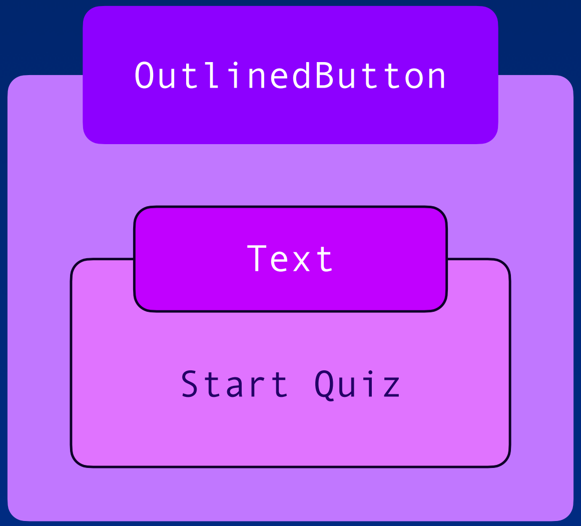
It’s all about Widgets!

Flutter UIs are created by combining & nesting Widgets



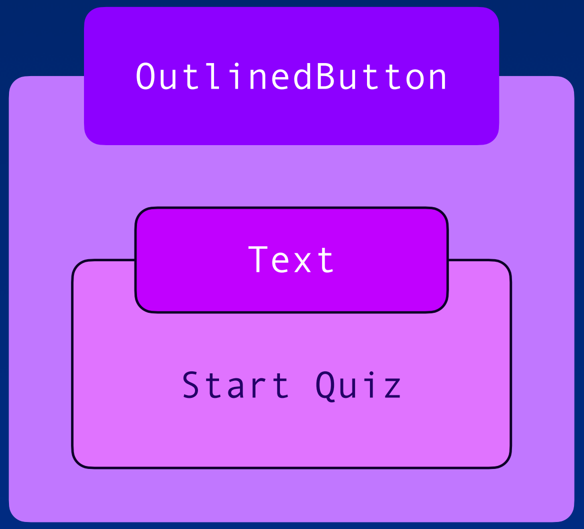
OutlinedButton

Text

Start Quiz

It’s all about Widgets!

Flutter UIs are created by combining & nesting Widgets



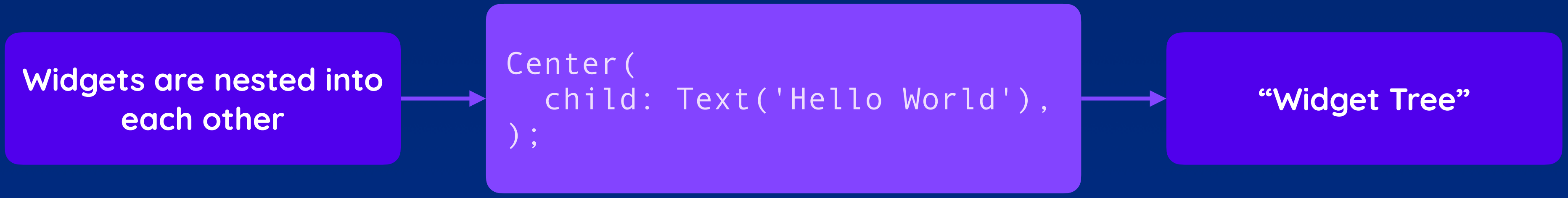
- ✓ **Flutter provides many built-in Widgets**
  e.g., Buttons, form inputs, layout widgets, …
- ✓ **You can also build your own Widgets**
  Based on the built-in Widgets

Flutter UIs Are Built With Widgets

!

When using Flutter, you **build your user interface with code**

A combination of widgets



Widgets are nested into each other

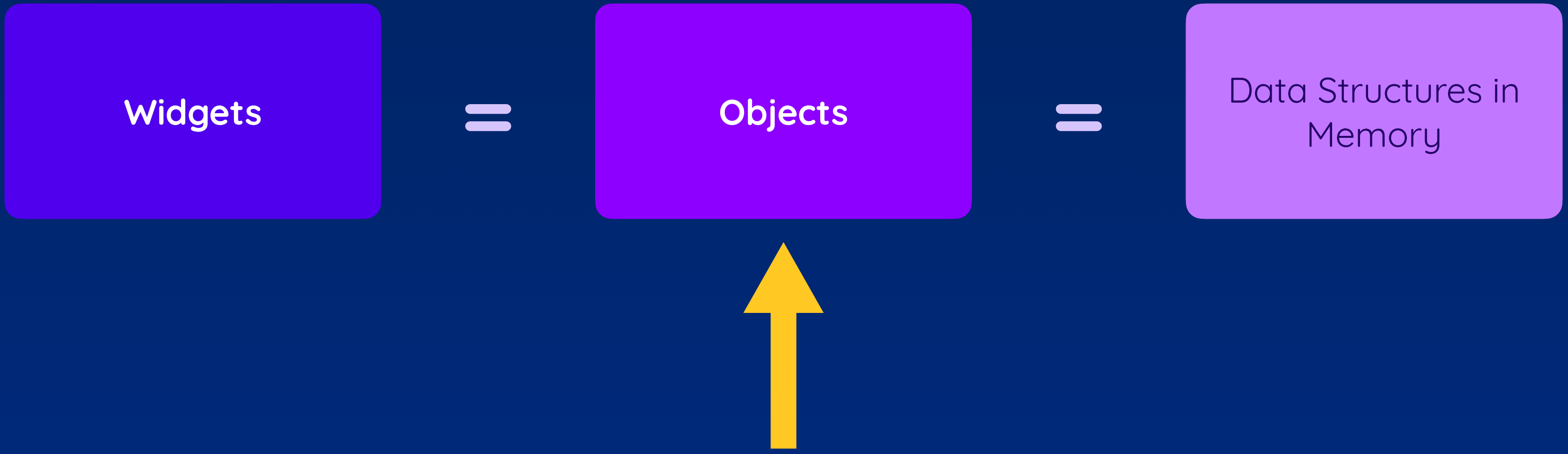
```
Center(
  child: Text('Hello World'),
);
```

“Widget Tree”

Built-in `Center` widget centers its content horizontally + vertically

Built-in `Text` widget displays some text on the screen

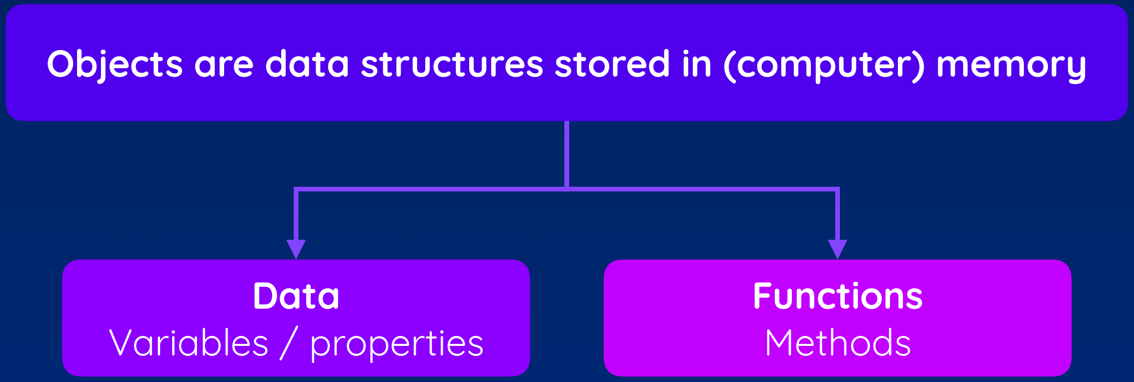
Widgets Are Objects



Widgets = Objects = Data Structures in Memory

Core Dart Concept!

Objects = Data Structures



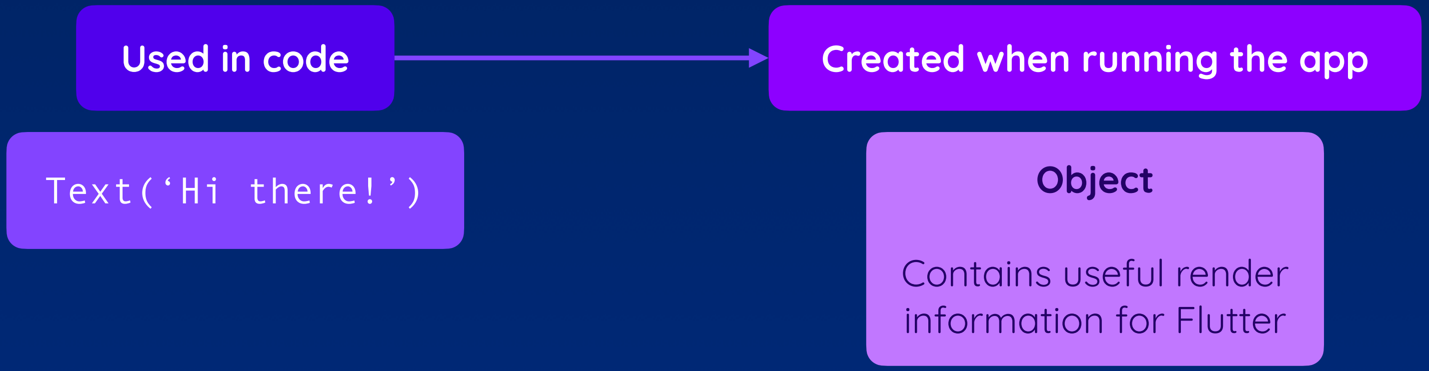
Objects are data structures stored in (computer) memory

Data
Variables / properties

Functions
Methods

Objects help with organizing data & separating logic

Widgets Are Complex Objects



Used in code

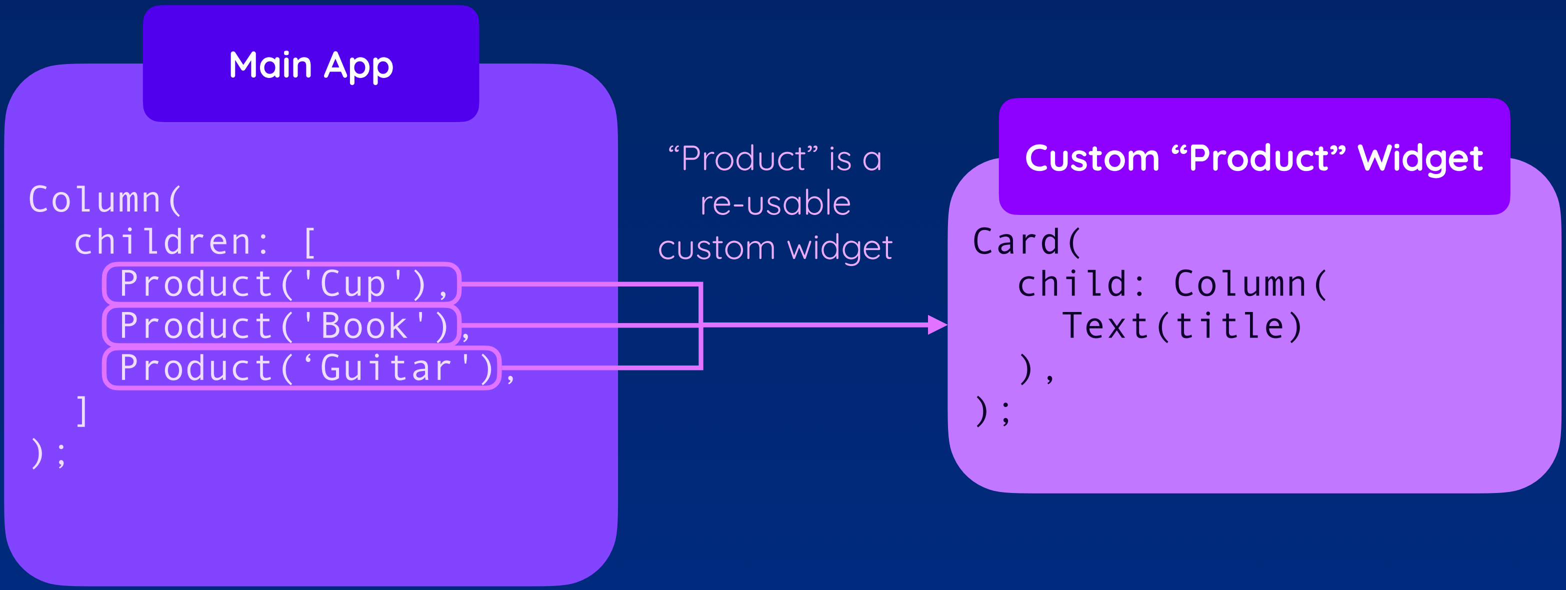
```
Text(‘Hi there!’)
```

Created when running the app

Object

Contains useful render information for Flutter

Building Custom Widgets



Main App

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

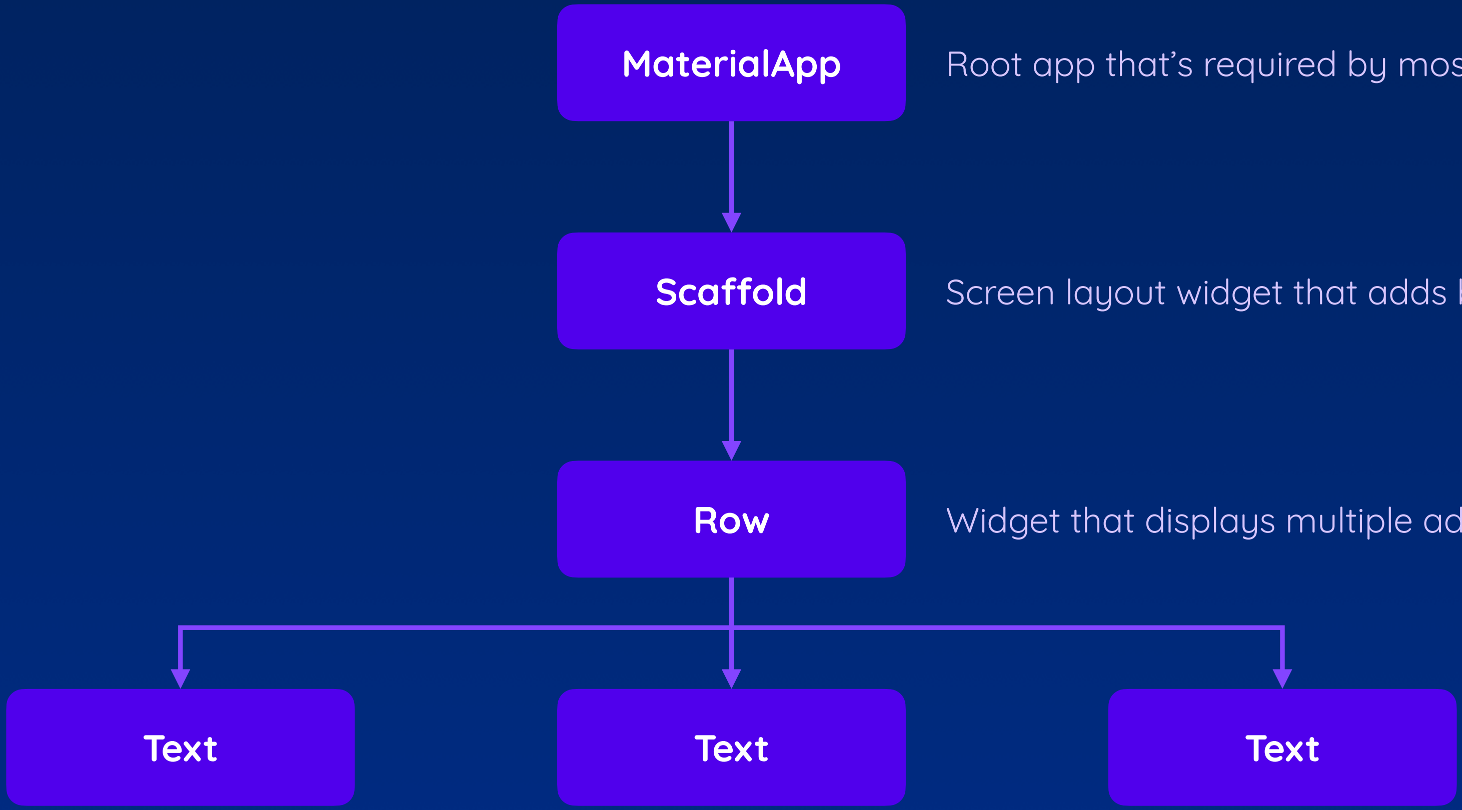
“Product” is a re-usable custom widget

Custom “Product” Widget

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

(of course, Column, Card & many other built-in widgets are explained throughout the course)

It’s a Widget Tree!



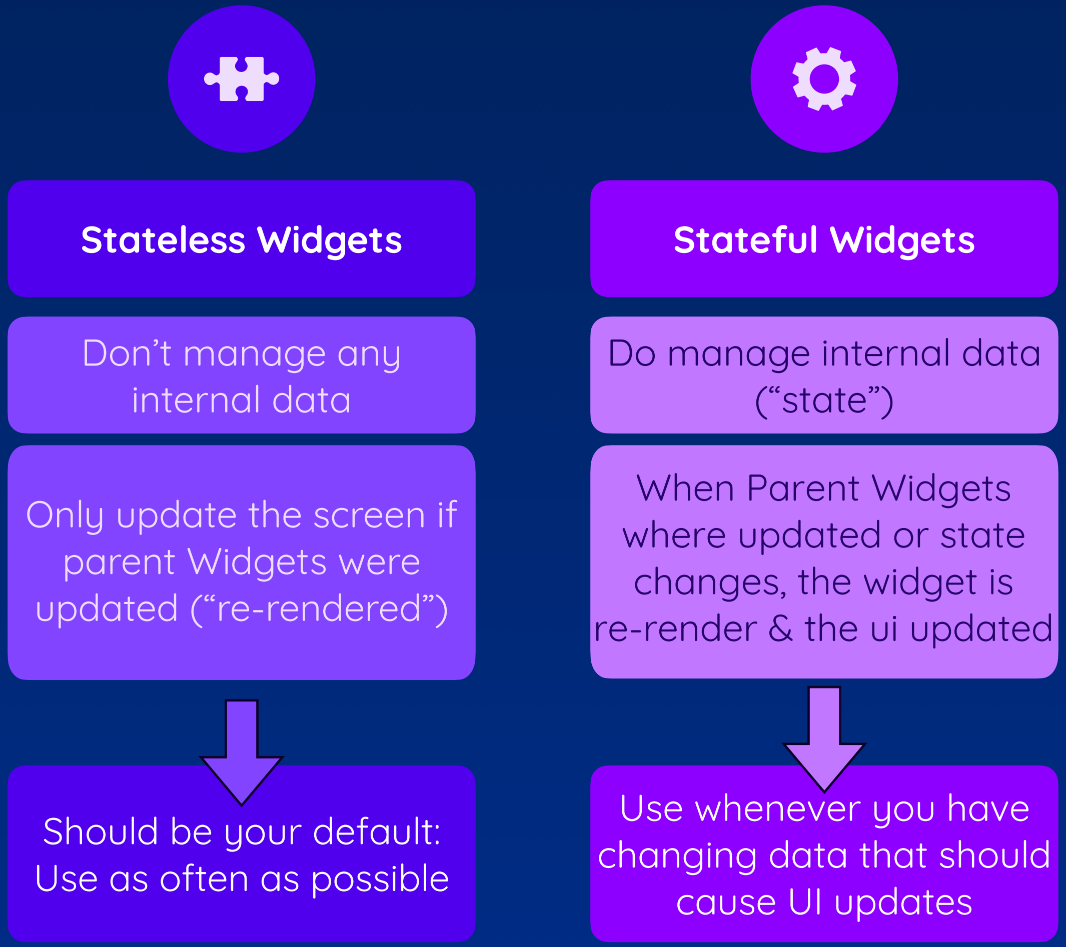
Root app that’s required by most other widgets

Screen layout widget that adds base styling & more

Widget that displays multiple adjacent child widgets

Widgets that display some text on the screen

Widget Types



Stateless Widgets

Don’t manage any internal data

Only update the screen if parent Widgets were updated (“re-rendered”)

Should be your default: Use as often as possible

Stateful Widgets

Do manage internal data (“state”)

When Parent Widgets where updated or state changes, the widget is re-render & the ui updated

Use whenever you have changing data that should cause UI updates

Basic Widgets

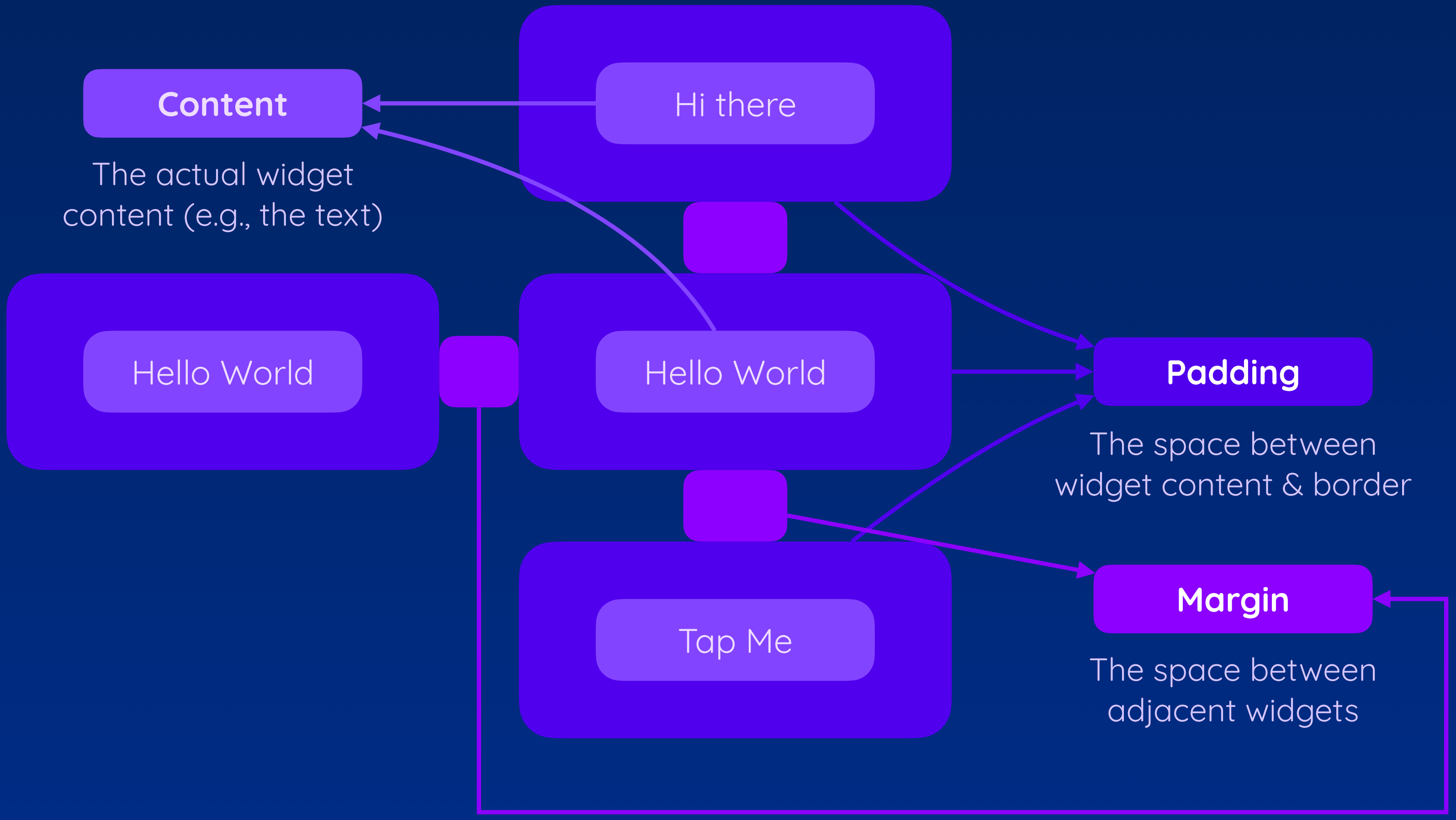
Container

```
Container(
 height: 40,
 width: 40,
 color: Colors.cyan,
 child:Text(title),
);
```

Text

```
Text(
 'any text here'
 style: TextStyle(
  fontSize:20,
 )
);
```

Margin & Padding



Content

The actual widget content (e.g., the text)

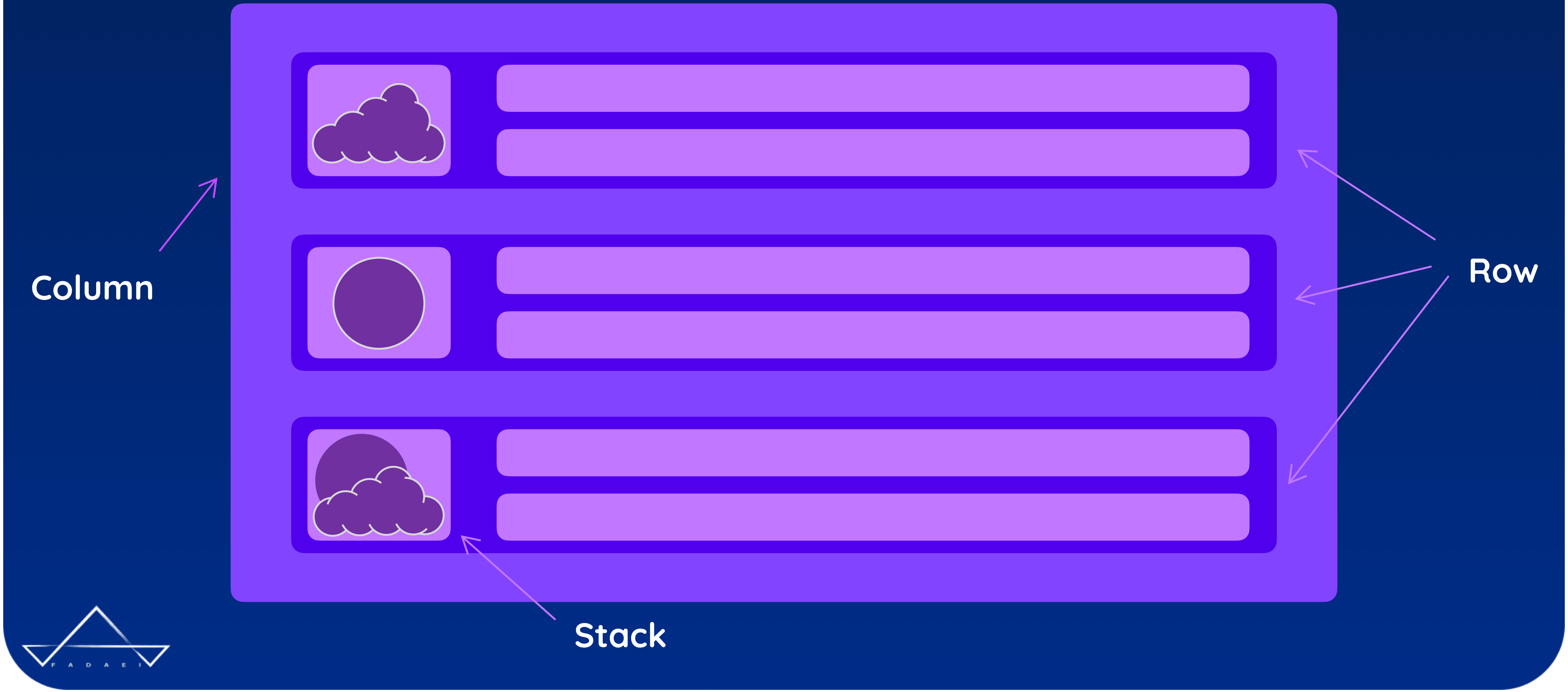
Padding

The space between widget content & border

Margin

The space between adjacent widgets

Layouts In Flutter



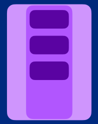
Column & Row

`Column()` & `Row()` can be used to place **multiple child widgets next to each other**

`Column()`

Main Axis: **Vertical** Axis

Cross Axis: Horizontal Axis



By default, occupies the **entire available height** but **only the width required** by its content (children)

`Row()`

Main Axis: **Horizontal** Axis

Cross Axis: Vertical Axis

By default, occupies the **entire available width** but **only the height required** by its content (children)

Layout Widgets part-1

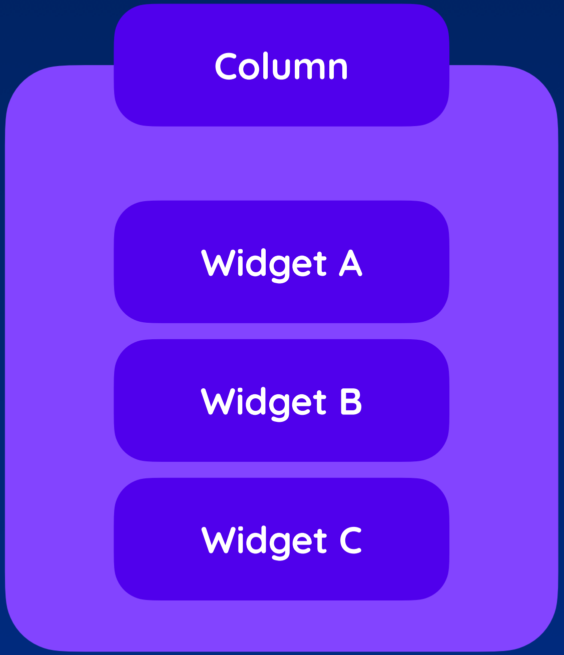
Column

```
Column(
 mainAxisAlignment: MainAxisAlignment.center,
 crossAxisAlignment: CrossAxisAlignment.center,
 chirdren: [
  ...
 ],
)
```

Row

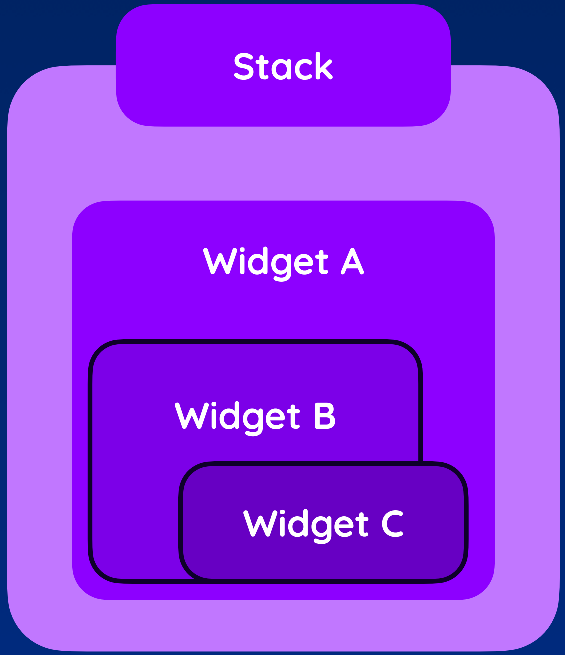
```
Column(
 mainAxisAlignment: MainAxisAlignment.center,
 crossAxisAlignment: CrossAxisAlignment.center,
 chirdren: [
  ...
 ],
)
```

The Stack Widget



Multiple widgets are positioned **next to each** other along the **Y-Axis**

e.g., a Text() above a TextField()



Multiple widgets are positioned **on top of each** other along the **Z-Axis**

e.g., a Text() on top of an Image()

Layout Widgets part-2

Stack

```
Stack(
 align: Alignment.center,
 chirdren: [
  Box(),
  Align(
   alignment: Alignment.center,
   child: Box(),
  ),
  Positioned(
   top: 100,
   left: 80,
   child: Box(),
  ),
 ],
)
```

Layout App

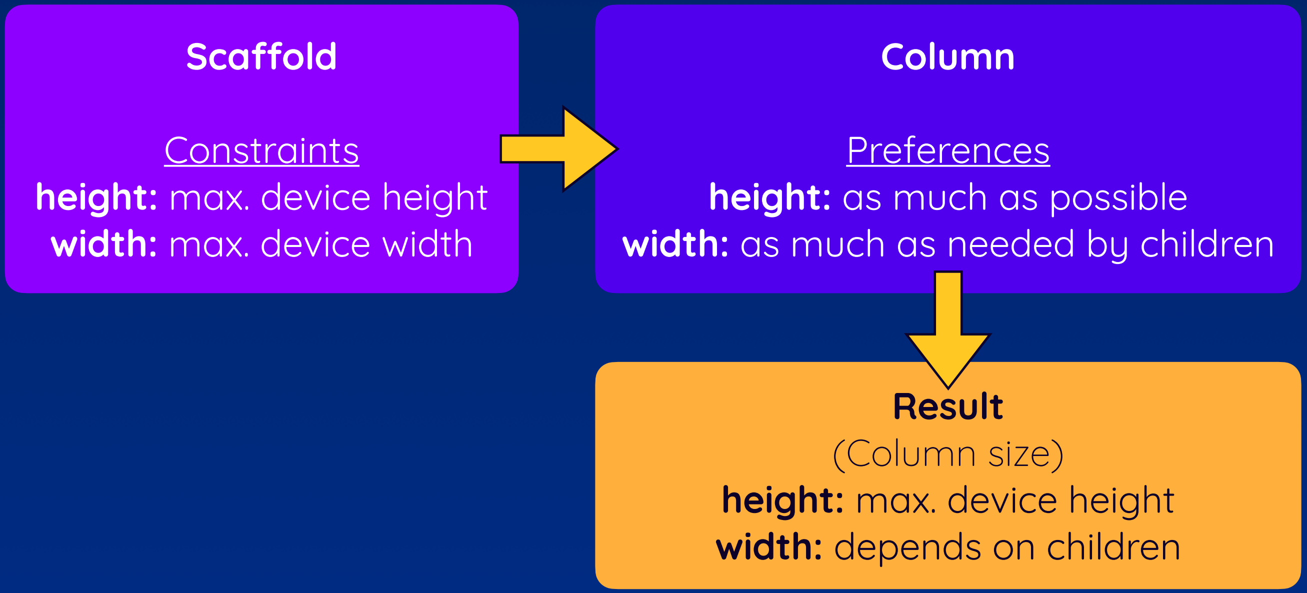
Responsive & Adaptive Apps

Adjusting Apps For Different Screen Sizes & Platforms

- Changing Layouts Based On Screen Sizes
- Detecting & Using Screen and Platform Information
- Building Adaptive Widgets

Understanding Widget Size Constraints

Widgets get sized based on their **size preferences** & parent widget **size constraints**



Scaffold

Constraints

height: max. device height

width: max. device width

Column

Preferences

height: as much as possible

width: as much as needed by children

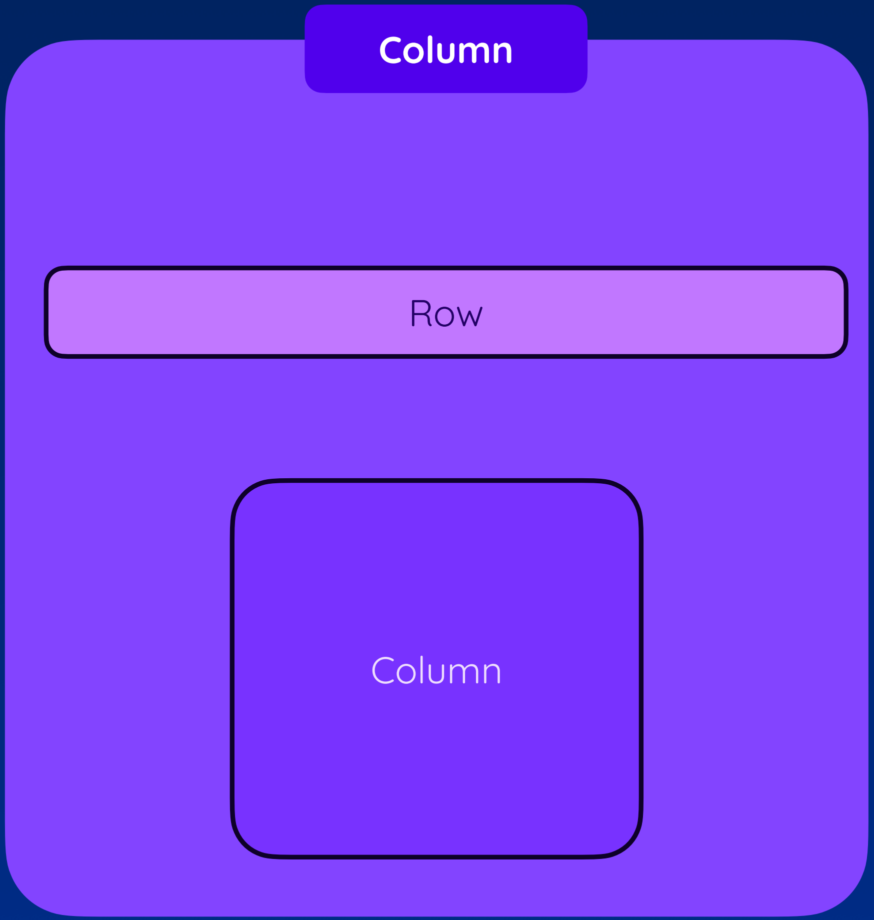
Result

(Column size)

height: max. device height

width: depends on children

Understanding Widget Size Constraints



Column constraints

Width: 0 → depends on children
Height: 0 → INFINITY

Row constraints

Width: 0 → INFINITY
Height: 0 → depends on children

Column constraints

Width: 0 → depends on children
Height: 0 → INFINITY

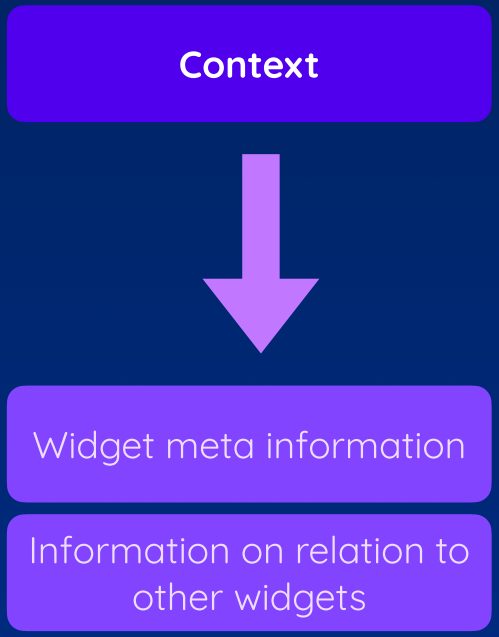
⚠ Problem: No height constraint from parent

Responsive Tools

MediaQuery.of(context)

Flexible/Expanded

Understanding Context



Context

Widget meta information

Information on relation to other widgets

Make Layout App Responsive

Building Adaptive, Cross-Platform Apps

You can use the same widgets & styling on Android & iOS!

But you can also adjust some widgets or styles

About Material Design

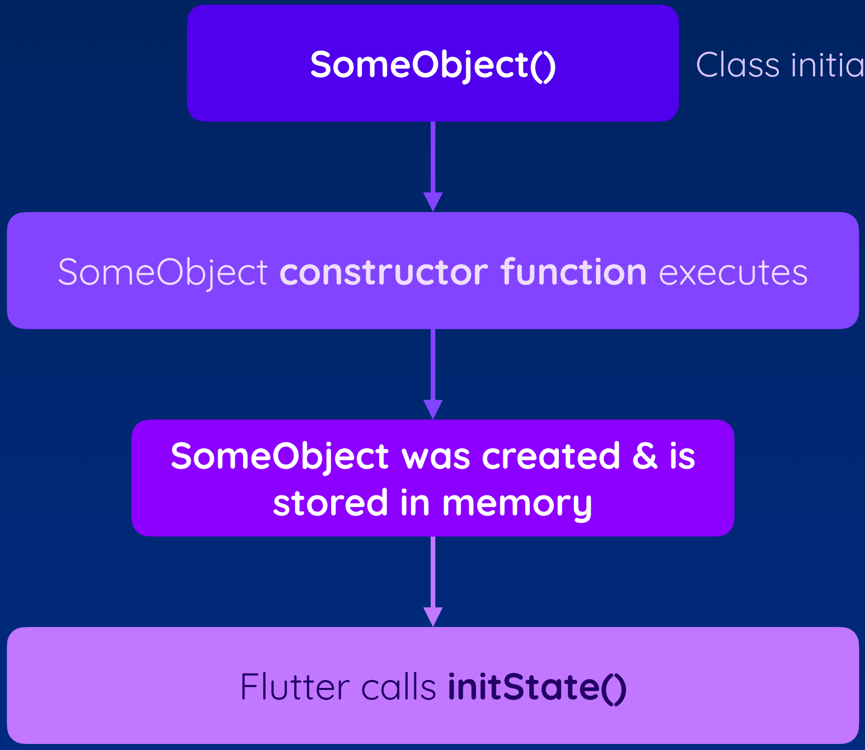
Google's flexible design system

A set of suggestions, rules & guidelines that help you build beautiful user interfaces

Highly customizable and extendable

Let's Code ToDo App

Using initState()



SomeObject()

Class initialization code

SomeObject **constructor function** executes

Instance variables & methods are created

SomeObject was created & is stored in memory

Flutter calls **initState()**

Can be used for further initilization tasks