자바스크립트의 비동기 처리 방식에는 어떤 것들이 있고, 각각의 특징은 무엇인가요?

콜백 함수 (Callback Functions)

콜백 함수는 함수의 인자로 전달되어, 특정 조건이나 이벤트가 발생했을 때 호출되는 함수.

콜백 함수를 이용하여 비동기 작업을 처리할 수 있다.

그러나 콜백 함수를 여러 번 중첩하여 사용하면 코드의 가독성이 떨어지고 유지보수가 어려워지는 '콜백 지옥'이라는 문제가 발생할 수 있다.

```
function fetchData(callback) { setTimeout(() => { const data = 'Hello, World!'; callback(null, data); }, 1000); } fetchData((error, data) => { if (error) { console.error('Error:', error); } else { console.log('Data:', data); } });
```

프로미스 (Promises)

프로미스는 비동기 작업의 최종 결과를 나타내는 객체.

콜백 함수보다 가독성이 높고, 프로미스 체인을 이용하여 여러 비동기 작업을 순차적으로 처리할 수 있다.

프로미스는 대기(pending), 이행(fulfilled), 거부(rejected)의 세 가지 상태를 가진다.

```
function fetchData() { return new Promise((resolve, reject) => { setTimeout(() => { const data = 'Hello, World!'; if (data) { resolve(data); } else { reject(new Error('Failed to fetch data')); } }, 1000); }); } fetchData() .then((data) => { console.log('Data:', data); }) .catch((error) => { console.error('Error:', error); });
```

async / await

async / await는 프로미스를 기반으로 한 비동기 처리 문법으로, ES8(ES2017)에 도입되었다.

async 함수는 항상 프로미스를 반환하며, await 키워드를 사용하여 프로미스의 결과를 기다릴 수 있다.

async / await를 사용하면 비동기 작업을 마치 동기 작업처럼 작성할 수 있어 코드의 가독성이 높아진다.

```
async function fetchData() { return new Promise((resolve, reject) => {
setTimeout(() => { const data = 'Hello, World!'; if (data) {
resolve(data); } else { reject(new Error('Failed to fetch data')); } },
1000); }); } (async () => { try { const data = await fetchData();
console.log('Data:', data); } catch (error) { console.error('Error:',
error); } })();
```

요약

콜백 함수, 프로미스, async/await는 자바스크립트에서 비동기 처리를 위해 사용되는 주요 방식. 각 방식의 사용법과 장단점을 이해하면 비동기 처리 작업을 더 효과적으로 수행할 수 있다.

콜백 함수를 사용할 때 발생하는 콜백 지옥 문제는 프로미스를 사용함으로써 해결할 수 있다. 프로미스는 비동기 작업을 체인 형태로 연결할 수 있어 코드의 가독성이 향상되며, 에러 처리도 명확하게 할 수 있다. 그러나 여전히 프로미스를 사용하면 코드가 다소 복잡해질 수 있다.

이에 비해 async/await는 코드를 마치 동기 작업처럼 작성할 수 있어 가독성이 높고, 비동기 처리 흐름을 쉽게 이해할 수 있다. async/await를 사용하면 프로미스의 then과 catch 메서드를 사용하지 않고도 비동기 작업의 결과를 기다리거나 에러 처리를 할 수 있다.

이와 같이, 콜백 함수, 프로미스, async/await는 각각 다른 문제를 해결하며 발전해 왔다. 프로젝트의 요구 사항에 따라 적절한 비동기 처리 방식을 선택하여 사용하면 된다. 그러나, 현대 자바스크립트 개발에서는 프로미스와 async/await가 주로 사용되는 추세. 이는 코드의 가독성과 유지보수성이 높아져 비동기 처리 작업을 더욱 효율적으로 수행할 수 있기 때문이다.