Q: Як виконується синхронний код у більшості мов програмування та в JavaScript, якщо не використовується асинхронність?

a) Декілька інструкцій одночасно.

b) Кожна інструкція чекає завершення попередньої, послідовно.

c) У випадковому порядку.

d) Залежно від навантаження на процесор.

A: 2

Q: Чому в однопотоковому JavaScript важливо використовувати асинхронність при роботі з такими операціями, як мережеві запити чи таймери?

a) Для прискорення виконання синхронного коду.

b) Щоб зробити JavaScript багатопотоковим.

c) Щоб уникнути блокування виконання основного коду та "зависання" інтерфейсу.

d) Асинхронність не має значення в однопотокових мовах.

A: 3

Q: Що відбувається в асинхронній моделі виконання коду, коли запускається "повільна" операція, наприклад, мережевий запит?

a) Програма зупиняється і чекає завершення цієї операції.

b) Програма продовжує виконувати інший код, не чекаючи завершення "повільної" операції.

c) Виникає помилка.

d) Операція скасовується автоматично.

A: 2

Q: Яка основна відмінність між асинхронністю в JavaScript та багатопотоковістю (паралелізмом)?

a) Асинхронність використовує декілька потоків виконання, багатопотоковість – один.

b) Асинхронність досягається через відкладені виклики функцій в одному потоці, багатопотоковість передбачає паралельне виконання в декількох потоках.

c) Асинхронність блокує основний потік, багатопотоковість – ні.

d) Це синоніми, що означають одне й те саме.

A: 2

Q: Який механізм використовується в JavaScript для досягнення асинхронності в однопотоковому середовищі?

a) Паралельне виконання коду на кількох ядрах процесора.

b) Розділення програми на окремі потоки.

c) Відкладені виклики функцій, де ініціалізація операції та обробка результату є різними діями.

d) Блокування виконання до завершення тривалих операцій.

A: 3

Q: Яка функція дозволяє запланувати одноразовий виклик іншої функції через певний проміжок часу?

a) setInterval()

b) clearInterval()

c) setTimeout()

d) requestAnimationFrame()

A: 3

Q: В яких одиницях вимірювання вказується затримка (delay) для функцій setTimeout() та setInterval()?

a) Секунди.

b) Мілісекунди.

c) Хвилини.

d) Наносекунди.

A: 2

Q: Що повертає виклик функції setTimeout()?

a) Promise.

b) Значення, яке повертає callback-функція.

c) Числовий ідентифікатор таймера.

d) Об'єкт помилки.

A: 3

Q: Яка функція використовується для скасування відкладеного виклику функції, запланованого за допомогою setTimeout()?

a) resetTimeout()

b) stopTimeout()

c) clearTimeout()

d) removeTimeout()

A: 3

Q: Який аргумент необхідно передати функції clearTimeout() для скасування таймера?

a) Назву callback-функції.

b) Час затримки таймера.

c) Числовий ідентифікатор таймера, повернутий setTimeout().

d) Рядок з назвою таймера.

A: 3

Q: Яка функція використовується для регулярного повторення виконання іншої функції через певний інтервал часу?

a) setTimeout()

b) clearInterval()

c) setInterval()

d) requestIdleCallback()

A: 3

Q: Що повертає виклик функції setInterval()?

a) Результат виконання callback-функції.

b) Числовий ідентифікатор інтервалу.

c) Promise.

d) undefined.

A: 2

Q: Яка функція використовується для скасування регулярного виконання функції, запланованого за допомогою setInterval()?

a) resetInterval()

b) stopInterval()

c) clearInteger()

d) clearInterval()

A: 4

Q: Який аргумент необхідно передати функції clearInterval() для скасування інтервалу?

a) Назву callback-функції.

b) Час інтервалу.

c) Числовий ідентифікатор інтервалу, повернутий setInterval().

d) Рядок з назвою інтервалу.

A: 3

Q: Де реалізовані браузерні таймери (setTimeout, setInterval) в JavaScript?

a) Вони вбудовані безпосередньо в ядро мови JavaScript.

b) Вони реалізовані в середовищі браузера і доступні на глобальному об'єкті window.

c) Вони є частиною специфікації ECMAScript.

d) Вони реалізовані на стороні сервера.

A: 2

Q: Яка приблизна мінімальна затримка існує для браузерних таймерів за стандартом?

a) 0 мілісекунд.

b) 4 мілісекунди.

c) 10 мілісекунд.

d) 100 мілісекунд.

A: 2

Q: Чому браузерні таймери можуть спрацьовувати рідше, ніж зазначено в параметрі delay?

a) Через помилки в коді таймера.

b) Через занадто мале навантаження на процесор.

c) Через занадто велике завантаження процесора або неактивність вкладки браузера.

d) Таймери завжди спрацьовують точно у вказаний час.

A: 3

Q: Що відбувається з асинхронною callback-функцією, переданою в setTimeout(), після того, як ця функція setTimeout() відпрацювала синхронно?

a) Callback-функція виконується негайно.

b) Callback-функція ставиться в асинхронну чергу на виконання через вказаний проміжок часу.

c) Callback-функція ігнорується.

d) Callback-функція виконується тільки після перезавантаження сторінки.

A: 2

Q: Як виглядатиме виведення в консоль для наступного коду? console.log("First"); setTimeout(() => { console.log("Second"); }, 1000); console.log("Third");

a) First, Second, Third

b) Second, First, Third

c) First, Third, Second

d) Third, First, Second

A: 3

Q: Яка з перерахованих операцій є прикладом асинхронної операції, згаданої в матеріалі?

a) Додавання двох чисел.

b) Читання змінної.

c) Завантаження даних з сервера.

d) Виконання циклу for.

A: 3