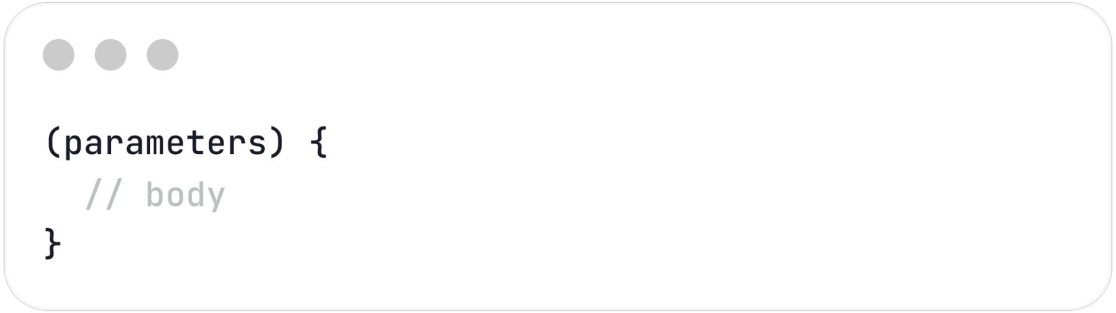
Advanced Functions

Anonymous functions | Higher-order functions (passing functions as arguments) | Recursive functions | Scope → local vs global variables

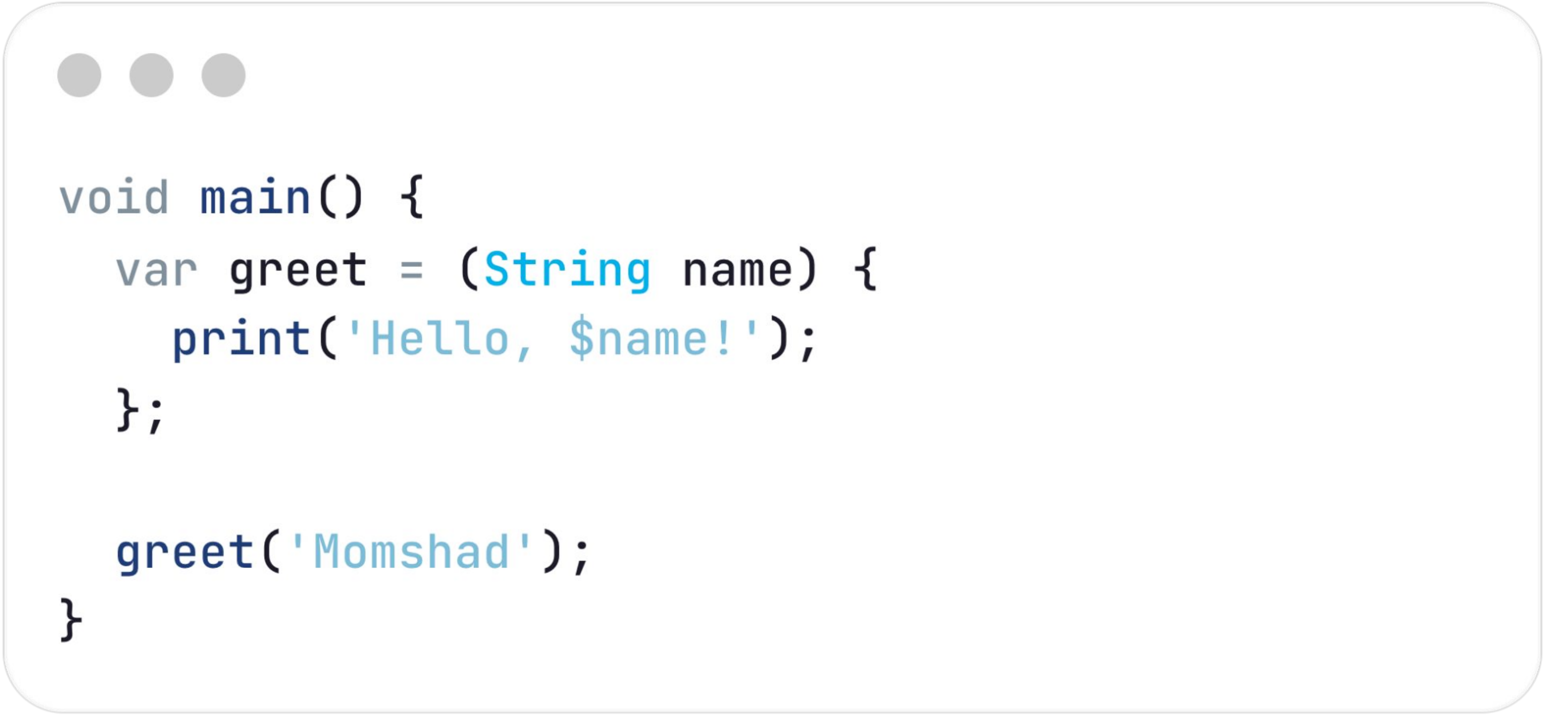
Anonymous Functions 1/3

- Also called **Lambda** or **Inline** functions.
- A function **without a name** — defined where it’s used.
- Often used for **short, one-time tasks**.
- Can be stored in variables or passed as arguments.



```
(parameters) {
  // body
}
```

Anonymous Functions 2/3

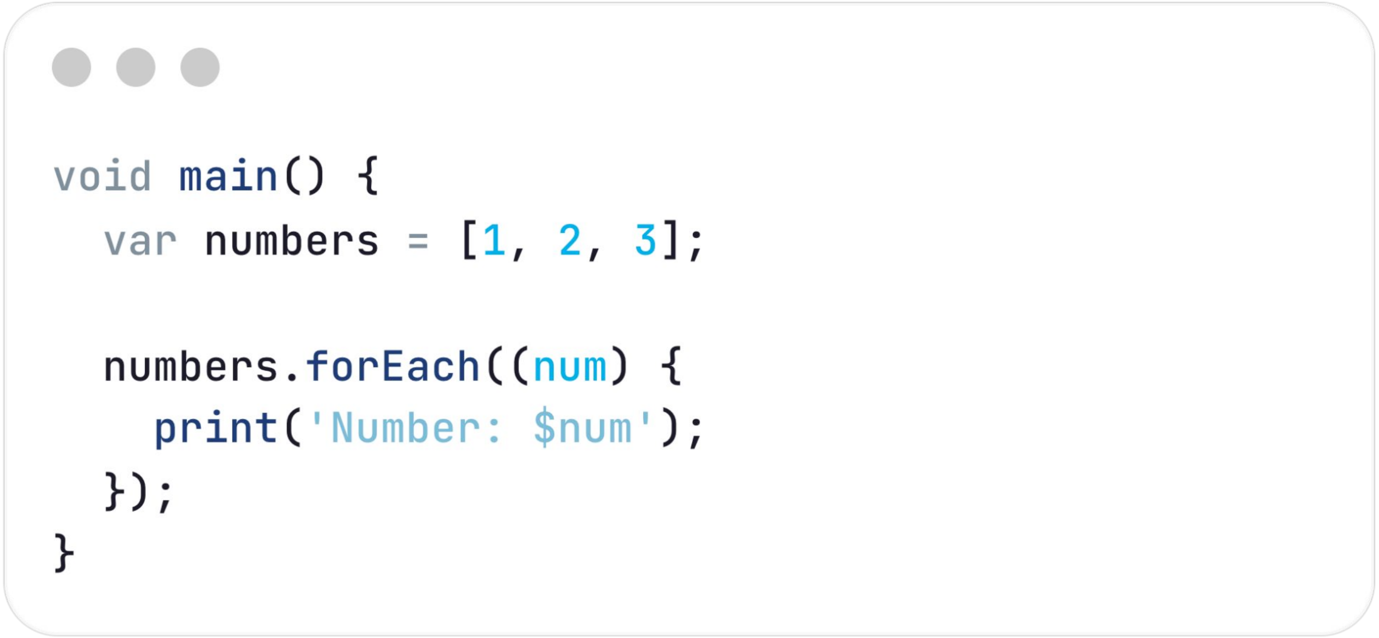


```
void main() {
  var greet = (String name) {
    print('Hello, $name!');
  };

  greet('Momshad');
}
```

Anonymous Functions 3/3

Anonymous functions are commonly used with collection methods like `.forEach()`.



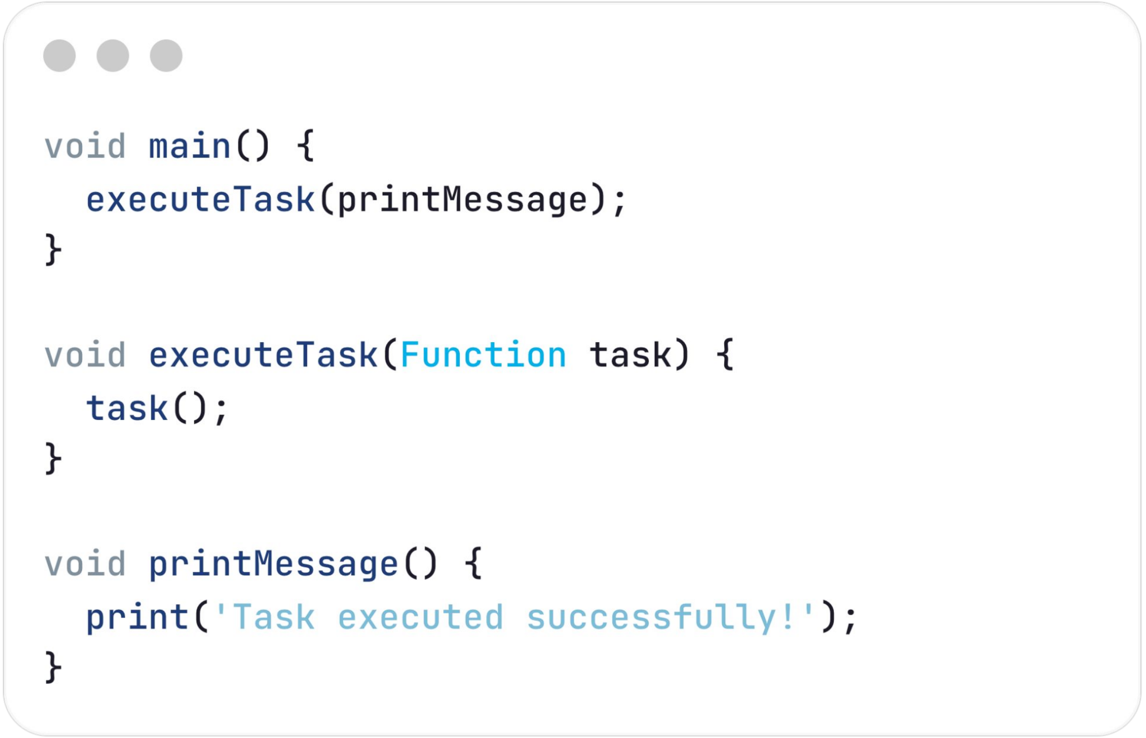
```dart
void main() {
  var numbers = [1, 2, 3];

  numbers.forEach((num) {
    print('Number: $num');
  });
}
```

Higher-Order Functions 1/3

- A Higher-Order Function (HOF) is a function that:
    - Takes another function as a parameter, OR
    - Returns a function as a result.
- Helps with code reuse, callbacks, and functional programming.

Higher-Order Functions 2/3



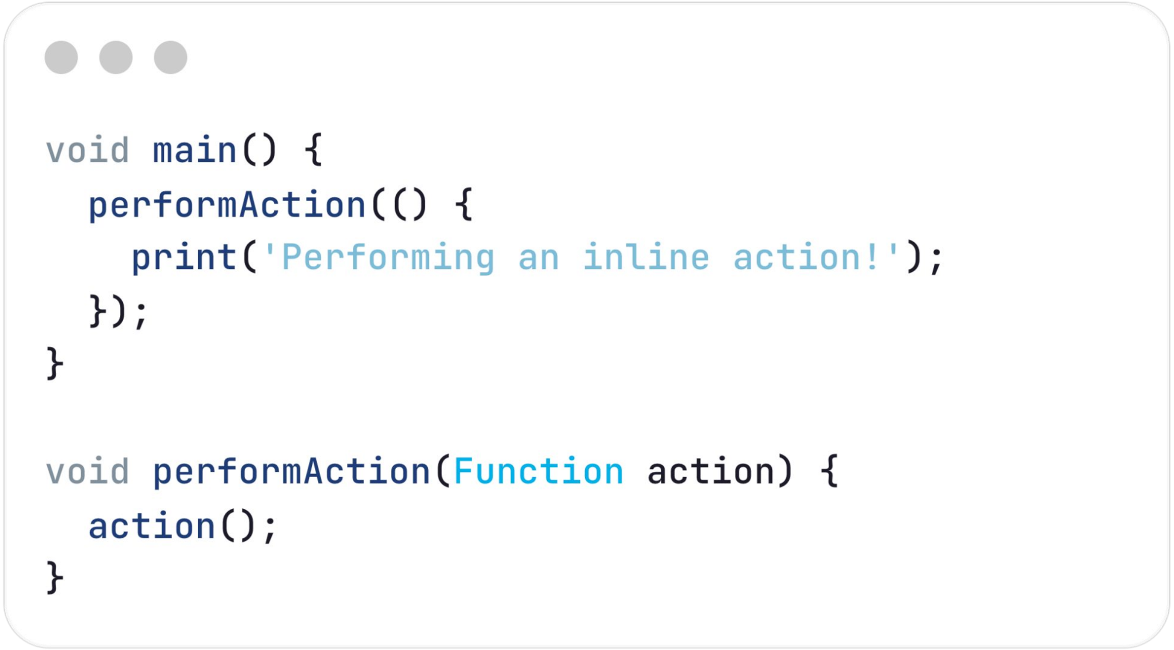
```dart
void main() {
  executeTask(printMessage);
}

void executeTask(Function task) {
  task();
}

void printMessage() {
  print('Task executed successfully!');
}
```

Higher-Order Function with Anonymous Function 3/3

You can also pass anonymous functions directly.

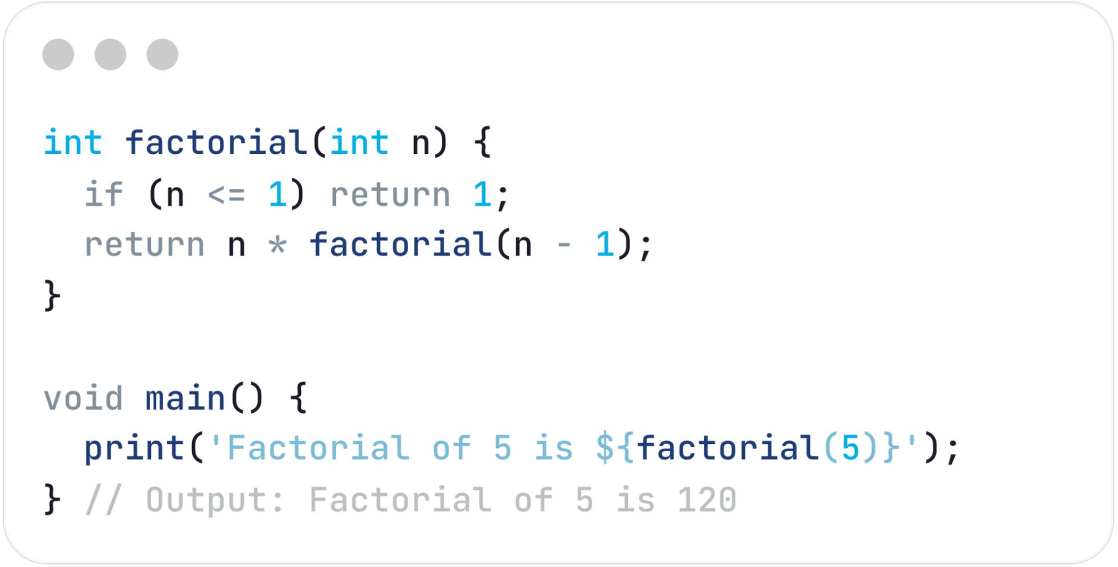


```
void main() {
  performAction(() {
    print('Performing an inline action!');
  });
}

void performAction(Function action) {
  action();
}
```

Recursive Functions 1/2

- A recursive function calls itself.
- Commonly used for problems like: Factorials, Fibonacci series, Tree traversal



```
int factorial(int n) {
  if (n <= 1) return 1;
  return n * factorial(n - 1);
}

void main() {
  print('Factorial of 5 is ${factorial(5)}');
} // Output: Factorial of 5 is 120
```

Recursive Functions 2/2

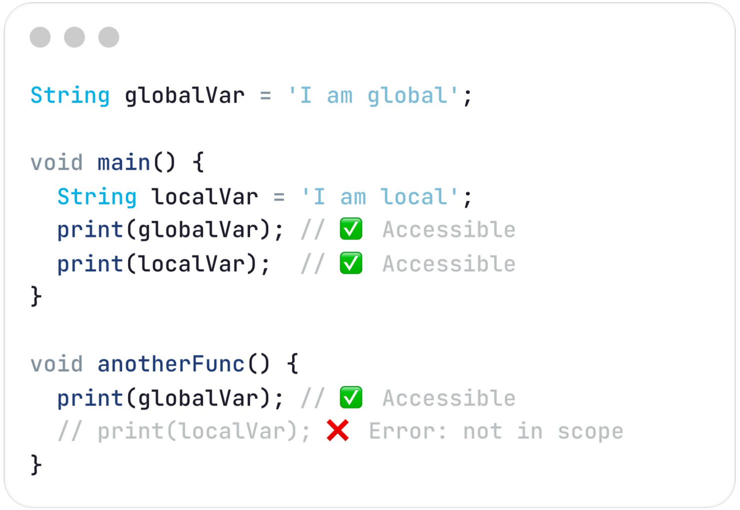
Fibonacci Sequence

```
int fibonacci(int n) {
  if (n <= 1) return n;
  return fibonacci(n - 1) + fibonacci(n - 2);
}

void main() {
  for (var i = 0; i < 6; i++) {
    print(fibonacci(i));
  }
} // Output: 0, 1, 1, 2, 3, 5
```

Scope in Dart

- Scope defines where a variable can be accessed.
- Two main types:
  - Global scope – Declared outside any function/class.
  - Local scope – Declared inside a function or block.



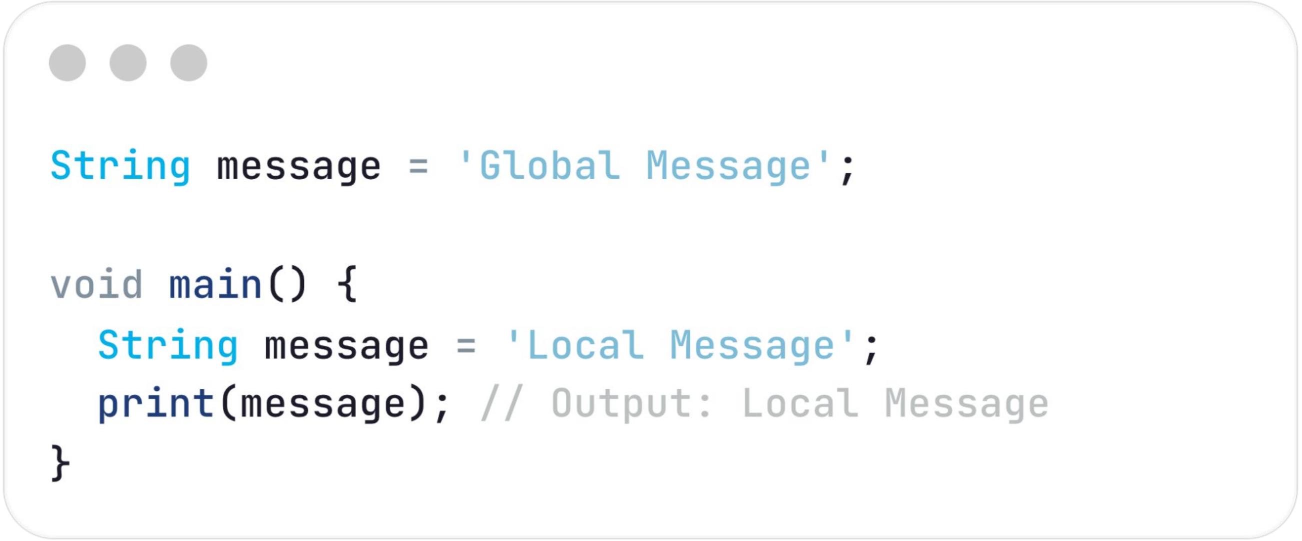
```dart
String globalVar = 'I am global';

void main() {
  String localVar = 'I am local';
  print(globalVar); // ✅ Accessible
  print(localVar);  // ✅ Accessible
}

void anotherFunc() {
  print(globalVar); // ✅ Accessible
  // print(localVar); ❌ Error: not in scope
}
```

Variable Shadowing

- A local variable can hide (shadow) a global variable with the same name.
- The nearest scope wins.



```
String message = 'Global Message';

void main() {
  String message = 'Local Message';
  print(message); // Output: Local Message
}
```