

Introduction to mobile development using Flutter

Mobile Hive - Palo IT (Thailand)

Krisada Vivek

Meet your instructor

Objective overview

In this course you will learn

- Understand how to use flutter framework
- Understand basic of Dart lang*
- Create your applications with multiple functionalities
- Know how to design mobile architecture
- Put your app into Production
- Know where to find changes of framework

Table of Contents

These are the topics that you are going to train throughout this course

- Chapter 1: Installation and Setup for Flutter Development
- Chapter 2: Creating 1st Flutter app
- ~~• Chapter 3: Basic Dart Lang*~~
- Chapter 4: Why Flutter?
- Chapter 5: Introduction to Widgets
- ~~• Chapter 6: Advanced Layout Building~~
- Chapter 7: Design System
- Chapter 8: Building form application in Flutter
- Chapter 9: Navigation
- Chapter 10: Introduction to State Management
- Chapter 11: E2E testing with Cucumber
- Chapter 12: Software Architecture
- Chapter 13: Working with 3rd party libs
- ~~• Chapter 14: Platform Specific Code~~
- Chapter 15: Running and releasing on physical devices

TLDR;

Agreements

- Understand, not remember
 - Cover multiple aspect
 - Please take notes – perspective, open discussion/opinion
- Follow Along - I'll. Go slow, 🐢 I'm promise 😊
- Practices is the key things
- Enjoy coding

Chapter 1: Installation and Setup for Flutter Development

Flutter SDK installation

Overview Process

- Go to Official Website -> <https://flutter.dev>
- Getting Start tab
- Proceed the instruction steps
 - Select your operating system (macOS)
 - Download .zip file
 - Extract the .zip
 - Place in desire location
 - Update system path ~> export PATH="\$PATH:`pwd`/flutter/bin" to ~/.bash_profile or ~/.bash_rc
 - Restart your Terminal

Flutter SDK installation

Overview Process

- In you Terminal type "flutter" to identify is your setup success or not
 - If "Success" it's should prompt out lie this ->

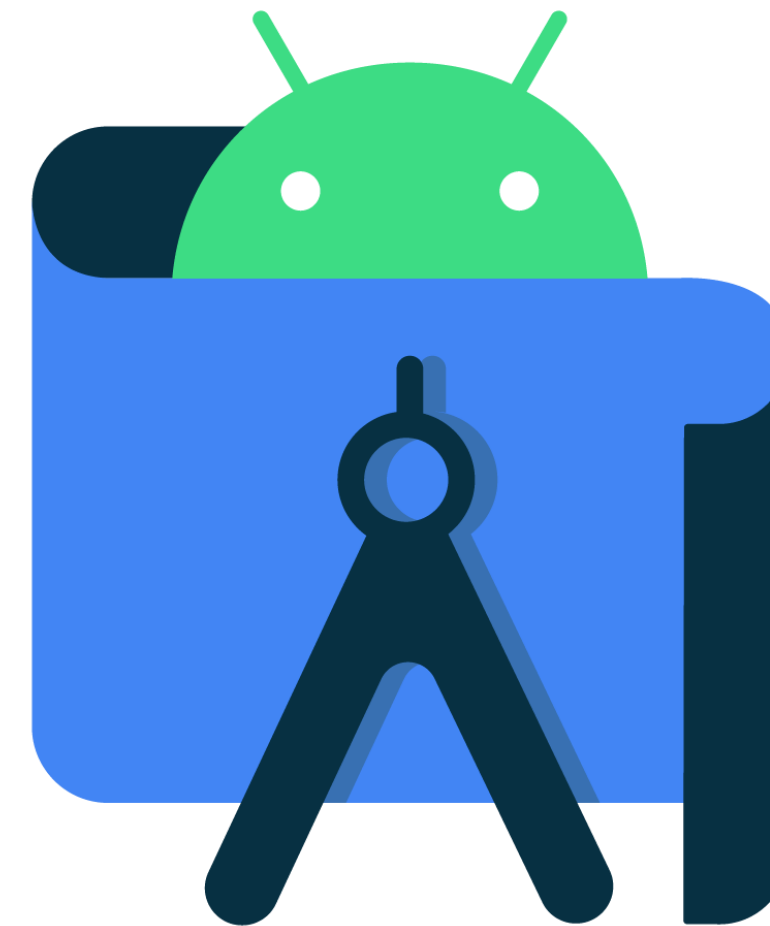
```
[O → flutter  
Manage your Flutter app development.  
  
Common commands:  
  
flutter create <output directory>  
  Create a new Flutter project in the specified directory.  
  
flutter run [options]  
  Run your Flutter application on an attached device or in an emulator.
```

- If "No" please check your system variable path

macOS setup

Prerequisites

- For developing a mobile application you need 2 IDEs



macOS setup

Prerequisites

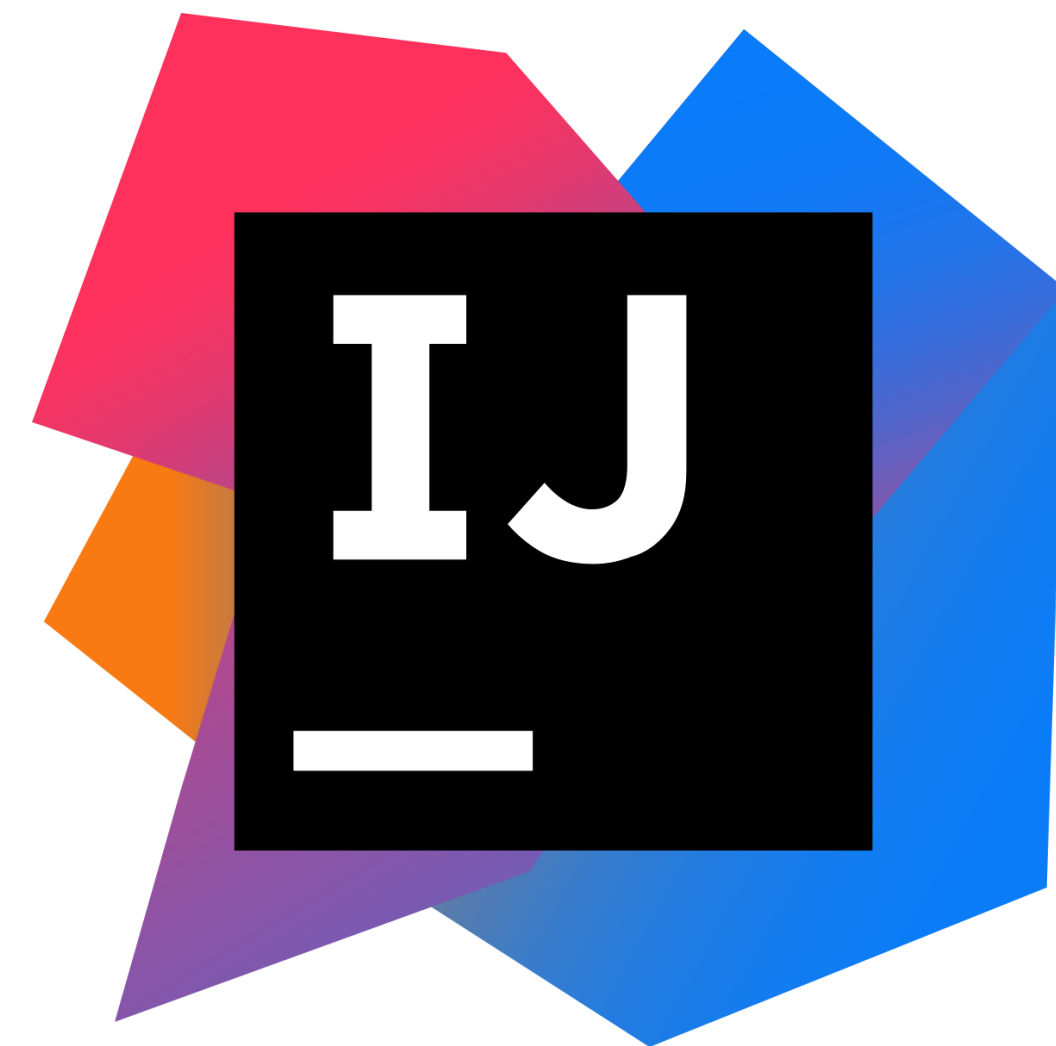
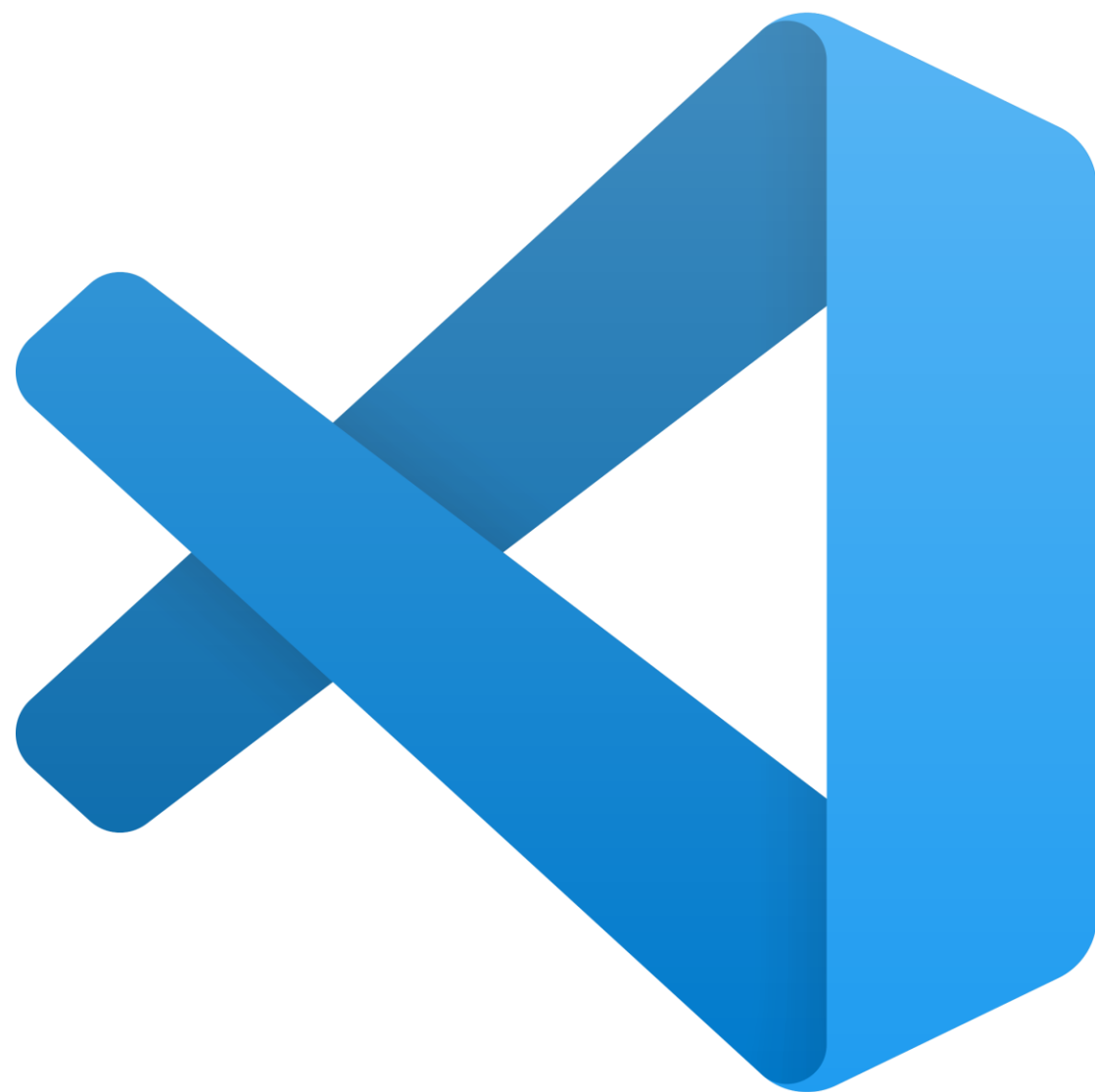
- For additional tools you will also need to install Homebrew => [The Missing Package Manager for macOS \(or Linux\) — Homebrew](#)



macOS setup

Select your IDEs

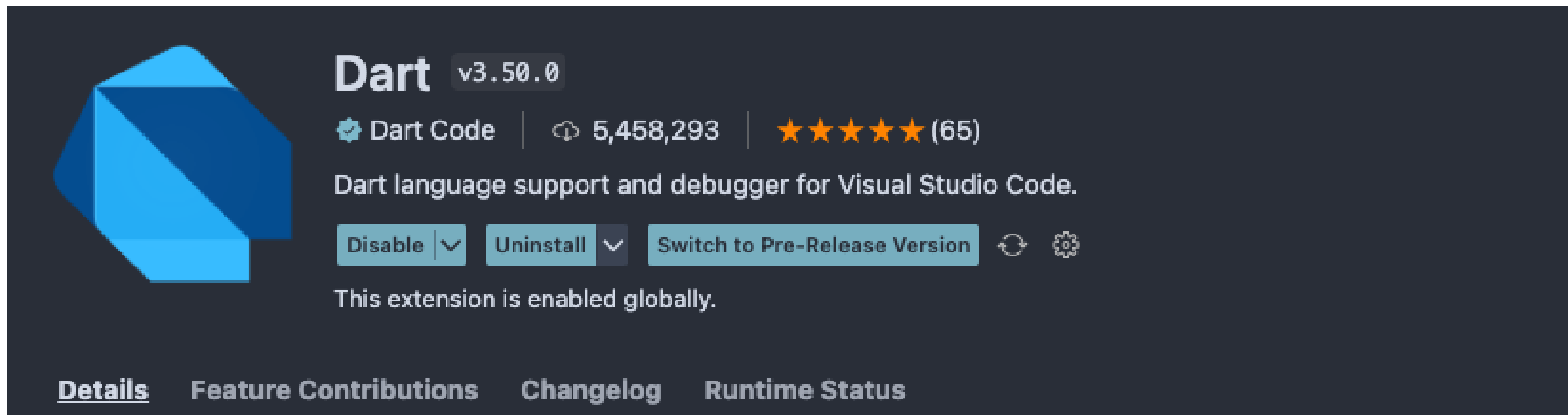
- VScode or IntelliJ



- But in this training we will use VScode as a main ide

macOS setup

Setup an extensions



Dart v3.50.0

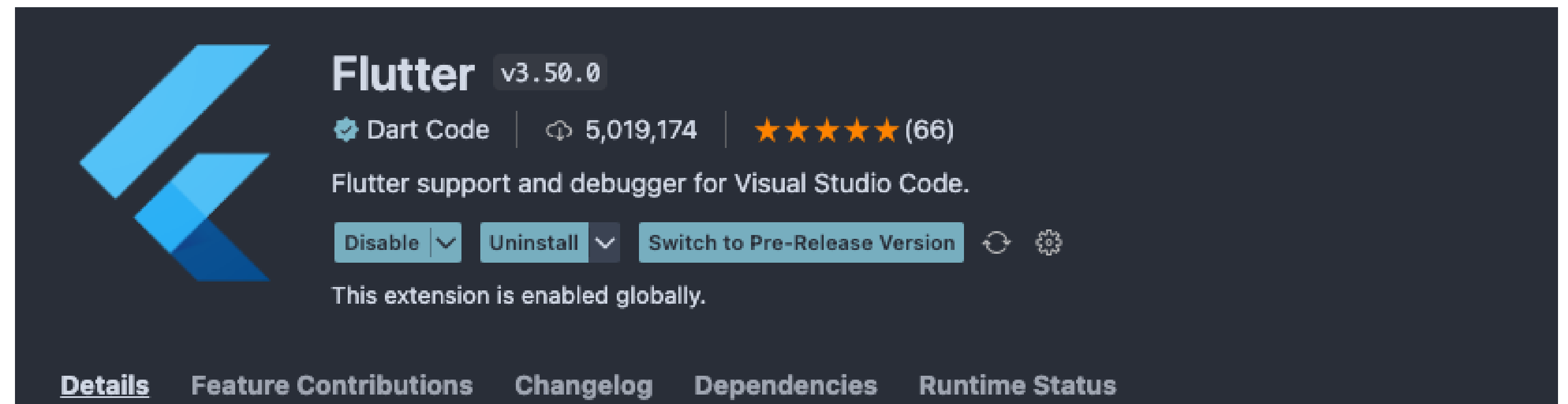
Dart Code | 5,458,293 | ★★★★★ (65)

Dart language support and debugger for Visual Studio Code.

Disable | Uninstall | Switch to Pre-Release Version

This extension is enabled globally.

[Details](#) | [Feature Contributions](#) | [Changelog](#) | [Runtime Status](#)



Flutter v3.50.0

Dart Code | 5,019,174 | ★★★★★ (66)

Flutter support and debugger for Visual Studio Code.

Disable | Uninstall | Switch to Pre-Release Version

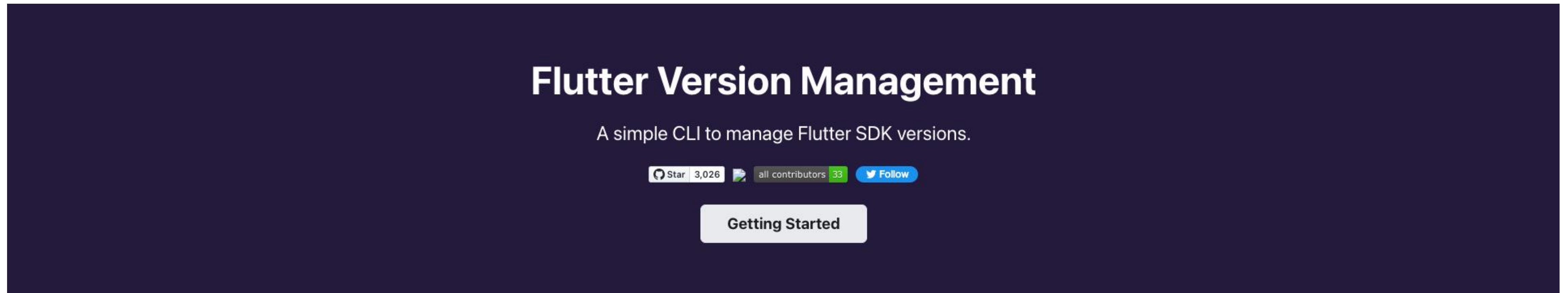
This extension is enabled globally.

[Details](#) | [Feature Contributions](#) | [Changelog](#) | [Dependencies](#) | [Runtime Status](#)

Can I install more than 1 flutter version?



Introduction to FVMs/DVMs

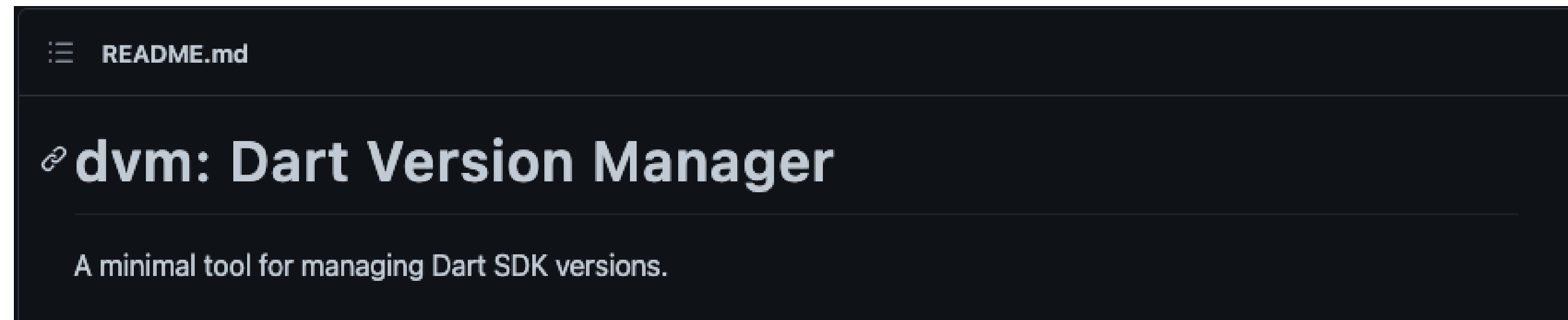


[Flutter Learning EP6: Managing Multiple version of Flutter SDK using FVM - YouTube](#)

Introduction to FVMs/DVMs

DVMs

- [cbracken/dvm: Dart Version Manager \(github.com\)](https://github.com/cbracken/dvm)



Chapter 2: Creating 1st flutter app

Introductions

In this chapter we are going to

- Create 1st flutter application via CLI
- Know how to use the basic commands
- Fundamental Structure
- Run 1st Application on an Emulator/Simulator
- Learn other essential dev tools – DevTools/Debugger

Flutter Commands

These are basic command you are going to use in this course

- flutter create <application_name> => crate flutter project
- flutter doctor (-v) => command that will give you a prompt about version and additional dependencies that required to develop a flutter application
- flutter pub get => get and install dependencies in your project
- flutter run => run your 1st flutter app
- flutter upgrade

Create 1st flutter application using CLI

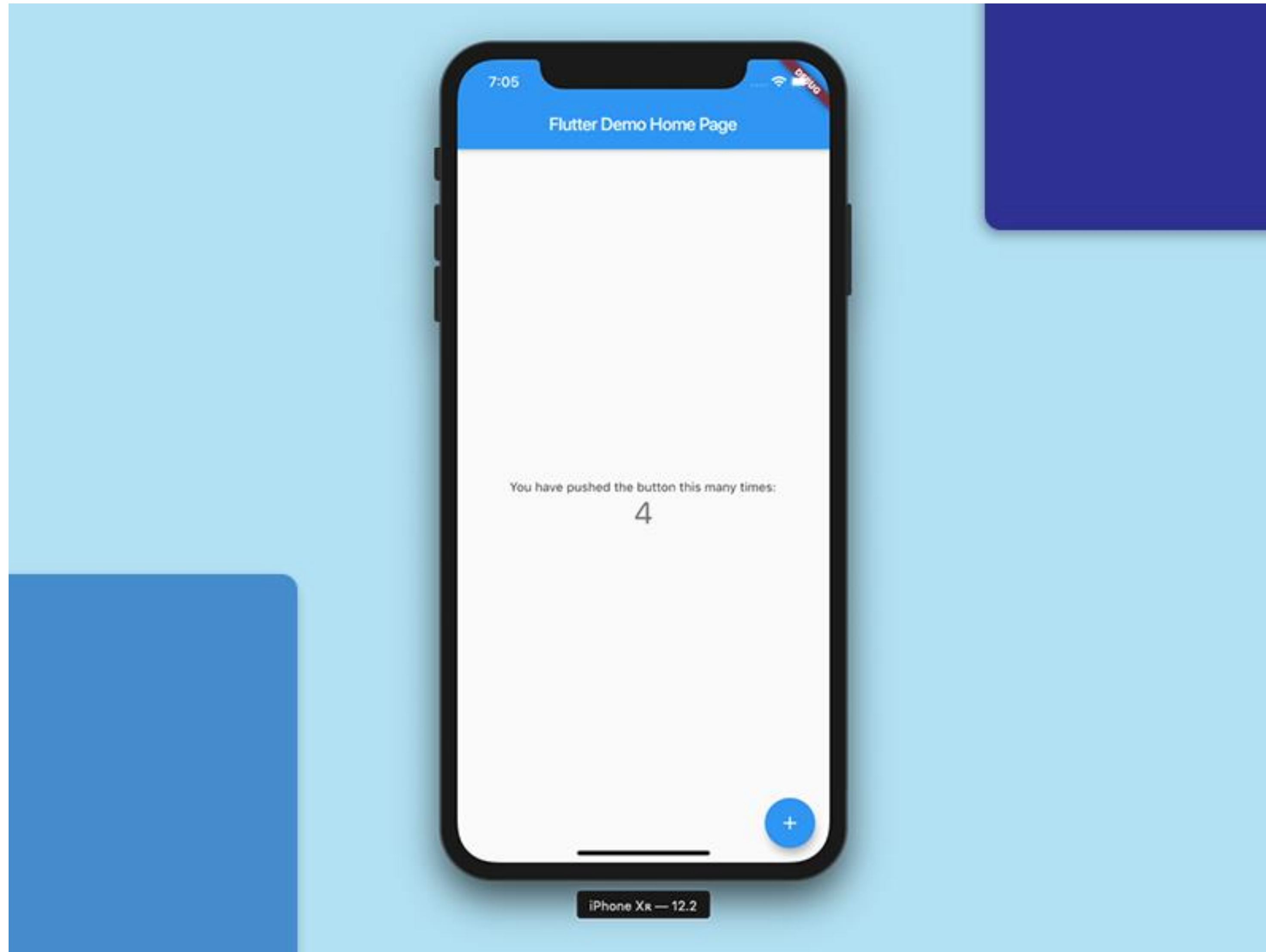
via command line

- Type "flutter create <application_name>" and lets magic happen
- Open IDE to see the code
- Take a quick look at the code

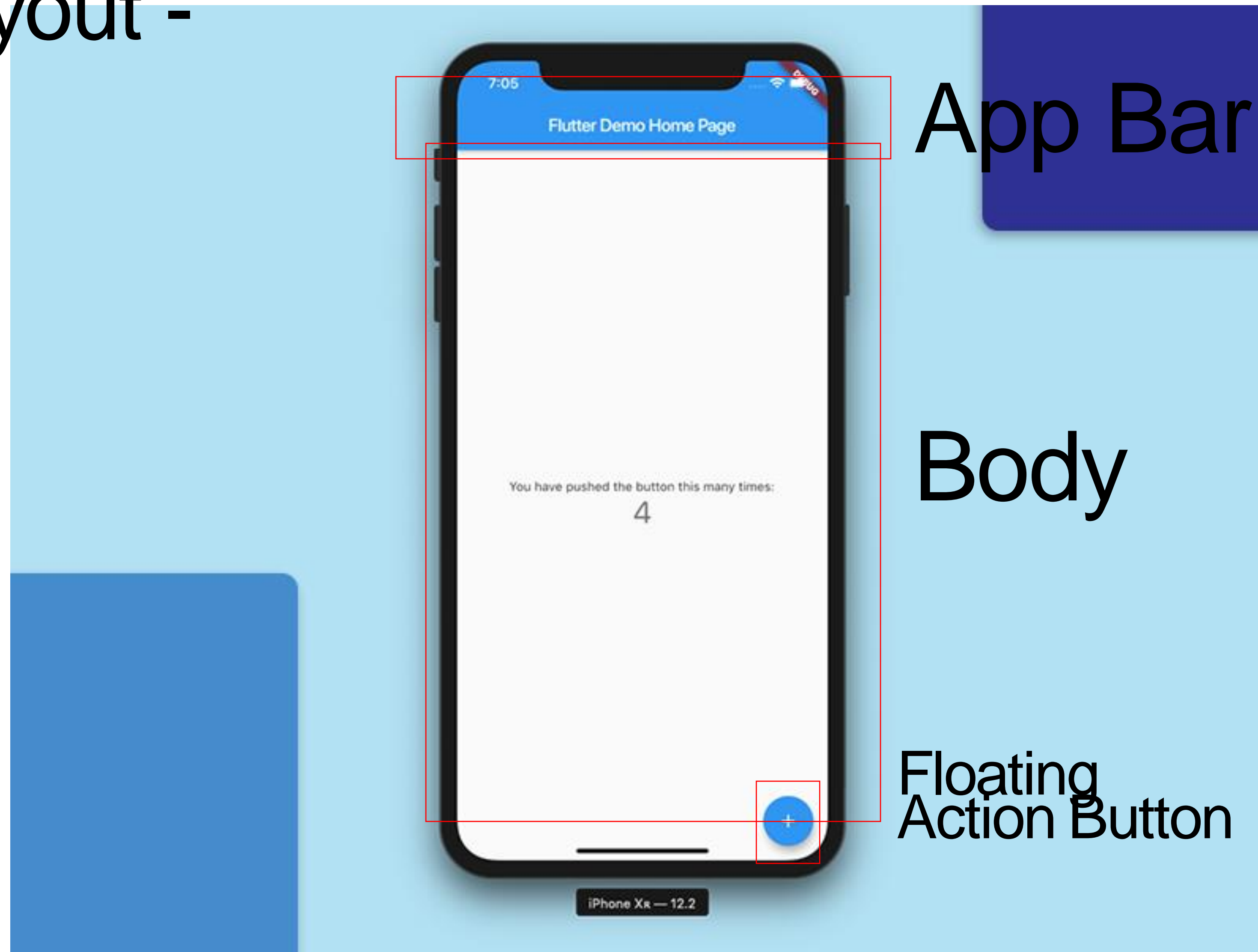
Run your 1st application

1st Checkpoint

- Start iOS Simulator ~> open –a Simulator
- Start Android Emulator ~> open Android Studio
- Type "flutter run" to start your application for the 1st time



Basic Layout - Scaffold



Run your 1st application

1st Checkpoint

- Look at your terminal, you will see additional information that essential to your development

```
o → flutter run
Launching lib/main.dart on iPhone 14 Pro Max in debug
mode...
Running Xcode build...
  └─Compiling, linking and signing...
Xcode build done.
20.4s
Syncing files to device iPhone 14 Pro Max...

Flutter run key commands.
r Hot reload. 🔥🔥🔥
R Hot restart.
h List all available interactive commands.
d Detach (terminate "flutter run" but leave application
running).
c Clear the screen
q Quit (terminate the application on the device).

👉 Running with sound null safety 👉

An Observatory debugger and profiler on iPhone 14 Pro Max
```


An essential Tools

DevTools

- Built-in tools that helps developer identify certain part of an application such as UIs, performance, memory consumptions

Debugger

- Internal VScode debugger

🔄 Refresh

🗑 Clear

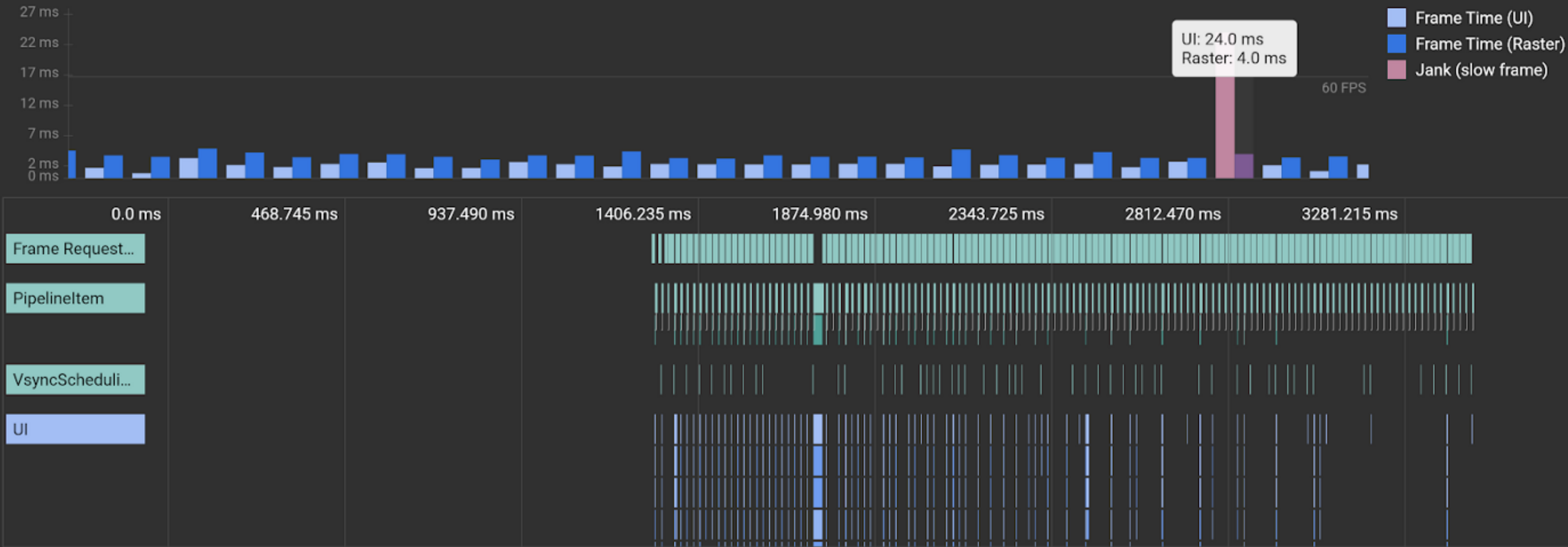
Profile granularity: medium ▾

📊 Performance Overlay

📋 Track Widget Builds

📄 Export

⋮



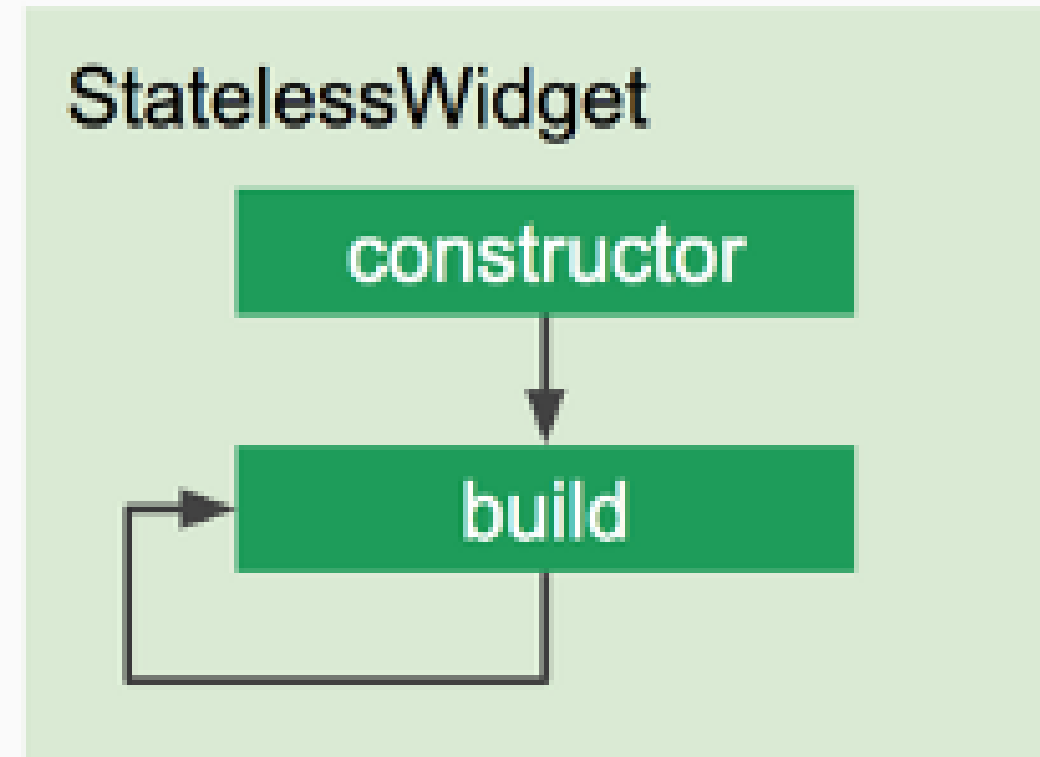
[No event selected]

Select an event from the Timeline to view details

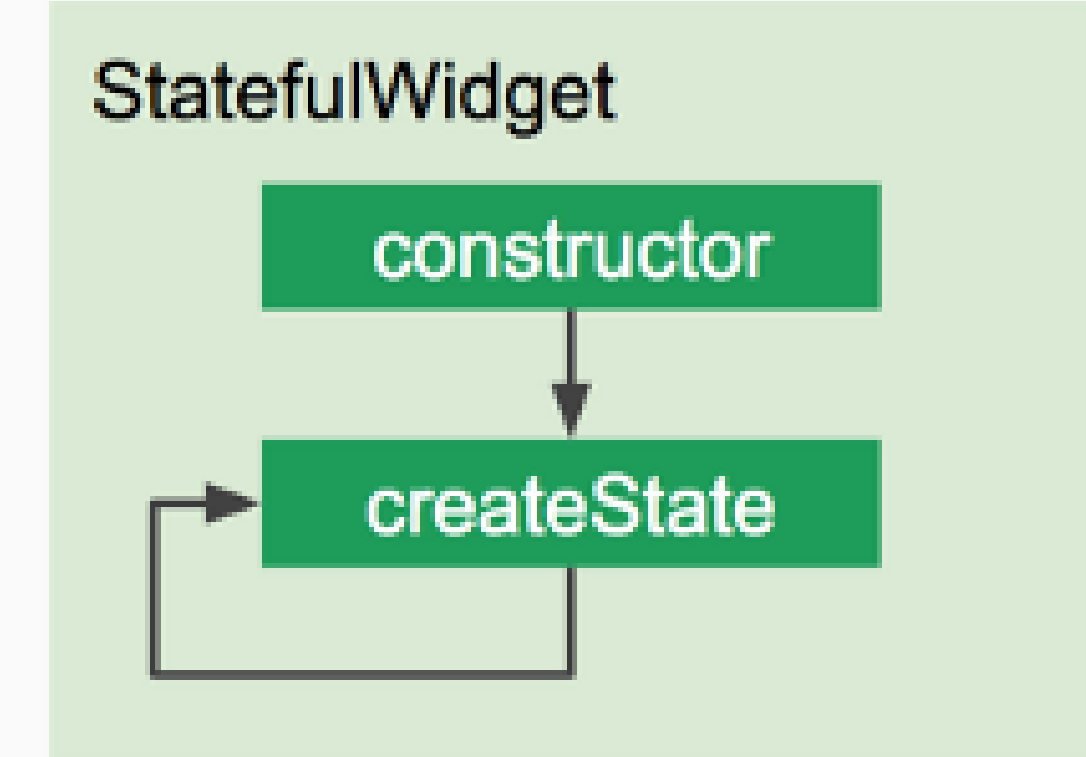
Fundamental structure

Stateless vs Stateful

- In flutter we have 2 types of core widget
 - Stateless Widget
 - Stateful Widget



A single StatelessWidget can build in many different BuildContexts



A StatefulWidget creates a new State object for each BuildContext

Stateful Widget

- Plenty of "state" -> mutable variable's value
- Change within its widget (local state)
- Life Cycle
- Can call setState((){})
- // add example of stateful state less (5 quick question)

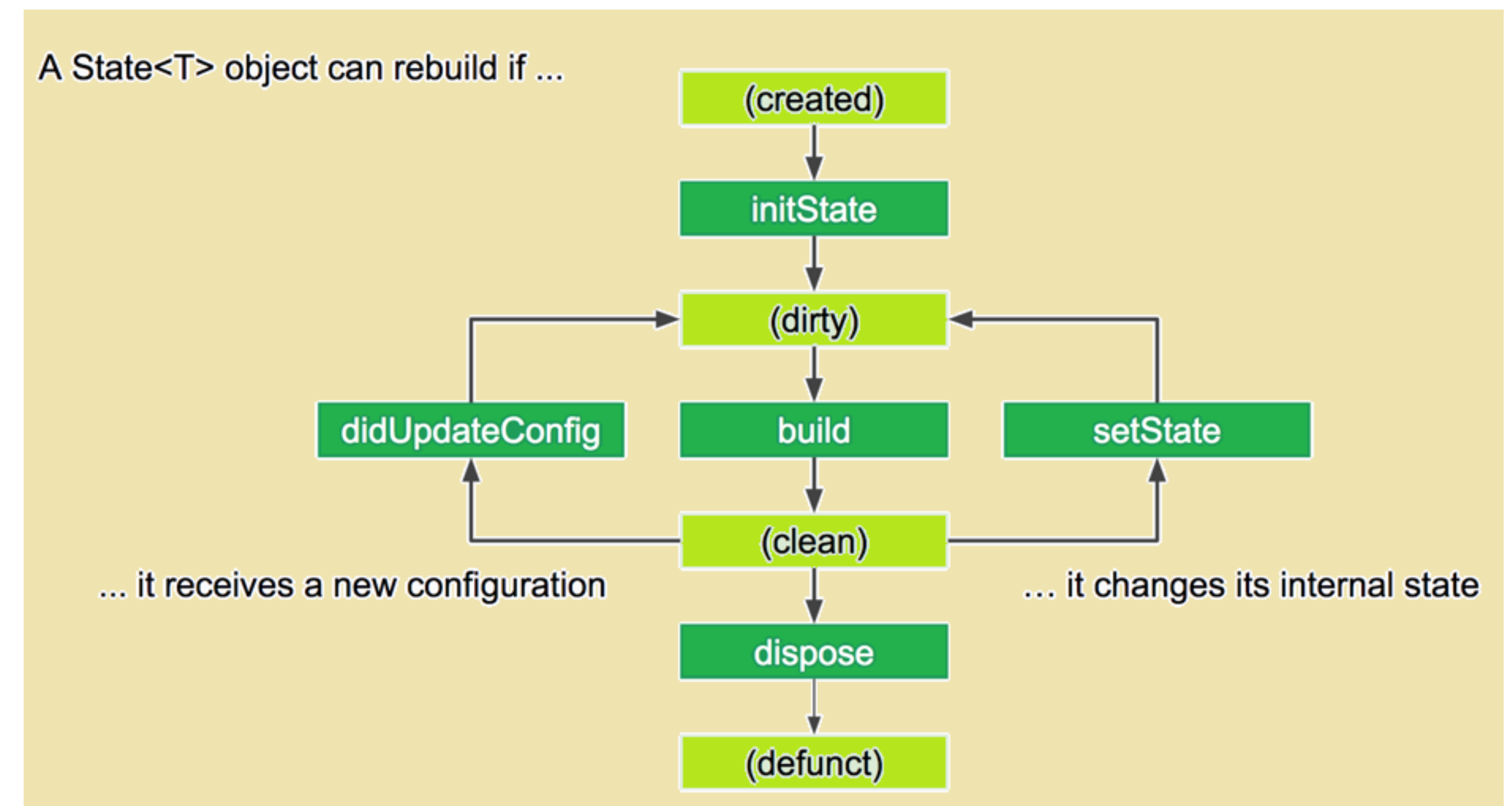


```
class MyApp extends StatefulWidget {  
  const MyApp({Key? key}) : super(key: key);  
  
  @override  
  State<MyApp> createState() => _MyAppState();  
}  
  
class _MyAppState extends State<MyApp> {  
  @override  
  Widget build(BuildContext context) {  
    return Container();  
  }  
}
```

Widget Lifecycle

Born – Live – Die – Repeat

- Terminologies in modern frontend framework,
- Manage things need to display – not display (UI element)
- Do some task before components appeared
- `initState()`, `dispose()`, `build()`, `didChangeDependencies()`



Mini Checkpoint 1

```
class MyHomePage extends <??> {  
  const MyHomePage({super.key, required this.title});  
  
  final String title;  
  
  @override  
  State<MyHomePage> createState() => _MyHomePageState();  
}  
  
class _MyHomePageState extends State<MyHomePage> {  
  int _counter = 0;  
  
  void _incrementCounter() {  
    setState(() {  
      _counter++;  
    });  
  }  
  
  @override  
  Widget build(BuildContext context) {  
    return Scaffold(  
      appBar:
```



```
class MyApp extends <??> {  
  const MyApp({super.key});  
  
  @override  
  Widget build(BuildContext context) {  
    return MaterialApp(  
      title: 'Flutter Demo',  
      theme: ThemeData(  
        primarySwatch: Colors.blue,  
      ),  
      home: const MyHomePage(title: 'Flutter Demo Home Page'),  
    );  
  }  
}
```



```
class SamplePage extends <??> {  
    const SamplePage({super.key});  
  
    @override  
    State<SamplePage> createState() => _SamplePageState();  
}  
  
class _SamplePageState extends State<SamplePage> {  
    @override  
    void initState() {  
        super.initState();  
  
        /// ...some code  
    }  
  
    @override  
    void dispose() {  
        /// ...some code  
        ///  
        super.dispose();  
    }  
}
```




```
class MiniCheckpoint15 extends <??>{  
  const MiniCheckpoint({Key? key}) : super(key: key);  
  
  @override  
  Widget build(BuildContext context) {  
    return ScaffoldLayout(  
      body: Center(  
        child: Image.asset("path_to_image"),  
      ),  
    );  
  }  
}
```

Stateless Widget

- "not" retain any variable value
- Give constructor, Then build
- State may control by it's parent

```
class MyApp extends StatelessWidget {
  const MyApp({super.key});

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // Try running your application with "flutter run". You'll see the
        // application has a blue toolbar. Then, without quitting the app, try
        // changing the primarySwatch below to Colors.green and then invoke
        // "hot reload" (press "r" in the console where you ran "flutter run",
        // or simply save your changes to "hot reload" in a Flutter IDE).
        // Notice that the counter didn't reset back to zero; the application
        // is not restarted.
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
```

Recap

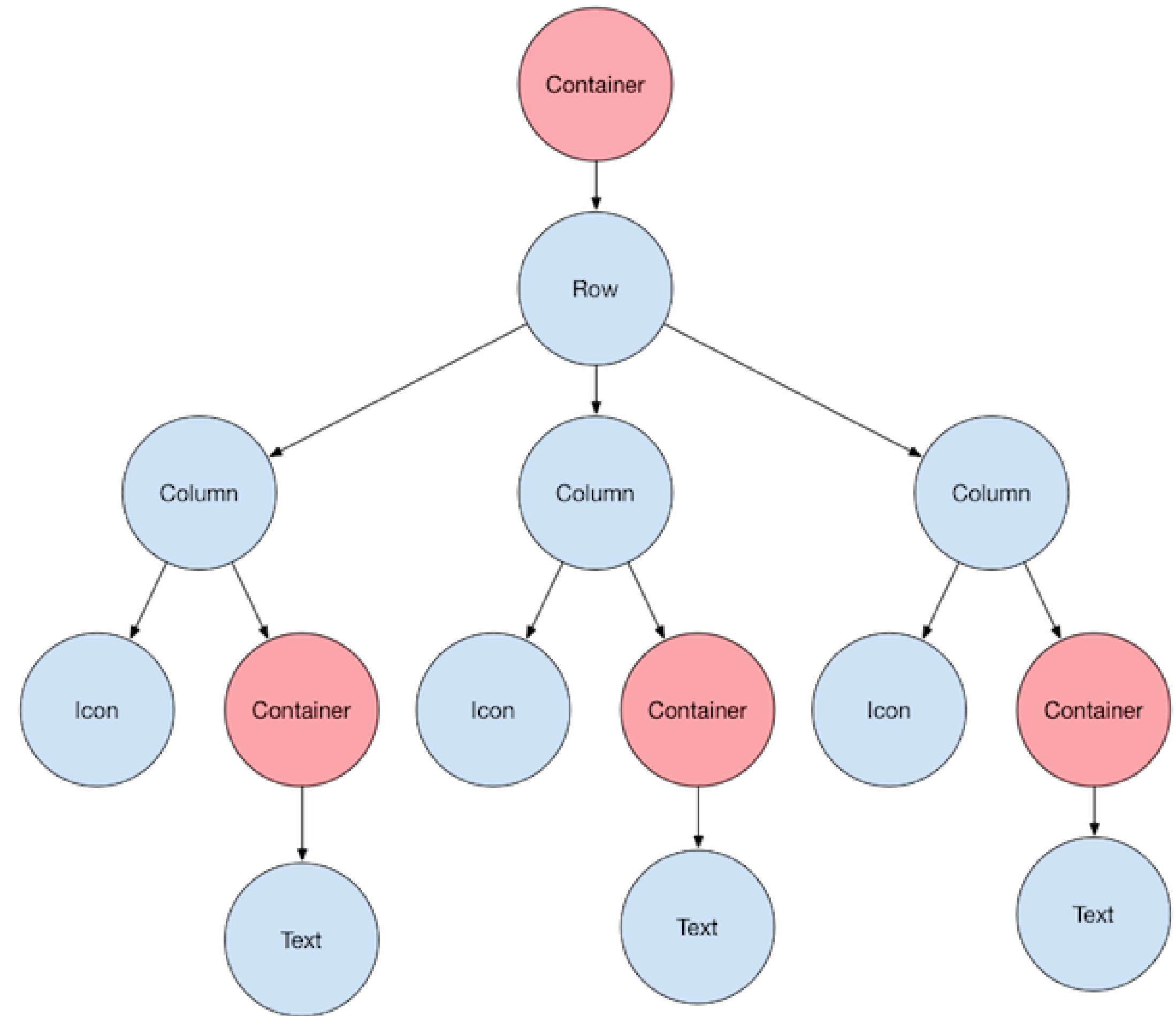
Stateless Stateful

- To identify which widget type are
 - Pattern
 - initState()
 - SetState
- 5 mins break

Widget Tree

How your code looks like

- Concept => Layers – Section - branches
- Terminologies
 - Parent
 - Child
 - Children
 - Nested Structure
- Performance – D.R.Y not W.E.T



Mini Checkpoint 2



Chapter 4: Why Flutter



TLDR;

Flutter is an open source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase.

Advantages

- Native feeling
- Single code base for Multiple platform (Universal)
- Rich Animations – 60FPS smoothness
- Beautiful UIs – Material Design

Limitations

- Single Code base => if else conditions
- Multiple platform -> knowledge in multiple platform to implement additional native functionalities
- Rich Animations – 60FPS smoothness => performance / battery life/ memory consumption management
- iOS -> any new feature need to wait for 3rd party (open-source community) to develop and maintain
- iOS -> most of the patch focus on iOS improvement and optimization
- Flutter try to be jack of all trades

Common Pits Fall

- Using Flutter for only single platform
 - Alternatives – Jetpack Compose, SwiftUI
- Code once, you need to choose which platform that can goes together
- Replacing -> co-exists

Take A Break 15 mins

Chapter 5: Introduction to Widgets

Introduction to Widgets

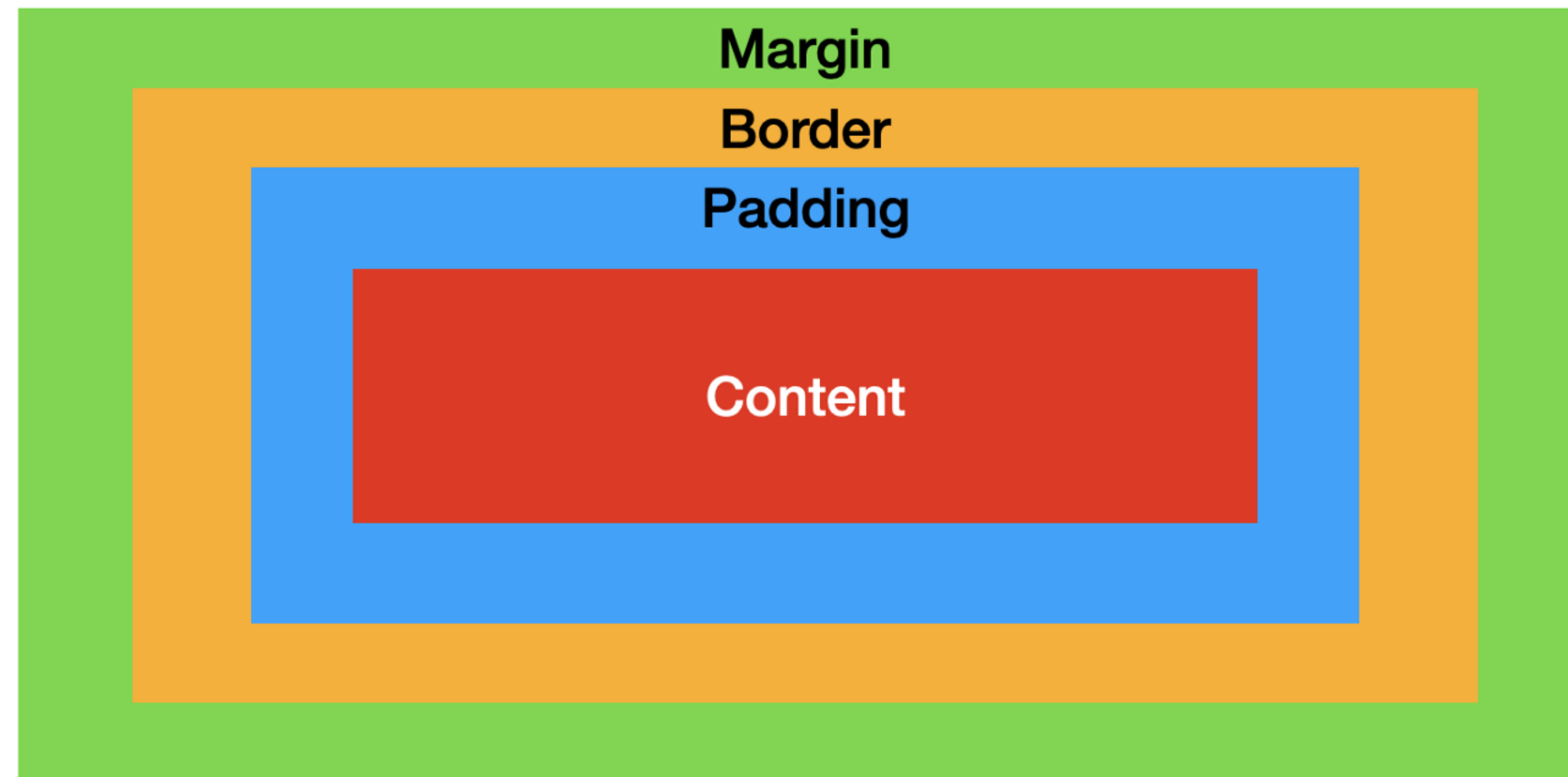
- Any UI components in flutter, either small or large are called "Widgets"

Basic widget - Container

- Multiple Properties
- BoxModel (CSS)
 - Padding
 - Margin
- Alternative
 - SizedBox()

```
Container(  
  constraints: BoxConstraints.expand(  
    height: Theme.of(context).textTheme.headline4!.fontSize! * 1.1 + 200.0,  
  ),  
  padding: const EdgeInsets.all(8.0),  
  color: Colors.blue[600],  
  alignment: Alignment.center,  
  transform: Matrix4.rotationZ(0.1),  
  child: Text('Hello World',  
    style: Theme.of(context)  
      .textTheme  
      .headline4!  
      .copyWith(color: Colors.white)),  
)
```

Box Model



Mini Checkpoint 3

Fill colors and spaces to your container

Basic widget - Text



```
Text(  
  'Hello, $_name! How are you?',  
  textAlign: TextAlign.center,  
  overflow: TextOverflow.ellipsis,  
  style: const TextStyle(fontWeight: FontWeight.bold),  
)
```

- we call this typography

Mini Checkpoint 4

Playing with Text

Basic widget - Image

To display image inside you application

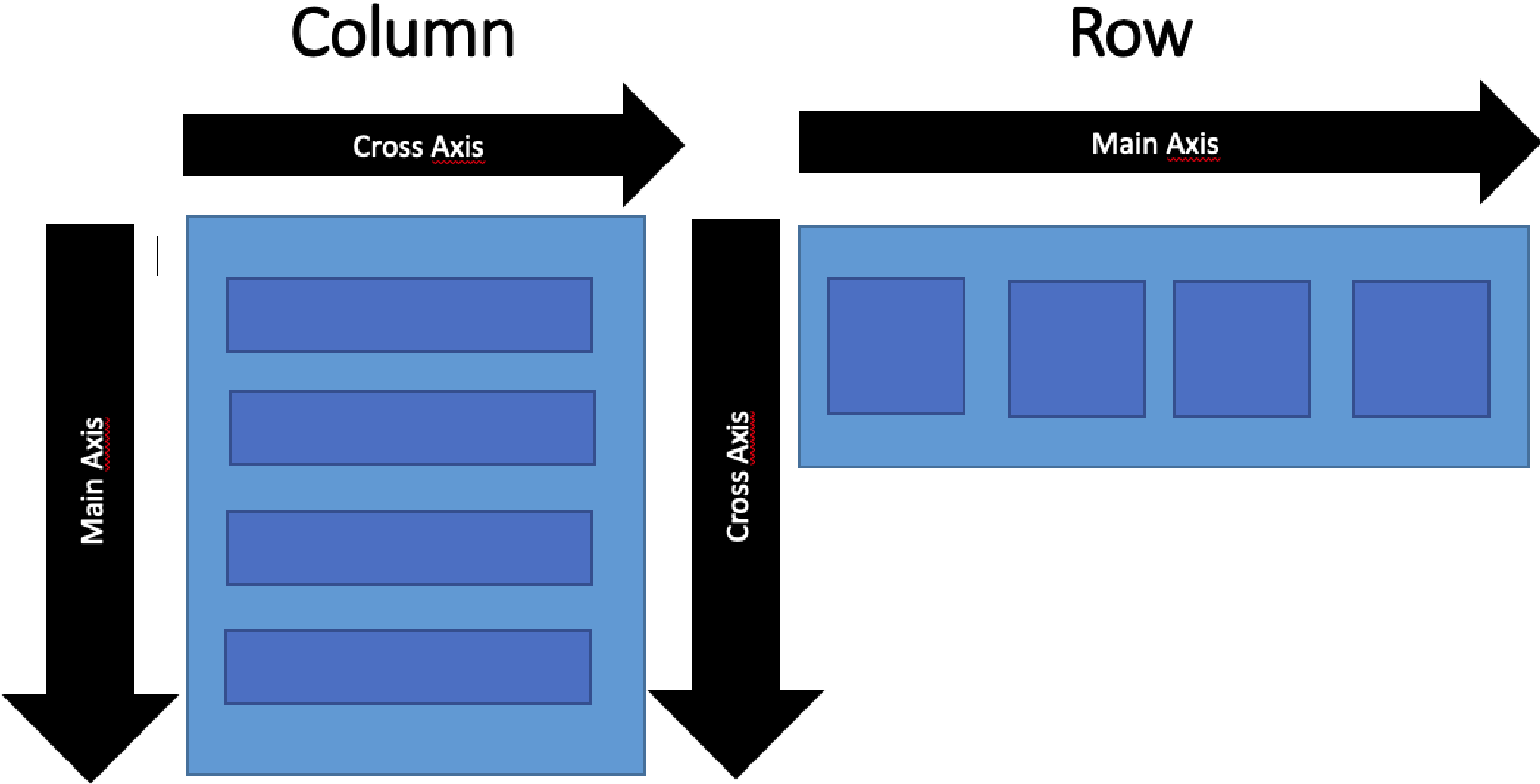
- Image.assets -> Load image from local
 - Create "assets" folder
 - Add path to pubspec.yaml
- Image.network -> Load image from network

```
const Image(  
  image: NetworkImage('https://flutter.github.io/assets-for-api-docs/assets/widgets/owl.jpg'),  
)
```

Mini Checkpoint 5

Playing with Image

Basic widget – Row and Column





Flutter Clutter

Implementing edge detection in Flutter



Google News

Amazing: Flutter ist the most famous mobile

Imaginary Computer Binding

Check for overflow in the widget

BOTTOM OVERFLOWED BY 32 PIXELS

OVERFLOWED BY 152 PIXELS

Mini Checkpoint 6

Adding scroll functionality to your column/row

- You encounter an overflow
- Wrap SingleChildScrollView
- Tell the direction of scroll
 - Axis.horizontal
 - Axis.vertical

Introduction to Alignments

- 2 main properties for both row and column
 - MainAxisAlignment
 - CrossAxisAlignment

| .center | .start | .end | .spaceEvenly | .spaceAround | .spaceBetween |
|---|---|---|---|---|---|
|  |  |  |  |  |  |

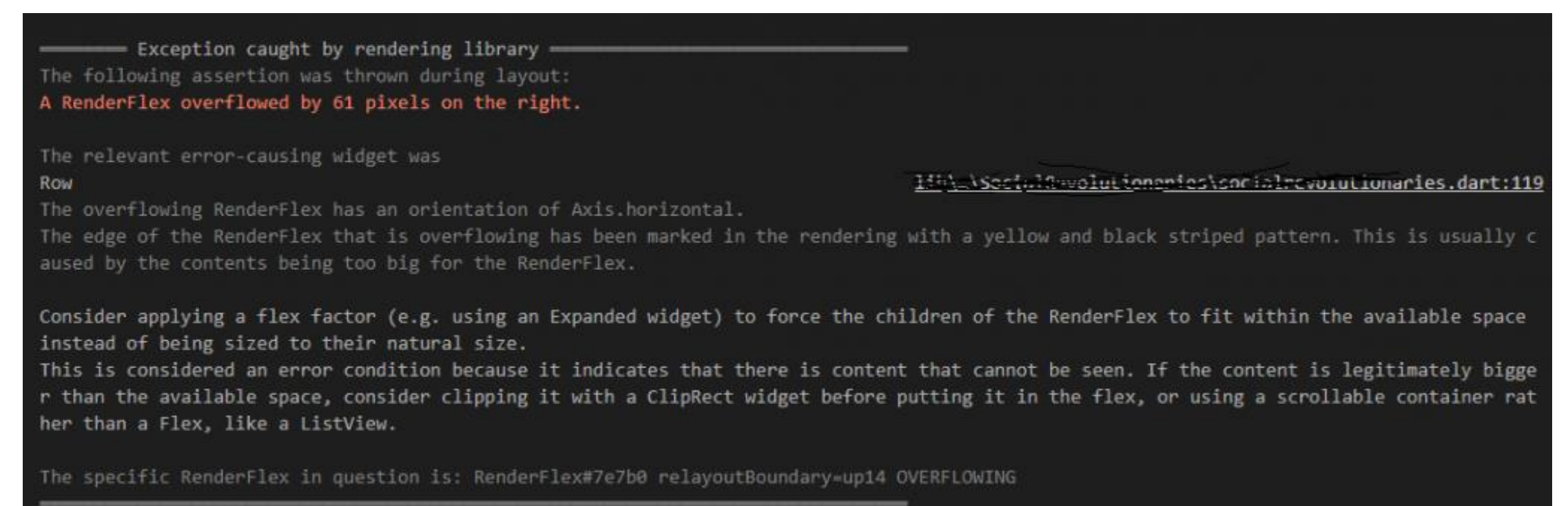
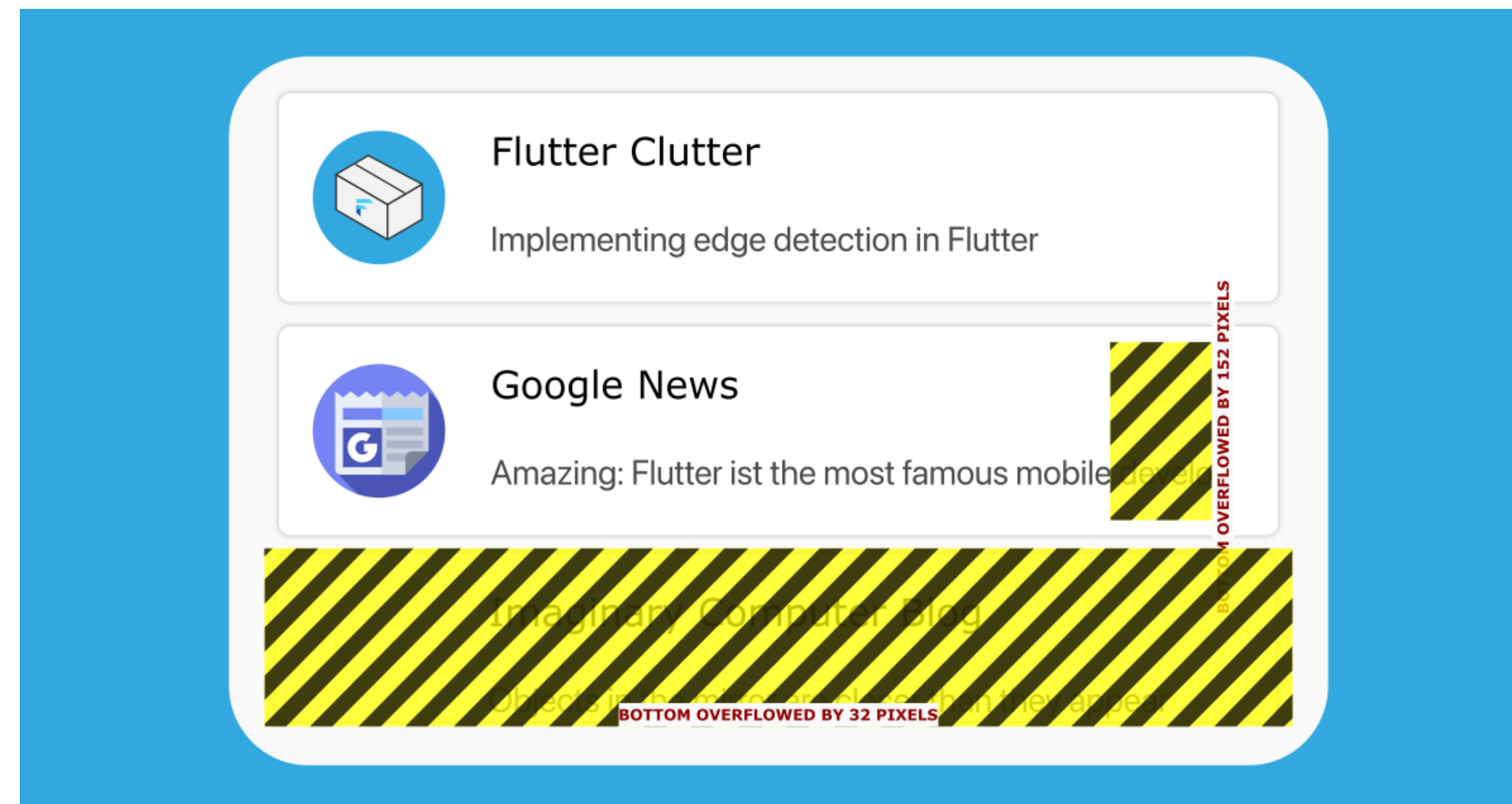
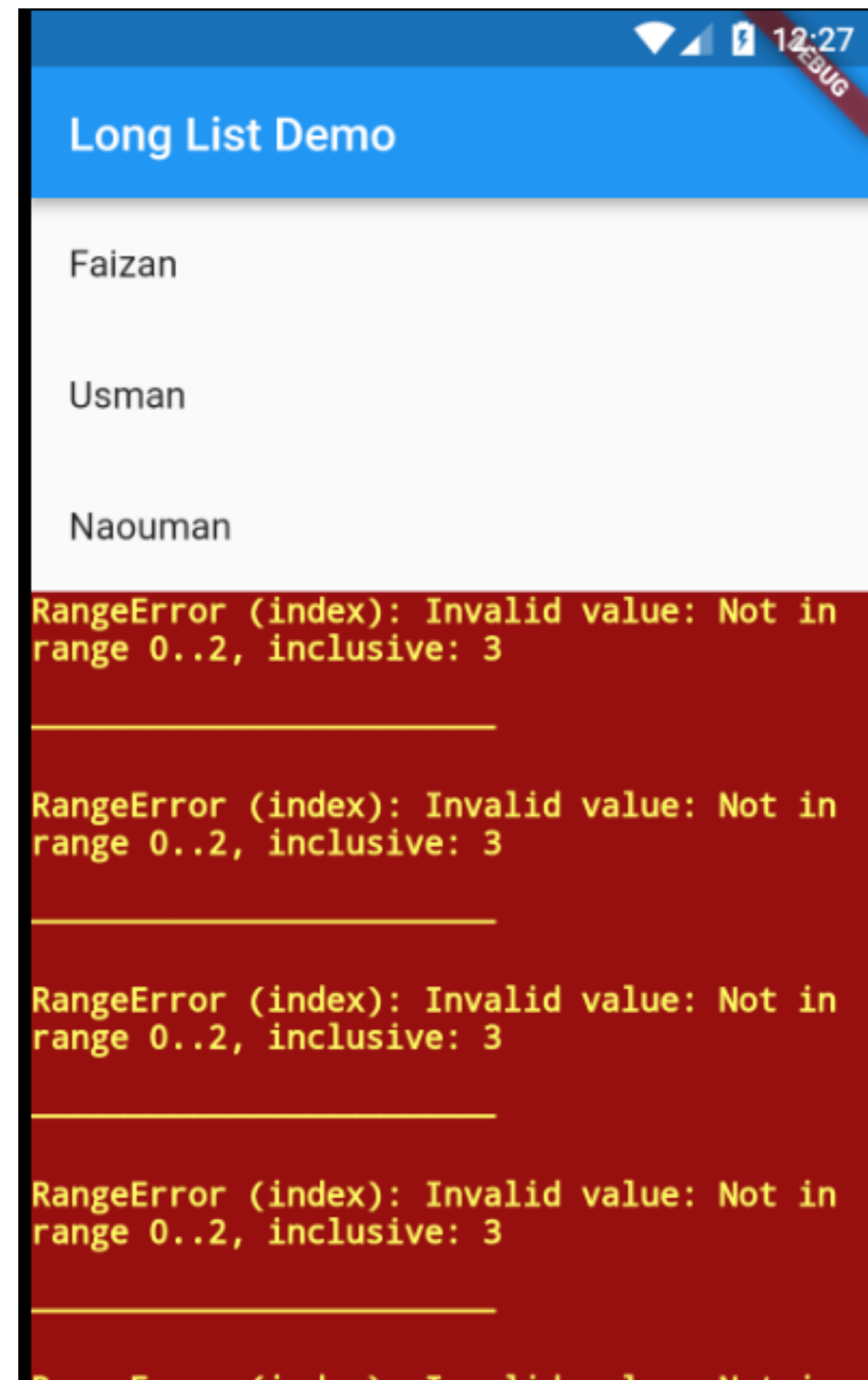
Mini Checkpoint 7

Align you widgets

Further Study

- [Flutter Layout Cheat Sheet. Do you need simple layout samples for... | by Tomek Polański | Flutter Community](#)

Widget Error Handling



Take A Break 10 mins

Chapter 6: *Advanced layout building*

Grid View

Now you know

- `GirdView.count()`
- `GridView.builder()`
- Pit fall -> Combine Row and Column to make a grid

List View

- `ListView.count()`
- `Listview.builder()`
- Pit fall -> List View != Column

Stack



```
Stack(  
  alignment: Alignment.center,  
  textDirection: TextDirection.rtl,  
  fit: StackFit.loose,  
  overflow: Overflow.visible,  
  clipBehavior: Clip.hardEdge,  
  children: <Widget>[]  
)
```

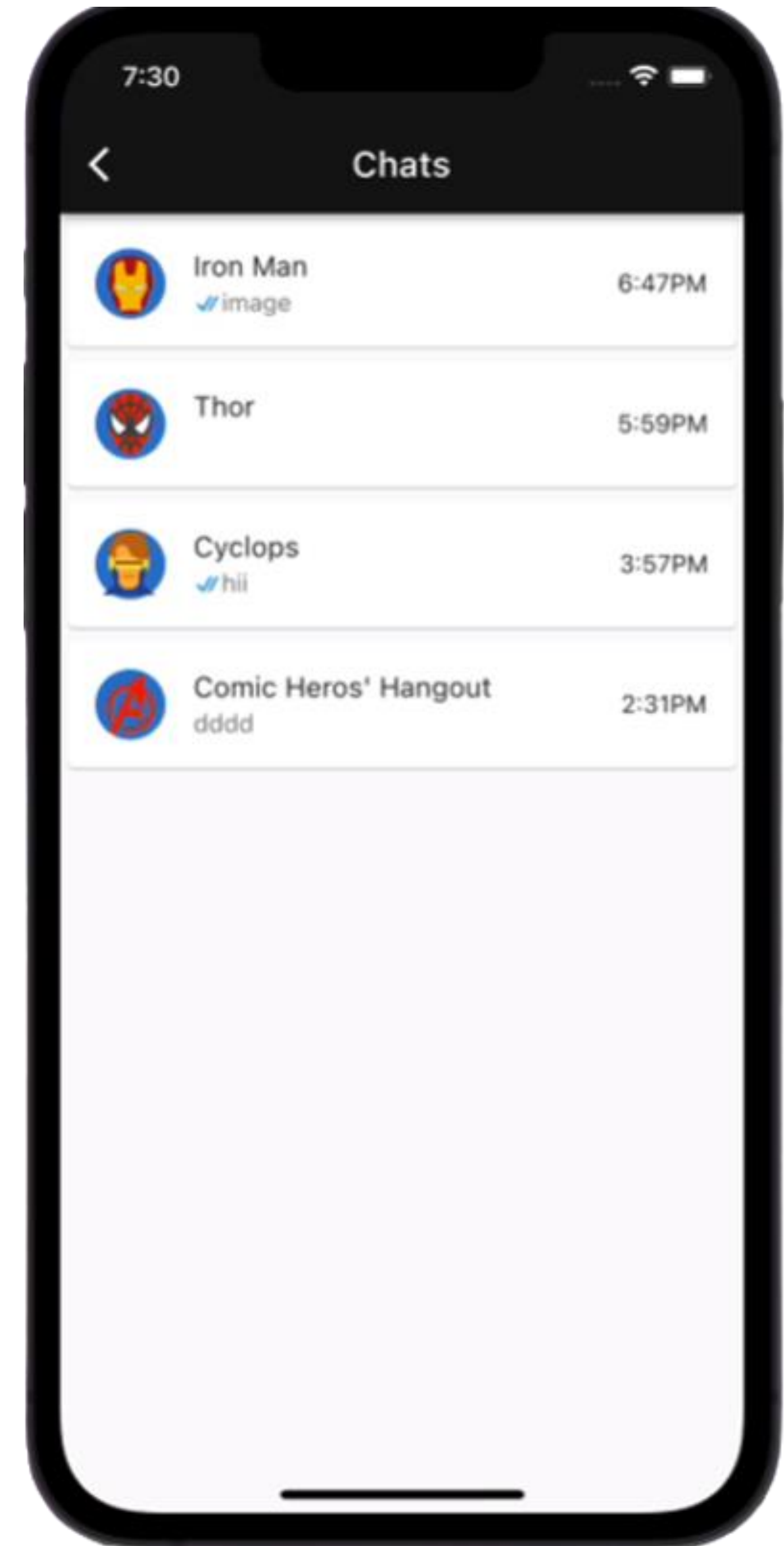
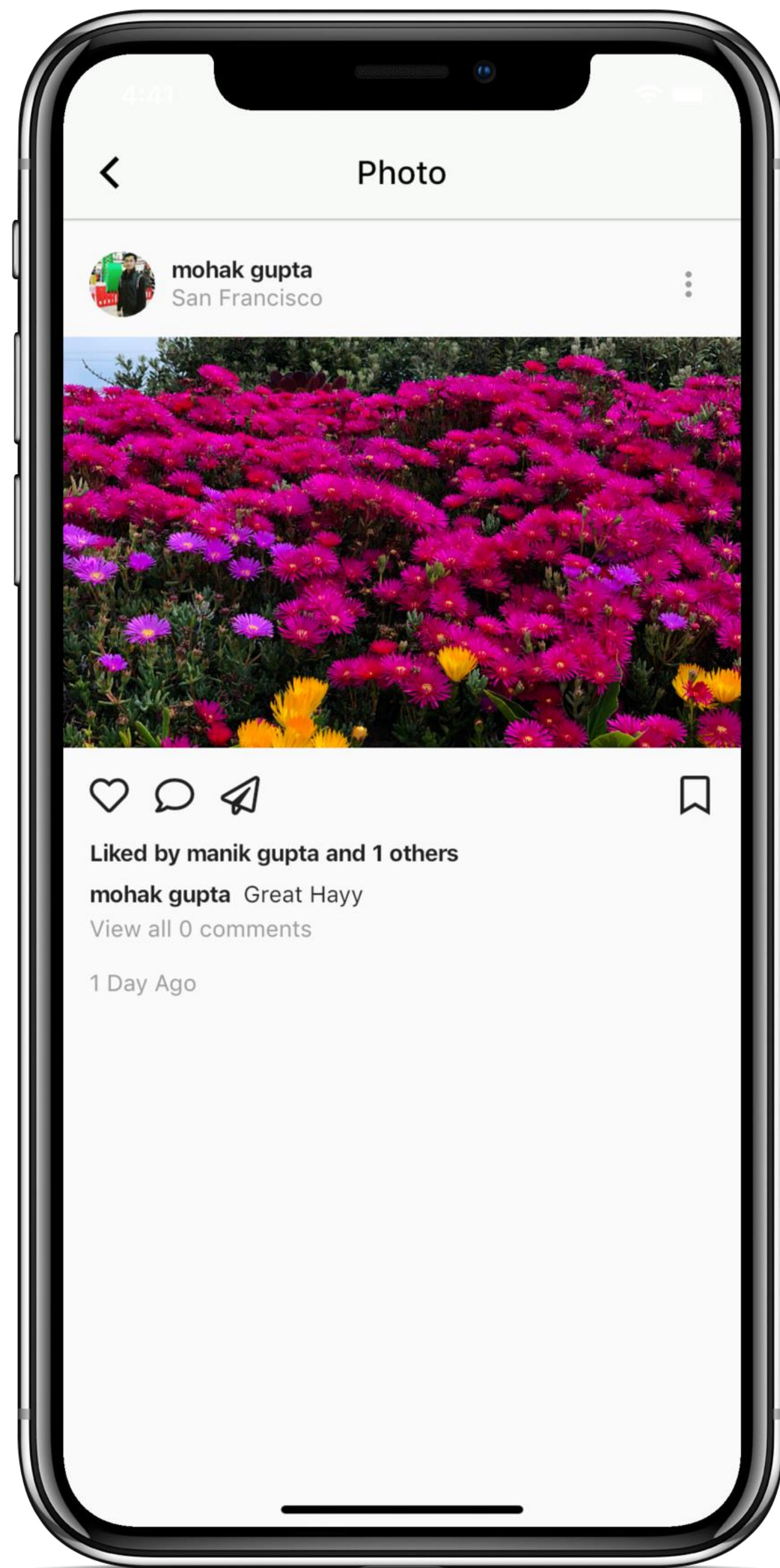
- When you want to stack components on top each other
- For example -> Overlay loading

Take A Break 10 mins

2nd Checkpoint – Creating UI from given image

aka Code in the dark

- Choose reference UIs – 5 min
- Think/discuss about how this will build using flutter (in terms of layout/ widgets) - 5 min
- Explain your plan - 5 min each
- 15 mins for coding
- Let discuss key take away - 5 mins each
- Q/A



Hints - Helpers

Scope down

- Static
- Standard Fonts
- Image -> fill with any color
- Internet is Allow

11:11

VoLTE 4G+ 100%
DEBUG

Ban Charoen Suk

Clouds

300.08°F



Nov 22, 2019

23:08



Ban Charoen Suk

Clouds

299.81°F



Nov 23, 2019

01:00

Ban Charoen Suk

Clouds

298.98°F



Nov 23, 2019

04:00

Ban Charoen S

Clouds

298.82°F



Nov 23, 2019

07:00

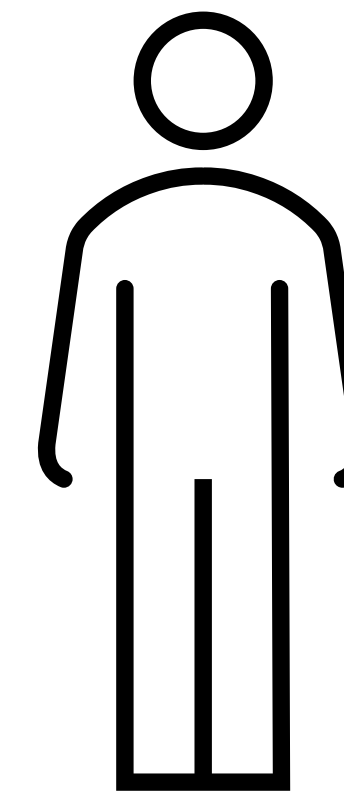
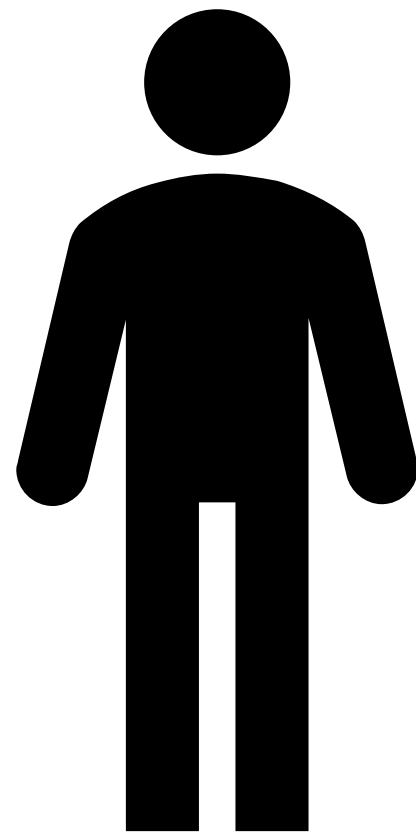
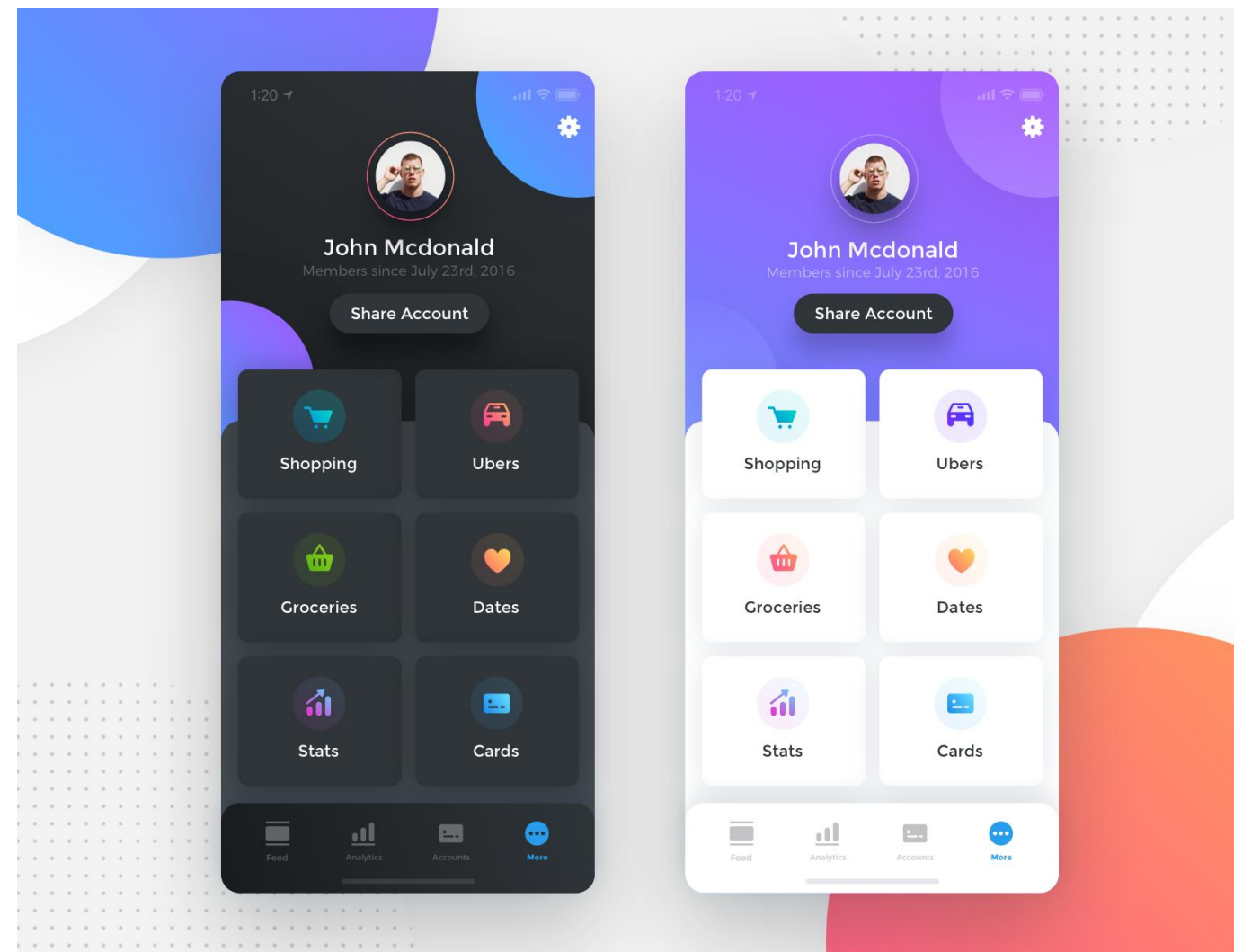


Hints - Helpers

Scope down

- Static
- Standard Fonts
- Image -> fill with any color
- Internet is Allow

Key take away from this checkpoint



Further Study

- <https://m3.material.io>
- [Atomic Design | Brad Frost](#)

Summaries

Now you know

- How to install/setup flutter SDK in your local machine
- Basic flutter command
- Fundamental of flutter components
- Dart Lang*
- Flutter Widgets
- Design system in general/ flutter/~~atomic design~~

Homework

- Create GitHub account – individual -> studentID_groupName
- And upload all the checkpoint code to GitHub
 - mini_checkpoint_x
 - checkpoint_x
- And submit git username -> I'll post the form in Discord Channel
- I'll upload the code within this **Friday** -> so everyone can review
- I'll send the additional information -> in Discord Channel

Recap Part 1

30 mins

TLDR;

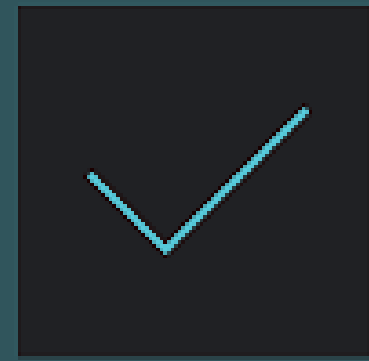
- Understand not remember
 - Cover multiple aspect
 - Take notes – perspective, discussion
- Follow Along - I'll. Go slow, 🐢 I'm promise 😊
- Practices is the key things
- Enjoy coding

Table of Contents

These are the topics that you are going to train throughout this course

- Chapter 1: Installation and Setup for Flutter Development
- Chapter 2: Creating 1st Flutter app
- ~~• Chapter 3: Basic Dart Lang*~~
- Chapter 4: Why Flutter?
- Chapter 5: Introduction to Widgets
- ~~• Chapter 6: Advanced Layout Building~~
- Chapter 7: Design System
- Chapter 8: Building form application in Flutter
- Chapter 9: Navigation
- Chapter 10: Introduction to State Management
- ~~• Chapter 11: E2E testing with Cucumber~~
- Chapter 12: Software Architecture
- Chapter 13: Working with 3rd party libs
- ~~• Chapter 14: Platform Specific Code~~
- Chapter 15: Releasing Application*

Chapter 7: Design System



Material Design

is an adaptable system of guidelines, components, and tools that support the best practices of user interface design

- Focus Platform
 - Mobile
 - Android (Native)
 - iOS (native) -> m3 not transition to iOS
 - Cross platform -> Flutter
 - Web
- Responsiveness
- Surface/ Shadows/ Reflection/ Ripple effect/ Transitions
- All the example in this section will refer to material design v2

Material Design

Typography

- Concept like CSS property
- Fonts
 - Size
 - Family
 - Weight – Bold, Regular,
 - Color
 - Etc.

Material Design

Iconography

- Refers to icons in material design
- Built-in to flutter framework
- Mostly 1:1 Ratio
- Multiple style – outlined, fill

Material Design

Components – buttons – show example

```
TextButton(  
  style: ButtonStyle(  
    foregroundColor: MaterialStateProperty.all<Color>  
(Colors.black87),  
    backgroundColor: MaterialStateProperty.resolveWith<Color>(  
      (Set<MaterialState> states) {  
        if (states.contains(MaterialState.hovered))  
          return Colors.blue.withOpacity(0.04);  
        if (states.contains(MaterialState.focused) ||  
            states.contains(MaterialState.pressed))  
          return Colors.blue.withOpacity(0.12);  
        return null; // Defer to the widget's default.  
      },  
    ),  
  ),  
  onPressed: () {},  
  child: Text('TextButton')  
)
```

```
ElevatedButton(  
  style: ElevatedButton.styleFrom(  
    onPressed: Colors.black87,  
    primary: Colors.grey[300],  
    minimumSize: Size(88, 36),  
    padding: EdgeInsets.symmetric(horizontal: 16),  
    shape: const RoundedRectangleBorder(  
      borderRadius:  
BorderRadius.all(Radius.circular(2)),  
    ),  
    onPressed: () {},  
    child: Text('Looks like a RaisedButton'),  
  )
```

- [Migrating to the New Material Buttons and their Themes - Google Docs](#)
- V1 - V2

Mini Checkpoint 8

- Add Buttons

Material Design

Theme

- Default theme for any modern platform – light / dark



```
MaterialApp(  
  theme: ThemeData.light(), // or ThemeData.dark()  
);
```

Material Design

Responsiveness – dynamic size

- Basic -> `MediaQuery.of(context).size` -> Apply example with button

A dark-themed code editor window with three colored window control buttons (red, yellow, green) in the top-left corner. The text `MediaQuery.of(context).size.width` is displayed in a monospaced font, with `MediaQuery` in purple, `.of(context)` in teal, `.size` in yellow, and `.width` in yellow. A red cursor is at the end of the line.

```
MediaQuery.of(context).size.width
```

A dark-themed code editor window with three colored window control buttons (red, yellow, green) in the top-left corner. The text `MediaQuery.of(context).size.height` is displayed in a monospaced font, with `MediaQuery` in purple, `.of(context)` in teal, `.size` in yellow, and `.height` in yellow. A red cursor is at the end of the line.

```
MediaQuery.of(context).size.height
```

Mini Checkpoint 9



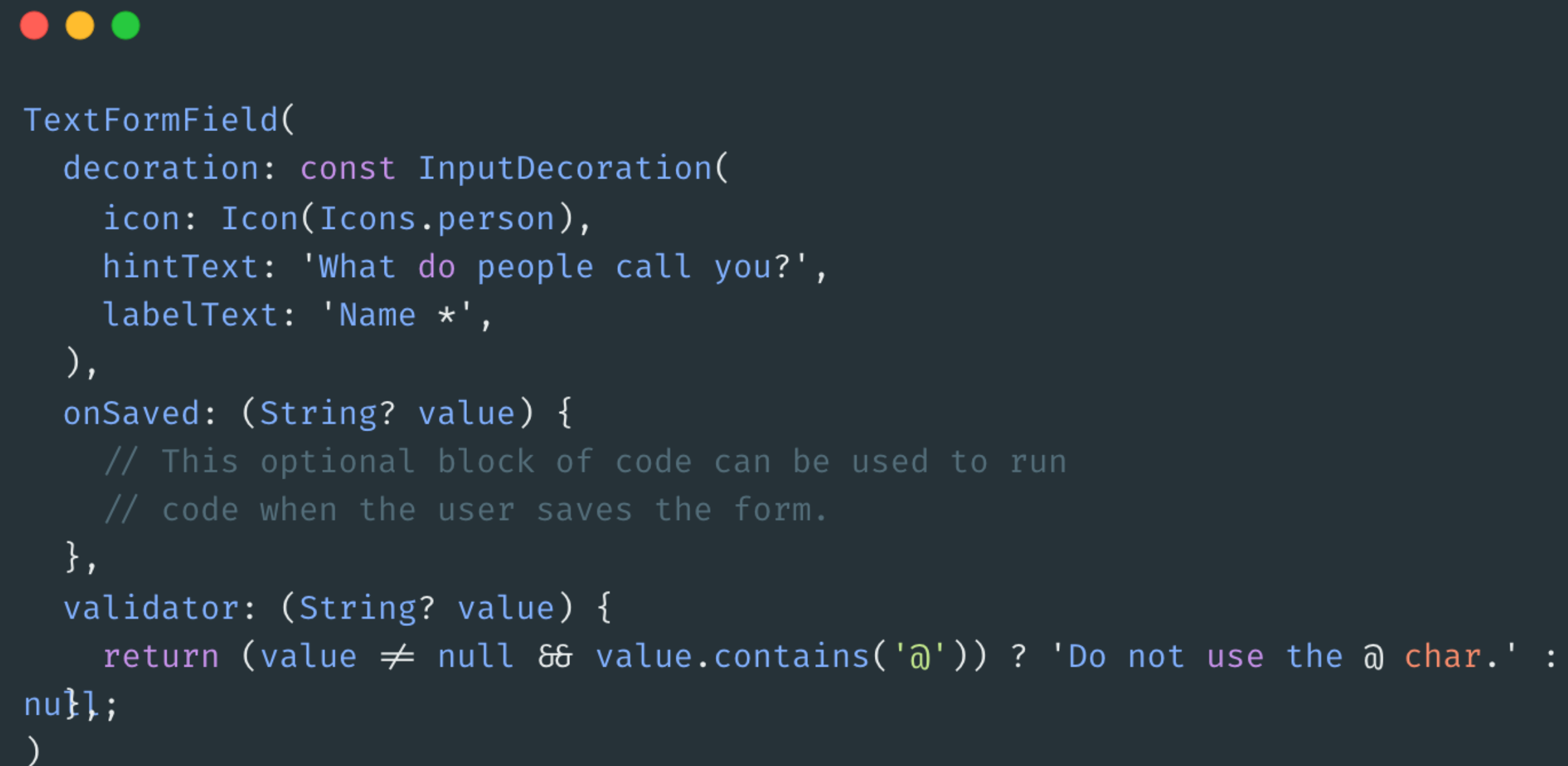
Take a break 10 mins

Chapter 8: Introduction to Widgets – Form

Widget - Form

- Group of inputs – Text, choices [radio, checkbox], Dropdown selection
- Can validate individual
- Or Validate entire form.


Widget – TextFormField



```
TextFormField(  
  decoration: const InputDecoration(  
    icon: Icon(Icons.person),  
    hintText: 'What do people call you?',  
    labelText: 'Name *',  
  ),  
  onSave: (String? value) {  
    // This optional block of code can be used to run  
    // code when the user saves the form.  
  },  
  validator: (String? value) {  
    return (value != null && value.contains('@')) ? 'Do not use the @ char.' :  
    null;  
  })
```

Mini Checkpoint 10

Widget – Radio button



```
Radio<String>(
  value: 'Male',
  groupValue: _gender,
  onChanged: (String gender)
{  setState(() {
      _gender = gender;
    });
},
),
```

Mini Checkpoint 11

Widget - Checkbox



```
Checkbox(  
  checkColor: Colors.white,  
  fillColor:  
MaterialStateProperty.resolveWith(getColor),  
  onChanged: (bool? value) {  
    setState(() {  
      isChecked = value!  
    });  
  },  
);
```


Mini Checkpoint 12

Widget - DropdownMenu

```
DropdownButton<String>(  
  value: dropdownValue,  
  icon: const Icon(Icons.arrow_downward),  
  elevation: 16,  
  style: const TextStyle(color: Colors.deepPurple),  
  underline: Container(  
    height: 2,  
    color: Colors.deepPurpleAccent,  
  ),  
  onChanged: (String? value) {  
    // This is called when the user selects an item.  
    setState(() {  
      dropdownValue = value!  
    });  
  },  
  items: list.map<DropdownMenuItem<String>>((String value)  
{    return DropdownMenuItem<String>(  
      value: value,  
      child: Text(value),  
    );  
}).toList(),  
)
```

Mini Checkpoint 13


Validator

- On each form element we can have validator to validate input match or not match and can be trigger the error message

```
TextFormField(  
  // The validator receives the text that the user has  
  validator: (value) {  
    if (value == null || value.isEmpty) {  
      return 'Please enter some text';  
    }  
    return null;  
  },  
)
```

3rd Checkpoint: Covid Patient Form

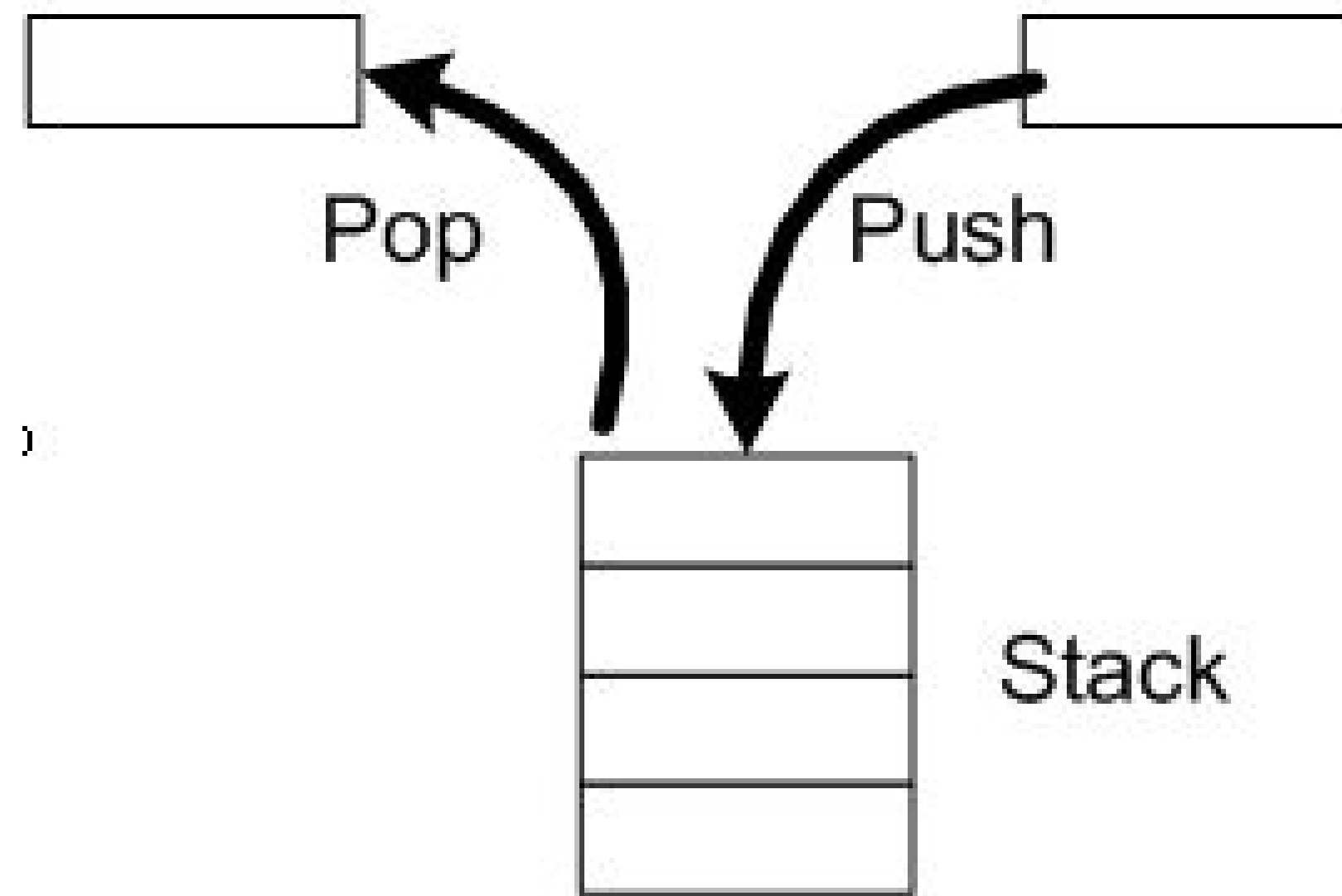
- Objective -> Create a form and pass data between pages.
- Code complexity -> Beginner
- Use Widget
 - Form
 - TextFormField
 - RadioButton
 - CheckBox
 - Navigation between page*
- Final code at here -> https://github.com/Krseisenh/covid_form



The screenshot displays a mobile application interface for a 'Patient Form'. The form is titled 'Patient Form' in a blue header bar. It contains several input fields: 'Firstname', 'Lastname', and 'Age' are text input fields. The 'Gender' section uses radio buttons for 'Male' and 'Female'. The 'Symptoms' section uses checkboxes for three categories: 'ไอ' (Cough), 'เจ็บคอ' (Sore throat), and 'มีไข้' (Fever). A 'Save' button is located at the bottom right of the form. The status bar at the top shows the time as 7:01, signal strength, and battery level at 97%. A red 'DEBUG' banner is visible in the top right corner.

Section 9: Navigation

Stack Concept (Push, Pop)



Nav 1.0



```
Navigator.push(  
  context,  
  MaterialPageRoute(builder: (context) => SecondPage())  
);  
  
Navigator.pop(context)  
  
Navigator.pushNamed(  
  context,  
  '/second_page',  
  arguments: {},  
);
```


Mini Checkpoint 14

Limitation

- Once the flow of the app is not linear, will be hard to manage widget stack using push and pop method
- Page required preloaded data
- Require to remapping widget to corresponding path

Section 11: Software Architecture

Introduction to “Mobile” Application Architecture – Concept

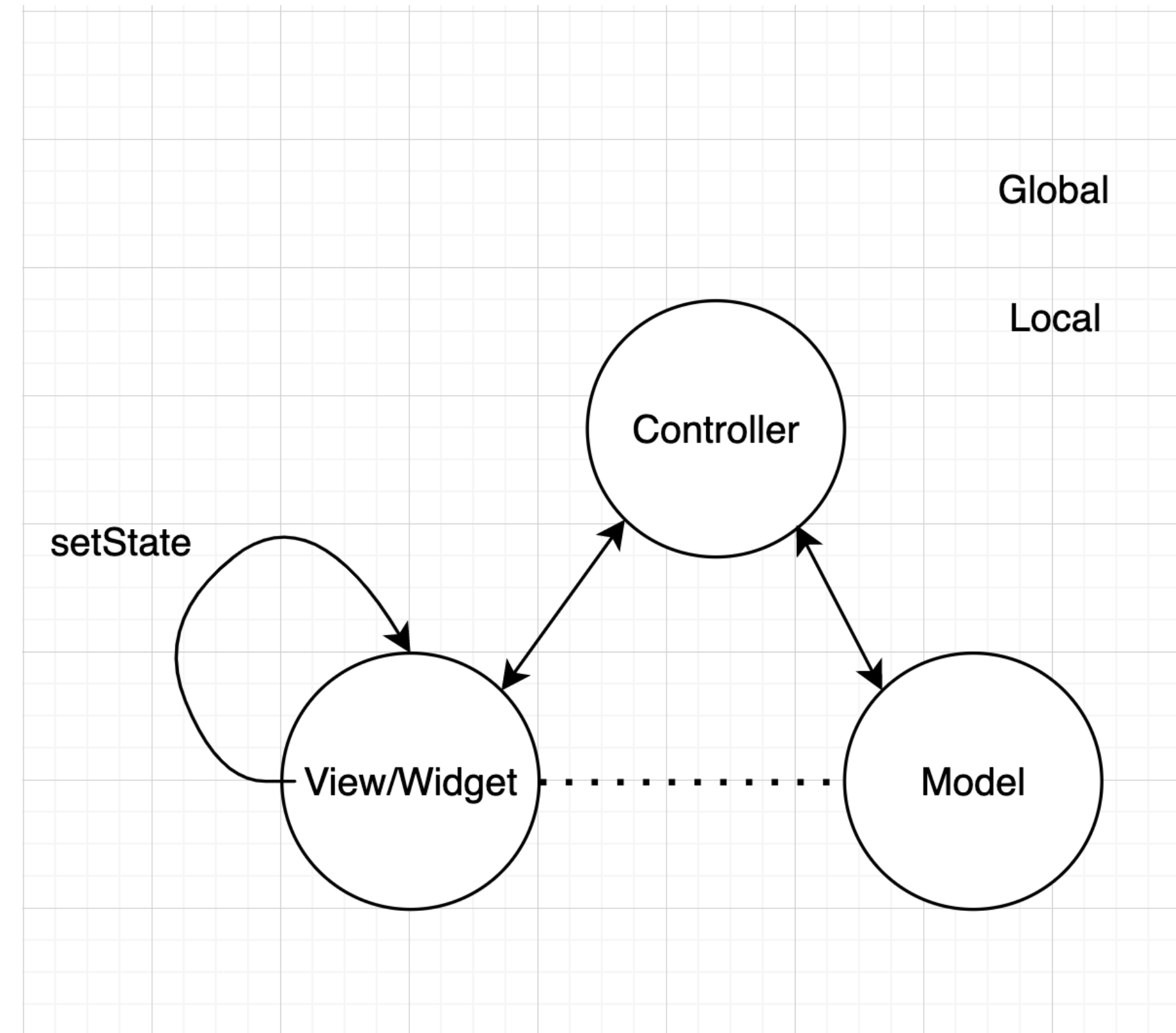
- Building a house





Mobile Clean Architecture

- MVVM
- MVC
- Separate between Business logic and UIs
- In flutter, we have things call Buildcontext
- for example, => logic to perform navigation



Section 12: Working with 3rd party libs

Introduction to Pub.dev

- Where all the "open-source" 3rd party library are located.
- You can submit your lib
- All platform support or platform specific support

How to choose 3rd lib in your project

Experience sharing

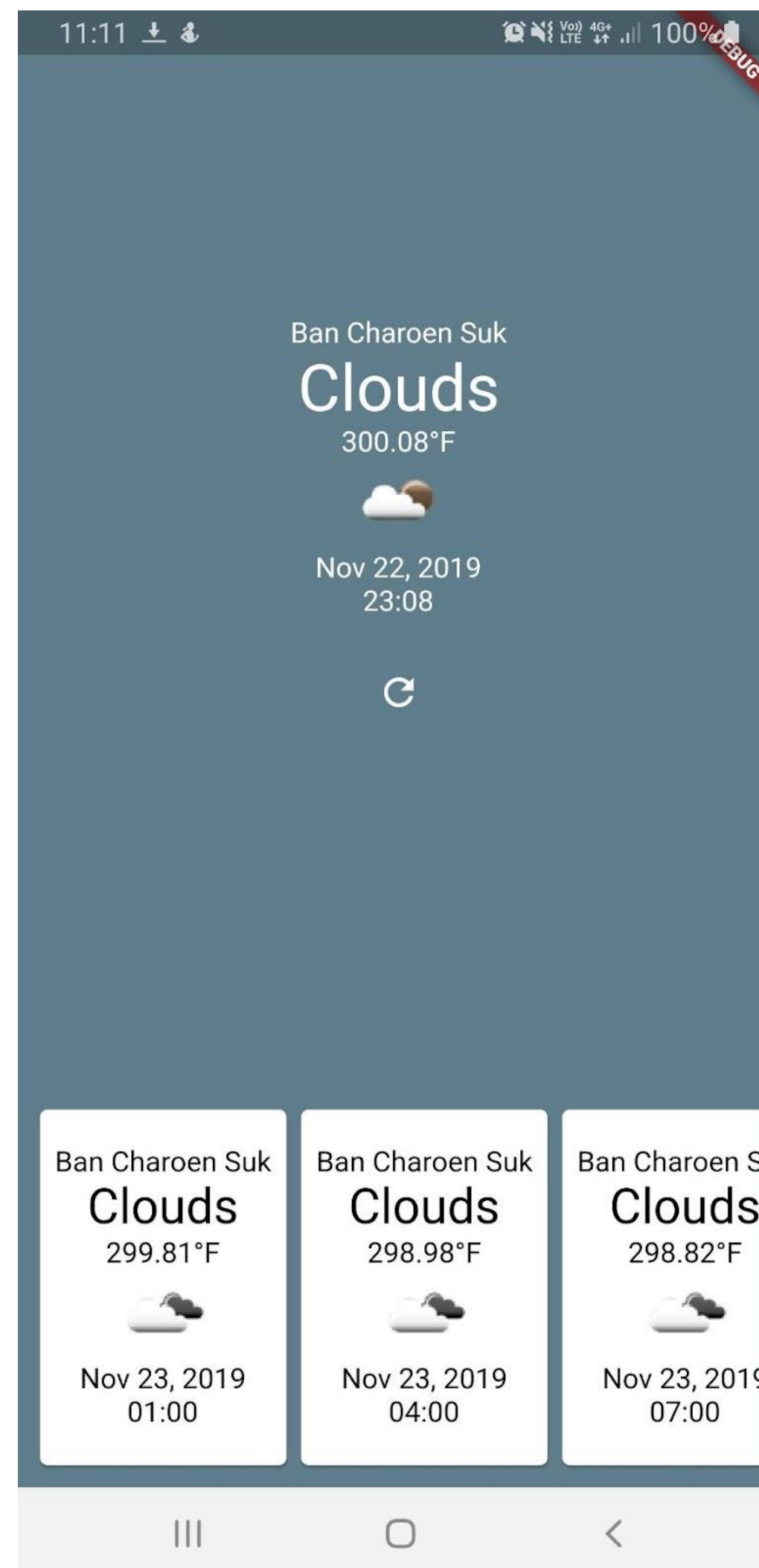
- By looking at the lib REAMMD.md file might not give you the exact functions you want
- Go to their GitHub repo
- Then clone the project and take a look at their example folder

Popular libraries

- [Dio](#)
- flutter_configs/flutter_dotenv
- [Location](#) -> Intro to Native Bridge
- ~~easy_localization~~

Section 13: Introduction to State Management

4th Checkpoint - setState



Dealing with Legacy Code

- Objective -> refactor code, setState, using 3rd party libs
- Code complexity-> intermediate
- Widget involve
 - Text
 - Image
 - Row
 - Column
- Final code at here -> https://github.com/3IVv0w/today_weather.git

Chapter 15: Releasing application

Android

Commands

- Generated keystore and update gradle file
- flutter clean
- flutter build apk --release
- flutter build appbundle --release

IOS - optional

- Open and follow along in the XCode

Summaries

Homework

- As Group – Start discussing what application do you want to do as final project
 - SPA (single page application) - MPA (2 pages)
 - Query data (API) and display in screen
 - Simple UI
- Upload all the checkpoint code to GitHub -> submit DL (this **Friday**)
 - mini_checkpoint_x
 - checkpoint_x
- I'll upload the code within this **Friday** -> so everyone can review
- I'll send the additional information -> in Discord Channel