**FLUTTER**

**Flutter Development setup:**

An sdk is a software development kit is the bundle of all the code, all of the classes and tools and everything which helps us to develop applications

Flutter sdk uses git which is just a version control which allows us to save the code somewhere safe.

Flutter allows us to develop applications, cross platform applications in this acse android and ios applications.

For android applications-> Android Studio

For IOS Apps -> XCode

**Testing Virtual Devices:**

Android Emulators

Ios Simulators

**Installation:**

Flutter.dev website

Install android studio

Install git

Install flutter sdk and setup env variables

**IMP TO LEARN ABOUT FLUTTUR:**

Flutter is UI framework developed by google by building cross platform applications.(ability to run on multiple os like android, ios , windows , mac , linux and web ). They use only one codebase and build all

**Features of Flutter:**

* Flutter is open source and uses dart programming language
* Provides hot reload(without restarting the entire application, we can see changes) for fast development . When you modify your code and save it, flutter injects the updated code into the running Dart virtual machine(DVM)
* Uses widgets to build UI’s i.e widgets are building blocks of flutter app. Everything in Flutter, from buttons and text to entire layouts, is a widget.
* **Can create native like apps**: Developers write one codebase using dart and deploy it on both android and ios and still provides native like experience.
* **Why does it feel native :** Uses flutter’s own rendering engine (skia) instead of relying on platform specific ui components. Offers access to device features like camera, gps , sensors etc.
* **High performance :** uses its own rendering engine(skia)

**DIFF BETN IOS AND ANDROID**

**A screenshot of a computer

Description automatically generated**

<https://docs.flutter.dev/get-started/flutter-for/android-devs> // for android documentation

[https://dart.dev/language //](https://dart.dev/language%20//) for dart documentation

* After installing all, install flutter in vs code

To create flutter code, go to cmd and “flutter create good\_app”

**Basics of dart:**

// void main() {

// //print("hello dart");

// //var name='jyo';

// int age=79;

// bool isTall=false;

// //print("$name is $age years old.");

// sayMyName();

// int age1=myAge();

// print("age is $age1");

// print(age1);

// var name=showMyName("juoo");

// print(name);

// }

// String showMyName(String name)

// {

// var n="my name is $name";

// return n;

// }

// void sayMyName()

// {

// print("helo form function");

// }

// int myAge()

// {

// int age=90;

// return age;

// }

String myBio(String name, int age)

{

var bio= ("$name is $age years old");

return bio;

}

void main()

{

var bio=myBio("jyoo",20);

print(bio);

}

**Flutter Widgets:**

Anything in flutter is a widget – anything we interact with on the screen is a widget

Widget is a fundamental building block for constructing user interfaces like navbar, image, button, etc..

To get padding: ctrl+ .(command)

**child (Single Widget)**

* Used when a widget can only have **one** child.
* Accepts a **single** widget.

**children (Multiple Widgets)**

* Used when a widget can have **multiple** children.
* Accepts a **list** of widgets (List<Widget>).

**Pubspec.yaml file:**

This is the central hub of the whole application.

We are going to specify all the things that governs our application

To insert images : add in pubspec.yaml and in main.dart

**Flutter Widgets Categories**

1. **Stateless:** Stateless Widgets does not have a state

These widgets are immutable meaning they cannot change their properties or appearance once they are created.

Ex: Text, IconButton

1. **Stateful :** These have state and change over time

Have mutable properties that can be modified or changed which causes the widget to rebuild and reflect those changes.

* they use setState to make changes which trigger the rebuilding of the widget

Ex: Input, data changes etc

A screenshot of a computer

Description automatically generated

A computer screen shot of a computer screen

Description automatically generated

**Widget Hierarchy:**

The parent or root widget is Material app-> Scaffold

We have appbar

Body

Column

Inside columns we have buttons, images, text etc

Adavnatges:

Widget Reusability, we can reuse widgets on another screen also

**The const Keyword:**

* 1. **Performance** : values are computed at compile time – memory us not required
  2. **Avoids Unnecessary rebuilds** : widgets marked const don’t rebuild if their values don’t change, responsiveness is increased which means application will not lag
  3. **Prevents Unwanted Mutations (Immutability):** const ensures that variables cannot be modified after initialization

**NOTE:**

**We can create the project using terminal by “flutter create utip”**

**Or**

**Open command pallete create new flutter project and continue to create project**

Placeholder is nothing but a widget which draws a box instead use scaffold for for white page

**BuildContext:**

It is immutable

Represents the location of a widget in the widget tree

It is used to access widget properties, navigate between screens, and obtain the theme or media query information.

**Key Uses of BuildContext**

1. **Accessing Widget Properties:** Helps retrieve widget-related data (e.g., theme, size, localization).
2. **Navigating Between Screens:** Used in Navigator.push() for screen transitions.
3. **Accessing InheritedWidget Data:** Helps retrieve shared state from widgets like Theme.of(context) and MediaQuery.of(context).
4. **Finding Parent Widgets:** Used to obtain references to ancestor widgets in the widget tree.

**Widgets, Objects and Classes**

Widgets are building blocks of application , these widgets are related to dart objects

Dart objects are instance of dart classes, here comes the OOP

**Class :** defines behaviours and structure of a certain type, it is indeed a blueprint that encapsulates these behaviours and everything that an object in this case that is ready to be used will have in the real world

**Printing name and age:**

void main() {

var person = new Person();

var age=person.age = 23;

var name=person.name = "jyo";

print("hello my name is $name of $age years old");

}

class Person {

String name = "";

int age = 0;

}

**Printing using Constructor:**

void main() {

var person = new Person("jyoo",20);

var name=person.name;

var age=person.age;

// print("Hello my name is $name of $age years old ");

var fullPerson= "Hello $name you are $age years old";

print(fullPerson);

}

class Person {

String name ;

int age ;

Person(this.name, this.age);

}

**Printing via behaviours**

void main() {

var person = Person("james", 30);

person.greeting();

}

class Person {

String name;

int age;

Person(this.name, this.age);

void greeting() {

print("Hello , my name is $name, I am $age years old");

}

}

**After section 7 we learned:**

Flutter widgets- intro

Widget tree hierarchy

The setstate

Tip calci building

Containers, columns

Code refractoring

**State Management in Flutter:**

**The setState({}):**

Good for simple apps

Problems:

* Bloats our code looks messy
* Error prone
* **Causes Unnecessary Widget Rebuilds**

When you call setState(), **the entire widget rebuilds**, even if only a small part of it needs updating. Even if only \_counter changes, **the entire widget tree re-renders**.

* **Doesn't Work Well for Global State**

You cannot share state between different widgets or screens

* **Poor Performance in Large Apps**

Frequent widget rebuilds slow down the UI.

Not suitable for **complex apps** with many interactive components.

Other management tools:

* Provider
* Bloc/Cubit
* Riverpod
* GetX

**Provider:**

The provider package is a state management solution that simplifies the way widgets interact with and share data. It is widely used for dependency injection and state management in Flutter applications.

Benefits:

**Scalability** : state can now be accessed across multiple wigets

**Performance**: only updates the widgets which needs rebuilt

**Separation of Concerns**: Keeps business logic separate from the UI.

**Readability and maintenance** increases

**Reusability**: Use the same code in different parts of the application without needing to do it from scratch

* Flutter pub add provider is the command

**Change Notifier**

Change Notifier is a built-in Flutter class that provides a simple and efficient way to manage state. It is often used with the provider package to notify widgets when state changes.

**Efficient** – Notifies only listeners that need updates.  
 **Simple API** – Uses notifyListeners() to update UI.  
**Works Well with Provider** – Commonly used in Flutter apps with provider.

**Consumer vs Provider.of()**

**Consumer:**

Rebuilds a specific part of widget tree

Provides better control over which parts of the widget is to be rebuilt

**Provider.of():**

A way to access the provided state directly, it doesn’t automatically rebuild the widget tree when the state changes

Must call this manually inside the build method for automatic updates

**Gesture detector:**

It allows us to wrap the widget and make that clickable

Gesture detector does not provide built in material design feedback

**InkWell:**

Both inkwell and gesture detector allows you to handle touch and tap gestures.

When we click there’s that sense of splash that happens around the clickable area