Q: Що таке Промісифікація?

a) Процес створення нового Promise.

b) Перетворення функції, що приймає колбеки, на функцію, що повертає проміс.

c) Використання async/await для обробки асинхронних операцій.

d) Обробка декількох промісів одночасно.

A: 2

Q: Яка головна проблема callback-орієнтованих функцій з точки зору залежності від зовнішнього коду?

a) Вони не можуть виконувати асинхронні операції.

b) Функція знає занадто багато про код, що використовуватиме результат її роботи, очікуючи конкретні колбеки.

c) Вони завжди призводять до "callback hell".

d) Вони не можуть повертати значення.

A: 2

Q: Що функція повертає у зовнішній код для обробки результату асинхронної операції?

a) Значення успішного виконання.

b) Об'єкт помилки.

c) Promise.

d) Колбек-функцію.

A: 3

Q: Яка відмінність між колбеками та промісами з точки зору їхнього типу?

a) Колбеки — об'єкти, проміси — функції.

b) Колбеки — рядки, проміси — числа.

c) Колбеки — функції, проміси — об'єкти.

d) Колбеки — змінні, проміси — константи.

A: 3

Q: Як колбеки передаються у функцію, що виконує асинхронну операцію?

a) Як значення, що повертається.

b) Як аргументи функції.

c) Через глобальну змінну.

d) Як властивості об'єкта.

A: 2

Q: Де створюється проміс при промісифікації функції за допомогою new Promise()?

a) Зовні функції, перед її викликом.

b) Всередині функції, що виконує асинхронну операцію.

c) В глобальній області видимості.

d) В окремому файлі.

A: 2

Q: Що використовується у зовнішньому коді для обробки успішного виконання промісу, повернутого промісифікованою функцією?

a) Метод .catch().

b) Метод .then().

c) Конструкція try...catch.

d) Оператор await.

A: 2

Q: Що використовується у зовнішньому коді для обробки помилки (відхилення) промісу, повернутого промісифікованою функцією?

a) Метод .then().

b) Метод .finally().

c) Метод .catch().

d) Оператор async.

A: 3

Q: Які статичні методи класу Promise використовуються для створення промісів, що миттєво успішно виконуються або відхиляються?

a) Promise.all() та Promise.race()

b) Promise.resolve() та Promise.reject()

c) Promise.allSettled() та Promise.any()

d) Promise.create() та Promise.finish()

A: 2

Q: Для чого можуть використовуватися Promise.resolve() та Promise.reject() при промісифікації синхронних функцій?

a) Щоб додати затримку у виконання функції.

b) Щоб створити ланцюжок промісів, коли вже є початкове значення і не потрібне очікування асинхронної операції.

c) Для виконання HTTP-запитів.

d) Для обробки декількох асинхронних подій одночасно.

A: 2

Q: Який метод використовується для створення промісу з різною затримкою виконання та можливістю успішного виконання або відхилення?

a) Promise.all()

b) Promise.race()

c) new Promise() у поєднанні з setTimeout()

d) Promise.resolve()

A: 3

Q: Який статичний метод класу Promise використовується, коли потрібно дочекатися виконання всіх промісів у масиві, і повернути результат у вигляді масиву значень, якщо всі успішні, або одразу відхилитись, якщо хоча б один відхилено?

a) Promise.race()

b) Promise.allSettled()

c) Promise.any()

d) Promise.all()

A: 4

Q: Що повертає Promise.all() у разі успішного виконання всіх промісів у масиві?

a) Значення першого промісу, що завершився.

b) Масив об'єктів зі статусами та значеннями кожного промісу.

c) Масив значень успішно виконаних промісів.

d) undefined.

A: 3

Q: Що відбувається, якщо хоча б один проміс у масиві, переданому в Promise.all(), відхиляється?

a) Promise.all() чекає завершення всіх інших, а потім відхиляється.

b) Promise.all() одразу відхиляється зі значенням помилки першого відхиленого промісу.

c) Promise.all() ігнорує відхилений проміс і повертає результати успішних.

d) Promise.all() повертає масив з результатами та помилками.

A: 2

Q: Який статичний метод класу Promise використовується, коли потрібно дочекатися завершення всіх промісів у масиві, незалежно від того, успішно вони виконані чи відхилені, і повернути масив об'єктів з описом результату кожного промісу?

a) Promise.all()

b) Promise.race()

c) Promise.allSettled()

d) Promise.any()

A: 3

Q: Що повертає Promise.allSettled()?

a) Масив значень успішно виконаних промісів.

b) Значення першого промісу, що завершився.

c) Масив об'єктів, кожен з яких описує результат (статус та значення/причину) відповідного промісу.

d) Promise, який може бути відхилений, якщо хоча б один вхідний проміс відхилено.

A: 3

Q: Який статичний метод класу Promise повертає результат або помилку першого промісу в масиві, який завершився (успішно чи з помилкою)?

a) Promise.all()

b) Promise.race()

c) Promise.allSettled()

d) Promise.any()

A: 2

Q: Якщо перший проміс у масиві, переданому в Promise.race(), відхиляється з помилкою, що поверне Promise.race()?

a) Результат першого успішно виконаного промісу.

b) Помилку першого відхиленого промісу.

c) Масив результатів усіх промісів.

d) undefined.

A: 2

Q: Що відбувається з іншими промісами в масиві після того, як один з них завершився (успішно чи з помилкою) при використанні Promise.race()?

a) Вони продовжують виконуватися до завершення.

b) Вони ігноруються і відкидаються.

c) Вони також повертаються у результаті Promise.race().

d) Вони стають частиною ланцюжка .then().

A: 2

Q: Якщо функція fetchData використовує колбек з двома аргументами (error, result) для повідомлення про результат, як слід змінити цю функцію при промісифікації, щоб вона повертала проміс?

a) Функція повинна повертати null або undefined.

b) Замість виклику колбека, вона повинна викликати resolve(result) у разі успіху або reject(error) у разі помилки всередині конструктора new Promise().

c) Функція повинна використовувати async/await.

d) Функція повинна повертати об'єкт з властивостями error та result.

A: 2