1. **등장배경**

* **iOS :** 오브젝티브-C, 스위프트로 개발
* **안드로이드 :** 자바, 코틀린으로 개발
* **네이티브 앱 :** 각 모바일 운영체제에 맞는 언어로 개발한 앱.
* 기존 방식으론 iOS와 안드로이드용으로 각각 **앱을 두번 개발**해야함.
* **하나의 소스로 iOS와 안드로이드 모두에서 실행**할 수 있는 방법을 찾기 시작한 결과 **하이브리드 앱, 웹앱** 탄생
* **웹앱 :** 웹 기술을 이용해서 만든 앱, 앱의 화면을 나타내는 뷰를 모바일용 웹으로 만들어서 다양한 기종과 해상도에 대응하며 빠르게 개발 가능
* **하이브리드 앱 :** 웹앱을 만든 후 별도의 프레임워크를 이용해 운영체제별로 동작하는 앱을 만드는 기술
* 하이브리드 앱, 웹앱은 네이티브 앱에 비해서 **상대적으로 속도가 느리고 애니메이션 사용에도 제약**이 있음. 이를 해결하기 위해 **리액트 네이티브와 플러터** 탄생
* **리액트 네이티브 :** 여러 운영체제에서 동작하는 앱을 개발할 수 있는 크로스 플랫폼 앱 개발 프레임워크, **자바스크립트언어** 사용. **(페이스북에서 개발)**
* **플러터 :** 리액트 네이티브와 마찬가지로 크로스 플랫폼 앱 개발 프레임워크, **다트(Dart)라는 언어**를 사용. 엔진계층은 C/C++로 만들어져 있다. **(구글에서 개발)**

1. **다트(Dart)의 특징**

* Main() 함수로 시작한다.
* 어디에서나 변수를 선언하고 사용할 수 있다.
* 모든 변수가 객체이다. 모든 객체는 Object 클래스를 상속받는다.
* 자료형이 엄격한 언어이다. 변수에 지정한 자료형과 다른 유형의 값을 저장하면 오류가 발생한다. 여러 자료형을 허용하려면 dynamic 타입을 이용할 수 있다.
* 제네틱 타입을 이용해 개발할 수 있다. List<int>처럼 int형을 넣을 수도 있고, List<dynamic>처럼 다양한 데이터를 넣을 수도 있다.
* Public, protected 같은 키워드가 없다. 만약 외부로 노출하고 싶지 않다면 변수나 함수 이름 앞에 언더스코어( \_ )를 이용해 표시할 수 있다.
* 변수나 함수의 시작은 언더스코어 또는 문자열로 시작하고 그 이후에 숫자를 입력할 수 있다.
* 삼항 연산자를 사용할 수 있다.
* **비동기 처리 방식을 지원한다.**

1. **비동기 방식**

* **비동기 함수를 선언할 때 Future 키워드 사용** ( ex) Future 함수명 async { … }; )
* 비동기로 선언된 함수 안에 await 키워드로 코드 실행을 대기 시킬 수 있다.
* then() 함수를 사용해 비동기 함수가 반환하는 값을 사용 가능하다.
* **데이터를 순차적으로 받기 위해 Stream 키워드 사용**

( ex) Stream<자료형> 함수명 async\* { … yield 반환값; } )

* **async\* 명령어는 앞으로 yield를 이용해 지속적으로 데이터를 전달하겠다는 의미**이다.
* **Return은 한번 반환하면 함수가 끝나지만 yield는 반환 후에도 계속 함수를** 유지한다.

1. **Future, Stream 예제**

**Future<int> sumStream(Stream<int> stream) async {**

**var sum = 0;**

**await for (var value in stream) {**

**sum += value;**

**}**

**return sum;**

**}**

**Future<int> sumStream(Stream<int> stream) async**

**{**

**var sum = 0;**

**await for (var value in stream) {**

**sum += value;**

**}**

**return sum;**

**}**

**Stream<int> countStream(int to) async\* {**

**for (int i = 1; i <= to; i++) {**

**yield i;**

**}**

**}**

**main() async {**

**var stream = countStream(10);**

**var sum = await sumStream(stream);**

**print(sum); // 55**

**}**