

Лекция 7 - ООП и Dart. Создание классов и объектов. Создание Flutter проекта

ООП – важная часть проектирования в программировании. Язык Dart является объектно-ориентированным языком программирования. В ходе урока мы с вами рассмотрим способ использования классов и объектов в языке Дарт.

На начальном этапе ООП – это тёмный лес, в котором многое непонятно и слишком усложнено. На самом деле всё вовсе не так. Предлагаем абстрагироваться от специфических (непонятных) определений и познакомиться с ООП простыми словами.

ООП простыми словами

Поскольку на примере все усвоить гораздо проще, то давайте за пример возьмем робота, которого постараемся описать за счёт классов в ООП.

Класс в случае с роботом – это его чертёж. Экземпляром класса (объектом) называют целый робот, который создан точно по чертежу.

Наследование – это добавление полезных опций к чертежу робота. К примеру, берем стандартный чертёж робота и дорисуем к нему лазеры, крылья и броню. Все эти дорисовки мы сделаем в классе наследнике, основной функционал которого взят из родительского класса.

Полиморфизм – это общий функционал для всех роботов и не важно что каждый робот может очень сильно отличаться друг от друга. К примеру, в главном классе мы указываем возможность передвижения для всех последующих роботов. Далее в классе наследнике мы можем дополнительно указать возможность левитации для робота, в другом же классе укажем возможность передвижения по воде и так далее. Получается, что есть общий функционал что записан в главном чертеже, но его можно переписать для каждого последующего робота (для каждого наследника).

А **инкапсуляция** является для нас бронёй, защищающей робота. Под пластырем брони находятся уязвимые элементы, вроде проводов и микросхем. После прикрытия брешей с помощью брони, робот полностью защищён от внешних вмешательств. По сути, мы делаем доступ ко всем полям лишь за счёт методов, тем самым прямой доступ к полю будет закрыт.

У всех классов методы могут отличаться, как и поля с конструкторами. Каждый класс позволяет создавать любое количество разных объектов, все из них имеют собственные характеристики.

Создание классов

Для создания класса необходимо прописать ключевое слово `class` и далее название для класса. Общепринято начинать названия классов с буквы в верхнем регистре, но если этого не сделать, то ошибки не будет.

В любом классе можно создавать поля (переменные), методы (функции), а также конструкторы.

Создав новый класс и поместив туда какую-либо информацию мы можем создавать на основе него новые объекты. Объекты будут иметь доступ ко всем характеристикам класса.

Пример простого класса приведен ниже:

```
.vscode > main2.dart > ...
1  class Book {
2      int pages;
3      String name;
4      double weight;
5
6      void getInfoBook () {
7          print('В книге $name находится $pages страниц. ');
8          print('При этом она весит $weight');
9      }
10 }
11
```

На основе такого класса мы можем создать множество объектов. Каждый объект в данном случае будет представлять из себя конкретную книжку. Для каждого объекта мы можем указать уникальные данные: количество страниц, название книги и её вес.

Чтобы создать объект нам потребуется следующий код:

```
.vscode > main2.dart > ...
1  var sherlock_holms = Book(); // Создание объекта
2  sherlock_holms.getInfoBook(); // Вызов метода класса
3
```

Чтобы брать данные из класса через объект необходимо ставить точку и указывать имя переменной или функции, которую мы хотим взять.

Пример 1 - Исходный код

```
main.dart > ...
Run | Debug
1 void main() {
2   var bob = User('Bob', 40, true, ['Football', 'Skate']);
3   bob.info();
4
5   var alex = User('Alex', 25, false, ['Basketball']);
6   alex.info();
7 }
8
9 class User {
10  late String name;
11  late int age;
12  late bool isHappy;
13  late List<String> hobbies;
14
15  User(String name, [int? age, bool? isHappy, List<String>? hobbies]) {
16    this.name = name;
17    this.age = age ?? 0; // Используем значение по умолчанию, если аргумент null
18    this.isHappy = isHappy ?? false; // Используем значение по умолчанию, если аргумент null
19    this.hobbies = hobbies ?? []; // Используем значение по умолчанию, если аргумент null
20  }
21
22  void info() {
23    var happyStatus = isHappy ? 'happy' : 'not happy';
24    print('User $name is $age years old. He is $happyStatus. His hobbies:');
25    for (var hobby in hobbies) {
26      print(hobby);
27    }
28  }
29 }
30
```

Задание 1 - Создание класса

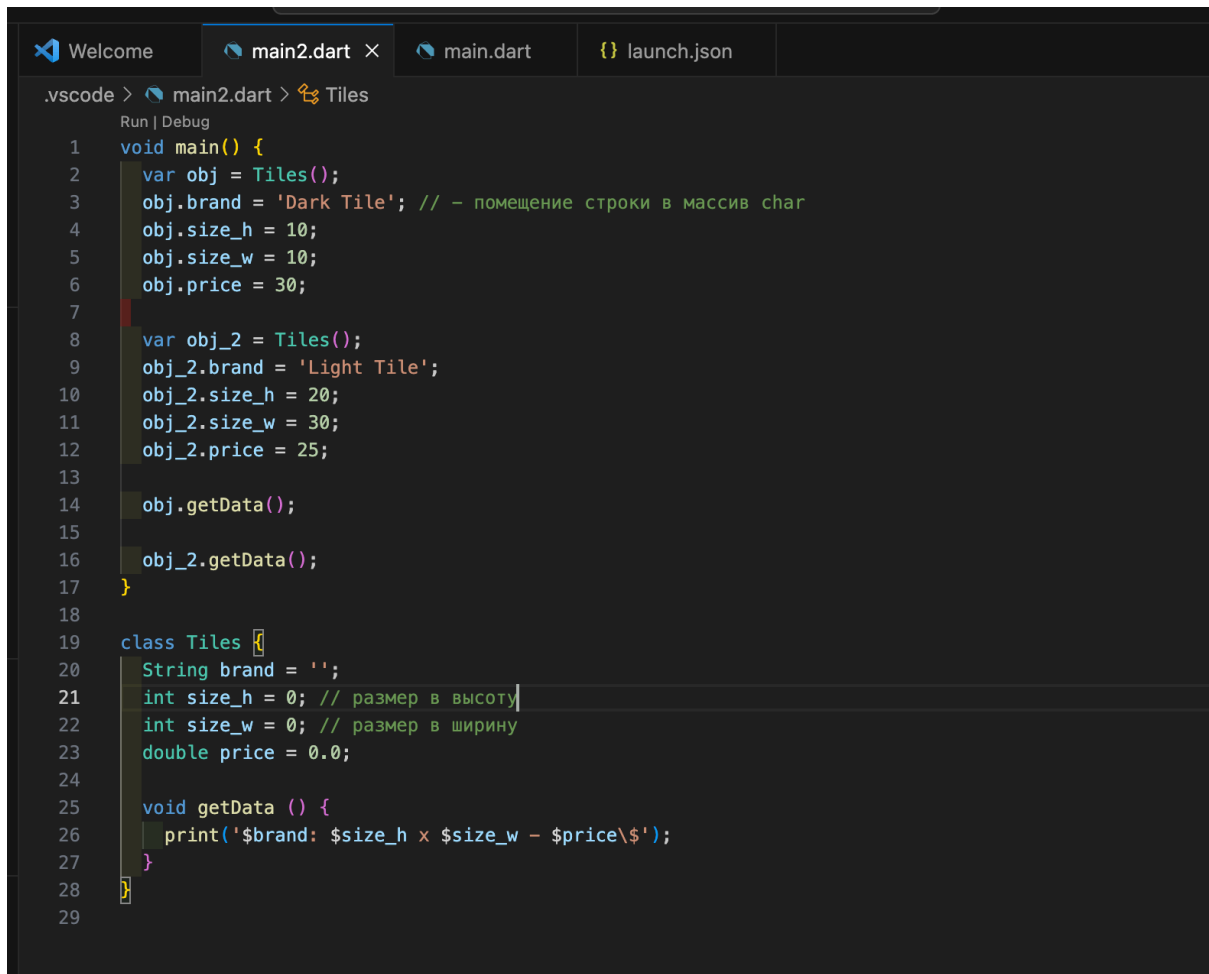
Создайте класс Tiles (кафель). Поместите в него:

поля: brand, size_h, size_w, price;

метод getData() для вывода всей информации.

Создайте 2 объекта на основе класса и внесите в них данные. Отобразить данные через метод getData().

Решение:



```
.vscode > main2.dart > Tiles
Run | Debug
1 void main() {
2   var obj = Tiles();
3   obj.brand = 'Dark Tile'; // - помещение строки в массив char
4   obj.size_h = 10;
5   obj.size_w = 10;
6   obj.price = 30;
7
8   var obj_2 = Tiles();
9   obj_2.brand = 'Light Tile';
10  obj_2.size_h = 20;
11  obj_2.size_w = 30;
12  obj_2.price = 25;
13
14  obj.getData();
15
16  obj_2.getData();
17 }
18
19 class Tiles {
20   String brand = '';
21   int size_h = 0; // размер в высоту
22   int size_w = 0; // размер в ширину
23   double price = 0.0;
24
25   void getData () {
26     print('$brand: $size_h x $size_w - $price$');
27   }
28 }
29
```

Создание Flutter проекта

При разработке на Flutter требуется виртуальное устройство, где можно будет тестировать проекты. За урок мы скачаем все необходимое и создадим шаблонный проект. Дополнительно мы запустим проект на виртуальном устройстве.

Полезные ссылки:

- Скачать [Андроид Студио](#);
- [Flutter SDK](#).

Создание программы

Создать проект на Flutter можно используя разные текстовые редакторы. В ходе курса мы используем редактор Андроид Студио, так как он сразу же предоставляет возможность запустить виртуальное устройство.

На самом деле, то вести разработку можно и в других программах, к примеру, в Visual Studio. Программа Андроид студио нам требуется лишь для одной обязательной вещи – для создания и запуска виртуального устройства.

При создании новых проектов вам всегда нужно указать Flutter SDK. Скачать его можно с их [официального веб сайта](#).

Тестирование программ

Тестировать Flutter проект можно тремя разными способами:

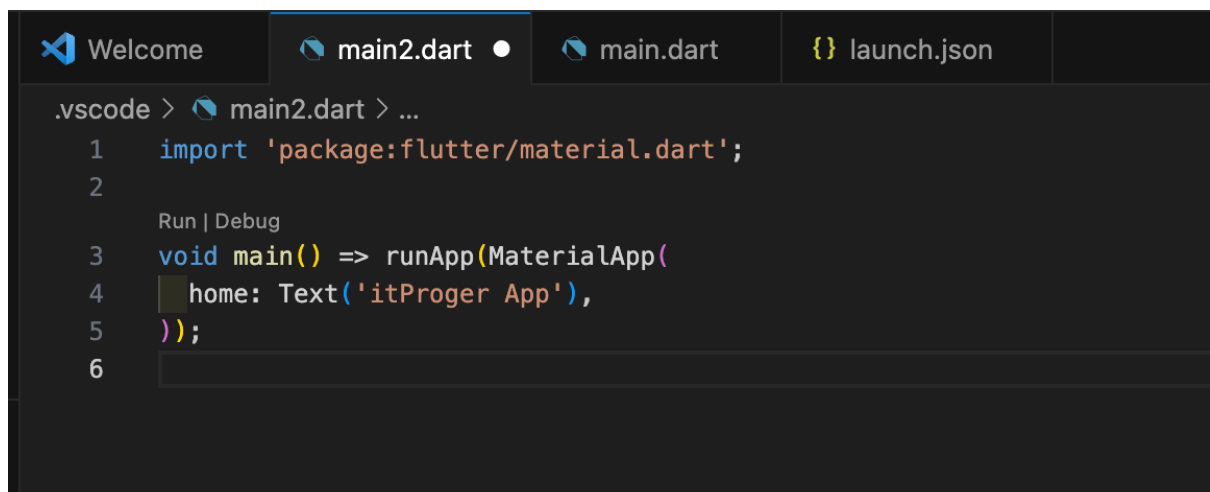
1. Тестирование проекта на Андроид
2. Тестирование проекта на iOS
3. Тестирование проекта в веб браузере

Если вы хотите протестировать проект на телефоне, то сперва нужно установить эмулятор на компьютер. Для iOS он устанавливается вместе с программой Xcode. Для Андроид он устанавливается вместе с программой Android Studio.

В Андроид Студио можно создать виртуальное устройство на базе Андроид. В программе Xcode можно создать виртуальное устройство на базе iOS.

Исходный код

Файл «main.dart»

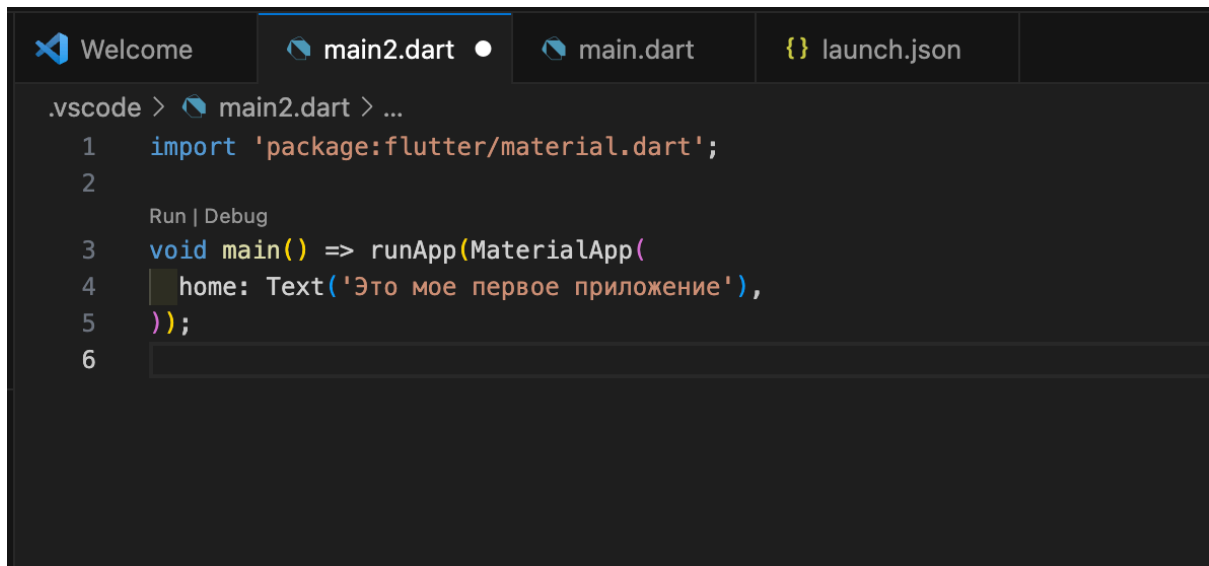


```
.vscode > main2.dart > ...
1  import 'package:flutter/material.dart';
2
   Run | Debug
3  void main() => runApp(MaterialApp(
4    home: Text('itProger App'),
5  ));
6
```

Задание 2 - Вывод информации

Выведите на экран текстовую надпись: «Это мое первое приложение».

Решение задания:



```
.vscode > main2.dart > ...
1  import 'package:flutter/material.dart';
2
   Run | Debug
3  void main() => runApp(MaterialApp(
4    home: Text('Это мое первое приложение'),
5  ));
6
```

Основные виджеты приложения

Любое приложение состоит из множества виджетов (объектов). За урок мы научимся прописывать различные виджеты и создавать базовый каркас приложения.

Полезные ссылки:

- [Документация](#) Flutter.

Основные виджеты

Любое Flutter приложение состоит из множества виджетов (объектов), что отвечают за вывод определенного элемента на экран. За урок мы рассмотрели такие виджеты, как:

- **Scaffold** – виджет для создания основного слоя для расстановки объектов
- **Text** – виджет для создания текстовых надписей
- **AppBar** – виджет для создания шапки приложения
- **Center** – виджет для расположения объектов по центру
- **FloatingActionButton** – виджет для создания «плавающей» кнопки сбоку приложения

Помимо этих основных виджетов существуют и другие виджеты, которые мы будем изучать по мере прохождения курса.

Полный перечень всех виджетов доступен на официальном сайте в разделе [документация](#).

Widgets: работа с ними

Каждый виджет не требуется импортировать по отдельности. Все они импортируются из общего пакета «flutter/material.dart». В каждый виджет вы можете прописать его уникальные свойства. Это могут быть: стили, характеристики, дочерние объекты и общие настройки для самого виджета.

Пример 1 -Исходный код

Файл «main.dart»

```
.vscode > main2.dart > ...
1  import 'package:flutter/material.dart';
2
   Run | Debug
3  void main() => runApp(MyApp());
4
5  class MyApp extends StatelessWidget {
6    @override
7    Widget build(BuildContext context) {
8      return MaterialApp(
9        theme: ThemeData(primaryColor: Colors.deepOrangeAccent),
10       home: Scaffold(
11         appBar: AppBar(
12           title: Text('itProger App'),
13           centerTitle: true,
14         ),
15         body: Center(
16           child: Text('itProger App', style: TextStyle(
17             fontSize: 20,
18             color: Colors.red,
19             fontFamily: 'GoblinOne'
20           )),
21         ),
22         floatingActionButton: FloatingActionButton(
23           child: Text('Нажми'),
24           backgroundColor: Colors.deepOrangeAccent,
25           onPressed: () {
26             print('Clicked');
27           },
28         ),
29       ),
30     );
31   }
32
33 }
34
```

Задание 1 - Стили к шапке

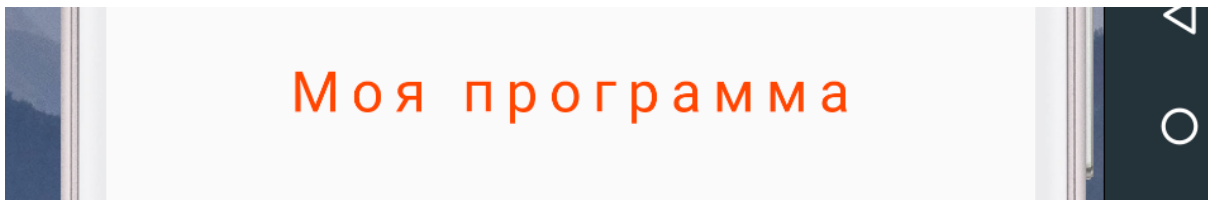
Добавьте шапку приложения. К тексту добавьте стили как показано на фото ниже:

Решение задания:

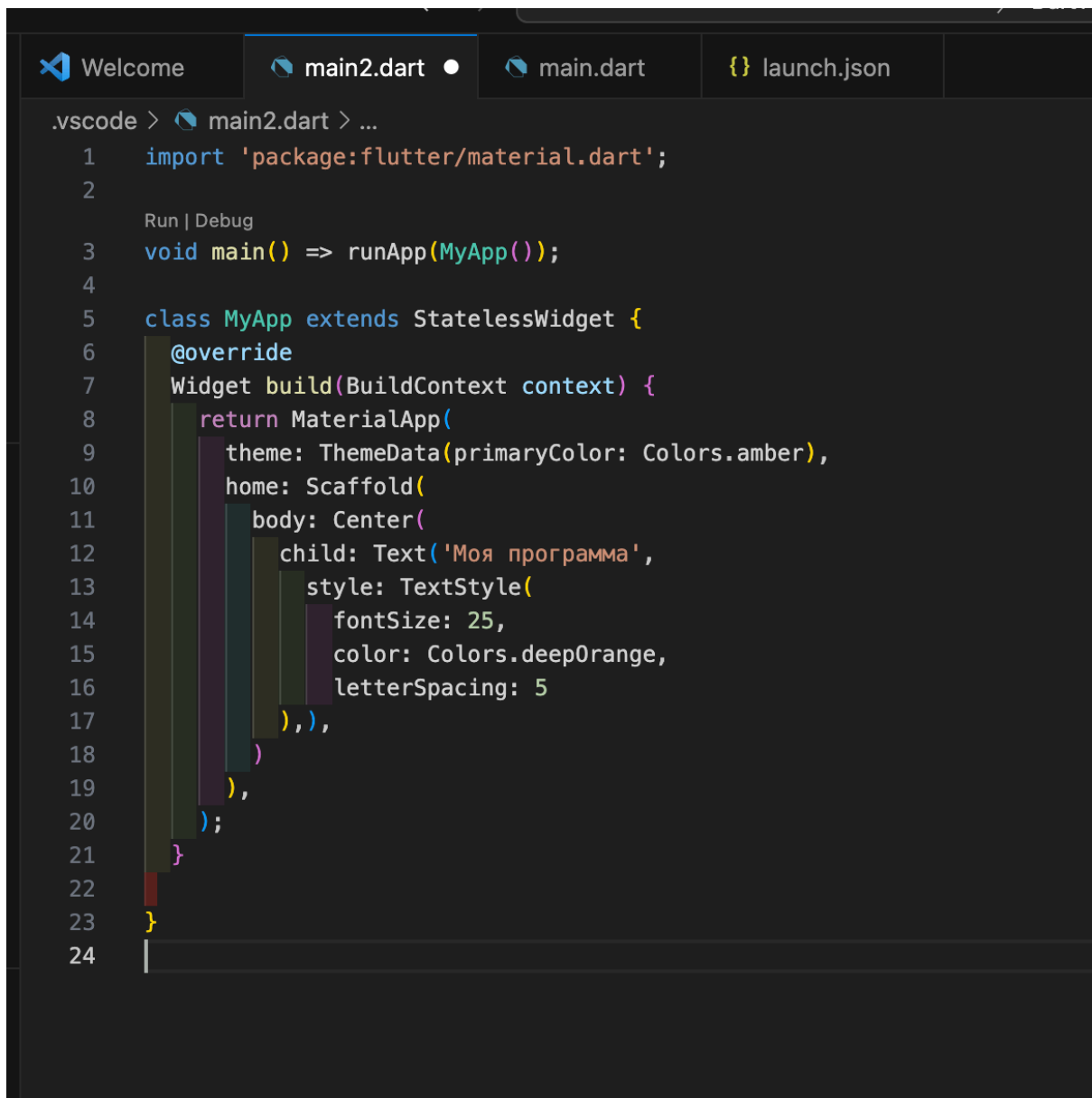
```
.vscode > main2.dart > ...
1  import 'package:flutter/material.dart';
2
   Run | Debug
3  void main() => runApp(MyApp());
4
5  class MyApp extends StatelessWidget {
6    @override
7    Widget build(BuildContext context) {
8      return MaterialApp(
9        theme: ThemeData(primaryColor: Colors.amber),
10       home: Scaffold(
11         appBar: AppBar(
12           title: Text('My App', style: TextStyle(
13             fontSize: 20,
14             color: Colors.black,
15             backgroundColor: Colors.orange,
16           )),
17         centerTitle: true,
18       ),
19     ),
20   );
21 }
22
23 }
24
25 |
```

Задание 2 - Основное содержимое

Создайте приложение, где не будет шапки, но будет основное содержимое. Оно должно выглядеть как на фото ниже:



Решение задания:



```
.vscode > main2.dart > ...
1  import 'package:flutter/material.dart';
2
   Run | Debug
3  void main() => runApp(MyApp());
4
5  class MyApp extends StatelessWidget {
6    @override
7    Widget build(BuildContext context) {
8      return MaterialApp(
9        theme: ThemeData(primaryColor: Colors.amber),
10     home: Scaffold(
11       body: Center(
12         child: Text('Моя программа',
13           style: TextStyle(
14             fontSize: 25,
15             color: Colors.deepOrange,
16             letterSpacing: 5
17           ),),
18       ),
19     ),
20   );
21 }
22
23 }
24
```

Изображения, кнопки и контейнеры

Flutter содержит огромное множество виджетов. За урок мы рассмотрим работу с иконками, картинками, кнопками и контейнерами. На основе этих виджетов можно создавать объекты дизайна, что будут взаимодействовать с пользователем.

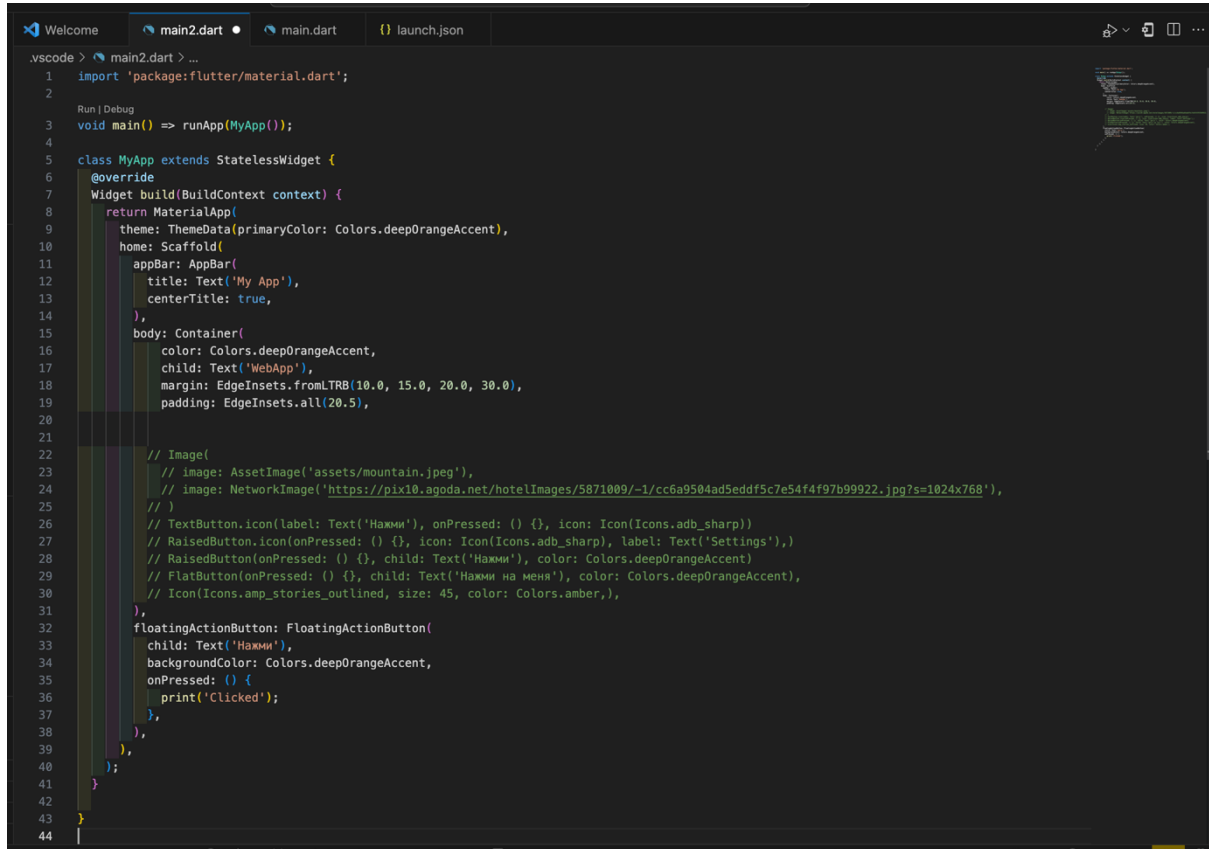
Использование CSS

В Flutter не используется язык стилей CSS. Вместо него вы можете использовать язык программирования Dart. В качестве названий свойств (стилей) вы указываете схожие названия с языком CSS, а в качестве значений вы указываете так же схожие названия значений из языка CSS.

Язык CSS хоть и не используется в Flutter, но его синтаксис используется повсеместно для создания дизайна приложения.

Пример 1 - Исходный код

Файл «main.dart»



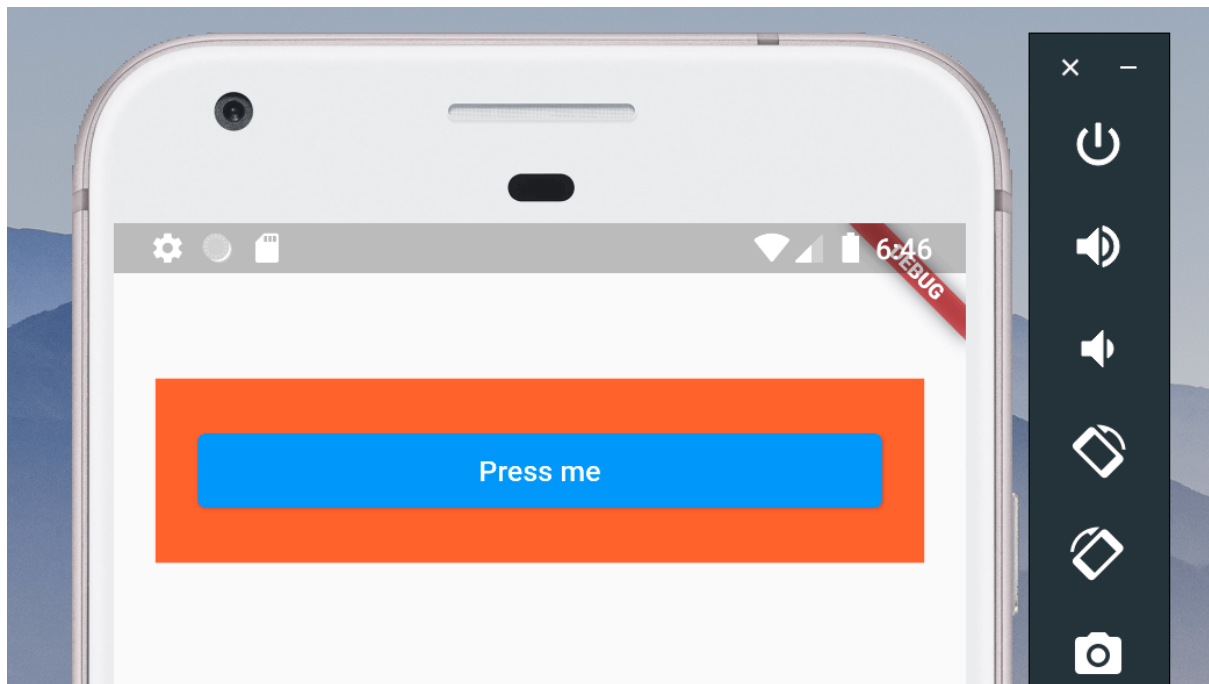
```
1 import 'package:flutter/material.dart';
2
3 void main() => runApp(MyApp());
4
5 class MyApp extends StatelessWidget {
6   @override
7   Widget build(BuildContext context) {
8     return MaterialApp(
9       theme: ThemeData(primaryColor: Colors.deepOrangeAccent),
10      home: Scaffold(
11        appBar: AppBar(
12          title: Text('My App'),
13          centerTitle: true,
14        ),
15        body: Container(
16          color: Colors.deepOrangeAccent,
17          child: Text('WebApp'),
18          margin: EdgeInsets.fromLTRB(10.0, 15.0, 20.0, 30.0),
19          padding: EdgeInsets.all(20.5),
20        ),
21        // Image(
22        //   image: AssetImage('assets/mountain.jpeg'),
23        //   image: NetworkImage('https://pix10.agoda.net/hotelImages/5871009/-1/cc6a9504ad5eddf5c7e54f4f97b99922.jpg?s=1024x768'),
24        // )
25        // TextButton(icon: Text('Нажми'), onPressed: () {}, icon: Icon(Icons.adb_sharp))
26        // RaisedButton(icon: Text('Нажми'), onPressed: () {}, icon: Icon(Icons.adb_sharp), label: Text('Settings'),)
27        // RaisedButton(onPressed: () {}, child: Text('Нажми'), color: Colors.deepOrangeAccent)
28        // FlatButton(onPressed: () {}, child: Text('Нажми на меня'), color: Colors.deepOrangeAccent),
29        // Icon(Icons.amp_stories_outlined, size: 45, color: Colors.amber,),
30      ),
31      floatingActionButton: FloatingActionButton(
32        child: Text('Нажми'),
33        backgroundColor: Colors.deepOrangeAccent,
34        onPressed: () {
35          print('Clicked');
36        },
37      ),
38    );
39  },
40 };
41
42
43
44
```

Задание 1 - Большая кнопка

Создайте контейнер и укажите для него:

- задний фон;
- ширину (width);
- отступы, дабы контейнер не наезжал на верхнее меню.

В контейнере разместите кнопку. По итогу должно получиться следующее приложение:



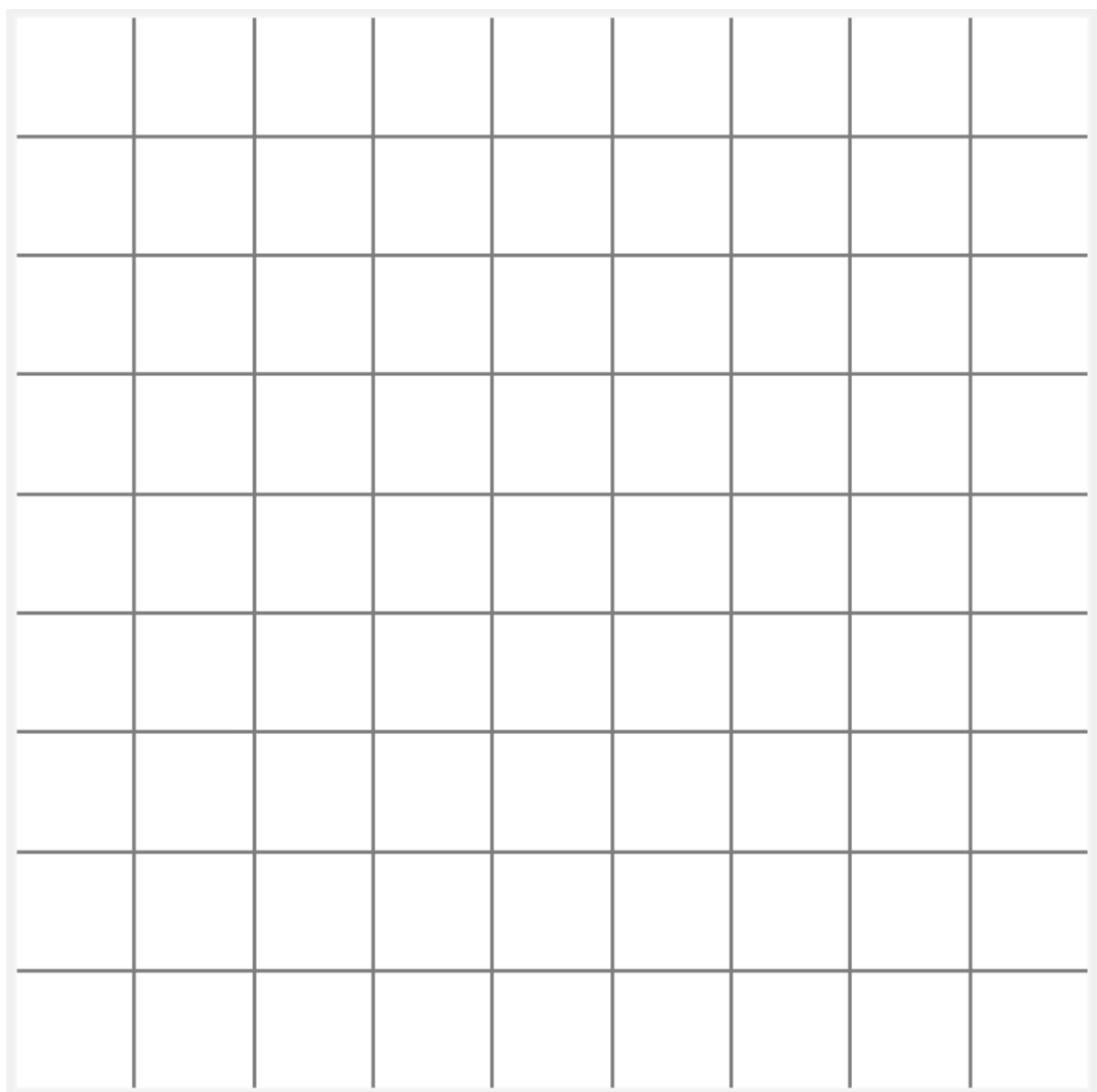
```
.vscode > main2.dart > ...
1  import 'package:flutter/material.dart';
2
   Run | Debug
3  void main() => runApp(MyApp());
4
5  class MyApp extends StatelessWidget {
6    @override
7    Widget build(BuildContext context) {
8      return MaterialApp(
9        home: Scaffold(
10         body: Container(
11           color: Colors.deepOrangeAccent,
12           child: ElevatedButton(child: Text('Press me'), onPressed: () {}),
13           margin: EdgeInsets.fromLTRB(20.0, 75.0, 20.0, 30.0),
14           padding: EdgeInsets.all(20.5),
15           width: 500,
16         ),
17       ),
18     );
19   }
20
21 }
22
23
```

Система сеток «Grid System». Создание приложения с дизайном

До этого урока мы всегда располагали лишь по одному объекту в «body». В ходе урока мы изучим формат Grid System (система сеток). Он позволяет располагать множество объектов на странице приложения, при чем в тех местах что вам нужно.

Расположение объектов в Flutter в основном осуществляется за счет системы сеток (Grid System). В сетке может быть множество рядов, равным счетом как и множество столбцов. В получившихся ячейках можно выводить любые виджеты для реализации нужного вам дизайна.

Пример сетки с ячейками:



Для создания сеток используются ряды и столбцы. Для рядов используйте класс Row, а для столбцов класс Column.

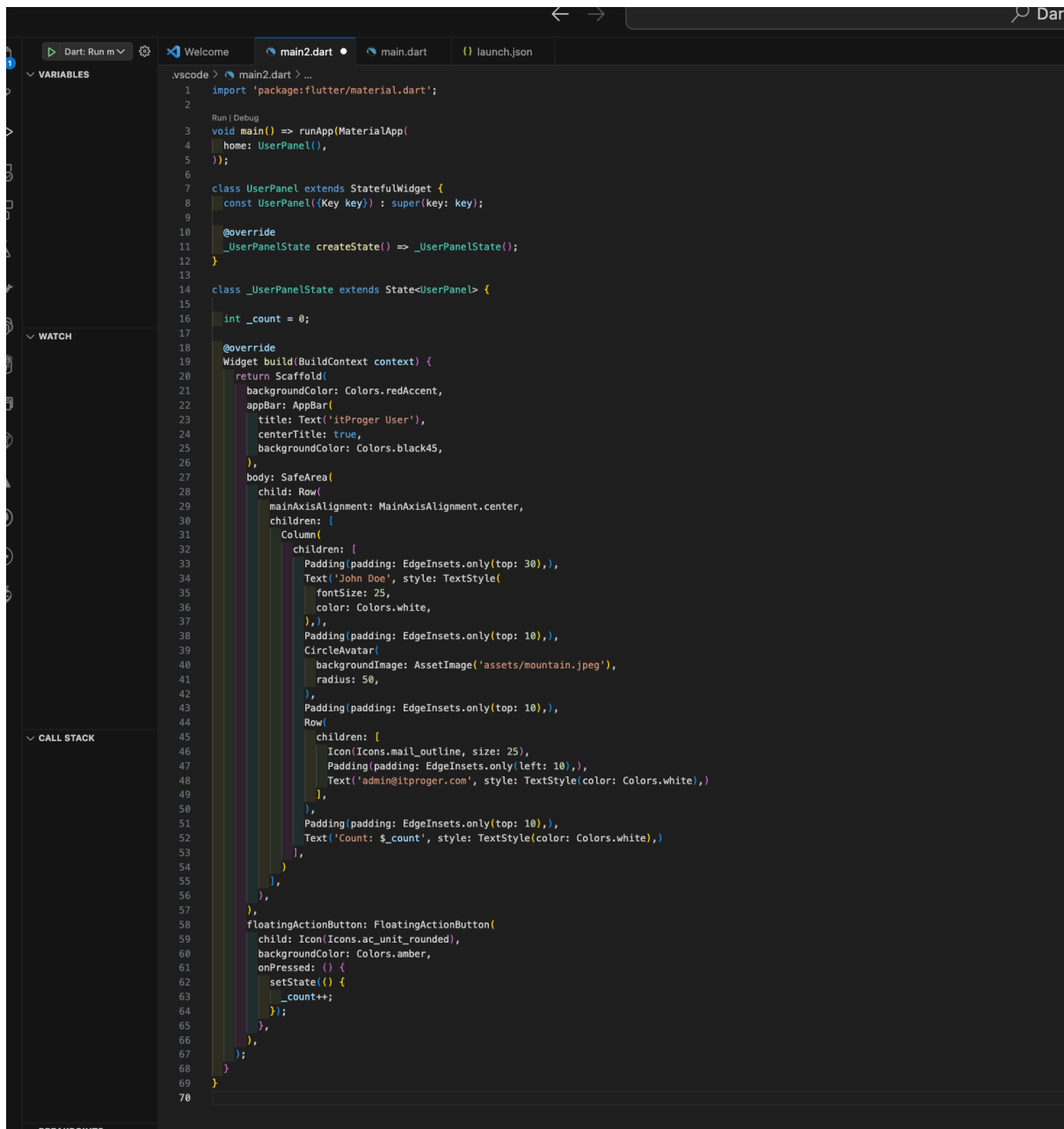
Пример - Исходный код

Система сеток

```

1  import 'package:flutter/material.dart';
2
3  void main() => runApp(MyApp());
4
5  class MyApp extends StatelessWidget {
6    @override
7    Widget build(BuildContext context) {
8      return MaterialApp(
9        theme: ThemeData(primaryColor: Colors.deepOrangeAccent),
10       home: Scaffold(
11         appBar: AppBar(
12           title: Text('itProger App'),
13           centerTitle: true,
14         ),
15         body: Row(
16           mainAxisAlignment: MainAxisAlignment.spaceAround,
17           children: [
18             Column(
19               mainAxisAlignment: MainAxisAlignment.end,
20               crossAxisAlignment: CrossAxisAlignment.start,
21               children: [
22                 Text('Hello'),
23                 TextButton(onPressed: () {}, child: Text('Hello')),
24                 Text('Hello'),
25                 Text('Hello'),
26               ],
27             ),
28             Column(
29               children: [
30                 Text('Hello'),
31                 TextButton(onPressed: () {}, child: Text('Hello'))
32               ],
33             )
34           ],
35         ),
36         floatingActionButton: FloatingActionButton(
37           child: Text('Нажми'),
38           backgroundColor: Colors.deepOrangeAccent,
39           onPressed: () {
40             print('Clicked');
41           },
42         ),
43       ),
44     );
45   }
46
47 }
48
49
```

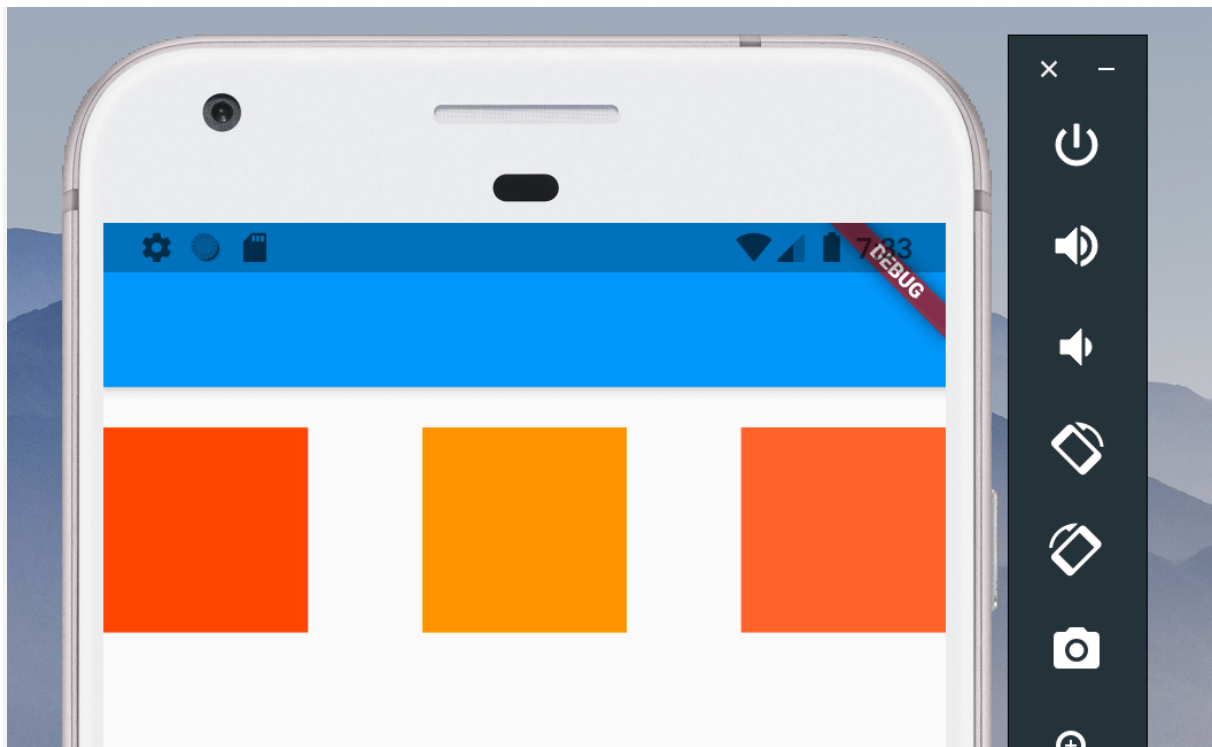
Система сеток



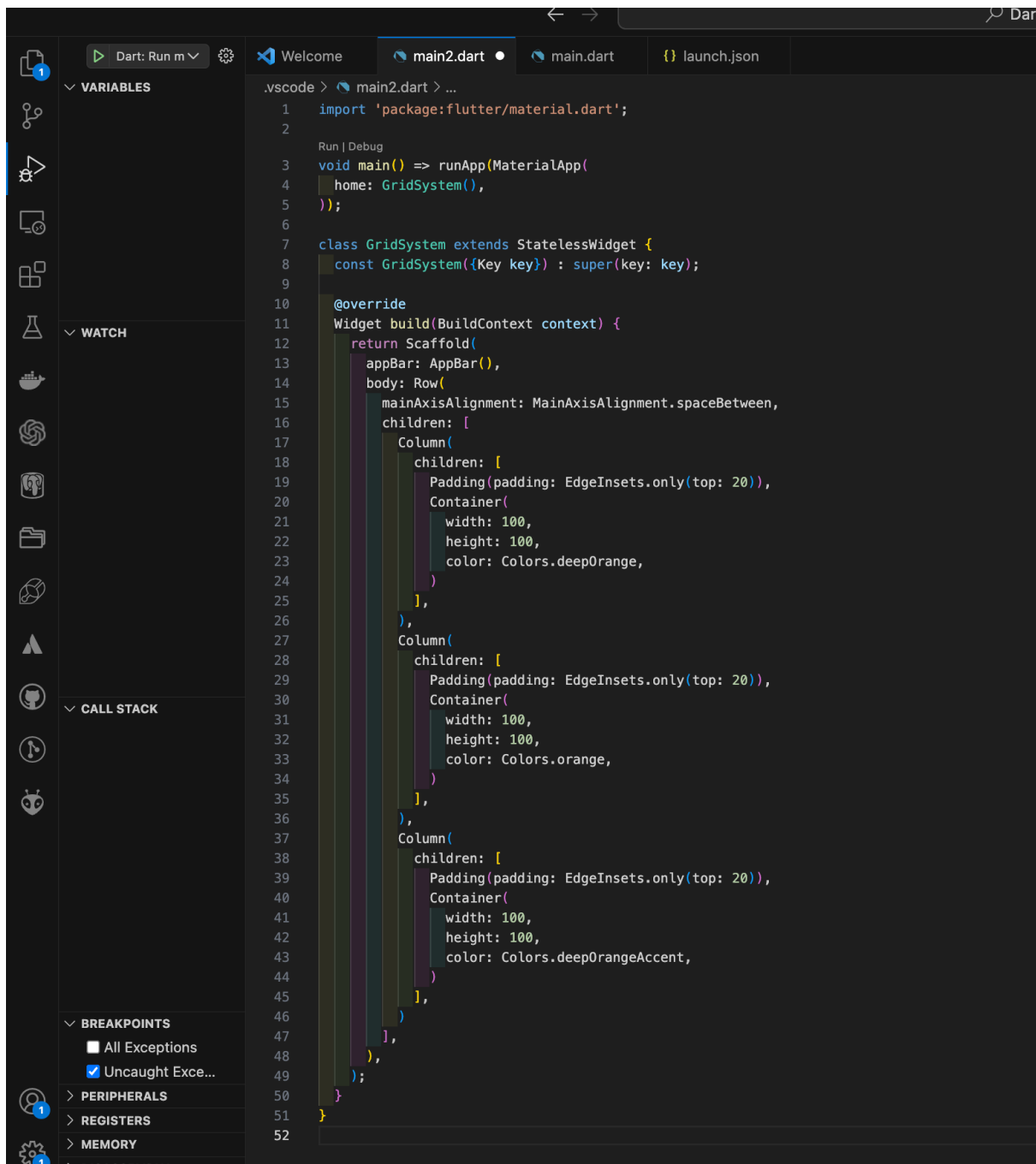
```
1 import 'package:flutter/material.dart';
2
3 void main() => runApp(MaterialApp(
4   home: UserPanel(),
5 ));
6
7 class UserPanel extends StatefulWidget {
8   const UserPanel({Key key}) : super(key: key);
9
10  @override
11  _UserPanelState createState() => _UserPanelState();
12 }
13
14 class _UserPanelState extends State<UserPanel> {
15
16   int _count = 0;
17
18   @override
19   Widget build(BuildContext context) {
20     return Scaffold(
21       backgroundColor: Colors.redAccent,
22       appBar: AppBar(
23         title: Text('itProger User'),
24         centerTitle: true,
25         backgroundColor: Colors.black45,
26       ),
27       body: SafeArea(
28         child: Row(
29           mainAxisAlignment: MainAxisAlignment.center,
30           children: [
31             Column(
32               children: [
33                 Padding(padding: EdgeInsets.only(top: 30)),
34                 Text('John Doe', style: TextStyle(
35                   fontSize: 25,
36                   color: Colors.white,
37                 )),
38                 Padding(padding: EdgeInsets.only(top: 10)),
39                 CircleAvatar(
40                   backgroundImage: AssetImage('assets/mountain.jpeg'),
41                   radius: 50,
42                 ),
43                 Padding(padding: EdgeInsets.only(top: 10)),
44                 Row(
45                   children: [
46                     Icon(Icons.mail_outline, size: 25),
47                     Padding(padding: EdgeInsets.only(left: 10)),
48                     Text('admin@itproger.com', style: TextStyle(color: Colors.white)),
49                   ],
50                 ),
51                 Padding(padding: EdgeInsets.only(top: 10)),
52                 Text('Count: $_count', style: TextStyle(color: Colors.white)),
53               ],
54             ),
55           ],
56         ),
57       ),
58       floatingActionButton: FloatingActionButton(
59         child: Icon(Icons.ac_unit_rounded),
60         backgroundColor: Colors.amber,
61         onPressed: () {
62           setState(() {
63             _count++;
64           });
65         },
66       ),
67     );
68   }
69 }
70
```

Задание - Расстановка блоков

Создайте три контейнера и расположите их как показано на фото ниже:

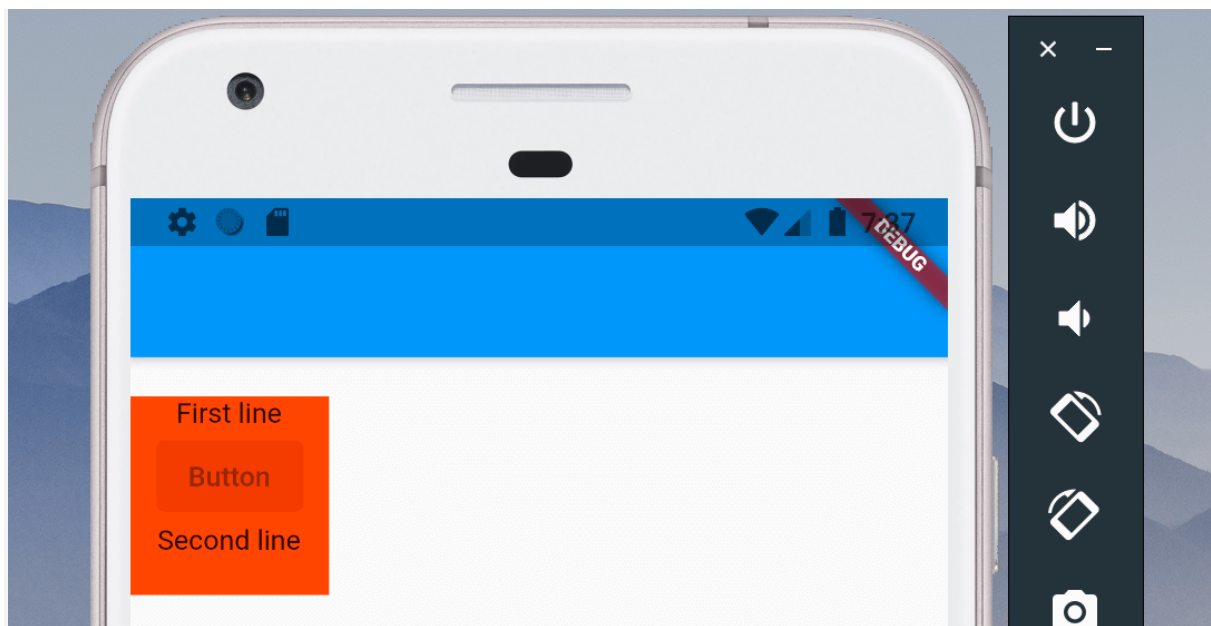


Решение задания:



Одно в другом

Создайте контейнер из прошлого задания. Разместите внутри несколько объектов друг под другом:

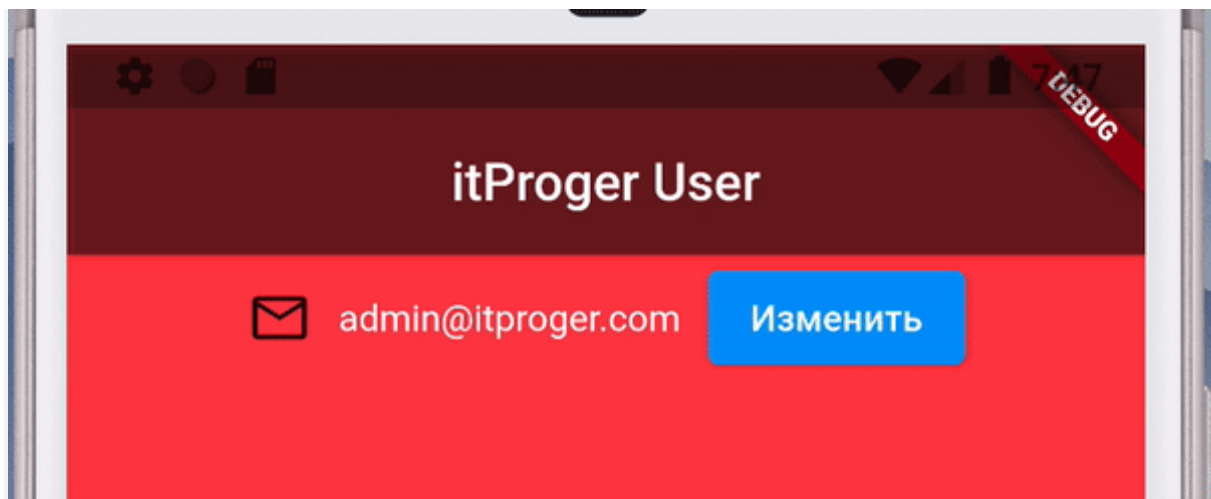


Решение задания:

```
.vscode > main2.dart > ...
1  import 'package:flutter/material.dart';
2
3  Run | Debug
4  void main() => runApp(MaterialApp(
5    home: GridSystem(),
6  ));
7
8  class GridSystem extends StatelessWidget {
9    const GridSystem({Key key}) : super(key: key);
10
11    @override
12    Widget build(BuildContext context) {
13      return Scaffold(
14        appBar: AppBar(),
15        body: Row(
16          mainAxisAlignment: MainAxisAlignment.spaceBetween,
17          children: [
18            Column(
19              children: [
20                Padding(padding: EdgeInsets.only(top: 20)),
21                Container(
22                  width: 100,
23                  height: 100,
24                  color: Colors.deepOrange,
25                  child: Column(
26                    children: [
27                      Text('First line'),
28                      ElevatedButton(child: Text('Button')),
29                      Text('Second line'),
30                    ],
31                  ),
32                ],
33            ),
34          ],
35        ),
36      );
37    }
38  }
39
```

Изменение почты

Создайте программу, в которой при нажатии на кнопку будет происходить изменение email адреса.



Подобную реализацию необходимо создать используя состояния.

Решение задания:

Welcome main2.dart main.dart {} launch.json

.vscode > main2.dart > ...

```
1 import 'package:flutter/material.dart';
2
3 Run | Debug
4 void main() => runApp(MaterialApp(
5   home: UserPanel(),
6 ));
7
8 class UserPanel extends StatefulWidget {
9   const UserPanel({Key key}) : super(key: key);
10
11   @override
12   _UserPanelState createState() => _UserPanelState();
13 }
14
15 class _UserPanelState extends State<UserPanel> {
16   int _count = 0;
17   // Создаем поле и сразу устанавливаем значение
18   String _email = 'admin@itproger.com';
19
20   @override
21   Widget build(BuildContext context) {
22     return Scaffold(
23       backgroundColor: Colors.redAccent,
24       appBar: AppBar(
25         title: Text('itProger User'),
26         centerTitle: true,
27         backgroundColor: Colors.black45,
28       ),
29       body: SafeArea(
30         child: Row(
31           mainAxisAlignment: MainAxisAlignment.center,
32           children: [
33             Icon(Icons.mail_outline, size: 25),
34             Padding(padding: EdgeInsets.only(left: 10)),
35             // Выводим здесь значение из поля
36             Text('$_email', style: TextStyle(color: Colors.white)),
37             Padding(padding: EdgeInsets.only(left: 10)),
38             ElevatedButton(onPressed: () {
39               setState(() {
40                 // При нажатии меняем текст
41                 _email = 'new_email@itproger.com';
42               });
43             }, child: Text('Изменить'))
44           ],
45         ),
46       ),
47     );
48   }
49 }
50
51
```