Q: Який синтаксис називається зручним способом написання асинхронного коду, що виглядає схожим на синхронний?

a) Promise.then().catch()

b) Callbacks

c) async/await

d) setTimeout

A: 3

Q: У якому стандарті ECMAScript був доданий синтаксис async/await?

a) ES6 (ECMAScript 2015)

b) ES7 (ECMAScript 2016)

c) ES8 (ECMAScript 2017)

d) ES9 (ECMAScript 2018)

A: 3

Q: Яке ключове слово необхідно додати перед оголошенням функції, щоб зробити її асинхронною?

a) await

b) function

c) async

d) return

A: 3

Q: Що завжди повертає асинхронна функція?

a) Значення, вказане у return.

b) undefined, якщо немає return.

c) Promise.

d) Об'єкт.

A: 3

Q: Як обробляється значення, що повертається з асинхронної функції за допомогою ключового слова return, якщо це не Promise?

a) Воно ігнорується.

b) Воно обгортається в Promise.resolve().

c) Воно спричиняє помилку.

d) Воно повертається як є.

A: 2

Q: Що повертається з асинхронної функції, якщо в ній не вказано значення для повернення за допомогою return?

a) null

b) Promise зі значенням undefined.

c) Помилка.

d) Пустий об'єкт.

A: 2

Q: Для чого використовується оператор await всередині асинхронної функції?

a) Для оголошення змінної.

b) Для створення нового Promise.

c) Для очікування завершення Promise перед продовженням виконання коду.

d) Для виклику іншої функції.

A: 3

Q: Де можна використовувати оператор await?

a) Будь-де в коді JavaScript.

b) Тільки всередині звичайних синхронних функцій.

c) Тільки всередині асинхронних функцій (оголошених з async).

d) Тільки в глобальній області видимості.

A: 3

Q: Що відбувається з виконанням асинхронної функції, коли інтерпретатор зустрічає await перед Promise, який ще не завершився?

a) Функція завершує виконання з помилкою.

b) Виконання функції призупиняється до завершення Promise.

c) Код після await виконується негайно.

d) Promise ігнорується, і функція продовжує роботу.

A: 2

Q: Яке значення повертає await, якщо Promise, на який він чекає, виконано успішно (fulfilled)?

a) undefined

b) Значення, з яким виконано Promise.

c) Promise об'єкт.

d) null

A: 2

Q: Що відбувається, якщо Promise, на який чекає await, відхилено (rejected)?

a) await повертає значення помилки.

b) Виконання функції продовжується, і помилка ігнорується.

c) Виникає помилка, яку потрібно обробляти (наприклад, за допомогою try/catch).

d) Promise перетворюється на успішний зі значенням undefined.

A: 3

Q: Яка основна перевага використання async/await при роботі з HTTP-запитами порівняно зі стандартним підходом на промісах (.then())?

a) Зменшення часу виконання запиту.

b) Збільшення кількості одночасних запитів.

c) Код виглядає послідовним, наче виконується синхронно, що покращує читабельність.

d) Автоматична обробка всіх можливих помилок.

A: 3

Q: Яка конструкція використовується для обробки помилок, що можуть виникати під час виконання коду (зокрема, з async/await та HTTP-запитами)?

a) if...else

b) switch...case

c) try...catch

d) while цикл

A: 3

Q: Якщо асинхронна функція з await повертає результат, як можна обробити помилку, що виникла всередині цієї функції, за межами її тіла?

a) Використовуючи try...catch всередині функції.

b) Використовуючи .catch() при виклику асинхронної функції.

c) Використовуючи finally() при виклику асинхронної функції.

d) Помилка не може бути оброблена за межами функції.

A: 2

Q: Чому при роботі з великими колекціями даних на сервері (тисячі записів) не варто завантажувати всю колекцію за один запит?

a) Це збільшує безпеку даних.

b) Це зменшує навантаження на мережу.

c) Це спричиняє велике навантаження на браузер, тривалий час завантаження та навантаження на сервер.

d) Це ускладнює сортування даних.

A: 3

Q: Який підхід використовується для оптимізації роботи з великими колекціями даних, коли сервер повертає лише частину записів за запитом?

a) Кешування даних на сервері.

b) Пагінація.

c) Використання стиснення даних.

d) Потокова передача даних.

A: 2

Q: Як клієнт зазвичай вказує серверу, яку частину (групу або "сторінку") даних він хоче отримати при пагінації?

a) Через заголовки HTTP-запиту.

b) Через тіло HTTP-запиту (POST).

c) Через спеціальні параметри в URL запиту (наприклад, \_page).

d) Через WebSocket з'єднання.

A: 3

Q: Які типові назви параметрів використовуються в URL запиту для визначення кількості елементів у відповіді при пагінації?

a) id, name

b) per\_page, limit, \_limit, count

c) sort, order

d) start, end

A: 2

Q: Як можна визначити на фронтенді, коли досягнуто кінця колекції при використанні пагінації, якщо сервер не повертає загальну кількість сторінок?

a) Порівнюючи номер поточної сторінки із загальною кількістю елементів.

b) Перевіряючи, чи повернув сервер порожній масив або менше елементів, ніж очікувалося за лімітом, після останнього запиту.

c) Використовуючи фіксоване значення для загальної кількості сторінок.

d) Надсилаючи запит на отримання всіх елементів одразу.

A: 2

Q: Як розраховується загальна кількість сторінок (totalPages) на фронтенді, якщо бекенд надає лише загальну кількість елементів (totalItems) та відома кількість елементів на сторінці (limit)?

a) totalPages = totalItems - limit

b) totalPages = totalItems / limit (з округленням вгору за необхідності)

c) totalPages = totalItems + limit

d) totalPages = totalItems \* limit

A: 2