DevFest

Bogor 2024

Evolving Dart: Macros

Erico Darmawan Handoyo

Google Developer Groups

Erico Darmawan Handoyo
Private Training & Online School

Erico Darmawan Handoyo

- Parahyangan Catholic University
- Bandung Institute of Technology

- Lecturer @ Maranatha Christian University
- The Most Awesome Spectacular Legendary Programming Book Author
- Flutter tutorial creato[illegible]be
- Flutter Trainer

Erico Darmawan Handoyo
Private Training & Online School

Google Developer Groups

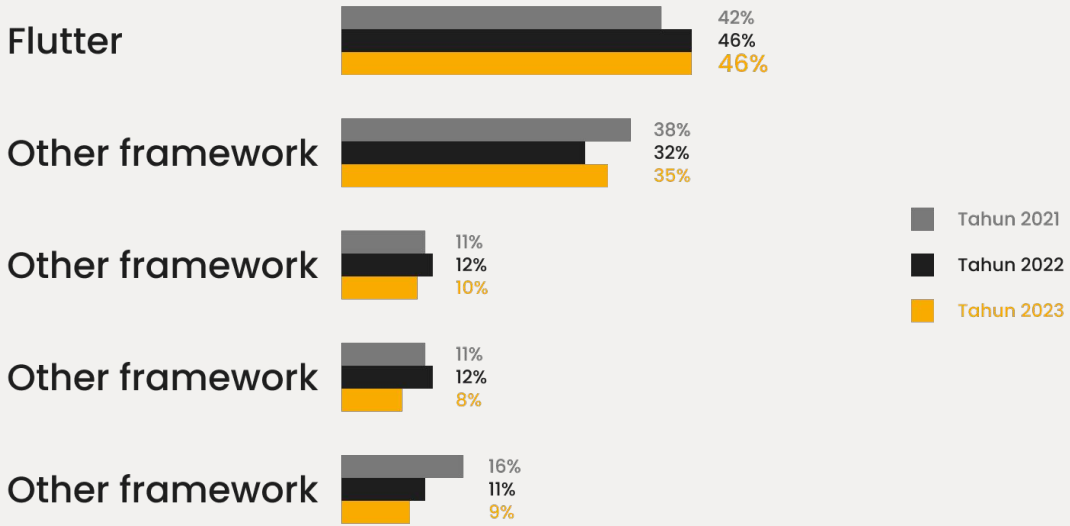
- Traveling ✈️✈️✈️✈️✈️
- Eating 🍗🍗🍗🍗🍗
- Watching movies 🎬🎬🎬🎬
- Anything costs money 💸💸💸💸💸
- ~~Coding~~ 💻➡️

Erico Darmawan Handoyo
Private Training & Online School

Google Developer Groups

youtube.com/@ericodarmawan
s.id/fb-ericodh
s.id/komunitas-flutter
s.id/ig-ericodh
s.id/flixid
s.id/in-ericodh
ericodarmawan.com

Which cross-platform mobile frameworks do you use?



Erico Darmawan Handoyo
Private Training & Online School

Source: https://www.jetbrains.com/lp/devecosystem-2023/development
https://www.jetbrains.com/lp/devecosystem-2022/miscellaneous

= Flutter+

Flock is Flutter+ • Flock is a fork of Flutter. Flock stays up to date with Flutter, and also adds new community features.

Flock = Flutter+

https://youtu.be/jAHH33e3Nd8

Why do Mobile Developers Love Flutter?

- Flutter is platform-agnostic
- Flutter apps offer an excellent user interface and experience
    - Flutter's widget system simplifies creating great-looking interfaces and works well across platforms.
    - It doesn't need platform-specific UI components.
- Numerous useful open-source packages.

Erico Darmawan Handoyo
Private Training & Online School

Google Developer Groups

Why do Mobile Developers Love Flutter?

- Flutter is easy to learn
  - Learning Flutter & Dart is easy for Java, Python, or C# devs due to its familiar syntax.
  - Excellent documentation.

```java
public class Person {
    public String name;
    public int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }
}

Person person = new Person("Alice", 30);
```

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

person = Person("Alice", 30)
```

```dart
class Person {
  final String name;
  final int age;

  Person({required this.name, required this.age});
}

Person person = Person(name: 'Alice', age: 30);
```

Google Developer Groups

Flutter
@DevFest

Google
Developer
Groups

Evolving Dart

Erico Darmawan Handoyo
Private Training & Online School

Evolving Dart

- Null Safety: prevents null errors by ensuring variables can't be null unless specified.
- Enhanced enum: allow enum to have additional features such as fields, methods, and constructors.
- Dart 3 new class modifiers: mixin, final, seal, base, & interface class.

```dart
enum Role {
  admin('Administrator', 'Has full access'),
  user('User', 'Has limited access'),
  guest('Guest', 'Has minimal access');

  final String name;
  final String description;

  const Role(this.name, this.description);

  void describe() {
    print('$name: $description');
  }
}
```

Google Developer Groups

Erico Darmawan Handoyo
Private Training & Online School

Flutter
@DevFest

Google
Developer
Groups

Macros

Erico Darmawan Handoyo
Private Training & Online School

Build Runner Package

Provides a concrete way of generating files using Dart code.

```dart
import 'package:json_annotation/json_annotation.dart';

part 'person.g.dart';

@JsonSerializable()
class Person {
  final String name;
  final int age;

  Person({required this.name, required this.age});
}
```

```
> dart run build_runner build
```

```dart
// // GENERATED CODE - DO NOT MODIFY BY HAND

part of 'person.dart';


Person _$PersonFromJson(Map<String, dynamic> json) => Person(
      name: json['name'] as String,
      age: (json['age'] as num).toInt(),
    );

Map<String, dynamic> _$PersonToJson(Person instance) => <String, dynamic>{
      'name': instance.name,
      'age': instance.age,
    };
```

What is Dart Macro?

Dart macro is a user-definable piece of code that takes in other code as parameters and operates on it in real-time to create, modify, or add declarations.

```dart
import 'package:json/json.dart';

@JsonCodable()
class Person {
  final String name;
  final int age;

  Person({required this.name, required this.age});
}
```

```dart
augment library 'file:///Users/ericodh/Dart%20Project/macros_demo/bin/macros_demo.dart';

import 'dart:core' as prefix0;

augment class Person {
  external Person.fromJson(prefix0.Map<prefix0.String, prefix0.Object?> json);
  external prefix0.Map<prefix0.String, prefix0.Object?> toJson();
  augment Person.fromJson(prefix0.Map<prefix0.String, prefix0.Object?> json, )
      : this.name = json[r'name'] as prefix0.String,
        this.age = json[r'age'] as prefix0.int;
  augment prefix0.Map<prefix0.String, prefix0.Object?> toJson() {
    final json = <prefix0.String, prefix0.Object?>{};
    json[r'name'] = this.name;
    json[r'age'] = this.age;
    return json;
  }
}
```

Google Developer Groups

Erico Darmawan Handoyo
Private Training & Online School

How to Use Macro?

1. Switch to the Flutter master channel.
2. Run dart --version and make sure you have Dart version 3.5.0-152 or later.
3. Edit the SDK constraint in your pubspec to require the Dart version: sdk: ^3.5.0-152.
4. Add the package json to dependencies: dart pub add json.

How to Use Macro?

5. Enable the experiment in your package's analysis_options.yaml file. file at the root of your project:

6. Run your project with the experimental flag:

   dart run --enable-experiment=macros your_file.dart

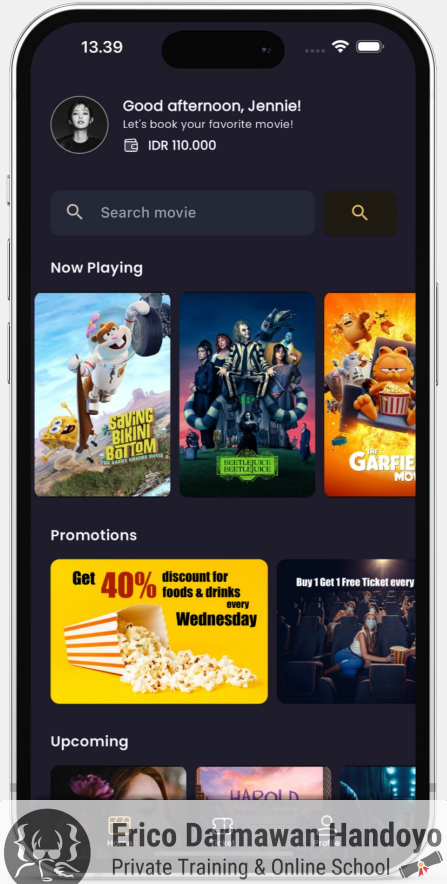
Flutter
@DevFest

Google
Developer
Groups

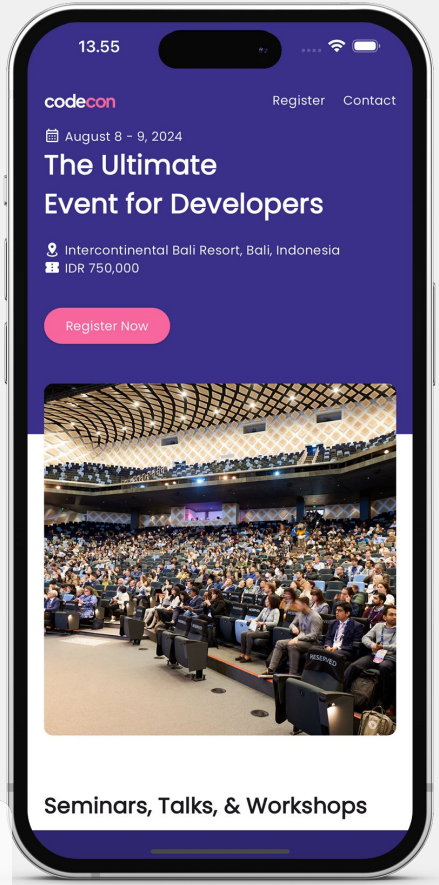
How to Create Macro (Demo)

Erico Darmawan Handoyo
Private Training & Online School

flixid



codecon



Thank You

SPECIAL OFFER 50% OFF

~~IDR 399.000~~

IDR 199.000

9 - 13 November 2024

s.id/flixid

s.id/codecon

Kode Promo: **DEVFESTBOGOR24**

Erico Darmawan Handoyo
Private Training & Online School