Кафедра комп’ютерної та програмної інженерії

УДУ імені М.П. Драгоманова

Факультет математики, інформатики та фізики

Кафедра комп’ютерної та програмної інженерії

ЛАБОРАТОРНА РОБОТА № 7

ТЕМА:Робота з асинхронними функціями.

з курсу

«Програмування в інтернеті»

Студентка Козар Валерія

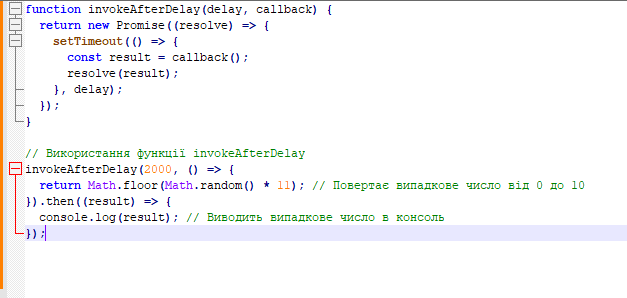
Група 31ІПЗ

Факультет МІФ

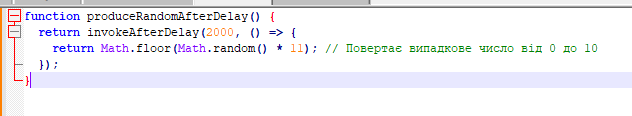
Викладач: Кархут В.Я.

Київ 2023

1. Напишіть функцію invokeAfterDelay, яка повертає проміс, який викликає задану функцію із заданою затримкою. Продемонструйте її роботу, повертаючи проміс, що містить випадкове число від 0 до 10. Отриманий результат виведіть в консолі.



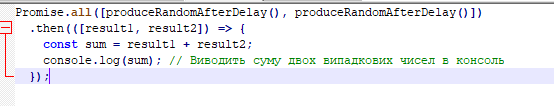
1. Створивши на базі попередньої функції функцію produceRandomAfterDelay. Викличте функцію produceRandomAfterDelay двічі і надрукуйте суму, після того як будуть отримані обидва результати.



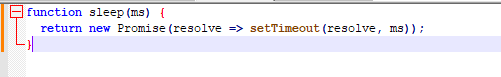
Ця функція повертає проміс, який викликає функцію invokeAfterDelay зі значенням 2000 для затримки та функцією, яка повертає випадкове число від 0 до 10 за допомогою Math.random().

Тепер, для того щоб викликати цю функцію двічі та отримати суму двох випадкових чисел, ми можемо використати метод Promise.all, який дозволяє одночасно виконувати кілька промісів та чекати на їхні результати.

Отже, код буде виглядати наступним чином:

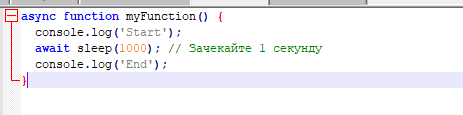


1. Напишіть функцію sleep, яка повертає проміс, який можна викликати так: await sleep(1000)



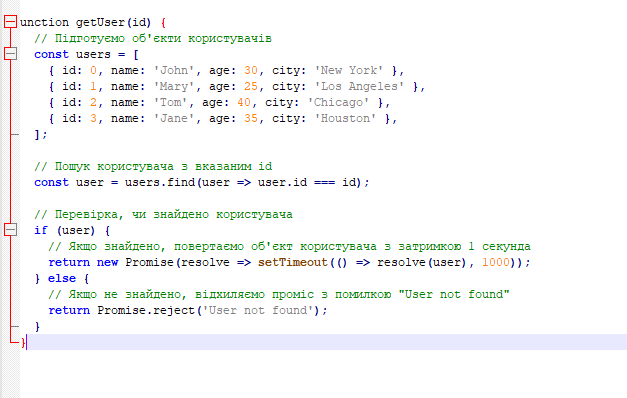
Ця функція створює новий проміс та використовує setTimeout для очікування заданої затримки (в мілісекундах). Після закінчення затримки проміс виконується та повертається.

Тепер, щоб використати цю функцію, просто викличіть її з необхідною затримкою за допомогою await:

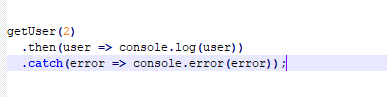


1. Напишіть функцію getUser яка приймає id та повертає проміс який виконується через 1 секунду з обєктом користувача з полями імя, вік, місто, id. Підготуйте 4 обєкти користувача з id від 0 до 3 які повертатимуться функцією відповідно до id. Якщо незнайомий id отриманий – проміс має бути відхилений з помилкою ‘User not found’.

Ось приклад реалізації функції getUser, яка повертає проміс з об'єктом користувача за його id. Якщо id не відповідає жодному користувачу, проміс буде відхилений з помилкою "User not found".



Тепер, щоб використати цю функцію, просто викличіть її з id користувача, якого потрібно отримати:



У цьому прикладі ми викликаємо функцію getUser з id 2. Якщо користувач з таким id знайдений, його об'єкт буде виведений в консолі через 1 секунду. Якщо користувач з таким id не знайдений, виведеться помилка "User not found".

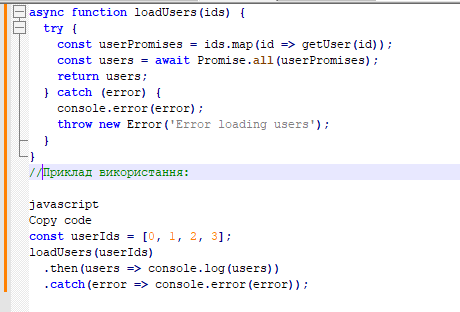
1. Напишіть функцію loadUsers яка приймає масив ідентифікаторів та повертає масив обєктів користувача використовуючи попередню функцію. Обробіть ситуацію коли один з промісів був відхилений.

Для розв'язання цієї задачі ми можемо скористатись методом Promise.all(), який дозволяє чекати на виконання всіх переданих промісів. Якщо хоча б один з переданих промісів буде відхилений, Promise.all() поверне проміс, який буде відхилений з помилкою, повернутою відповідним промісом.

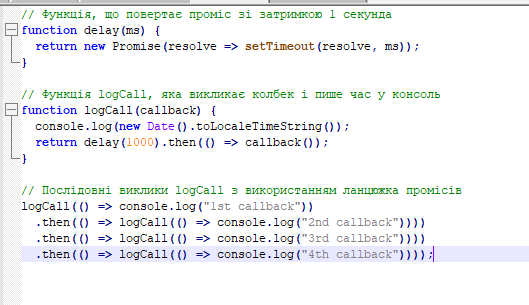
Отже, можна створити функцію loadUsers, яка приймає масив ідентифікаторів ids та викликає функцію getUser для кожного ідентифікатора. Після цього, ми можемо використати Promise.all() для очікування виконання всіх промісів та повернення масиву об'єктів користувачів.

Якщо хоча б один з промісів буде відхилений, ми можемо обробити помилку за допомогою блоку try/catch та повернути відхилення.

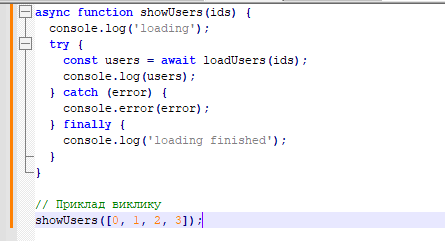
Ось код функції loadUsers:



1. Напишіть функцію logCall яка приймає функцію коллбек – викликає її через одну секунду та пише в консоль поточний час. Зробіть щоб дана функція повертала проміс. Зробіть 4 послідовних виклики даної функції використовуючи ланцюжок промісів.



1. Напишіть функцію яка showUsers яка симулює завантаження користувачів використовуючи loadUsers. Перед викликом loadUsers дана функція має вивести в консоль ‘loading’ при при успішному чи неуспішному завершенні виведе ‘loading finished’. Використайте синтаксис async/await при виконанні даного завдання.



У цьому прикладі ми спочатку виводимо повідомлення "loading", щоб показати, що процес завантаження користувачів розпочався. Далі ми викликаємо функцію loadUsers і очікуємо повернення масиву користувачів за допомогою ключового слова await. Якщо завантаження користувачів відбулося успішно, ми виводимо результат в консоль. Якщо сталася помилка, ми виводимо повідомлення про помилку. В будь-якому випадку, незалежно від того, чи відбулося завантаження користувачів успішно чи ні, ми виводимо повідомлення "loading finished", щоб показати, що процес завантаження користувачів завершився.