Методические указания к Практической работе №3
по дисциплине «Разработка кроссплатформенных мобильных приложений»

Тема работы: Виджеты. Stateful Widget

План практической работы:

- Знакомство с подключением ресурсов в приложении
- Знакомство со Stateful Widget
- Выполнение практической работы №3 в соответствии с заданием

Последовательность выполнения практической работы:

1. Знакомство с подключением ресурсов в приложении

Flutter-проект может содержать не только файлы с кодом, но и различные файлы ресурсов, которые называются ассетами *(англ. assets)*. К самым распространённым типам ресурсов относятся статические данные (например, файлы JSON), файлы конфигурации, шрифты, иконки, изображения.

Для начала необходимо перенести нужные файлы в папку проекта. Общепринятое решение — создавать папку с именем assets на уровне папки lib, в ней подпапки с именами images, fonts, icons и т.д. и хранить все ресурсы проекта в них. Дальше необходимо указать расположение ресурса в файле pubspec.yaml в секции flutter:.

Рассмотрим процесс на примере:

- изображений;
- иконок приложения;
- шрифтов.

Изображения и иконки

Пример добавления изображений и иконок в файл проекта pubspec.yaml:

```
flutter:
    assets:
     - assets/images/bg.jpg   # Добавится изображение fon.jpg
     - assets/images/         # Добавятся все изображения из папки images
     - assets/icons/icon.png # Добавится иконка icon.png
```

После добавления в проект доступ к изображению или иконке можно получить через класс AssetImage или Image.asset, например:

```
Image(
  image: AssetImage('assets/images/fon.jpg'),
), //Image
Image.asset('assets/icons/icon.png')
```

Шрифты

Для добавления шрифтов следует добавить файлы формата .ttc, .ttf или .otf, содержащие конфигурацию шрифта (например, скачать с сайта https://fonts.google.com) в папку assets/fonts, и указать их в поле fonts в файле проекта pubspec.yaml::

```
fonts:
   - family: IndieFlower
     fonts:
       - asset: assets/fonts/IndieFlower-Regular.ttf
```

Пример использования зарегистрированного шрифта:

```
Text('Hello, Flutter!',
    style:TextStyle(
      fontSize: 30,
      color: Colors.grey,
      fontFamily: 'IndieFlower',
     ), //TextStyle
)
```

2. Stateful Widget

Statefull Widget – виджеты с изменяемым внутренним состоянием, например в зависимости от какого-то события: нажатие, время и т.д.).

Это виджеты, которые могут менять свое состояние несколько раз и перерисовываться на экране любое количество раз во время работы приложения (текстовые поля, флажки, слайдеры и т.д.).

State – текущее состояние пользовательского интерфейса. Изменение состояния вызывает перерисовку интерфейса.

Statefull Widget имеет объект состояние или State, в котором хранится информация о текущем состоянии виджета.

Создание Statefull Widget :

1) Расширение класса от класса StatefulWidget
2) Переопределение метода State <T> createState(), который возвращает экземпляр класса State

Пример приложения, в котором можно включать и выключать лампочку, изменяя изображение и текст на экране.

Рисунок 1 – Приложение Light Switch

Программный код приложения:

```
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
```

```
  @override
  Widget build(BuildContext context) {
    return MaterialApp(home: LightSwitch());
  }
}

class LightSwitch extends StatefulWidget {
  @override
  LightSwitchState createState() => LightSwitchState();
}

class LightSwitchState extends State<LightSwitch> {
  bool isOn = false; // Переменная состояния света

   @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: Text('Light Switch')),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Image.asset(
              isOn
                  ? 'assets/images/light_on.jpg'
                  : 'assets/images/light_of.jpg',
              // Switch the image based on the light's state
              width: 300,
            ),
            SizedBox(height: 20),
            Text(
              isOn ? 'Свет включен' : 'Свет выключен',
            ), // Изменение текста
            SizedBox(height: 20),
            ElevatedButton(
              onPressed: () {
                setState(() {
                  isOn = !isOn; //Изменение логического значения на противоположное
                });
              },
              child: Text(
                isOn ? 'Выключить свет?' : 'Включить свет?',
              ), // Изменение текста на кнопке
            ),
          ],
        ),
      ),
    );
  }
}
```

В зависимости от значения поля isOn выбираем изображение, текст сообщения и текст на кнопке. Изменяется состояние при помощи обработчика нажатия onPressed у виджета ElevatedButton.
При нажатии на кнопку происходит следующее:

1. Вызываем метод setState для изменения состояния.
2. Меняем значение поля isOn на противоположное.
3. В build-методе возвращаем новые виджеты, используя новое значение isOn. Фреймворк встраивает новые виджеты на экран, и на экране виден актуальный UI.

setState — метод, определённый у State, который уведомляет фреймворк о том, что State изменился и его нужно перерисовать.

Пример приложения, в котором меняется цвет квадрата при нажатии на него: с красного на синий и обратно.

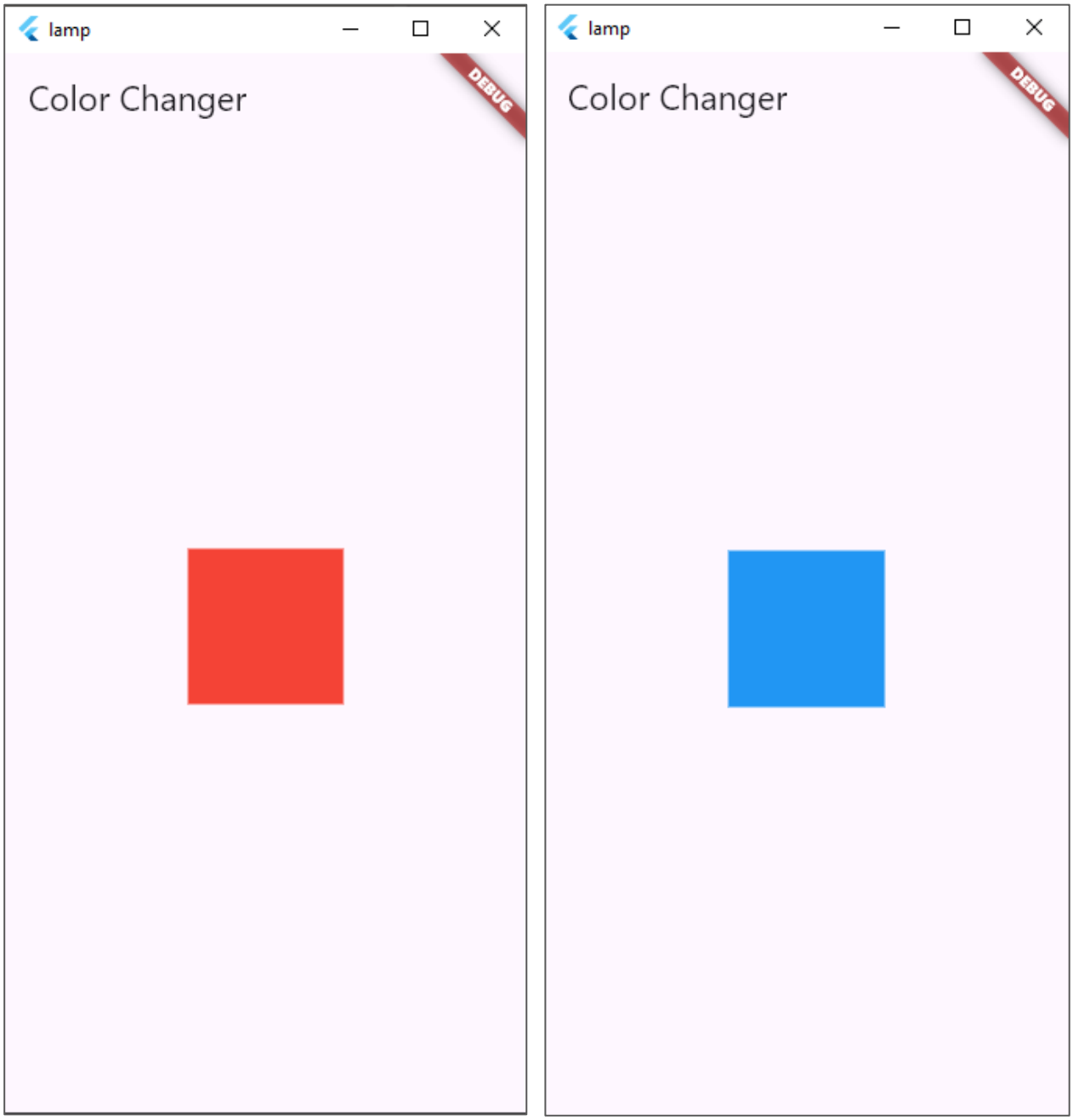

Рисунок 2 – Приложение Color Chander

Программный код приложения:

```dart
import 'package:flutter/material.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Color Changer',
      home: Scaffold(
        appBar: AppBar(title: Text('Color Changer')),
        body: Center(child: ColorChanger()),
      ),
    );
  }
}

class ColorChanger extends StatefulWidget {
```

```
  const ColorChanger({super.key});

  @override
  ColorChangerState createState() => ColorChangerState();
}

class ColorChangerState extends State<ColorChanger> {
  Color boxColor = Colors.blue; // Инициализируем переменную, задав синий цвет
  void changeColor() {
    setState(() {
      boxColor = boxColor == Colors.blue ? Colors.red : Colors.blue;
    });
  }

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: changeColor, // Изменяем цвет при нажатии
      child: Container(
        width: 100,
        height: 100,
        color: boxColor, // Меняем цвет
      ),
    );
  }
}
```

3. Задание на практическую работу

Изменить приложение, созданное на предыдущей практической работе, выполнив следующие действия:

1) Установить для заголовка кастомный шрифт
2) Подготовить 5 изображений по своему варианту и изменить приложение, чтобы в нем производилась циклическая смена изображений в области картинки на макете, при нажатии на кнопку или нажатии на изображении.