

# MOBILE APPLICATION DEVELOPMENT

Dart Introduction 2

Week 3

# Contents

A thick yellow horizontal bar spans the width of the slide, with a vertical yellow bar extending downwards from its right end.

- Synchronous Programming
- Asynchronous Programming
- Packages
- Pub.dev
- Debugging
- Software Testing

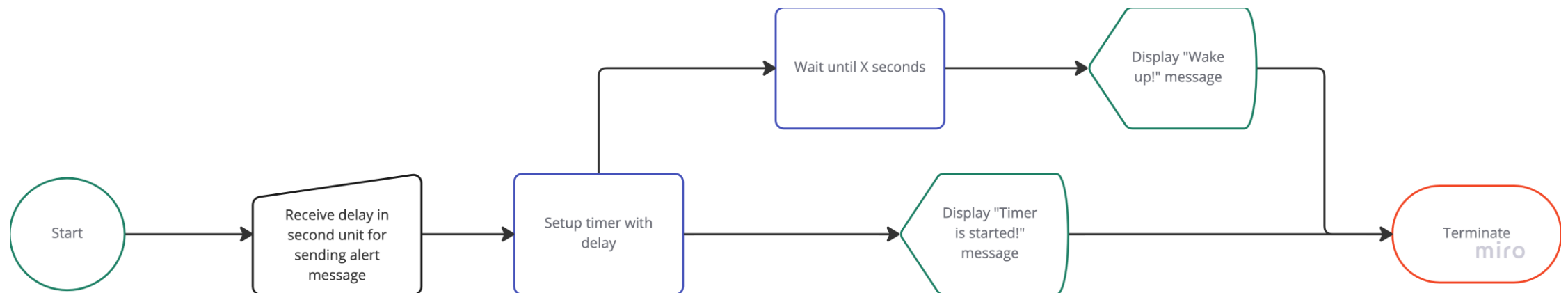
# Synchronous Programming

Basically, we can run a set of code from the top to bottom line and we wait to receive the input from the user, and then continue for the rest which is called blocking I/O or synchronous programming.



# Asynchronous Programming

In the frontend (mobile app or web app), we need to write code to get data from external APIs, libraries, or user input after submitting data through the UI which we need to wait for a second then we get a result. Meanwhile, we can do other things until it calls back to us. We call non-blocking I/O or asynchronous programming.



# Asynchronous Programming - Future

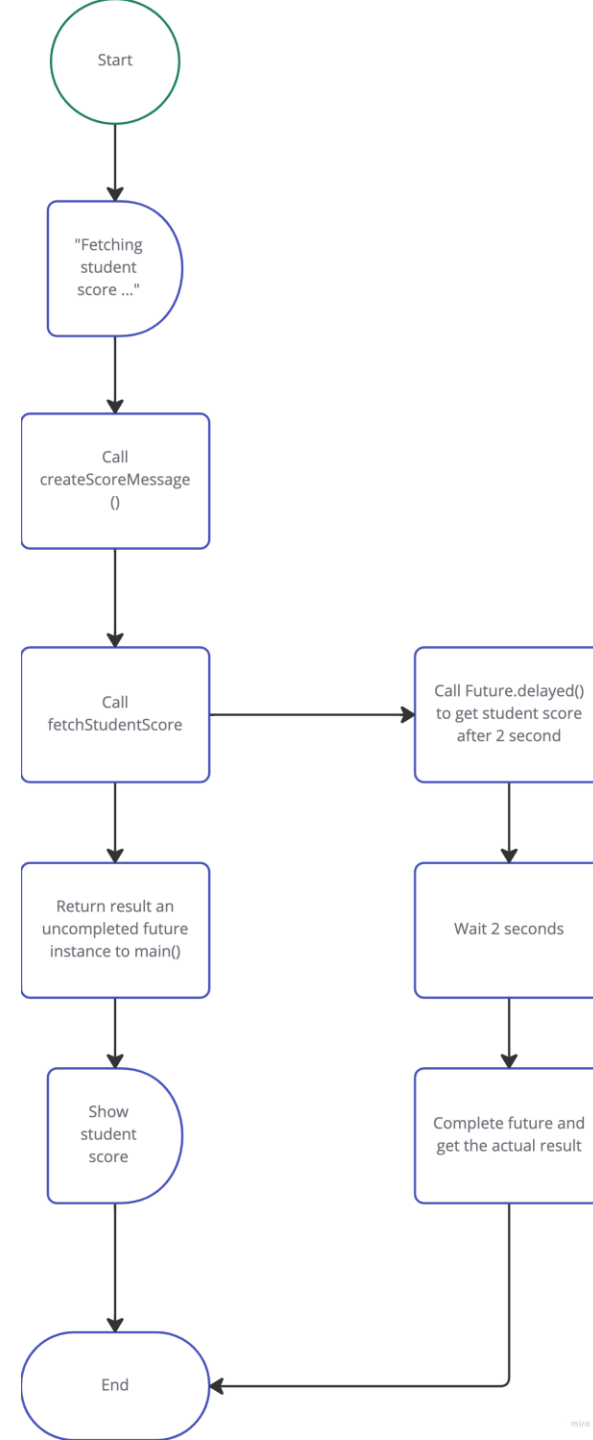
- A future (lower case “f”) is an instance of the Future (capitalized “F”) class. A future represents the result of an asynchronous operation, and can have two states
- Uncompleted : When you call an asynchronous function, it returns an uncompleted future. That future is waiting for the function’s asynchronous operation to finish or to throw an error.
- Completed : If the asynchronous operation succeeds, the future completes with a value. Otherwise, it completes with an error.

# Asynchronous Programming – Uncompleted Future

```
Future<int> fetchStudentScore() =>
// Imagine that this function is
// more complex and slow.
    Future.delayed(
        const Duration(seconds: 2),
        () => 99,
    );

String createScoreMessage() {
    var score = fetchStudentScore();
    return 'Your score is: $score';
}

void main() {
    print('Fetching student score...');
    print(createScoreMessage()); //?
}
```



# Asynchronous Programming – Async, Awaits

The `async` and `await` keywords provide a declarative way to define asynchronous functions and use their results. Remember these two basic guidelines when using `async` and `await`:

- To define an `async` function, add `async` before the function body:
- The `await` keyword works only in `async` functions.

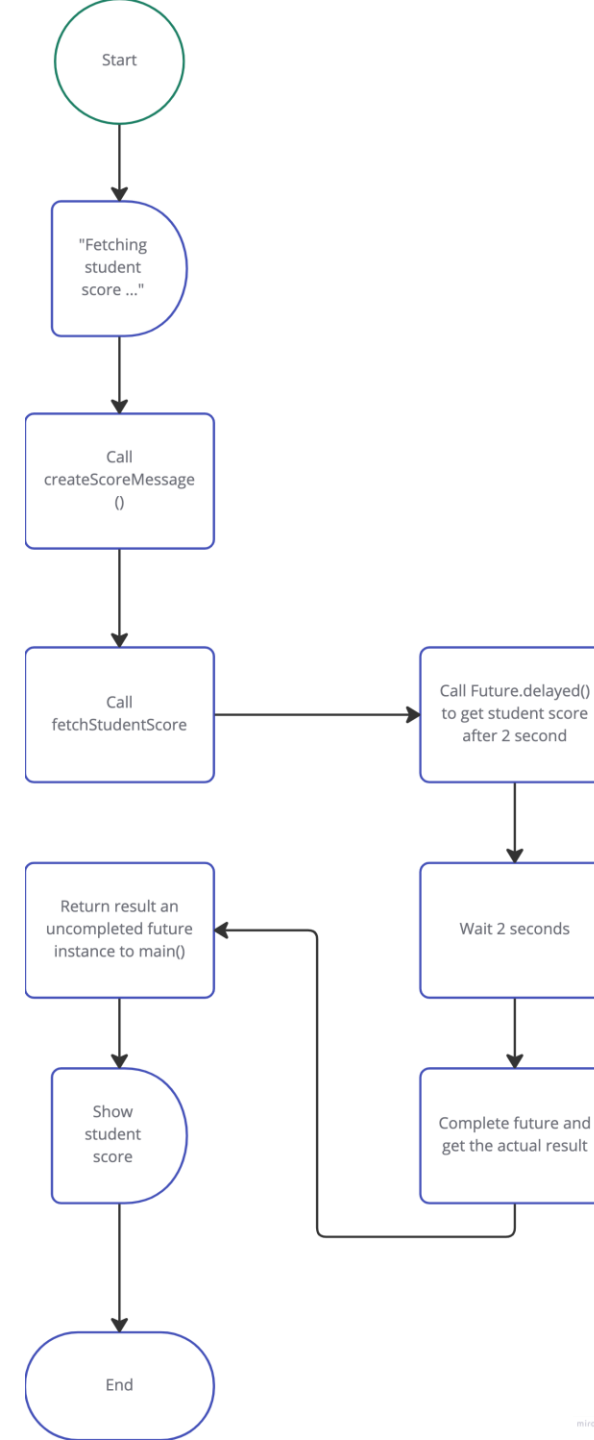
```
void main async () {  
    await <any_function_return_future_object>();  
}
```

# Asynchronous Programming – Completed Future

```
Future<int> fetchStudentScore() =>
// Imagine that this function is
// more complex and slow.
    Future.delayed(
        const Duration(seconds: 2),
        () => 99,
    );

Future<String> createScoreMessage() async{
    var score = await fetchStudentScore();
    return 'Your score is: $score';
}

void main() async {
    print('Fetching student score...');
    print(await createScoreMessage()); //?
}
```





# Asynchronous Programming

- What are the pros?
- What are the cons?

# Packages

- The Dart ecosystem uses *packages* to manage shared software such as libraries and tools. To get Dart packages, you use the [pub package manager](#).

# Packages

To use a package, do the following:

- Create a pubspec (a file named pubspec.yaml that lists package dependencies and includes other metadata, such as a version number).
- You can simply create the pubspec by [dart create <project-name>](#)
- If your Dart code depends on a library or other source code in the package, import the library.

# Pub.dev

**The Pub** is the package manager for the Dart programming language, containing reusable libraries & packages for Flutter and general Dart programs.

- Use [dart pub get](#) to retrieve your package's dependencies.
- Use [dart pub add <library-name>](#) to install a new library from [the pub.dev](#) which is the official package repository.

# Pub.dev – Library Selecting Best Practice

- Have a clear setup and API document.
- Have a high number of contributors, latest publish, likes, flags.
- How fast the developer replies to the user.
- Their source code is understandable to maintain. This is for the worst scenario that the dev doesn't maintain anymore in the future.

# Debugging

- Basically, we can use `print()` function to debug and check the current value.
- By the way, it is very difficult if the code is quite complex. We can use Dart/Flutter plugin in VSCode to debug and step down to a single line of code to see what the root cause is.

# Debugging – VSCode Debugger

- There are 7 actions that you can do in debug mode
  - Continue:
    - Find another break point then stop.
  - Step Over:
    - Go back to the executed line previously.
  - Step Into:
    - Continue to run the next line from the current breakpoint.
  - Hot Refresh:
    - Load the recent code that we changed without restarting the app.
  - Restart:
    - Re-compile and reset the program state then restart from the first line of the program
  - Stop:
    - Stop program.
  - Active CPU Profiler :
    - Activate CPU profiler to list the CPU usage for each executed function.

# Debugging – VSCode Debugger

```
Future<void> showCurrentTime() {  
    return Future.delayed(const Duration(seconds: 1),  
        () => print("Current Time Test ${DateTime.now()}"));  
}
```

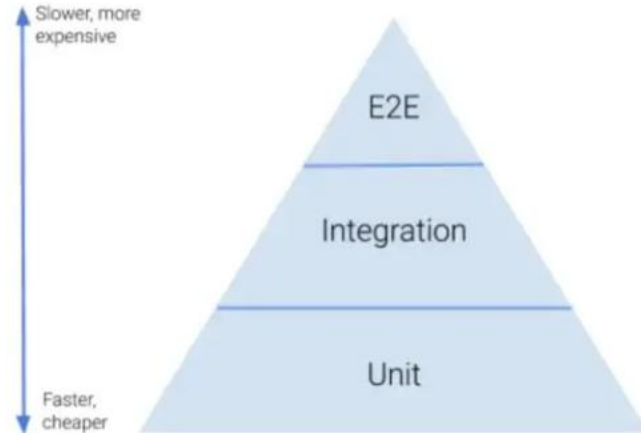
```
void main() async {  
    while (true) {  
        await showCurrentTime();  
    }  
}
```





# Software Testing

- Software testing is the process of using specialized tools and software to execute test cases and check for bugs and errors in a system.
- There are 3 levels of test which are unit test, integration test and end to end test.



# Automation Testing – Unit Test

- Unit tests are used to test individual units of code, such as functions or classes.
- In Dart, unit tests are written using the test function and are run using the test command.
- Unit tests can be used to test the logic and behavior of the code, as well as its output.

# Automation Testing – Example Unit Test

- Write `calculateRectangleArea()` in `rectangle.dart` based on rectangle area formula:

Rectangle Area = Width \* Height

// Source Code

```
num calculateRectangleArea(num width, num height) {  
    return width * height;  
}
```

# Automation Testing – Example Unit Test

- Write the unit test file to test calculateRectangleArea()
- Create a new file in the test folder of your project.
- Use test() to set up the test scenario and use the check calculateRectangleArea() and expected result

```
import 'package:dart_introduction/math/rectangle.dart';
import 'package:test/test.dart';

void main() {
    test('calculate rectangle area', () {
        expect(calculateRectangleArea(10, 3), 30);
    });
}
```

# Automation Testing – Example Unit Test

- To run the test, we can press “Run” above of test() function in VSCode or run “dart test” command in terminal.
- If you want to run specified test only, you can add --name <keyword> to run:

```
dart test --name calculate
```

# Preparing your self for Flutter programming

- In the next session, we're going to teach you about Flutter programming which requires basic knowledge of object-oriented programming. We encourage you to learn more about it from this document:

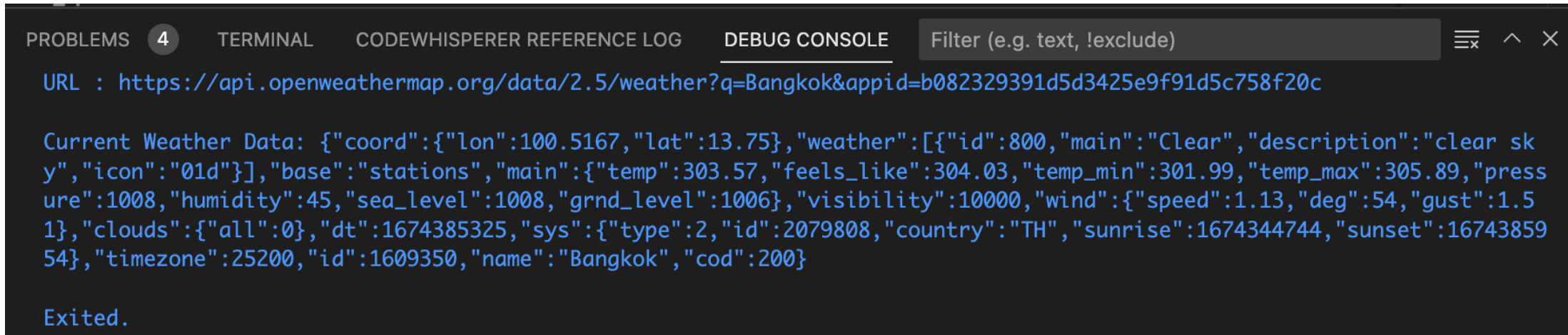
<https://dart.dev/guides/language/language-tour#classes>

# Homework 2 : Weather API Calling

- Create a new Dart project with dart command
- Install [dio](#) library in your project
- Create a function called **fetchData(String url)** that takes in a string value representing a URL and returns a `Future<String>` representing the data fetched from that URL using the [dio](#) package.
- In the function, you need to create an instance of [Dio class](#) then you can call [dio.get\(\)](#) to send a request to the API server.
- When the server returned the response data, you can read it from the **response.data** which is returned by [dio.get\(\)](#).
- Use the `async/await` syntax to handle the asynchronous fetching of data
- Create the **main()** that calls the **fetchData()** function with current weather API , then prints the fetched data to the console.

# Homework 2 : Weather API Calling

- API URL:  
`https://api.openweathermap.org/data/2.5/weather?q=Bangkok&appid=b082329391d5d3425e9f91d5c758f20c`
- Expected Output in console



```
PROBLEMS 4 TERMINAL CODEWHISPERER REFERENCE LOG DEBUG CONSOLE Filter (e.g. text, !exclude)
URL : https://api.openweathermap.org/data/2.5/weather?q=Bangkok&appid=b082329391d5d3425e9f91d5c758f20c

Current Weather Data: {"coord":{"lon":100.5167,"lat":13.75},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01d"}],"base":"stations","main":{"temp":303.57,"feels_like":304.03,"temp_min":301.99,"temp_max":305.89,"pressure":1008,"humidity":45,"sea_level":1008,"grnd_level":1006},"visibility":10000,"wind":{"speed":1.13,"deg":54,"gust":1.51},"clouds":{"all":0},"dt":1674385325,"sys":{"type":2,"id":2079808,"country":"TH","sunrise":1674344744,"sunset":1674385954},"timezone":25200,"id":1609350,"name":"Bangkok","cod":200}

Exited.
```



# Homework 2 : Weather API Calling

- After completing the project, remove .dart\_tool folder then zip the project as homework2.zip then submit through the it onlearn platform.
- The due date is Feb 1<sup>st</sup>, 2023.