What is a widget?A widget is a building block for your user interface. Using widgets is like combining  
Legos. Like Legos, you can mix and match widgets to create something amazing.

Flutter’s declarative nature makes it super easy to build a UI with widgets. A widget  
is a blueprint for displaying your app **state**.

You can think of widgets as a function of UI. Given a state, the build() method of a  
widget constructs the widget UI.

**Widget trees**Every widget contains a build() method. In this method, you create a UI  
composition by nesting widgets within other widgets. This forms a **tree-like data  
structure**. Each widget can contain other widgets, commonly called **children**

Rendering widgets

Flutter’s architecture contains three layers

1. Framework (Dart)
2. Engine(C/C++)
3. Embedder (platform specific)

**Framework Layer:**

Comprises of:

• **Material** and **Cupertino** are UI control libraries built on top of the widget layer.  
They make your UI look and feel like Android and iOS apps, respectively.  
• The **Widgets** layer is a composition abstraction on widgets. It contains all the  
primitive classes needed to create UI controls. Check out the official  
documentation here: https://api.flutter.dev/flutter/widgets/widgets-library.html.  
• The **Rendering** layer is a layout abstraction that draws and handles the widget’s  
layout. Imagine having to recompute every widget’s coordinates and frames  
manually. Yuck!  
• **Foundation**, also known as the **dart:ui** layer, contains core libraries that handle  
animation, painting and gestures.

**Three trees**Flutter’s framework actually manages not one, but three trees in parallel:  
• Widget Tree  
• Element Tree  
• RenderObject Tree  
Here’s how a single widget works under the hood:  
1. **Widget**: The public API or blueprint for the framework. Developers usually just  
deal with composing widgets.  
2. **Element**: Manages a widget and a widget’s render object. For every widget  
instance in the tree, there is a corresponding element.  
3. **RenderObject**: Responsible for drawing and laying out a specific widget instance.  
Also handles user interactions, like hit-testing and gestures.

**Types of elements**There are two types of elements:  
1. **ComponentElement**: A type of element that’s composed of other elements. This  
corresponds to composing widgets inside other widgets.  
2. **RenderObjectElement**: A type of element that holds a render object.  
You can think of **ComponentElement** as a group of elements, and  
**RenderObjectElement** as a single element. Remember that each element contains a  
render object to perform widget painting, layout and hit testing.

**Example trees for Card2**The image below shows an example of the three trees for the **Card2** UI:

