuje na główny przycisk,
- `btn-success` — zielony kolor ma zachęcać do wykonania pożądanej akcji,
- `btn-info` — błękit wskazujący na przycisk pokazujący informacje,
- `btn-warning` — pomarańczowy przycisk wskazuje, by korzystając z niego, być ostrożnym,
- `btn-danger` — czerwony przycisk, czynność potencjalnie niebezpieczna!
- `btn-link` — pozwala zmniejszyć znaczenie przycisku, nadając mu wygląd linku, zachowuje jednak znaczenie i semantykę przycisku.

Możemy też modyfikować rozmiary przycisków, dodając klasy:

- `btn-lg` — duży przycisk,
- `btn-sm` — mały,
- `btn-xs` — bardzo mały,
- `btn-block` — zajmujący 100% dostępnej szerokości.

Klasy responsywne

By szybciej tworzyć strony responsywne, Bootstrap daje nam do dyspozycji kilka klas, które w zależności od rozmiaru ekranu wyświetlają lub chowają element, do którego są przypisane.

Aby schować element przy różnych szerokościach ekranu, dodajemy po prostu jedną z klas, w zależności od interesującego nas rozmiaru. Konwencja jest podobna, jak w przypadku nazewnictwa klas gridu, gdzie po myślniku oznaczony jest rozmiar:

- `.hidden-xs` — element ukryty na bardzo małych urządzeniach
- `.hidden-sm` — element ukryty na małych urządzeniach
- `.hidden-md` — element ukryty na średnich urządzeniach
- `.hidden-lg` — element ukryty na dużych urządzeniach

W podobny sposób możemy skorzystać z klas, by nadać elementom wyświetlanie tylko na którymś z dostępnych rozmiarów, w pewien określony sposób.

Te klasy to:

- `.visible-xs-*` — element wyświetla się na bardzo małych urządzeniach
- `.visible-sm-*` — element wyświetla się na małych urządzeniach
- `.visible-md-*` — element wyświetla się na średnich urządzeniach
- `.visible-lg-*` — element wyświetla się na dużych urządzeniach.

W miejscu gwiazdki dodajemy odpowiedni rodzaj wyświetlania: block, inline lub inline-block. Dzięki tym klasom jeden element może mieć, np. różne wyświetlanie na różnych ekranach:

```
<div class="visible-sm-inline visible-md-block">
```

Zadanie: Prosta strona

Twoim zadaniem będzie zbudowanie prostej stronki z małym formularzem kontaktowym i RWD przy wykorzystaniu omówionych elementów i klas pomocniczych.

1. Dodaj do strony jedną sekcję. Wyśrodkuj tekst, dodając odpowiednie klasy. Wewnątrz stwórz nagłówek pierwszego poziomu z podtytułem wewnątrz elementu `<small>`.
2. Do sekcji dodaj przycisk o kolorze niebieskim.
3. Poniżej dodaj jedną sekcję, a w niej nagłówek. Następnie stwórz galerię złożoną z 4 obrazków wewnątrz tego gridu. Niech każdy obrazek będzie responsywny. Dodaj odpowiednie klasy zmieniające szerokość elementu na mniejszych ekranach, tak by na tabletach były 2 obrazki w wierszu, a na smartfonach jeden.
4. Poniżej dodaj tabelkę. Niech składa się z nagłówka `<thead>` oraz ciała `<tbody>`. Wewnątrz dodaj kilka rzędów (co najmniej 3).
5. Stwórz formularz kontaktowy. Musi składać się z co najmniej dwóch pól tekstowych oraz pola na treść wiadomości. Dodaj odpowiednie podpisy do każdego elementu. Nie zapomnij o przycisku "Submit"!

Podgląd zadania

Przejdź do projektu ✔

5.4. Komponenty CSS Bootstrapa

Sugestia

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

Wprowadzenie

Jak już wiesz, Bootstrap jest stosunkowo dużym narzędziem pozwalającym na usprawnienie procesu budowania strony.

Komponenty w Bootstrapie to "klocki", z których budujemy stronę. Są wśród nich elementy takie jak **navbar** (pasek nawigacji), **jumbotron** — atrakcyjne zdjęcie z hasłem i przyciskiem z zachęcającym do podjęcia działania na stronie, tak zwanym *call to action* (CTA), czy też listy.

Jeśli potrzebujemy go tylko w celu dodania na stronie kilku elementów, możemy w projekcie skorzystać jedynie z niezbędnych komponentów, zamiast umieszczać w nim pliki zawierające wszystkie elementy Bootstrapa. Jest to praktyka pozwalająca na zmniejszenie rozmiaru projektu i szybsze ładowanie strony.

W pliku Bootstrapa są domyślnie zawarte wszystkie jego komponenty, ale za pomocą tego narzędzia możliwe jest utworzenie spersonalizowanej wersji zawierającej tylko te rzeczy, których potrzebujemy.

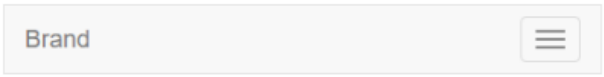
Listę dostępnych komponentów oraz ich dokumentację znajdziesz na stronie Bootstrap Components.

Navbar

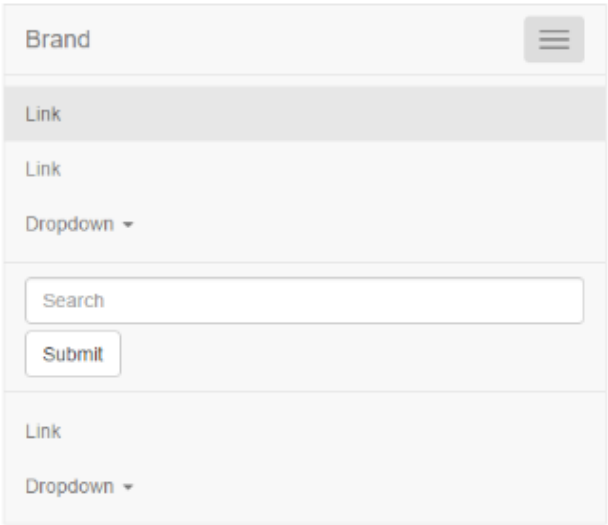
Pasek nawigacyjny można nazwać jednym z podstawowych elementów każdej strony. Dlatego też framework Bootstrap umożliwia nam wiele możliwości konfiguracji tego modułu.

Bootstrapowy navbar jest domyślnie responsywny, co oznacza, że nie musimy stosować `@media query`, aby zmienić jego zachowanie w zależności od rozdzielczości ekranu urządzenia, na którym oglądana jest strona. Do jego prawidłowego działania potrzebna jest biblioteka JS Bootstrapa.

W przypadku urządzeń mobilnych menu ukrywa się i przyjmuje postać tzw. hamburgera, jak na poniższym przykładzie:



Kliknięcie przycisku po prawej stronie spowoduje rozwinięcie opcji dostępnych w menu:



Jak budujemy navbar?

Po pierwsze, musimy dodać element <nav>, w którym umieścimy wszystkie elementy paska. Następnie przypisujemy tam klasę navbar. To doda odpowiednią stylizację.

Użycie klasy `navbar-inverse` spowoduje utworzenie paska nawigacji o odwróconej kolorystyce (ciemne tło z jasnymi napisami). Następnie możemy dodać wewnątrz kontener: może to być zarówno klasa `container`, jak i `container-fluid`.

Jeżeli chcemy dodać nazwę naszej strony, jedyne, co musimy zrobić, to dodać jeden div z klasą `navbar-header`, a wewnątrz niego link z klasą `navbar-brand`. Linki dodajemy jako listę (`<ul>`) z klasą `nav` oraz `navbar-nav`. Wewnątrz listy poszczególne linki dodajemy jako zwykłe elementy `<li>`.

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header">
      <a class="navbar-brand" href="#">WebSiteName</a>
    </div>
    <ul class="nav navbar-nav">
      <li class="active"><a href="#">Home</a></li>
      <li><a href="#">Page 1</a></li>
      <li><a href="#">Page 2</a></li>
      <li><a href="#">Page 3</a></li>
    </ul>
  </div>
</nav>
```

W linku poniżej znajduje się podstawowy przykład nawigacji:
https://codepen.io/kodilla/pen/bb1916cbb7d1e358046264eb4f838dbe

Fixed navbar

W ramach Bootstrapa możliwe jest utworzenie tzw. "fixed navbara", czyli paska nawigacji "przyklejonego" do góry (lub dołu!) ekranu, niezmieniającego swej pozycji wraz z przewijaniem strony.

Służą do tego klasy `navbar-fixed-top` oraz `navbar-fixed-bottom`. W takim przypadku pamiętaj o nadaniu elementowi `<body>` odpowiedniego paddingu odpowiadającego swoją wartością wysokości paska nawigacji.

Dzięki temu prostemu zabiegowi navbar nie będzie przysłaniał zawartości strony.

Gdy spojrzysz na paski nawigacji, to zauważysz, że często niektóre elementy (np. pole logowania) znajdują się po prawej stronie paska. By przesunąć elementy w prawą stronę, wystarczy dodać do elementu klasę `navbar-right` oprócz klas `nav` oraz `navbar-nav` lub `navbar-form`.

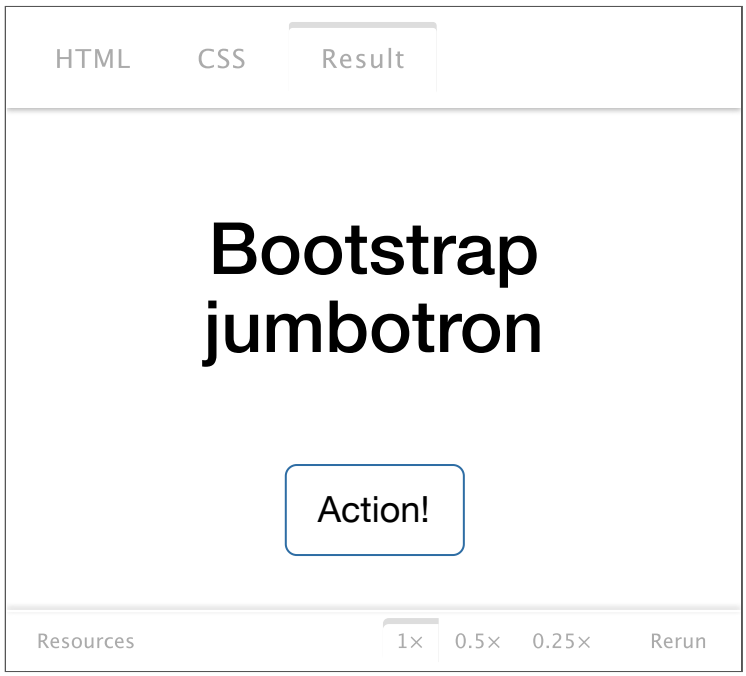
```
<form class="navbar-form navbar-right" role="search">
    <div class="form-group">
        <input type="text" class="form-control" placeholder="Search">
    </div>
    <button type="submit" class="btn btn-default">Submit</button>
</form>
```

Jumbotron

Z jumbotronem zapoznaliśmy Cię niemal na samym początku przygody z tym bootcampem. Bootstrap znacznie ułatwia jego tworzenie, udostępniając do Twojej dyspozycji klasę jumbotron, która domyślnie pozycjonuje swoją zawartość zgodnie z przyjętymi standardami dotyczącymi tego elementu.

Jeśli chcesz, aby jumbotron miał taką szerokość, jak reszta strony, umieść go w divie o klasie `container`. W przeciwnym wypadku w strukturze HTML umieść najpierw div o klasie `jumbotron`, a dopiero za nim `container`. To wyrówna jego zawartość do szerokości kontenera.

W przykładzie poniżej dodaliśmy jedynie gradient w tle jumbotrona oraz odrobinę marginesu, by oddalić od siebie dwa elementy.



Panels

Panele są czymś w rodzaju pudełek w szafce. Możesz w nich umieścić niemal wszystko :) Służą m.in. wyróżnieniu jakiegoś fragmentu tekstu (np. dodanie wyodrębnionej ciekawostki lub dygresji na jakiś temat) lub uporządkowaniu elementów na stronie (lista postów w bocznej szpalcie bloga, tabela z wynikami zawodów itp.).

W zależności od zapotrzebowania, możemy stworzyć panel z nagłówkiem (div o klasie `panel-heading`), stopką (`panel-footer`), zmienić jego kolory w zależności od kontekstu, jak również umieścić w nim tabelę bądź listę.

```
<div class="panel panel-default">
    <div class="panel-body">
        Basic panel example
    </div>
</div>
```

Thumbnails

Pamiętasz galerię tworzoną w module 2? Thumbnails to prosty sposób na stworzenie takiej właśnie galerii miniatur obrazków. W tym celu należy stworzyć div o klasie "row", a następnie kilka kolumn o odpowiedniej szerokości zgodnie z gridem. Potem w każdej z nich dodajemy obrazek o klasie "img-thumbnail". Aby obrazek był responsywny, przypisujemy do niego klasę img-responsive. Z uwagi na to, że umieściliśmy go w elemencie grida, będzie zawsze zajmował określoną w procentach szerokość.

W ten sposób możemy stworzyć bardzo szybko galerię obrazków. Aby stworzyć prosty grid z galerią responsywnych obrazków, wystarczy dodać do odpowiednich elementów klasy:

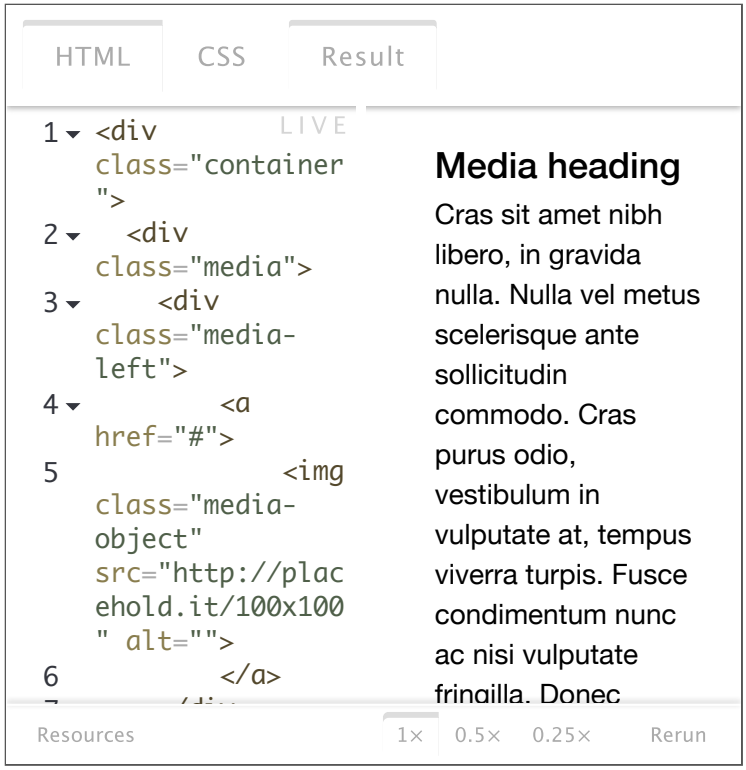
```
<div class="container">
    <div class="row">
        <div class="col-md-3"><img src="http://placehold.it/300x300"
        <div class="col-md-3"><img src="http://placehold.it/300x300"
        <div class="col-md-3"><img src="http://placehold.it/300x300"
        <div class="col-md-3"><img src="http://placehold.it/300x300"
    </div>
</div>
```

Media object

Komponent media objects służy do tworzenia elementów takich jak komentarze na blogach, wpisy itp. składających się ze zdjęcia (avatara) oraz bloku tekstu.

Możliwe jest zagnieżdżanie ich w sobie (np. w celu stworzenia drzewka komentarzy). Są one dość abstrakcyjnymi obiektami — nie mają jednoznacznego przeznaczenia. Poniżej znajduje się przykład, jak możemy wykorzystać je do stworzenia komentarzy na blogu:

```
<div class="media">
    <div class="media-left">
        <a href="#">
            <a class="img-modal-group" href="..." alt="..."><img src=
        </a>
    </div>
    <div class="media-body">
        <h4 class="media-heading">Media heading</h4>
        <p>Cras sit amet nibh libero, in gravida nulla. Nulla vel met
    </div>
</div>
```



List group

Komponent list group to nic innego, jak lista elementów :) Możesz ją skonfigurować, nadać elementom listy nagłówki, umieścić linki, jak i "zablokować" (poprzez wyszarzenie).

Istnieje także możliwość stworzenia listy przycisków. Tworzymy ją za pomocą listy nieuporządkowanej <ul> z klasą `list-group`, a następnie każdemu elementowi przypisujemy klasę "list-group-item".

```
<ul class="list-group">
    <li class="list-group-item">Cras justo odio</li>
    <li class="list-group-item">Dapibus ac facilisis in</li>
    <li class="list-group-item">Morbi leo risus</li>
    <li class="list-group-item">Porta ac consectetur ac</li>
    <li class="list-group-item">Vestibulum at eros</li>
</ul>
```

Możemy też skorzystać z gotowych klas, by dodać tam tzw. badge, czyli mały element informacyjny:

```
<ul class="list-group">
    <li class="list-group-item">
        <span class="badge">14</span>
        Cras justo odio
    </li>
</ul>
```

Spróbujmy teraz nieco zmodyfikować naszą listę. Możemy sprawić, by lista zawierała linki, tak jak w przykładzie poniżej:

```
<div class="list-group">
    <a href="#" class="list-group-item active">
        Cras justo odio
    </a>
    <a href="#" class="list-group-item">Dapibus ac facilisis in</a>
    <a href="#" class="list-group-item">Morbi leo risus</a>
    <a href="#" class="list-group-item">Porta ac consectetur ac</a>
    <a href="#" class="list-group-item">Vestibulum at eros</a>
</div>
```

Zadanie: Komponenty Bootstrapa

W tym zadaniu rozbudujesz prostą stronę z poprzedniego ćwiczenia o kolejne elementy Bootstrapa :)

1. Skopiuj kod z poprzedniego ćwiczenia.
2. Dodaj do strony zrobionej w poprzednim ćwiczeniu pasek nawigacji. Niech składa się z linku z nazwą firmy, kilku pozycji na liście oraz pola z wyszukiwaniem po prawej stronie. Przyklej go do góry strony. Pamiętaj o tym, by dodać odpowiedni padding dla zawartości strony pod spodem.
3. Do sekcji utworzonej w poprzednim zadaniu dodaj klasę jumbotron. Zwróć uwagę, jakie różnice w stylach to powoduje.
4. Poniżej tabeli z poprzedniego ćwiczenia stwórz galerię obrazków, korzystając z klasy thumbnail.
5. Skorzystaj z media group, by stworzyć na stronie link do artykułu. Powinien składać się z nagłówka, obrazka oraz krótkiego tekstu “zajawki".
6. Sprawdź w dokumentacji Bootstrapa dowolny komponent CSS, którego nie omówiliśmy i znajdź dla niego zastosowanie na stronie :)

Podgląd zadania

Przejdź do projektu ✔

5.5. Komponenty JS Bootstrapa

Sugestia

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

Nieco bardziej zaawansowaną kategorią "klocków", które udostępnia nam Bootstrap, są komponenty korzystające z JavaScript. Spokojnie — nie musimy znać JavaScript, by z niego korzystać. Jedyne, co musimy zrobić, to dodać plik z bootstrapowymi skryptami do pliku index.html, a potem dodać odpowiednie klasy i atrybuty do poszczególnych elementów. Po tym zabiegu możemy w pełni korzystać z bardziej wystrzałowych elementów Bootstrapa.

Zanim zaczniesz testować poniższe rozwiązania, dodaj poniższe tagi tuż przed zakończeniem tagu </body>, tak by uruchomić dodatki JavaScript.

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.2/jque
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/boots
```

Uwaga!

W niniejszym submodule znajdują się gotowe rozwiązania zbudowane za pomocą Bootstrapa. Do poprawnego działania wykorzystują one `jQuery` i `bootstrap.min.js`, dlatego pamiętaj o podłączeniu tych skryptów. Wystarczy, że zrobisz to **RAZ** przed tagiem kończącym </body>

Carousel

Popularna karuzela to sposób przedstawiania grafiki za pomocą przewijających się slajdów. Zwróć uwagę na przykład:

Jeśli się przyjrzysz, zobaczysz, że karuzela składa się z kilku elementów. Mamy 4 przewijające się slajdy, strzałki odpowiedzialne za nawigację w lewo i prawo, a na dole wskaźnik pokazujący, na którym z kolei slajdzie się znajdujemy. Zbudujemy teraz wszystkie komponenty.

Standardowo zaczynamy od dodania kontenera z klasą `container`. Wewnątrz musi dodać kolejny z odpowiednimi atrybutami:

```
<div id="myCarousel" class="carousel slide" data-ride="carousel">
```

Aby element poprawnie działał, konieczne jest nadanie mu id, w tym wypadku `myCarousel`, oraz atrybutu `data-ride="carousel"`. Następnie widoczne są dwie klasy `carousel`, która odpowiada za wygląd karuzeli, oraz slide odpowiedzialna za efekt przewijania.

Poniżej widzimy kod odpowiedzialny za nawigację w karuzeli. Atrybut data target musi być równy atrybutowi ID naszej karuzeli. `data-slide-to` odpowiada za przewijanie karuzeli do wskazanego slajdu. Musimy też pamiętać o klasie `carousel-indicators`, by zapewnić odpowiedni wygląd.

```
<ol class="carousel-indicators">
    <li data-target="#myCarousel" data-slide-to="0" class="active"></
    <li data-target="#myCarousel" data-slide-to="1"></li>
    <li data-target="#myCarousel" data-slide-to="2"></li>
    <li data-target="#myCarousel" data-slide-to="3"></li>
</ol>
```

Następnie czas dodać nasze slajdy. Poniżej tworzymy kolejny `<div>` z klasą `carousel-inner` oraz atrybutem `role="listbox"`. Wewnątrz umieścimy slajdy. Każdy z nich to pojedynczy `<div>` z klasą item. Dodaliśmy również klasy odpowiedzialne za wygląd.

```
<div class="carousel-inner" role="listbox">
  <div class="item active example-1">
  </div>
  <div class="item example-2">
  </div>
  <div class="item example-3">
  </div>
  <div class="item example-4">
  </div>
</div>
```

Jedyne, co nam zostało, to dodanie strzałki po lewej i po prawej stronie. Dzięki nim będzie można przewinąć karuzelę do sąsiedniego slajdu.

```
<a class="left carousel-control" href="#myCarousel" role="button" dat
  <span class="glyphicon glyphicon-chevron-left" aria-hidden="true"><
  <span class="sr-only">Previous</span>
</a>
<a class="right carousel-control" href="#myCarousel" role="button" da
  <span class="glyphicon glyphicon-chevron-right" aria-hidden="true">
  <span class="sr-only">Next</span>
</a>
```

Jak widać, to nic innego jak linki, wewnątrz których znajdują się ikony. Jedyne, o czym musimy pamiętać, to odpowiednie klasy (`carousel-control`, `left` oraz `right`), atrybut `href` równy id karuzeli i atrybuty `role="button"` oraz `data-slide`, odpowiednio `next` oraz `prev`.

Nasza karuzela jest gotowa!

Modal

Modal to okno dialogowe, które pojawia się po określonej akcji użytkownika. Możemy go użyć np. do wyświetlenia warunków umowy i zaakceptowania ich przez użytkownika. Sprawdź przykład działania poniżej:

Zacznijmy od przycisku, który będzie otwierał nasz modal:

```
<button type="button" class="btn btn-primary btn-lg" data-toggle="mod
  Launch demo modal
</button>
```

Widzimy tam dodane znane nam już klasy odpowiedzialne za wygląd przycisku. Oprócz tego, widzimy atrybuty niezbędne do kontroli naszego dialogu, czyli `data-toggle="modal"` oraz `data-target="#myModal"`. Data-target musi być równe atrybutowi id, jaki później nadamy naszemu dialogowi. Teraz przejdźmy do stworzenia zawartości naszego okna dialogowego. Umieszczamy je wewnątrz poniższej struktury:

```
<div id="myModal" class="modal fade" role="dialog">
    <div class="modal-dialog">
        <div class="modal-content">
            <!-- our content -->
        </div>
    </div>
</div>
```

W tym wypadku jedyne, o czym musimy pamiętać, to nadanie odpowiedniego id. Pozostałą strukturę pozostawiamy niezmienną i możemy ją skopiować z powyższego przykładu.

Okno pojawiające się po kliknięciu możemy podzielić na trzy sekcje za pomocą klas modal-header, modal-body oraz modal-footer.

```
<div class="modal-header">
    <h4>Modal header</h4>
</div>
<div class="modal-body">
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit. Nisi
</div>
<div class="modal-footer">
    <p>Some text in the footer</p>
</div>
```

Brakuje jeszcze jednego elementu: przycisku do zamykania dialogu. Wystarczy dodać przycisk tak, jak w poprzednich ćwiczeniach, dodając jeszcze atrybut `data-dismiss="modal"`:

```
<button type="button" class="btn btn-default" data-dismiss="modal">Cl
```

W ten nieskomplikowany sposób stworzyliśmy proste okno dialogowe bez konieczności pisania JavaScript.

Karty (tabs)

Większość stworzonych przez nas do tej pory stron korzystała z nawigacji zamkniętej wewnątrz paska nawigacji. Dzięki Bootstrapowi możemy nieco urozmaicić poruszanie się po treści strony, tworząc eleganckie zakładki. Najprostsze zastosowanie wymaga od nas jedynie dodania do listy `<ul>` klasy nav-tabs, tak jak na przykładzie poniżej:

```
<ul class="nav nav-tabs">
  <li class="active"><a href="#">Home</a></li>
  <li><a href="#">Menu 1</a></li>
  <li><a href="#">Menu 2</a></li>
  <li><a href="#">Menu 3</a></li>
</ul>
```

Klasa `.active` wskazuje na element, który akurat został wybrany. Alternatywnie możemy też skorzystać z klasy `nav-pills`, by upodobnić linki do przycisków:

```
<ul class="nav nav-pills">
  <li class="active"><a href="#">Home</a></li>
  <li><a href="#">Menu 1</a></li>
  <li><a href="#">Menu 2</a></li>
  <li><a href="#">Menu 3</a></li>
</ul>
```

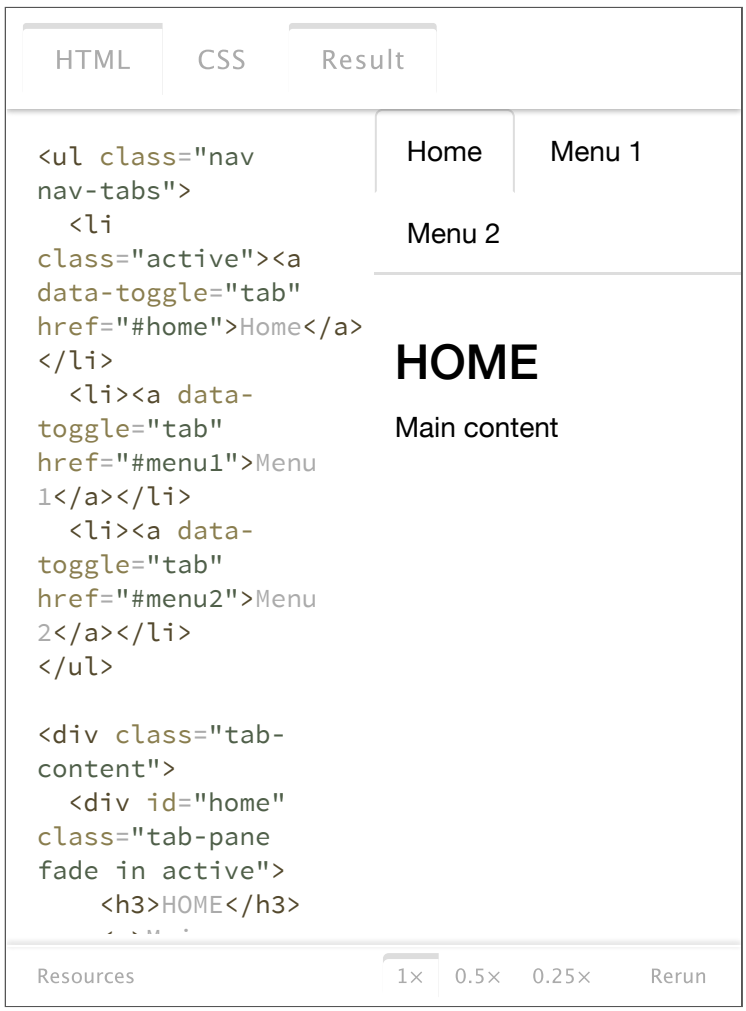
By wyśrodkować elementy i rozciągnąć je na całą szerokość kontenera, wystarczy dodać klasę `.nav-justified`.

Pozostaje nam jedynie sprawić, by nasze menu było interaktywne. Aby to zrobić, dodajemy do każdego linku w liście atrybut `data-toggle="tab"`. Dodajemy też atrybut `href` równy id odpowiedniej zawartości.

Zawartość każdej z zakładek powinna zostać następnie zagnieżdżona wewnątrz oddzielnego elementu div z klasą `.tab-content`. Następnie dodajemy klasę `.tab-pane` wraz z unikalnym id do zawartości każdej zakładki. Aby stworzyć animację, wystarczy dodać klasę `.fade` do elementu z klasą `.tab-pane`. Przykład nawigacji przy użyciu zakładek znajduje się poniżej. Przejrzyj, krok po kroku, jak dodawać odpowiednie klasy i atrybuty.

```
<ul class="nav nav-tabs">
  <li class="active"><a data-toggle="tab" href="#home">Home</a></li>
  <li><a data-toggle="tab" href="#menu1">Menu 1</a></li>
  <li><a data-toggle="tab" href="#menu2">Menu 2</a></li>
</ul>

<div class="tab-content">
  <div id="home" class="tab-pane fade in active">
    <h3>HOME</h3>
    <p>Some content.</p>
  </div>
  <div id="menu1" class="tab-pane fade">
    <h3>Menu 1</h3>
    <p>Some content in menu 1.</p>
  </div>
  <div id="menu2" class="tab-pane fade">
    <h3>Menu 2</h3>
    <p>Some content in menu 2.</p>
  </div>
</div>
```



Collapse

Collapse jest z pozoru bardzo prostym pluginem. Za jego pomocą możemy jednak zbudować kilka bardzo interesujących komponentów. Robi on jedną prostą rzecz: po naciśnięciu np. przycisku możemy schować lub pokazać jakiś inny element.

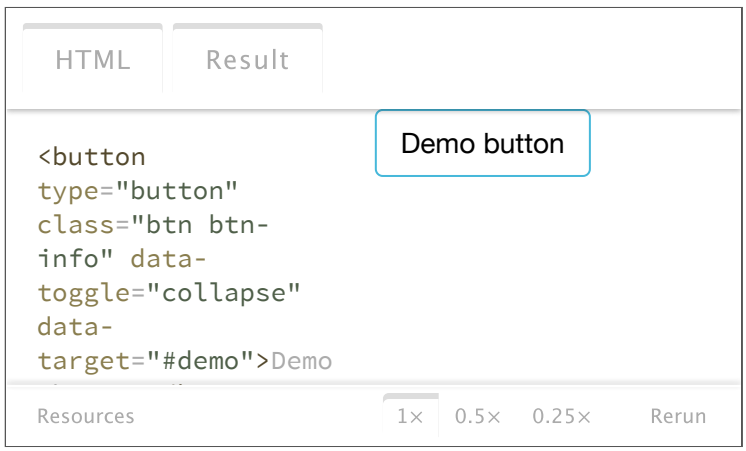
Najprostsze zastosowanie wygląda następująco: tworzymy przycisk z atrybutem `data-toggle="collapse"` . Następnie dodajemy przyciskowi atrybut `data-target` , który powinien być równy atrybutowi ID elementu, który chcemy ukryć lub pokazać. Gotowy przycisk będzie wyglądał następująco:

```
<button type="button" class="btn btn-info" data-toggle="collapse" dat
```

Teraz przejdźmy do markupu elementu, który będzie pokazywany za pomocą przycisku. Jedyne, co musimy zrobić, to dodać mu odpowiedni ID, taki jak w atrybucie `data-target`, oraz klasę `collapse`.

```
<div id="demo" class="collapse">
    Lorem ipsum dolor text....
</div>
```

Domyślne element jest schowany. Aby go pokazać, wystarczy dodać klasę `in`.



Dropdown

Zacznijmy od szkieletu naszego dropdowna. Zamknijmy menu w elemencie `<div>` z klasą `dropdown`.

```
<div class="dropdown">
```

Następnie dodajmy przycisk. Jedyne, o czym musimy pamiętać, to że oprócz klas odpowiedzialnych za wygląd przycisku dodajemy do niego klasę dropdown-toggle oraz atrybut `data-toggle="dropdown"`. Oprócz tego wewnątrz przycisku dodajemy też element `<span>` — będzie to drobna, trójkątna ikonka, która zasugeruje użytkownikowi, że element ten to rozwijane menu.

```
<button class="btn btn-primary dropdown-toggle" type="button" data-to
  <span class="caret"></span>
</button>
```

Ostatnią rzeczą, jaką musimy zrobić, jest dodanie zawartości menu. W tym celu zastosujemy standardowo listę nieuporządkowaną `ul`< wraz z elementami `<li>`. Dodajemy klasę `dropdown-menu`, by zapewnić odpowiedni wygląd. Gotowy dropdown będzie wyglądał zatem następująco:

```
<div class="dropdown">
  <button class="btn btn-default dropdown-toggle" type="button" id="m
  <span class="caret"></span></button>
  <ul class="dropdown-menu" role="menu" aria-labelledby="menu1">
    <li role="presentation"><a role="menuitem" href="#">HTML</a></li>
    <li role="presentation"><a role="menuitem" href="#">CSS</a></li>
    <li role="presentation"><a role="menuitem" href="#">JavaScript</a
    <li role="presentation" class="divider"></li>
    <li role="presentation"><a role="menuitem" href="#">About Us</a><
  </ul>
</div>
```

HTML Result EDIT ON

```
<div class="dropdown">
  <button class="btn btn-default
dropdown-toggle" type="button"
id="menu1" data-
toggle="dropdown">Dropdown
  <span class="caret"></span>
</button>
  <ul class="dropdown-menu"
role="menu" aria-
labelledby="menu1">
    <li role="presentation"><a
role="menuitem" href="#">HTML</a>
</li>
    <li role="presentation"><a
role="menuitem" href="#">CSS</a>
</li>
    <li role="presentation"><a
role="menuitem"
href="#">JavaScript</a></li>
    <li role="presentation"
```

Dropdown

Resources 1× 0.5× 0.25× Rerun

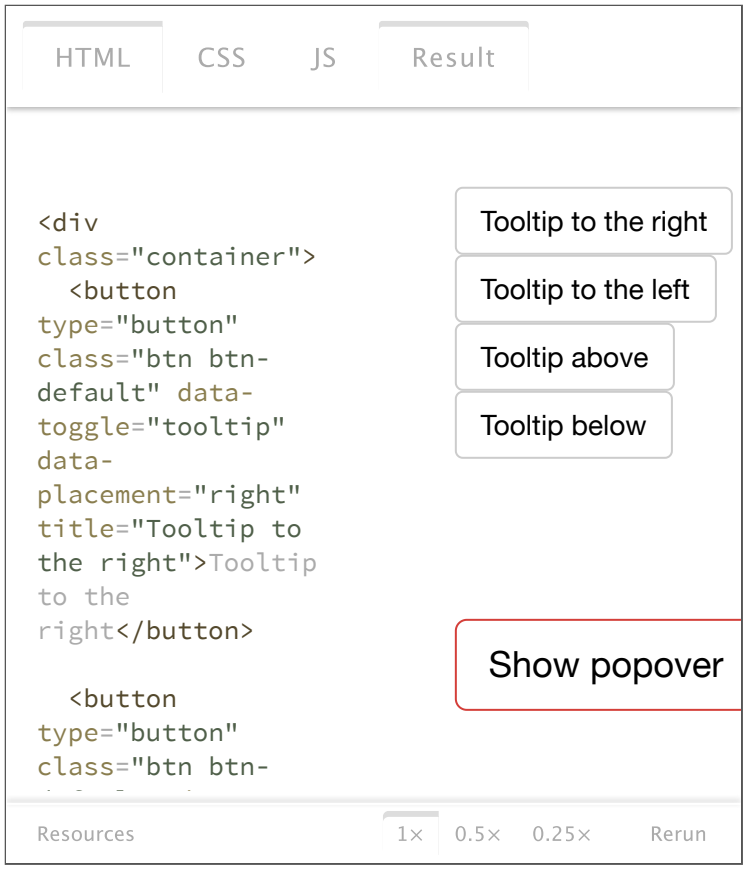
Tooltip/popover

Tooltip to mała chmurka z tekstem, która pojawia się po najechaniu myszką na jakiś element. Stworzenie go jest bardzo proste: wystarczy dodać do elementu atrybut `data-toggle="tooltip"`, zaś w atrybucie `title` wpisać tekst, który ma się wyświetlić po najechaniu na element. Dodatkowo za pomocą atrybutu `data-placement` możemy też określić miejsce, w którym tooltip ma się pojawić.:

```
<a href="#" data-toggle="tooltip" data-placement="top" title="Hooray!
<a href="#" data-toggle="tooltip" data-placement="bottom" title="Hoor
<a href="#" data-toggle="tooltip" data-placement="left" title="Hooray
<a href="#" data-toggle="tooltip" data-placement="right" title="Hooro
```

Nieco bardziej rozbudowanym wariantem jest tzw. popover. Możemy tutaj dodać więcej zawartości oraz podzielić ją na tytuł i treść. Dodajemy standardowo atrybut `data-toggle`, a także atrybuty `title` oraz `data-content` odpowiedzialne za zawartość komponentu.

```
<a href="#" data-toggle="popover" title="Popover Header" data-content
```



Zadanie: Dodajemy komponenty JS

W poprzednim zadaniu udało Ci się stworzyć stronę opartą na gotowych komponentach CSS Bootstrapa. Kolejnym krokiem będzie zbudowanie strony wykorzystującej komponenty JavaScript.

1. Ponownie utwórz pasek nawigacji, umieść w nim logo po lewej stronie oraz listę 3 opcji.
2. Korzystając z komponentu carousel, utwórz karuzelę przedstawiającą 5 zdjęć.
3. Poniżej karuzeli zrób sekcję wykorzystując taby. Użyj minimum trzech tabów i napisz w nich coś o sobie.
4. Dodaj przycisk, którego naciśnięcie spowoduje otwarcie modala. Zawrzyj w nim zwięzłe informacje o autorze oraz tematyce strony.

Podgląd zadania

Przejdź do projektu ✔