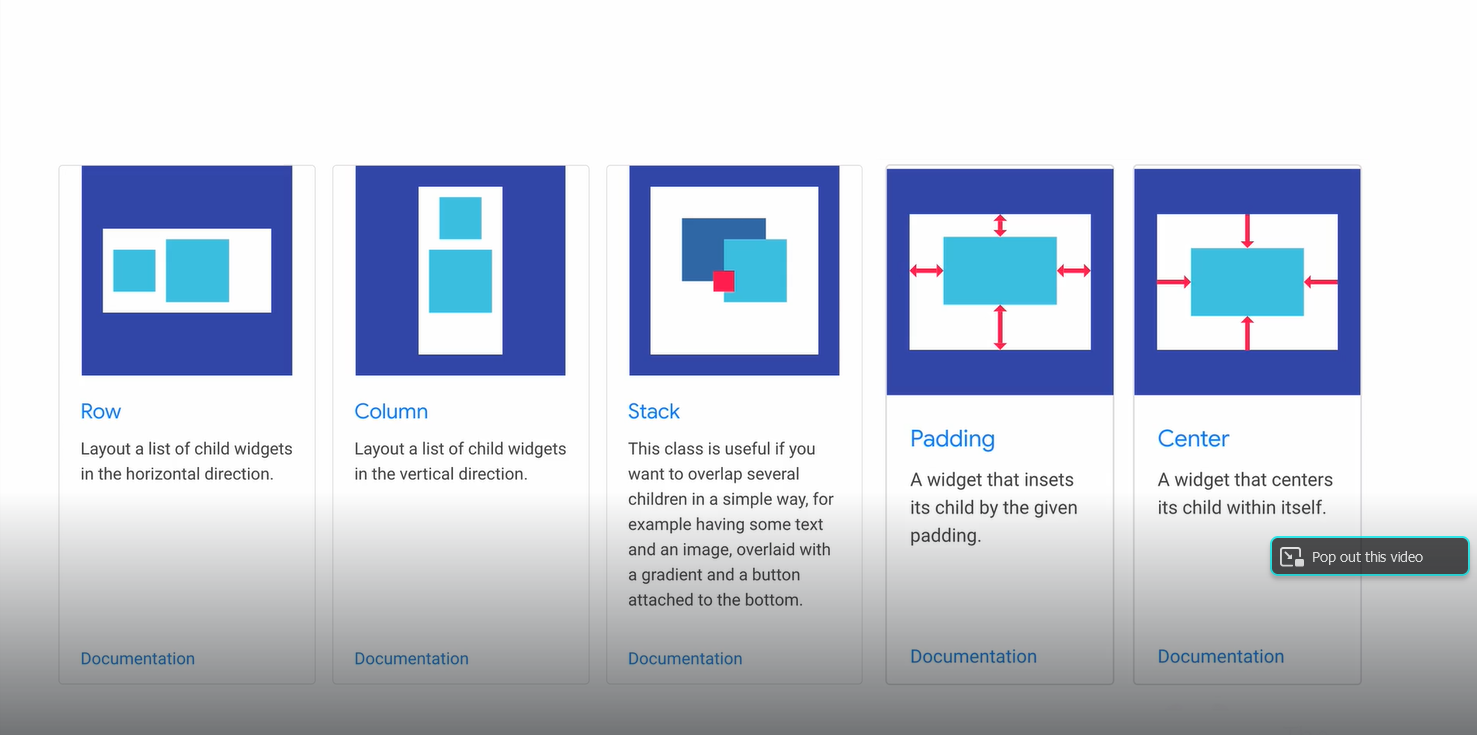
What is flutter?

The flutter is framework which is an open-source UI toolkit that is developed by Google. It was officially announced in May 2017 and first stable release, Flutter 1.0, was launched on December 4, 2018.

What are widgets?

In Flutter, a widget is a basic building block of the user interface. Widgets define how a part of the app should look and behave. They can represent anything from a button or a text label to complex layouts or entire screens. Widgets in Flutter are immutable and can be composed together to create more complex UIs.



**Note**: Android and iOS it will just provide a blank windows or canvas however all the drawing and design can be done via widgets from the flutter.

**Note:** In flutter, widgets are building block and by plugin together it builds an appealing application.

**Platform and Device independent**

Once an apps is developed in the flutter framework it can deployed everywhere in platform and devices without having rewrite the code or learning new technology or new skill or new programming language.

**Why prefer flutter?**

It prefers because of the reason:

1. One place to debug.
2. One place to maintain.
3. One place to Update.

Which it facilitates to one codebase to rule in all of them but not only of them the flutter facilitate to a simple and flexible layout system.

**Constraints**

Constraints are more complex as the number of elements grow on screen.

We no longer in 3.5-inch screen era anymore.

Constraints each of them in relation to get more complex.

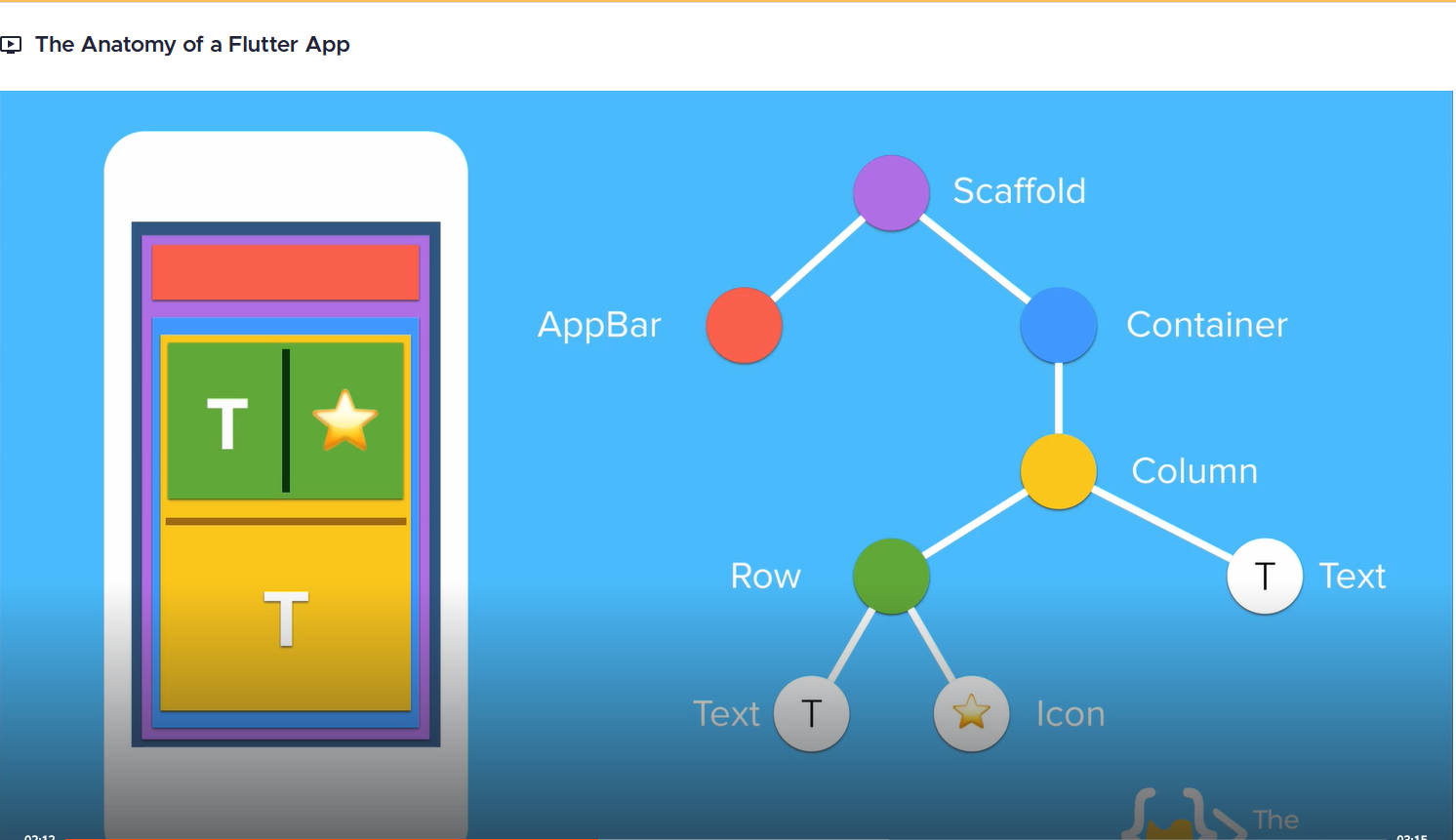
In website there are number that can adjust the user interface like Bootstrap, CSS Grid etc. but in mobile app there comes a flutter development.

**Split Second Reload or Hot Reload (it facilitates to reload super-fast)**

Hot Reload in Flutter is a feature that allows you to see changes you make to your code almost instantly without restarting your app. It is designed to speed up the development process by preserving the current state of your app while you update its code.

**access to the original source code**:

In Flutter, access to the original source code means being able to view, edit, and manage the actual Dart files and other resources that make up a Flutter application.



**Perquisites for building app with flutter**

1. Development Environment: Install Dart and Flutter SDK.
2. IDE Setup: Install Flutter and Dart Plugins and IDE Android Studio.
3. Platform-Specific Requirements: Configure Android SDK and Set Up an Android Emulator.

**Notes:**

Testing Android Apps from flutter is easy.

Testing iOS Apps from flutter is difficult because of code signing.

**List of Apps tester tools**

1. Mobile Emulator

* Android Emulator (Platform Android, Android Studio Emulator).
* iOS Emulator (Platform iOS mac).

1. Mobile Simulator

* iOS Simulator (can be considered both a simulator and an emulator Platform: iOS).
* Android Virtual Device (AVD) Platform: Android.

Codemagic: Bult, test and deliver you apps

**Developed Android apps with flutter you can use:**

Android Apps:

* MAC or PC
* Android Studio.
* Android Emulator or Physical Devices.

iOS Apps:

* Mac.
* Android Studio.
* iOS Simulator or Physical Devices.

All practical requirement for developed Android apps

Android Studio.

Flutter SDK (Version 3.22.0)

**Flutter Architecture**

Flutter's architecture is based on the concept of widgets, which are the building blocks of any Flutter application. We can divide the architecture of a Flutter app into three main components:

1. Flutter Framework
2. Dart Programming Language
3. Flutter Engine

**Flutter Framework**

Flutter framework is a UI toolkit that provides a set of **pre-built widgets** and tools for building beautiful and responsive apps for mobile, Web, and desktop platforms. It offers a modern and reactive programming model, where widgets can efficiently respond to user input and changes in the app state.

**Dart Programming Language**

Dart is an object-oriented programming language used to build Flutter applications. It is designed to be easy to learn, with a syntax similar to other popular programming languages like [Java and JavaScript](https://www.scaler.com/topics/javascript/difference-between-java-and-javascript/). Dart is a type-safe language that can detect errors at compile time, and it supports both just-in-time and ahead-of-time compilation. Dart is also optimized for the Flutter framework, which allows it to provide high-performance and fast development cycles.

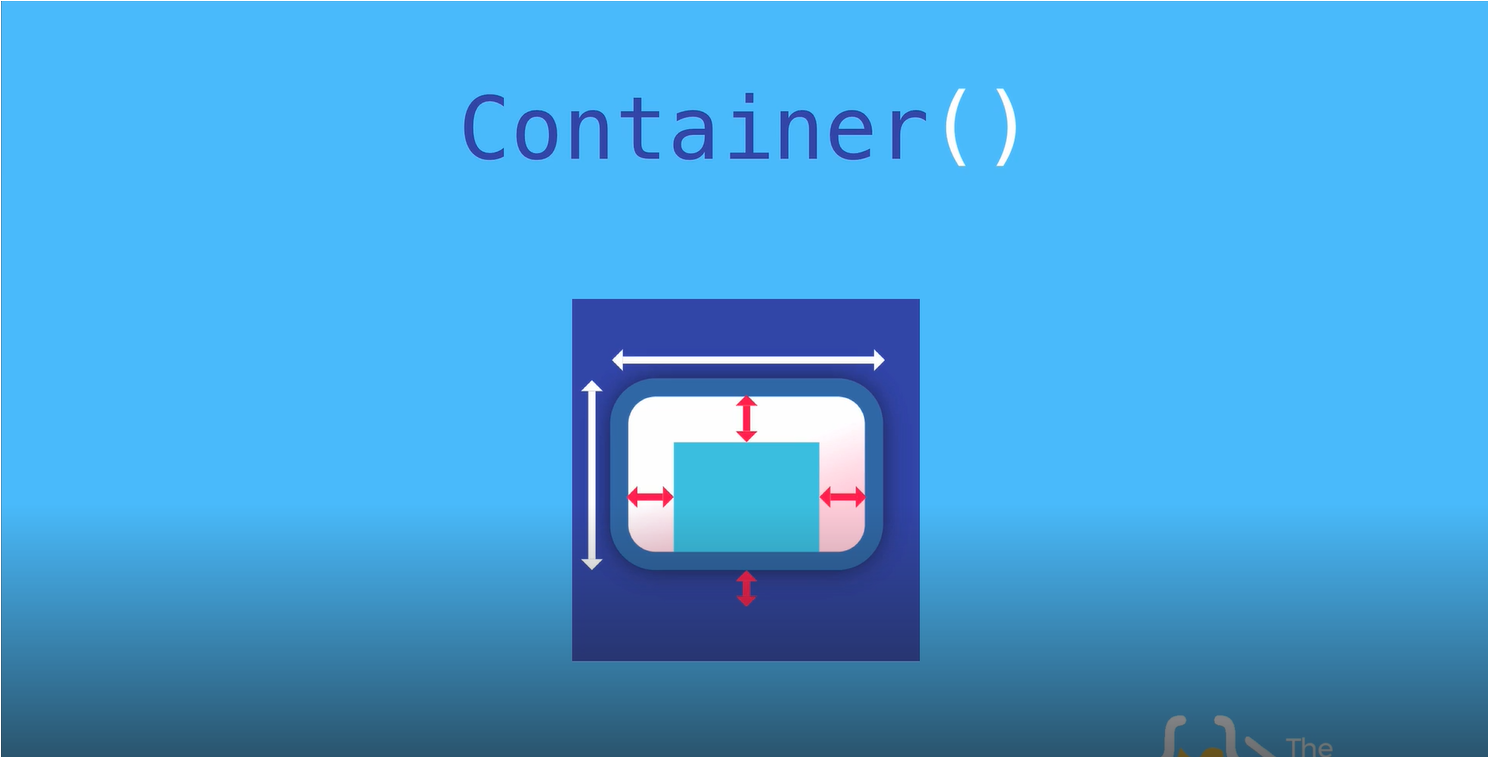
**Flutter Engine**

In Flutter architecture, the engine is the core of the framework and is responsible for managing the entire application lifecycle. It provides a complete set of low-level services that handle everything from rendering and layout to animation and gesture recognition.

The Flutter engine is written in C++ and is designed to be platform-independent, allowing it to run on a wide range of devices and platforms. It also includes a **virtual machine (VM)** called **Dart VM**, which is used to execute the application code written in Dart programming language.

Flutter engine is optimized for high performance and provides a fast and smooth user experience, even on low-end devices. One of the unique features of the engine is its support for **hot reload**, which allows developers to make changes to the application code and see the results immediately without the need to recompile the entire application.

**Container Widget (It is similar to the dev in Web Dev)**



In Flutter, a Container widget is a versatile and commonly used widget that allows you to create a box with various customizable properties.

1. Add padding and margins around its child widgets.
2. Set a background colour or image.
3. Define the size (width and height).
4. Apply borders and corner radius.
5. Align and position child widgets.

It's essentially a building block for designing UI layouts and can be used to wrap other widgets to apply styling and positioning.

SafeArea() widget

In Flutter, the SafeArea widget is used to ensure that your app's content is displayed within the safe, non-obstructed areas of the screen. This is particularly important on devices with notches, rounded corners, or other screen cutouts that might cover or obscure part of your UI.

**single-child layout widgets**

In Flutter, single-child layout widgets are widgets designed to manage and control the layout of a single child widget. They provide a way to position, size, and decorate this child within their own constraints. Essentially, these widgets take one child widget and apply layout rules or visual adjustments to it.

**Multi-child layout widget**

In Flutter, a "Multi-child layout widget" refers to any widget that can have multiple children and is responsible for arranging them within its own layout constraints. These widgets manage the positioning and sizing of their child widgets based on their own rules or constraints.

Widget build(BuildContext context)

In Flutter, the Widget build (BuildContext context) method is a crucial part of the widget lifecycle. It is used to describe how the UI of a widget should look based on its current configuration and state.

**Parameters**

**BuildContext context**: This parameter provides information about the location of the widget in the widget tree. It allows the widget to access other widgets and the overall theme, among other things.

**Purpose**

The build method returns a Widget that represents the part of the UI that this widget manages. The method is called whenever Flutter needs to change the appearance of the widget, such as when a widget's state changes or when the widget is first inserted into the widget tree.

**2. Building**

* **build() Method**: After initialization, the widget’s build method is called. This method returns a widget tree, which describes how the widget should be rendered on the screen.
* **Rebuilding**: Whenever the widget's dependencies change (e.g., data updates or state changes), the build method is called again to update the UI. For StatefulWidgets, this occurs when setState() is called.

**3. Updating**

* **State Changes**: In StatefulWidgets, when you call setState(), the framework schedules a rebuild. This causes the build method to be called again to reflect changes in the widget's state.
* **InheritedWidget Updates**: Widgets that depend on InheritedWidgets will rebuild when the InheritedWidget they depend on changes.

Stateless Widget (Abstract Class)

Build().

Statefull Widget (Abstract Class)

CreateState()

**Types of Widgets**

Flutter widget is split into two categories:

1. Visible (Output and Input)
2. Invisible (Layout and Control)