Do it! 플러터 앱 프로그래밍

2장 다트(Dart) 기초

목차

02-1 기초문법

- 다트는 main() 함수로 시작합니다.
- 다트는 어디에서나 변수를 선언하고 사용할 수 있습니다.
- 다트에서는 모든 변수가 객체입니다. 그리고 모든 객체는 Object 클래스를 상속받습니다.
- Java 와 C#과 비슷한 형태로 코드를 작성할 수 있습니다.

다트 프로그램 예

```
// 함수 정의
printInteger(int aNumber) {
  print('The number is $aNumber.');  // 콘솔에 출력
}

// main() 함수에서 시작
main() {
  var number = 42;       // 동적 타입 변수 지정
  printInteger(number);  // 함수 호출
}
```

▶ 실행 결과

```
The number is 42.
```

02-1 기초문법

https://dart.dev/

- 다트는 자료형이 엄격한 언어입니다. 이 말은 변수에 지정한 자료형과 다른 유형의 값을 저장하면 오류가 발생한다는 의미입니다. 만약 여러 자료형을 허용하려면 dynamic 타입을 이용할 수 있습니다.
- 다트는 제네릭 타입을 이용해 개발할 수 있습니다. 그리고 List<int>처럼 int형을 넣을 수 도 있고, List<dynamic>처럼 다양한 데이터를 넣을 수도 있습니다.
- 다트는 public, protected 같은 키워드가 없습니다. 만약 외부로 노출하고 싶지 않다면 변 수나 함수 이름 앞에 언더스코어(_)를 이용해 표시할 수 있습니다.

표 2-1 다트가 제공하는 주요 자료형

구분	자료형	설명
숫자	int	정수형 숫자. 예 (1, -500, 0)
	double	실수형 숫자. 예 (3.14, -7.1)
	num	정수형 또는 실수형 숫자
문자열	String	텍스트 기반 문자
불리언	bool	True나 False
자료형 추론	var	입력받은 값에 따라 자료형 결정. 한 번 결정된 자료형은 변경 불가
	dynamic	입력받은 값에 따라 자료형 결정. 다른 변수 입력하면 자료형 변경 가능

```
// class 내부 함수는 언더스코어 ( _ )로 표시
String temperature = _readThermometer();
```

02-1 기초문법

구구단 프로그램 만들기

- **실습 1-1**
  - 2단부터 9단까지 순차적으로 나오는 구구단 프로그램 완성
  - For문을 이용한 반복문 생성
  - Print를 이용한 결과값 출력

구구단 프로그램 예

```
void main() {
  int i;
  int j;

  for (i = 2; i <= 9; i++) {
    for (j = 1; j <= 9; j++) {
      print('$i * $j = ${i * j}');
    }
  }
}
```

▶ 실행 결과

```
2 * 1 = 2
2 * 2 = 4
(...생략...)
9 * 8 = 72
9 * 9 = 81
```

02-2 Dart 비동기 처리하기

- 다트는 비동기 처리를 지원하는 언어입니다.
- 비동기(asynchronous)란 언제 끝날지 모르는 작업을 기다리지 않고 다음 작업을 처리하게 하는 것을 의미합니다

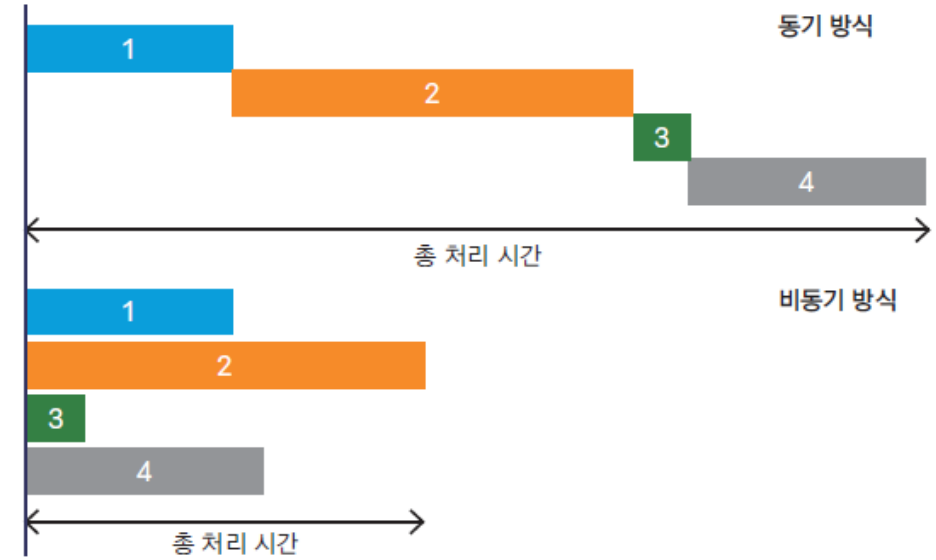

그림 2-1 동기와 비동기 처리 방식 비교

02-2 Dart 비동기 처리하기

- 다트는 async와 await 키워드를 이용해 비동기 처리를 구현합니다.

1. 함수 이름 뒤, 본문이 시작하는 중괄호 { 앞에 async 키워드를 붙여 비동기로 만든다.
2. 비동기 함수 안에서 언제 끝날지 모르는 작업 앞에 await 키워드를 붙인다.
3. 2번 작업을 마친 결과를 받기 위해 비동기 함수 이름 앞에 Future(값이 여러 개면 Stream) 클래스를 지정한다.

비동기 처리 예

```
void main() {
  checkVersion();
  print('end process');
}
Future checkVersion() async {
  var version = await lookUpVersion();
  print(version);
}

int lookUpVersion() {
  return 12;
}
```

▶ 실행 결과

```
end process
12
```

02-2 Dart 비동기 처리하기

- 비동기 함수가 반환하는 값을 처리하려면 then() 함수를 이용합니다.
- lookUpVersionName()에서 리턴한 값을 getVersionName() 함수에서 호출
- then(value) => { } 함수를 이용하여 value의 값을 출력

then() 함수 사용법

```
void main() async {
  await getVersionName().then((value) => {
    print(value)
  });
  print('end process');
}

Future<String> getVersionName() async {
  var versionName = await lookUpVersionName();
  return versionName;
}

String lookUpVersionName() {
  return 'Android Q';
}
```

▶ 실행 결과

```
Android Q
end process
```

02-2 Dart 비동기 처리하기

다트의 스레드

- Future.delayed() 함수는 Duration 기간 동안 기다린 후에 진행
- Duration에는 분(minutes)이나 밀리초(milliseconds) 등 다양한 값을 넣을 수 있습니다.

await 키워드 활용 예

```
void main() {
  printOne();
  printTwo();
  printThree();
}

void printOne() {
  print('One');
}

void printThree() {
  print('Three');
}

void printTwo() async {
  Future.delayed(Duration(seconds: 1), () {
    print('Future!!');
  });
  print('Two');
}
```

1번

3번

4번

2번

▶ 실행 결과

```
One
Two
Three
Future!!
```

02-3 JSON 데이터 주고받기

JSON

- **JSON** (JavaScript Object Notation)은 경량의 DATA-교환 형식이다.
- Key – value 형태로 이루어짐
- 배열의 경우 대괄호 ([,])를 이용하여 json Data 생성

Sample JSON Schema

```
{
        "title": "Example Schema",
        "type": "object",
        "properties": {
                "firstName": {
                        "type": "string"
                },
                "lastName": {
                        "type": "string"
                },
                "age": {
                        "description": "Age in years",
                        "type": "integer",
                        "minimum": 0
                }
        },
        "required": ["firstName", "lastName"]
}
```

02-3 JSON 데이터 주고받기

Dart JSON 디코딩

- JSON을 사용하려면 소스에 convert라는 라이브러리를 포함해야 함

- jsonDecode() 함수에 전달한 후 그 결과를 scores 변수에 저장했습니다. jsonDecode() 함수는 JSON 형태의 데이터를 dynamic 형식의 리스트로 변환해서 반환해 줍니다

JSON 데이터 디코딩 예

```dart
import 'dart:convert';

void main() {
  var jsonString = '''
    [
      {"score": 40},
      {"score": 80}
    ]
  ''';
  var scores = jsonDecode(jsonString);
  print(scores is List);              // true 출력
  var firstScore = scores[0];
  print(firstScore is Map);           // true 출력
  print(firstScore['score'] == 40);   // true 출력
}
```

02-3 JSON 데이터 주고받기

Dart JSON 디코딩

- JSON을 사용하려면 소스에 convert라는 라이브러리를 포함해야 함

- jsonDecode() 함수에 전달한 후 그 결과를 scores 변수에 저장했습니다. jsonDecode() 함수는 JSON 형태의 데이터를 dynamic 형식의 리스트로 변환해서 반환해 줍니다

JSON 데이터 디코딩 예

```dart
import 'dart:convert';

void main() {
  var jsonString = '''
    [
      {"score": 40},
      {"score": 80}
    ]
  ''';
  var scores = jsonDecode(jsonString);
  print(scores is List);              // true 출력
  var firstScore = scores[0];
  print(firstScore is Map);           // true 출력
  print(firstScore['score'] == 40);   // true 출력
}
```

02-3 JSON 데이터 주고받기

JSON 인코딩

- Map을 jsonEncode() 함수를 이용하여 변경
- 변경 후 서버에 전달할 수 있도록 json형태의 String으로 변환

JSON 데이터 인코딩 예

```dart
import 'dart:convert';

void main() {
  var scores = [
    {'score': 40},
    {'score': 80},
    {'score': 100, 'overtime': true, 'special_guest': null}
  ];

  var jsonText = jsonEncode(scores);
  print(jsonText ==
      '[{"score":40},{"score":80},'
      '{"score":100,"overtime":true,'
      '"special_guest":null}]');          // true 출력
}
```

Reference

- https://www.javatpoint.com/dart-programming
- https://www.darttutorial.org/
- https://www.w3adda.com/dart-tutorial/dart-introduction
- https://linuxhint.com/category/dart/
- https://dart-tutorial.com/
- https://www.tutorialkart.com/dart/
- https://www.tutorialandexample.com/dart-tutorial