To do before next session

1 - Watch **video**

https://www.youtube.com/watch?v=wE7khGHVkYY

2 - Answer to following questions

- What are the benefits of State widgets ?

MOBILE DEVELOPMENT

W3-S3 – Stateless Widgets

CADT IDT

Course Objectives

- ✓ Compose **widgets, class** and **attributes**
- ✓ Understand the OOP inheritance concepts (**super, abstract, overwrite, extends**)
- ✓ Understand the syntax of a **StatelessWidget** and the **build**() methods
- ✓ Create a re-usable **StatelessWidget** and use it multiple times
- ✓ Separate custom widgets in **different files**

REVIEW

Where are you in your **last week tasks ?**

- ✓ Make Flutter run on **every computers**
  Using an Android / IOS virtual device

- ✓ Create a first app with the widgets we have presented
  - `scaffold`
  - `text`
  - `row`
  - `column`
  - `container`
  - `center`

DEVTOOL BOARD

EXPLAIN

Const Widgets

*In Flutter, the const keyword is used to create **compile-time constant** values.*

```
const Text( 'This is static text!’);
```

…So the widget cannot be changed at runtime.

..the string is then also marked as const…

The Text widget is marked as const…

When a ***const*** widget is used, Flutter can **skip the process of rebuilding the** widget if the properties have not changed, which can lead to faster app performance.

MORE INFO

EXPLAIN

Const Widgets

*When tagging a widget as **const**,*
all of widget properties and children <u>must also be constants</u>.

```
Text const text = Text('Hello');
```

This widget can be const
as its properties are constant

```
String name = "ronan";

Text ~~const~~  text = Text('Hello $name');
```

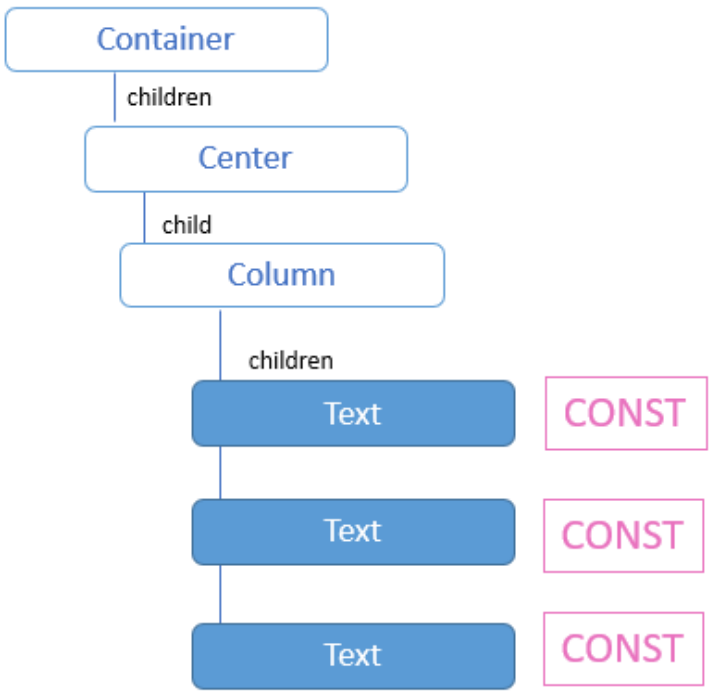
This widget cannot be const
as it's the string depend on a variable

? In which cases a widget **should not be marked as const**?

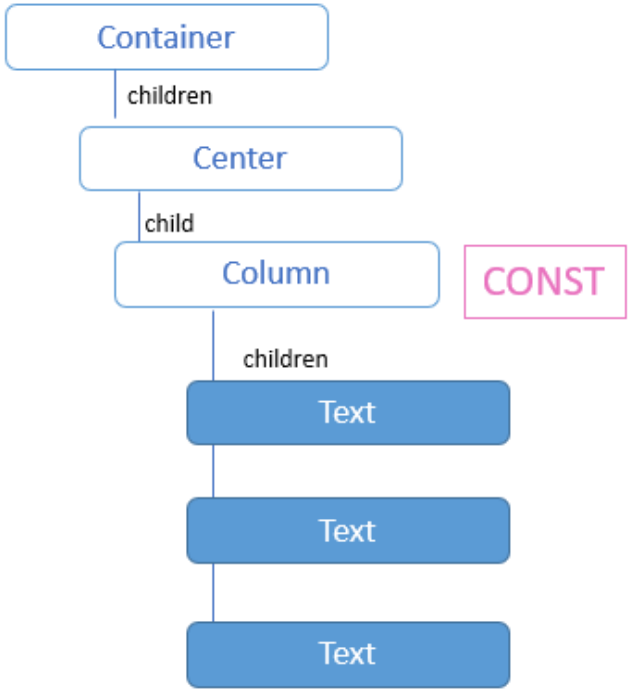
EXPLAIN

Const Widgets

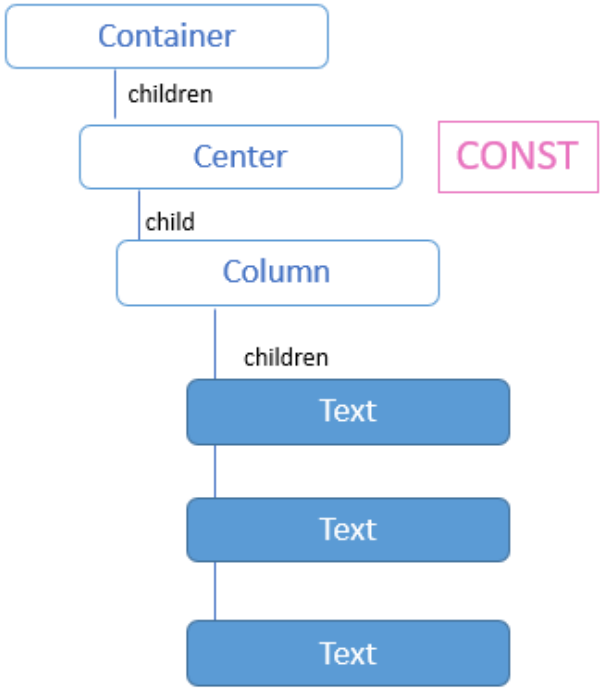
Const should be tagged as much as possible on the top element



All Text are const…



….So the const can be moved up to Column



….and to Center

Container is not a const widget
So it cannot be tagged as const

ACTIVITY

Activity 1

https://zapp.run/edit/flutter-zeue06saeuf0?entry=lib/main.dart&file=lib/main.dart

✓ Identify the 2 widgets that can be **const**

```
void main() {
  runApp(MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Styled Static Widgets'),
        ), // AppBar
        body:  Center(
          child: Container(
            width: 200,
            height: 200,

            child: Center(
              child: Text(
                'Styled Container',
                style: TextStyle(color: Colors.white, fontSize: 20),
              ), // Text
            ), // Center
          ), // Container
        ), // Center
      ), // Scaffold
    )); // MaterialApp
}
```

Activity 1

https://zapp.run/edit/flutter-zeue06saeuf0?entry=lib/main.dart&file=lib/main.dart

✓ Identify the 2 widgets that can be **const**

```
void main() {
  runApp(MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Styled Static Widgets'),
        ), // AppBar
        body:  Center(
          child: Container(
            width: 200,
            height: 200,

            child: Center(
              child: Text(
                'Styled Container',
                style: TextStyle(color: Colors.white, fontSize: 20),
              ), // Text
            ), // Center
          ), // Container
        ), // Center
      ), // Scaffold
    )); // MaterialApp
}
```

CONST

CONST

ACTIVITY

Activity 2

Identify the types of the 6 constructor arguments

```
MyWidget(a, [b], [c = 10], { d, required e, f = 10 }) { … }
```

Parameter	Positional / Named	Mandatory / Optional	Default value?
a			
b			
c			
d			
e			
f			

ACTIVITY

Activity 2

Identify the types of the 6 constructor arguments

```
MyWidget(a, [b], [c = 10], { d, required e, f = 10 }) { … }
```

Parameter	Positional / Named	Mandatory / Optional	Default value?
a	positional	mandatory	
b	positional	optional	
c	positional	optional	Default value
d	named	optional	
e	named	mandatory	
f	named	optional	Default value

QUIZ

Which of the following instantiations are valid? Select all that apply:

```
MyWidget(a, [b], [c = 10], { d, required e, f = 10 }) { … }
```

```
1.MyWidget(1, 2, 3, e: 5)
2.MyWidget(1, e: 5)
3.MyWidget(1, 2, 3, d: 4, f: 6)
4.MyWidget(1, e: 5, d: 6)
5.MyWidget(a: 1, e: 5)
```

QUIZ

Which of the following instantiations are valid? Select all that apply:

```
MyWidget(a, [b], [c = 10], { d, required e, f = 10 }) { … }
```

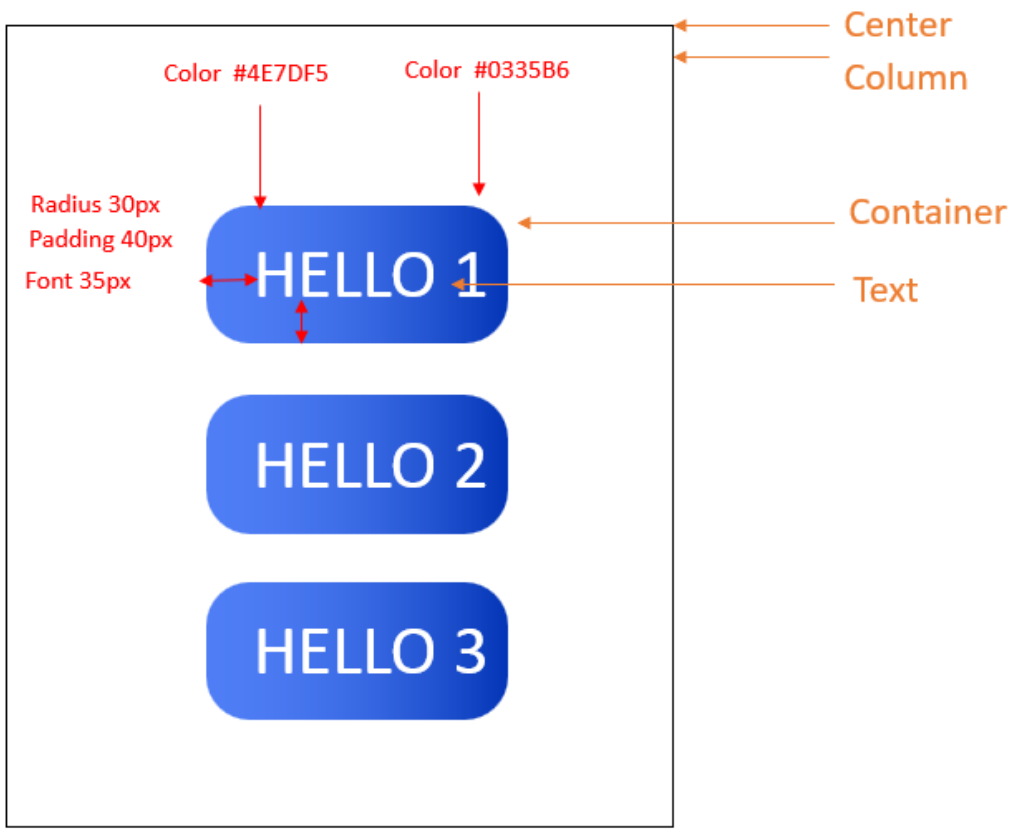
1. `MyWidget(1, 2, 3, e: 5)`
2. `MyWidget(1, e: 5)`
3. `MyWidget(1, 2, 3, d: 4, f: 6)` e is required
4. `MyWidget(1, e: 5, d: 6)`
5. `MyWidget(a: 1, e: 5)` A is positional, cannot be named

ACTIVITY

Activity 3

Composition of widgets

- ✓ Create in 3 steps a **composition of widgets**
- ✓ Use **Flutter SDK** to configure your widgets and classes




```
style: TextStyle(
    col
    dec
// Materi
```

```
(const) TextStyle TextStyle({
  bool inherit = true,
  Color? color,
  Color? backgroundColor,
  double? fontSize,
  FontWeight? fontWeight,
  FontStyle? fontStyle,
  double? letterSpacing,
  double? wordSpacing,
  TextBaseline? textBaseline,
  double? height,
  TextLeadingDistribution? lead
  Locale? locale,
```

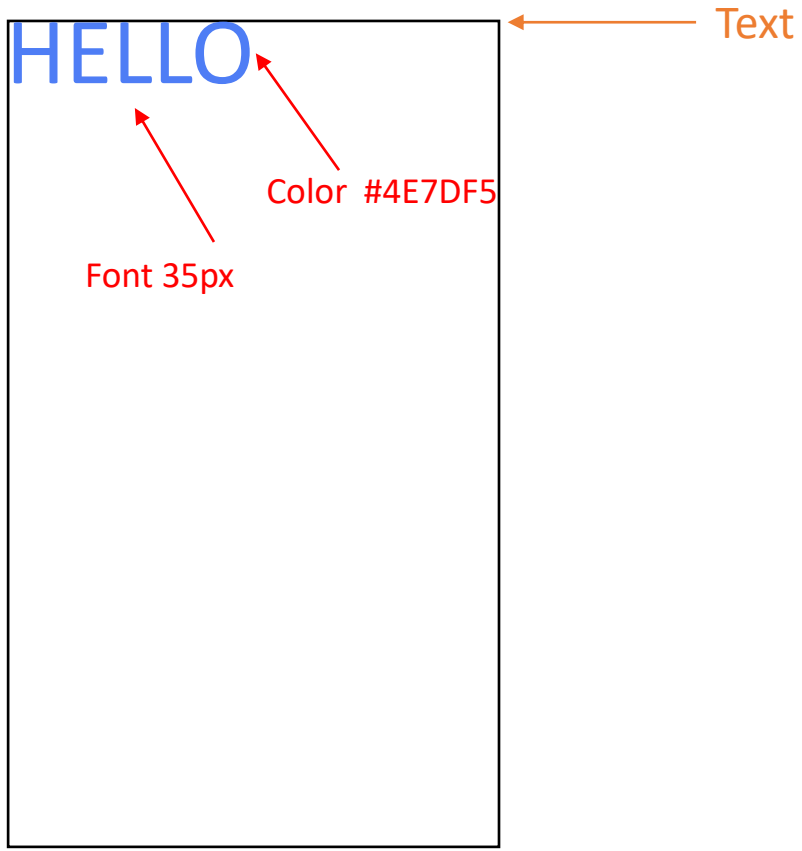
ACTIVITY

Activity 3

Step 1 – Simple Text

Use the following **widgets, properties, classes** :

```
Text
    style: TextStyle
                color : Color
                decoration: BoxDecoration
```



Activity 3.1

EXPLANATIONS

```
void main() {
  runApp(const MaterialApp(
    home: Text("HELLO",
        style: TextStyle(
            color: Color(0xff4E7DF5),
            decoration: TextDecoration.none)), // TextStyle // Text
  )); // MaterialApp
}
```

Color constructor take an hexadecimal number
Following the syntax: AA RR GG BB *(alpha, red, green , blue)*

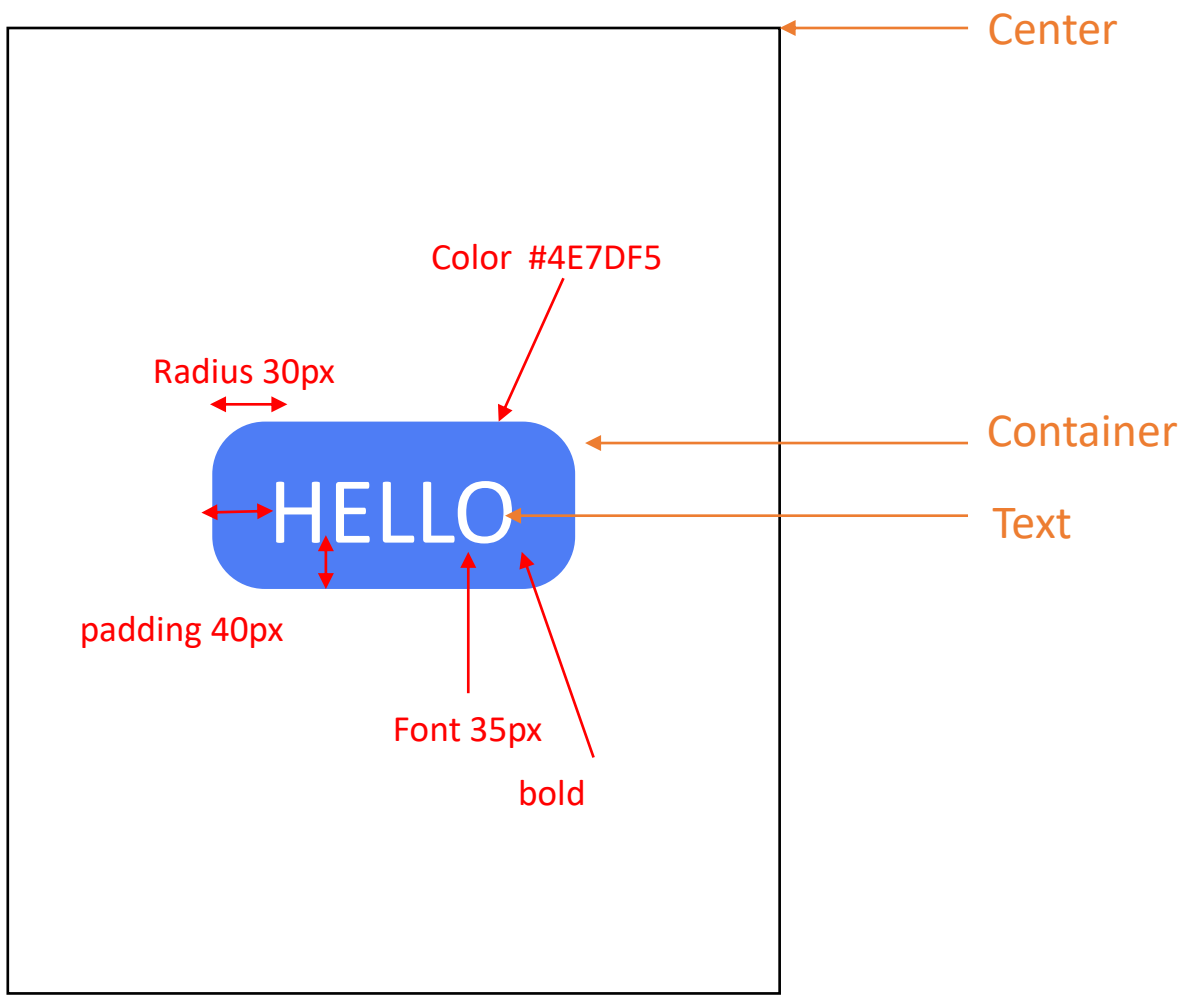
ACTIVITY

Activity 3

Step 2 – Simple Text with decoration, padding

Use the following **widgets, properties, classes** :

```
Center
   child: Container
       padding: EdgeInsets

       decoration: BoxDecoration
             color: Color
             borderRadius: BorderRadius

       child:  Text
             style: TextStyle
```



Activity 3.2

EXPLANATIONS

```
void main() {
  runApp(MaterialApp(
    home: Scaffold(
      body: Center(
        child: Container(
          padding: const EdgeInsets.all(40),
          decoration: BoxDecoration(
            color: const Color(0xff4E7DF5),
            borderRadius: BorderRadius.circular(30)
          ), // BoxDecoration
          child: const Text("HELLO", style: TextStyle(
            fontSize: 35.0,
            fontWeight: FontWeight.w700,
            color: Colors.white,
            decoration: TextDecoration.none), // TextStyle
            ), // Text
        ), // Container
      ), // Center
    ), // Scaffold
  )); // MaterialApp
}
```

Circular is a named constructor

Static values for font weights

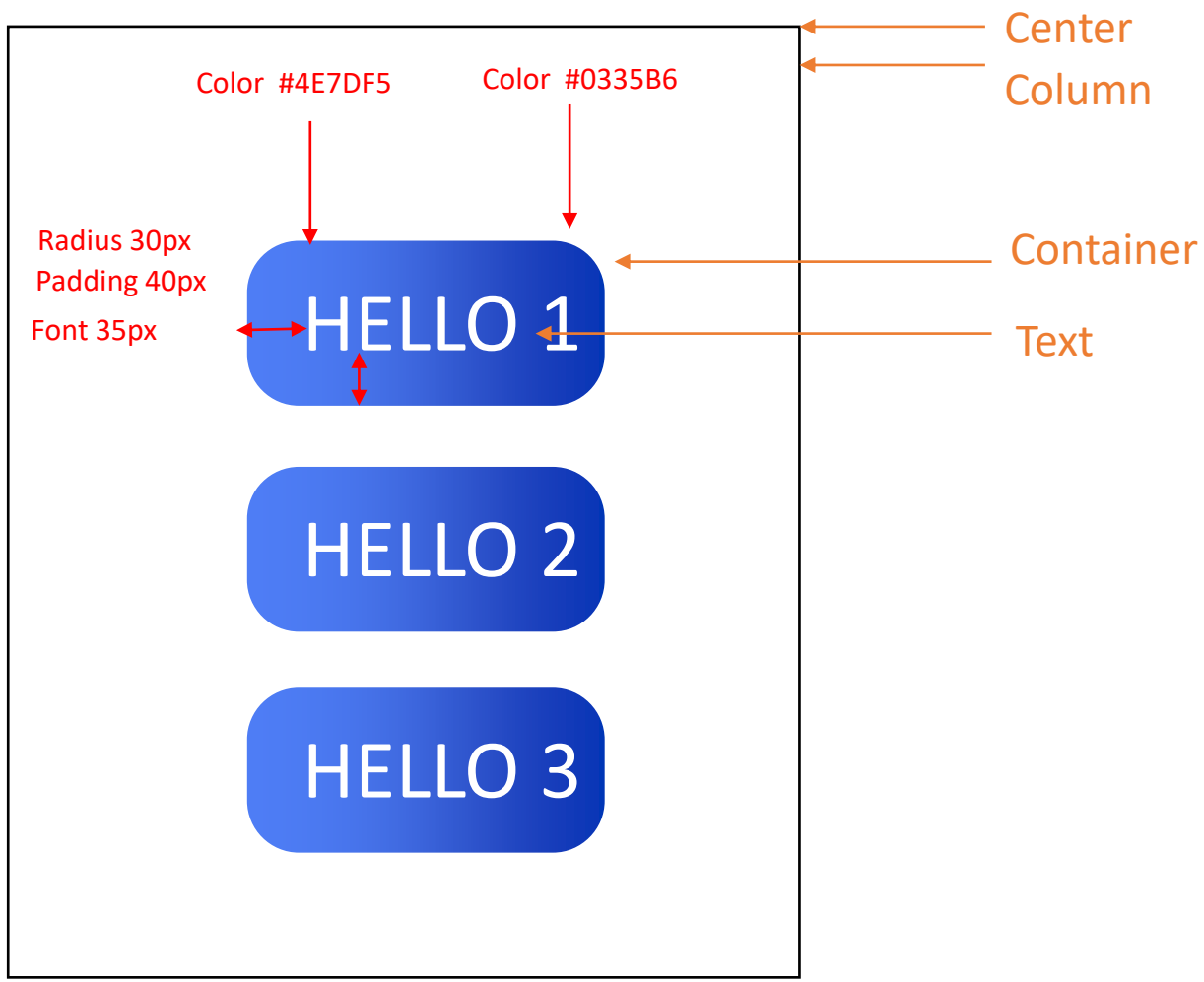
ACTIVITY

Activity 3

Step 3 – Buttons with gradient

Use the following **widgets, properties, classes** :

```
Column
    children: Widget[]

Container
    decoration: BoxDecoration
        gradient: LinearGradient
```



Activity 3.3

EXPLANATIONS

- What a lost of **duplicated** code !
- Any idea to make it **better**?

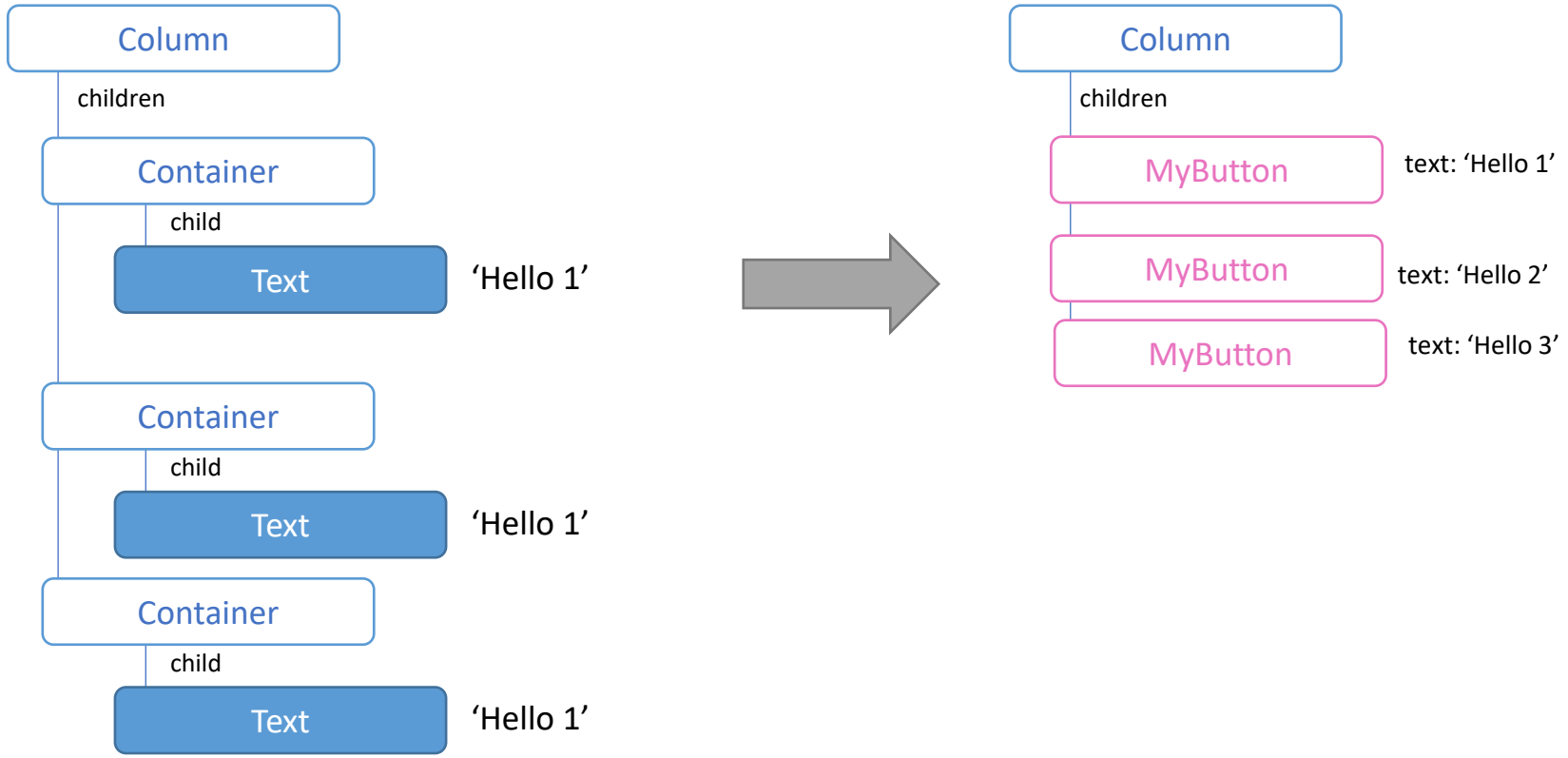
```
body: Center(
  child: Column(
    children: [
      Container(
        margin: const EdgeInsets.all(20),
        padding: const EdgeInsets.all(40),
        decoration: BoxDecoration(
            gradient: const LinearGradient(colors: [Color(0xFF4E7DF5), Color(0xFF0335B6)]),
            borderRadius: BorderRadius.circular(30)), // BoxDecoration
        child: const Text(
          "HELLO 1",
          style: TextStyle(
              fontSize: 35.0,
              fontWeight: FontWeight.w700,
              color: Colors.white,
              decoration: TextDecoration.none), // TextStyle
        ), // Text
      ), // Container
      Container(
        margin: const EdgeInsets.all(20),
        padding: const EdgeInsets.all(40),
        decoration: BoxDecoration(
            gradient: const LinearGradient(colors: [Color(0xFF4E7DF5), Color(0xFF0335B6)]),
            borderRadius: BorderRadius.circular(30)), // BoxDecoration
        child: const Text(
          "HELLO 2",
          style: TextStyle(
              fontSize: 35.0,
              fontWeight: FontWeight.w700,
              color: Colors.white,
              decoration: TextDecoration.none), // TextStyle
        ), // Text
      ), // Container
      Container(
        margin: const EdgeInsets.all(20),
        padding: const EdgeInsets.all(40),
        decoration: BoxDecoration(
            gradient: const LinearGradient(colors: [Color(0xFF4E7DF5), Color(0xFF0335B6)]),
            borderRadius: BorderRadius.circular(30)), // BoxDecoration
        child: const Text(
          "HELLO 3",
          style: TextStyle(
              fontSize: 35.0,
              fontWeight: FontWeight.w700,
              color: Colors.white,
              decoration: TextDecoration.none), // TextStyle
        ), // Text
      ), // Container
    ],
```

EXPLAIN

Reusable Widgets is the cornerstone of efficient and effective Flutter apps

MORE INFOS

- ✓ Provides **code reusability** and organization.
- ✓ **Reduces** the overall **codebase** leading to easier maintenance.
- ✓ Ensures **consistency** across your application.



ACTIVITY

Dart OOP - **Inheritance**

Look at the code, answer the questions

Q1 - What does `abstract` do?

Q2 – *Why this method has no body?*

Q3 – What does `extends` do?

Q4 – What does `super.id` do?

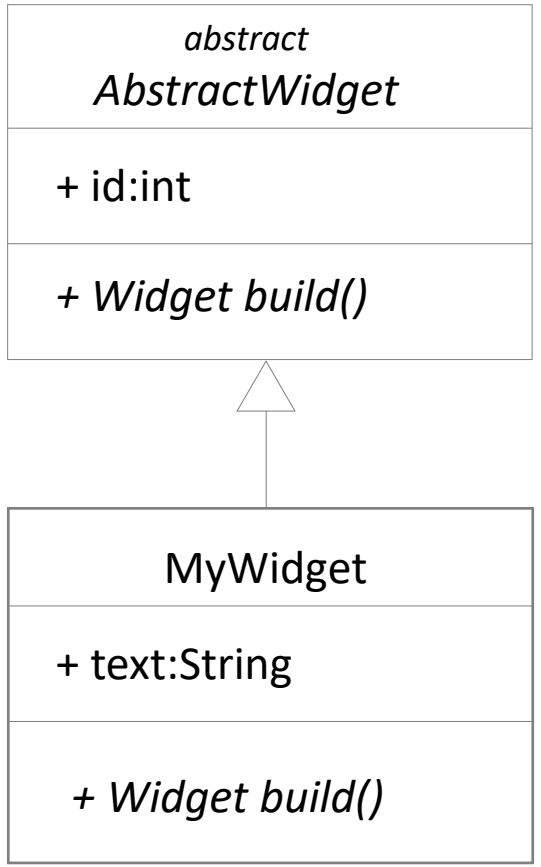
Q5 – What does `@override` do?

```
abstract class AbstractWidget {

  AbstractWidget(this.id);

  final int id;

  Widget build();
}


class MyWidget extends AbstractWidget{

  MyWidget(super.id, this.text);

  final String text;

  @override
  Widget build() {
    return Text(text);
  }
}
```

EXPLAIN

Dart OOP - **Inheritance**

```
abstract class AbstractWidget {

  AbstractWidget(this.id);

  final int id;

  Widget build();
}


class MyWidget extends AbstractWidget{

  MyWidget(super.id, this.text);

  final String text;

  @override
  Widget build() {
    return Text(text);
  }
}
```



This class cannot be instantiated

This method must be implemented by sub classes

MyWidget inherits form The parent class

We overwrite the build method

- Super refers to the parent class instance
- This refers to the current class instance

Stateless widget

Main App

```
Column(
  children: [
    Product('Cup'),
    Product('Book'),
    Product('Guitar'),
  ]
);
```

"Product" is a re-usable custom widget

Custom "Product" Widget

```
Card(
  child: Column(
    Text(title)
  ),
);
```

EXPLAIN

How to create a **stateless widget**?

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

  @override
  Widget build(BuildContext context) {
      ...
  }
}
```

1 – Extends StatelessWidget class

2 – Implement build method

3 – We will use this context later

The build method of a stateless widget is called when the widget need to be re-build :
- When the

ACTIVITY

Activity 3

Step 4– Create a custom widget

✓ Refactor the previous code with a **stateless widget**, taking 3 argument as follows:

```
GradiantButton  (stateless)
    text: String
    start: Color
    end: Color
```

```
children: [
  GradiantButton("hello 1", start: Colors.blue, end: Colors.red),
  GradiantButton("hello 1", start: Colors.blue, end: Colors.red),
  GradiantButton("hello 2", start: Colors.blue, end: Colors.red),
],
```

Example of widget instantiation

EXPLAIN

Separate widgets in different files

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

This file is located on the same folder

```
Run | Debug | Profile
void main() {
  runApp(const MaterialApp(
    home: Scaffold(
      body: Center(
        child: Column(
          children: [
            GradiantButton("hello 1", start: Colors.blue, end: Colors.red),
            GradiantButton("hello 1", start: Colors.blue, end: Colors.red),
            GradiantButton("hello 2", start: Colors.blue, end: Colors.red),
          ],
        ), // Column
      ), // Center
    ), // Scaffold
  )); // MaterialApp
}
```

What we have learnt today

- ✓ Understand the **importance of const** for Flutter runtime optimization
- ✓ Be able to define **required, optional, named or position and default** arguments
- ✓ Be able to **compose widgets, class and attributes**

- ✓ Understand the OOP inheritance concepts (**super, abstract, overwrite, extends**)

- ✓ Understand the syntax of a **StatelessWidget** and the **build**() methods
- ✓ Create a re-usable **StatelessWidget** and use it multiple times

- ✓ Separate custom widgets in **different files**

TODO AFTER

For next week

1 Review the theory

✓ Create **re usable widgets**
https://www.youtube.com/watch?v=ePXg8rqzl54

✓ Create a **stateless widget**
https://api.flutter.dev/flutter/widgets/StatelessWidget-class.html

✓ Use **custom font**
https://www.classcentral.com/classroom/youtube-flutter-tutorial-for-beginners-45851/60c82bddab9f2

✓ Use **assets images**
https://www.classcentral.com/classroom/youtube-flutter-tutorial-for-beginners-45851/60c82bddaba0a

3 Code your stateless widgets

✓ Create multiple stateless widgets
✓ Use **font** and **assets images**