UNIVERSITY OF PADOVA
DEPARTMENT OF INFORMATION ENGINEERING

Biomedical Wearable Technologies
for Healthcare and Wellbeing

Understanding UI

A.Y. 2023-2024

Giacomo Cappon

DEPARTMENT OF
INFORMATION
ENGINEERING
UNIVERSITY OF PADOVA

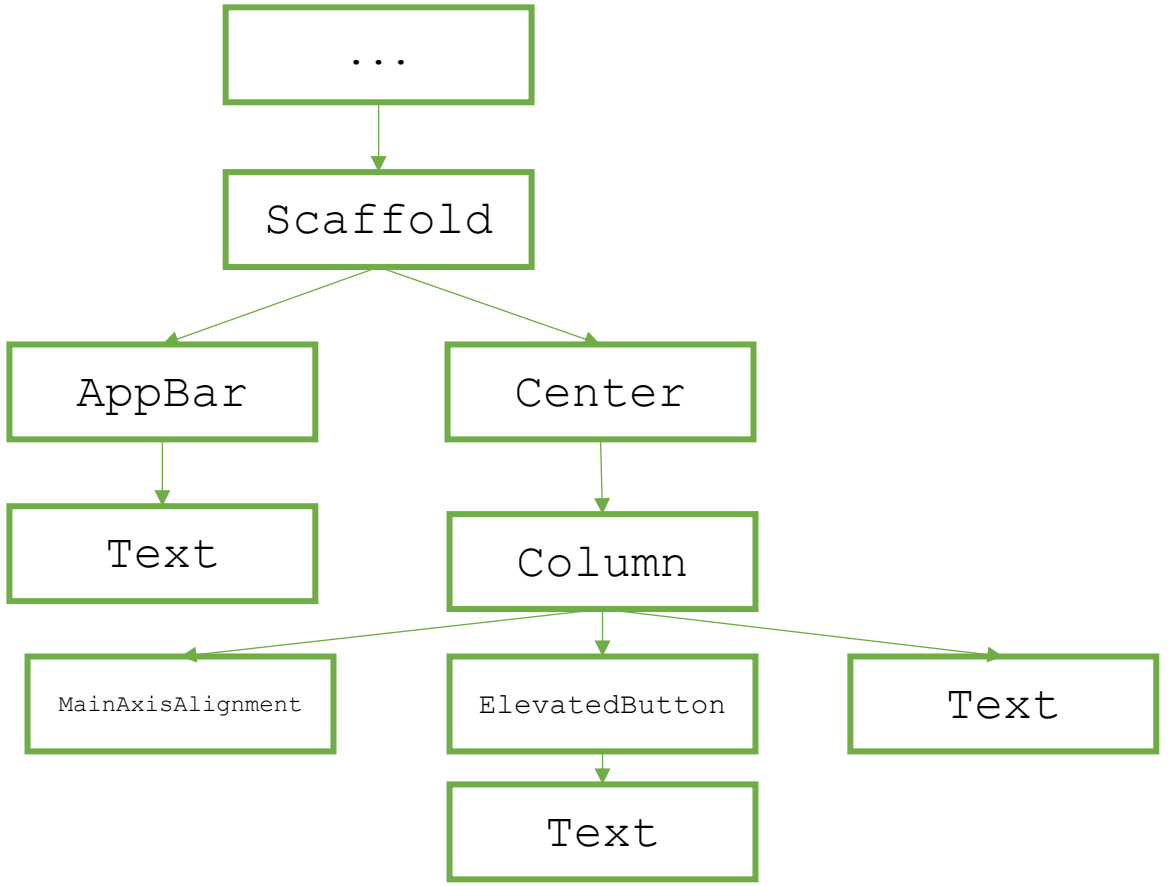
Outline

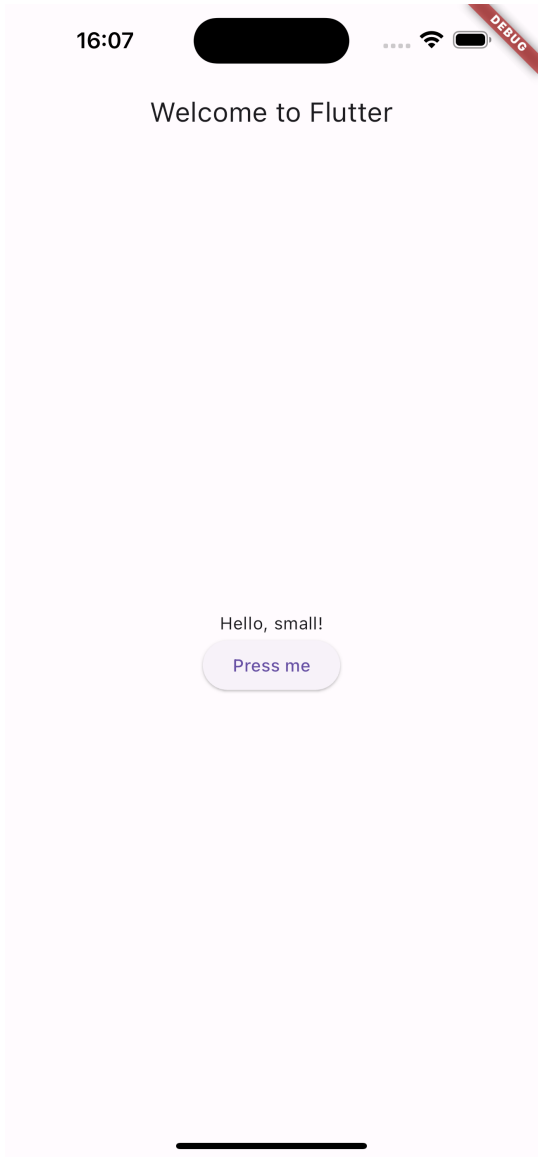
- **Recap**
- Some fundamental Flutter Widgets
- Layout in Flutter
- App#1: layout_basics
- App#2: scaffolding
- ListView

- Exercise
- Homework
- Resources

Recap: Widget tree

- UIs are represented in Flutter hierarchically through a Widget tree





Outline

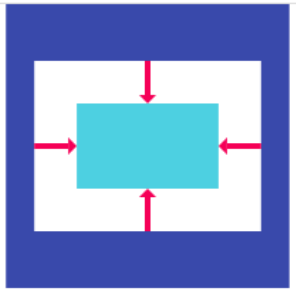
- Recap
- **Some fundamental Flutter Widgets**
- Layout in Flutter
- App#1: layout_basics
- App#2: scaffolding
- ListView

- Exercise
- Homework
- Resources

Fundamental Widget for UI

Scaffold

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



Center

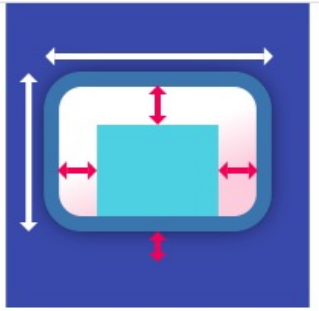
A widget that centers its child within itself.

Column

Layout a list of child widgets in the vertical direction.

Row

Layout a list of child widgets in the horizontal direction.



Container

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

- Of course, there are many others…

Stack

This class is useful if you want to overlap several children in a simple way, for example having some text and an image, overlaid with...

SizedBox

A box with a specified size. If given a child, this widget forces its child to have a specific width and/or height (assuming values are...

Expanded

A widget that expands a child of a Row, Column, or Flex.

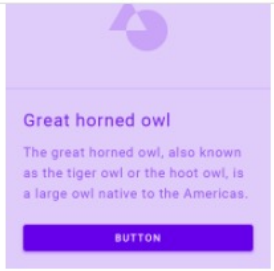
Card

A Material Design card. A card has slightly rounded corners and a shadow.

Fundamental Widget for UI

Text

A run of text with a single style.



ElevatedButton

A Material Design elevated button. A filled button whose material elevates when pressed.

Icon

A Material Design icon.

FutureBuilder

Widget that builds itself based on the latest snapshot of interaction with a Future.

StreamBuilder

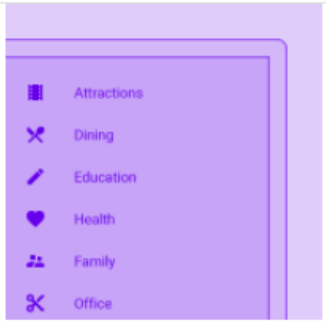
Widget that builds itself based on the latest snapshot of interaction with a Stream.

Form

An optional container for grouping together multiple form field widgets (e.g. TextField widgets).

- Of course, there are many others…

Fundamental Widget for UI



ListView

A scrollable, linear list of widgets. ListView is the most commonly used scrolling widget. It displays its children one after another in the scroll direction....

GridView

A grid list consists of a repeated pattern of cells arrayed in a vertical and horizontal layout. The GridView widget implements this component.

CircularProgressIndicator

A material design circular progress indicator, which spins to indicate that the application is busy.

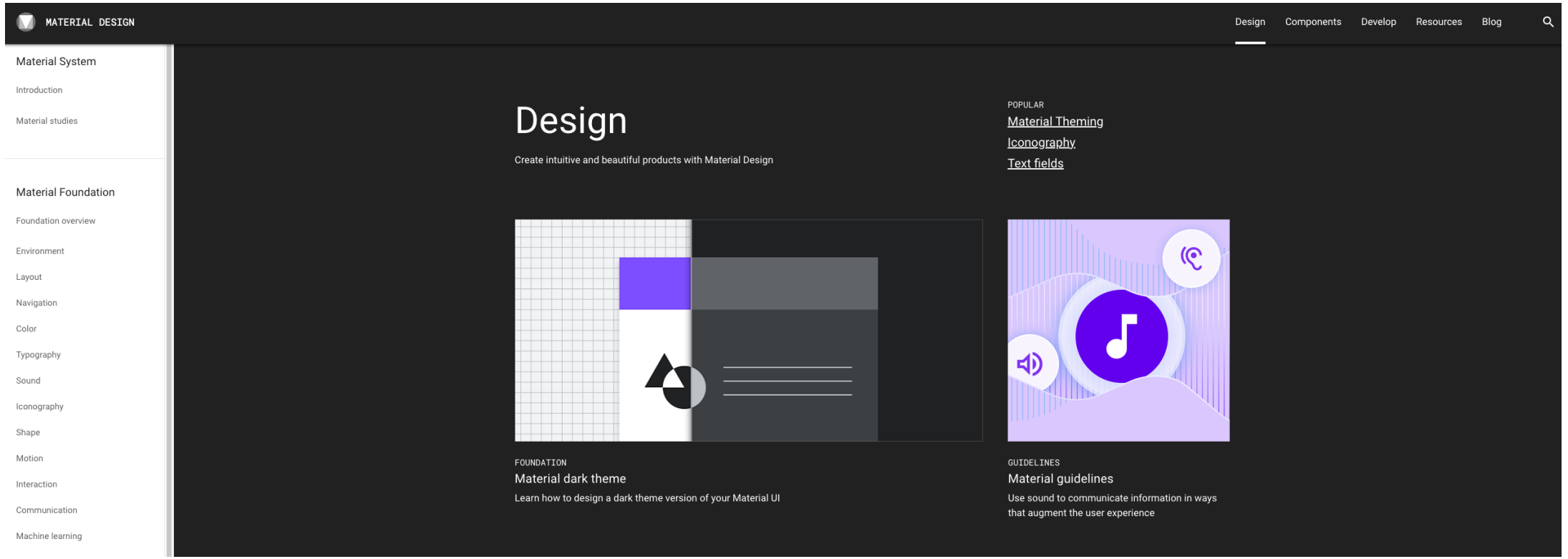
Navigator

A widget that manages a set of child widgets with a stack discipline. Many apps have a navigator near the top of their widget hierarchy...

- Of course, there are many others…

Extensively discussed in the next lesson

Designing your UI - Material



- Many approaches can be followed
- **Material** is an adaptable system of guidelines, components, and tools to support **best practices of UI design**
- Flutter includes many widgets from Material

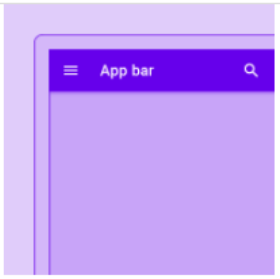
Material UI Widget – App Structure and Navigation

MaterialApp

A convenience widget that wraps a number of widgets that are commonly required for applications implementing Material Design.

Scaffold

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

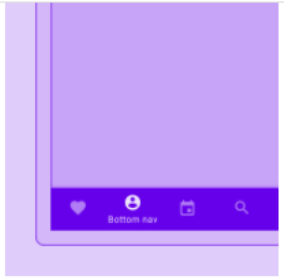


Appbar

A Material Design app bar. An app bar consists of a toolbar and potentially other widgets, such as a TabBar and a FlexibleSpaceBar.

Drawer

A Material Design panel that slides in horizontally from the edge of a Scaffold to show navigation links in an application.

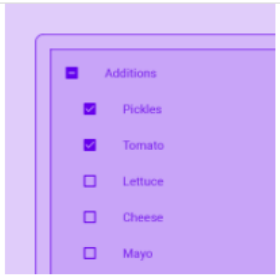


BottomNavigationBar

Bottom navigation bars make it easy to explore and switch between top-level views in a single tap. The BottomNavigationBar widget implements this component.

- Of course, there are many others…

Material UI Widget – Buttons



Checkbox

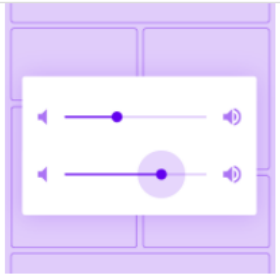
Checkboxes allow the user to select multiple options from a set. The Checkbox widget implements this component.

Date & Time Pickers

Date pickers use a dialog window to select a single date on mobile. Time pickers use a dialog to select a single time (in the...

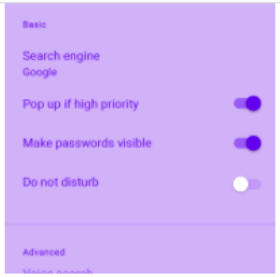
Radio

Radio buttons allow the user to select one option from a set. Use radio buttons for exclusive selection if you think that the user needs...



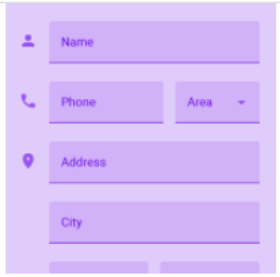
Slider

Sliders let users select from a range of values by moving the slider thumb.



Switch

On/off switches toggle the state of a single settings option. The Switch widget implements this component.

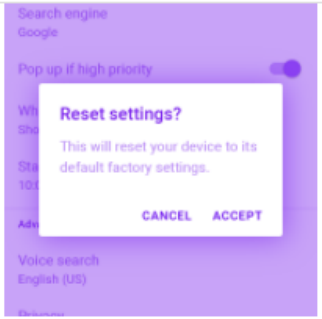


TextField

Touching a text field places the cursor and displays the keyboard. The TextField widget implements this component.

- Of course, there are many others…

Material UI Widget – Dialogs, alerts, and panels



AlertDialog

Alerts are urgent interruptions requiring acknowledgement that inform the user about a situation. The AlertDialog widget implements this component.

BottomSheet

Bottom sheets slide up from the bottom of the screen to reveal more content. You can call showBottomSheet() to implement a persistent bottom sheet or...

ExpansionPanel

Expansion panels contain creation flows and allow lightweight editing of an element. The ExpansionPanel widget implements this component.

SimpleDialog

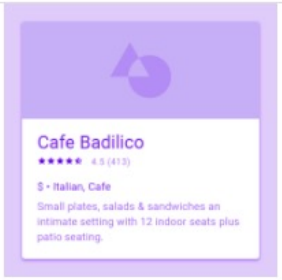
Simple dialogs can provide additional details or actions about a list item. For example they can display avatars icons clarifying subtext or orthogonal actions (such...

SnackBar

A lightweight message with an optional action which briefly displays at the bottom of the screen.

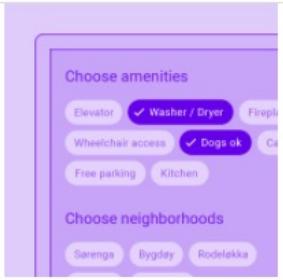
- Of course, there are many others…

Material UI Widget – Information displays



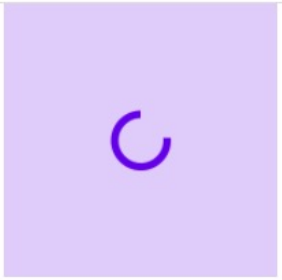
Card

A Material Design card. A card has slightly rounded corners and a shadow.



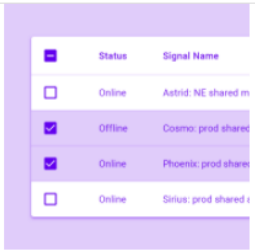
Chip

A Material Design chip. Chips represent complex entities in small blocks, such as a contact.



CircularProgressIndicator

A material design circular progress indicator, which spins to indicate that the application is busy.



DataTable

Data tables display sets of raw data. They usually appear in desktop enterprise products. The DataTable widget implements this component.

GridView

A grid list consists of a repeated pattern of cells arrayed in a vertical and horizontal layout. The GridView widget implements this component.

Icon

A Material Design icon.

Image

A widget that displays an image.

LinearProgressIndicator

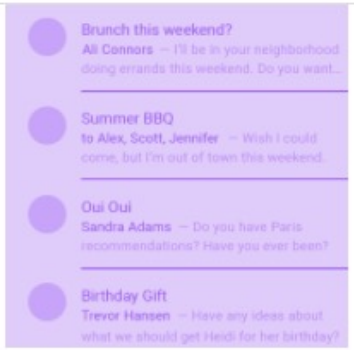
A material design linear progress indicator, also known as a progress bar.

Tooltip

Tooltips provide text labels that help explain the function of a button or other user interface action. Wrap the button in a Tooltip widget to...

- Of course, there are many others…

Material UI Widget – Layout



Divider

A one logical pixel thick horizontal line, with padding on either side.

ListTile

A single fixed-height row that typically contains some text as well as a leading or trailing icon.

Stepper

A Material Design stepper widget that displays progress through a sequence of steps.

- Of course, there are many others…

Outline

- Recap
- Some fundamental Flutter Widgets
- **Layout in Flutter**
- App#1: layout_basics
- App#2: scaffolding
- ListView

- Exercise
- Homework
- Resources

Layouts in Flutter

- Layouts are organized in a Widget tree
- But how a widget is sized and positioned somewhere?
- To answer this question, we need to fully understand this rule:

Constraints go down. Sizes go up. Parent sets position.

The set of 4 doubles: minimum and maximum height, minimum and maximum width.

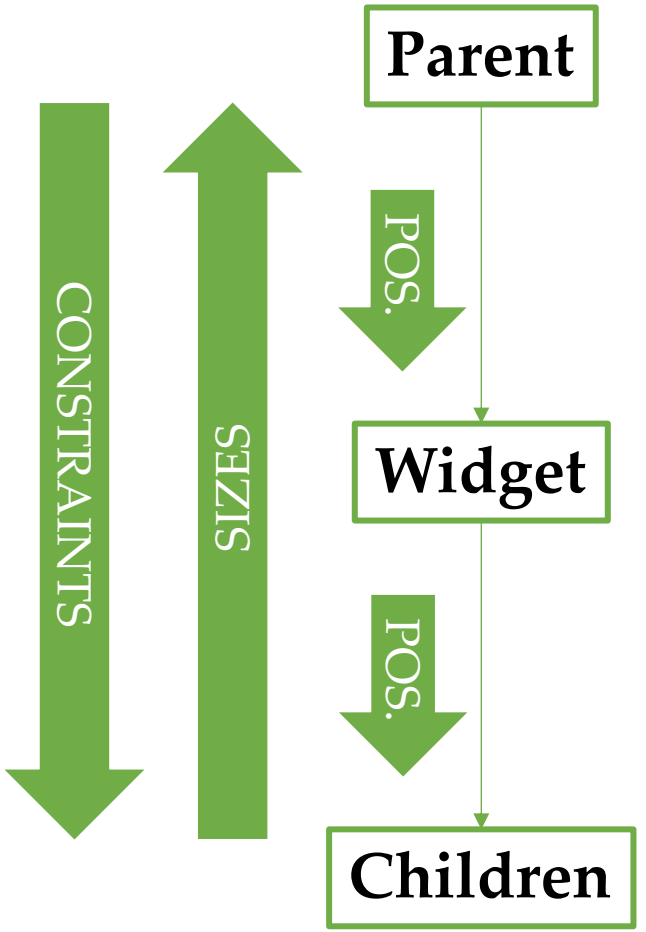
The set of 2 doubles: height and width.

The set of 2 doubles: x and y.

Layouts in Flutter

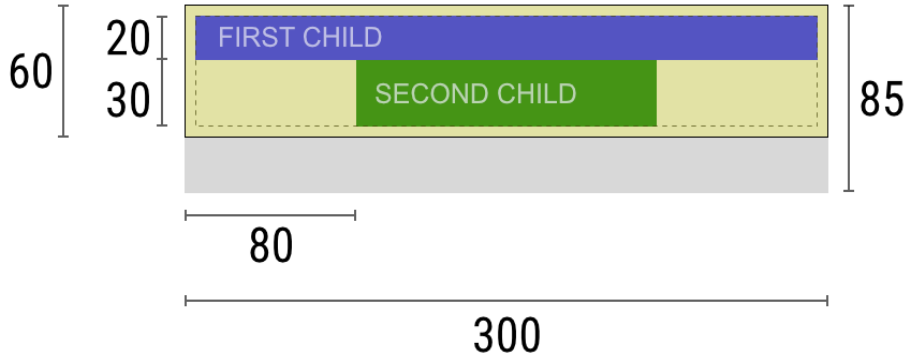
Constraints go down. Sizes go up. Parent sets position.

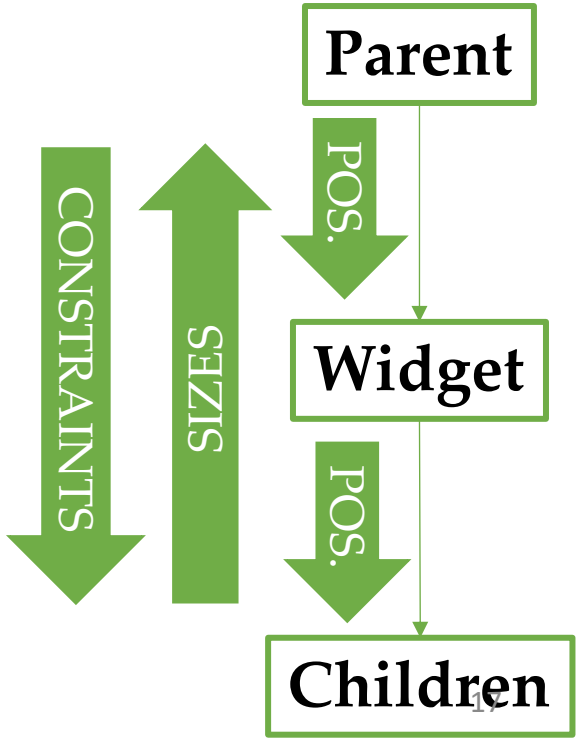
- A widget gets its constraints from its parent and tells its children what are the constraints (i.e., "**constraints go down**").
- A widget then asks its children the sizes they want to be (i.e., "**sizes go up**").
- A widget positions its children (i.e., "**parent sets position**").



Negotiating constraints, sizes, and position

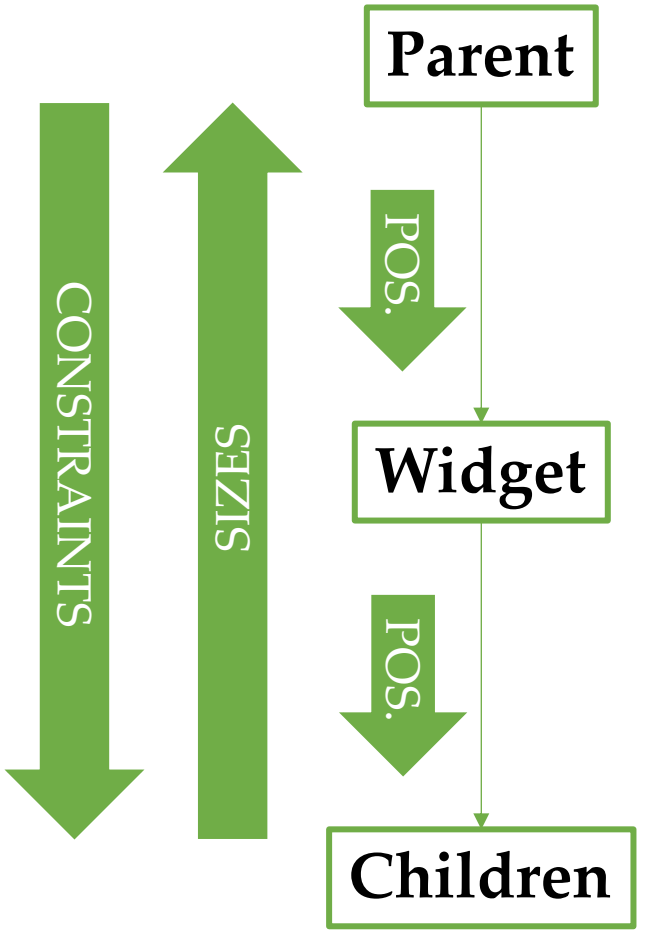
- **Widget**: "Hey parent, what are my constraints?"
- **Parent**: "You must be from 80 to 300 pixels wide, and 30 to 85 tall."
- **Widget**: "Since I want to have 5 pixels of padding, then my children can have at most 290 pixels of width and 75 pixels of height."
- **Widget**: "Hey first child, you must be from 0 to 290 pixels wide, and 0 to 75 tall."
- **First child**: "OK, then I wish to be 290 pixels wide, and 20 pixels tall."
- **Widget**: "Since I want to put my second child below the first one, this leaves only 55 pixels of height for my second child."
- **Widget**: "Hey second child, You must be from 0 to 290 wide, and 0 to 55 tall."
- **Second child**: "OK, I wish to be 140 pixels wide, and 30 pixels tall."
- **Widget**: "Very well. My first child has position x: 5 and y: 5, and my second child has x: 80 and y: 25."
- **Widget**: "Hey parent, I've decided that my size is going to be 300 pixels wide, and 60 pixels tall."





Limitations

- A widget can decide its own size only within the constraints given to it by its parent: a widget **can't have any size it wants**.
- A widget **can't know and doesn't decide its own position in the screen**.
- It's impossible to precisely define the size and position of any widget without taking into consideration the tree as a whole.
- If a child wants a different size from its parent and the parent doesn't have enough information to align it, then **the child's size might be ignored**.



Outline

- Recap
- Some fundamental Flutter Widgets
- Layout in Flutter
- **App#1: layout_basics**
- App#2: scaffolding
- ListView

- Exercise
- Homework
- Resources

Dive into examples

- In the “Understanding constraints” article by Marcelo Glasberg https://docs.flutter.dev/development/ui/layout/constraints (you will also find the link in the Resources section of this presentation) there are a lot (29) of examples explaining how all of this works.

- Let’s report the most interesting ones.

Example 1

```
Widget _example1() => Container(color:
Colors.red);
```

- The screen is the parent of the `Container`, and it forces the `Container` to be exactly the same size as the screen.
- So the `Container` fills the screen and paints it red.

Example 2

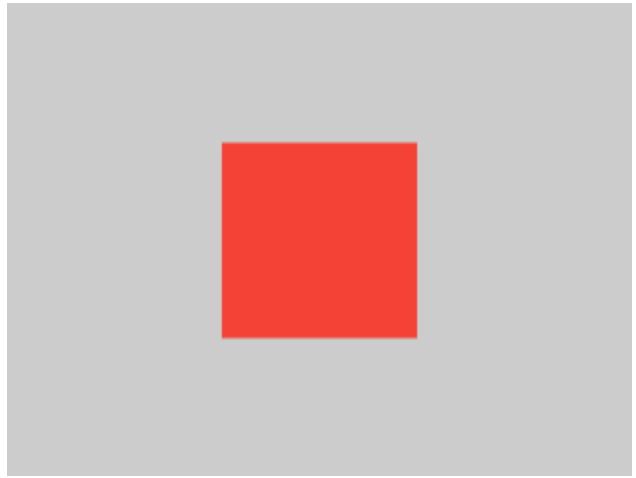
```
Widget _example2() => Container(width:
100, height: 100, color: red);
```

- The red `Container` wants to be 100 × 100, but it can't, because the screen forces it to be exactly the same size as the screen.

- The `Container` fills the screen.

Example 3

```
Widget _example3() => Center(
  child: Container(width: 100, height: 100,
color: red),
);
```



- The screen forces the `Center` to be exactly the same size as the screen, so the `Center` fills the screen.
- The `Center` tells the `Container` that it can be any size it wants, but not bigger than the screen. Now the `Container` can be 100 × 100.

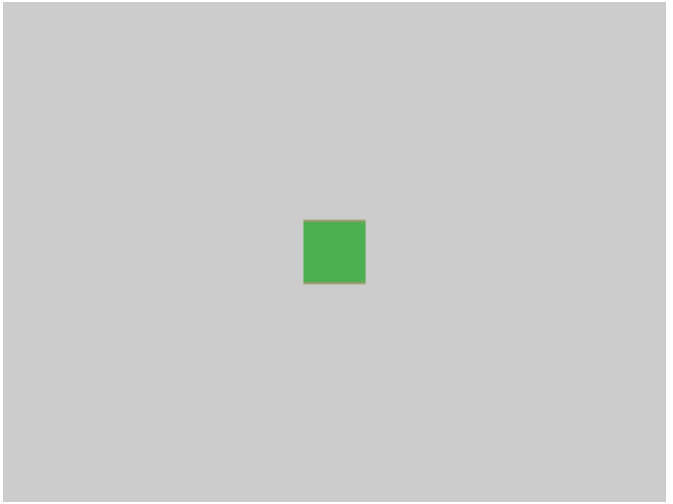
Example 6

```
Widget _example6() => Center(
   child: Container(color: red),
)
```

- The screen forces the `Center` to be exactly the same size as the screen, so the `Center` fills the screen.
- The `Center` tells the `Container` that it can be any size it wants, but not bigger than the screen. Since the `Container` has no child and no fixed size, it decides it wants to be as big as possible, so it fills the whole screen.

Example 7

```
Widget _example7() => Center(
  child: Container(
    color: red,
    child: Container(color: green, width: 30, height:
30),
  ),
);
```



- The screen forces the `Center` to be exactly the same size as the screen, so the `Center` fills the screen.
- The `Center` tells the red `Container` that it can be any size it wants, but not bigger than the screen. Since the red `Container` has no size but has a child, it decides it wants to be the same size as its child.
- The red `Container` tells its child that it can be any size it wants, but not bigger than the screen.
- The child is a green `Container` that wants to be 30 × 30. Given that the red `Container` sizes itself to the size of its child, it is also 30 × 30. The red color isn't visible because the green `Container` entirely covers the red `Container`.

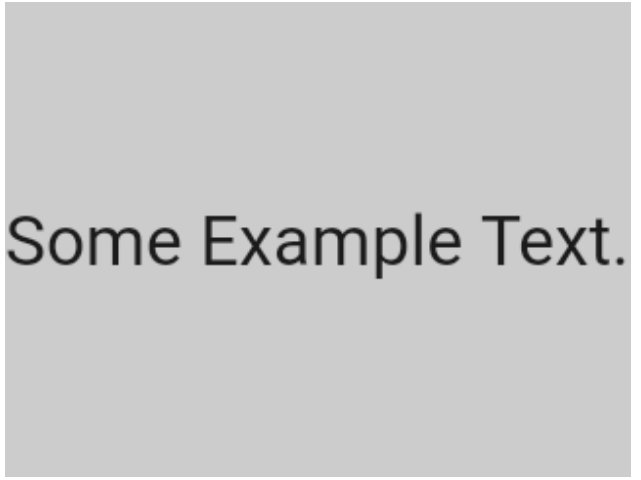
Example 14

```
Widget _example14() => UnconstrainedBox(
  child: Container(color: red, width: 4000,
height: 50),
);
```

- The screen forces the `UnconstrainedBox` to be exactly the same size as the screen,
  and `UnconstrainedBox` lets its child `Container` be any size it wants.
- Unfortunately, in this case the `Container` is 4000 pixels wide and is too big to fit in the `UnconstrainedBox`, so the `UnconstrainedBox` displays the much dreaded "overflow warning".

Example 18

```
Widget _example18() => const FittedBox(
  child: Text('Some Example Text.'),
);
```

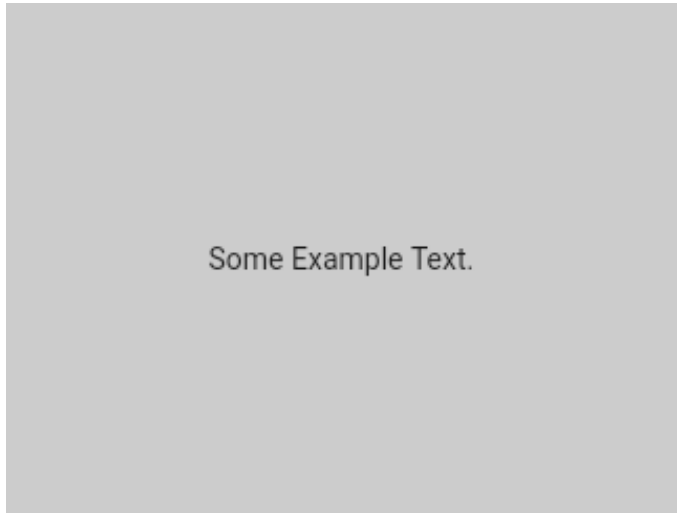


- The screen forces the `FittedBox` to be exactly the same size as the screen. The `Text` has some natural width (also called its intrinsic width) that depends on the amount of text, its font size, and so on.

- The `FittedBox` lets the `Text` be any size it wants, but after the `Text` tells its size to the `FittedBox`, the `FittedBox` scales the `Text` until it fills all of the available width.

Example 19

```
Widget _example19() => const Center(
  child: FittedBox(
    child: Text('Some Example Text.'),
  ),
);
```



- But what happens if you put the `FittedBox` inside of a `Center` widget? The `Center` lets the `FittedBox` be any size it wants, up to the screen size.
- The `FittedBox` then sizes itself to the `Text`, and lets the `Text` be any size it wants. Since both `FittedBox` and the `Text` have the same size, no scaling happens.

Full example in lab_05-understanding_ui/layout_basics/

Outline

- Recap
- Some fundamental Flutter Widgets
- Layout in Flutter
- App#1: layout_basics
- **App#2: scaffolding**
- ListView

- Exercise
- Homework
- Resources

Focus on the Scaffold

- Let's focus into the Scaffold Widget
- It provides a framework which implements the basic material design visual layout structure
- Detailed info: https://api.flutter.dev/flutter/material/Scaffold-class.html

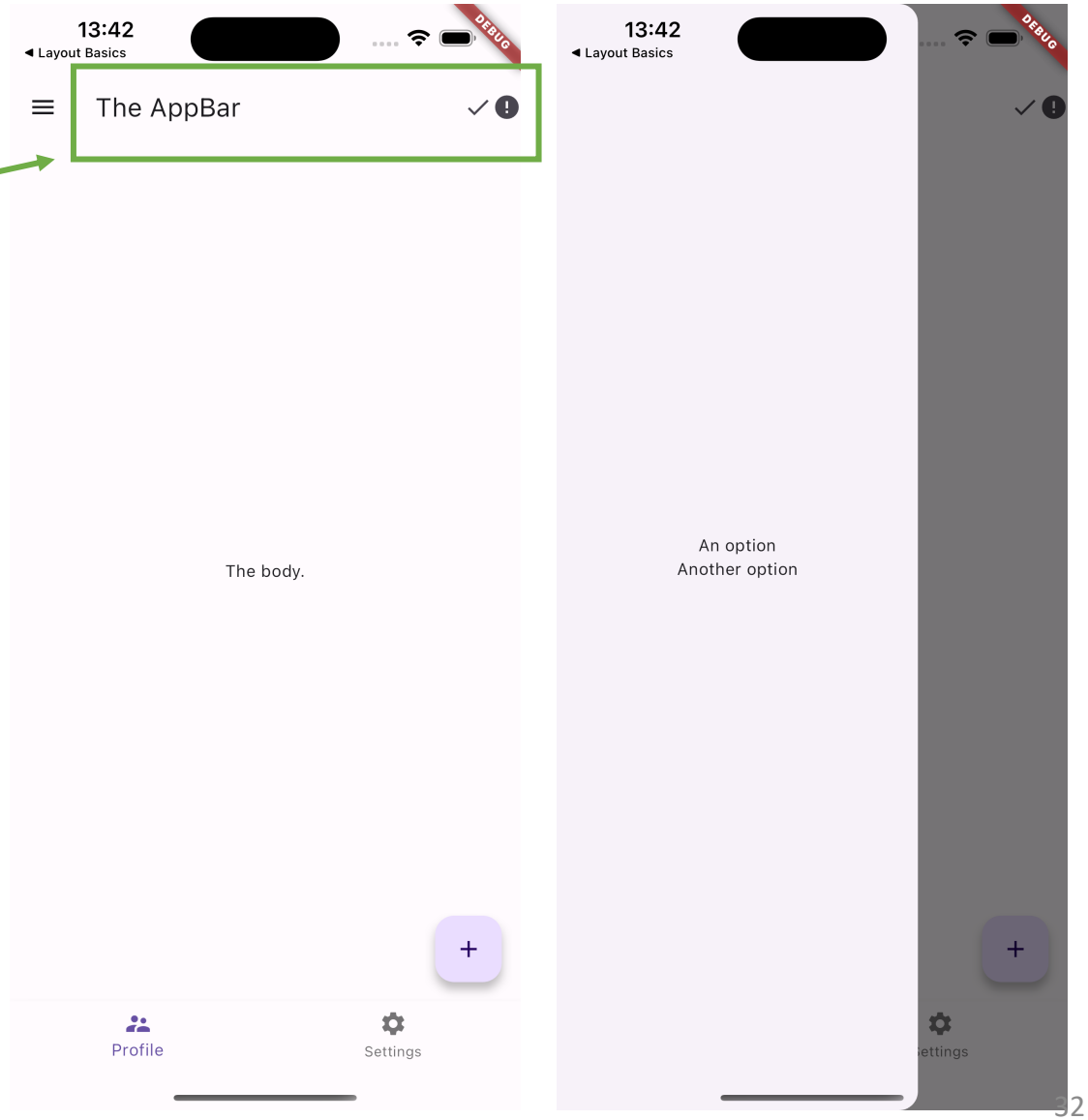
```
//Constructor of Scaffold
const Scaffold({
  Key key,
  this.appBar,
  this.body,
  this.floatingActionButton,
  this.floatingActionButtonLocation,
  this.floatingActionButtonAnimator,
  this.persistentFooterButtons,
  this.drawer,
  this.endDrawer,
  this.bottomNavigationBar,
  this.bottomSheet,
  this.backgroundColor,
  this.resizeToAvoidBottomPadding = true,
  this.primary = true,
})
```

Focus on the Scaffold

```
//Constructor of Scaffold
const Scaffold({
  Key key,
  this.appBar,
  this.body,
  this.floatingActionButton,
  this.floatingActionButtonLocation,
  this.floatingActionButtonAnimator,
  this.persistentFooterButtons,
  this.drawer,
  this.endDrawer,
  this.bottomNavigationBar,
  this.bottomSheet,
  this.backgroundColor,
  this.resizeToAvoidBottomPadding = true,
  this.primary = true,
})
```

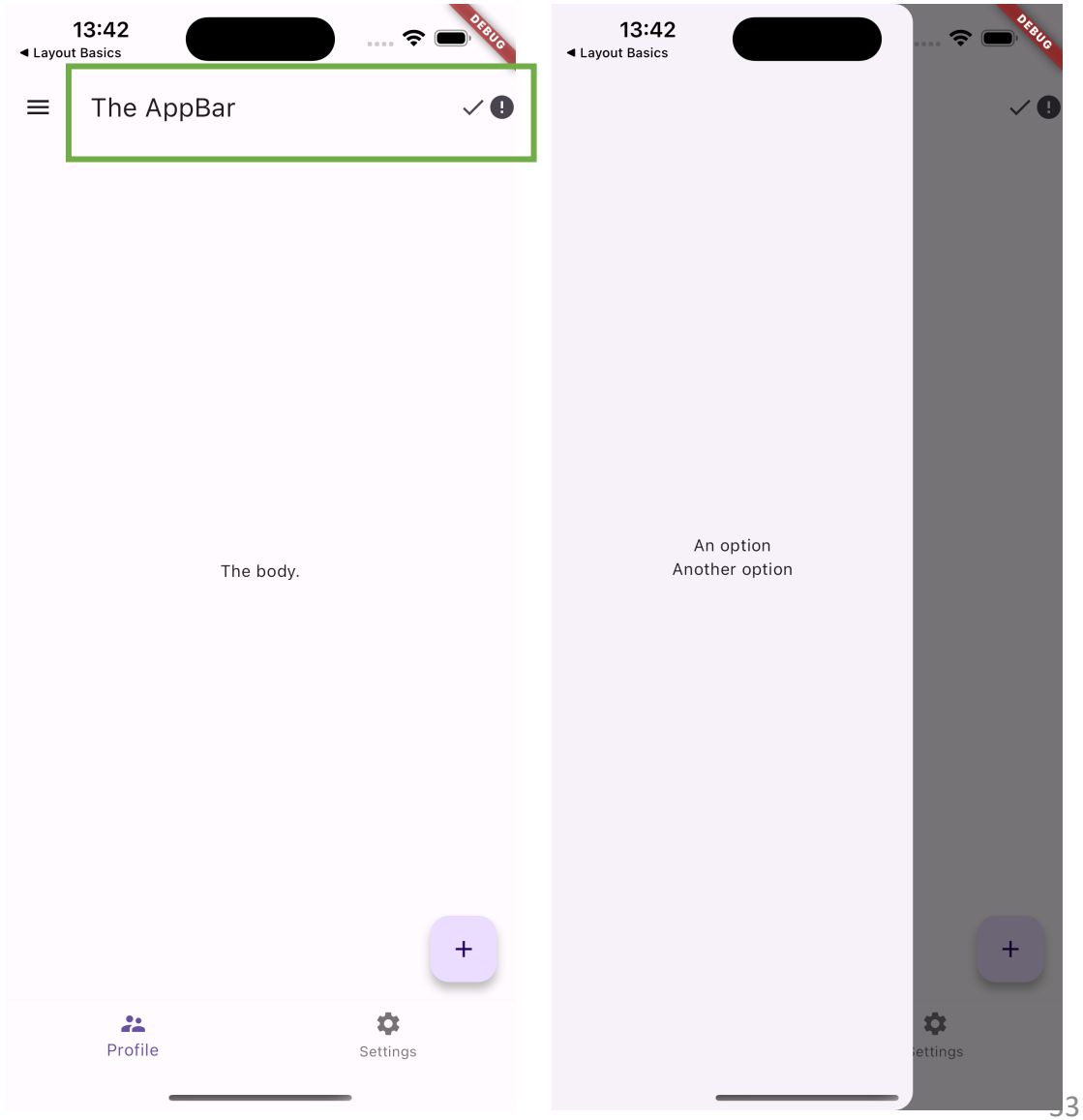
Focus on the Scaffold

```
//Constructor of Scaffold
const Scaffold({
  Key key,
  this.appBar,
  this.body,
  this.floatingActionButton,
  this.floatingActionButtonLocation,
  this.floatingActionButtonAnimator,
  this.persistentFooterButtons,
  this.drawer,
  this.endDrawer,
  this.bottomNavigationBar,
  this.bottomSheet,
  this.backgroundColor,
  this.resizeToAvoidBottomPadding = true,
  this.primary = true,
})
```



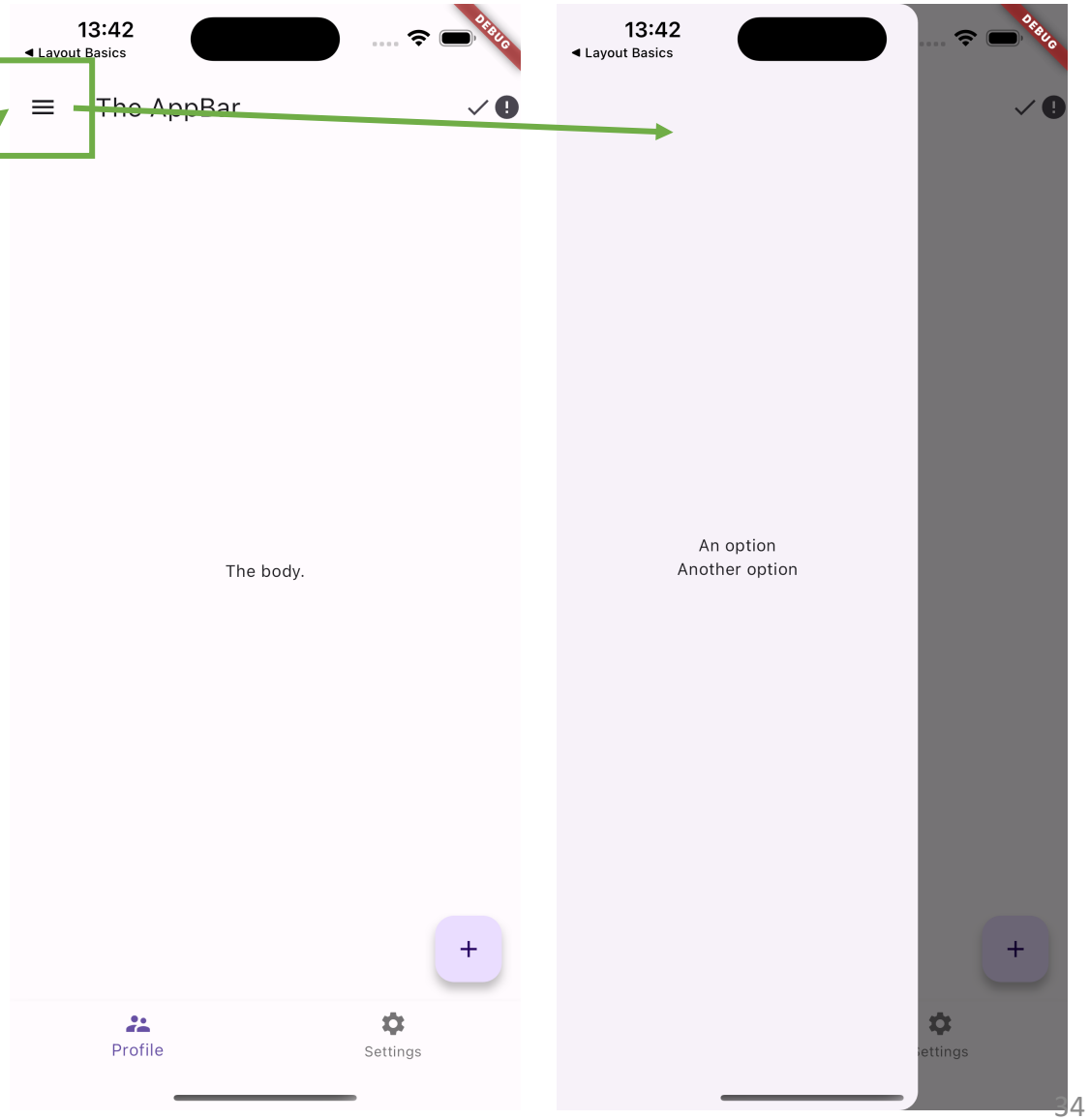
Focus on the Scaffold

```
...
home: Scaffold(
  appBar: AppBar(
    actions: [
      Icon(Icons.done,),
      Icon(Icons.error),
    ],
    title: Text('The AppBar'),
  ),
...
```



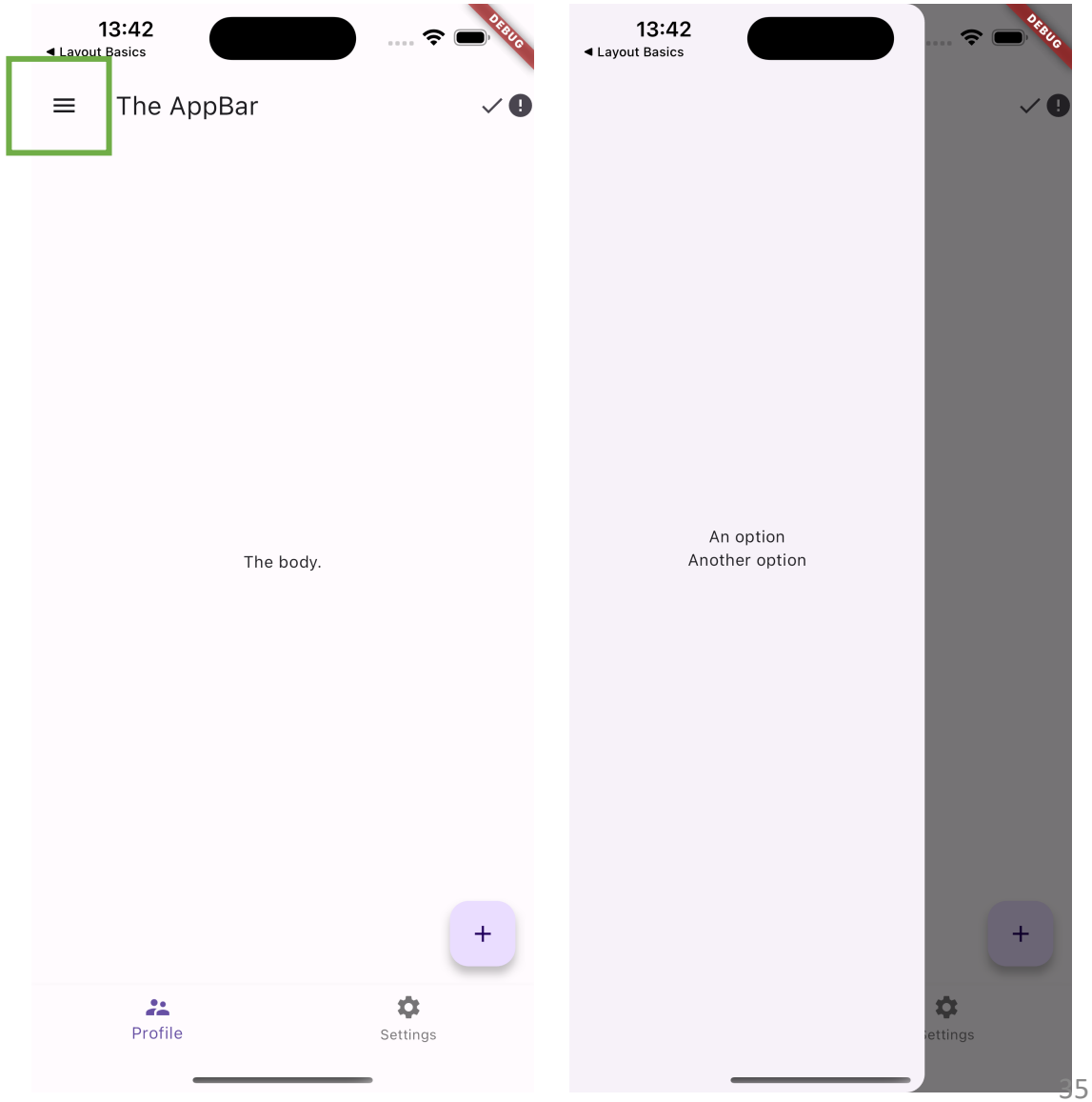
Focus on the Scaffold

```
//Constructor of Scaffold
const Scaffold({
  Key key,
  this.appBar,
  this.body,
  this.floatingActionButton,
  this.floatingActionButtonLocation,
  this.floatingActionButtonAnimator,
  this.persistentFooterButtons,
  this.drawer,
  this.endDrawer,
  this.bottomNavigationBar,
  this.bottomSheet,
  this.backgroundColor,
  this.resizeToAvoidBottomPadding = true,
  this.primary = true,
})
```



Focus on the Scaffold

```
...
home: Scaffold(
  ...
  drawer: Drawer(
    child: Column(
      mainAxisAlignment:
MainAxisAlignment.center,
      children: [
        Text('An option'),
        Text('Another option'),
      ],
    ),
  ),
...
```

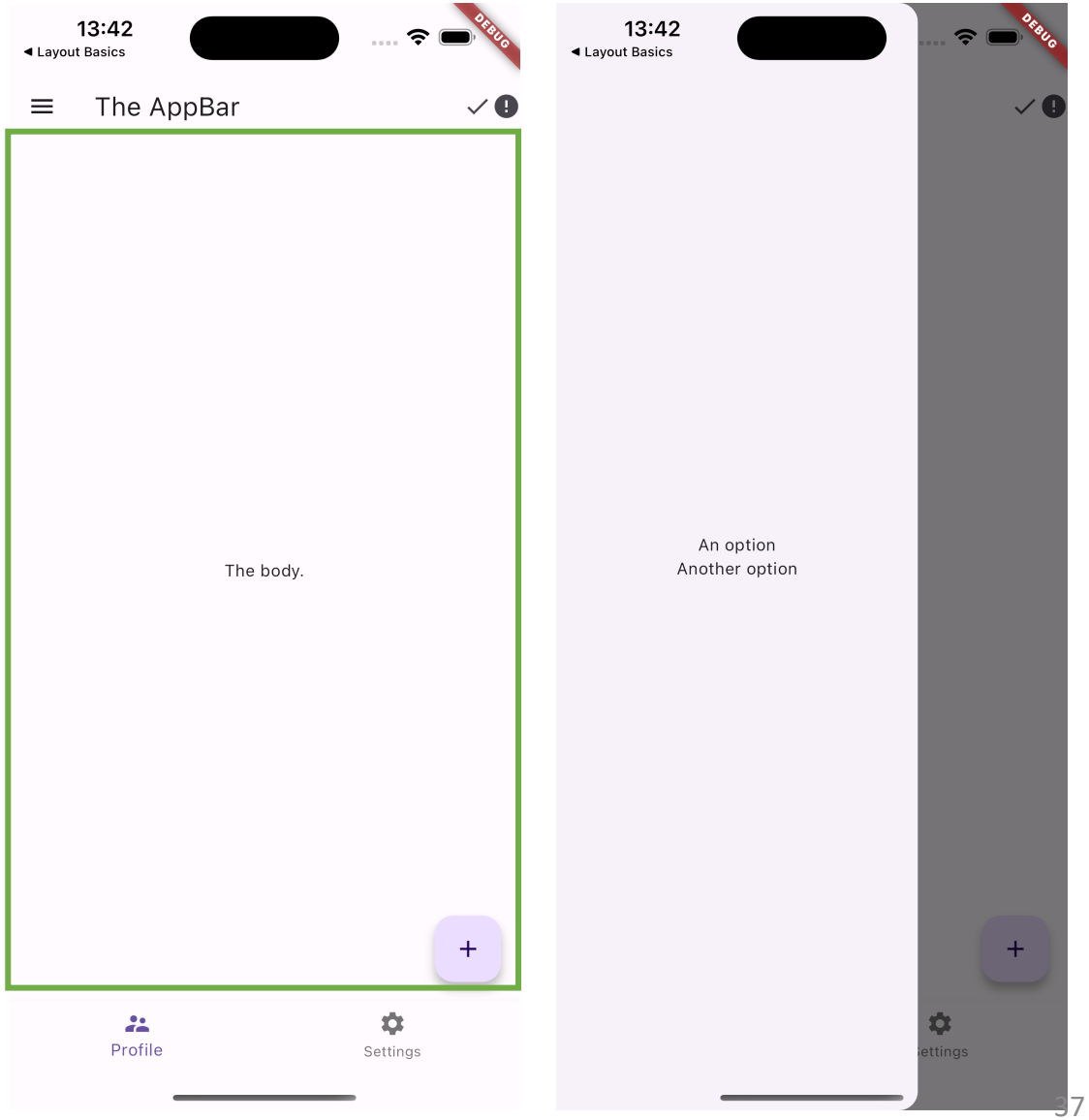


Focus on the Scaffold

```
//Constructor of Scaffold
const Scaffold({
  Key key,
  this.appBar,
  this.body,
  this.floatingActionButton,
  this.floatingActionButtonLocation,
  this.floatingActionButtonAnimator,
  this.persistentFooterButtons,
  this.drawer,
  this.endDrawer,
  this.bottomNavigationBar,
  this.bottomSheet,
  this.backgroundColor,
  this.resizeToAvoidBottomPadding = true,
  this.primary = true,
})
```

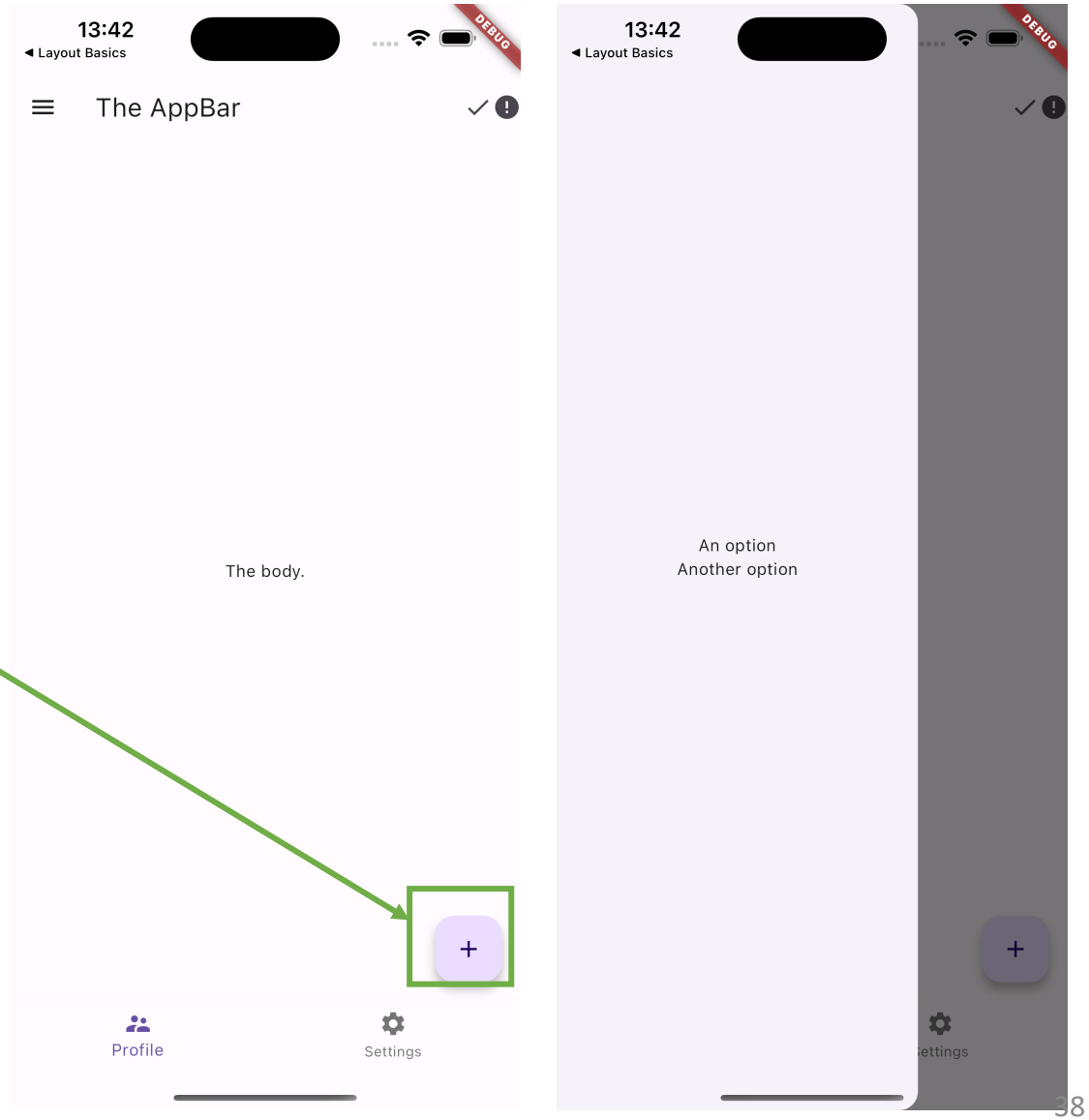
Focus on the Scaffold

```
...
home: Scaffold(
  ...
  body: Center(
    child: Text('The body.'),
  ),
...
```



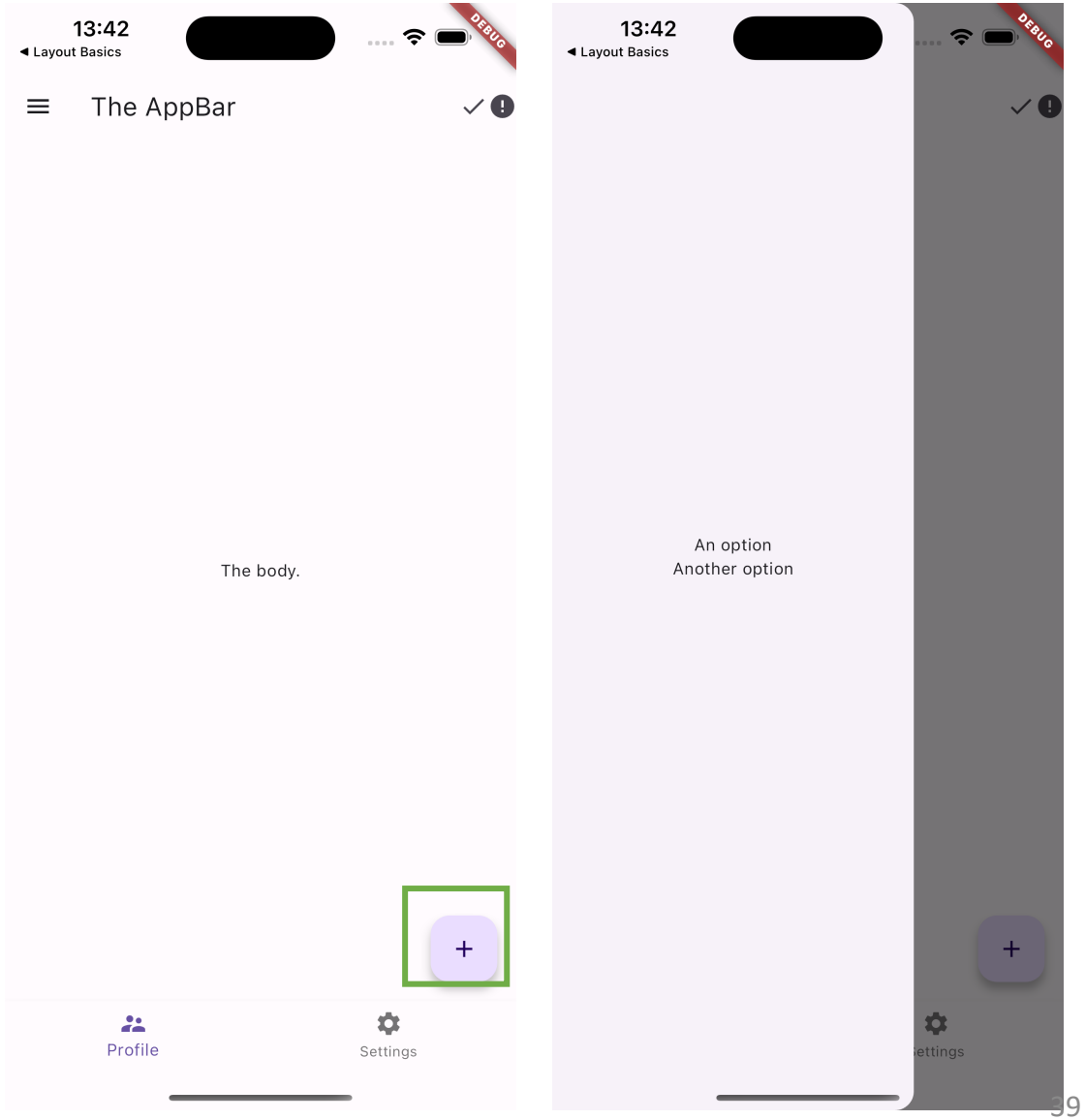
Focus on the Scaffold

```
//Constructor of Scaffold
const Scaffold({
  Key key,
  this.appBar,
  this.body,
  this.floatingActionButton,
  this.floatingActionButtonLocation,
  this.floatingActionButtonAnimator,
  this.persistentFooterButtons,
  this.drawer,
  this.endDrawer,
  this.bottomNavigationBar,
  this.bottomSheet,
  this.backgroundColor,
  this.resizeToAvoidBottomPadding = true,
  this.primary = true,
})
```



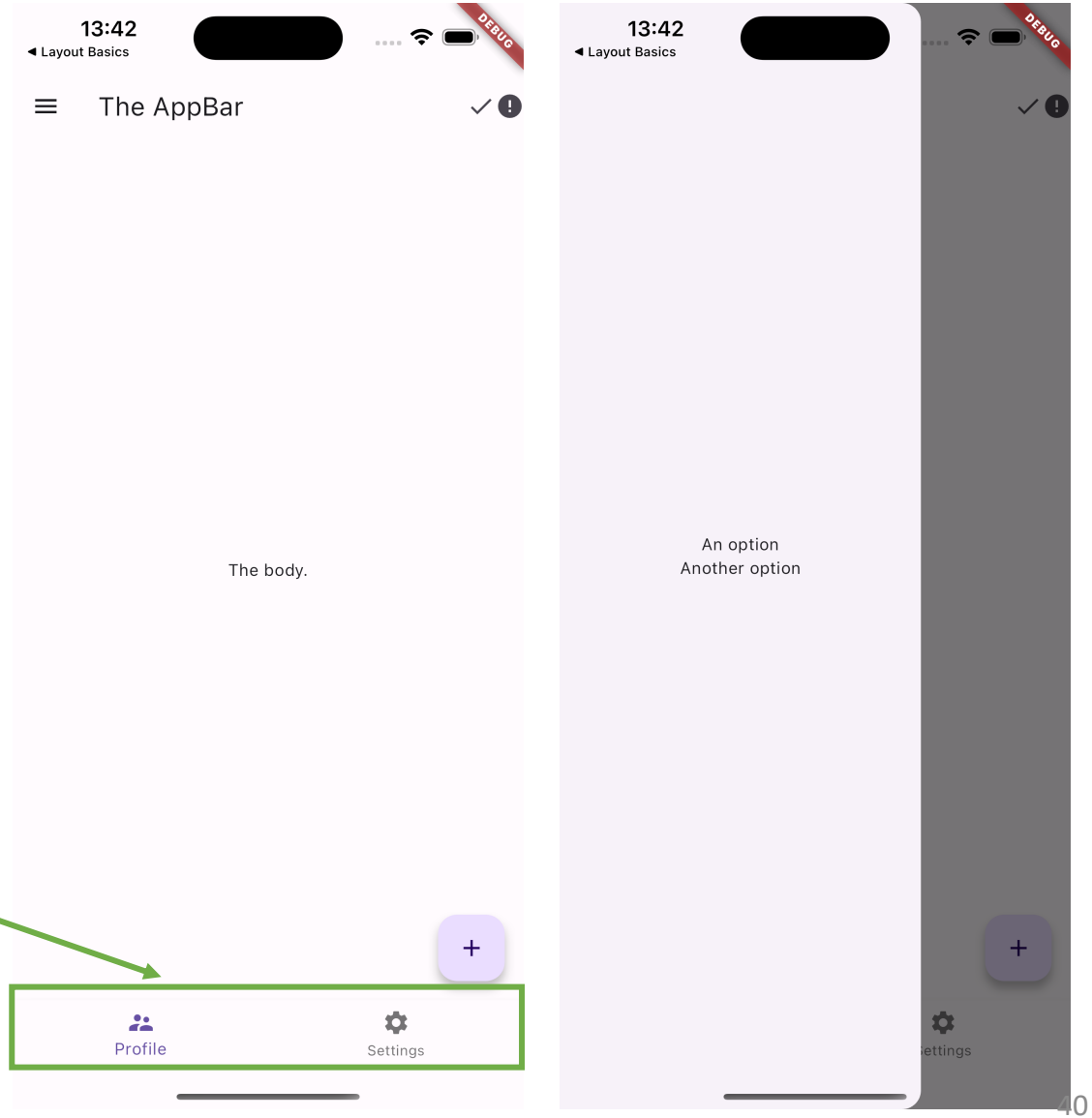
Focus on the Scaffold

```
...
home: Scaffold(
  ...
  floatingActionButton:
    FloatingActionButton(
      child: Icon(Icons.add),
      onPressed: () {},
    ),
...
```



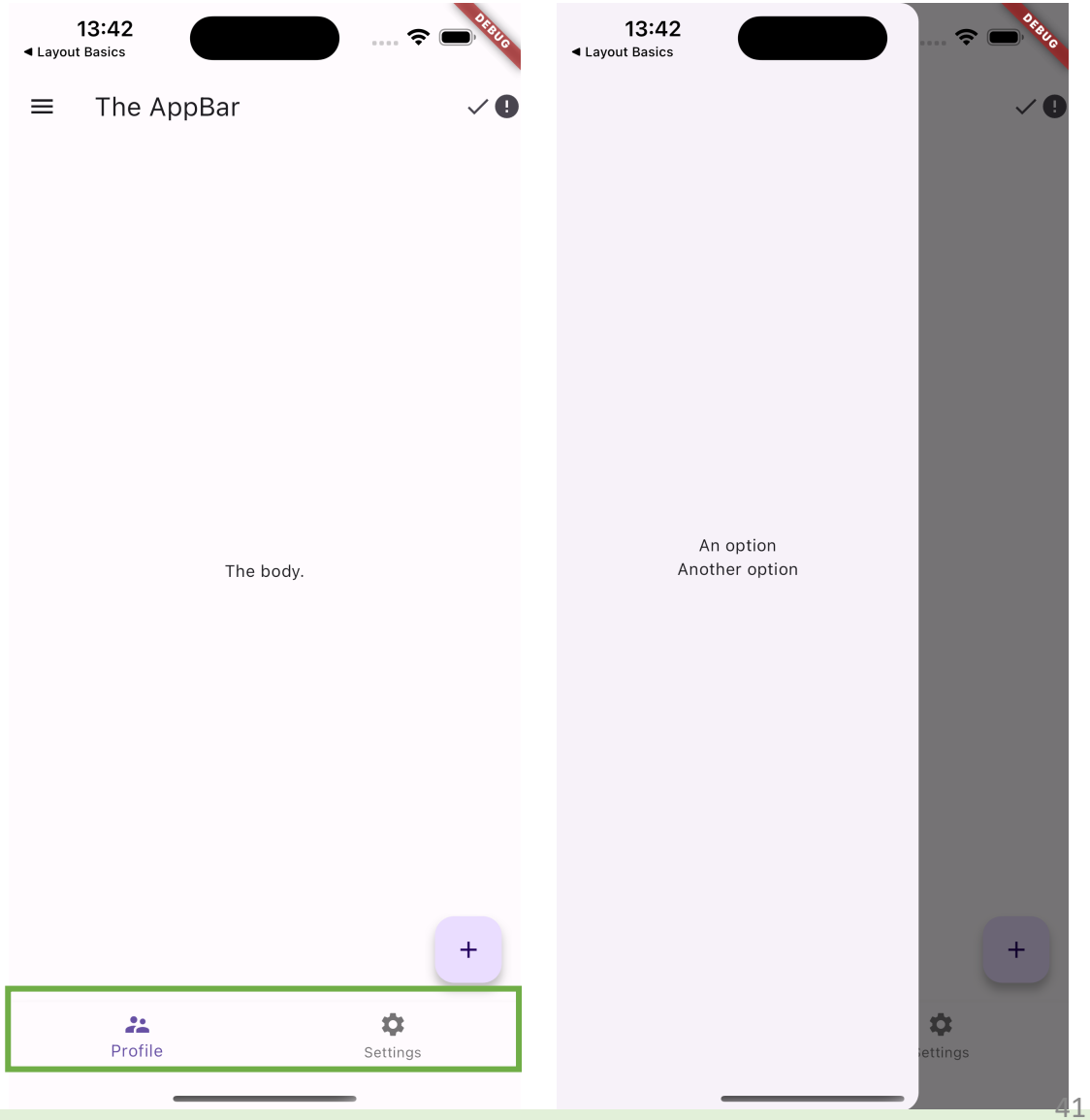
Focus on the Scaffold

```
//Constructor of Scaffold
const Scaffold({
  Key key,
  this.appBar,
  this.body,
  this.floatingActionButton,
  this.floatingActionButtonLocation,
  this.floatingActionButtonAnimator,
  this.persistentFooterButtons,
  this.drawer,
  this.endDrawer,
  this.bottomNavigationBar,
  this.bottomSheet,
  this.backgroundColor,
  this.resizeToAvoidBottomPadding = true,
  this.primary = true,
})
```



Focus on the Scaffold

```
...
home: Scaffold(
  ...
  bottomNavigationBar:
    BottomNavigationBar(
      items: [
        BottomNavigationBarItem(
          icon:
Icon(Icons.supervisor_account),
          label: 'Profile',
        ),
        BottomNavigationBarItem(
          icon: Icon(Icons.settings),
          label: 'Settings',
        )
      ],
  ),
),
```



Full example in lab_05-understanding_ui/scaffolding/

Outline

- Recap
- Some fundamental Flutter Widgets
- Layout in Flutter
- App#1: layout_basics
- App#2: scaffolding
- **ListView**

- Exercise
- Homework
- Resources

ListView

- Let's replace the body with something cooler: ListView

- ListView is the most commonly used scrolling widget. It displays its children one after another in the scroll direction. In the cross axis, the children are required to fill the ListView.

- Detailed info: https://api.flutter.dev/flutter/widgets/ListView-class.html

ListView

```
...
home: Scaffold(
  ...
  body: ListView(
    children: [
      ListTile(
        leading: Icon(Icons.settings),
        title: Text('ListTile 1 title’),
        subtitle: Text('ListTile 1 subtitle’),
        trailing: Icon(Icons.arrow_right),
      ),
      ListTile(
        leading: Icon(Icons.settings),
        title: Text('ListTile 2 title’),
        subtitle: Text('ListTile 2 subtitle'),
        trailing: Icon(Icons.arrow_right),
      ),
    ],
  ),
```

Full example in lab_05-understanding_ui/scaffolding/

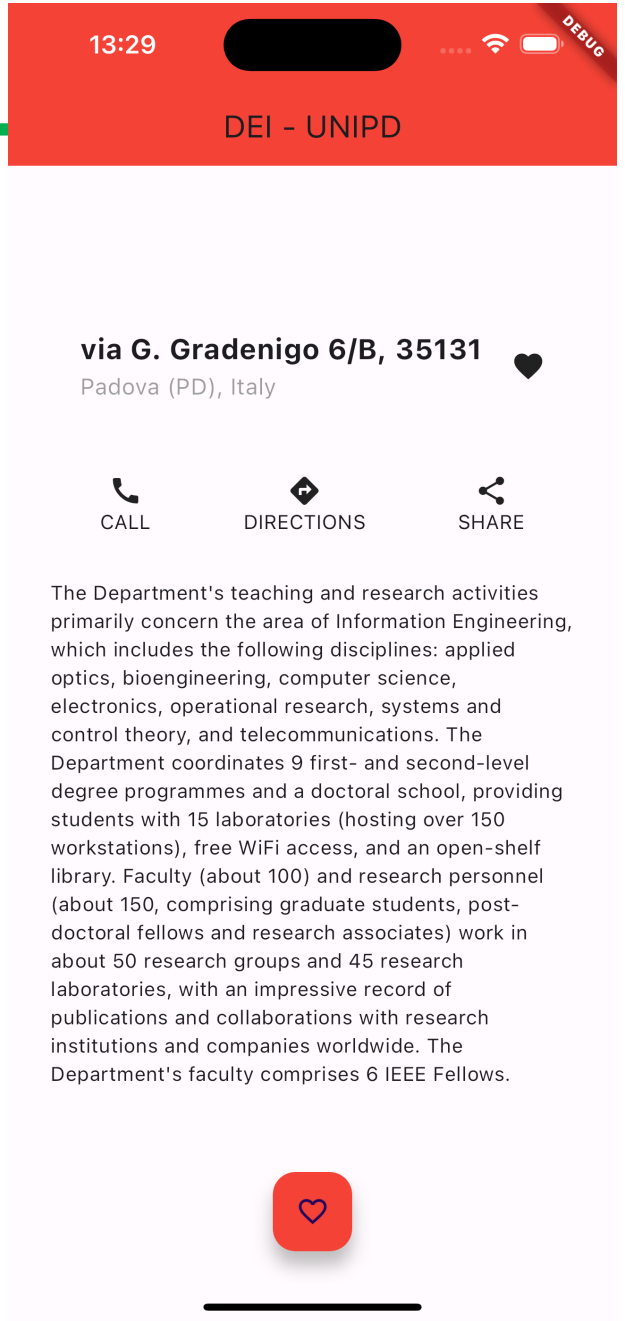
Outline

- Recap
- Some fundamental Flutter Widgets
- Layout in Flutter
- App#1: layout_basics
- App#2: scaffolding
- ListView

- **Exercise**
- Homework
- Resources

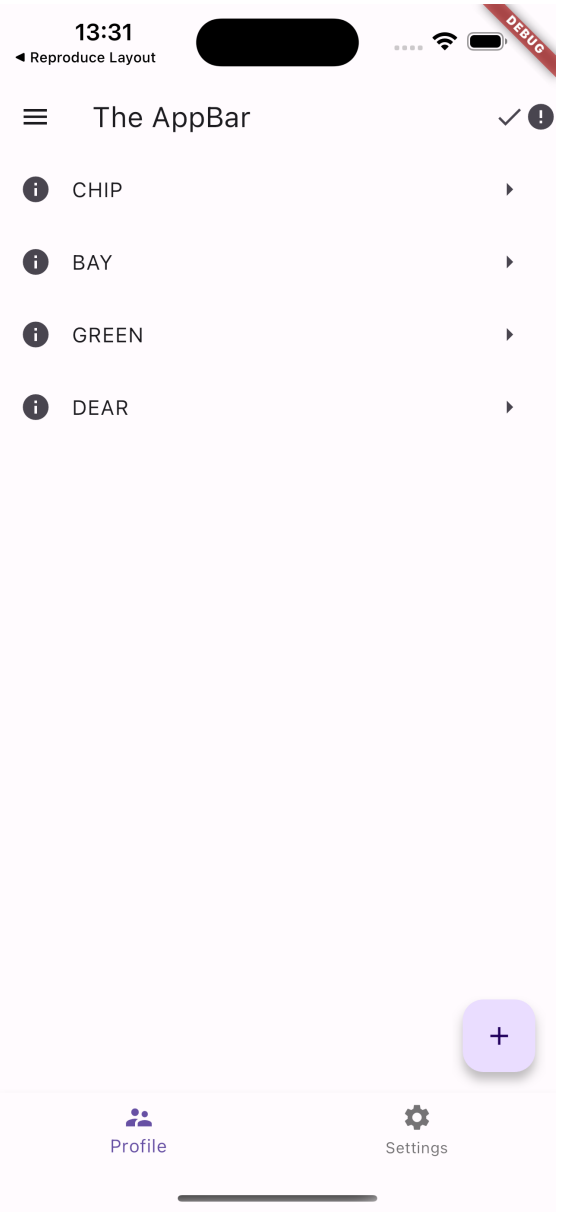
Exercise

- Exercise 05.01 (easy)
  - Create a new project ‘reproduce_layout’
  - Reproduce, as close as possible, the layout on the right.
  - Hint: I used the following widgets: AppBar, Text, Container, Icon, SizedBox, Column, Row, Colors, and some others…



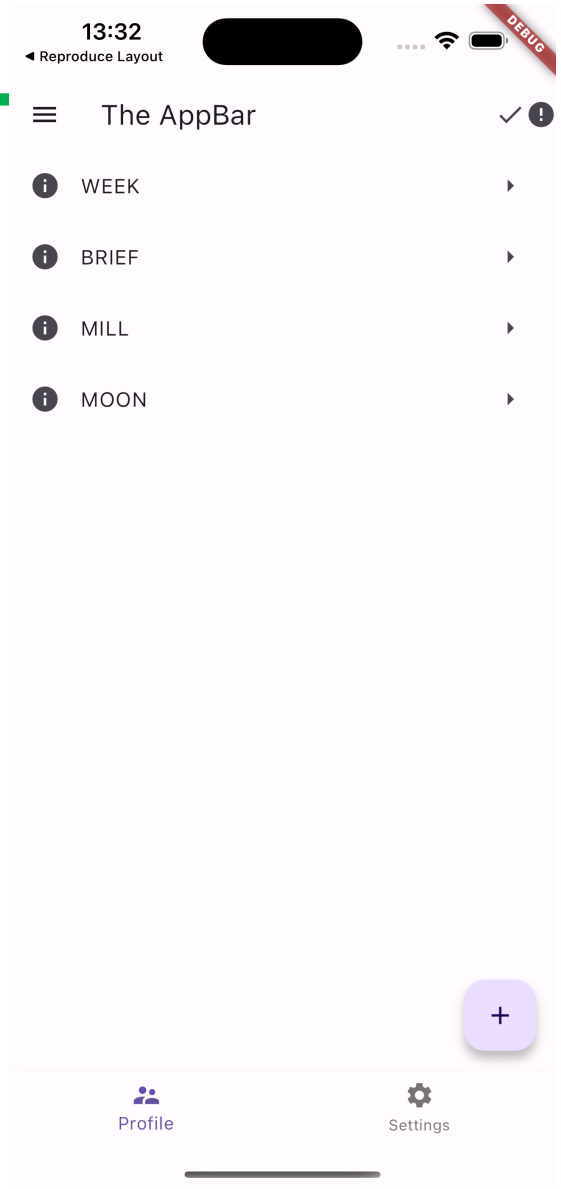
Exercise

- Exercise 05.02 (easy)
  - Create a new project ‘exercise_listview’
  - Copy the code of main.dart of ‘scaffolding’ into the main.dart of ‘exercise_view’
  - Modify the ListView so that the app looks like the app on the right (hint: I used Icons.info)



Exercise

- Exercise 05.03 (easy)
  - Install the english_words package into the ‘exercise_listview’ app
  - Starting from the code of exercise 05.02, modify each ListTile in order to have as a title a random word in uppercase generated using the english_words package just installed



Exercise

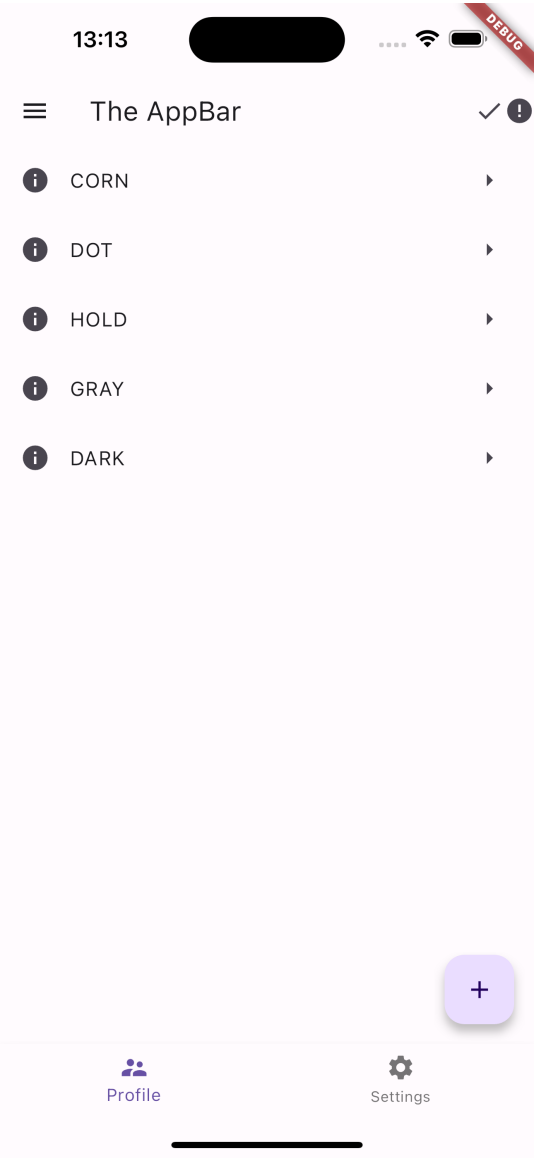
- Exercise 05.04 (medium)
  - Starting from the code of exercise 05.03, refactor MyApp so that it becomes a StatefulWidget
  - Add to the state of MyApp the current list of ListTiles to be displayed in the ListView
  - Replace the harcoded list of ListTiles with the “state” list just implemented
  - Note that when the app is reloaded or restarted, the ListView will look empty.

Exercise

- Exercise 05.05 (medium-hard)

  - Modify the code of exercise 05.04 so that when the user pushes the FloatingActionButton on the bottom right, a new ListTile is added to the ListView and displayed to the user.

  - Hint: ListView has not just 1 but 4 constructurs! ListView.builder could be of help. Look at the docs.



Outline

- Recap
- Some fundamental Flutter Widgets
- Layout in Flutter
- App#1: layout_basics
- App#2: scaffolding
- ListView

- Exercise
- **Homework**
- Resources

Homework

- Get familiar with UI constraints
- Get fluent in writing simple UIs

Outline

- Recap
- Some fundamental Flutter Widgets
- Layout in Flutter
- App#1: layout_basics
- App#2: scaffolding
- ListView

- Exercise
- Homework
- **Resources**

Resources

- Code repository of today's lesson and exercises solution
  - https://github.com/gcappon/bwthw/tree/master/lab_05-understanding_ui
- Material Widgets for UI
  - https://docs.flutter.dev/development/ui/widgets/material
- (Some) Widget for UI Catalog
  - https://docs.flutter.dev/development/ui/widgets
- Flutter Widget list
  - https://docs.flutter.dev/reference/widgets
- Understanding constraints
  - https://docs.flutter.dev/development/ui/layout/constraints
- Layouts in Flutter
  - https://docs.flutter.dev/development/ui/layout