Couleur

Nous avons généralement trois façon de mettre une couleur dans widget :

- Placer directement une couleur dans widget
- Créer des variables de couleurs et fixer des couleurs a ces variables pour les appeler
- Créer un composant de couleurs pour importer ce composant et utiliser les couleurs

Container

Un container est un widget qui peut contenir un ou plusieurs widgets, tout dépend du travail à faire.

```
Container(
        hauteur,
        largeur,
        couleur ,
        decoration,
        autre widgets etc ......
  )
```

Text

Un texte en flutter peut ou non être dans un container.

```
Text(
        'texte brut',
        style(taile, coleur, etc ......)
)
```

Icon

Une icone peut ou non être dans un container

```
Icon(
        icon,
        taile,
```

```
        coleur etc ....
  )
```

Container & expended

- Le widget Expended contient le widget Container
- Le widget Expended est très intélligent il occupe automatiquement l'espace restante lorsqu'on diminue ou augment e la taille d'un autre widget

```
Expended(
        child: Container(
                ----mettre tout ce qu'on veut---
        )
)
```

Column

Le widget Colonne nous permet de disposer nos widgets en colonne. Une colonne peut contenir une infinité de widgets.
Une encore tout dépend de ce qu'on veux en faire.

- Une colonne contient comme widget principale une list d'éléments (children)

```
children: [
        -----contient une infinité de widgets----
]
```

Nous allons à l'intérieur de children , écrire trois widgets de type Container

- Nous allons utiliser les propriétés du positonnement:

```
body : Column(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        crossAxisAlignment: CrossAxisAlignment.center,
)
```

Flex

La propriété flex est utilisée pour réorganiser les containers contenu dans les widgets de type Expended.

- Expended a donc pour prémière propriété : flex : n,
- n est une variable.

```
Expended(
  flex : 1,
  Container(
        color: Colors.blue,
        ---aucune hauteur, ni largeur n'es permise ici --------------
  )
)
```

- Pour exemple, nous allons dans la Column, dans Children, créer trois widgets Expended qui contiennent donc un widget Container chacun.
- Pour le deuxième widget Expended nous allons donc utiliser la propriété flex et jouer avec la variable n.

Listview

Cette propriété permet d'éviter le comportement du dépacement de taille de l'écran.

- Maintenant dans notre Children, nous allons éffacer les widgets Expended et laisser que les widgets de type container (les trois).
- En suite fixer donc une hauteur de 500 pour chaque container.. La hauteur de 500, pour c'est un choix alors, mettez y ce que voulez.
- On remarque en suite un problème du dépassement de taille.
- Pour corriger ce problème alors nous allons donc:
  - Effacer le widget Column et le remplacer par le widget "ListView"
  - ListView est un widget scrollable, ce qui signifie que, peut importe la taille, lui prend sa en compte dans son scroll infini.

Scrolldirection

Comme nous avons vu comment agencer nos container, il faut pouvoir les scroller.
On utilise donc "ScrollDirection" qui prend une valeur (horizontal ou vertical).

- Ici dans notre exemple précédent, si placer directement cette propriété vous n'y verrez rien . Pour la simple raison que nos containers nous pas encore de largeur.
- Fixer donc une largeur au choix a vos containers.

Row

Rien à faire, il s'agit des même propriétés que Colonne

Construire ma propre listview grâce a la propriété : listview.builder

Cette propriété a pour rôle de faire passer des données dans les composants.

```
body: ListView.Builder(
    itemCount: n,
    itemBuilder: (context, index) => ListTile(
        title: Text(index.toString()),
    ),
)
```

- Créons donc la liste des mois de l'année puis les afficher.

```
class MyApp extends StatelessWidget {
    //le constructor n'est pas une constant lorsqu'il y a des attributs dans la
classe.
  MyApp({super.key});
  List names = ["Janvier", "fevrier", "Mars", "Avril", "Mai", "Juin", "Jueillet",
"Août", "Septembre", "Octobre", "Novembre", "Decembre"];
    home: Scaffold(
        body: ListView.builder(
            itemCount: names.length,
            itemBuilder: (context, index) => ListTile(
            title: Text(names[index]),
            ),
        )
    ),
}
```

Gridview.builder

Pour utiliser le grid, nous avons besions des propriétés comme :

- itemCount: n; il s'agit du nombre de boite de la grid,
- gridDelegate: il s'agit du nombre de boîte avec un alignement en colonnes.
- itemBuilder : pour construire nos grid.

```
body: GridView.builder(
    //il s'agit du nombre de boites
    itemCount: 64,
    //il s'agit du nombre de de colonnes'
    gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(crossAxisCount: 4),
    //Pour chaque grid faire ce qu'on veux dans le widget de type Container
    itemBuilder: (context, index) => Container(
        color: Colors.deepPurple,
        margin: EdgeInsets.all(2),
    ),
),
```

Stack

Il s'agit d'un widget qui permet d'empiler les autres widgets l'un sur l'autre.

- Il contient deux propriétés : " alignement" et "children".
- Donc a l'intérieur de children libre de faire ce qu'on desire.

Gesturedetector

Comme son nom l'indique il s'agit du detecteur de geste.
Il contient principalement : un evenement et un widget de type child,

- Pour evenement, exemple : "onTap", "onDoubleTap", etc .....

```
body:GestureDetector(
    onTap: () {
    print("Bonjour tout le monde");
    },
    child: Center(
    child: Container(
        height: 200,
        width: 200,
        color: Colors.deepPurple[300],
        child: Center(child: Text("Tap me !", style: TextStyle(color: Colors.white,
fontSize: 25),)),
    ),
    ),
)
```

ou assicions une fonction de type void par exemple .

```
class MyApp extends StatelessWidget {
  MyApp({super.key});
  void userTapped(){
    print("Je suis content");
  }
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      home: Scaffold(
        body:GestureDetector(
          onTap: () {
            userTapped();
          },
          child: Center(
            child: Container(
              height: 200,
              width: 200,
              color: Colors.deepPurple[300],
              child: Center(child: Text("Tap me !", style: TextStyle(color:
Colors.white, fontSize: 25),)),
            ),
          ),
        )
      ),
    );
  }
}
```

🖊 Navigation

```
 A)  Simple Navigator
```

Cette propriété permet d'aller de page en page.
créons un répertoire pages, qui contient deux fichiers : first_part et second_page.

- Dans notre fichier main.dart
- Ecrire un AppBar et un body.

```
-------fichier main.dart

home: FirstPage(),
```

```
------fichier first_page

body: Center(
    child: ElevatedButton(
        child: Text("Go To Second Page"),
        onPressed: () {
        Navigator.push(
            context,
            MaterialPageRoute(
            builder: (context) => SecondPage(),
            )
        );
        },
    ),
 ),
```

```
B) routes
```

```
-----------code du main.dart

home: FirstPage(),
routes: {
    '/firstpage':(context) => FirstPage(),
    '/secondpage':(context) => SecondPage(),
},

----------code du first_page.dart

body: Center(
    child: ElevatedButton(
        child: Text("Go to Second Page"),
        onPressed: () {
            Navigator.pushNamed(context, '/secondpage');
        },
    ),
),
```

Drawer

Comporte des propriétés suivantes : la couleur, un widget column, qui contient un children qui contient : DrawerHeader(), ListTitle(Leading, title, onTap etc ....)

- Exemple

```
drawer: const Drawer(
    backgroundColor: Color.fromARGB(255, 206, 188, 206),
    child: Column(
        children: [
        DrawerHeader(
            child: Icon(
            Icons.favorite,
            size: 50,
            ),
        ),

        ListTile(
            leading: Icon(Icons.home, size: 30),
            title: Text("H O M E"),
            onTap: () {
            Navigator.pop(context);
            Navigator.pushNamed(context, '/homepage');
            },
        ),

        ListTile(
            leading: Icon(Icons.settings, size: 30),
            title: Text("SETTINGS"),
            onTap: null,
        )
        ],
    ),
),
```

puisque j'ai créer plusieurs page : home_page.dart, settings_page.dart alors, je vais faire le routage au niveau de main.dart

```
home: FirstPage(),
    routes: {
    '/firstpage':(context) => FirstPage(),
    '/homepage':(context) => HomePage(),
    '/settingspage':(context) => SettingsPage(),
},
```

Bottomnavigationbar & statefullwidget

Pour parler de cette partie nous devrions être minucieux.
Au départ nous créons une classe qui étend de StatelessWidget mais, maintenant qu'il y a changement d'état, nous ne pouvons pas encore utiliser une classe statique alors il nous faudra utiliser une classe dynamic dans laquelle nous allons interragir avec les données.
Il suffit donc de faire un clique sur StatelessWidget puis prendre l'option convertir en StatefullWidget c'est tout.

-------BottomNavigationBar----------
BottomNavigationBar contient qu'une liste de Widgets. Une fonction onTap

-----StatefullWidget-------------------
Nous declarons un attribut privé qu'on initialise à 0.
Une fonction void, qui est un setter.
__ Une liste des composant.

```dart
import 'package:basis/pages/home_page.dart';
import 'package:basis/pages/profile_page.dart';
import 'package:basis/pages/settings_page.dart';
import 'package:flutter/material.dart';

//au depart c'était un stateless maintenant il faut faire
//un clique sur statelessWidget puis convertire en statefullWidgets
class FirstPage extends StatefulWidget {
  FirstPage({super.key});
  @override
  State<FirstPage> createState() => _FirstPageState();
}

class _FirstPageState extends State<FirstPage> {
  //this keeps track of current page to display
  int _selectedIndex = 0;
  //this method updates the new selected index
  void _navigateBottomBar(int index) {
    setState(() {
      _selectedIndex = index;
    });
  }

  final List _pages =[
    // homepage
    HomePage(),
    //Profilepage
    ProfilePage(),
    //settings page
```

```
    SettingsPage(),
  ];

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("Mi Amor", style: TextStyle(color:
Colors.white),), centerTitle: true, backgroundColor: Colors.blue,),
      //body: _pages[0],
      body: _pages[_selectedIndex],
      bottomNavigationBar: BottomNavigationBar(
        //
        currentIndex: _selectedIndex,
        //
        onTap: _navigateBottomBar,
        items: const [
          //home
          BottomNavigationBarItem(
            icon: Icon(Icons.home),
            label: 'Home',
          ),
          //home
          BottomNavigationBarItem(
            icon: Icon(Icons.person),
            label: 'Profile',
          ),
          //home
          BottomNavigationBarItem(
            icon: Icon(Icons.settings),
            label: 'settings',
          )
        ],
      ),
    );
  }
}
```

🖊 Statelesswidget & statefullwidget (attribut, method, design)

__ D'abord parlant du fichier main, nous avons créer un composant de type StatefullWidget pour faire juste une incrementation de valeur.

```
import 'package:flutter/material.dart';
class CountPage extends StatefulWidget {
```

```dart
  const CountPage({super.key});
  @override
  State<CountPage> createState() => _CountPageState();
}


class _CountPageState extends State<CountPage> {
  //variables (attributs)
  int _counter =0;
  // method
  void _increment() {
    setState(() {
      _counter ++;
    });
  }
  //UI (user interface)
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text("You pushed the button this many times : "),
            Text(_counter.toString(),
            style: TextStyle(fontSize: 50),
            ),
            //button
            ElevatedButton(
              onPressed: _increment,
              child: Text('Increment'),
            )
          ],
        ),
      ),
    );
  }
}
```

User input

```dart
import 'package:flutter/material.dart';
class TodoPage extends StatefulWidget {
  const TodoPage({super.key});
```

```dart
  @override
  State<TodoPage> createState() => _TodoPageState();
}

class _TodoPageState extends State<TodoPage> {
  //controller pour recuper ce que l'utilisateur a entrer
  TextEditingController myController = TextEditingController();
  //variable message pour afficher le text et éditer
  String greetingMessage = '';
  //greeet user method
  void greetUser(){

    //affichage console
    //print(myController.text);
    //afficher visible par l'utilisateur
    String userName = myController.text;
    setState(() {
      greetingMessage = "Bonjour, " + userName;
    });
  }
  @override

  Widget build(BuildContext context) {
    return Scaffold(
      body: Center(
        child: Padding(
          padding: const EdgeInsets.all(25.0),
          child: Column(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              //C'est ici j'ai choisie afficher text remplie par l'utilisateur
              Text(greetingMessage),
              //textfield
              TextField(
                controller: myController,
                decoration: const InputDecoration(
                  border: OutlineInputBorder(),
                  hintText: "Enter your name:",
                  prefixIcon: Icon(Icons.person),
                ),
              ),

              //button
              ElevatedButton(
                onPressed: greetUser,
                child: Text("Entrer"),
```

```
              )
            ],
          ),
        ),
      ),
    );
  }
}
```

Gesturedetector

```
```