Reusable & Customizable Functions using Composition

Erico Darmawan Handoyo
Lecturer, Flutter Trainer

Erico Darmawan Handoyo

- Parahyangan Catholic University
- Bandung Institute of Technology

- Lecturer @ Maranatha Christian University
- The Most Awesome Spectacular Legendary Programming Book Author
- Flutter tutorial creator @ YouTube
- Flutter Trainer

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

The more you code, the greater the chance of making mistakes

The more you code, the greater the chance of making mistakes

What to do:

- Your codes must be reusable.
- Your codes must be customizable.

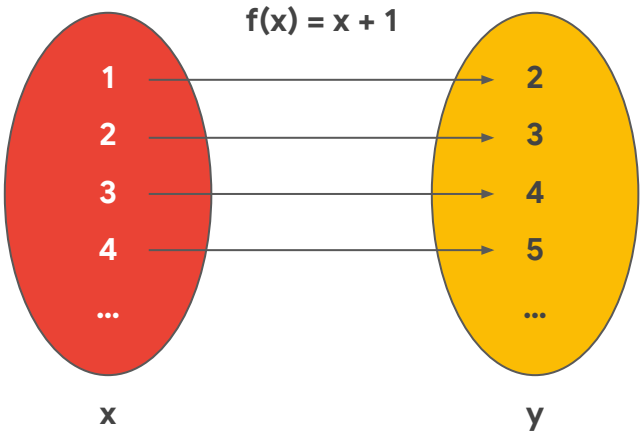
Reusable & customizable functions using **Composition**

The benefits:

- Shorter development time.
- Reducing the possibility of making mistakes.
- Efficient use of resources.

Functions in Mathematics

Function: an expression, rule, or law that defines a relationship between domain and codomain.

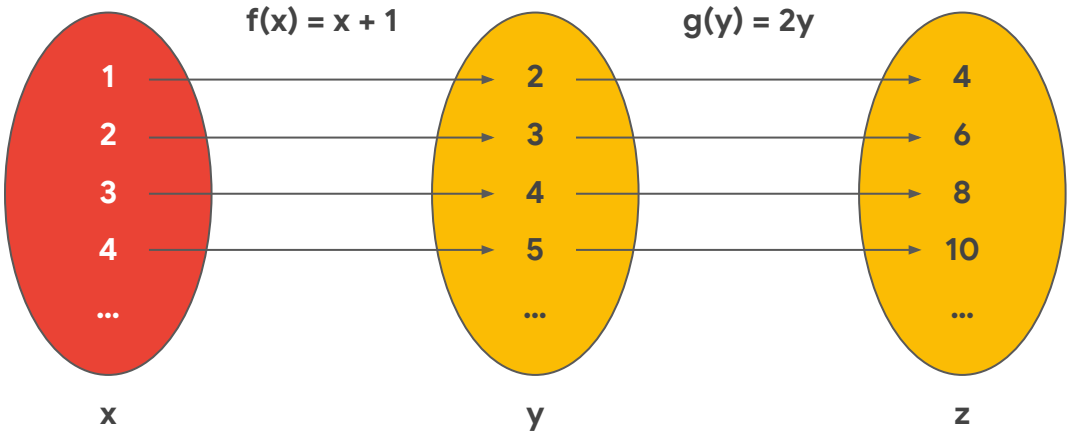



```
int f(int x) {
  return x + 1;
}
```

```
int f(int x) => x + 1;
```

Function Composition in Mathematics

The combination of two or more function to form a new function



$y = f(x) = x + 1$

$z = g(y) = 2y$

$z = (g \circ f)(x) = g(f(x))$

How to Make Function Composition

1. Make reusable simple functions which has one thing to do.
2. Combine those simple functions to make a more complex function as required, using following rules:
    a. Put the first function to execute at the end of the list and always add the next function at the start of the list.
    b. The return type of a function must be the same as the parameter type of the function to be executed next.
    c. Create a function that runs the list of functions backwards.

Function Composition Example

Example 01: Calculating Total Price

```dart
void main(List<String> args) {
  double price = 125000;

  double totalPrice = calculatePrice(price, [applyTax(11), applyDiscount()]);
  print(totalPrice);
}

double Function(double price) applyDiscount([double discount = 20]) =>
    (price) => price - price * discount / 100;

double Function(double price) applyTax([double tax = 10]) =>
    (price) => price + price * tax / 100;

double calculatePrice(double price, List<Function> modifiers) {
  for (var modifier in modifiers.reversed) {
    price = modifier(price);
  }

  return price;
}
```

See also:

- Positional & Named Parameter
- Arrow Syntax, Function sebagai first-class Object, Anonymous Function
- Var & Dynamic Data Type
- List & List Mapping

Example 02: Naming Convention (snake_case → PascalCase)

```
List<String> splitWordsWithUnderscore(String string) {
  return string.split('_');
}

List<String> capitalizeWords(List<String> words) {
  return words
      .map((word) => word[0].toUpperCase() + word.substring(1))
      .toList();
}

String joinWords(List<String> words) {
  return words.join();
}

String Function(String) stringModifier(List<Function> modifiers) => (string) {
      dynamic result = string;
      for (var modifier in modifiers.reversed) {
        result = modifier(result);
      }

      return result;
    };
```

See also:

- Strings

Example 02: Naming Convention (snake_case → PascalCase)

```dart
void main(List<String> args) {
  String snakeCase = 'this_is_snake_case';

  String Function(String) snakeToPascal = stringModifier([
    joinWords,
    capitalizeWords,
    splitWordsWithUnderscore,
  ]);

  String pascalCase = snakeToPascal(snakeCase);

  print(pascalCase);
}
```

Compose vs Pipe

Compose and pipe serve the same purpose: chaining together several functions and executing them sequentially. The difference is, compose executes the functions from right to left / bottom to top, while pipe executes functions from left to right / top to bottom.

Thank You

youtube.com/@ericodarmawan

s.id/komunitas-flutter

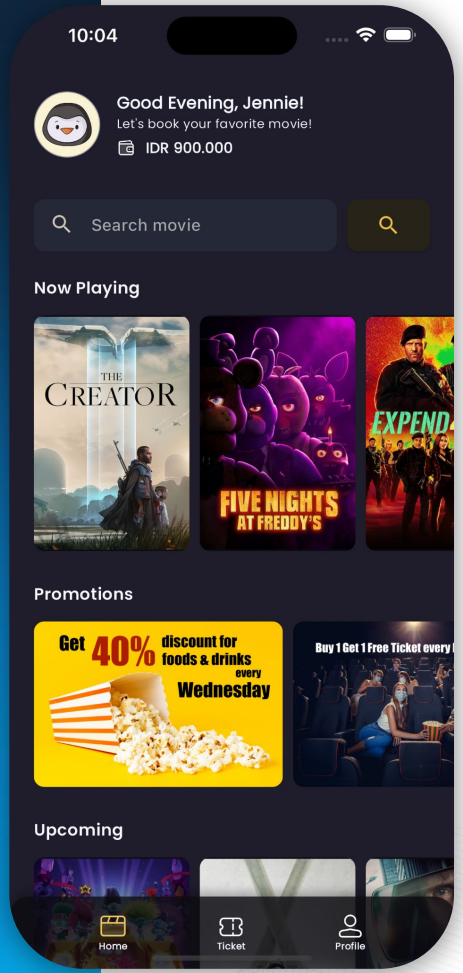
s.id/flixid

ericodarmawan.com

s.id/fb-ericodh

s.id/ig-ericodh

s.id/in-ericodh



Erico Darmawan Handoyo
Flutter Tutorial Youtube Channel

- Basic Programming with Dart
- Object Oriented Programming
- Flutter Essentials
- Flutter State Management
- Flutter Data Storage
- Dart Frog (Backend with Dart)

flixid

- Separation of Concerns
- Clean Architecture
- Riverpod State Management
- Firebase Auth, Storage, & Firestore