1. Lập trình bất đồng bộ là gì?

Lập trình bất đồng bộ (Asynchronous Programming) là một mô hình lập trình cho phép chương trình khởi tạo và xử lý các tác vụ mà không cần chờ đợi tác vụ trước đó hoàn thành; chương trình có thể tiếp tục thực hiện các công việc khác trong khi chờ đợi kết quả từ các tác vụ bất đồng bộ.

Thay vì thực thi từng dòng lệnh một cách tuần tự từ trên xuống dưới, lập trình bất đồng bộ cho phép một tác vụ được "gửi đi" và chương trình có thể tiếp tục với các tác vụ khác.

2. Tại sao cần lập trình bất đồng bộ?

Nhu cầu về lập trình bất đồng bộ xuất phát từ những hạn chế cố hữu của mô hình lập trình đồng bộ truyền thống, đặc biệt trong bối cảnh các ứng dụng hiện đại:

- Trải nghiệm người dùng kém
- Hiệu suất và Khả năng mở rộng hạn chế
- Tận dụng tài nguyên hiệu quả
- Xử lý tác vụ không chặn

3. Tại sao cần lập trình bất đồng bộ?

Các đặc điểm nổi bật của nó:

- Không chặn (Non-blocking)
- Phi tuần tự (Non-sequential Execution)
- Sử dụng cơ chế chờ (Waiting Mechanisms)
- Tăng cường khả năng phản hồi (Responsiveness)
- Khó Debug hơn

4. Lập trình bất đồng bộ hoạt động như thế nào?

Trong một số môi trường như JavaScript, lập trình bất đồng bộ dựa vào cơ chế vòng lặp sự kiện (event loop), giúp theo dõi và xử lý các callback hoặc Promise khi các tác vụ bất đồng bộ hoàn thành. Promise là một đối tượng đại diện cho kết quả của một tác vụ bất đồng bộ, còn vòng lặp sự kiện chịu trách nhiệm phân phối các callback tương ứng.

Là một phần của vòng lặp sự kiện, bạn có thể tạo một callback, cho phép vòng lặp sự kiện cung cấp thông tin từ chương trình cho một phần mã khác, thường là chức năng chính của chương trình.

Trong thời gian đó, ứng dụng có thể thực hiện các tác vụ khác trong khi bạn đang chờ đợi kết quả từ chương trình. Điều này cho phép các tác vụ tốn kém về mặt tài nguyên có thể chạy mà không khiến người dùng phải chờ đợi đến khi hoàn thành.

Await là một từ khóa trong nhiều ngôn ngữ lập trình hiện đại, cho phép tạm dừng thực thi bên trong một hàm bất đồng bộ (async function) cho đến khi tác vụ bất đồng bộ hoàn thành.

5. Ưu, nhược điểm của lập trình bất đồng bộ

Ưu điểm

- Cải thiện hiệu suất và thông lượng
- Nâng cao trải nghiệm người dùng
- Tận dụng tài nguyên hệ thống tốt hơn
- Khả năng mở rộng cao

Nhược điểm

- Độ phức tạp của mã tăng lên
- Khó Debugging
- Quản lý trạng thái và dữ liệu
- Overhead nhất định

6. Các kỹ thuật thường dùng trong lập trình bất đồng bộ

- Callback
- Promise
- Async/Await

7. So sánh lập trình đồng bộ (Synchronous) và bất đồng bộ

Tiêu chí	Synchronous (Đồng bộ)	Asynchronous (Bất đồng bộ)
Luồng thực thi	Tuần tự, theo thứ tự từng dòng lệnh một.	Không tuần tự, các tác vụ có thể chạy song song hoặc chồng lấn.
Hành vi chặn	Bị chặn (blocking). Luồng chính phải chờ tác vụ hoàn thành.	Không chặn (non-blocking). Luồng chính tiếp tục thực hiện công việc khác.
Hiệu suất	Thường kém hơn cho các tác vụ I/O-bound.	Cao hơn cho các tác vụ I/O-bound, tận dụng tài nguyên tốt.
Trải nghiệm người dùng	Giao diện có thể bị treo, không phản hồi.	Giao diện luôn mượt mà, phản hồi tốt.

Độ phức tạp mã	Đơn giản, dễ đọc, dễ debug.	Phức tạp hơn, cần quản lý callback, Promise, async/await.
Khả năng mở rộng	Hạn chế, cần nhiều tài nguyên hơn để xử lý song song.	Cao, có thể xử lý nhiều yêu cầu với ít tài nguyên hơn.
Xử lý lỗi	Đơn giản với try-catch truyền thống.	Phức tạp hơn, cần xử lý lỗi trong các chuỗi bất đồng bộ.
Các ví dụ điển hình	Tính toán cục bộ CPU-bound.	Gọi API, truy vấn DB, đọc/ghi file, xử lý sự kiện UI.