Node.js의 비동기 프로그래밍 방식

Node.js는 자바스크립트 실행 환경으로, 이벤트 주도적인 비동기 프로그래밍을 지원합니다. 이를 통해 CPU 바운드 작업을 처리하는 데 효율적입니다. 이번 프레젠테이션에서는 Node.js의 대표적인 비동기 프로그래밍 기법인 콜백, 프로미스, async/await을 자세히 살펴보겠습니다.

작성자: 현욱

콜백 함수의 작동 원리

1

비동기 작업 요청

노드 환경에서 비동기 작업을 수행할 때, 콜백 함수를 인자로 전달합니다.

2

콜백 함수 대기

비동기 작업이 완료되면 콜백 함수가 호출되어 결과를 처리합니다.

3

동기화 문제 발생

콜백 지옥으로 인해 코드 가독성이 저하되고 오류 처리가 복잡해질 수 있습니다.

콜백 함수의 장단점

장점

- 비동기 작업을 쉽게 구현할 수 있음
- 오래된 API와의 호환성이 우수함

단점

- 콜백 지옥으로 인한 가독성 저하
- 에러 처리가 복잡함
- 비동기 흐름 제어가 어려움

프로미스의 작동 원리

1

프로미스 객체 생성

프로미스 객체를 생성하고 비동기 작업을 수행합니다.

2

상태 변화 감지

작업이 완료되면 프로미스 객체의 상태가 변화합니다.

3

후속 처리

then(), catch(), finally() 메서드로 결과 처리 및 에러 핸들링을 합니다.

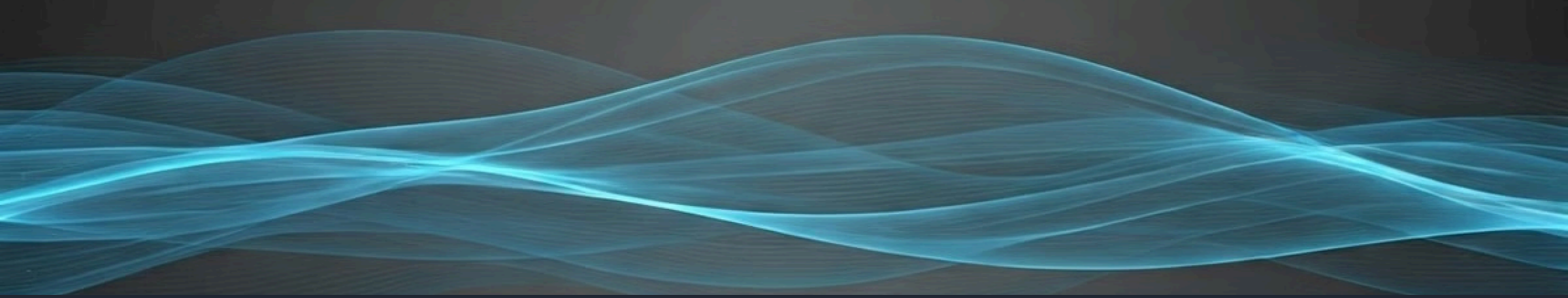
프로미스의 장단점

장점

- 가독성 향상
- 에러 처리가 용이
- 비동기 흐름 제어가 쉬움

단점

- 상속 및 에러 전파 문제 발생
- 오래된 API와의 호환성이 낮음



async/await의 작동 원리

async 함수 선언

async 키워드로 비동기 함수를 정의합니다.

1

2

await 사용

프로미스 객체를 반환하는 비동기 작업 앞에 await을 사용합니다.

3

동기 코드 실행

await을 통해 프로미스가 처리될 때까지 기다리므로 동기 코드와 유사하게 실행됩니다.

async/await의 장단점

장점

- 동기 코드와 유사한 가독성
- 에러 처리가 간단해짐
- 비동기 흐름 제어가 용이

단점

- ES2017(ES8) 이상 지원 필요
- 프로미스 사용이 필수

비동기 프로그래밍 방식 비교

1 콜백

가장 기본적인 비동기 패턴으로, 오래된 API와의 호환성이 좋지만 콜백 지옥 문제가 있습니다.

2 프로미스

콜백 지옥을 해결하고 에러 처리 및 흐름 제어를 개선했지만, 상속 및 호환성 이슈가 있습니다.

3 async/await

프로미스를 기반으로 하여 동기 코드와 유사한 가독성을 제공하며, 에러 처리와 흐름 제어가 편리합니다.