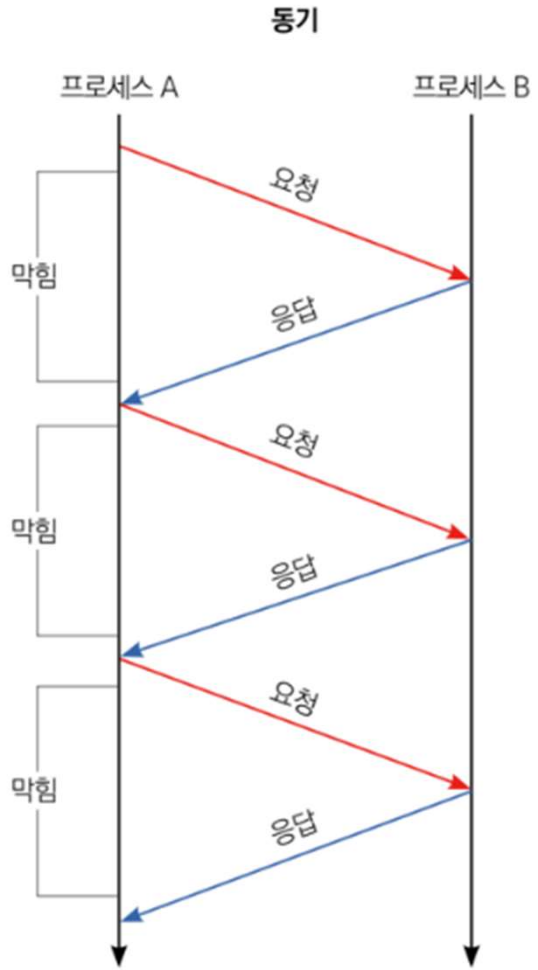
Chapter03

다트 비동기 프로그래밍

목차

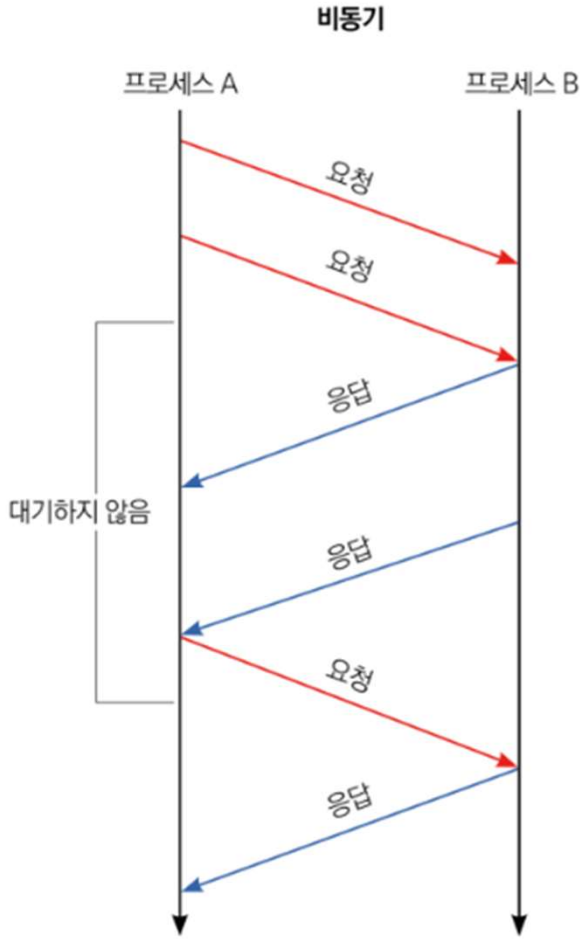
동기식 프로그래밍 이란?

프로그램이 작업을
순차적으로 실행하는
프로그래밍 방식을 말합니다.

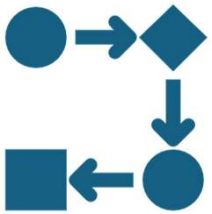


비동기식 프로그래밍이란?

- 작업이 완료되기를 기다리지 않고 다음 코드 라인으로 넘어가는 프로그래밍 방식을 말합니다.



동기식 대 비동기식



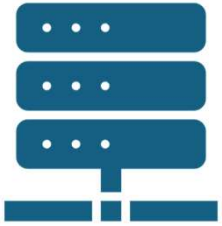
두 접근 방식의 차이점

실행 흐름

자원 활용

응답성

복잡성



비동기식 프로그램이 필요한 경우

웹 서버

데이터베이스 쿼리

파일 I/O

네트워크 요청

플러터의 비동기식 프로그래밍 모델

- Future
  - Future 클래스는 미래에 받아올 값을 뜻합니다. List나 Set처럼 제네릭으로 어떤 미래의 값을 받아올지 정할 수 있습니다.
  - 예제 : 비동기적으로 파일에서 데이터 읽기

```
import 'dart:io';

// 비동기적으로 파일에서 데이터를 읽습니다.
Future<String> readFile(String filepath) async {
  try {
    final file = File(filepath);
    String contents = await file.readAsString();
    return contents;
  } catch (e) {
    // 에러 처리
    return 'Error: $e';
  }
}

void main() {
  readFile('example.txt').then((contents) => print(contents));
}
```

플러터의 비동기식 프로그래밍 모델

- Stream
  - 시간이 지남에 따라 여러 개의 비동기 이벤트를 나타냅니다
  - 예제 : 숫자를 시간 간격으로 비동기적으로 생성

```
import 'dart:async';

// 일정 시간 간격으로 숫자를 생성하는 Stream을 생성합니다.
Stream<int> generateNumbers(int max) async* {
  for (int i = 0; i < max; i++) {
    await Future.delayed(Duration(seconds: 1)); // 1초 대기
    yield i; // 숫자를 스트림에 추가
  }
}

void main() {
  // 생성된 숫자를 출력합니다.
  generateNumbers(5).listen((number) {
    print(number);
  });
}
```

플러터의 비동기식 프로그래밍 모델

- async
  - 함수가 비동기적으로 실행될 것임을 나타냄
- Await
  - 비동기 연산의 완료를 기다리는 동안 현재 함수의 실행을 일시 중지

- 설명
  - Async함수 : 함수 앞에 'async'를 붙여서 선언, 'Future'를 반환
  - Await 표현식 : 'async'함수 내에서, 비동기 연사의 결과가 필요할 때 'await'를 사용하여 해당 연산의 완료를 기다림. 'await' 다음에 오는 표현식은 'Future'를 반환

플러터의 비동기식 프로그래밍 모델

- 예시 :

```
import 'package:http/http.dart' as http;

// 비동기적으로 웹 API로부터 데이터를 가져오는 함수
Future<void> fetchUserData() async {
  final response = await http.get(Uri.parse('https://jsonplaceholder.typicode.com/users/1'));

  if (response.statusCode == 200) {
    // 성공적으로 데이터를 받아왔을 때의 처리
    print('User data: ${response.body}');
  } else {
    // 에러가 발생했을 때의 처리
    throw Exception('Failed to load user data');
  }
}

void main() {
  fetchUserData();
}
```

간단한 코드 활용 예시1

- 네트워크 요청 :
  - import 'package:http/http.dart' as http;
  - // 비동기적으로 웹 API에서 데이터를 가져옵니다.
  - Future<void> fetchData() async {
  - final response = await http.get(Uri.parse('https://example.com/data'));
  - print(response.body);
  - }
  - void main() {
  - fetchData();
  - }

간단한 코드 활용 예시2

- 파일 쓰기 작업 :
  - import 'dart:io';

  - // 비동기적으로 파일에 데이터를 씁니다.
  - Future<void> writeFile(String filepath, String data) async {
  - final file = File(filepath);
  - await file.writeAsString(data);
  - print('File write complete');
  - }

  - void main() {
  - writeFile('example.txt', 'Hello, Flutter!');
  - }

간단한 코드 활용 예시3

- 타이머/시간 지연 :
  - import 'dart:async';

  - // 지정된 시간 후에 작업을 실행합니다.
  - Future<void> delayedPrint(int seconds, String message) async {
  - await Future.delayed(Duration(seconds: seconds));
  - print(message);
  - }

  - void main() {
  - delayedPrint(5, 'This message is printed after 5 seconds');
  - }

실습

- 어떤 비동기식 프로그래밍이 가능할까요?