Witaj w trzecim tygodniu Bootcampu!

Witaj w kolejnym tygodniu zmagań! Mam nadzieję, że poprzedni był dla Ciebie ciekawy i zawierał przydatne informacje. Drugi tydzień mógł zawierać nieco trudniejsze zagadnienia, ale wierzę, że ich zrozumienie jest kluczowe by skutecznie pracować w językiem JavaScript. W tym tygodniu natomiast poznasz kolejne narzędzia i technologie, które, jak się przekonasz, nie są trudne do opanowania. Będzie to bardzo popularny framework **AngularJS**, narzędzie wspomagające proces tworzenia aplikacji - **Webpack**, a także zobaczysz najważniejsze konstrukcje nowej specyfikacji, na której opiera się język JavaScript - **EcmaScript 2016** (ES6).

Grzegorz Róg

eduweb.pl

--- Zadania na tydzień 3 ---

Wszystkie prace, jak poprzednio, umieść w serwisie GitHub w osobnych repozytoriach.

1. Przepisanie kodu w standardzie ES5 na ES6

Przepisz znajdujący się pod adresem http://pastebin.com/bYKUGkvA fragment kodu (napisany z użyciem EcmaScript 5) na jego odpowiednik w standardzie ES6. Skorzystaj z nowego zapisu **class**. Kod możesz bez obaw testować np. w najnowszej wersji przeglądarki **Google Chrome**, która w pełni wspiera tę konstrukcję.

2. Rozwiązanie problemu z kontekstem dla funkcji

W kodzie, który znajduje się pod adresem http://pastebin.com/sHVGy1Gf występuje pewien problem z kontekstem (patrz funkcja zwrotna w **setInterval**). Wykorzystując pewną nową konstrukcję, dostępną w standardzie **EcmaScript 2015**, rozwiąż ten problem. Aplikacja powinna **10 razy** wyświetlić w konsoli komunikat, zawierający wiadomość z podanym przy tworzeniu nowego obiektu imieniem i nazwiskiem.

3. Blog z użyciem AngularJS

Wykorzystując framework AngularJS (“jedynkę”), stwórz aplikację, która na stronie głównej wyświetlać będzie posty. Każdy post powinien zawierać następujące informacje: **title** i **body**. Utwórz widok tak, aby tytuł był również odnośnikiem kierującym do nowej podstrony aplikacji według schematu **“#/post/{{ post.id }}”**. Po przejściu pod ten adres, powinien się pojawić widok z pojedynczym wpisem, gdzie również wyświetl jego tytuł (**title**) i treść (**body**), a także przycisk “**Powrót do listy wpisów**”, który kierować powinien z powrotem do pełnej listy. Ogranicz liczbę wpisów na stronie głównej do **15**. Jako źródło wpisów wykorzystaj ten adres https://jsonplaceholder.typicode.com/posts Zauważ, że każdy obiekt w tablicy zawiera potrzebne informacje: **title**, **body** i **id**. To właśnie dzięki **id** stworzysz odnośnik do pełnego wpisu. Kiedy natomiast wyświetlana będzie podstrona z pełnym

wpisem, wyślij po niego zapytanie pod adres https://jsonplaceholder.typicode.com/posts/ID gdzie **ID** to liczba spod klucza **id** w obiekcie (zobacz np. https://jsonplaceholder.typicode.com/posts/1).

4. Walidacja formularza z użyciem AngularJS

Stwórz walidator formularza napędzany frameworkiem AngularJS. W formularzu umieść pola: **imię**, **nazwisko**, **email**, **hasło**, a także **checkbox** dotyczący akceptacji regulaminu oraz **przycisk** do wysyłania formularza. Pola wymagane (walidowane) to **imię**, **email**, **hasło** oraz **zaznaczony checkbox**. Aplikacja powinna walidować pola "**na żywo**" w momencie, gdy użytkownik wpisuje dane. Jeśli dane są niepoprawne (np. pole jest puste lub hasło jest za krótkie), pole powinno zostać zaznaczone na czerwono, a przycisk do wysyłania zablokowany. W przeciwnym wypadku pole powinno być zielone, a przycisk aktywny. W przypadku hasła, uznawaj je za poprawne, jeśli ma przynajmniej 5 znaków.

5. Workflow z modułami ES6 i Webpack

Przygotuj swój własny workflow z użyciem Webpacka. Skonfiguruj to narzędzie tak, aby pozwalało na tworzenie kodu aplikacji z podziałem na moduły, importowane następnie z użyciem zapisu dostępnego w ramach **ES6**. Dodatkowo, wszystkie pliki powinny dawać możliwość pisania kodu JavaScript w ES6, a więc finalnie powinny być przez Webpacka kompilowane do standardu ES5, a następnie łączone w tzw. **bundle**. Kod JavaScript dla modułów może być dowolny i nie musi go być dużo. Istotą jest tutaj fakt, aby każdy moduł zawierał jakąś prostą funkcjonalność i ją eksportował. W innym module można ją wtedy zaimportować, a także z niej skorzystać. Poprawnie przygotowany workflow powinien kompilować kod wszystkich modułów z ES6 do ES5, następnie łączyć je w jeden plik i minifikować. Plik ten finalnie wczytywany będzie w pliku **index.html** za pomocą odwołania **<script src=“”></script>**.

Framework AngularJS

Choć język JavaScript dojrzał do tego, by tworzyć za jego pomocą ambitne i rozbudowane aplikacje internetowe, to pisanie kodu “od zera” zazwyczaj nie jest najlepszym rozwiązaniem. W miarę tego jak tworzona aplikacja rośnie, pojawiają się problemy z czytelnością i organizacją kodu, a także wiele nieoczekiwanych błędów.

Z tego powodu “rynek” JavaScriptowych bibliotek i frameworków przeżywa dziś burzliwy, acz bardzo owocny okres. Z jednej strony ilość powstających codziennie rozwiązań może przytłaczać, a z drugiej strony wśród tak wielu z nich, co jakiś czas pojawia się jedno, które zyskuje ogromną popularność. Tak stało się z frameworkiem **AngularJS** stworzonym przez firmę Google. Dzięki AngularJS nadasz tworzonym aplikacjom strukturę, a także wykorzystasz “magię”, która kryje się za tym frameworkiem. AngularJS wpasowuje się w architekturę **MVC** czyli **Model View Controller**, lecz robi to w nieco inny sposób, niż wiele wcześniejszych rozwiązań.

Rozpoczęcie tworzenia aplikacji z użyciem tego narzędzia jest proste, gdyż pozwala ono “rozszerzać” funkcjonalności języka HTML. Z tego powodu nie trzeba uczyć się nowego języka do zapisu widoków, a jedynie specjalnych poleceń (tzw. dyrektyw), które odpowiednio użyte w standardowym kodzie HTML, zamienią go w dynamiczny język.

Z dowolną sekcją kodu HTML (widokiem) połączyć można dane (model), a także kontroler (kod JavaScript zarządzający widokiem i modelem). Umiejętne połączenie wbudowanych funkcjonalności AngularJS z własnym kodem JavaScript sprawia, że tworzenie aplikacji internetowych typu **SPA** (**Single Page Application**) staje się dużo prostsze i szybsze niż z użyciem innych frameworków.

EcmaScript 2015

Język JavaScript opiera się na specyfikacji **EcmaScript**. Choć jest ona ciągle rozwijana, to przez wiele lat nie obserwowaliśmy przełomowych zmian w tym języku. Aż do **czerwca 2015** roku kiedy pojawiła się nowa wersja EcmaScript oznaczona numerem 2015. Poprzednia edycja nazywana była **EcmaScript 5**, a więc nowa szybko zyskała miano **ES6** i pod tymi dwoma nazwami (**ES6 i EcmaScript 2015**) kryje się dziś dokładnie to samo.

I choć komitet **TC-39**, który pracuje nieustannie nad nowym standardem języka, zapowiedział, że nowa wersja będzie wydawana co roku, to nie należy się spodziewać aż tak wielkich, jednoczesnych zmian, jak miało to miejsce w przypadku ES6. Co zatem nowego wnosi ta edycja?

Przede wszystkim kilka rozwiązań, które określane są w żargonie programistycznym jako tzw. "**syntactic sugar**". Oznacza to, że dodany został pewien nowy zapis (syntax), który pozwala zrealizować to samo zadanie, ale w inny sposób. I tak do JavaScriptu zawitały **klasy**, choć wciąż opierają się one na dziedziczeniu prototypowym i brakuje im prywatności. I choć sam ich zapis łudząco przypomina rozwiązania znane z innych języków programowania, to wciąż nie można określić JavaScriptu językiem, który implementuje klasyczny model programowania zorientowanego obiektowo.

Kolejną nowością są tzw. "**arrow functions**", znane m. in. z języka **CoffeScript**. Pozwalają one na tworzenie funkcji anonimowych bez użycia słowa kluczowego **function**, a także na zwracanie wartości bez użycia słowa **return**. Rozwiązują one również pewne problemy z kontekstem wykonania, które nierzadko musiały być obchodzone w standardowych funkcjach.

Istotnych zmian doczekały się również łańcuchy znaków, które notorycznie łączone były za pomocą operatora **+** by "skleić" statyczny tekst z dynamicznymi danymi. Od teraz możliwa jest tzw. interpolacja, a więc umieszczanie wyrażeń JavaScriptowych bezpośrednio wśród innych znaków stringu. Nowy zapis pozwala także na tworzenie tekstu rozciągającego się na wiele linijek, co nie było możliwe wcześniej bez konkatenacji.

Warto wspomnieć również o tym, że słowo kluczowe **var**, służące do tworzenia zmiennych, doczekało się nowych przyjaciół. Pojawiły się dwa nowe słowa - **let** oraz **const**. To pierwsze pozwala tworzyć zmienne "widoczne" wyłącznie w danym bloku kodu. Oznacza to, że zmienna może być utworzona w bloku instrukcji warunkowej **if** i nie będzie dostępna poza tym blokiem. Słowo **const** pozwala natomiast tworzyć stałe. Można o nich myśleć tak samo jak o zmiennych z tą różnicą, że nie można zmienić ich wartości. Nie oznacza to jednak, że gdy do stałej przypiszemy jakiś obiekt, to staje się on niemodyfikowalny. W praktyce oznacza to jedynie, że nie można nadpisać wartości takiej stałej w dalszej części kodu, gdyż próba takiego działania wygeneruje błąd.

Jeszcze jednym z ciekawych rozwiązań, którego od zawsze brakowało w języku JavaScript, jest możliwość przypisywania **domyślnych wartości parametrów**. Dzięki temu definiując funkcję, możemy ustawić wybrany parametr na jakąś domyślną wartość, która zostanie użyta, jeśli argument nie zostanie przekazany przy wywołaniu tej funkcji.

To tylko niektóre z wielkich zmian, jakie zawitały do języka JavaScript dzięki ES6. I choć wsparcie dla tej specyfikacji w najnowszych przeglądarkach internetowych jest bardzo dobre, to aby w pełni cieszyć się poprawnym działaniem aplikacji w jak największej liczbie przeglądarek i środowisk, kod napisany w użyciem ES6 **kompiluje** się do kodu w standardzie ES5. I choć na pierwszy rzut oka to zadanie może się wydawać bezsensownym, to po wypróbowaniu nowych możliwości języka, obraz ten się zmienia. Dzięki takim narzędziom jak np. Babel, możemy cieszyć się możliwością pisania w ES6 i nie martwić się o to czy nasz kod zadziała w środowiskach, które nie obsługują w pełni (lub w ogóle) tego standardu.

Webpack

Choć rozwiązania takie jak np. **AngularJS** czy nowy standard języka **ES6** pozwalają tworzyć lepsze, czytelniejsze i bardziej zorganizowane aplikacje, to wciąż pewne czynności (oprócz pisania kodu) deweloper wykonywać musi. Wszystkie powtarzalne zadania można jednak zautomatyzować. Służą to tego tzw. **automatory zadań**. Narzędzie **Webpack** jest jednym z nich, choć jego główną ideą jest tworzenie tzw. **bundli** kodu.

W praktyce oznacza to, że aplikację podzielić możemy na wiele modułów, według kilku standardów (**AMD**, **CommonJS** czy zapisu modułów z **ES6**). Każdy moduł może eksportować jakieś funkcjonalności, które następnie w innym module mogą być zaimportowane i wykorzystane. To rozwiązanie znane jest w wielu językach programowania. Importujemy wyłącznie te moduły, z których korzystać będziemy w danej części aplikacji.

I choć samo dzielenie aplikacji na mniejsze części mogłoby się wydawać zadaniem stosunkowo prostym, to problem pojawia się wtedy, gdy w przeglądarce należy wczytać wszystkie te zależności. Robienie tego ręcznie poprzez wstawianie niezliczonej ilości znaczników **<script>** nie jest zadaniem należącym do przyjemnych i wydajnych. Dzięki Webpackowi proces ten można uprościć i zautomatyzować. Tak naprawdę wstawianie

wielu skryptów również nie jest najlepszą praktyką, dlatego Webpack zadba o to, by wiele mniejszych modułów odpowiednio połączyć i wczytać na stronie jako jeden plik (nie jest to regułą, plików może być więcej) określany mianem bundla.

W procesie budowania takiej “paczki” z kodem, może on zostać przetransformowany. Transformacje, jakich możemy dokonać, to np. kompilacja kodu napisanego w ES6 do ES5 z użyciem Babel czy minifikacja tego kodu. Dzięki odpowiedniej konfiguracji Webpacka, za pomocą jednego polecenia wygenerowany zostanie tzw. **build** aplikacji, który może być gotowy do wstawiania jej do produkcji.

Zaletą Webpacka jest również fakt, że posiada on rozbudowany system pluginów (loaderów). Dzięki temu społeczność ciągle tworzy dla tego narzędzia nowe wtyczki, rozwiązujące dany problem. Stąd oprócz rozwiązań pracujących wyłącznie na kodzie JavaScript, znajdziemy również takie, które pozwolą np. skompilować kod **SASS** do **CSS** czy zoptymalizować obrazki.

Webpack to potężne narzędzie, które w odpowiedni sposób użyte, pozwoli Ci w mgnieniu oka wykonać ogrom powtarzalnych czynności, oszczędzając Twój czas i energię.