Hacker Rank

Coding Challenges

- The next 3 Fridays
- A2024 – 10.15 AM – 11.30AM
- 30 students max for sessions

Coaches are welcome !!!

REGISTER HERE

CADT IDT Institute of Digital Technology

MOBILE DEVELOPMENT

W3-S1 – A First Flutter App

CADT IDT

Flutter

Course Objectives

DART

- ✓ Identify **Positional**, **Named**, **Optional**, **Mandatory** & **Default** *arguments*

REVISIONS

FLUTTER

- ✓ Overview of **Flutter architecture**
- ✓ Main steps to **run a Flutter project**

- ✓ Flutter **Widget-oriented** approach
- ✓ Basic Flutter **widgets**

- ✓ **const** and final to **optimize runtime** performances

EXPLAIN

Dart arguments can be ‘**named**’ or ‘**positional**’

EXPLAIN

Positional, named, optional, mandatory, default **arguments**

Positional argument

Named argument

```
class Text {
  const Text(
            this.data,
      {     required this.textAlign,
            this.textDirection,
            this.overflow=TextOverflow.clip
       });

  final String data;
  final TextDirection? textDirection;
  final TextAlign? textAlign;
  final TextOverflow overflow;
}

main() {
  Text t1 = const Text("hello", textAlign: TextAlign.center);
}
```

Cannot be null, but the argument
Is optional , because the default value
Is provided

Argument	Positional / Named	Mandatory / Optional	Default value?
data	positional	mandatory	no
textAlign	named	optional	no
textDirection	named	optional	no
overflow	named	optional	default values

QUIZ

Which of the following constructors calls **are valid**? (*Select all that apply*)

```
class ExampleWidget  {

  final String title;
  final String? subtitle;
  final int count;
  final Color color;

  const ExampleWidget(
    this.title,
    {
    this.subtitle,
    required this.count,
    this.color = Colors.blue,
  });
}
```

1. `ExampleWidget('Welcome', count: 5)`
2. `ExampleWidget('Hello', subtitle :'Subtitle', count: 10, color: Colors.red)`
3. `ExampleWidget('Hi', color: Colors.green)`
4. `ExampleWidget('Greetings', 'Special Subtitle')`
5. `ExampleWidget(title: 'Hey', count: 3)`
6. `ExampleWidget('Welcome', count: 7, color: Colors.yellow)`

QUIZ

Which of the following constructors calls **are valid**? (*Select all that apply*)

```
class ExampleWidget  {

  final String title;
  final String? subtitle;
  final int count;
  final Color color;

  const ExampleWidget(
    this.title,
    {
    this.subtitle,
    required this.count,
    this.color = Colors.blue,
  });
}
```

1. `ExampleWidget('Welcome', count: 5)`
2. `ExampleWidget('Hello', subtitle :'Subtitle', count: 10, color: Colors.red)`
3. `ExampleWidget('Hi', color: Colors.green)`
   Count is required
4. `ExampleWidget('Greetings', 'Special Subtitle')`
   Count is required
5. `ExampleWidget(title: 'Hey', count: 3)`
   Title is positional
6. `ExampleWidget('Welcome', count: 7, color: Colors.yellow)`

ACTIVITY

Activity 1

*Complete the table with the **right attributes***

```
class ExampleWidget  {

  final String title;
  final String? subtitle;
  final int count;
  final Color color;

  const ExampleWidget(
    this.title,
    {
    this.subtitle,
    required this.count,
    this.color = Colors.blue,
  });
}
```

Parameter	Positional / Named	Mandatory / Optional	Default value?
title			
subtitle			
count			
color			

ACTIVITY

Activity 1

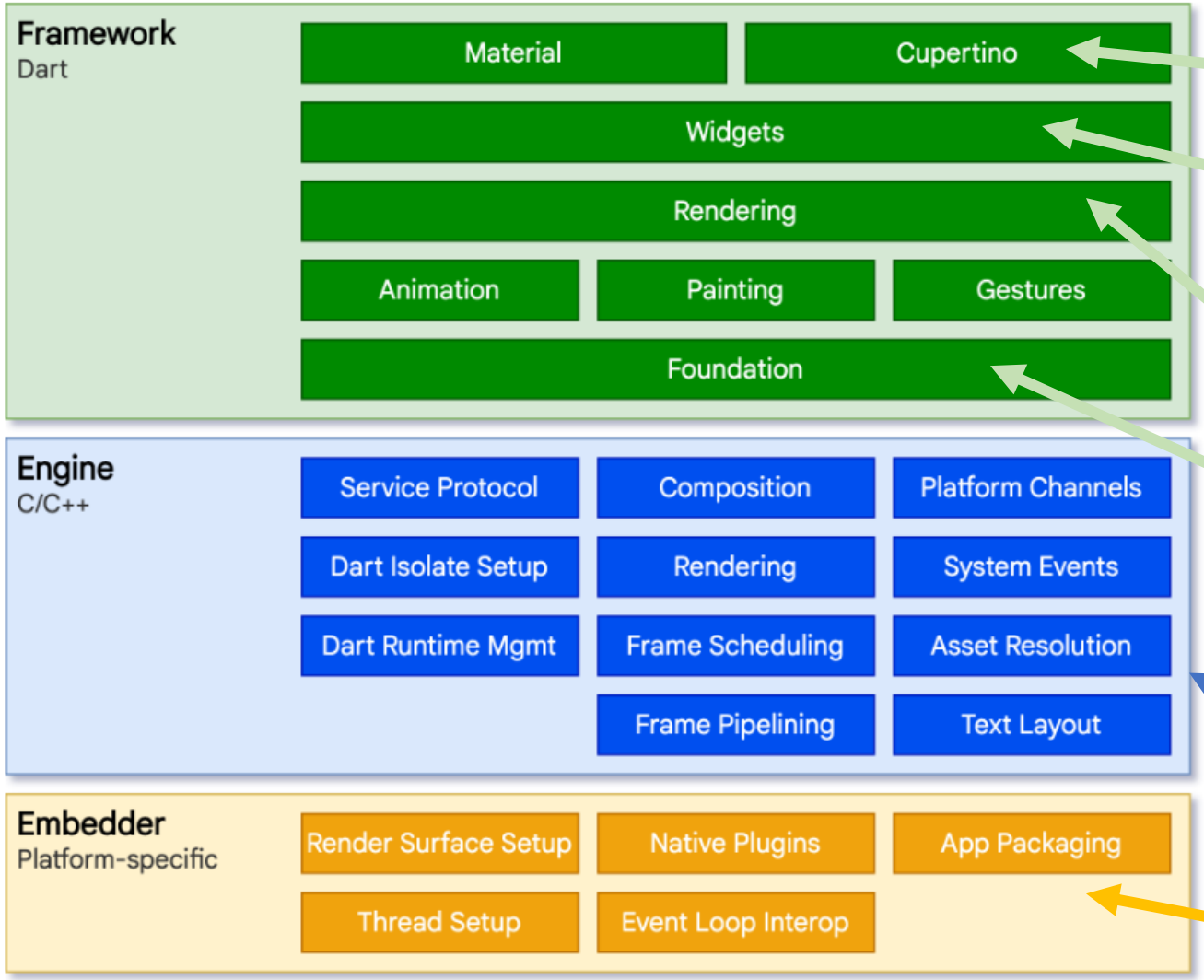
*Complete the table with the **right attributes***

```
class ExampleWidget  {

  final String title;
  final String? subtitle;
  final int count;
  final Color color;

  const ExampleWidget(
    this.title,
    {
    this.subtitle,
    required this.count,
    this.color = Colors.blue,
  });
}
```

Parameter	Positional / Named	Mandatory / Optional	Default value?
title	POSITIONAL	MANDATORY	NO
subtitle	NAMED	OPTIONAL	NO
count	NAMED	MANDATORY	NO
color	NAMED	OPTIONAL	YES

EXPLAIN

Flutter **Architecture**



Material and Cupertino libraries use widget composition to implement Material and iOS design languages.

The widgets layer offers composition abstractions, introduces reactive programming, and allows reusable class combinations.

The rendering layer manages layout by building and dynamically updating a tree of **renderable objects**.

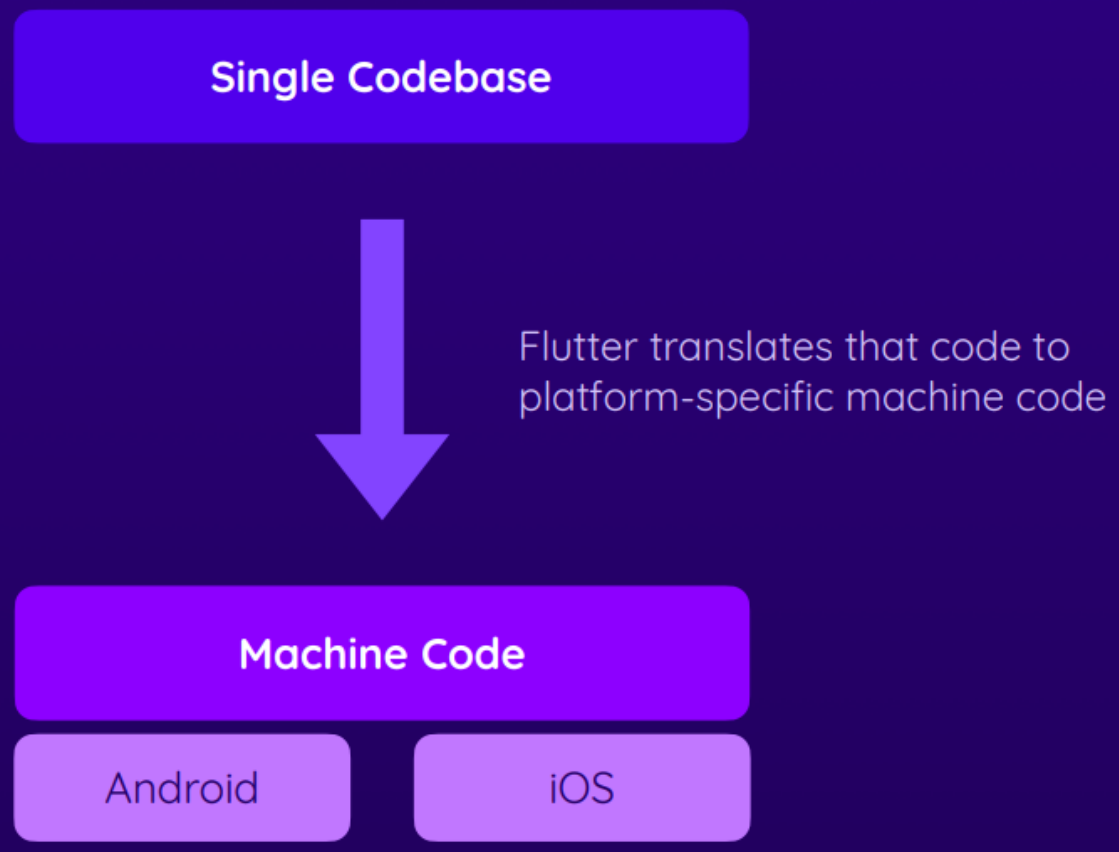
Basic classes and services (animation, painting, gestures) provide common abstractions.

Input output, rendering…. *Everything we don't need to see as developer*

Material and Cupertino libraries use widget composition to implement Material and iOS design languages.

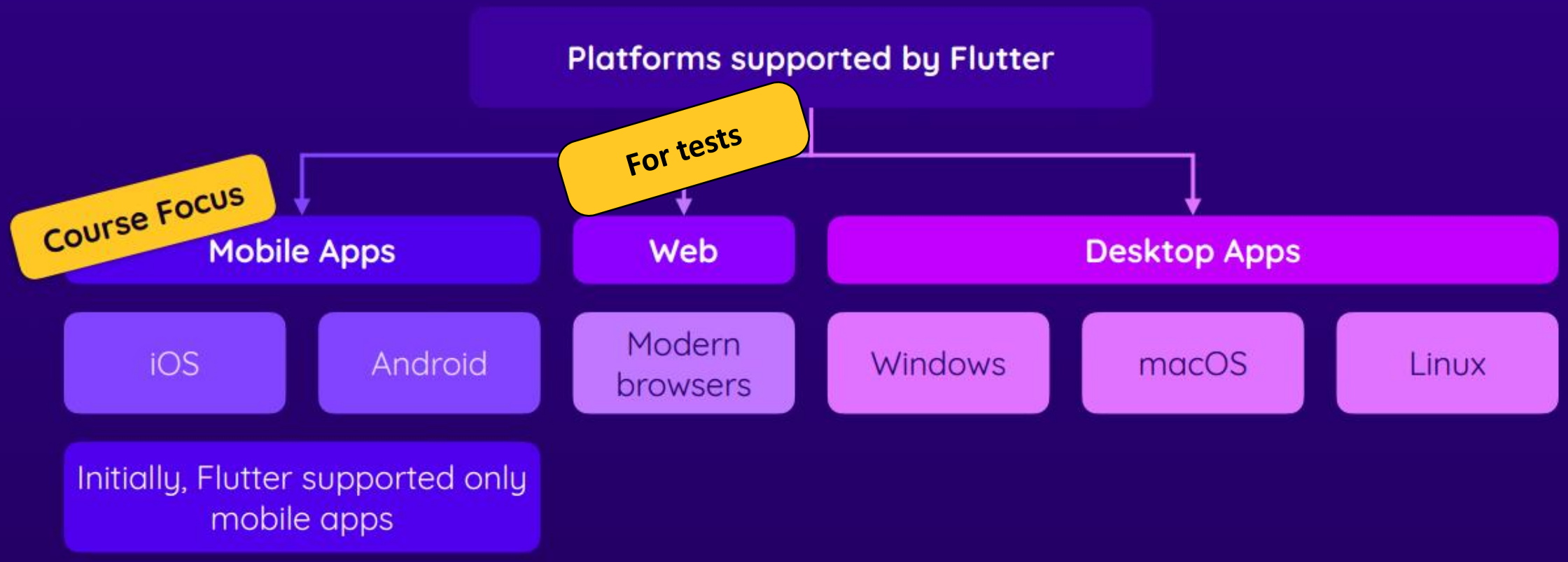
EXPLAIN

Form Flutter Code To **Platform Code**



EXPLAIN

One Codebase, **Multiple** Apps



EXPLAIN

Flutter Is Not A Programming Language!

It's a framework for building user interfaces with **Dart**

Framework

A collection of packages & utility functions you may use in your code

Dart

A programming language developed by Google

Main usage: Flutter app development

EXPLAIN

Flutter **Setup**

1 **Flutter SDK**

Flutter SDK	For managing Flutter projects
Git	Version control software, used internally by Flutter SDK

2 **Platform Tools**

Android Studio	Used by Flutter SDK & needed for Android app deployment
XCode	Used by Flutter SDK & needed for iOS app deployment

3 **Virtual Devices**

Android	Preview Flutter apps on virtual Android devices
iOS	Preview Flutter apps on virtual iOS devices

EXPLAIN

Emulator **Setup**

	on Windows	on macOS	on Linux
build **iOS Apps**	Not possible	Download & install XCode Configure XCode command-line tools Create local iOS simulator	Not possible
build **Android Apps**	Download & install Android Studio Install SDK, command-line tools & build tools Create local Android emulator		

FOR NEXT WEEK

1 week team mission

Make Flutter run on your team computers !

STEPS

- Follow the Dev Tool guide
- Install Android Studio (or XCode)
- Create a virtual Device (Android or IOS)
- Create the Flutter doctor works
- Create a Flutter project
- Run the Flutter project on Chrome
- **Run the Flutter project on the virtual Device**

RULES

THE DEVTOOL BOARD HERE

- **Work as a team**
- Team work is validated if Flutter un on every computer
- When you finished with your computer, **support your teammate**

CADT IDT

MOBILE DEVELOPEMENT

YOUR DEV TOOLS GUIDE!

Follow OUR Dev tools guide steps

```
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.24.0, on Microsoft Windows [Version 10.0.22631.4169], locale en-US)
[√] Windows Version (Installed version of Windows is version 10 or higher)
[√] Android toolchain - develop for Android devices (Android SDK version 35.0.0)
[√] Chrome - develop for the web
[X] Visual Studio - develop Windows apps
    X Visual Studio not installed; this is necessary to develop Windows apps.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[√] Android Studio (version 2024.1)
[√] IntelliJ IDEA Ultimate Edition (version 2024.1)
[√] VS Code (version 1.93.1)
[√] Connected device (3 available)
[√] Network resources
```

Flutter doctor shall pass (expect the Visual Studio)

DEMO

Let’s create the flutter workspace

```
flutter create you_first_project_name
```

Flutter workspace

- .dart_tool
- .idea
- android
- assets
- ios
- lib
- linux
- macos
- test
- web
- windows
- .gitignore
- .metadata
- analysis_options.yaml
- pubspec.yaml
- README.md
- test1.iml

assets → assets

Source code → lib

Dependencies → pubspec.yaml

*Lib Folder – **For Learning***

- lib
  - W3-S1- LEARNING - First Flutter App
  - W3-S2 - PRACTICE - Manipulate basic widgets
    - EX-1
    - EX-2
    - EX-3
    - EX-4
  - W3-S3 - LEARNING - Stateless widgets
  - W4-S1 - PRACTICE - Assets and stateless widgets
  - W4-S2 - LEARNING - Statefull widgets
  - W4-S3 - PRACTICE- Statefull widgets
  - W5-S1 - LEARNING - Layouting
  - W5-S2 - PRACTICE- Inputs, Lists
  - XX-MY-CODE

- *1 folder per practice , project*
- *Commit only **source code** and **assets** on Github*

EXPLAIN

The Flutter **Widget** Paradigm

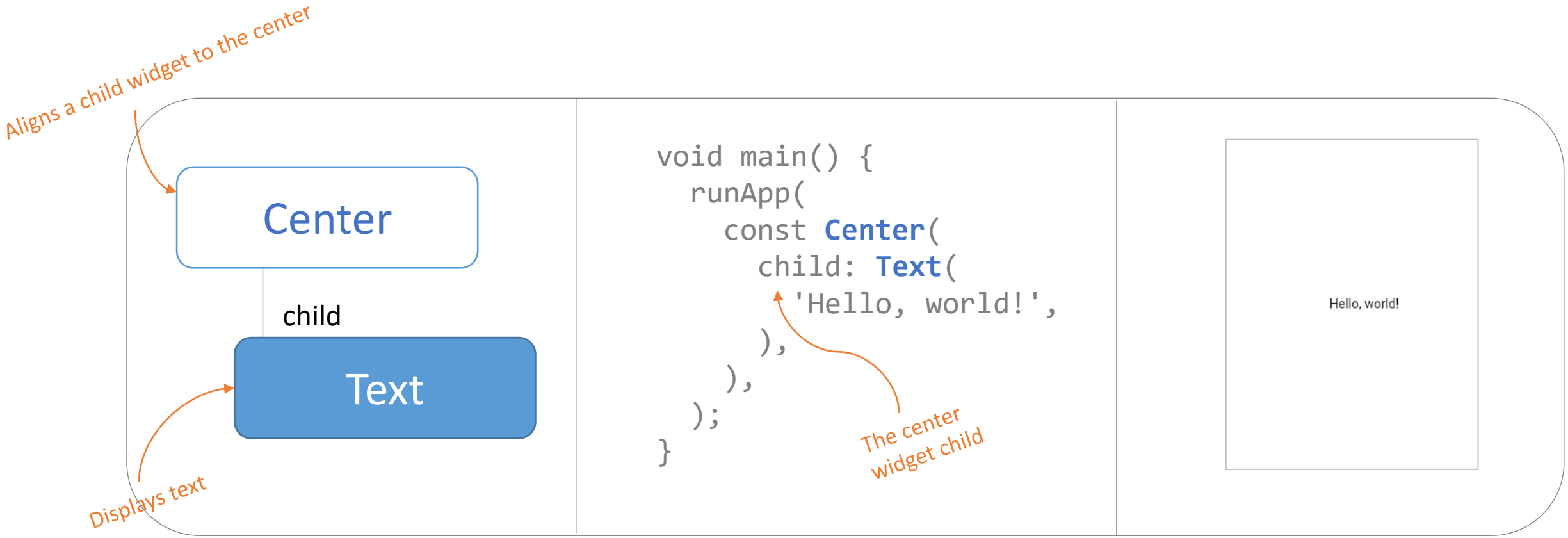
"Extreme composition instead of inheritance"

- ✓ A widget is a **self-contained, reusable component** that represents a **piece** of the user interface
- ✓ From the simplest text labels to complex layouts, **every visual element** in a Flutter app is a widget
- ✓ Just as **building blocks** come together to form structures, widgets come together to shape an app's user interface.

DEMO-1

Widgets are **nested** in a **tree**

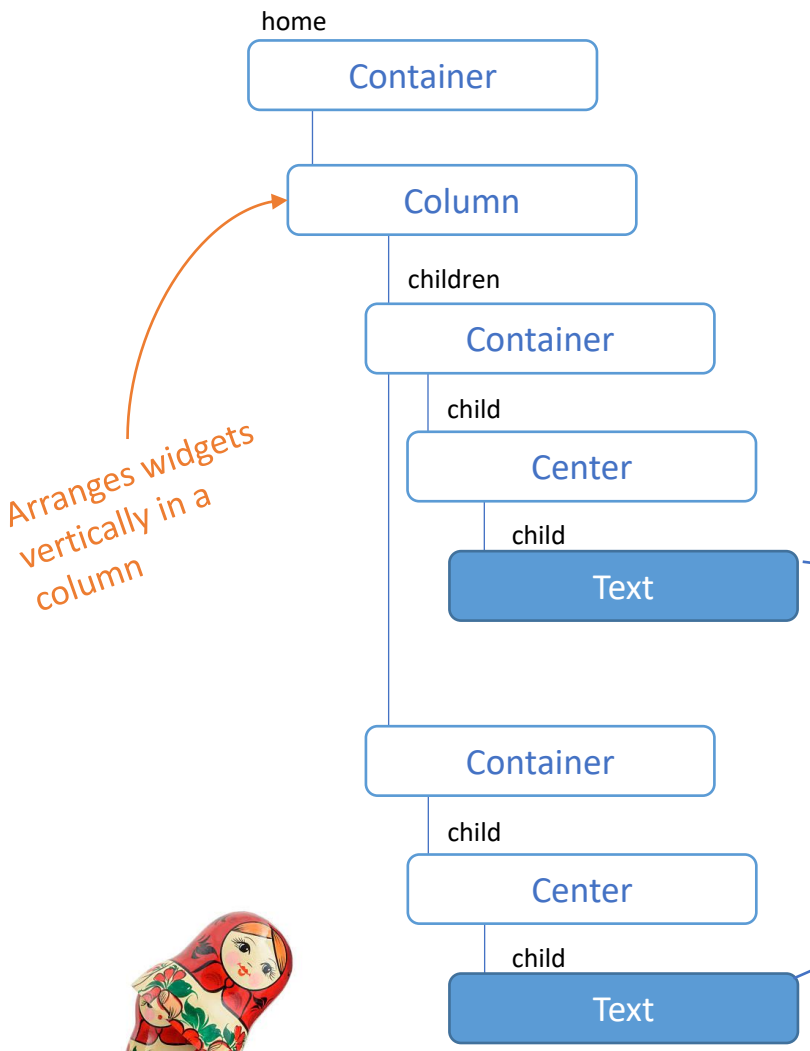
In this example, the widget tree consists of two widget : the Center widget and its child, the Text widget.



DEMO-2

Widgets are **nested** in a **tree**

In this example, the widget tree consists contains many invisible widgets to **layout** and **control** the 3 texts



ACTIVITY

10 MIN

Activity 1

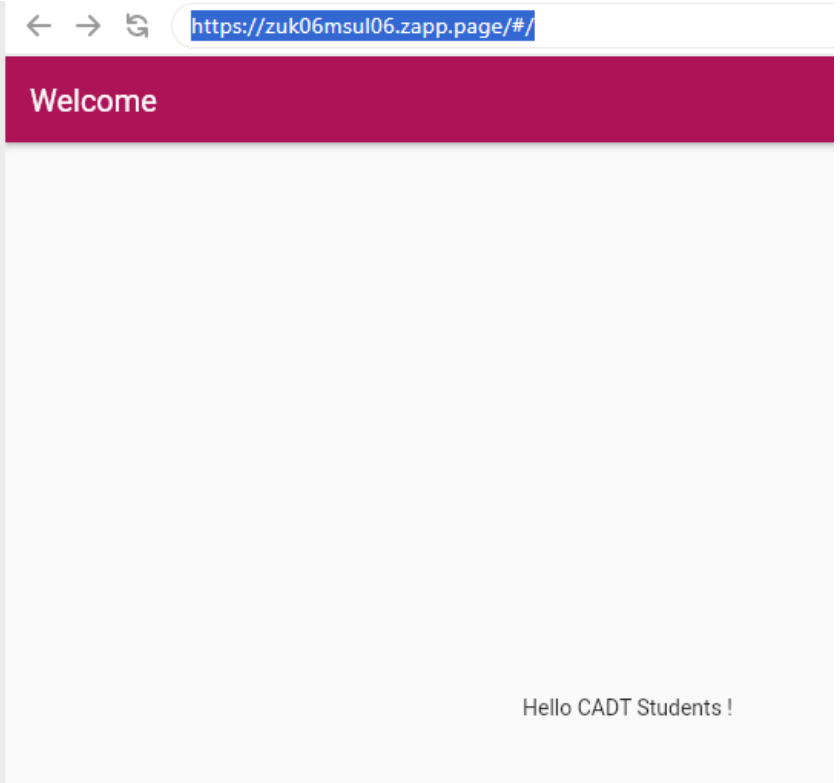
- ✓ Open this online editor
  https://zapp.run/edit/flutter-zuk06msul06?entry=lib/main.dart&file=lib/main.dart

- ✓ **Draw the Widget Tree of this app**

main.dart

lib > main.dart

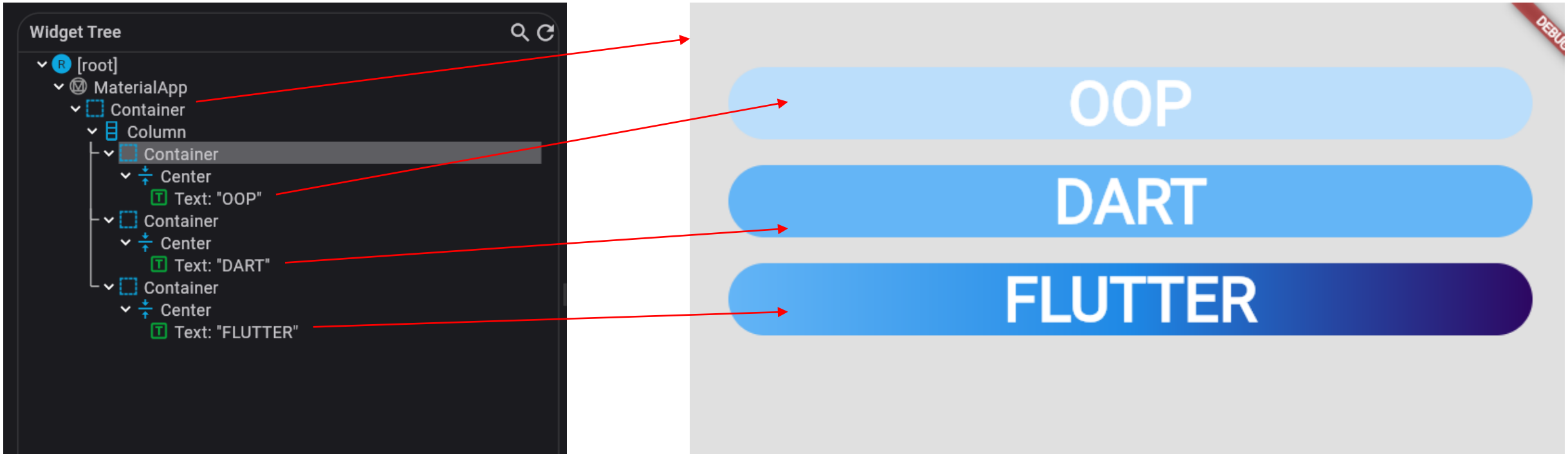
```dart
import 'package:flutter/material.dart';


Run Application
void main() {
  runApp(
    MaterialApp(
      title: 'My app', // used by the OS task switcher
      home: Scaffold(
        appBar: AppBar(
          backgroundColor: Colors.pink[800],
          title: Text("Welcome"),
        ), // AppBar
        body: const Center(
          child: Text('Hello CADT Students !'),
        ), // Center
      ), // Scaffold
      ), // MaterialApp
  );
}
```



DEMO

Use the **Flutter** inspector

The Flutter widget inspector is a powerful tool for visualizing and exploring Flutter widget trees.



More information about Flutter Inspector ? What this video !

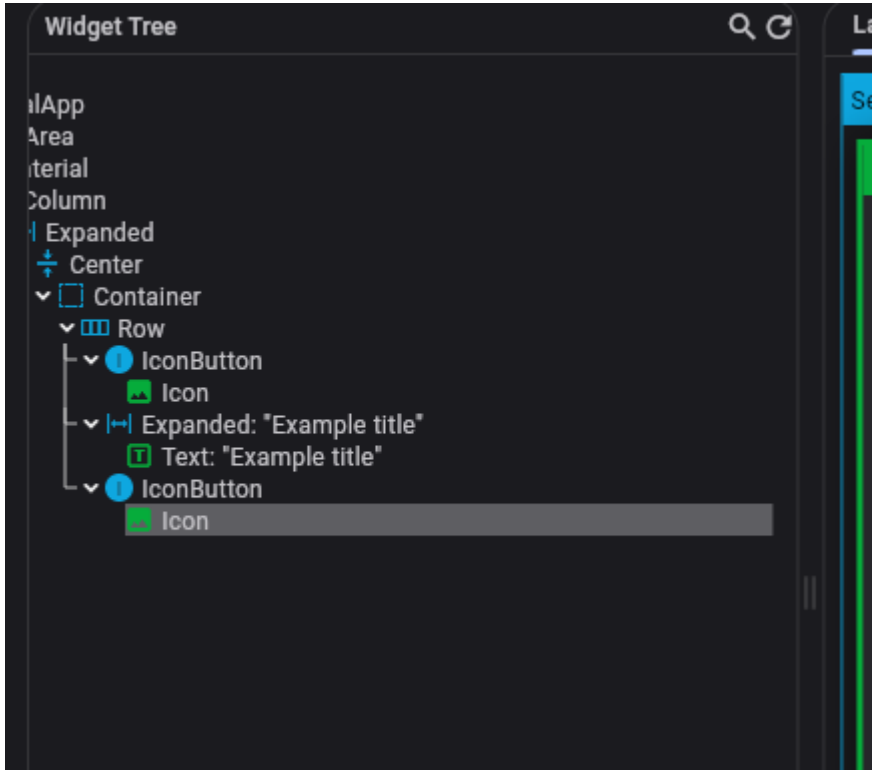
ACTIVITY

10 MIN

Activity 4

- ✓ **Copy** /activity-1 in your /lib folder
- ✓ Execute the main.dart (web device) and inspect it using **Flutter DevTools**

- ✓ **Draw the Widget Tree of this app**



EXPLAIN

The Flutter **Widgets catalog**

https://docs.flutter.dev/ui/widgets

- ✓ **Explore by yourself** the Widget catalog

Constructors

Align ({Key? key, AlignmentGeometry alignment = Alignment.center, double? widthFactor
Creates an alignment widget.
const

Properties

alignment → AlignmentGeometry
How to align the child.
final

child → Widget?
The widget below this widget in the tree.
final inherited

- ✓ Write your **own** code to test each widget

```
/// Test the Align widget
void main() {

  runApp(
     Center(
      child: Container(
        margin:   const EdgeInsets.all(10.0),
        color: Colors.amber[600],
        width: 300.0,
        height: 300.0,
        child: const Align(
          alignment: Alignment.topRight,
          child: Icon(
            Icons.favorite,
            color: Colors.pink,
            size: 24.0,
          ),
        ),
      ),
    ),
  );
}
```

EXPLAIN

Basic widgets

Widgets to know before building your first Flutter app.

https://docs.flutter.dev/ui/widgets/basics

Scaffold

Implements the basic Material Design visual layout structure. This class provides APIs for showing drawers, snack bars, and bottom sheets.

Text

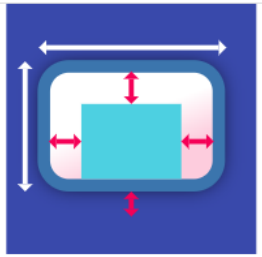
A run of text with a single style.

Row

Layout a list of child widgets in the horizontal direction.

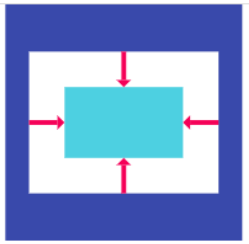
Column

Layout a list of child widgets in the vertical direction.



Container

A convenience widget that combines common painting, positioning, and sizing widgets.



Center

Alignment block that centers its child within itself.

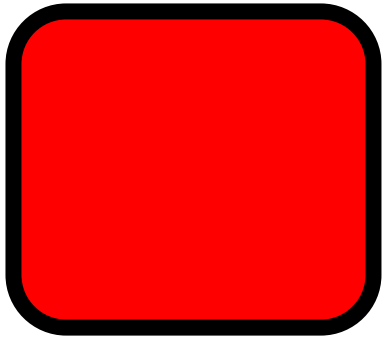
EXPLAIN

The Flutter **configuration objects**

Flutters provides a set of objects to configure widgets

TextStyle	EdgeInsets	**BoxDecoration**	TextStyle
Styles text appearance	Sets padding or margin	Styles a container's background, border, etc.	Styles text appearance

```
Container(
  decoration: BoxDecoration(
    color: const Colors.red[100]
    border: Border.all(width: 8),
    borderRadius: BorderRadius.circular(12),
  ),
)
```

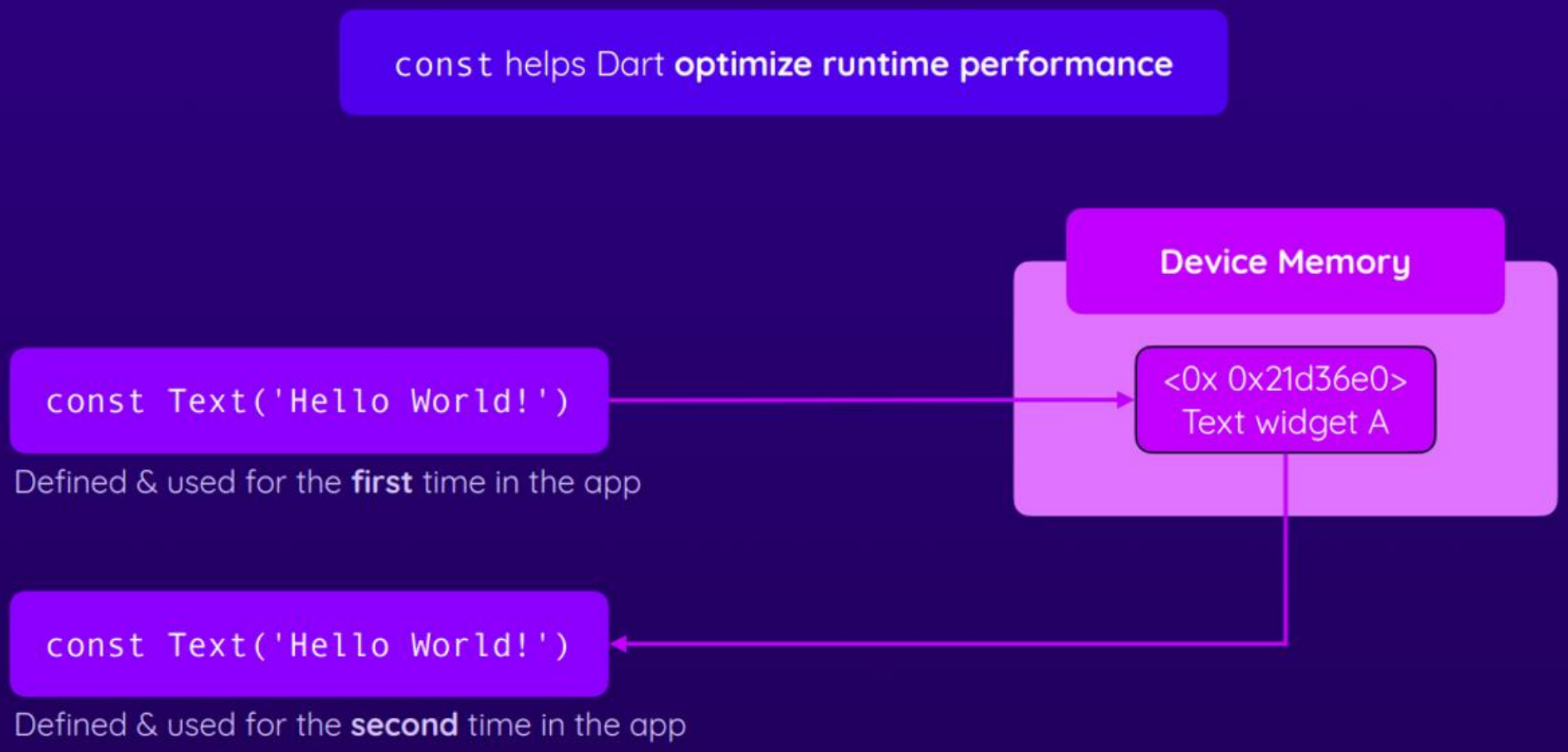


To style de container, add a BoxDecoration object

https://api.flutter.dev/flutter/painting/painting-library.html

EXPLAIN

Why should `const` be used as much as possible ?

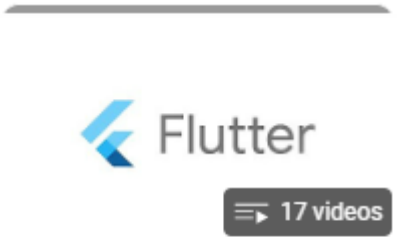



Flutter
in Focus

Resources – *Flutter Officials*

https://www.youtube.com/@flutterdev/playlists

BASICS



Begin learning Flutter

View full playlist

Flutter in Focus

View full playlist

Flutter Widget of the Week

View full playlist

ADVANCED

Learning to Fly

View full playlist

The Boring Flutter Development Show

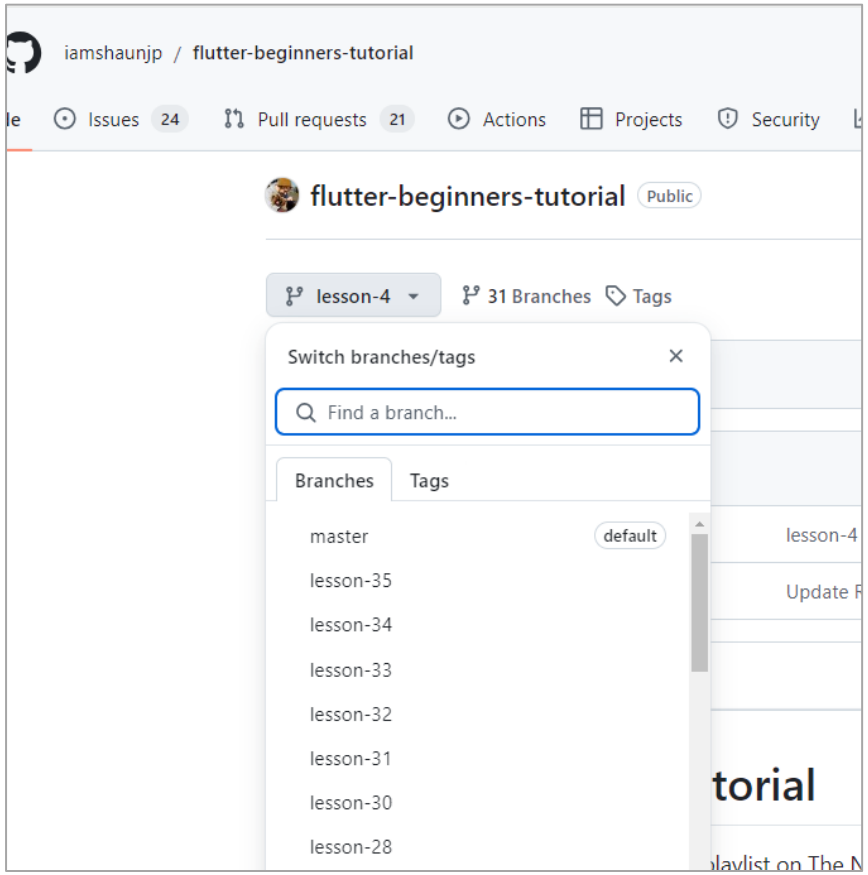
Making Animations in Flutter

View full playlist

Resources – *Net Ninja Tutorial*

[Tutorial videos]

[Tutorial correction per lesson]



TODO AFTER

Before next session

1 Introduction to Flutter

Read the Flutter doc
Understand the flutter inspector

Net NINJA Tutorial
Net NINJA code correction

Understand the anatomy of Flutter widgets

2 Read Tool Dev guide

OUR DEV TOOL GUIDE

3 Perform your team mission

✓ Make Flutter run on **your team computers**
Using an Android virtual device

THE DEVTOOL BOARD HERE

4 Start your first app

✓ Create a first app with the widgets we have presented

- `scaffold`
- `text`
- `row`
- `column`
- `container`
- `center`