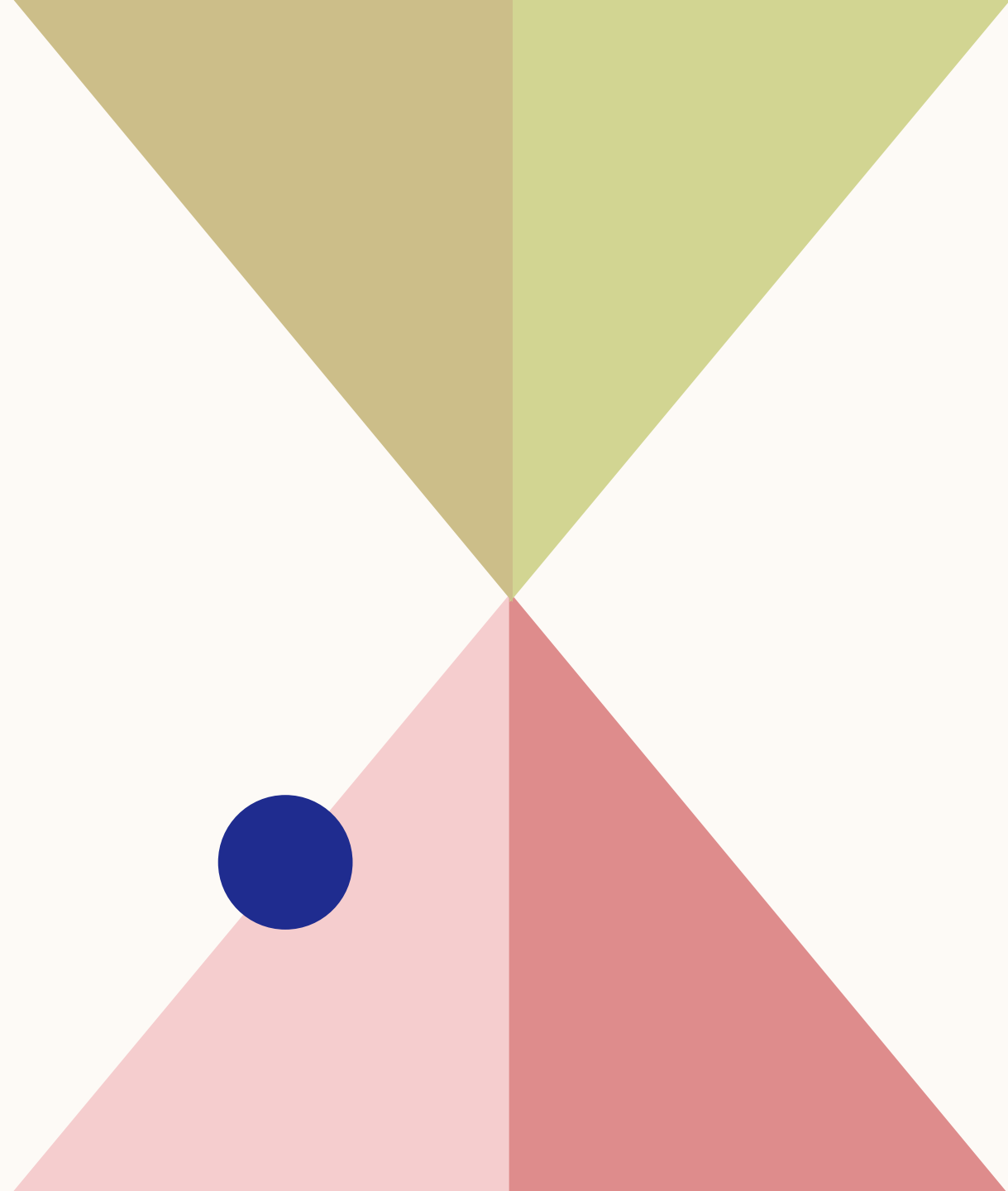
DART FUNCTIONS

SCOPE:

- Defining and Calling Functions
- Function with Parameters
- Returning Values from Functions
- Arrow Functions



FUNCTION

- A function is a block of code that performs a specific task, can be reused, and can be called whenever needed.

WHY USE FUNCTIONS?

- Code reuse.
- Modularity.
- Easier debugging and testing.

DEFINING AND CALLING FUNCTIONS

How to Define a function?

```
void sayHello() {
  print('Hello!');
}
```

Calling a Function:

```
sayHello();  // Output: Hello!
```

*Note: **void:** No value is returned from this function.*
***sayHello():** Function call.*

F RS

- Fu g
th

```dart
import 'dart:io';

// Function to get user input and pass it as parameters
void processUserInput() {
  // Get name input
  stdout.write("Enter your name: ");
  String? name = stdin.readLineSync();
  name = name ?? "Unknown"; // Handle null case

  // Get age input
  stdout.write("Enter your age: ");
  String? ageInput = stdin.readLineSync();
  int age = int.tryParse(ageInput ?? "") ?? 0; // Handle invalid or null input

  // Call the function that uses the inputs as parameters
  greetUser(name, age);
}

// Function that takes name and age as parameters and prints a greeting
void greetUser(String name, int age) {
  print("Hello, $name! You are $age years old.");
}

void main() {
  // Call the processUserInput function to get input and pass it as parameters
  processUserInput();
}
```


```dart
import 'dart:io';

// Function to get user input and pass it as parameters
void processUserInput() {
  // Get name input
  stdout.write("Enter your name: ");
  String? name = stdin.readLineSync();
  name = name ?? "Unknown"; // Handle null case

  // Get age input
  stdout.write("Enter your age: ");
  String? ageInput = stdin.readLineSync();
  int age = int.tryParse(ageInput ?? "") ?? 0; // Handle invalid or null input

  // Call the function that uses the inputs as parameters
  greetUser(name, age);
}

// Function that takes name and age as parameters and prints a greeting
void greetUser(String name, int age) {
  print("Hello, $name! You are $age years old.");
}

void main() {
  // Call the processUserInput function to get input and pass it as parameters
  processUserInput();
}
```

FUNCTION WITH PARAMETERS

Functions can take inputs, called parameters, that are passed during the function call.

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

greet('Alice');  // Output: Hello, Alice!


greet();
```

Note: Parameter: String name is passed when the function is called.

Function Call: greet('Alice') passes the value "Alice" to the function.

```
import 'dart:io';

// Function to get user input and pass it as parameters
void processUserInput() {
  // Get name input
  stdout.write("Enter your name: ");
  String? name = stdin.readLineSync();
  name = name ?? "Unknown"; // Handle null case

  // Get age input
  stdout.write("Enter your age: ");
  String? ageInput = stdin.readLineSync();
  int age = int.tryParse(ageInput ?? "") ?? 0; // Handle invalid or null input

  // Call the function that uses the inputs as parameters
  greetUser(name, age);
}

// Function that takes name and age as parameters and prints a greeting
void greetUser(String name, int age) {
  print("Hello, $name! You are $age years old.");
}

void main() {
  // Call the processUserInput function to get input and pass it as parameters
  processUserInput();
}
```

RETURNING VALUES FROM FUNCTIONS

Return Statement: Functions can return a value using the return keyword.

```
int multiply(int a, int b) {
  return a * b;
}

int result = multiply(4, 5);  // result = 20
```

Key Differences:

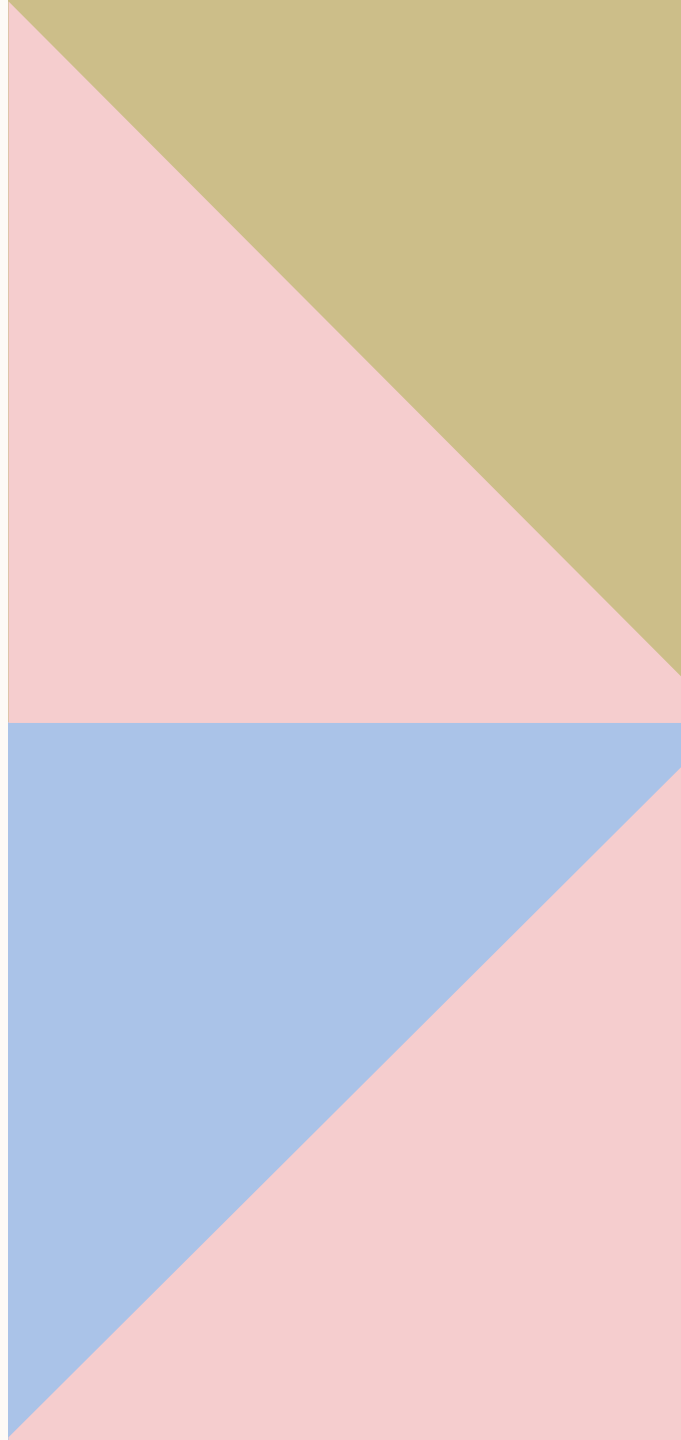
Feature	With Return Value	Without Return Value (Void)
Return Type	Any data type (`int` , `String` , `double` , etc.)	`void` (no return value)
Use in Expressions	Can be assigned to a variable or passed to other functions	Cannot be used in expressions or assigned
Result Handling	Returns a result that the caller can use	Performs an action but returns nothing
Example Use Cases	Functions that compute a value (e.g., calculations)	Functions that print to the console or modify data directly

```
// With return value
double calculateArea(double radius) {
  return 3.14 * radius * radius; // Returns the area of a circle
}

// Without return value
void printArea(double radius) {
  double area = 3.14 * radius * radius;
  print("The area is $area"); // Prints the area, but doesn't return anything
}

void main() {
  // Using the return value
  double area = calculateArea(5.0); // Can be stored and used later
  print("Calculated Area: $area"); // Prints: Calculated Area: 78.5

  // Calling the void function
  printArea(5.0); // Directly prints: The area is 78.5
}
```

In summary:

- **Return value functions** are useful when you need to store or further process the result.

- **Void functions** are useful for side effects like printing to the console, changing the state of an object, or interacting with the user, where you don’t need to return a result.



ARROW FUNCTIONS (FAT ARROW NOTATION)

- Arrow functions, also known as fat arrow functions in Dart, provide a shorthand syntax for defining functions. They are useful when you have simple functions that contain a single expression. Instead of writing the entire function block with curly braces {}, you can use the => notation.

ARROW FUNCTIONS (FAT ARROW NOTATION)

```
int add(int a, int b) => a + b;

void main() {
  print(add(2, 3));  // Output: 5
}
```

Note:

- *add() uses arrow notation to define a function that returns the sum of a and b in one line.*
- *No need for the return keyword or curly braces.*

ARROW FUNCTIONS (FAT ARROW NOTATION)

Example 2: Arrow Function with No Parameters

```
String greet() => "Hello, World!";

void main() {
  print(greet());  // Output: Hello, World!
}
```

ARROW FUNCTION VS NORMAL FUNCTION

Normal Function:

```
int multiply(int a, int b)
{
  return a * b;
}
```

Arrow Function Equivalent:

```
int multiply(int a, int b) => a * b;
```

THANK YOU