Começando a usar Flutter & Dart

Flutter

Inclui:

- Reactive Framework
- 2D rendering Engine
- Várias ferramentas de desenvolvimento
- Ready-made widgets

Main Concepts

- Widgets
- States
- Material Design
- User Interaction and gestures
- Packages

Widget

- Is a description of part of a UI
- Flutter has no separate files for layout or customization, all code relating to the UI element is defined in the corresponding widget
- Every widget has a series of attributes and event-based functions
  - textTheme
  - Color
  - onPressed
  - Shape
  - ..
- Widgets have state, which changes as the user interacts with it

Flutter Dev Tools

- Hot reload
  - Changes in the code are reflected instantantaneosly on the UI (on device or emulator)
- Flutter Inspector
  - Relates and lets you navigate back and forth: pixel-level UI position – widget tree – source code
- Code auto-formatter (`dartfmt`)
  - Formats your code so that it becomes clearer to maintain

Dart Code Style

Identifiers:

- Type names: `MyType`
- Libraries/packages/directories/source files: `lowercae_with_underscores`
- Import prefixes: `lowercae_with_underscores`
- Other idenifiers: `lowerCamelCase`
- Constant names: `lowerCamelCase`
- Don't use prefix letters

```
class SliderMenu { ... }

class HttpRequest { ... }

typedef Predicate<T> = bool Function(T value);
```

Ordering:

- Place "dart:" imports before other imports
- Place "package:" imports before relative imports
- Prefer placing external "package:" imports before other imports
- The exports should go in a separate section after all imports
- Sort all imports and imports alphabetically

Dart Code Sytle

Formatting

- Use the Dart fomatter (`dartfmt`)
- Dartfmt cannot work miracles, so consider making your code more formatter-friendly.
  - shorten a local variable name or hoist out an expression into a new local variable
- Avoid lines longer than 80 chars
- Use curly braces for ALL control structures

Avoid dangling else clause:

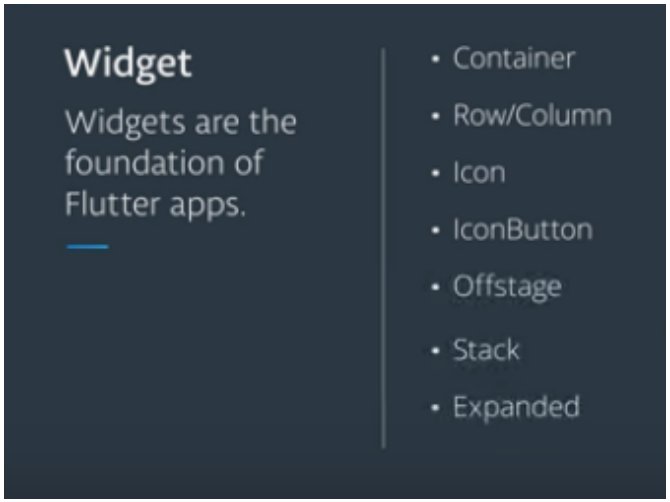
```
if (isWeekDay) {
  print('Bike to work!');
} else {
  print('Go dancing or read a book!');
}
```

When the “if” wraps to the next line:

```
if (overflowChars != other.overflowChars) {
  return overflowChars < other.overflowChars;
}
```


Widgets

- The entire UI is made of (nested) widgets ➔ widget tree
- Flutter comes with a rich set of predefined widgets



- Stateless widgets ➔ once properties are instantiated they cannot be changed (background-color, height, width)
  - Container widget – for subdividing screen, decorating other widgets, etc.
- Stateful widgets ➔ are the UI elements that embody an interaction

Widget catalog at: https://flutter.io/docs/development/ui/widgets

Widgets – 3 Princípios

Todos os widgets devem obedecer a 3 princípios:

- Aparência
  - Devem ser visualmente agradáveos, bonitos, preservem o “Look & Feel” de cada plataforma (se desejado). L&F do Android difere daquele do Cuperino / ioS
  - Widgets com estilo Cupertino (Apple) e Material Design (Google) para versões antigas

- Alta performance
  - Quando são acionados devem produzir uma reacão rápida e suave, incluindo transições e animações

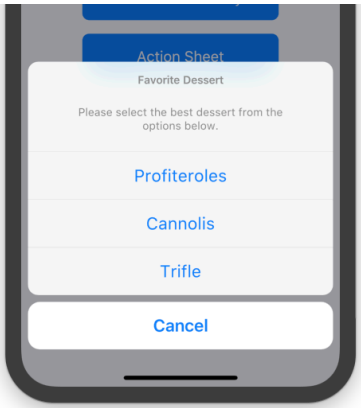
- Extensibilidade e Customização
  - Posibilidade de customizar/modificar quae tudo no wirdget para manter o estilo & branding do aplicativo

Isso é possivel porque nõa existe mais tradução/ponte: **os widgets são renderizados pelo próprio aplicativo** e não na plataforma

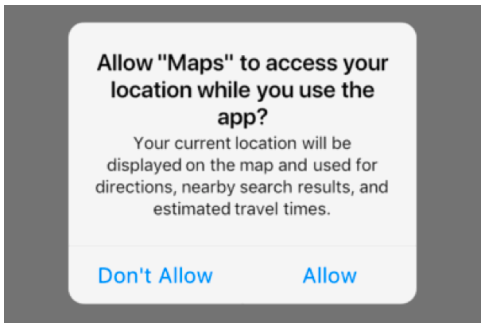
Widgets - Catálogo

Widget Catalog: https://flutter.dev/docs/development/ui/widgets

- Existem wirdegts para TUDO MESMO, desde estrutura/ layout de uma tela, icones, buttons, animações, Input, Semântica de widgets, Modelos de Interação, scrolling, display de texto, theme style, etc.
- Cada widget em seu app é uma classe Dart que estende uma classe que vem em um package (exemplo: package:flutter/material.dart)
- Exemplos (Cupertino iOS)

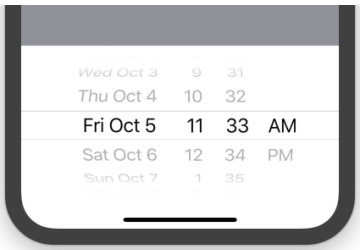


Bottom action sheet



Alert Dialog

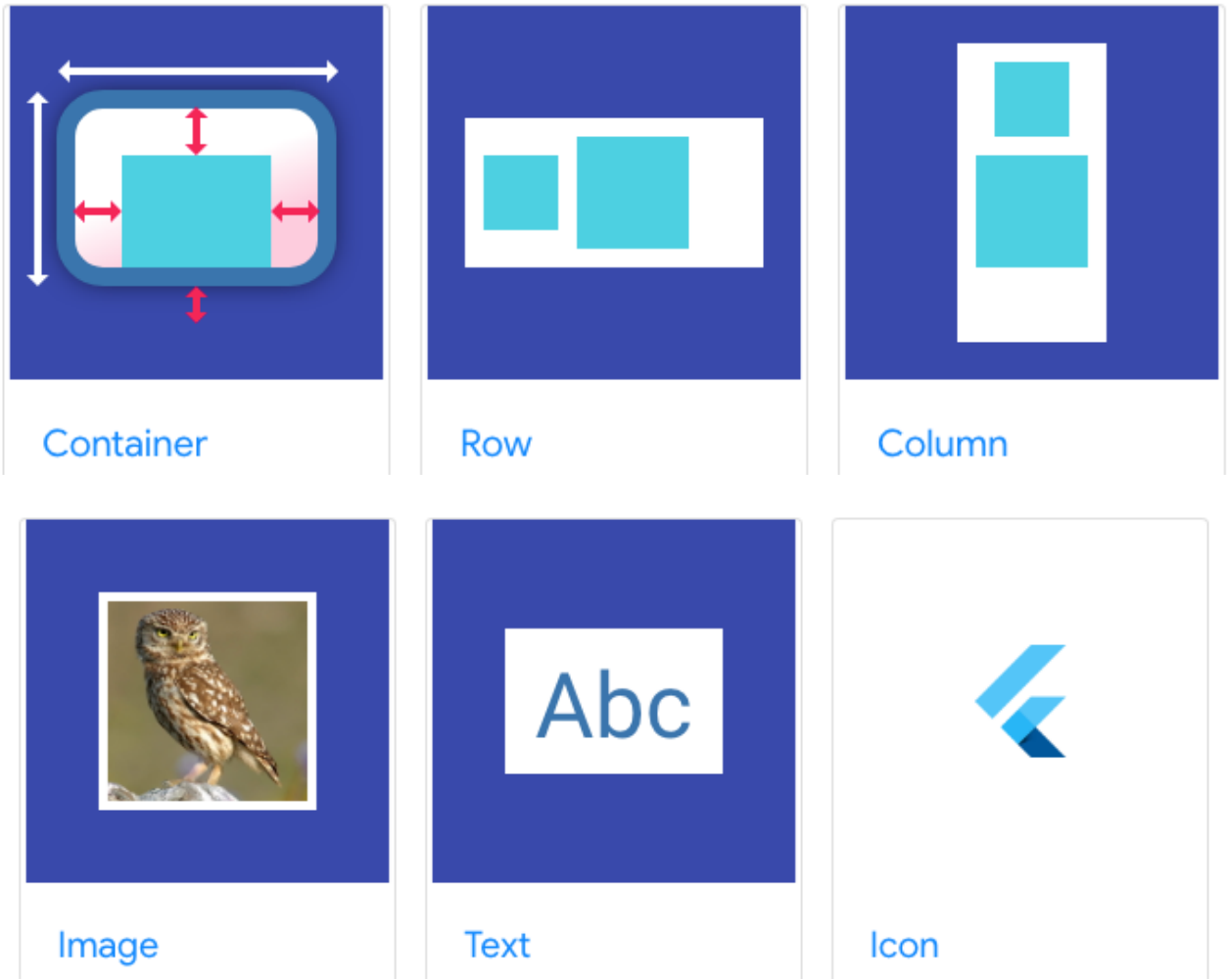
Activity Indicator/s



Date Pickers

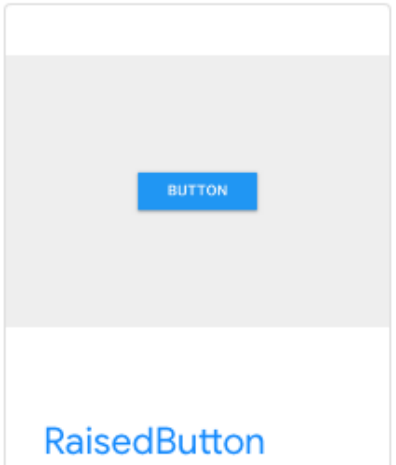
Widgets – the essential ones

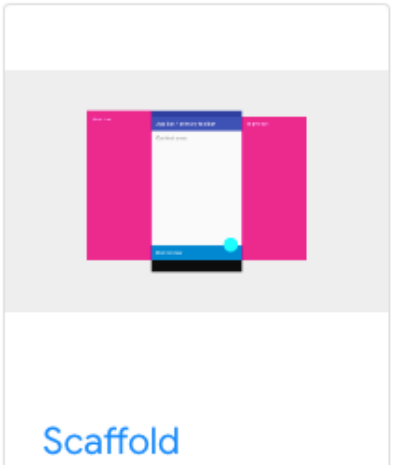
- Necessários para quase toda app (Basic Widgets)

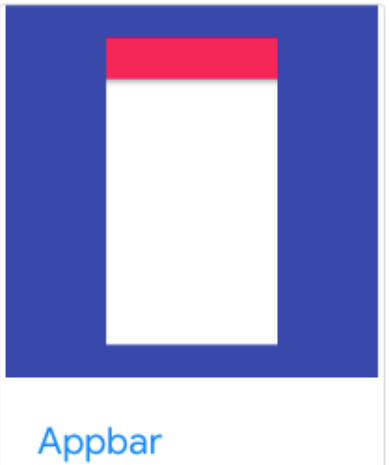


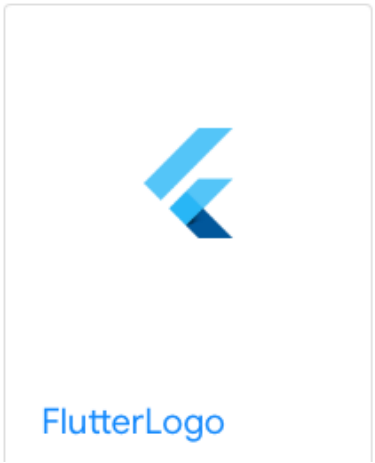
Widgets – the essential ones

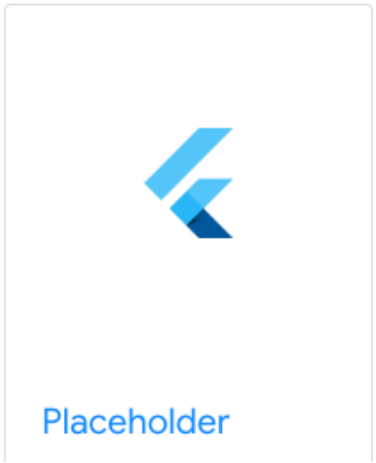
- Necessários para quase toda app (Basic Widgets)







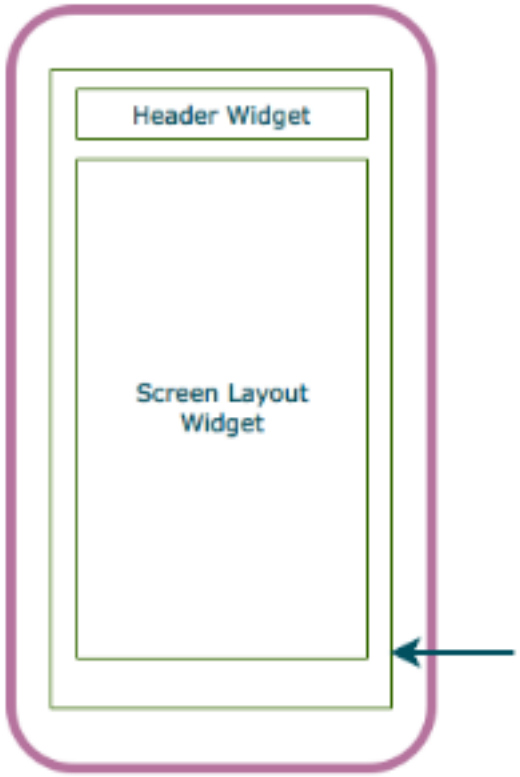




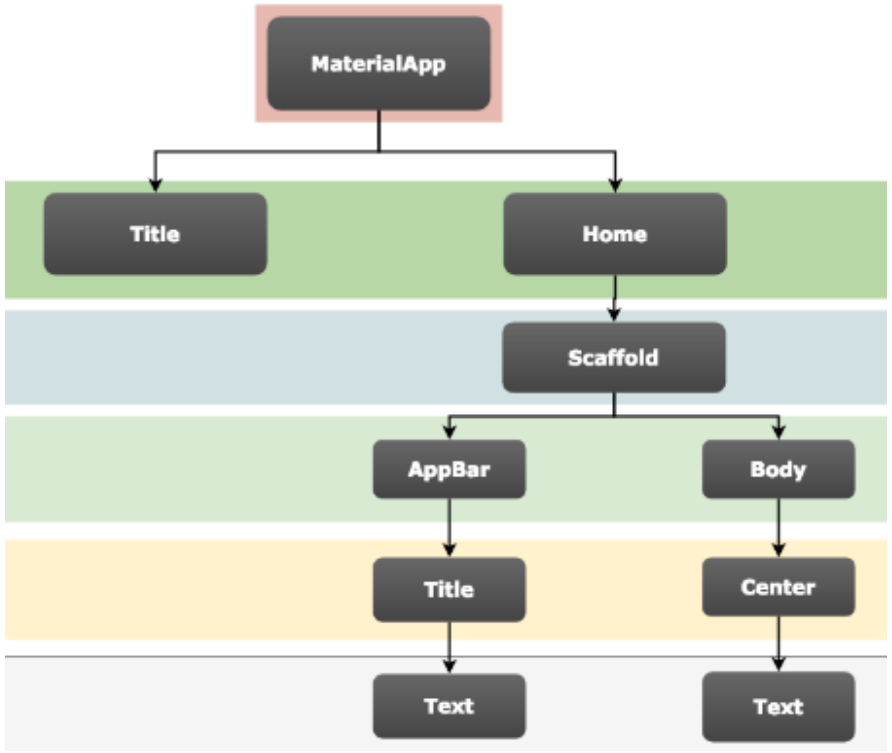
Scaffold: Implementa a estrutura basica de layout do Material Design.
Com APIs para posicionar e mostrar drawers, bottom sheets, snackbars, etc.

Widgets - Hierarchy

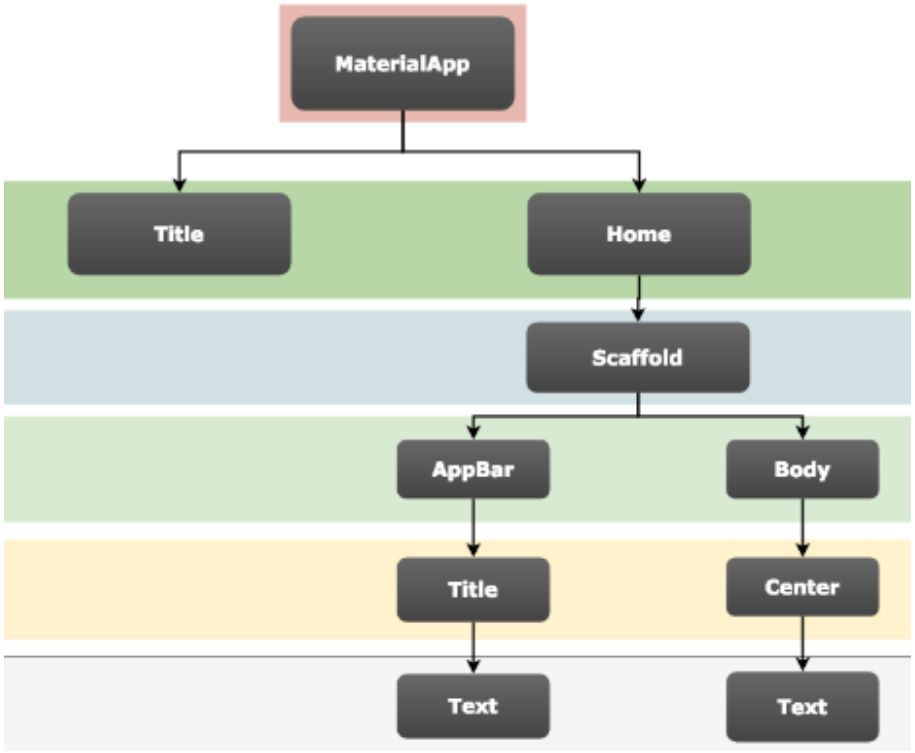
A very basic app



The widget hierarchy



Widgets - Hierarchy




```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext ctxt) {
    return new MaterialApp(
      title: "MySampleApplication",
      home: new Scaffold(
        appBar: new AppBar(
          title: new Text("Hello Flutter App"),
        ),
        body: new Center(
          child: new Text("Hello Flutter"),
        ),
      ),
    );
  }
}
```

Hello Flutter

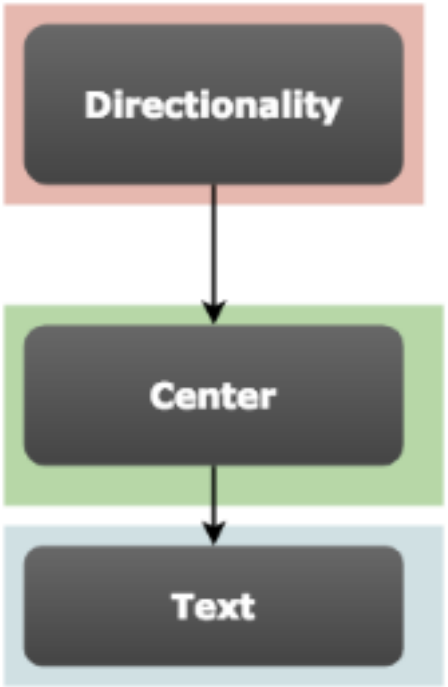
- Usaremos um Container Widget chamado Directionality e um Text Widget
- Precisamos importar o pacote material design (pois é ele que contém os widgets)

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

void main() => runApp(new MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext ctxt) {
    return new Directionality(
        textDirection: TextDirection.ltr,
        child: new Text("Hello Flutter")
    );
  }
}
```

- MyApp é uma widget que vai criar o layout da tela.
- Criamos um container Directionality que terá um sub-widget chamado 'Text'
- Cada widget them um método "build" e retorna um widget.

Hello Flutter

- Se quisermos centralizar o texto, criamos um sub-widget filho de Directionality, e pai do widgetText.




```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext ctxt) {
    return new Directionality(
        textDirection: TextDirection.ltr,
        child: new Center(
                child: new Text("Hello Flutter"),
         )
    );
  }
}
```

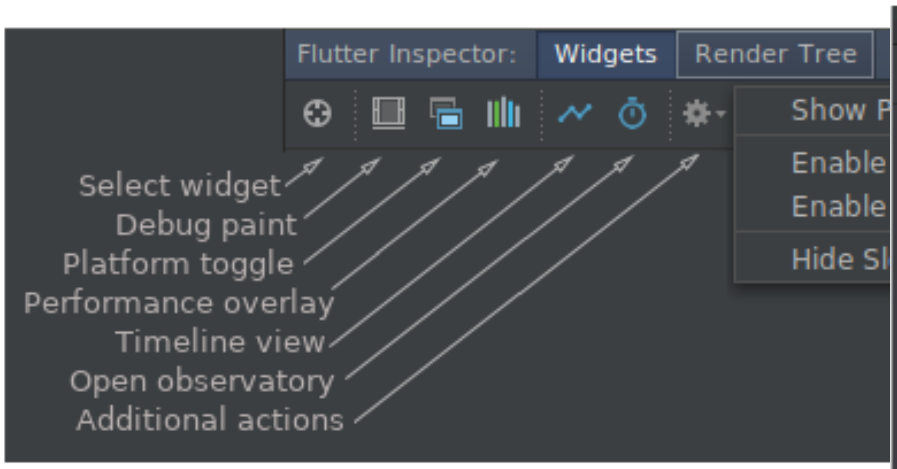
Widget Inspector

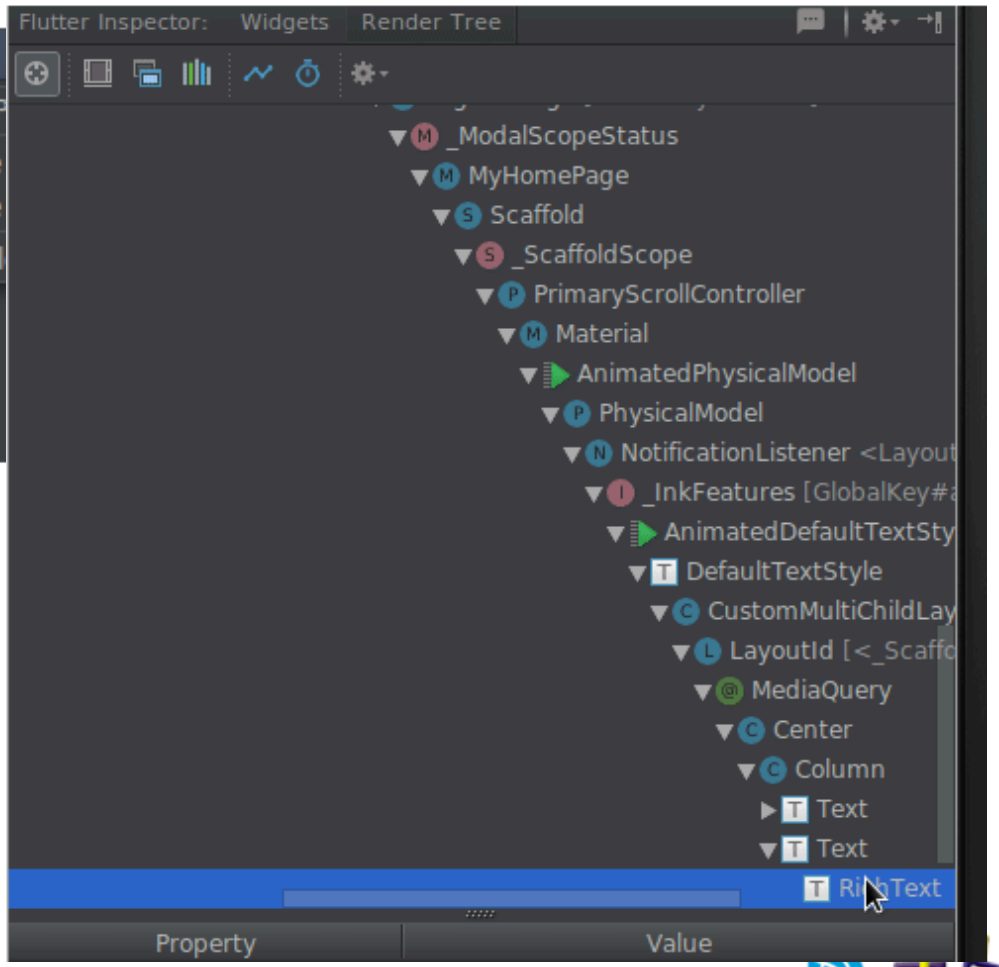
- Flutter framework uses widgets as the core building block for anything from controls (text, buttons, toggles, etc.) to layout (centering, padding, rows, columns, etc.).

- **Widget inspector** is powerful tool for visualizing and exploring Flutter widget trees.
- It can be helpful when:
  - Understanding existing layouts
  - Diagnosing layout issues

- The inspector is currently available in the Flutter plugin for Android Studio, or IntelliJ IDEA.

Widget Inspector

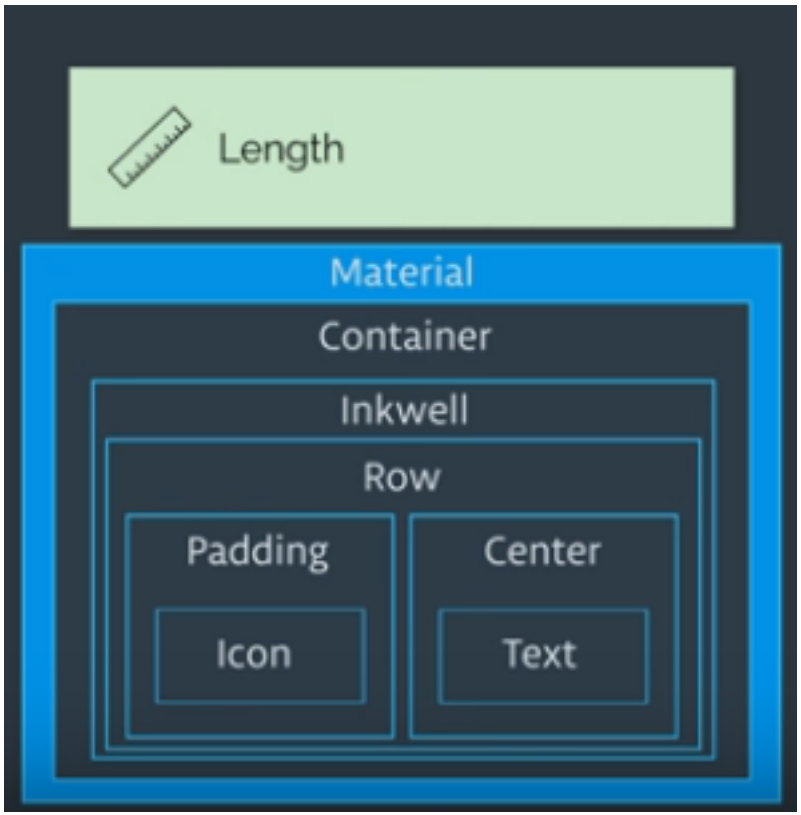




https://flutter.io/docs/development/tools/inspector

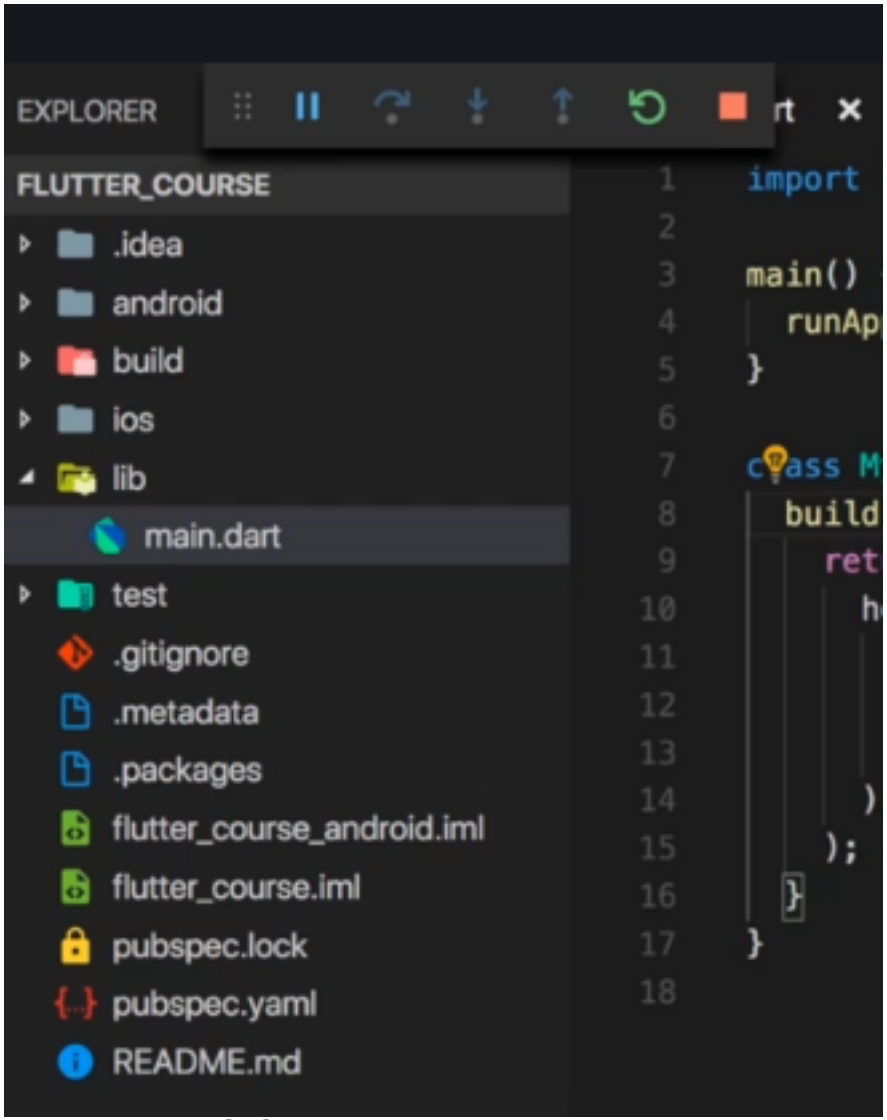
Widgets to set UI characteristics

Screen is list of container widgets



Nested/ hierarchy of widgets:
Inkwell:: animation when one taps on it
Each widget may has its own padding (those are defined through edge inserts)

Um projeto de App Flutter



No VS Code com plugin Flutter

- lib/main.dart é onde é colocado o seu código dart
- .idea: usados pelo VS Code ou outras IDEs (não alterar)
- android e ios: código nativo gerado a partir do dart (não alterar)
- build: arquivos gerados no processo de build multi-plataforma (não alterar)
- test: pode ser usado para criar testes automatizados
  - .gitignore: para controle de versão no git
  - .metadata, .packages, etc.: arquivos de configuração
  - pubspec.yaml: descrição das dependências de módulos de código nativos