



# Flutter

## \*플러터 개요

Introduction of Flutter

\*플러터(*Flutter*)는  
단일 코드 베이스(*Single codebase*)에  
서 고성능(*High performance*) 크로스  
플랫폼(*Cross platform*) 앱을 만들 수  
있도록 지원하는 오픈소스 프레임워  
크다!

\*플러터란?

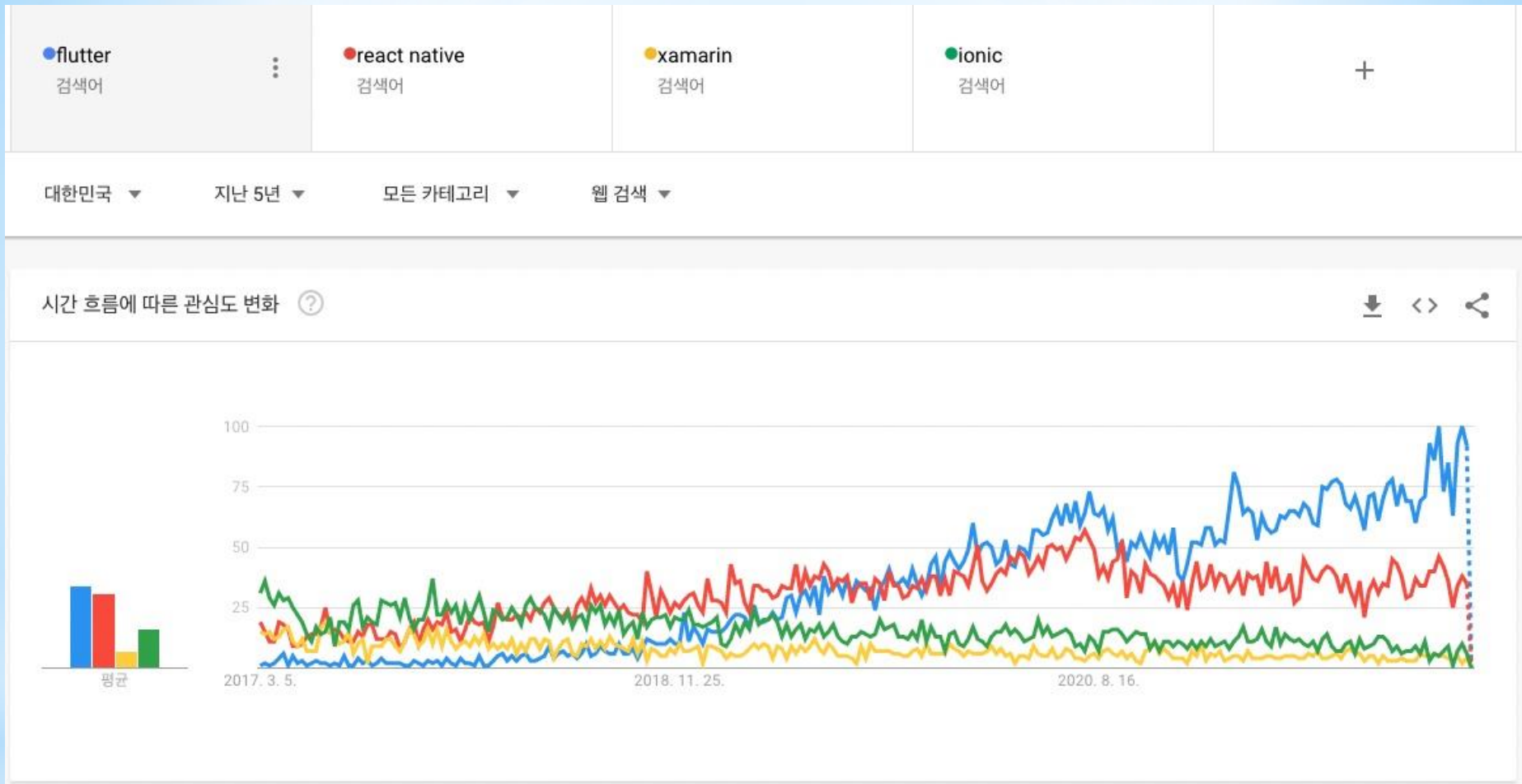
- \* 플러터는 Google의 오픈 소스 프레임워크
- \* 단일 코드 베이스에서 다중 플랫폼 모바일/웹/데스크탑/임베디드 앱을 빌드, 테스트 및 배포할 수 있음
- \* 모든 장치에서 빠르게 수행될 수 있도록 ARM, Intel 기계어 및 JavaScript로 컴파일함
- \* 핫 리로드(*Hot reload*)로 상태를 유지하면서 빠르게 결과를 확인할 수 있음
- \* 모든 픽셀을 제어하여 모든 화면에서 유연한 맞춤형 디자인을 만듦

## \* 플러터 상세요약

## \* 크로스 플랫폼

- \* 전문적인 앱 개발자에게도 Android와 iOS에서 모두 잘 동작하는 앱을 개발하는 것은 몹시 어려움
- \* 이런 이유로 하나의 코드 베이스에서 개발하면 각각의 플랫폼에 맞게 빌드 해주는 크로스 플랫폼 도구들이 인기있음
- \* 플러터 이전에 리액트 네이티브(*React Native*)가 인기를 끌었으며 현재는 플러터가 인지도를 높여가고 있습니다.

# \* 플러터는 인기있나?



\* 구글 트렌드 조사

\* iOS 개발자 Bob이 있습니다. Bob은 나의 취향에 맞는 음식 추천 서비스를 개발했습니다. 이 앱은 꽤 인기가 있는데 한 가지 문제가 있습니다. 사용자들이 이런 질문을 하는 것이었죠.

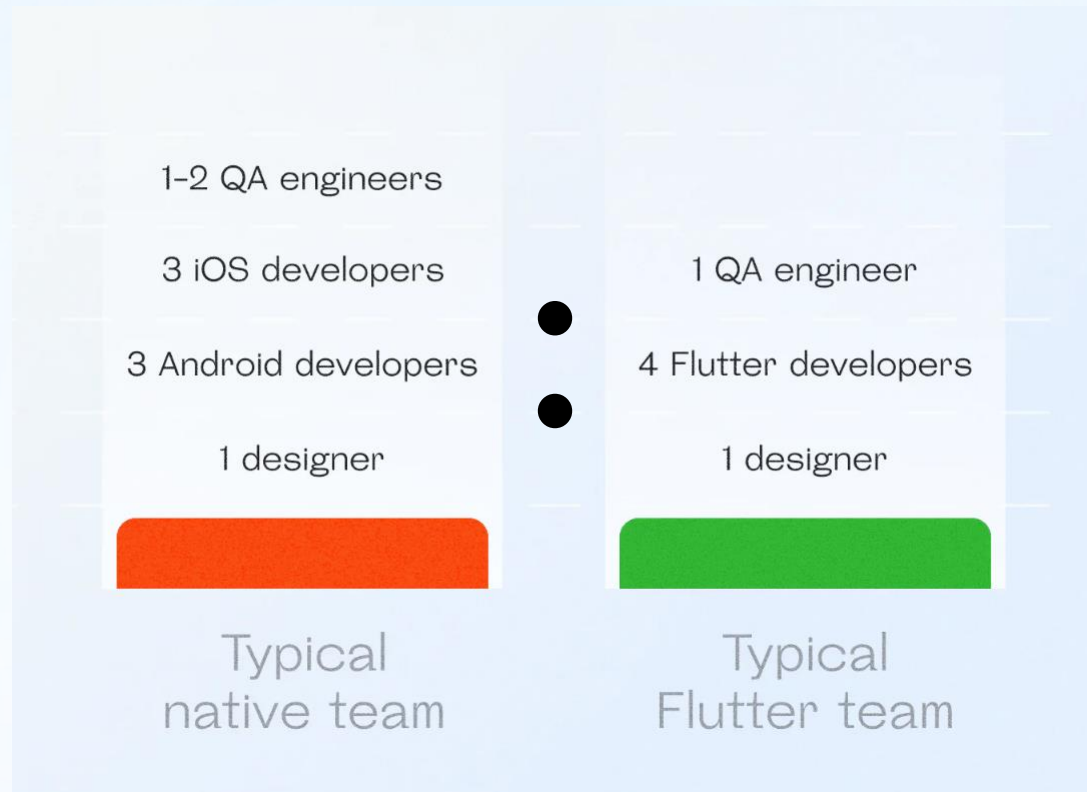
\* "안드로이드는 언제 나오나요?"

\* Bob은 Android 앱을 만들기로 결심하고 Java와 Android를 공부하기 시작합니다. 마침내 Bob은 자신의 웹사이트에 "Get Google Play and App Store" 두 버튼을 모두 갖게 됩니다.

이제 Bob은 앱에 기능을 추가하거나 버그를 수정하려 할 때 서로 다른 기술의 두 개의 코드 베이스를 수정해야 하기 때문에 작업량이 두 배입니다. 관리해야 할 마켓, OS, 단말기 등 모든 일이 이전보다 힘들어집니다.

\* 크로스 플랫폼의 필요성

9



6

\*일반 네이티브 vs 플러터

- \* 닥트는 모든 플랫폼에서 빠르게 실행되는 앱을 위한 클라이언트에 최적화된 언어입니다.
- \* 유저 인터페이스 생성 요구에 특화된 프로그래밍 언어로 개발되었습니다.
- \* 반복적인 변경에 대해 핫 리로드를 이용해 실행 중인 앱에서 즉시 결과를 확인할 수 있습니다.
- \* 모바일/데스크톱/백엔드용 ARM, x64 기계어로 컴파일하거나 웹용 JavaScript로 컴파일 할 수 있습니다.

## \* 닥트 언어











































기본적으로 C언어의 문법과 거의 같으며 Java, C#, Javascript와 같은 기능적 스트럭처를 추가한 언어로, 언급된 언어보다 간결하고 강력한 기능을 지원



# Worst Programming Languages to Learn in 2018 Rankings

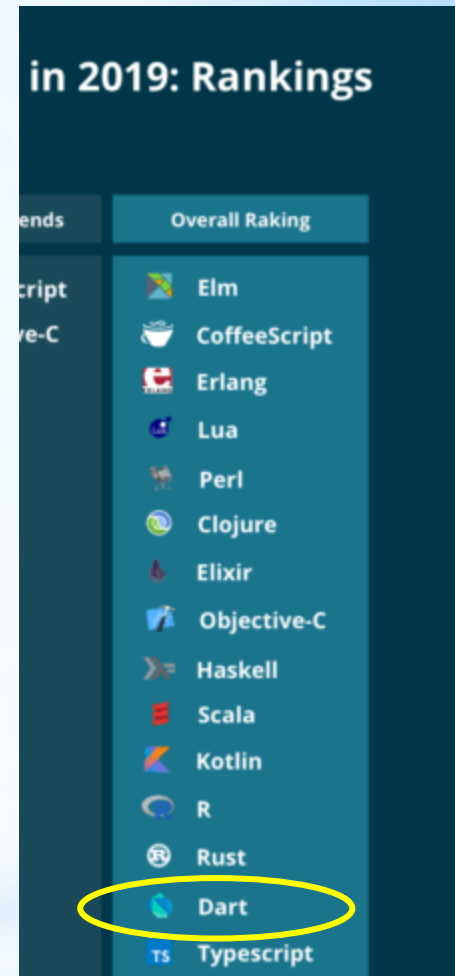
Ranked from Worst to Best Languages to Learn

	Overall Rankings	Community Engagement	Job Market	Growth and Trends
1	 Dart	 Dart	 Dart	 Objective-C
2	 Objective-C	 CoffeeScript	 Rust	 CoffeeScript
3	 CoffeeScript	 Objective-C	 Elm	 Dart
4	 Erlang	 Lua	 Lua	 Perl
5	 Lua	 Elm	 Erlang	 Erlang
6	 Clojure	 Clojure	 Clojure	 Clojure
7	 Perl	 Elixir	 Kotlin	 Ruby
8	 Elm	 Erlang	 Elixir	 C#
9	 Elixir	 Kotlin	 R	 Lua
10	 Haskell	 Perl	 Perl	 C

\* 닥트언어는 인기없다?!

\* 2018년에는 배우기 싫은 언어 1위였던 다트가  
2018년 12월 플러터 1.0이 발표되면서 2019년에는  
14위로 자리를 옮김

\* 이런 변화는 스택 오버플로우에서 진행하는 개발자 설문조사에서도 나타나는데, 그야말로 플러터 덕분에 다트 언어의 입지가 생김!



	Overall Ranking
Elm	
CoffeeScript	
Erlang	
Lua	
Perl	
Clojure	
Elixir	
Objective-C	
Haskell	
Scala	
Kotlin	
R	
Rust	
Dart	
Typescript	

\* 인기가 올라감?!

- \* 다트는 JIT(*Just in time*)과 AOT(*Ahead of time*)의 두 가지 방식의 컴파일을 지원
- \* 대상 플랫폼과 목표에 따라서 다양한 방식으로 실행하는 것을 지원
- \* JIT 컴파일을 통해 개발 시점에 빠른 주기의 개발이 가능하도록 지원
- \* AOT 컴파일은 프로덕션에서 더 빠르고 안정적으로 실행될 수 있도록 최적화함
- \* 이런 유연한 컴파일 지원은 몹시 빠른 앱 개발 주기와 빠른 실행 및 시작 시간이라는 장점을 제공하면서 앱 개발 프로세스를 혁신함

## \* 유연한 컴파일 지원

iOS 앱을 개발하는 경우 수정 결과를 보기 위해  
서 컴파일 및 시뮬레이터를 재시작하는데 걸리  
는 시간이 길게는 1분 이상 소요되기도 함!

- \* JIT 컴파일 덕분에 플러터 앱 개발 시 현재 화면의 상태를 유지한 채로 변경된 부분만 다시 적용하여 결과를 확인
- \* 수정 내용에 오류가 있는 경우에도 오류를 수정하고 나면 다시 원래의 상태를 유지
- \* 수정 결과를 확인할 수 있음. 기다리지 않고 빠르게 원하는 것을 시도해 볼 수 있다는 사실은 개발자들에게 훌륭한 개발 경험을 제공함

\* **Stateful hot reload**