Chapter 3:

Flutter overview

Plan

1. **Introduction to Flutter.**
    - What is Flutter ? And What is it good for ?
    - Flutter’s features.
    - What exactly is Flutter ? Understanding Flutter as Framework.
2. **Exploring Flutter architecture.**
    - Flutter layers.
    - Specific design languages: Cupertino & Material design.
    - Widgets and concept of "State".
    - Rendering.
    - Gesture, Painting, Animation.
    - Foundation.

Plan

1. **Introduction to Flutter.**
    - What is Flutter ? And What is it good for ?
    - Flutter's features.
    - What exactly is Flutter ? Understanding Flutter as Framework.
2. **Exploring Flutter architecture.**
    - Flutter layers.
    - Specific design languages: Cupertino & Material design.
    - Widgets and concept of "State".
    - Rendering.
    - Gesture, Painting, Animation.
    - Foundation.

Introduction to Flutter

- **What is Flutter ?**

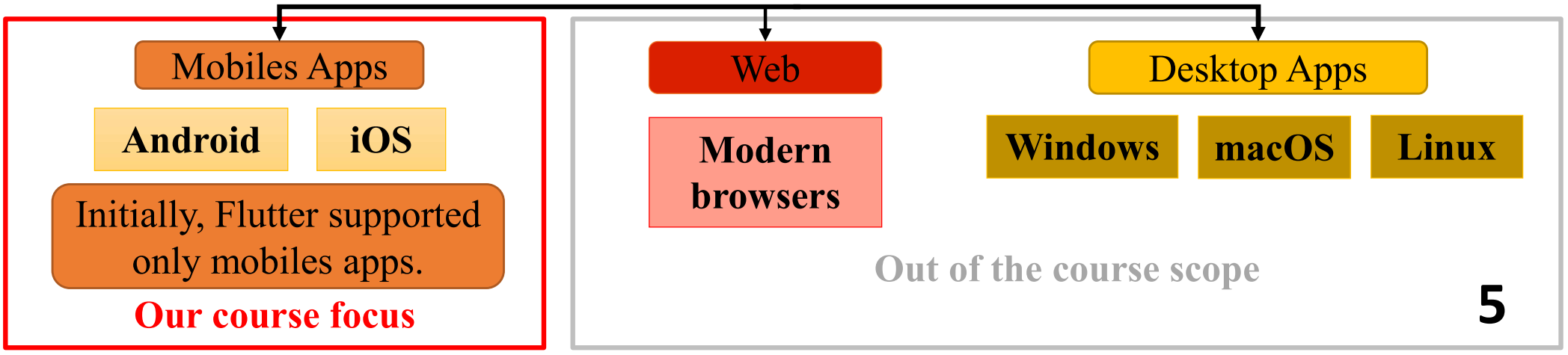
  - An **open source** framework (software development kit SDK) created by Google.

- **When was Flutter introduced ?**

  - **Flutter** was released in **May 2017** (the alpha version).
  - Announcement to Mobile World Congrees in **Feb 2018** (the beta version).
  - It becomes popular in **December 2018** (the official stable version 1).
  - Flutter version 2 in **March 2021**.

Introduction to Flutter

- **Why is Flutter used ?**
  - Creating and building **high performance (natively-compiled)** mobile, desktop and web applications across **multiple platforms** (Android and iOS as mobile OS and Windows, Linux and macOS as Desktop OS).
- **Target platforms supported by Flutter :**



Introduction to Flutter

- **What is Flutter good for ?**

  - A **flexible** technology to create a wide variety of apps both small for startups and large for enterprises.

- **Flutter is NOT a programming language !**

It is a framework for building user interfaces with Dart.

Framework : a collection of packages & utility functions that may be used in the code.

Dart : a programming language developed by Google. **Main usage**: Flutter app development.

Introduction to Flutter

- **Features of Flutter :**

  - ➢ Cross-platform development.
  - ➢ Fast and low-cost development.
  - ➢ Attractive and customizable UI.
  - ➢ High performance (native) and awsome apps.
  - ➢ Large community .
  - ➢ Open-source and free.
  - ➢ Hot reload (quick & ease app refresh).

See instantly the changes make to the code reflected in the UI

Introduction to Flutter

- **Concrelety, what is axactly Flutter ?**

Introduction to Flutter

UI Framework

Flutter allows you to build **multi-platform apps** based on **one single codebase** and programming language.

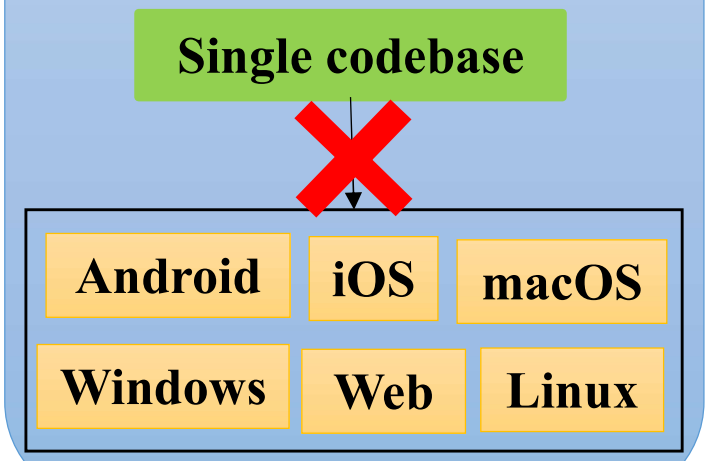
Collection of tools

Code packages & utility functions for writing **cross-platform** app code.

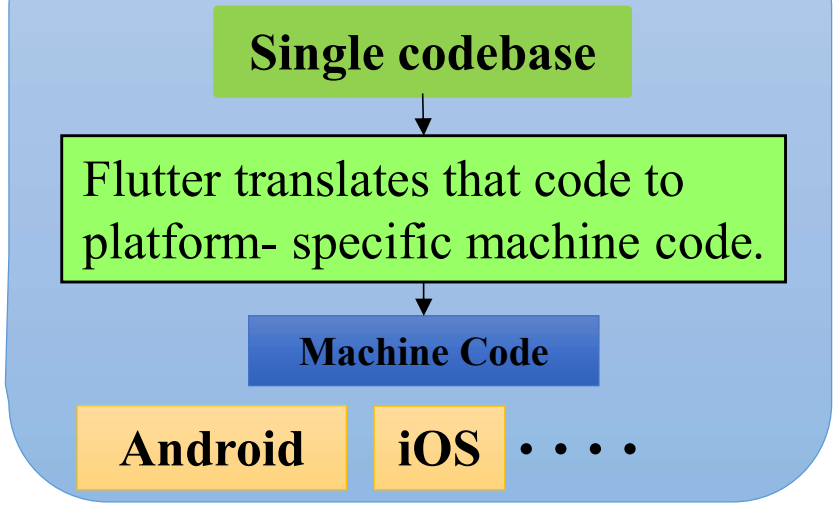
+

CLI(*) & software that helps with developping, testing & **building** cross-platform apps.

One Codebase, Multiple Apps



From Flutter Code to Platform Code



(*) Command Line Interface

Plan

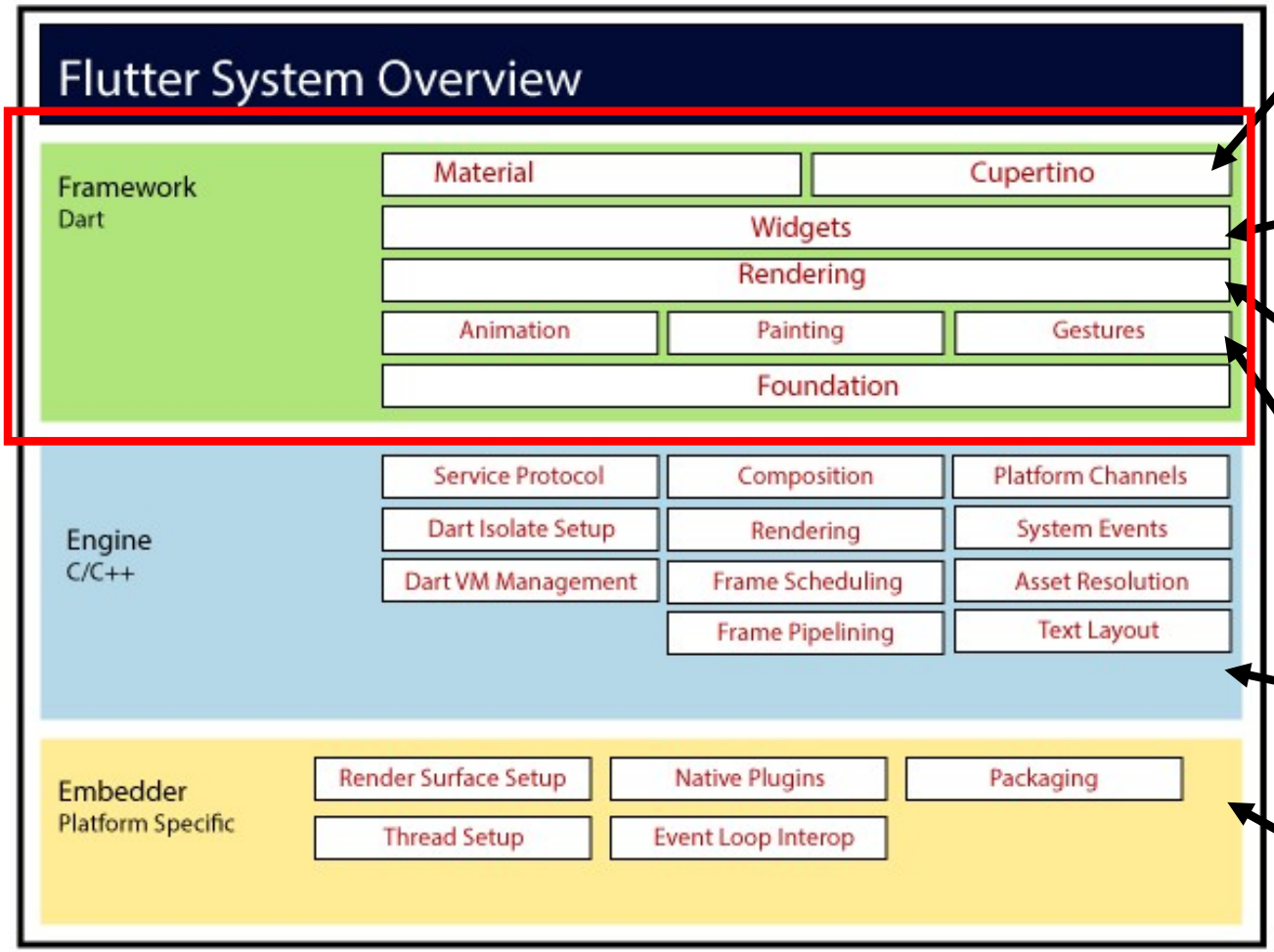
1. **Introduction to Flutter.**
    - What is Flutter ? And What is it good for ?
    - Flutter's features.
    - What exactly is Flutter ? Understanding Flutter as Framework.
2. **Exploring Flutter architecture.**
    - Flutter layers.
    - Specific design languages: Cupertino & Material design.
    - Widgets and concept of "State".
    - Rendering.
    - Gesture, Painting, Animation.
    - Foundation.

Plan

1. **Introduction to Flutter.**
    - What is Flutter ? And What is it good for ?
    - Flutter’s features.
    - What exactly is Flutter ? Understanding Flutter as Framework.
2. **Exploring Flutter architecture.**
    - Flutter layers.
    - Specific design languages: Cupertino & Material design.
    - Widgets and concept of "State".
    - Rendering.
    - Gesture, Painting, Animation.
    - Foundation.

Flutter architecture

- **Layers of Flutter:** categories with decreasing hierarchical level of complexity.



Widget specific to the OS of the device.

Natif Flutter widget (UI component, gesture detectors, ..).

Visible components.

Animations, image flow, gestures.

Library to support primitives, I/O, Dart runtime, …

Turns code into app or library

Plan

1. **Introduction to Flutter.**
   - What is Flutter ? And What is it good for ?
   - Flutter's features.
   - What exactly is Flutter ? Understanding Flutter as Framework.

2. **Exploring Flutter architecture.**

   - Flutter layers.
   - Specific design languages: Cupertino & Material design.
   - Widgets and concept of "State".
   - Rendering.
   - Gesture. (painting, animation).
   - Foundation.

Specific design language

- **Flutter UI Design:** is the graphical and interactive components of a Flutter app (e.g. layouts, colors, animations, etc.).

Every Flutter UI design **employs** a **design language.**

- **Design language:** is a set of standards and rules that guide the visual and interactive design of an app.

- Flutter provides excellent support for two major design languages : **Material Desing** & **Cupertino.**

Specific design language

- **Material Desing:** is designed by **Google** and is used on **Android** platforms.

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

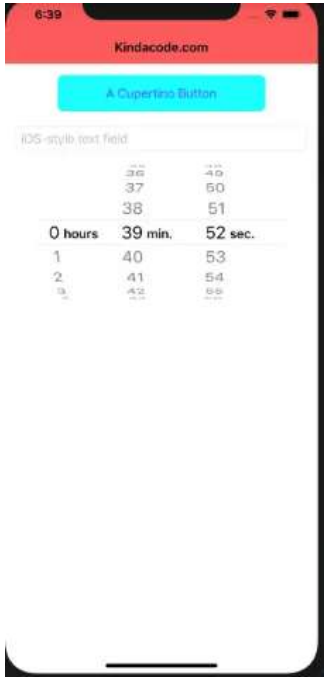
- **Cupertino Design:** is an **iOS** interface guideline designed by **Apple**.

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

- Flutter provides a wide range of material components. While creating a new Flutter project, developers can maintain platform consistency or **mix** and match components from the two design systems => **custom look & feel.**

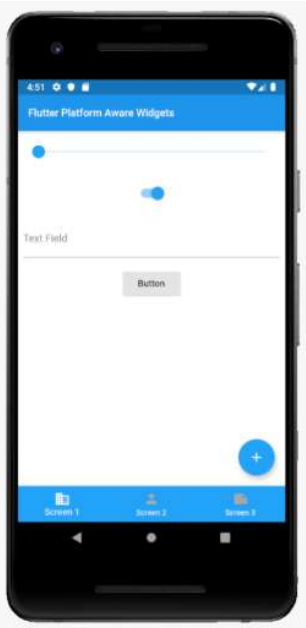
Specific design language

- **Material Desin vs Cupertino Desing:** different components, apparences, interaction patterns, navigation styles, etc.



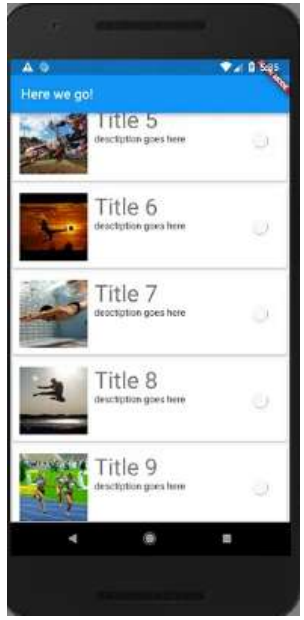
iOS style on Android platform

iOS



Android

iOS

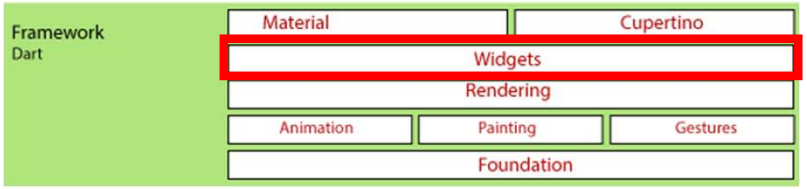


Android

Plan

1. **Introduction to Flutter.**
   - What is Flutter ? And What is it good for ?
   - Flutter's features.
   - What exactly is Flutter ? Understanding Flutter as Framework.

2. **Exploring Flutter architecture.**



   - Flutter layers.
   - Specific design languages: Cupertino & Material design.
   - Widgets and concept of "State".
   - Rendering.
   - Gesture, Painting, Animation.
   - Foundation.

Widgets and concept of "State"

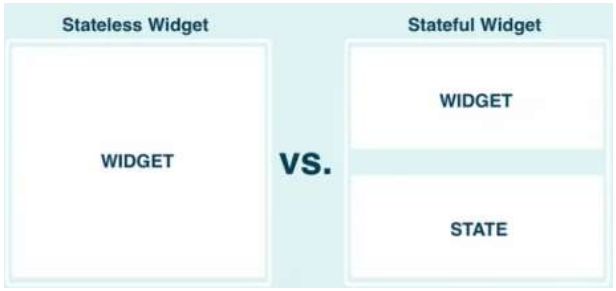
- **Widget:** is a key component of the UI. It's used to describe the structure and layout of UI elements (button, image, text, list, …).

Once created, its properties can't be modified.

- It acts as a UI for the user to interact with the app.
- Divisible, customizable and reusable, elements.
- It's **immutable** **(unchanging).**
- New widgets can be composed out of existing ones.
- The root widget is called **MaterialApp/CupertinoApp**.

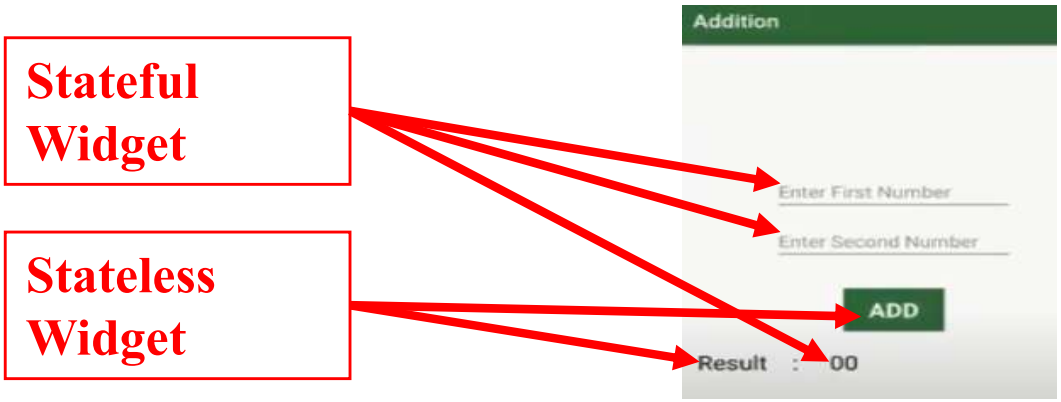
Widgets and concept of "State"

- **Widget:** two types of widgets.



1. **StatelessWidget:** the widgets whose state **can NOT** be altered once they are built.
2. **StatefulWidget:** the widgets whose state **can** be altered once they are built. **States** are **mutable**.

- **Example:**



Widgets and concept of "State"

- **Widgets:** two types of widgets.

  - A ***stateful*** widget is a dynamic widget that can change its appearance in response to user interactions or to data reception.

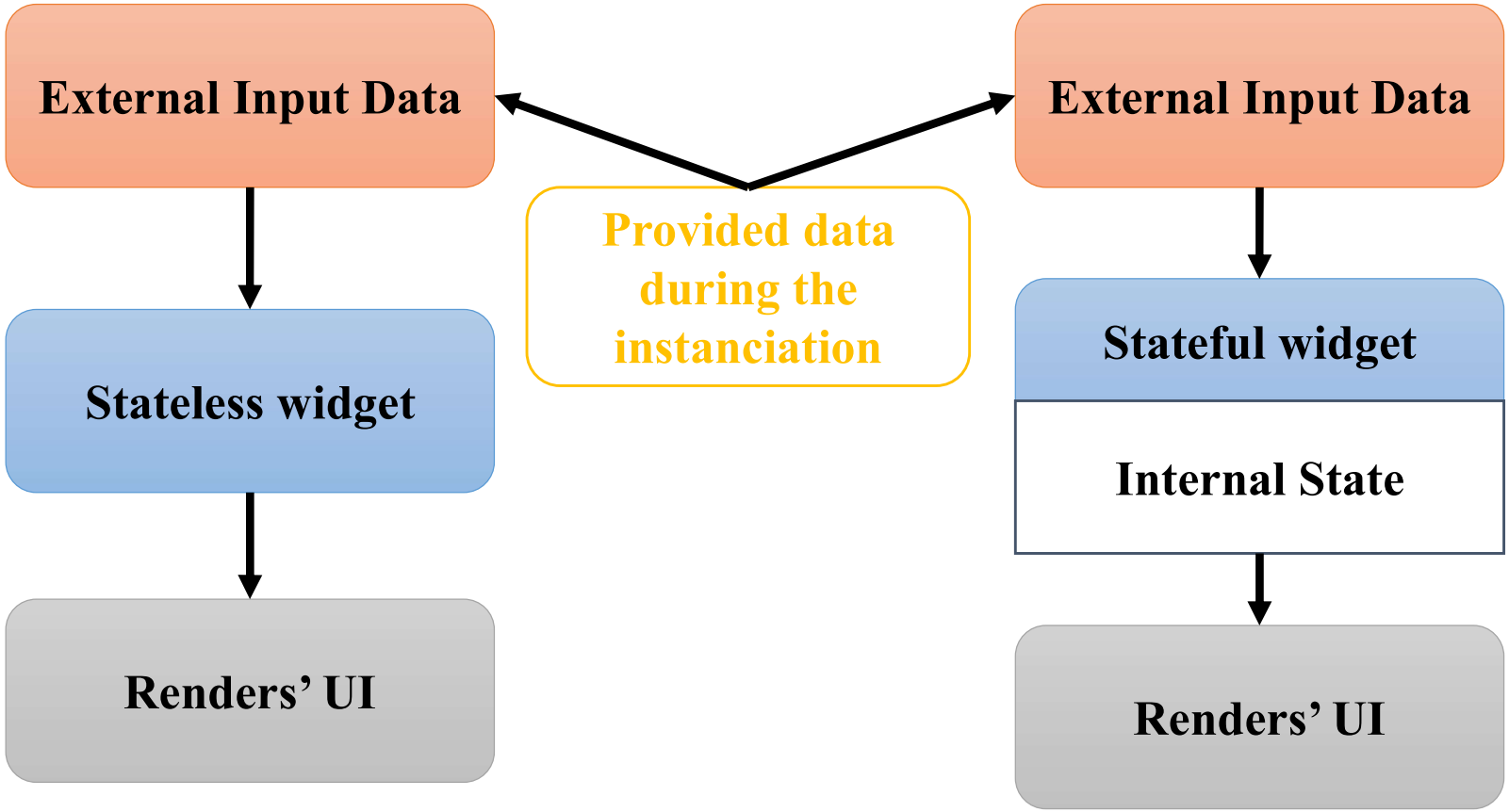
  - Stateful widgets subclass **StatefulWidget.**

  - A widget's **State** consists of values that can change. It is stored in a **State** object, separating the widget's state from its appearance.

  - **Examples**: *Checkbox, Radio, Slider, Form, TextField … State = Current value of slider.*

Widgets and concept of "State"

- **Widgets:** two types of widgets.

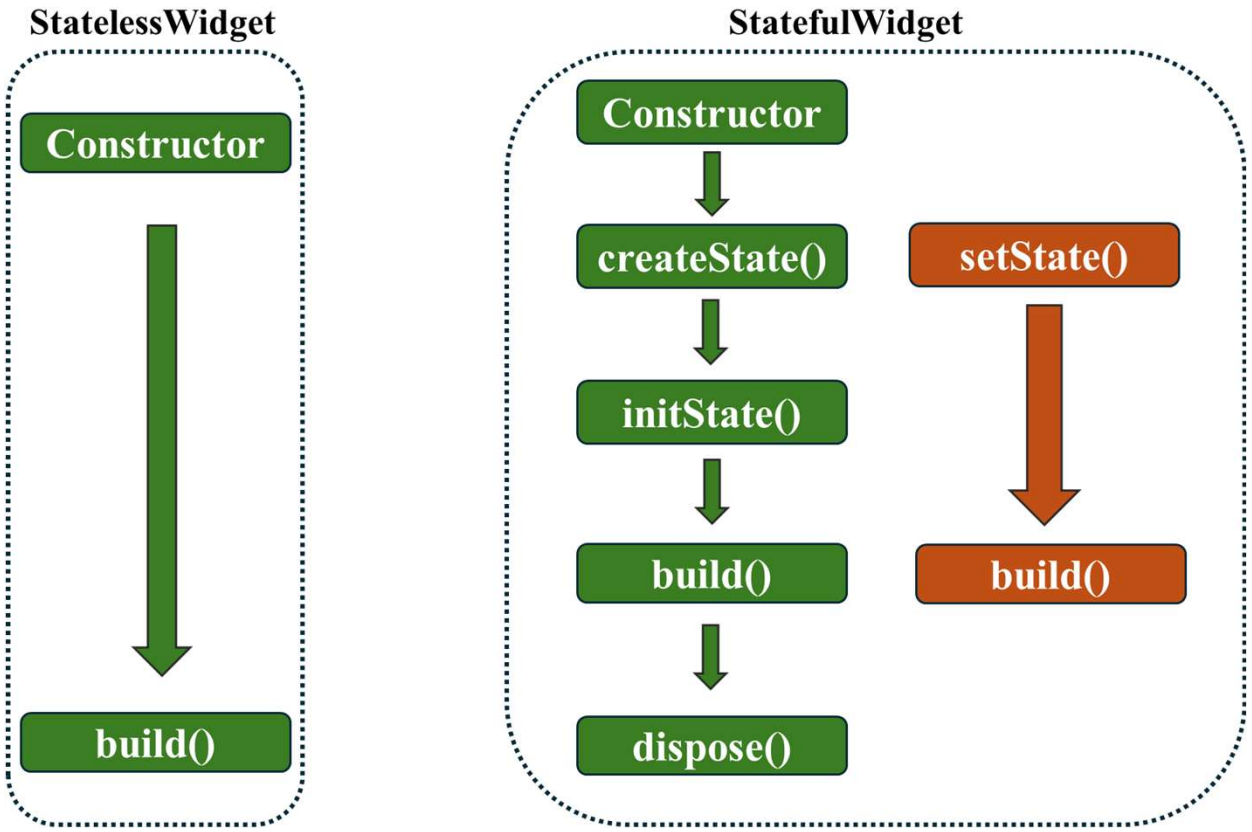


The widget is build ONLY when external data change

The widget is (re-) build when :
- **External data change.**
- **Internal state change.**

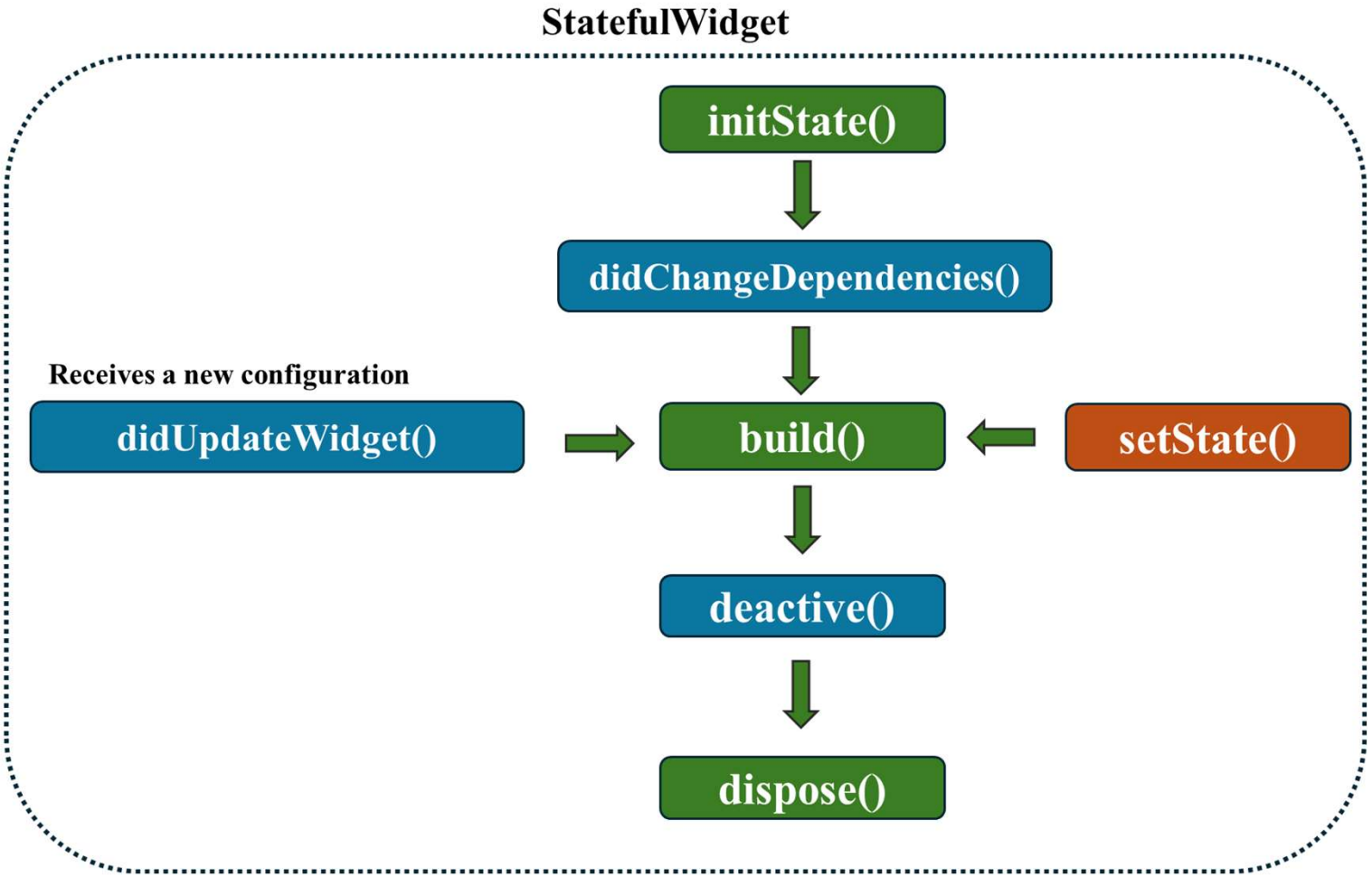
Widgets and concept of "State"

- **Widgets:** widget **lifecycle** is a sequence of events that occur when a widget is created, updated, or destroyed.



Widgets and concept of "State"

- **Widgets:** widget **lifecycle** is a sequence of events that occur when a widget is created, updated, or destroyed.



Widgets and concept of "State"

- **State:** refers to the data (information) stored inside a widget that can change during its lifetime and that Flutter uses to build and update the UI.

  - If the state changes, Flutter will need to rebuild the UI according to the new state.

  - Two types of state:

    - **Local State** (eg. checkbox , textfield).

    - **Global State** (app state eg. User's login status, User payment status, Purchased items by the user).

Plan

1. **Introduction to Flutter.**
   - What is Flutter ? And What is it good for ?
   - Flutter's features.
   - What exactly is Flutter ? Understanding Flutter as Framework.

2. **Exploring Flutter architecture.**

   - Flutter layers.
   - Specific design languages: Cupertino & Material design.
   - Widgets and concept of state.
   - Rendering.
   - Gesture, Painting, Animation.
   - Foundation.

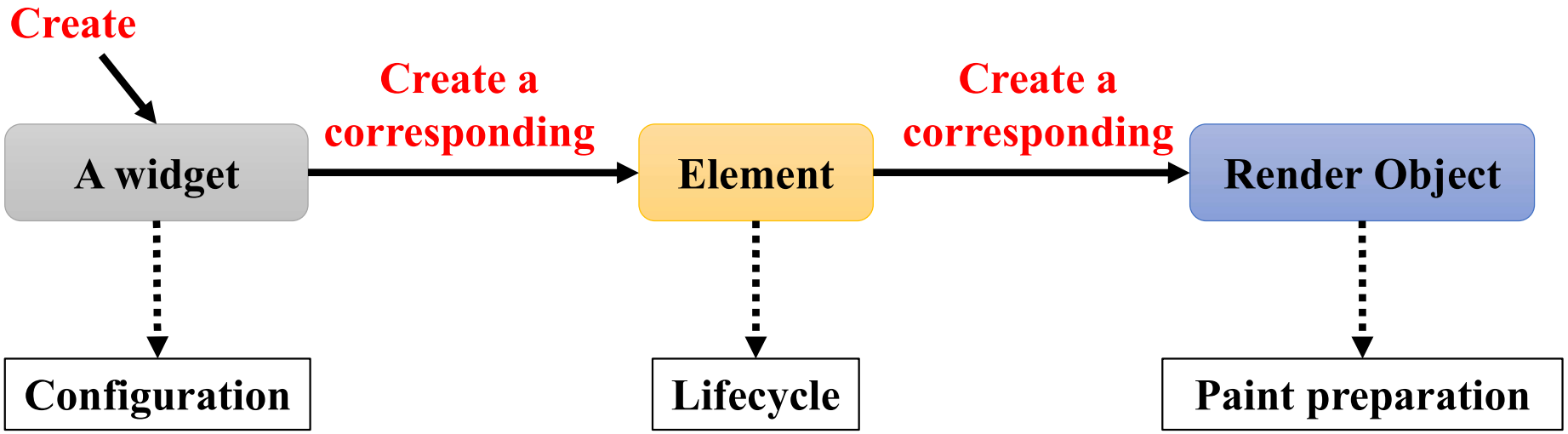
Rendering

- **"Everything is a widget":** is true in practice when writing Flutter code, since every element in the UI is represented as a widget.

- **Rendering:** refers to the process of turning widgets into pixels on the screen.

- Behind the scene, for **each widget**, Flutter creates several **underlying components** that work together to manage the rendering process.

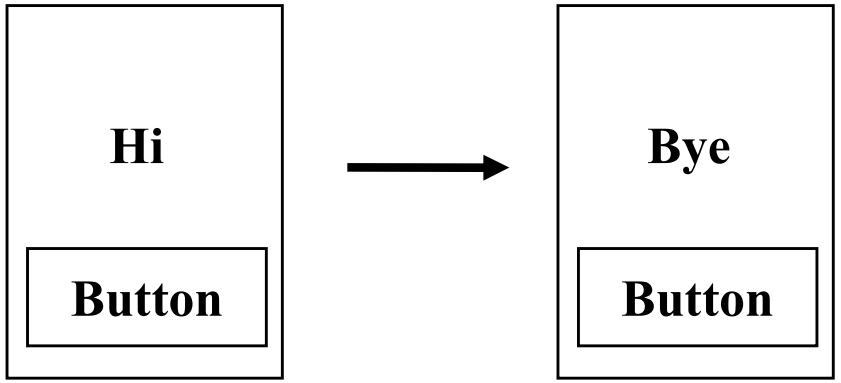
Rendering

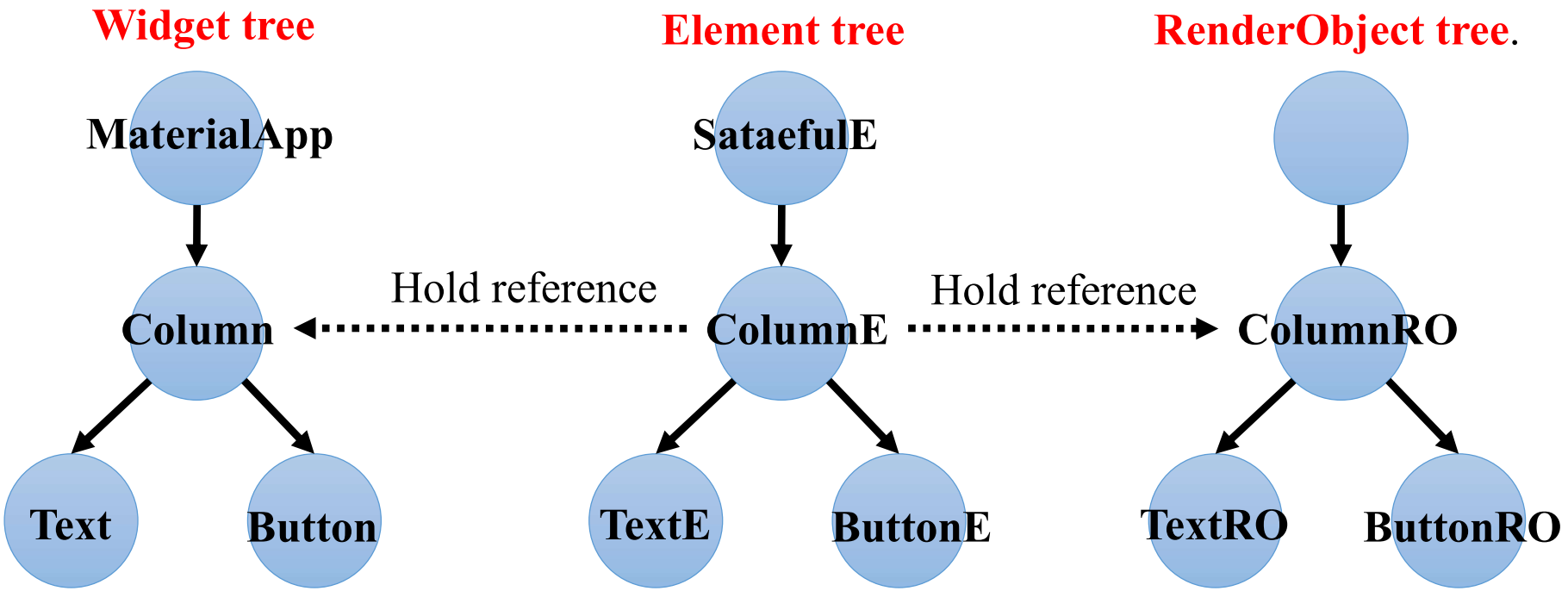


- These aspects result in three trees: **Widget tree**, **Element tree** & **RenderObject tree**.

Rendering

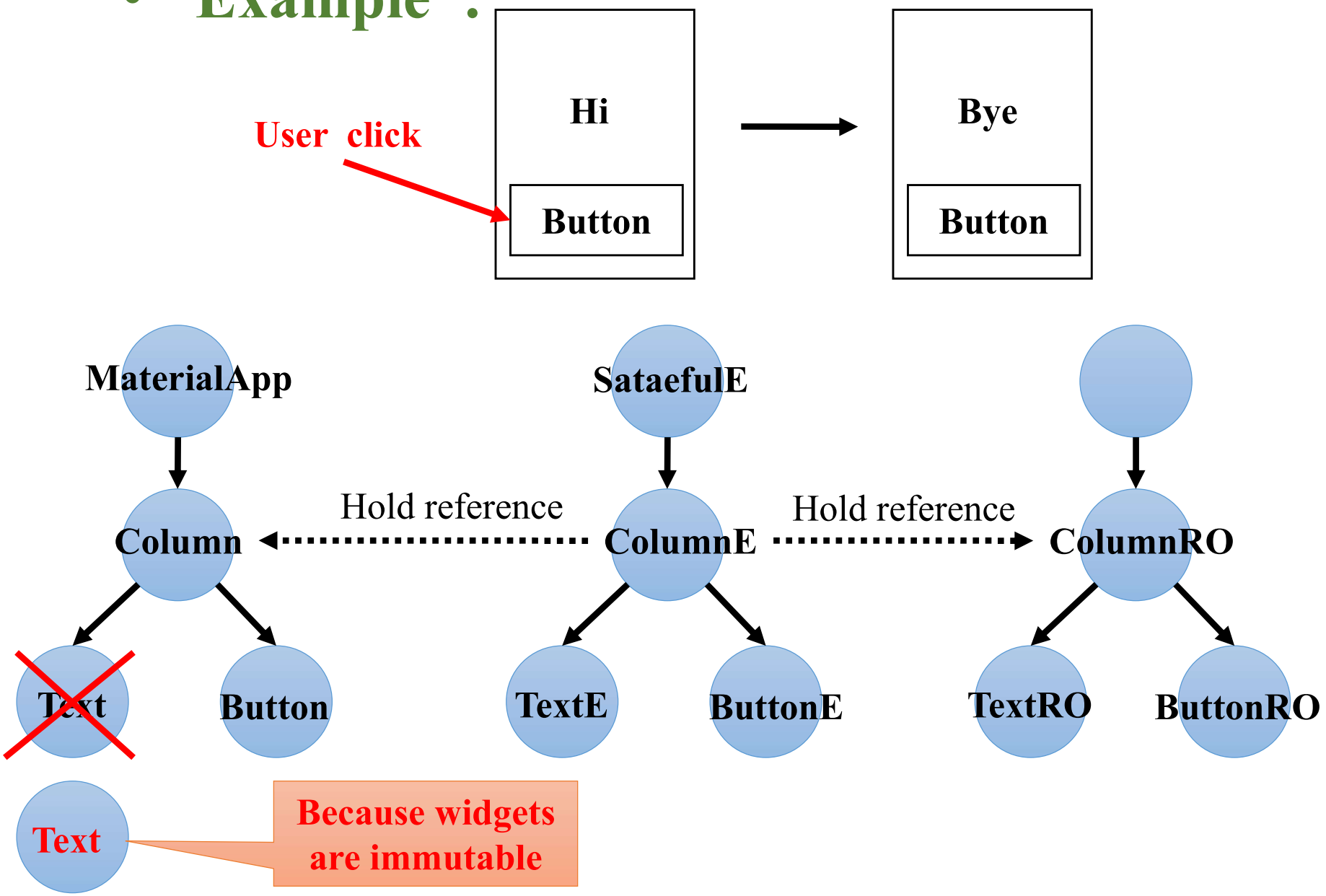
- **Example :**





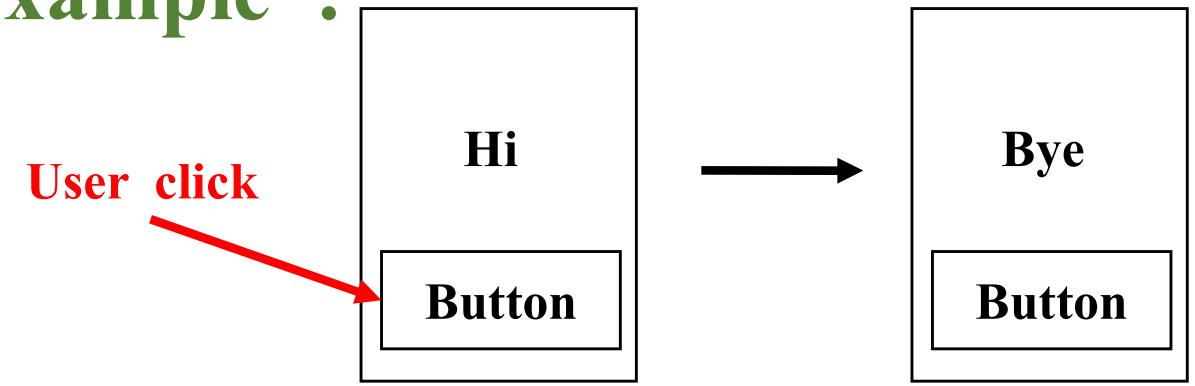
Rendering

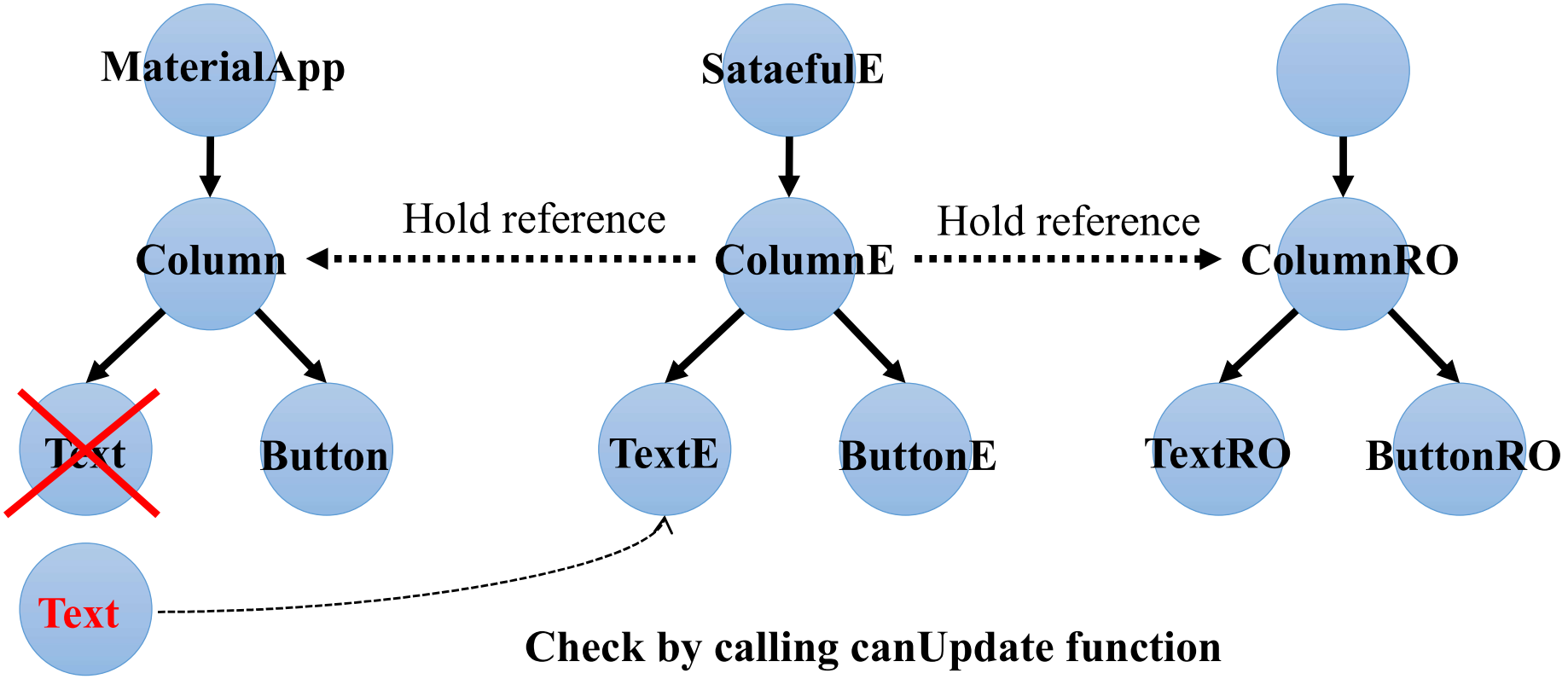
- **Example :**



Rendering

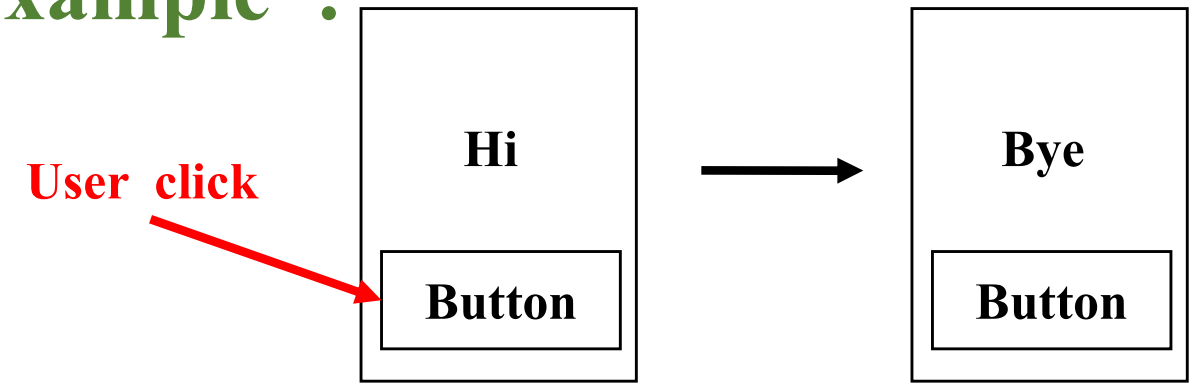
- **Example :**

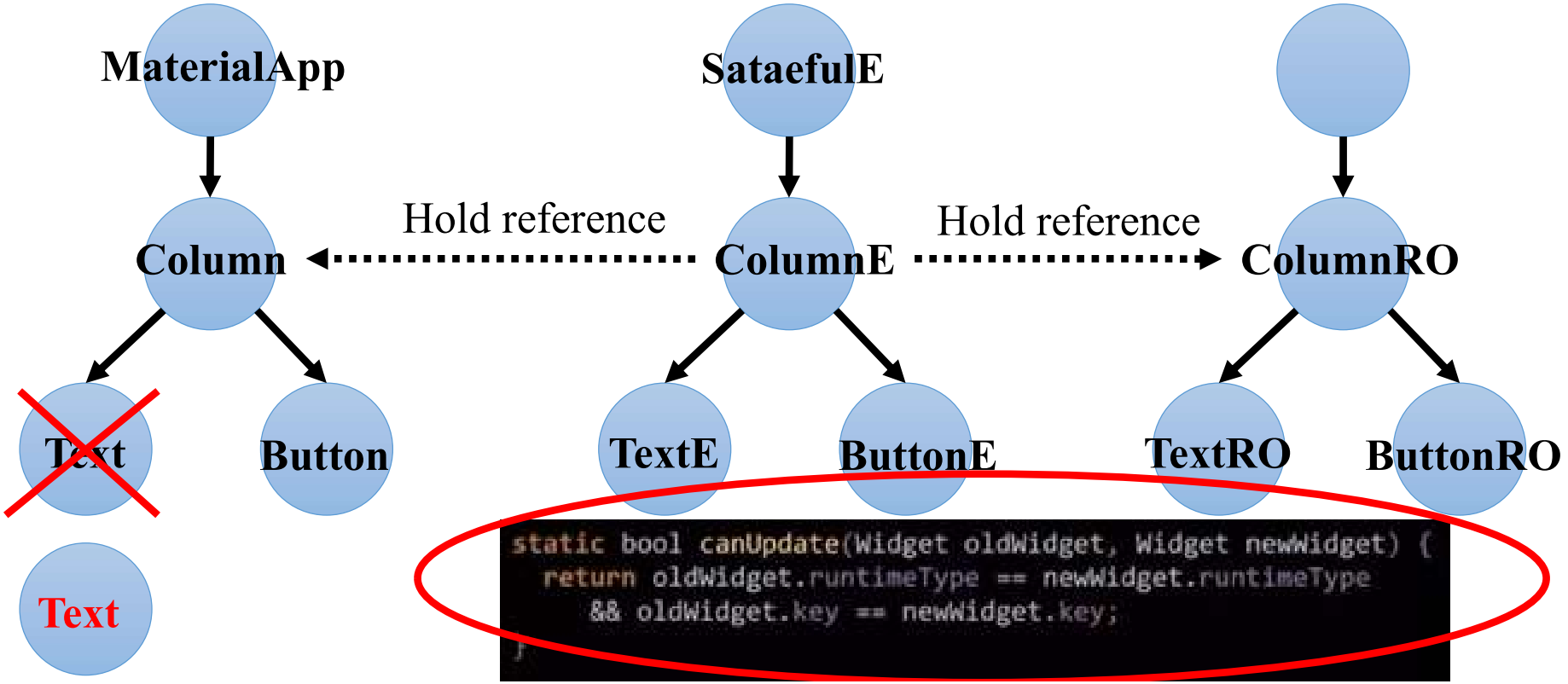




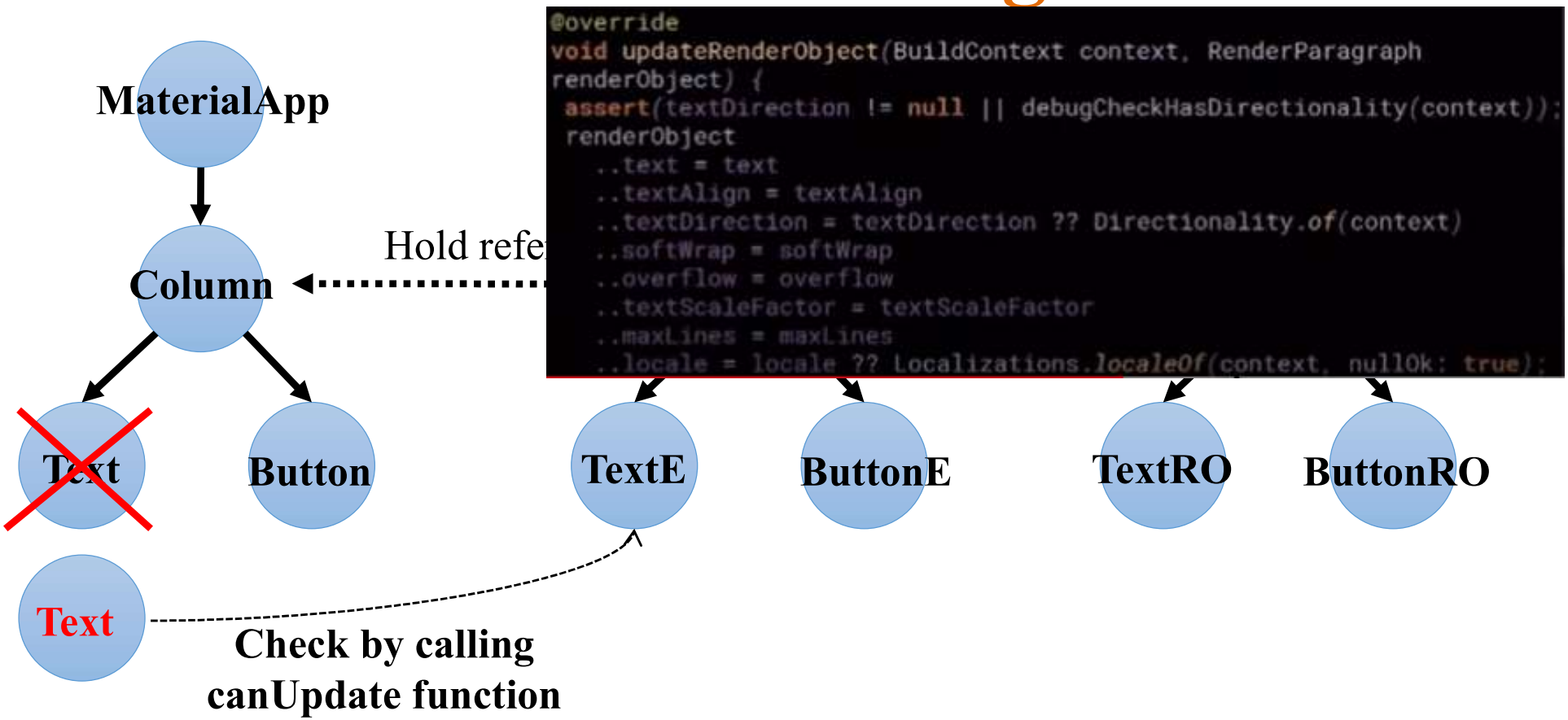
Rendering

- **Example :**



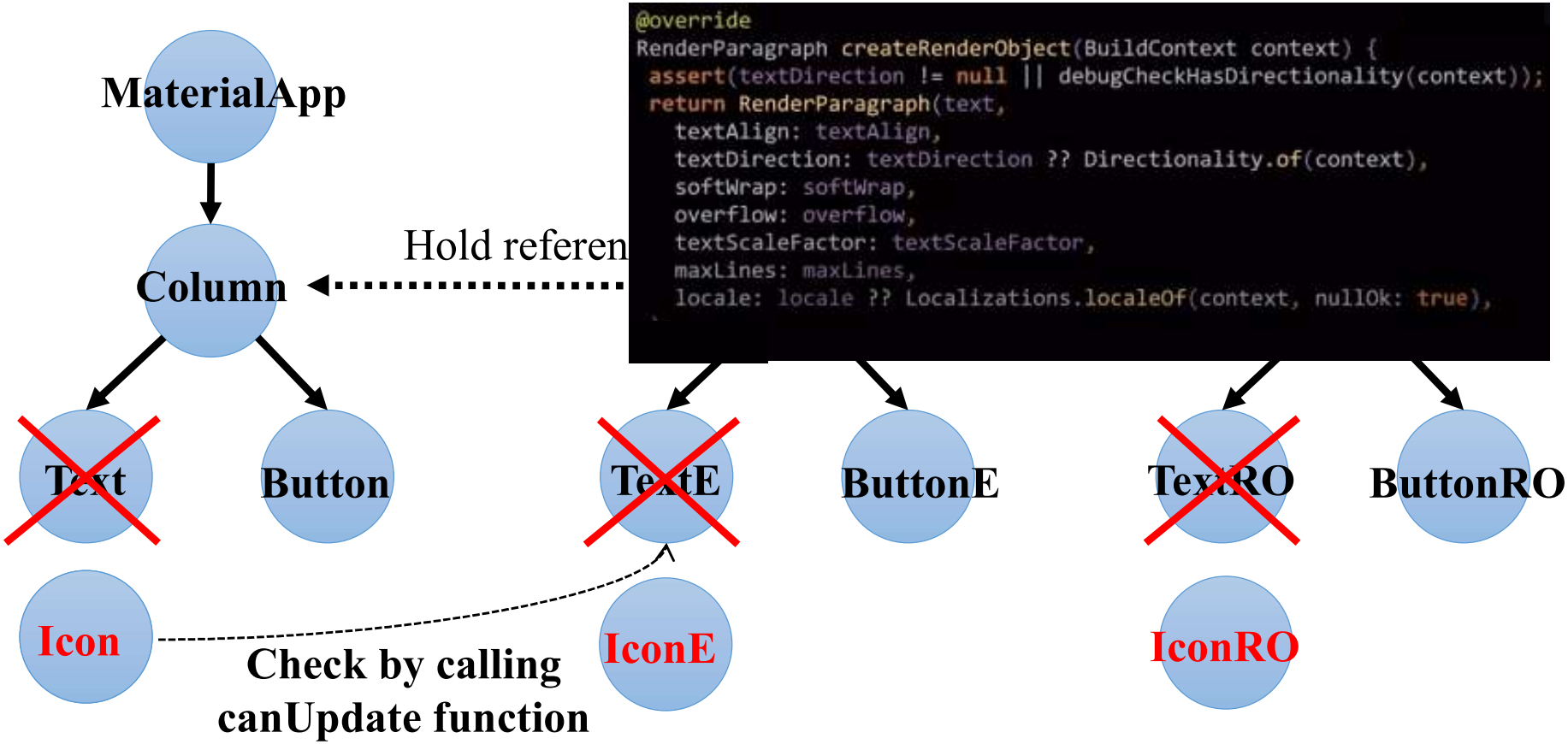


Rendering



If canUpdate returns **true** => call **updateRenderObject** function to set the new value to existing TextRO.

Rendering

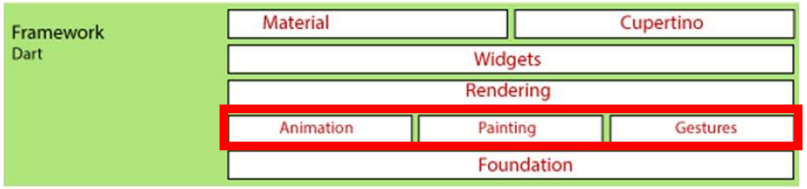


If canUpdate returns **true** => call **updateRenderObject** function to set the new value to existing TextRO.

If canUpdate returns **false** => call **createRenderObject** function to create a new IconRO in the render object tree.

Plan

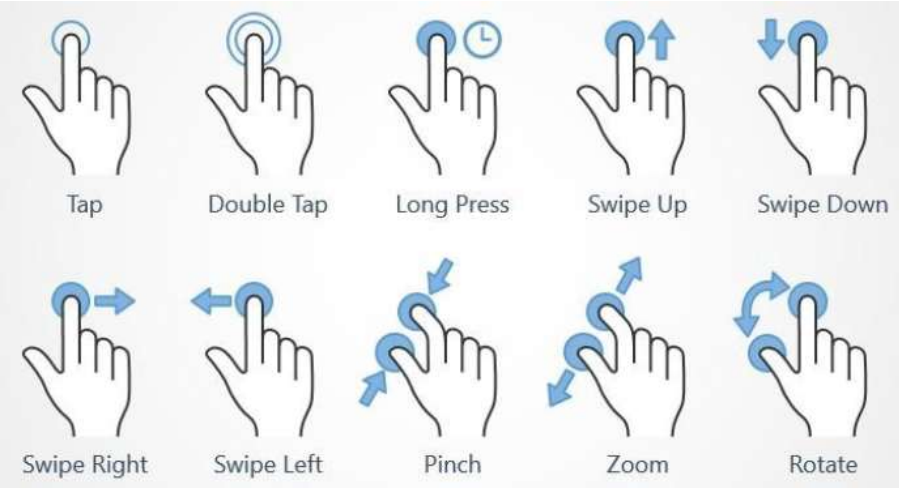
1. **Introduction to Flutter.**
   - What is Flutter ? And What is it good for ?
   - Flutter's features.
   - What exactly is Flutter ? Understanding Flutter as Framework.
2. **Exploring Flutter architecture.**



   - Flutter layers.
   - Specific design languages: Cupertino & Material design.
   - Widgets and concept of state.
   - Rendering.
   - Gesture, Painting, Animation.
   - Foundation.

Gesture, Painting & Animation

- **Gesture:** is any physical action/movement of a user when is interacting with the Flutter application.
- Pre-defined gestures that are detected and interpreted by interactive widgets (e.g. *GestureDetector*, *buttons*), allowing apps to respond easily to user interactions.
- **E.g**. tapping, dragging, scaling, swiping,…



Gesture, Painting & Animation

- **Painting:** draws the *RenderObjects* on the screen.
  - It handles *colors*, *shapes*, *text* and *images* on the *Canvas*.
- **Animation:** changes values such as *position*, *size* and *color* over time.
  - It does not draw but workswith *Rendering* and *Painting* layers.
  - It enables dynamic interfaces.

Plan

1. **Introduction to Flutter.**
    - What is Flutter ? And What is it good for ?
    - Flutter's features.
    - What exactly is Flutter ? Understanding Flutter as Framework.

2. **Exploring Flutter architecture.**

    - Flutter layers.
    - Specific design languages: Cupertino & Material design.
    - Widgets and concept of state.
    - Rendering.
    - Gesture, Painting, Animation.
    - Foundation.

Foundation

- The lowest-level utility classes and functions used by all other layers of the Flutter framework.
- Provides core classes & utilities that higher-level layers rely on.
- Includes **essential APIs** that indirectly support layout, gestures, painting, and animation.