Node.js

Laboratorium 4

Dla przypomnienia konstrukcja Promise wygląda następująco:

```
const myPromise = new Promise((resolve, reject) => {
      if (...) {
            resolve('jakiś obiekt/wartość ...');
      } else {
            reject('błąd/przyczyna błędu itp...');
      }
});
```

Dodatkowe informacje o Promise:

- https://javascript.info/promise-basics
- https://medium.com/dailyjs/asynchronous-adventures-in-javascript-promises-1e0da27a3b4
- http://exploringjs.com/es6/ch_promises.html
- https://developers.google.com/web/fundamentals/primers/promises

1. Stwórzmy pierwszy nasz obiekt Promise, który zwróci nam w rezultacie `Hello world!`

```
const myPromise = ... <------ stworzenie obiektu Promise i logiki

myPromise.then(result => {
    console.log(result);
});
```

2. Stwórzmy Promise wykorzystujący funkcję `setTimeout` która wykona się po 5 sekundach i rozwiąże nasz Promise z naszym przywitaniem `Hello world!`
3. Stwórzmy aplikację która będzie odejmowała 2 liczby. Wykorzystując `Promise` dodajmy regułę jeżeli wynik odejmowania będzie niższy niż 0 ma zwrócić błąd z informacją(rejected), jeżeli rezultat będzie dodatni ma zwrócić wynik.

```
function sub(a, b) {
```

```
    return ... <------ stworzenie obiektu Promise i logiki
}
sub(1, 3).then(
    result => ...,
    error => ...
);
```

4. Wykorzystując wiedzę z poprzednich zajęć użyjmy zewnętrznej biblioteki `request` i pobierzmy użytkownika dane(https://jsonplaceholder.typicode.com/users/2). Przeróbmy tak wywołanie naszego zapytania aby wykorzystywało Promise zamiast Callback:

```
function getUser(id) {
    const url = ...
    return ... <---- implementacja Promise...
}
getUser(2)
    .then(user => ...)
    .catch(error => ...)
```

5. Zmodyfikujmy zadanie 4 tak aby pobrać kilku użytkowników w tej samej chwili wykorzystując `Promise.all()`. Wyświetlmy ich imiona w konsoli. (id użytkowników: 2,3,5,7,8,10). Poinformujmy iż nasz Promise został w pełni rozwiązany.

```
Promise.all([getUser(), ...])
    .then(...)
    .catch(...)
    .finally(...);
```

6. Dodajmy do naszej aplikacji z zadania 4 pobieranie pogody dla naszego użytkownika. Podobnie jak w poprzednim laboratorium. Pamiętajmy o odpowiednim owrapowaniu naszego zapytania do pogody. Analogicznie jak w zadaniu 4.

   Endpoint do pogody:
   https://api.openweathermap.org/data/2.5/weather?appid=0ed761300a2725ca778c07831ae64d6e&lat={LAT}&lon={LNG}

   Zarys wywołania aplikacji:

```
getUser(2)
      .then(getWeather)
      .then(...) <---- wypisz w konsoli temperature
      .catch(...)
      .finally(...);
```

7. Rozszerzając zadanie 6 z wykorzystaniem wiedzy z poprzednich zajęć, zapiszmy nasz cały obiekt pogody do pliku wykorzystując wbudowany moduł `fs` i funkcję `writeFile`. Oczywiście zadanie polega na odpowiednim dostosowaniu funkcji aby obsługiwała Promise.

```
function saveToFile(obj) {
      return ... <----- implementacja Promise
}
getUser(2)
      .then(getWeather)
      .then(...) <---- wypisz w konsoli temperature
      .then(weather => {
            return saveToFile(weather)
      })
      .catch(...)
      .finally(...);
```

8. Jak wiadomo świat Node.js jest bardzo rozbudowany i nie trzeba za każdym razem tworzyć od nowa konstrukcji asynchronicznych żądań do serwera. Są od tego biblioteki 😊
   Na stronie https://npmjs.org możemy znaleźć dużo różnych implementacji bibliotek które udostępniają już gotowe obiekty przystosowane do Promise, np.:
   - **axios** (https://www.npmjs.com/package/axios)
   - request-promise (https://www.npmjs.com/package/request-promise)

Ok, przejdźmy jednak do zadania ... Wykorzystując bibliotekę `**axios**` zamieńmy z zadania 7 `request` wraz z naszymi Promise na bibliotekę `axios`.

9. Stwórzmy aplikację która pobierze informacje o użytkowniku oraz jego pierwszym na liście albumie i przypisanych do niego zdjęciach.
   Z pobranego użytkownika wyświetlmy na ekranie nazwę użytkownika.

Z pobranego albumu wyświetlmy ilość wszystkich albumów oraz nazwę pierwszego z nich.
Z pobranych zdjęć wyświetlmy wszystkie tytuły.
Pamiętajmy również o tym aby obsłużyć błędy zapytania łapiąc występujący wyjątek (.catch())

endpoint do użytkownika: https://jsonplaceholder.typicode.com/users/2
endpoint do albumów: https://jsonplaceholder.typicode.com/albums?userId=2
endpoint do zdjęć: https://jsonplaceholder.typicode.com/photos?albumId=1

Ćwiczenie to ma za zadanie pokazać jak działają tzw. `chain Promise`(łańcuch 'obietnic').

10. Dodajmy do zadania 9 kolejny element łańcucha o zapis wczytanej listy zdjęć.

Wykorzystujemy tutaj moduł wbudowany `fs` i funkcje `writeFile`. Zamiast opakowywać samodzielnie funkcję w Promise wykorzystajmy wbudowane narzędzie z modułu `util` (więcej na: https://nodejs.org/dist/latest-v11.x/docs/api/util.html#util_util_promisify_original) np.:

```
const util = require('util');

const fs = require('fs');

const writeFilePromise = util.promisify(fs.writeFile);

...
```