bootcamp online

JAVASCRIPT

Witaj w szóstym tygodniu Bootcampu

Przed Tobą ostatni tydzień nauki w Bootcampie JavaScript. Mam nadzieję, że poprzedni tydzień był dla Ciebie ciekawy oraz zawierał wiele cennych informacji. W tym tygodniu kontynuujemy poznawanie najnowszej edycji języka JavaScript. Dowiesz się między innymi jak pracować z modułami, jak upewnić się, że Twój nowoczesny kod zadziała również w starszych przeglądarkach, a także co nowego w standardach **EcmaScript 2016** i **2017**. Spokojnie. One nie wnoszą już tak wielu zmian jak edycja z 2015 roku.

Piotr Palarz

Piotr Palarz

eduweb.pl

--- Zadania na tydzień 6 ---

*Zadania domowe spakuj ZIPem i umieść na kanale **#prace_domowe** na Slacku.*
W przypadku zadań, gdzie niezbędny jest dodatkowy kod HTML i CSS, nie musisz się skupiać na tym, aby strona wyglądała pięknie. Jeśli masz czas i chęci, dopracuj swój projekt, ale najważniejszy pozostaje i tak kod JavaScript.

1. Wiązanie elementów DOM z danymi z użyciem WeakMap

Pamiętasz funkcję `getJSON`, którą stworzyłeś w tygodniu trzecim? Za jej pomocą pobierz dane JSON z tego adresu: http://code.eduweb.pl/bootcamp/json/. Następnie w funkcji callback, gdzie te dane będą już zamienione na tablicę JavaScript, dla każdego elementu tej tablicy (będzie to obiekt) utwórz dynamicznie element `<li>` i wstaw go na stronę, do wcześniej utworzonego elementu `<ul>`. W elemencie `<li>` wstaw na początku wyłącznie wartość `name` z obiektu, ale przypisz też do tego elementu obsługę zdarzenia `click`. Kiedy element zostanie kliknięty, zamień jego wewnętrzny tekst na `name` oraz `email`, np. w ten sposób:

```
<li>Leanne Graham <a href="mailto:Sincere@april.biz">Sincere@april.biz</a></li>.
```

Być może zastanawiasz się, jak powiązać element `<li>` z obiektem, który przechowuje potrzebne dla niego dane. W tym celu wykorzystaj `WeakMap`, gdzie jako klucz podasz element z drzewa DOM, czyli `<li>`, a jako wartość obiekt z danymi. Dzięki temu po kliknięciu na element `<li>`, w funkcji obsługi tego zdarzenia pod `this` znajdzie się odwołanie do tego elementu DOM. Kiedy przekażesz `this` do `WeakMapy`, otrzymasz w zamian obiekt z pozostałymi danymi. To z nich wyciągniesz `name` i `email`, aby zaktualizować tekst elementu `<li>`. Plusem tego rozwiązania jest fakt, że jeśli element `<li>` zostanie usunięty, to również obiekt z danymi zostanie usunięty z pamięci, bo takie zadanie realizuje właśnie `WeakMap`.

2. Polyfill dla metody Array.from

Napisz polyfill dla metody `Array.from`, korzystając **wyłącznie** z zapisu **ES5**. W nowoczesnych przeglądarkach ta metoda jest zaimplementowana, a zatem dodając własną, poprzedź ją przedrostkiem `my`, w ten sposób `Array.myFrom`. Metoda ta pozwala

zamienić obiekt tablicopodobny, a więc zawierający numerowane indeksy i właściwość `length`, na pełnoprawną tablicę. Metodę tę będziesz mógł zatem przetestować np. na zmiennej `arguments` lub na obiekcie `NodeList`, który zwrócony zostanie z metody `document.querySelectorAll()` kiedy wyszukasz w drzewie DOM wybranych elementów HTML.

Zwróć uwagę, że metoda `Array.from` przyjmuje **do trzech** argumentów. Jeśli jako drugi argument podana zostanie funkcja, to powinna być ona przekazana do metody map, a więc każdy element tej tablicy powinien być "przepuszczony" przez tę funkcję. Z kolei ostatni argument pozwala na ustawienie kontekstu dla tej funkcji mapującej, a więc tego, co znajdzie się pod słowem `this` wewnątrz tej funkcji.

Przykładowe użycie tego kodu powinno wyglądać następująco: https://pastebin.com/q4MC6zXJ

3. Praca z modułami ES6

Utwórz kilka niezależnych modułów i z każdego z nich coś wyeksportuj, w dowolny, wybrany przez Ciebie sposób. Niech jeden z modułów zawiera jakąś przykładową klasę z prostymi funkcjonalnościami, drugi inną klasę, która mogłaby dziedziczyć z pierwszej, a trzeci (główny moduł) powinien z tych eksportowanych funkcjonalności móc skorzystać.

Możesz utworzyć dowolny kod, ale jeśli nie masz pomysłu, to wykorzystaj kod przepisany na standard ES6 z **zadania trzeciego** z poprzedniego tygodnia. W jednym module umieść klasę `EventEmitter`, a w innym klasę `Database`. W module głównym utwórz nową instancję z klasy `Database` tak, jak to miało miejsce we wspomnianym przykładzie. Aby moduły działały poprawnie, wykorzystaj element `<script type=“module">`.

Uwaga: pamiętaj, że konieczne może być ustawienie odpowiedniej flagi, np. w przeglądarce Chrome po przejściu pod adres `chrome://flags` włącz *Experimental Web Platform features* poprzez kliknięcie *Enable*.

4. Funkcja asynchroniczna dla ładowania obrazów

Przepisz utworzony w poprzednim tygodniu w **zadaniu piątym** kod tak, aby korzystał z funkcji asynchronicznej (`async await`). Oczywiście nadal będzie ona współpracować z `Promise`, ale samo jej użycie będzie inne niż w przypadku `.then` i `.catch`. Zadbaj również

o to, aby w ciele funkcji asynchronicznej znalazł się blok `try...catch()`, który przechwyci ewentualne błędy, gdy `Promise` zakończy się w stanie *rejected*.

5. Bundlowanie modułów z webpack i praca z Babel

Z napisanego w **zadaniu trzecim** kodu (lub innego, jeśli chcesz) utwórz z pomocą **webpacka** bundle, a więc jeden plik, który zawierał będzie wszystkie potrzebne moduły. Dodatkowo skonfiguruj **babel-loader** tak, aby kod napisany w standardzie **ES6**, został przetranspilowany do kodu w standardzie **ES5** (**babel-preset-es2015**), a także zminifikowany.

Przed umieszczeniem plików dla tego zadania na Slacku, usuń katalog `node_modules`, ale pozostaw plik `package.json`, w którym wylistowane powinny być wszystkie potrzebne zależności tego projektu. Dzięki temu jednym poleceniem (`npm install`) będzie można zainstalować te zależności.

Nowe kolekcje

W poprzednim tygodniu poznałeś zdecydowaną większość nowości, które wniósł do języka JavaScript nowy standard **EcmaScript 2015**. To jednak nie wszystko!

Wraz z ES6 pojawiły się nowe konstrukcje, które pozwalają na przechowywanie danych. Są to `Set`, `WeakSet`, `Map` oraz `WeakMap`. `Set` podobny jest do tablic, gdyż pozwala na przechowywanie dowolnych danych, ale co ciekawe, gwarantuje ich unikalność. Jeśli zatem spróbujemy drugi raz dodać do setu tą samą wartość, nie zostanie tam ona umieszczona. Łatwiej również pracuje się z usuwaniem danych z setu. W przypadku tablic należałoby skorzystać np. z metody splice, co wymaga wcześniej określenia indeksu usuwanego elementu z użyciem metody indexOf. W przypadku `Set`, problem rozwiązujemy prostą metodą `delete`.

`Map` z kolei podobny jest do obiektów, gdyż umożliwia przechowywanie danych w postaci par **klucz/wartość**. W zwyczajnych obiektach można jednak podawać jako klucz jedynie `String`, `Number` lub (od ES6) `Symbol`. W przypadku mapy, kluczem może być dowolna wartość, np. jakiś obiekt. Takie zachowanie umożliwia dodawanie tzw. metadanych do obiektów, bo możemy w mapie pod kluczem, którym jest dany obiekt, zapisać jakieś inne dane. Dzięki temu te dane skojarzone są z obiektem wyłącznie poprzez mapę. Kiedy przekażemy ten obiekt do mapy, otrzymamy z powrotem zapisane wcześniej pod tym kluczem dane.

`Set` i `Map` mają również swoje pochodne w postaci `WeakSet` i `WeakMap`. Ich najważniejszą cechą jest fakt, że jeśli przechowywane w nich dane zostaną “na zewnątrz” utracone, a więc nie będziemy mieli do nich już żadnej referencji, to będą one automatycznie usuwane z tych kolekcji. Dzięki temu możemy np. w przypadku `WeakMap`, jako klucz podać jakiś element z drzewa DOM, a jako wartość np. obiekt z danymi. W ten sposób skojarzymy dane z obiektem DOM, ale kiedy ten obiekt zostanie usunięty, to i dane znikną z `WeakMapy`, a co za tym idzie z pamięci operacyjnej.

Moduły

Jedną z najlepszym rzeczy, jaka zawitała wreszcie wraz z ES6 do języka JavaScript, są moduły. To możliwość dzielenia kodu na mniejsze porcje (pliki), gdzie z każdego takiego modułu można coś wyeksportować, a w innym module to zaimportować. Dzięki temu każdy moduł może zawierać swoje prywatne, lokalne zmienne, funkcje, klasy itd., a dopiero później eksportować wyłącznie to, co powinno być widoczne "na zewnątrz".

Przez lata społeczność programistów JavaScript implementowała wiele wzorców, które pozwalały na pracę z modułami. Najpopularniejsze z nich to **CommonJS**, **AMD** czy **UMD**. Nie mieliśmy jednak natywnego, zapisanego w specyfikacji EcmaScript sposobu, na realizację modułowości. Od wersji ES6 wszystko się jednak zmieniło, a sama praca z kodem JavaScript jest dzięki temu dużo przyjemniejsza.

Problemem pozostaje jednak fakt, że specyfikacja EcmaScript mówi jak ma wyglądać zapis modułów i jak mają się one zachowywać, ale nie mówi, jak powinna wyglądać ich implementacja w różnych środowiskach. Z tego powodu przez długi czas moduły nie były zaimplementowane w przeglądarkach internetowych. Jakiś czas temu jednak, w specyfikacji HTML pojawiła się kwestia modułów, a twórcy przeglądarek powoli tę funkcjonalność implementują. Realizowana jest ona poprzez nowy typ dla elementu `<script>`, mianowicie "`module`". Z uwagi na fakt, że konstrukcja ta jest bardzo słabo wspierana, a w większości przeglądarek dostępna dopiero po ustawieniu w opcjach odpowiedniej flagi, praca z modułami w praktyce, wygląda zupełnie inaczej.

I choć możemy wykorzystywać wszystkie dobrodziejstwa płynące z pisania modułów w ES6, to aby poprawnie załadować je do przeglądarki internetowej, najlepiej "zbundlować" je do jednego pliku. W takiej operacji pomoże nam np. narzędzie webpack. Rozumie ono zapis modułów ES6, a zatem możemy je tworzyć, a następnie są one odpowiednio łączone w jeden plik, który podłączamy standardowo do przeglądarki.

ES2016+

Choć standard EcmaScript 2015 przyniósł naprawdę bardzo wiele zmian, to kolejne edycje tego języka są już dużo mniejszym krokiem. Twórcy specyfikacji założyli, że nowa wersja standardu powinna być wydawana co roku. Z tego powodu w 2016 roku pojawiła się wersja

EcmaScript 2016, która wniosła tylko 2 niewielkie nowości. W roku 2017 planowane jest wydanie kolejnej wersji, która znowu nie wprowadza zbyt wielu nowych konstrukcji. Dzięki temu twórcy przeglądarek mogą szybko implementować nowości, a my - deweloperzy - możemy szybko je przyswoić.

Jeśli zatem dobrze zrozumiesz większość nowości wprowadzonych wraz z ES6, to nauka każdej kolejnej edycji, sprowadzi się do przeczytania krótszego lub dłuższego wpisu na blogu i kilku eksperymentów z kodem.