Google Developer Student Clubs

Introduction to

Flutter

to-do list

Contents

1 Introduction

In this workshop your goal is to create a simple todo-list application. This mobile application will run using Flutter and on both ios and android.

1.1 Installing Flutter

To properly install Flutter on your computer, follow the instructions given in the Flutter docs. To be able to complete this tp, you need to have installed everything needed to code mobile applications.

Tips

To check that you have installed everything correctly, simply run **flutter doctor** in your terminal.

1.2 Setting up a new flutter project

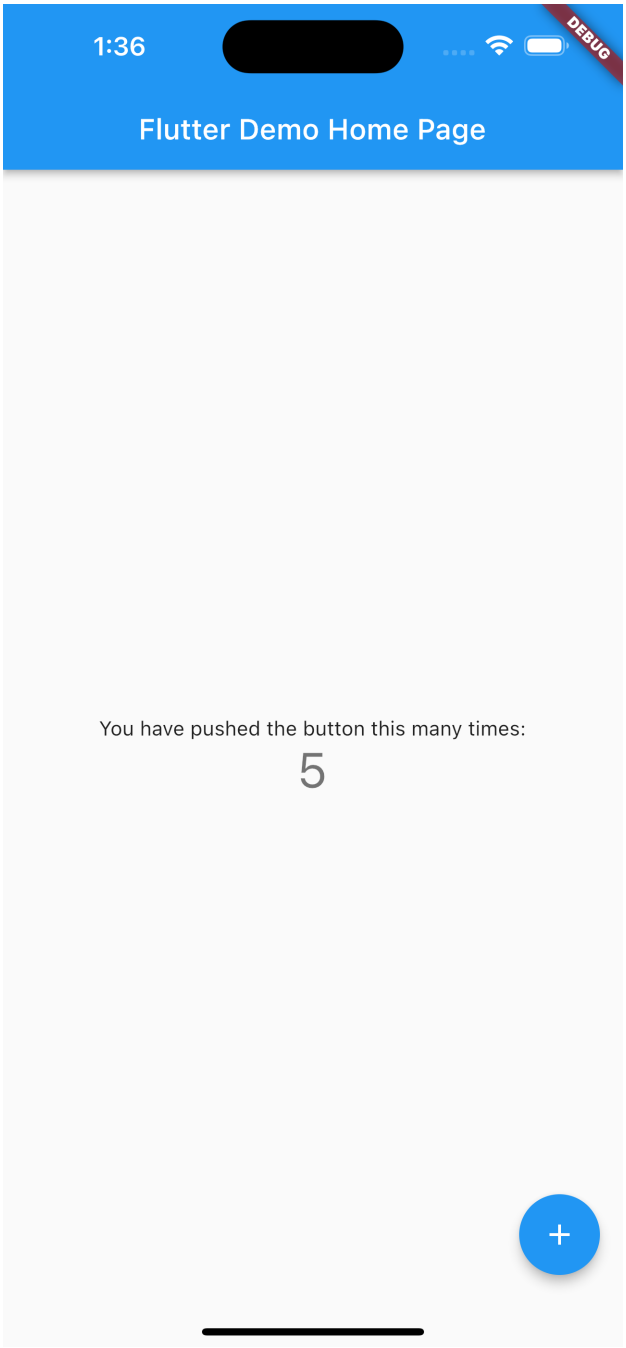
Once you have installed Flutter, you can create a simple Flutter app.

```
$ flutter create mytodo
Creating project mytodo...
Running "flutter pub get" in mytodo... 1,213ms
Wrote 127 files.

All done!
In order to run your application, type:

    $ cd mytodo
    $ flutter run

Your application code is in mytodo/lib/main.dart.
```

Now go to the application's folder and try running it for the first time.

```
$ cd mytodo
/mytodo$ ls
README.md  lib  pubspec.lock  windows  analysis_options.yaml
linux  pubspec.yaml  android  macos  test  ios  mytodo.iml
    web
/mytodo$ flutter run
```

If everything went well, when the build is finished, the following screen should appear.

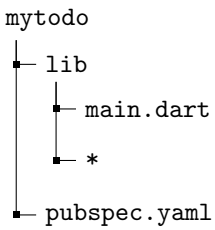


Congratulations, you have correctly setup your Flutter workspace and you are now ready to begin coding the application!

2 Todo list

2.1 Architecture

File tree



The only files interesting for you will be the ones located in the lib directory and pubspec.yaml. Files in the lib directory are the source files of our application. When the application starts, the code in main.dart is executed.

As in many other languages, it is the main function that is executed.

```
1 void main() {
2         runApp(const MyApp());
3 }
```

Pubspec.yaml contains all the dependencies and packages that we are using in our application. So any time, we want to use a new package in our app. By default, you can find explanations of how to use this file in the file itself.

You are allowed to add as many files into the lib directory as you want, but try to not make it too confusing for yourself!

2.2 General

Since this is most probably your first project using Flutter and Dart, I advice you to read the Flutter docs carefully! Most of the answers to your questions can be found in them.

3 Core features

Tip

Todo list application is a pretty simple application to start learning flutter, so start simple and build from there!

In a todolist, each todo will at minimum have a content (name) and a value stating whether the task is finished or not.

```
1 class Todo {
2         String name;
3         bool checked;
4 }
```

The two most basic features that our todo list application simply must have is add/create and delete/check a todo.

3.1 Create a todo

3.1.1 Add Button

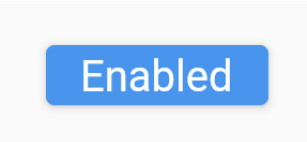
First, you are going to need a button that will trigger the event of creating a todo. For that, we can use the ElevatedButton class. Note that there are other variatons of the Button class, if you wish to give your app a different look.

Tip

You can just modify the floating button that has been used to increase the counter in the demo application that starts up when you create an app in Flutter.

```
1 final ButtonStyle style =
2         ElevatedButton.styleFrom(textStyle: const TextStyle(fontSize: 20));
3
4 . . .
5
6 ElevatedButton(
7         style: style,
8         onPressed: () ,
9         child: const Text('Enabled'),
10 ),
```

In the code snippet above, you can see an example usage of the ElevatedButton class. On lines 1 and 2, we defined a style for our button which we then used in the definition of the button. We also assigned a child for out button which is a Text box, displaying the word "Enabled". Such button will look something like this.

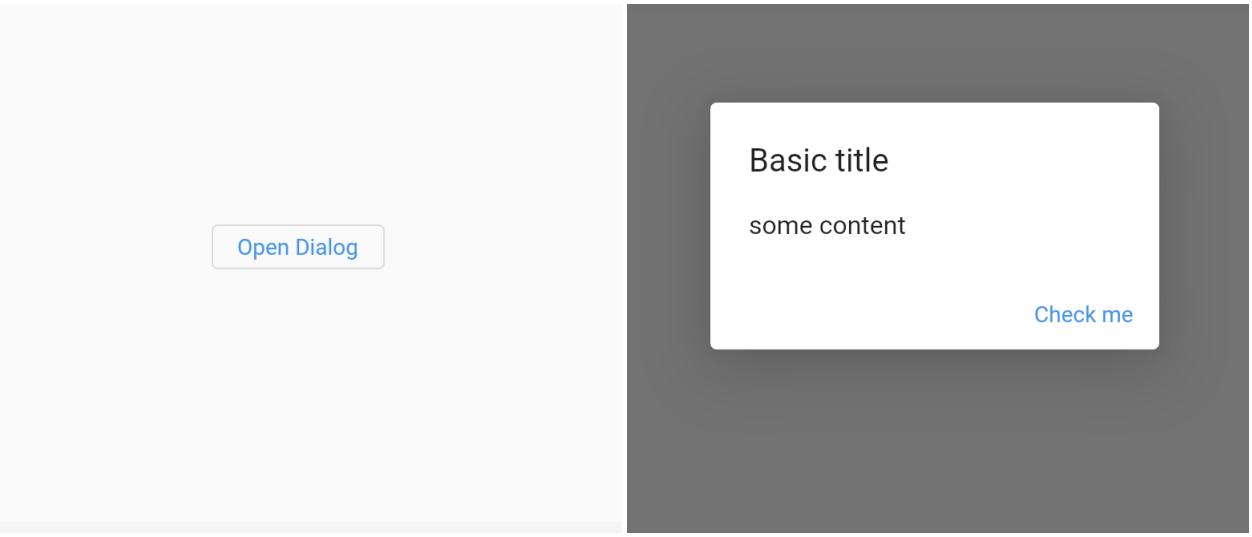


3.1.2 Add todo dialog

After the user clicks on the add button, we want a dialog to appear which would allow the user to write the name of his todo and confirm or discard this new todo.
This can be simply done using the showDialog function and the AlertDialog class. We invite you to study the documentation to fully understand this function.

```
return showDialog<void>(
  context: context,
  builder: (BuildContext context) {
    return AlertDialog(
      title: const Text('Basic title'),
      content: const Text('some content'),
      actions: <Widget>[
        TextButton(
          style: TextButton.styleFrom(
            textStyle: Theme.of(context).textTheme.labelLarge,
          ),
          child: const Text('Check me'),
          onPressed: () {
            Navigator.of(context).pop();
          },
        ),
```

In the code snippet above you can see an example usage of showDialog and AlertDialog. After clicking on the button Open Dialog a dialog appears, as shown in the two pictures below. You can see the full interactive example in the docs of showDialog.



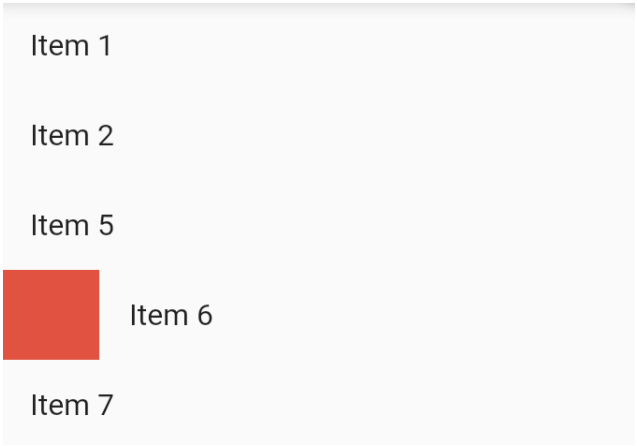
3.2 Check a todo

After the user checks a todo, the boolean check should be changed to true. After that, the todo item should be crossed to demonstrate that the task has been completed.

3.3 Delete a todo

As any ordinary todo, you need to handle the delete operation. You can add a simple delete button on the side of each todo, using for example the IconButton class.

Or, if you feel like challenging yourself a little more, you can implement a swipe to dismiss pattern. Here you can find an interactive example of the swipe to dismiss pattern.



4 Advanced features

If you had finished all the core features of the todo list, first of all congratulations! You can add as many advanced features as you want and customise it according to your taste!

4.1 Adding tags to todos

As seen in the apple native app Reminders, where you can add a tag to a task in order to clearly remember what is it related to.

4.2 Add a deadline

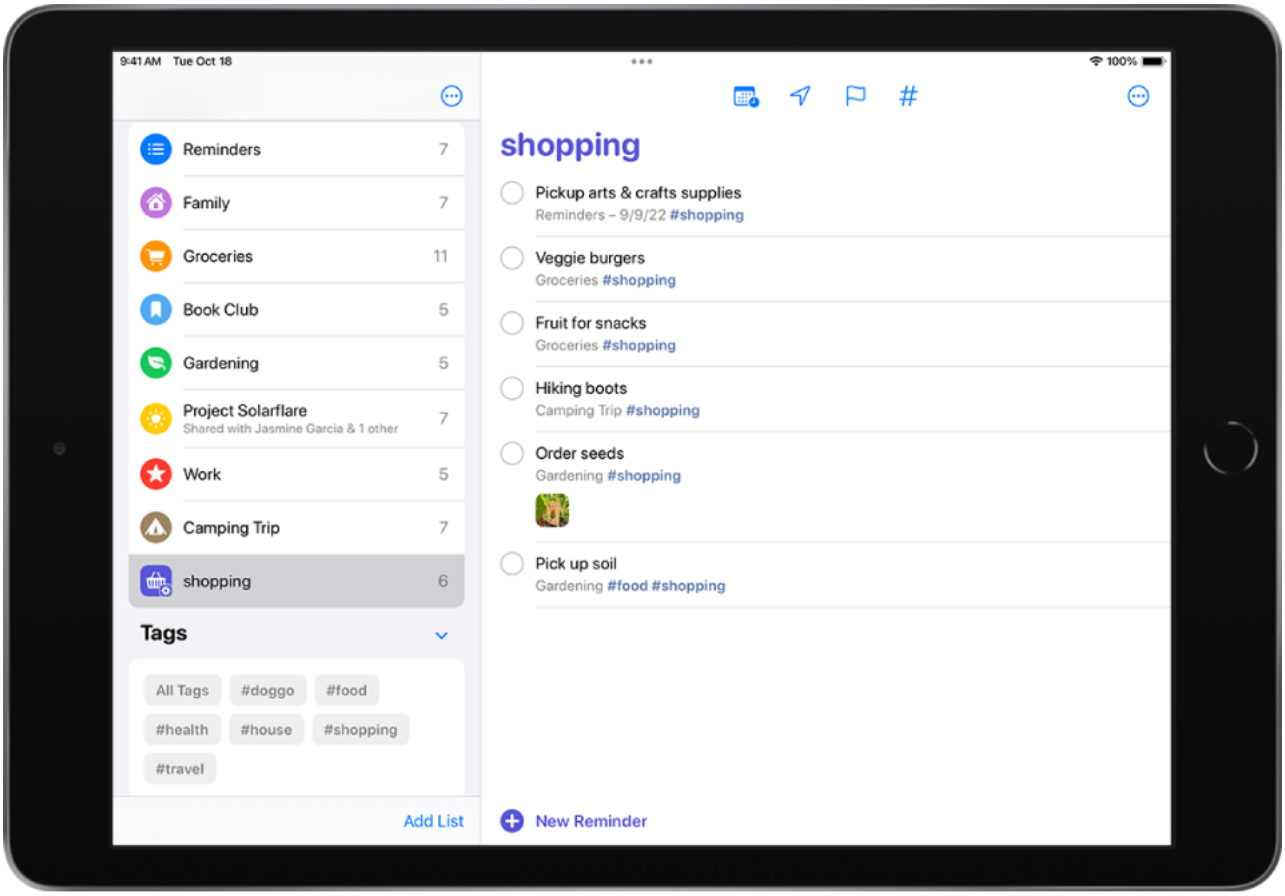
When is the task due? Add a deadline to the todo!

4.3 Multiple lists

Let the user create different lists of todos, name them as they want and let them assign a task to a particular list.

4.4 Make it pretty

Customize the living hell out of it! Change the colours! Add emojis, customized classes, wallpaper, anything that you can come up with!



5 Documentation

For any questions regarding the Dart syntax, check out the Dart tutorial on https://dart.dev/tutorials. Also make sure to go through all the Flutter widgets can be found on https://docs.flutter.dev.