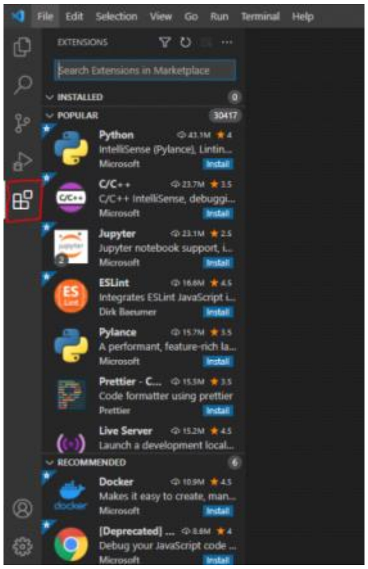
VStudioCode

Wednesday, October 6, 2021

1:19 PM

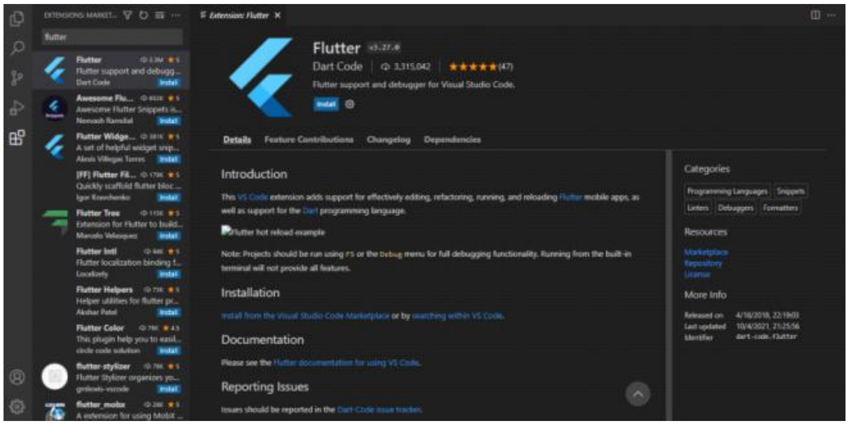
App development we are using VSCode. Follow https://flutter.dev/docs/development/tools/vs-code

1. Click on Extension as shown in RED BOX.


File Edit Selection View Go Run Terminal Help
EXTENSIONS
Search Extensions in Marketplace
INSTALLED 0
POPULAR 30417
Python 43.1M ★4
IntelliSense (Pylance), Lintin...
Microsoft Install
C/C++ 23.7M ★3.5
C/C++ IntelliSense, debuggi...
Microsoft Install
Jupyter 23.1M ★2.5
Jupyter notebook support, i...
Microsoft Install
ESLint 16.6M ★4.5
Integrates ESLint JavaScript i...
Dirk Baeumer Install
Pylance 15.7M ★3.5
A performant, feature-rich la...
Microsoft Install
Prettier - C... 15.5M ★3.5
Code formatter using prettier
Prettier Install
Live Server 15.2M ★4.5
Launch a development local...
RECOMMENDED 6
Docker 10.9M ★4.5
Makes it easy to create, man...
Microsoft Install
[Deprecated] ... 8.8M ★4
Debug your JavaScript code ...
Microsoft Install

2. Type Flutter in search.

   Install flutter as shown in below snapshot


EXTENSIONS: MARKET...
flutter
Flutter
Flutter support and debugg...
Dart Code Install
Awesome Flu...
Awesome Flutter Snippets is...
Install
Flutter Widge...
A set of helpful widget snip...
Install
[FF] Flutter Fil...
Quickly scaffold flutter bloc ...
Install
Flutter Tree
Extension for Flutter to build...
Install
Flutter Intl
Flutter localization binding f...
Localizely Install
Flutter Helpers
Helper utilities for flutter pr...
Install
Flutter Color
This plugin help you to easil...
Install
flutter-stylizer
Flutter Stylizer organizes yo...
Install
flutter_mobx
A extension for using MobX ...
Extension: Flutter
Flutter v3.27.0
Dart Code 3,315,042 ★★★★★(47)
Flutter support and debugger for Visual Studio Code.
Install
Details Feature Contributions Changelog Dependencies
Categories
Programming Languages Snippets Linters Debuggers Formatters
Resources
Marketplace Repository License
More Info
Released on 4/18/2018, 22:19:03
Last updated 10/4/2021, 21:25:56
Identifier dart-code.flutter

Introduction

This VS Code extension adds support for effectively editing, refactoring, running, and reloading Flutter mobile apps, as well as support for the Dart programming language.

Flutter hot reload example

Note: Projects should be run using F5 or the Debug menu for full debugging functionality. Running from the built-in terminal will not provide all features.

Installation

Install from the Visual Studio Code Marketplace or by searching within VS Code.

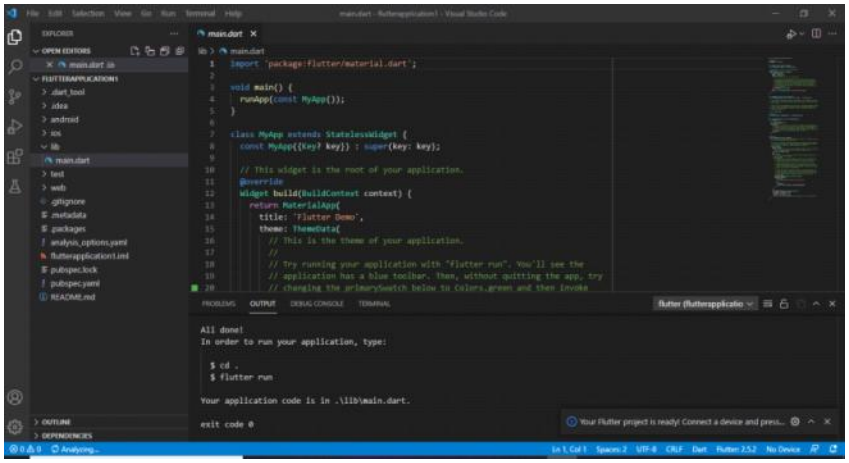
Documentation

Please see the Flutter documentation for using VS Code.

Reporting Issues

Issues should be reported in the Dart-Code issue tracker.

3. Go to View and type Flutter and select Flutter Project and then New Application.
   It will ask : Application name and create a project structure. As in below snapshot.



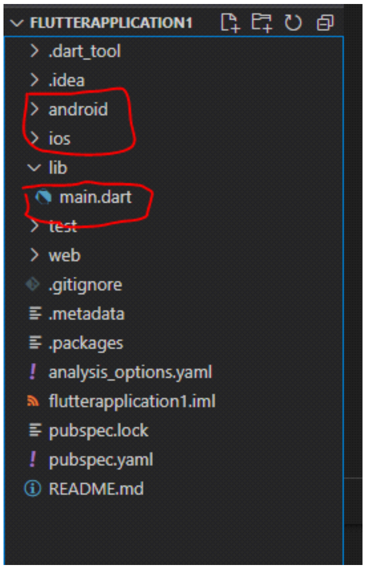
```
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // Try running your application with "flutter run". You'll see the
        // application has a blue toolbar. Then, without quitting the app, try
        // changing the primarySwatch below to Colors.green and then invoke
```

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

  $ cd .
  $ flutter run

Your application code is in .\lib\main.dart.

exit code 0
```

Your Flutter project is ready! Connect a device and press...

4. As you can see in Project Explorer.

Flutter provides android and ios directories as well for android and ios related apps.
Also it has important folder "lib" and has main.dart file.



```
FLUTTERAPPLICATION1
  > .dart_tool
  > .idea
  > android
  > ios
  v lib
      main.dart
  > test
  > web
    .gitignore
    .metadata
    .packages
    analysis_options.yaml
    flutterapplication1.iml
    pubspec.lock
    pubspec.yaml
    README.md
```

If issue in running ADB : Follow below step : Add path in Antivirus

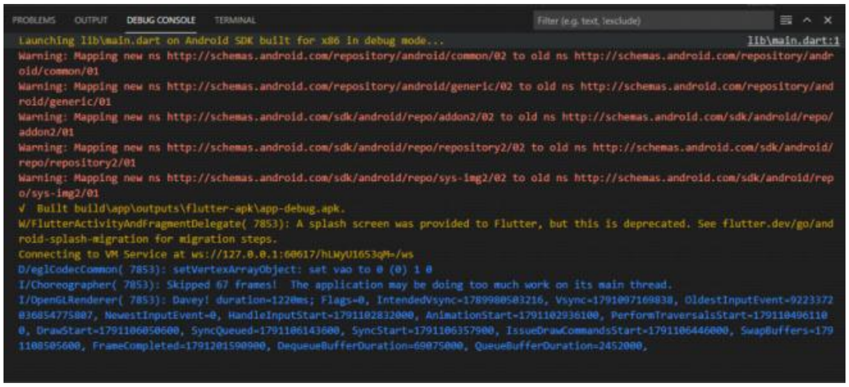
https://stackoverflow.com/questions/46290668/error-could-not-automatically-detect-an-adb-binary-some-emulator-functionality/48035180

1. Default App : Where click on icon increase the counter.

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

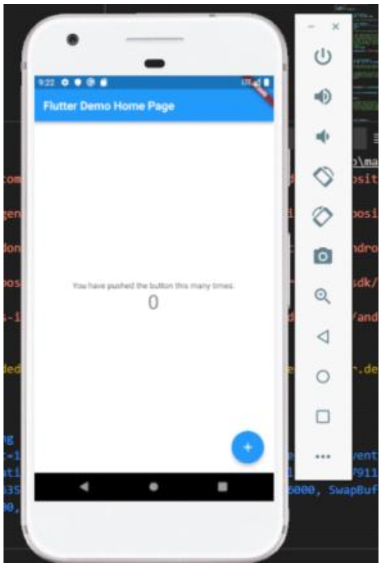
Above package is required to add functionality in App.

2. After successful run : below are the debug logs :



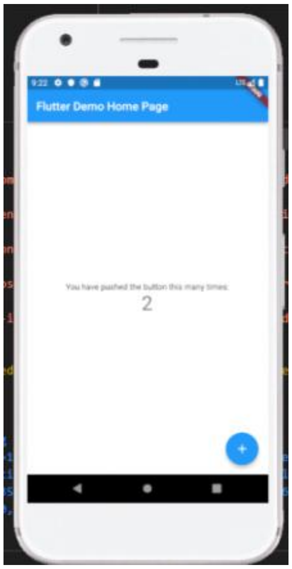
```
Launching lib\main.dart on Android SDK built for x86 in debug mode...                                           lib\main.dart:1
Warning: Mapping new ns http://schemas.android.com/repository/android/common/02 to old ns http://schemas.android.com/repository/andr
oid/common/01
Warning: Mapping new ns http://schemas.android.com/repository/android/generic/02 to old ns http://schemas.android.com/repository/and
roid/generic/01
Warning: Mapping new ns http://schemas.android.com/sdk/android/repo/addon2/02 to old ns http://schemas.android.com/sdk/android/repo/
addon2/01
Warning: Mapping new ns http://schemas.android.com/sdk/android/repo/repository2/02 to old ns http://schemas.android.com/sdk/android/
repo/repository2/01
Warning: Mapping new ns http://schemas.android.com/sdk/android/repo/sys-img2/02 to old ns http://schemas.android.com/sdk/android/rep
o/sys-img2/01
√  Built build\app\outputs\flutter-apk\app-debug.apk.
W/FlutterActivityAndFragmentDelegate( 7853): A splash screen was provided to Flutter, but this is deprecated. See flutter.dev/go/and
roid-splash-migration for migration steps.
Connecting to VM Service at ws://127.0.0.1:60617/hLWyU1653qM=/ws
D/eglCodecCommon( 7853): setVertexArrayObject: set vao to 0 (0) 1 0
I/Choreographer( 7853): Skipped 67 frames!  The application may be doing too much work on its main thread.
I/OpenGLRenderer( 7853): Davey! duration=1220ms; Flags=0, IntendedVsync=1789980583216, Vsync=1791097169838, OldestInputEvent=9223372
036854775807, NewestInputEvent=0, HandleInputStart=1791102832000, AnimationStart=1791102936100, PerformTraversalsStart=179110496110
0, DrawStart=1791106050600, SyncQueued=1791106143600, SyncStart=1791106357900, IssueDrawCommandsStart=1791106446000, SwapBuffers=179
1108505600, FrameCompleted=1791201590900, DequeueBufferDuration=69075000, QueueBufferDuration=2452000,
```

App will open as :



You have pushed the button this many times:

0



You have pushed the button this many times:

2

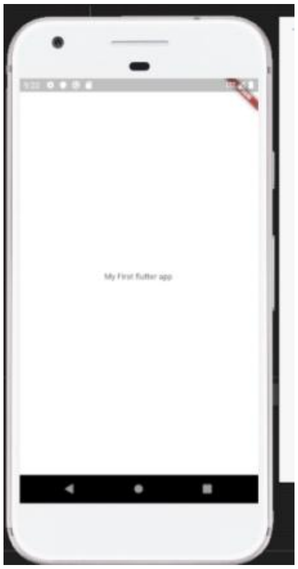
3. pubspec.yaml -> External dependencies or packages need to add in this file.

4. build() method in main.dart is important , it is used to run UI of APP.
   Any updation in app , we need to refresh build method.

Code :

```
import 'package:flutter/material.dart';
void main() {
  runApp(const MyApp());
}
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        // This is the theme of your application.
        //
        // Try running your application with "flutter run". You'll see the
        // application has a blue toolbar. Then, without quitting the app, try
        // changing the primarySwatch below to Colors.green and then invoke
        // "hot reload" (press "r" in the console where you ran "flutter run",
        // or simply save your changes to "hot reload" in a Flutter IDE).
        // Notice that the counter didn't reset back to zero; the application
        // is not restarted.
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
class MyHomePage extends StatefulWidget {
  const MyHomePage({Key? key, required this.title}) : super(key: key);
  // This widget is the home page of your application. It is stateful, meaning
  // that it has a State object (defined below) that contains fields that affect
  // how it looks.
  // This class is the configuration for the state. It holds the values (in this
  // case the title) provided by the parent (in this case the App widget) and
  // used by the build method of the State. Fields in a Widget subclass are
  // always marked "final".
  final String title;
  @override
  State<MyHomePage> createState() => _MyHomePageState();
}
class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;
  void _incrementCounter() {
    setState(() {
      // This call to setState tells the Flutter framework that something has
      // changed in this State, which causes it to rerun the build method below
      // so that the display can reflect the updated values. If we changed
      // _counter without calling setState(), then the build method would not be
      // called again, and so nothing would appear to happen.
      _counter++;
    });
  }
  @override
  Widget build(BuildContext context) {
    // This method is rerun every time setState is called, for instance as done
    // by the _incrementCounter method above.
    //
    // The Flutter framework has been optimized to make rerunning build methods
    // fast, so that you can just rebuild anything that needs updating rather
    // than having to individually change instances of widgets.
    return Scaffold(
      appBar: AppBar(
        // Here we take the value from the MyHomePage object that was created by
        // the App.build method, and use it to set our appbar title.
        title: Text(widget.title),
      ),
      body: Center(
        // Center is a layout widget. It takes a single child and positions it
        // in the middle of the parent.
        child: Column(
          // Column is also a layout widget. It takes a list of children and
          // arranges them vertically. By default, it sizes itself to fit its
          // children horizontally, and tries to be as tall as its parent.
          //
          // Invoke "debug painting" (press "p" in the console, choose the
          // "Toggle Debug Paint" action from the Flutter Inspector in Android
          // Studio, or the "Toggle Debug Paint" command in Visual Studio Code)
          // to see the wireframe for each widget.
          //
          // Column has various properties to control how it sizes itself and
          // how it positions its children. Here we use mainAxisAlignment to
          // center the children vertically; the main axis here is the vertical
          // axis because Columns are vertical (the cross axis would be
          // horizontal).
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            const Text(
              'You have pushed the button this many times:',
            ),
            Text(
              '$_counter',
              style: Theme.of(context).textTheme.headline4,
            ),
          ],
        ),
      ),
      floatingActionButton: FloatingActionButton(
        onPressed: _incrementCounter,
        tooltip: 'Increment',
        child: const Icon(Icons.add),
      ), // This trailing comma makes auto-formatting nicer for build methods.
    );
    }
  }
```

5. Simple App " where displaying text in center"



Code :

```
import 'package:flutter/material.dart';
void main() {
  runApp(const MyApp());
}
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Material(
        child: Center(
          child: Container(
            child: Text("My First flutter app"),
          ),
        ),
      ),
    );
  }
}
```

Note : Container shall be inside MaterialApp , or Widget.