Flutter

1. Setting up
   1. install Flutter [here](https://docs.flutter.dev/get-started/install)
   2. in CLI test flutter by  
      flutter --version  
      flutter doctor
2. create a new project
   1. In CLI type in:  
      flutter create [project name]
3. Project structure
   1. an application in Flutter is always a bunch of nested widgets (sort of like Russian dolls)

Widget – each widget is a Class

1. general idea
   1. each widget is a class and to create it, just call the constructor
   2. to add item into the class/widget, specify the item that is to be added as arguments in the constructor
   3. some useful arguments
      1. child 🡪 take another (single) widget
      2. children 🡪 take a list of widgets
2. Stateless widget
   1. Parent class for widgets
   2. Only have one state
   3. Used to create a custom widget by extending this class
   4. Ex:  
      class MyScrapElement extends StatelessWidget {

const MyScrapElement({super.key});

@override

Widget build(BuildContext context) {

return const Text('this widgets is a text');

}

}

1. Stateful widget
   1. Create widget with multiple states, such as input box. A widget has multiple states means that the widget have attributes that changes it states/value throughout the lifetime.
   2. Vscode shorthand: stful
   3. All the override below must be written to make stateful widget
   4. To update a state of a stateful widget (redraw the whole widget), setState(void func()) must be called
   5. To Access the Widget class’s attribute from the WidgetState class, WidgetState class has attribute named "widget" that have all the attributes from Widget class
   6. Ex:  
      class MyWidget extends StatefulWidget {

const MyWidget({super.key});

@override

State<MyWidget> createState() => \_MyWidgetState();

}

class \_MyWidgetState extends State<MyWidget> {

@override

Widget build(BuildContext context) {

return const Placeholder();

}

}

1. CallBack idea
   1. Situation: two widget has to interact between each other, such as a parent widget has a text field widget and another button widget. The button needs to change the state of the parent widget by changing its text
   2. Solution: create a function on the parent class that will change the text, and parse that function to the button widget as a argument for its constructor. The button widget would have to create a new attribute for the function placeholder. Call the parent function through the place holder inside the button.
   3. Ex: (visualization)  
      class A {  
       Function changeText()//this will change the text  
       Has(  
       Text('')  
       ButtonB()  
       )  
      }  
        
      class ButtonB(){  
       Function callback//this is the placeholder  
       Constructor ButtonB(Function func){  
       this.callback = func;  
       }  
       onClick() {callback()}  
      }
2. Container()
   1. basically, a box container
3. Text()
   1. a text widget
4. MaterialApp()
   1. has home, doesn't have child/children
5. Material()
   1. is an opensource widget
   2. all children of material will have component of Material
6. Column()
   1. give a column widget (vertical array of widgets)
7. TextField()
   1. Is stateful widget.
   2. To control the widget use controller parameter that takes of type TextEditingController
   3. To redraw/change the state of TextField, use onChanged parameter, that takes a void Function(String) as a parameter. Remember to use setState(void Function()) to change a state.
   4. Ex:  
      class TextInputWidget extends StatefulWidget {

const TextInputWidget({super.key});

@override

State<TextInputWidget> createState() => \_TextInputWidgetState();

}

class \_TextInputWidgetState extends State<TextInputWidget> {

final TextEditingController myController = TextEditingController();

String text = 'as';

@override

void dispose() {

super.dispose();

myController.dispose();

}

void changeStateText(String msg) {

setState(() => this.text = msg); //setState will change the state of the widget

if (this.text == 'Hello World'){

setState(() => this.text = '');

}

}

void Function(String) changeStateText2(){

return (msg) => changeStateText(msg);

}//alternative

@override

Widget build(BuildContext context) {

return Column(

children: [

TextField(

controller: myController,

decoration: const InputDecoration(

prefixIcon: Icon(Icons.message),

labelText: 'Put something here',

),

onChanged: changeStateText, //this will call the function with the text from the textbox as the function argument

),

),

Text(this.text),

],

);

}

}