**1. Polyfill metody repeat dla String**

W standardzie **EcmaScript 2015** (o którym więcej w 5 i 6 tygodniu Bootcampu) pojawiła się nowa metoda dostępna na obiektach typu String o nazwie repeat. Jej użycie wygląda następująco:

“hej ”.repeat(3) // zwraca “hej hej hej ”

Metoda ta jest dostępna we wszystkich nowoczesnych przeglądarkach internetowych, ale aby poćwiczyć rozszerzanie wbudowanych typów, utwórz jej **polyfill**. W kodzie sprawdź najpierw czy taka metoda już w przeglądarce została zaimplementowana, a jeśli nie, to dopisz własną funkcję, która będzie mogła być na dowolnym stringu wywołana w podany wyżej sposób.

**2. Wrapper Toggler dla elementów z drzewa DOM**

Do przygotowanego pod adresem http://pastebin.com/hUK5tnh3 kodu dodaj konstruktor

(klasę) o nazwie Toggler. Przy tworzeniu nowych jej instancji z użyciem słowa kluczowego

new (jak możesz zobaczyć w przygotowanym kodzie) przekazywać będziemy selektor. Za

jego pomocą należy znaleźć na stronie odpowiedni element (skorzystaj z metody

document.querySelector) i zapisać go w nowo stworzonym obiekcie. Następnie dodaj 3

metody. Pierwsza z nich o nazwie getElem powinna po prostu zwrócić znaleziony

wcześniej element. Metoda show i hide powinny kolejno **pokazywać** i **ukrywać** element.

Jeśli wszystko wykonasz poprawnie, kod który został już napisany powinien działać bez

żadnych modyfikacji. Zauważ, że do elementu o identyfikatorze **button** zostało przypisane

zdarzenie kliknięcia. Taki element musisz wstawić na stronę, podobnie jak i element,

którego selektor zostanie przekazany przy tworzeniu nowego obiektu klasy Toggler.

**3. Ajaxowy polyfill dla funkcji fetch**

Napisz **polyfill** dla funkcji fetch (nie będziemy się tutaj trzymać dokładnie tego, w jaki

sposób ona działa, stworzysz jedynie prostą jej wersję). Wykorzystaj obiekt

XMLHttpRequest w ten sposób, aby docelowo korzystanie z funkcji fetch wyglądało

następująco:

fetch("url", function(data) {

console.log("Sukces");

console.log(data);

}, function(err) {

console.log("Wystąpił błąd!");

console.log(err);

});

a zatem jako pierwszy argument przekazujemy adres **URL** (wyślij pod niego zapytanie

**GET**), jako drugi funkcję, którą należy wykonać **jeśli wszystko się powiedzie** (przekaż jej

pobrane dane), a jako trzeci funkcję, która wykona się **na wypadek błędu** (przekaż jej

obiekt z błędem lub komunikat tekstowy). W nowoczesnych przeglądarkach istnieje już

funkcja fetch, a zatem aby jej nie nadpisywać, możesz nadać jej inną nazwę, np. fecz.

Jako adres URL, z którego pobierane bedą dane, możesz wykorzystać http://

code.eduweb.pl/bootcamp/users/

**4. Funkcja getJSON**

Mając już rozwiązanie zadania trzeciego, przepisz swój kod tak, aby stworzyć nową funkcję

o nazwie getJSON. Idea działania i użycie będą bardzo podobne:

getJSON("url", function(data) {

console.log("Sukces");

console.log(data);

// typeof data powinno zwrócić “object”

}, function(err) {

console.log("Wystąpił błąd!");

console.log(err);

});

Różnica jest taka, że tym razem pod parametrem data przekazanym w funkcji callback,

powinien się znajdować javascriptowy obiekt, a nie string przysłany z serwera. Serwer,

który przygotowaliśmy pod tym adresem: http://code.eduweb.pl/bootcamp/users/ działa tak,

że po otrzymaniu standardowego zapytania **GET**, np. kiedy wpiszesz ten adres w

przeglądarce lub wyślesz zapytanie Ajaxem, zwróci dane zawarte w kodzie HTML. Jeśli

jednak przy wysyłaniu zapytania dodasz nagłówek Accept: application/json, to serwer

zwróci te same dane, ale w formacie **JSON** (bez kodu HTML). Z poziomu Twojego kodu

JavaScript, otrzymane Ajaxem dane to cały czas typ String, ale za pomocą metody

JSON.parse możesz ten string łatwo zamienić na javascriptowy obiekt. To jest właśnie

Twoje zadanie. Wysyłając żądanie do serwera, dodaj wspomniany wcześniej nagłówek za

pomocą metody setRequestHeader, a otrzymane dane sparsuj za pomocą JSON.parse i

dopiero wtedy przekaż jako parametr data do funkcji callback.

**5. Parser adresu URL**

Stwórz funkcję o nazwie getPage, która pozwoli odczytać numer strony z adresu URL.

Dane na temat adresu URL strony, na której wykonuje się Twój kod JavaScript, możesz

odczytać za pomocą obiektu window.location. Dopisz do adresu Twojej strony **?page=2** i wykorzystując odpowiednią właściwość obiektu location, a także odpowiednie wyrażenie

regularne, zwróć z funkcji 2 (typ Number) lub null, jeśli query string **?page=X** nie będzie

podany lub będzie niepoprawny, np. **?page=tekst**

Użycie funkcji powinno wyglądać następująco:

// dla adresu np. http://localhost/test/?page=2

getPage(); // zwraca 2

// dla adresu np. http://localhost/test/

getPage(); // zwraca null

// dla adresu np. http://localhost/test/?page=nieliczba

getPage(); // zwraca null