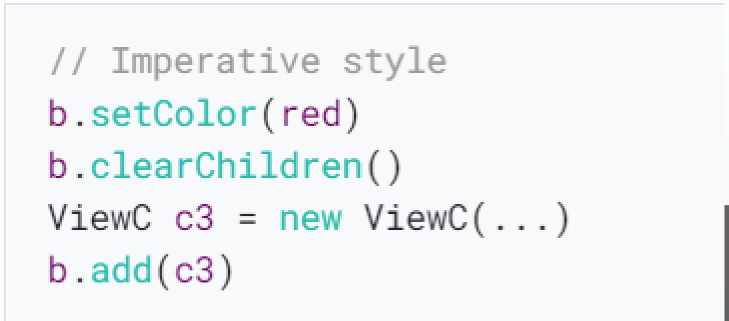
LUT
University

Flutter and widgets

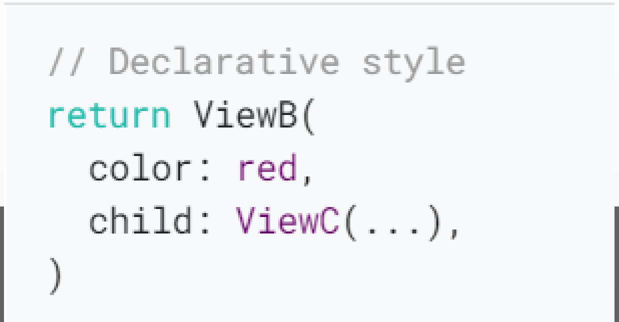
It’s all widgets

Widgets describe what their view should look like, given current configuration and state.

- Widgets have properties and children (which are other widgets)
- When program state changes, the framework replaces changed widgets with new ones
- Declarative UI, not imperative



```
// Imperative style
b.setColor(red)
b.clearChildren()
ViewC c3 = new ViewC(...)
b.add(c3)
```



```
// Declarative style
return ViewB(
  color: red,
  child: ViewC(...),
)
```

What kind of widgets exists?

- Content: Text, Image
- Formatting: Row, Column, Stack
- Constraints: Container
- Elements: Card

Anatomy of a Flutter program (minimal example)

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

void main() {
  runApp(
    Center(
      child: Text(
        'Hello, world!',
        textDirection: TextDirection.ltr,
      ),
    ),
  );
}
```

Breaking down the example

- Import: loading libraries (material elements in this case)
- Void main(): Starting point
- runApp(): Always starts a Flutter program, gets one Widget
- Center(): a Widget, contains one Text
  - textDirection as a property (reading from left to right)

Note that even here there are two widgets already inside each other!

Custom widget, minimal example

```
class LunchCard extends StatelessWidget {
  ...

  @override
  Widget build(BuildContext context) {
    return Card(
        color: Theme.of(context).accentColor,
        child: ListTile(
          // See available icons at https://material.io/resources/icons/
          leading: Icon(Icons.fastfood),
          title: Text(”Student”),
          subtitle: Text("Student Union House"),
        ));
  }
}
```

Basic demo

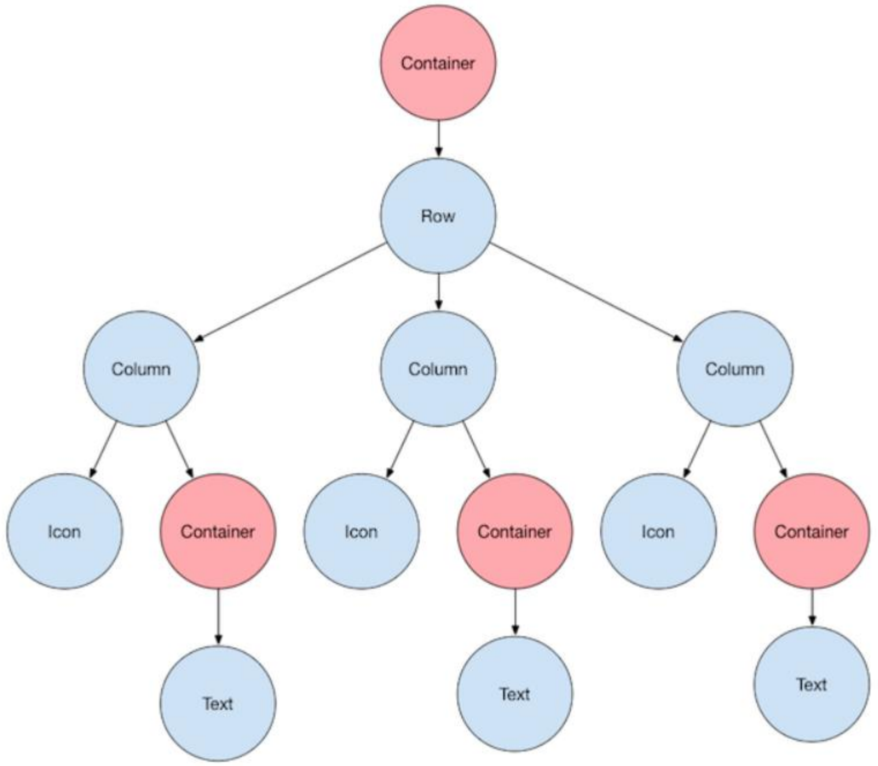
Further reading

Intro to Widgets
https://flutter.dev/docs/development/ui/widgets-intro

All available Material style widgets
https://flutter.dev/docs/development/ui/widgets/material

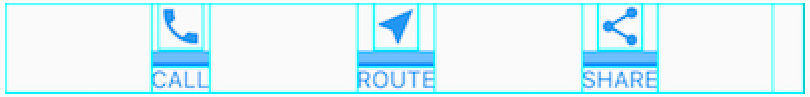
Flutter layouts

It’s all Widgets (even layouts)



It’s all Widgets (even layouts)

- Use Columns, Rows, and boxes
- Compose widgets inside each other: e.g. columns can contain cards, which can contain text, images, etc.
- Set color, font, etc. as widget properties
- Use containers to add margins, padding, and other restrictions or fine-tuning



Layout, example

```
class CardColumn extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // https://api.flutter.dev/flutter/widgets/Column-class.html
    return Column(mainAxisSize: MainAxisSize.min, children:
<Widget>[
      LunchCard('Kebab', 'Available at student union'),
      LunchCard('Soup', 'Available at main building'),
      LunchCard('Salad', 'Available at student union'),
    ]);
  }
}
```

Theming

- Create a theme class and attach it to MaterialApp at start
- Widgets can access the theme and read defaults

https://api.flutter.dev/flutter/material/Theme-class.html

Theming, example

```
 Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
          brightness: Brightness.light,
          primaryColor: Colors.lightGreen[600],
          accentColor: Colors.amber[800],


...

  @override
  Widget build(BuildContext context) {
    return Card(
        color: Colors.blueGrey[100],
```

Further reading

Further tutorial on creating layouts
https://flutter.dev/docs/development/ui/layout

Further tutorial on setting themes
https://flutter.dev/docs/cookbook/design/themes

Material layout widgets
https://flutter.dev/docs/development/ui/widgets/material#Layout

Layout widgets:
https://flutter.dev/docs/development/ui/widgets/layout

Extra credit

Adaptive layouts – implementing responsive design
https://flutter.dev/docs/development/ui/layout/adaptive-responsive

Programming this is outside the scope of the course, but if you start creating your own program, this is essential reading.

(however: designing responsive apps is in scope – details of programming will be handled later, e.g. in web applications)

Layouts demo