Witaj w szóstym tygodniu Bootcampu!

Witaj w kolejnym, ostatnim już tygodniu zmagań! Mam nadzieję, że poprzedni był dla Ciebie ciekawy i zawierał przydatne informacje. W tygodniu szóstym poznasz przydatne API dostępne w przeglądarkach internetowych, które tworzone są pod wspólną nazwą **HTML5**, a także kilka przydatnych narzędzi. Zobaczysz m. in. jak korzystać z geolokalizacji, jak przechowywać dane po stronie klienta, wykorzystywać **WebSockets** czy pracować z multimediami. Dowiesz się również, jak cały swój workflow zautomatyzować z użyciem narzędzi takich jak **Gulp** i **Grunt**. By lepiej je zrozumieć, poznasz również najważniejsze koncepcje platformy **Node.js**. Powodzenia!

Grzegorz Róg

--- Zadania na tydzień 6 ---

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

1. Link do mapy z położeniem użytkownika

Wykorzystaj API **Geolokalizacji**, by stworzyć odnośnik do map **Bing**, który otworzy mapę z aktualnym położeniem użytkownika. Stwórz na stronie internetowej przycisk, po kliknięciu którego pobrane zostanie położenie odwiedzającego. Z uzyskanych danych wyłuskaj **latitude** i **longitude**, a następnie wstaw je odpowiednio w następujący URL: http://bing.com/maps/default.aspx?cp=LAT~LON uzyskując w ten sposób np. http://bing.com/maps/default.aspx?cp=52.162050~21.071350
Na koniec, wyświetl użytkownikowi link, po kliknięciu którego zostanie przeniesiony pod wcześniej skonstruowany adres.

2. Abstrakcyjna baza danych z użyciem localStorage

Korzystając z API **localStorage**, stwórz kolejną warstwę abstrakcji, która pozwoli pracować z tym interfejsem w taki sposób, aby możliwe było tworzenie nazywanych baz danych. Dane powinny być zapisywane poprzez ich konwersję na **JSON**, a odczytywane poprzez parsowanie. Przykładowe użycie takiej bazy danych może wyglądać następująco: http://pastebin.com/6nntRLRn

3. Czat z użyciem Node.js i WebSockets

Stwórz czat, który wykorzystywał będzie serwer utworzony za pomocą platformy **Node.js**, a także protokół **WebSockets**, który umożliwi dwukierunkowe przesyłanie danych. Daj użytkownikom możliwość podania swojego pseudonimu. Zaimplementuj również wyświetlanie statusów, gdy ktoś dołączy do czata lub czat opuści. Wszystkie przesyłane wiadomości, a także statusy, powinny być widoczne dla wszystkich podłączonych klientów.

4. Odtwarzacz filmów

Stwórz odtwarzacz filmów, który udostępni podstawowe funkcjonalności. Będą to możliwość odtwarzania, pauzowania, wyświetlania paska postępu (który umożliwi również przewijanie). Wyświetl również czas trwania całego filmu, a także aktualny czas postępu. **Porada**: aby ułatwić sobie życie, jako pasek postępu wykorzystać możesz element **<input>** o type range: bit.ly/input-range

5. Workflow z użyciem Gulpa lub Grunta

Z użyciem wybranego automatora zadań, stwórz swój własny workflow, który ułatwi codzienną pracę przy tworzeniu stron lub aplikacji internetowych. Utwórz zadanie **watch**, które pozwoli śledzić zmiany w wybranych plikach i np. kompilować kod **SASS** do kodu **CSS**, dodawać prefiksy (np. **-webkit-**) czy odświeżać widok w przeglądarce. Dodatkowo utwórz zadanie **build**, które pozwoli zminifikować i połączyć wiele plików JavaScript w jeden i dzięki temu utworzyć produkcyjną wersję aplikacji. Zadania jakie stworzysz zależą od Ciebie!

Porada: najlepiej poznać obydwa z prezentowanych narzędzi, ale jeśli masz czas wyłącznie na jedno, zacznij od **Gulpa**, gdyż jest prostszy w opanowaniu.

UWAGI

Przy wykonywaniu powyższych zadań posłużyć się możesz zdobytą wcześniej wiedzą, np. wykorzystując bibliotekę **jQuery** tam, gdzie ułatwi Ci ona pracę. Podobnie możesz skorzystać np. z **EcmaScript 2015** czy np. frameworka **AngularJS**. Oczywiście nie jest to wymagane, ale jeśli tylko chcesz, przećwicz połączenie tych technologii z nowymi zagadnieniami z tygodnia szóstego.

Język JavaScript vs Środowisko uruchomieniowe

Na tym etapie powinieneś już dobrze znać język JavaScript, a także rozumieć czym różni się **Obiektowy Model Dokumentu** (DOM) od samego języka JavaScript (implementowanego zgodnie ze standardem **EcmaScript**). Jeżeli tak jest, to zapewne na pytanie “Czy w języku JavaScript da się zrobić X?” odpowiedziałbyś “To zależy”.

Wszystko bowiem zależy od tego, w jakim środowisku wykonywane będą pisane przez Ciebie skrypty. Wiesz być może, że w “czystym” JavaScripcie, a więc tym zgodnym ze specyfikacją EcmaScript, nie mamy funkcji pozwalających np. na otwieranie i zapisywanie plików. Ale kiedy wykonamy program napisany w tym języku w środowisku Node.js, to już takie możliwości się pojawiają. Podobnie jest z AJAXem, realizowanym za pomocą obiektu **XMLHttpRequest**. Nie jest on częścią samego języka JavaScript, a wyłącznie jednym z API, dostępnym w przeglądarkach internetowych.

Widzisz zatem, że sam język to po prostu zapis (syntax), a także np. typy dostępnych danych (string, number, array) i jakieś podstawowe funkcje. To natomiast, co możemy za jego pomocą zrealizować, w tym przypadku zależy głównie od środowiska uruchomieniowego i jego API.

Moc drzemiąca w nowych API HTML5

Skoro wiesz już jak działa sam język JavaScript i jak pisać w nim programy czy proste skrypty, pozostało poznać przydatne API, z których za jego pomocą możesz korzystać.

Dobra wiadomość jest taka, że w liczba dostępnych w przeglądarkach internetowych API jest bardzo duża i z każdym miesiącem rośnie. Oznacza to, że używając języka JavaScript możesz np. pobrać aktualną lokalizację użytkownika, umożliwić mu odtwarzanie multimediów czy np. zapisywać dane po stronie klienta nie korzystając z ciasteczek. To tylko niektóre z wielu dostępnych API, skupionych wokół standardu HTML5. Niektóre są bardzo proste w użyciu, inne wymagają nieco większych nakładów pracy, ale łączy je jedno - pracujemy z nimi za pomocą języka JavaScript!

JavaScript poza przeglądarką internetową

Język JavaScript powstał po to, aby do statycznych stron internetowych pisanych w użyciem HTML i CSS dodać nieco interaktywności. Przez wiele lat miał z tego powodu łatkę niepoważnego języka programowania, którą w ostatnich latach udało się jednak odkleić. Wraz z rozwojem tego języka, a także rosnącą liczbą osób, które z niego korzystają, powstawały różne pomysły na to, by wykorzystać go poza środowiskiem przeglądarki internetowej.

Najskuteczniejszym z nich okazał się projekt **Node.js**, wykorzystujący silnik do interpretowania kodu JavaScript, napisany przez firmę Google dla przeglądarki Chrome. Silnik ten nosi nazwę **V8**. Okazało się, że wystarczy dopisać do niego kilkanaście modułów, opakować wszystko odpowiednio i wypuścić w formie niezależnej platformy. Tak właśnie powstało w **2009** roku środowisko Node.js.

Platforma ta pozwala wykonywać kod JavaScript poza przeglądarką internetową. Ma to swoje niewątpliwe zalety, gdyż pozwala tworzyć kod uruchamiany na serwerze. Należy tutaj rozumieć, że nie mamy wówczas do dyspozycji tych API, które są dostępne w przeglądarkach i na odwrót. Dzięki Node.js możemy np. zapisać plik na dysku, a API przeglądarek, ze słusznych powodów, nie pozwalają nam tego zrobić.

Node.js okazał się natomiast przydatny nie tylko do pisania kodu serwerowego, ale także do tworzenia niezliczonej ilości narzędzi, wspomagających pracę deweloperów. Z wielu z nich już do tej pory korzystałeś/aś (np. webpack). Warto zatem w pracy Front-end Developera wykorzystywać również Node.js, nawet jeśli nie jesteśmy zainteresowani pisaniem typowego backendowego kodu.

Automatory zadań

Podczas trwania tego Bootcampu wykorzystywaliśmy już takie narzędzie jak **webpack**. Choć najprościej mówiąc, jest to tzw. **bundler** kodu, to można go również określić mianem **automatora zadań**. Wykonuje on bowiem pewne powtarzalne czynności w sposób zautomatyzowany.

Nie w każdym jednak projekcie webpack będzie niezbędny. W wielu z nich wykorzystać możemy takie narzędzia jak **Gulp** czy **Grunt**, a często można je również połączyć z webpackiem. Te narzędzia świetnie sprawdzą się do zadań takich jak kompilacja kodu **SASS do CSS**, dodawanie prefiksów, automatyczne odświeżanie strony w przeglądarce, łącznie wielu plików JavaScript w jeden, minifikacja kodu HTML, wstawianie styli CSS do nagłówka **<head>**, optymalizacja obrazków czy wreszcie wysyłanie kodu na serwer FTP.

Jak się przekonasz, za pomocą ogromnej liczby pluginów, automatory te można niemal dowolnie skonfigurować. Istotne jest tutaj to, aby zadać sobie pytanie “Jaką czynność muszę wykonywać ręcznie, a mogłaby się wykonywać sama?”. Dla każdej takiej czynności znajdziemy jakieś rozwiązanie, które będzie można odpowiednio skonfigurować. W ten sposób stworzysz swój wymarzony workflow, który pozwoli Ci się skupić wyłącznie na pisaniu kodu!

Warto również zapamiętać, że zarówno webpack, Gulp jak i Grunt, to narzędzia napisane w języku JavaScript i uruchamiane w obrębie środowiska **Node.js**!