Item 25

비동기 코드에는 콜백 대신 async 함수 사용하기

ryan-dia (강철원)

결론

콜백보다는 프로미스를 사용하는 게 코드 작성과 타입 추론 면에서 유리하다.

가능하다면 프로미스를 생성하기보다는 async와 await를 사용하는 것이 좋다.

ES2015 전

```
function fetchURL(url: string, cb: (response: string) => void) {
  cb(url);
}
const url1 = '1';
const url2 = '2';
const url3 = '3';

fetchURL(url1, function(response1) {
  fetchURL(url2, function(response2) {
    fetchURL(url3, function(response3) {
      // ...
      console.log(1);
    });
    console.log(2);
  });
  console.log(3);
});
console.log(4);
```

ES2015 프로미스 도입

```
const page1Promise = fetch(url1);
page1Promise.then(response1 => {
  return fetch(url2);
}).then(response2 => {
  return fetch(url3);
}).then(response3 => {
  // ...
}).catch(error => {
  // ...
});
```

ES2017

```
async function fetchPages() {
const response1 = await fetch(url1);
const response2 = await fetch(url2);
const response3 = await fetch(url3);
// ...
}
```

```
async function fetchPages() {
  try {
    const response1 = await fetch(url1);
    const response2 = await fetch(url2);
    const response3 = await fetch(url3);
    // ...
  } catch (e) {
    // ...
  }
}
```

콜백보다 프로미스나 async/await를 사용해야 하는 이유

- 콜백보다는 프로미스가 코드를 작성하기 쉽다.
- 콜백보다는 프로미스가 타입을 추론하기 쉽다.
- 콜백보다는 프로미스가 가독성이 좋다.

📌 프로미스보다 async/await를 사용하면 좋은점

1. 가독성과 유지보수성: 'async/await' 구문은 비동기 코드를 동기적인 스타일로 작성할 수 있게 해준다.

2. 오류처리 : 'async/await'는 'try/catch' 블록을 사용하여 동기적인 방식으로 오류를 처리할 수 있다. 프로미스 체인에서 오류 처리를 하려면 '.catch()' 메서드를 사용해야 했는데, 'async/await'를 사용하면 예외 처리를 일반적인 방식으로 할 수 있다.

3. 디버깅 :'async/await' 는 디버깅을 용이하게 만든다. 프로미스 체인을 디버깅하면 중간 단계에서 어떤 값이 반환되는지 추적하기 어렵다. 그러나 'async/await'를 사용하면 단계별로 코드를 실행하고 각 단계의 결과를 쉽게 확인할 수 있다.

4. 제어 흐름 : 'async/await'는 프로미스 체인보다 제어 흐름을 더 잘 다룰 수 있다. 프로미스 체인에서는 'then()'메서드를 연속적으로 사용해야 하므로 코드의 로직을 이해하기가 어려울 수 있다. 'async/await'는 코드의 제어 흐름을 명확하게 표현하고, 비동기 작업 간의 의존성을 처리하기 쉽게 해준다.