02

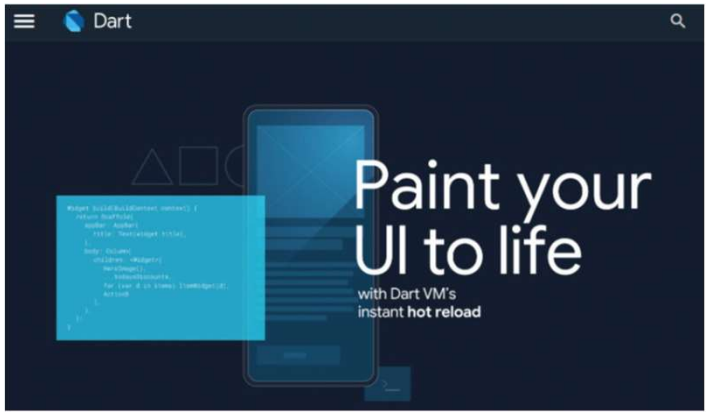
Overview

Dart

다트 언어란?

플러터 덕분에 떠오른 다트

- 다트는 2011년에 구글이 발표한 프로그래밍 언어
- 구글이 플러터를 개발하는 언어로 다트를 선택
- 다트는 크로스 플랫폼에 기반을 둔 프런트엔드 프로그래밍 언어



다트 언어란?

다트 파일 실행하기

- 다트 파일은 main() 함수를 프로그램의 진입점

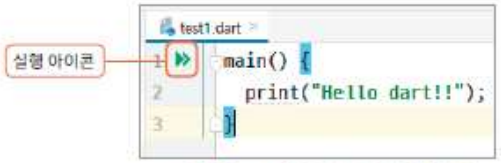


그림 3-2 main() 함수와 다트 파일 실행하기

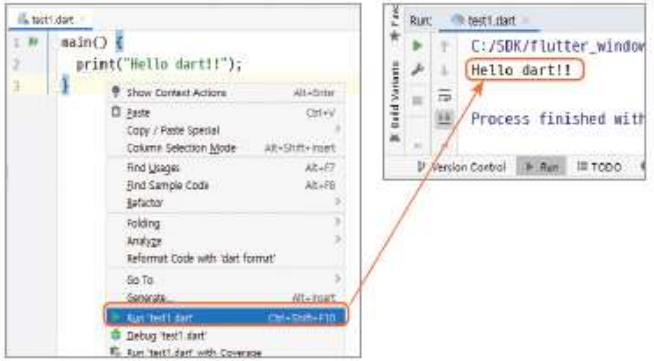


그림 3-3 다트 파일 실행 메뉴(왼쪽)와 결과 확인하기(오른쪽)

다트 언어란?

다트 파일의 구성 요소

- 다트 파일은 톱 레벨에 프로그램의 구성 요소인 변수와 함수, 클래스를 선언

• 톱 레벨에 변수, 함수, 클래스 선언

```
int no = 10;          // 변수 선언

void sayHello() {     // 함수 선언
  print('hello, $no');
}

class User {          // 클래스 선언
  int no = 10;

  void sayHello() {
    print('world, $no');
  }
}
```

다트 언어란?

다트 엔진의 라이브러리

표 3-1 플랫폼별 라이브러리

플랫폼 종류	라이브러리 종류	지원하는 내용
멀티 플랫폼	dart:async	비동기 프로그래밍
	dart:collection	LinkedList, HashMap 등 집합 데이터
	dart:convert	JSON 같은 데이터의 인코딩과 디코딩
	dart:core	내장(built-in) 타입, 컬렉션 등
	dart:developer	디버거나 인스펙터 등 개발자 도구
	dart:math	수학 함수
네이티브 플랫폼	dart:io	파일, 소켓, HTTP 등 앱에서 발생하는 입출력
	dart:isolate	동시성 프로그래밍(일종의 스레드 프로그래밍)
웹 플랫폼	dart:html	HTML 요소
	dart:indexed_db	키-값 형태의 데이터 저장
	dart:web_audio	오디오 핸들링
	dart:web_gl	3D 그래픽
	dart:web_sql	SQL 기반 데이터 저장

다트 파일 구성

라이브러리 불러오기 – import

- 선언하지 않고 사용할 수 있는 라이브러리는 다트 엔진에서 제공하는 dart:core

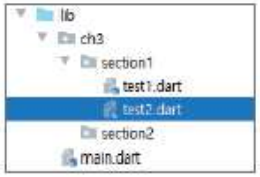


그림 3-4 파일 구성 예

• test1.dart

```dart
int no = 10;
void sayHello() {
 print('hello, $no');
}

class User {
 int no = 10;
  void sayHello() {
    print('world, $no');
  }
}
```

• test2.dart

```dart
import 'test1.dart';

main() {
  print('$no');
  sayHello();
  User user = User();
  user.sayHello();
}
```

다트 파일 구성

라이브러리 불러오기 – import

- 상대 경로로 불러오기

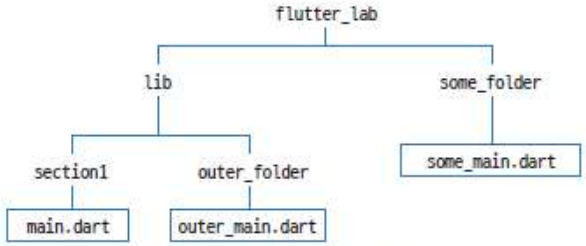


그림 3-5 파일 구성 예

• main.dart

```
import '../outer_folder/outer_main.dart';    // 성공
import '../../some_folder/some_main.dart';   // 오류
```

다트 파일 구성

라이브러리 불러오기 – import

- package 접두사로 불러오기

그림 3-6 package import 구성

• package 접두사로 불러오기

```
import 'package:flutter_lab/outer_folder/outer_main.dart';
```

- dart 접두사로 불러오기
  - 다트 언어에서 기본으로 제공하는 라이브러리를 불러올 때는 dart 접두사를 사용

• dart 접두사로 불러오기

```
import 'dart:core';
import 'dart:async';
```

다트 파일 구성

외부에서 사용할 수 없게 제한하기

- 다트에서 프로그램의 구성 요소를 선언하면 기본으로 public 상태
- 다트에서 접근 범위를 한정하려면 밑줄 사용

• test2.dart

```dart
import 'test1.dart';

main() {
  no1 = 20;
  _no2 = 30;      // 오류

  sayHello1();
  _sayHello2();   // 오류

  User1 user1 = User1();
  _User2 user2 = _User2();    // 오류

  user1.name = 'kkang';
  user1._address = 'busan';   // 오류
}
```

• test1.dart

```dart
int no1 = 10;
int _no2 = 20;

void sayHello1() { }
void _sayHello2() { }

class User1 {
  String? name;
  String _address='seoul';
}

class _User2 { }

main() {
  no1 = 20;
  _no2 = 30;

  sayHello1();
  _sayHello2();

  User1 user1 = User1();
  _User2 user2 = _User2();

  user1.name = 'kkang';
  user1._address ='busan';
}
```

다트 파일 구성

식별자에 별칭 정의하기 – as

- as 예약어는 식별자에 별칭을 정의할 때 사용

• 별칭 정의

```dart
import 'test1.dart' as Test1;

main() {
  no1 = 30;   // 오류
  Test1.no1 = 30;

  Test1.sayHello1();
  Test1.User1 user1 = Test1.User1();
}
```

다트 파일 구성

특정 요소 불러오기 - show

• 특정 요소만 불러오기

```dart
import 'test1.dart' show no1, User1;

main() {
  no1 = 30;
  User1 user1 = User1();

  sayHello1();   // 오류
}
```

다트 파일 구성

특정 요소 제외하기 - hide

• 특정 요소만 제외

```
import 'test1.dart' hide sayHello1, User1;

main() {
  no1 = 30;
  sayHello1();   // 오류
  User1 user1 = User1();   // 오류
}
```

라이브러리 만들기

- a.dart, b.dart 파일을 작성하고 두 파일을 myLib.dart에 포함해 외부에서 이용할 때는 myLib.dart 파일만 불러와 사용
- a.dart, b.dart 파일을 만들 때 이 파일의 내용을 라이브러리에 등록

• a.dart

```
part of my_lib;
int aData = 10;
```

- 이렇게 part of로 선언한 다트 파일은 외부에서 import 구문으로 이용 불가
- part of로 선언된 파일을 포함해 외부에 공유할 다트 파일은 다음처럼 library와 part로 선언

• myLib.dart

```
library my_lib;
part 'a.dart';
part 'b.dart';
```

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare