Flutter For Android Developers

Michael Yotive

State of Mobile in 2019

- Development is Slow (expensive)
- Custom is Difficult
- Multi-platform complicates things

What is Flutter?

- **Flutter** is an open-source UI toolkit created by Google.
- Flutter allows developers to build high-performance, high-fidelity apps from a single codebase.
- Current platforms are iOS and Android.
- Applications are built using Dart.

Why Dart?

- “Ahead of Time” (AOT) compiled to native binaries (Android NDK and LLVM).
- Can also be “Just in Time” (JIT) compiled for on-the-fly updates.
- Garbage Collection without locking which allows for better performance and smooth animations at 60 fps.

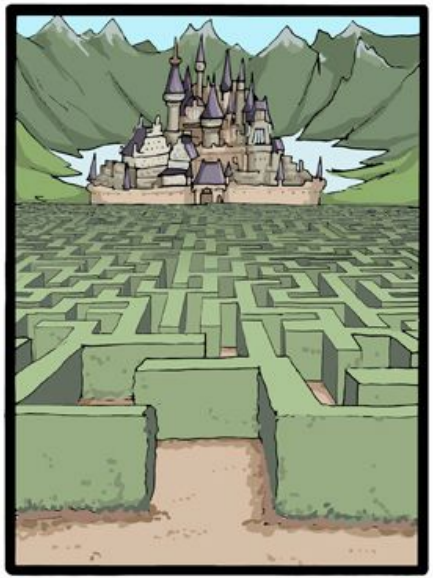
Why Flutter Uses Dart
https://hackernoon.com/why-flutter-uses-dart-dd635a054ebf

What is Flutter?

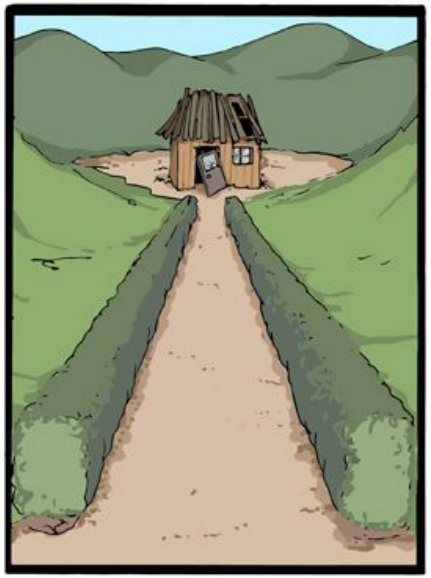
- Ready-made widgets
- Modern “React-like” Framework
- Fast, 2D rendering engine
- Rich support for Development Tools (Android Studio/VS Code)

The dilemma of mobile apps development

Develop a native app for each device and maintain several projects

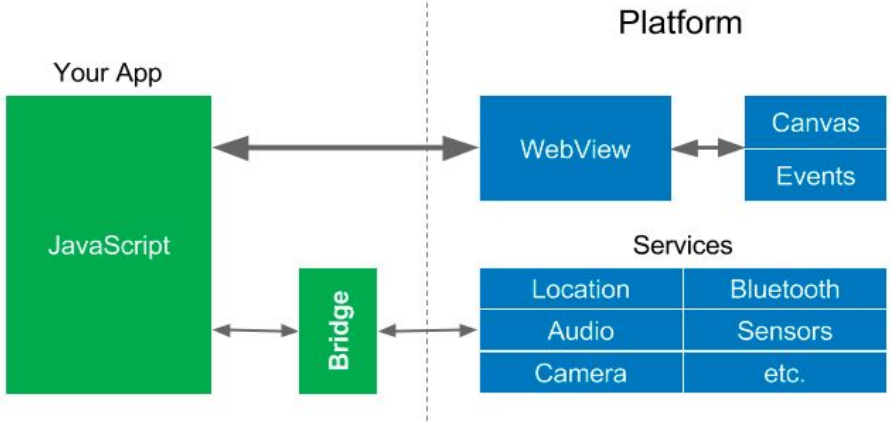


Use a unique framework (Phonegap, Adobe Air, Appcelerator) and maintain only one project

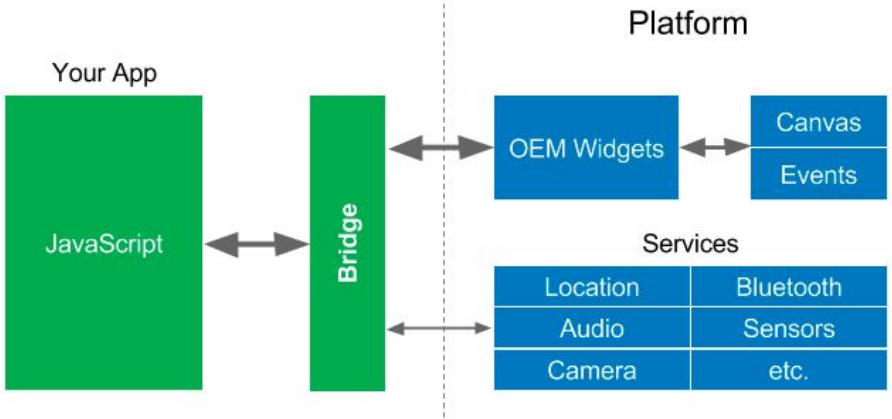


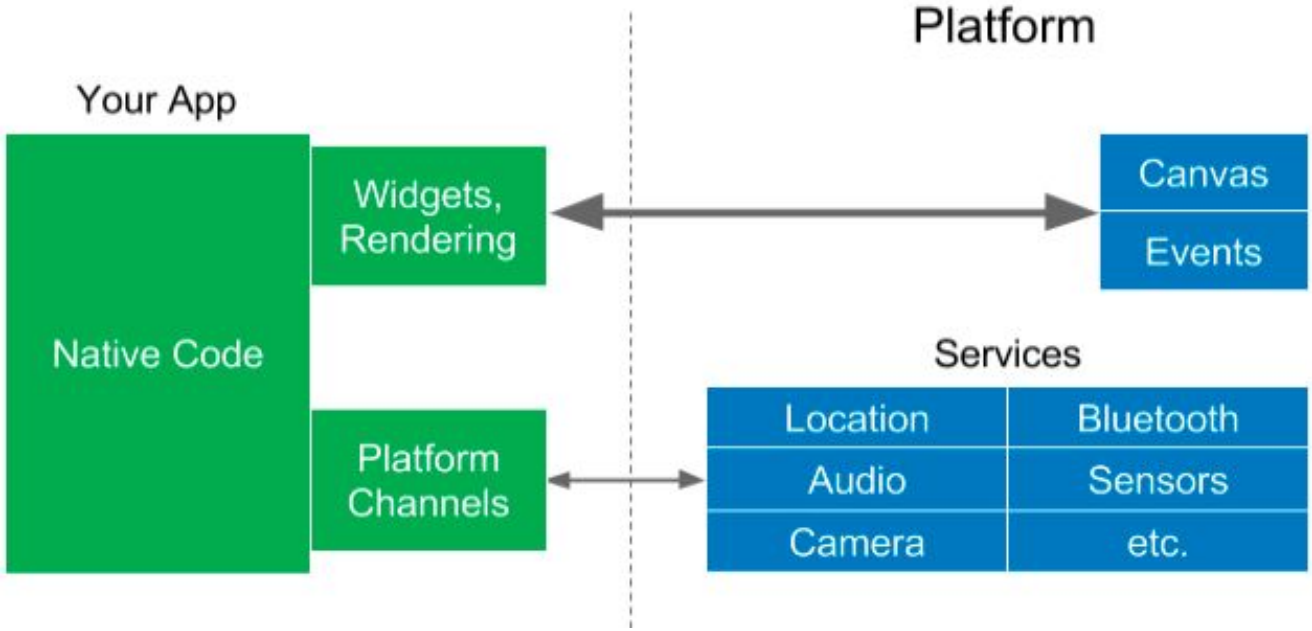
CommitStrip.com

Cordova

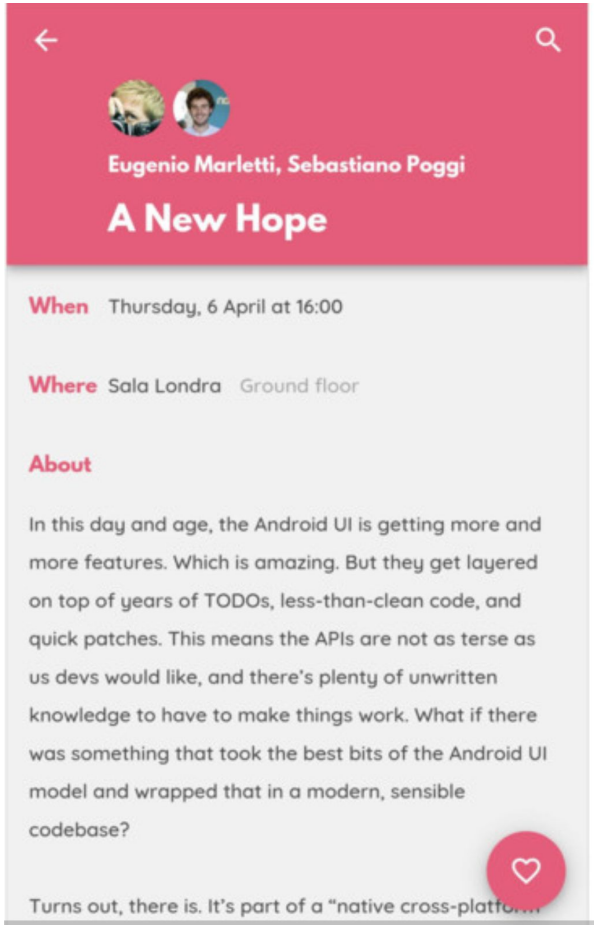


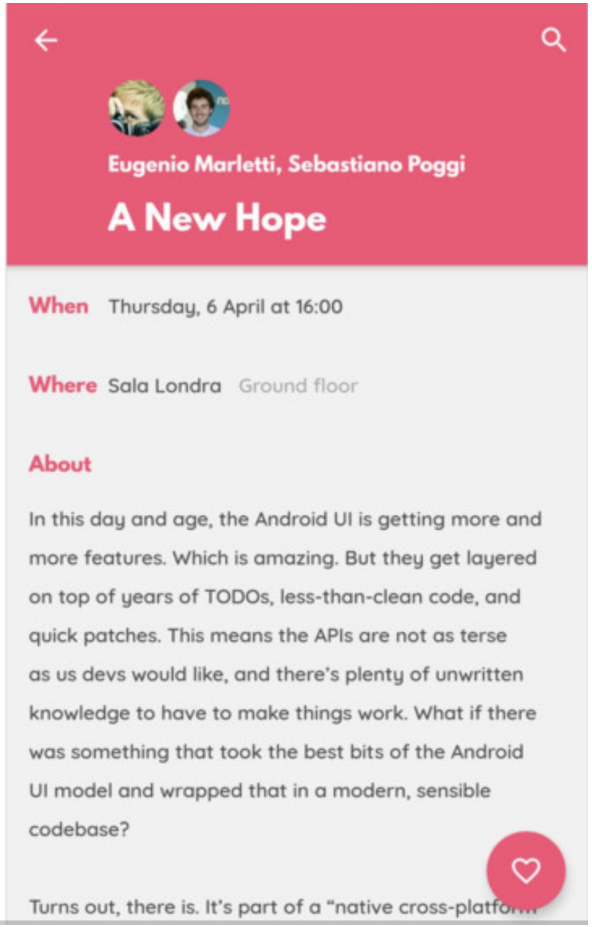
React Native




Platform
Your App
Native Code
Widgets,
Rendering
Platform
Channels
Canvas
Events
Services
Location
Bluetooth
Audio
Sensors
Camera
etc.

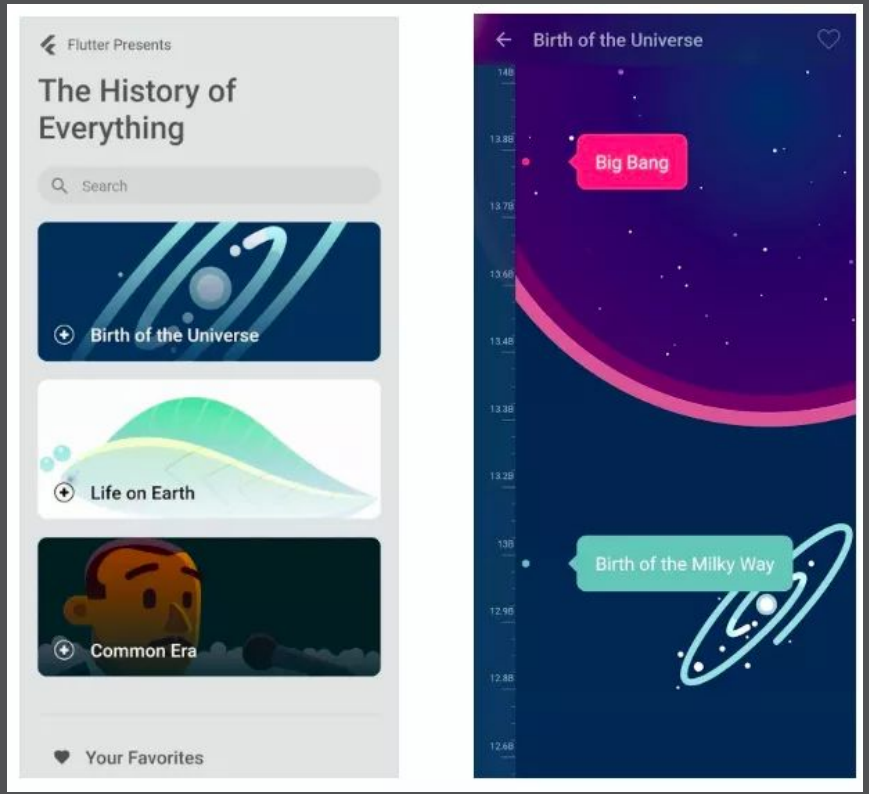

IT FEELS
"NATIVE"
memegenerator.net

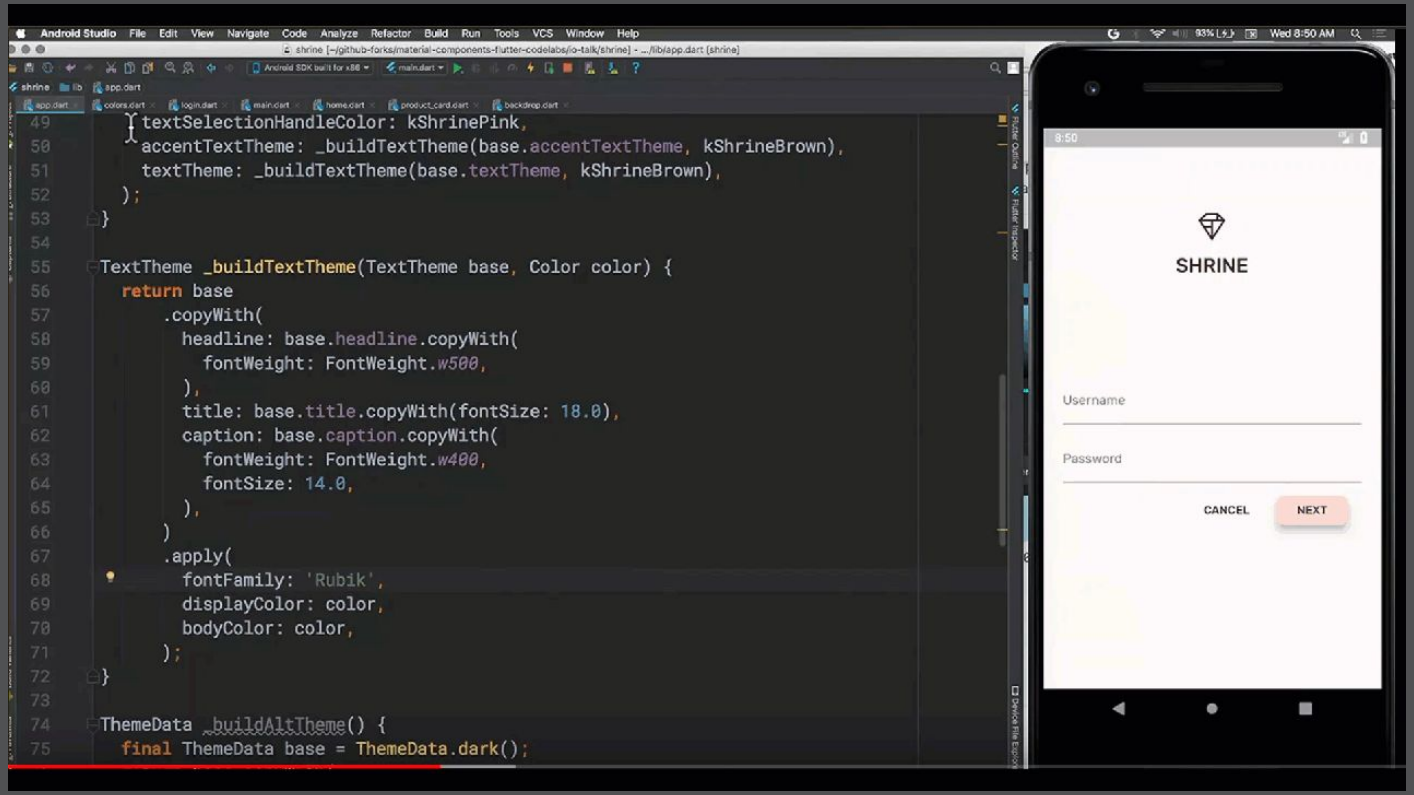




Droidcon Italy 2017 // A new hope - Eugenio Marletti & Sebastiano Poggi
https://www.youtube.com/watch?v=0ijVuVtu6a4

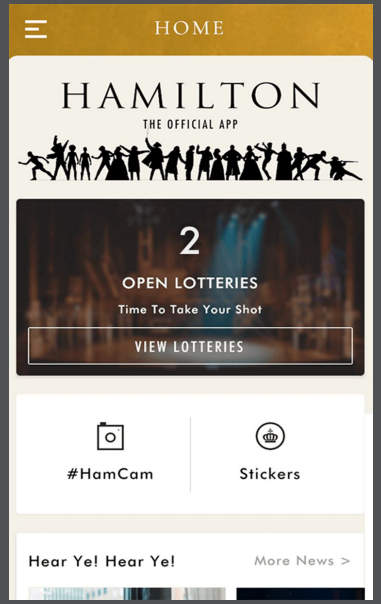
Why Flutter?



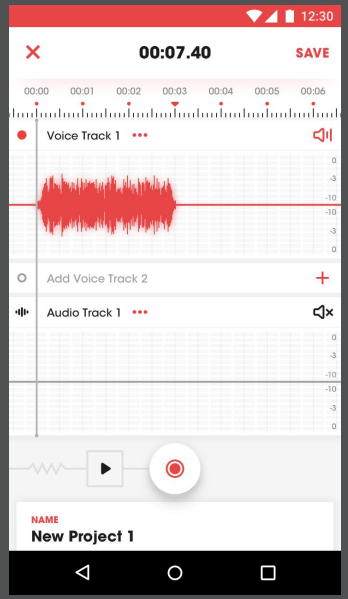


Code beautiful UI with Flutter and Material Design (Google I/O '18)
https://www.youtube.com/watch?v=hA0hrpR-o8U

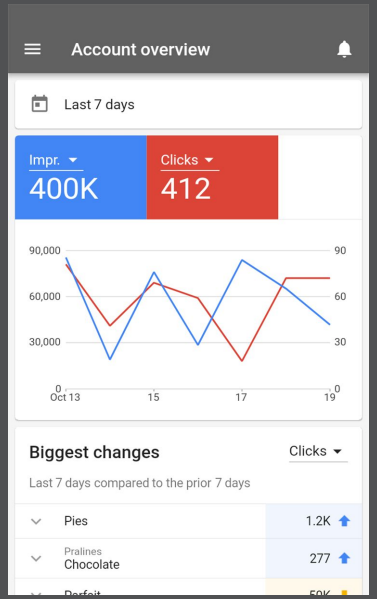
Who's using Flutter?



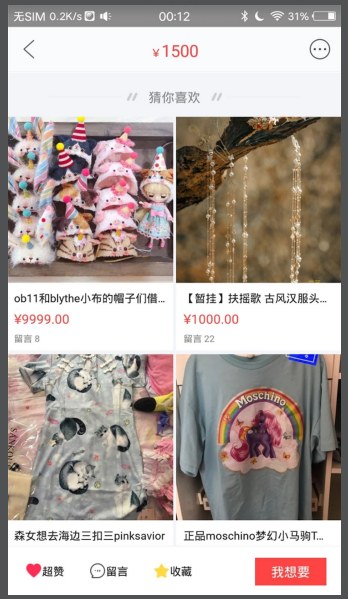
Hamilton



Topline



Google Adwords



Alibaba Xianyu

https://www.itsallwidgets.com

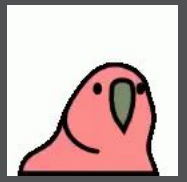

How To Flutter

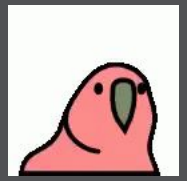
Common Android Tasks

- Activities/Views
- AsyncTask (Handlers, Threads, RxJava, Coroutines)
- Intents
- Services

Activities

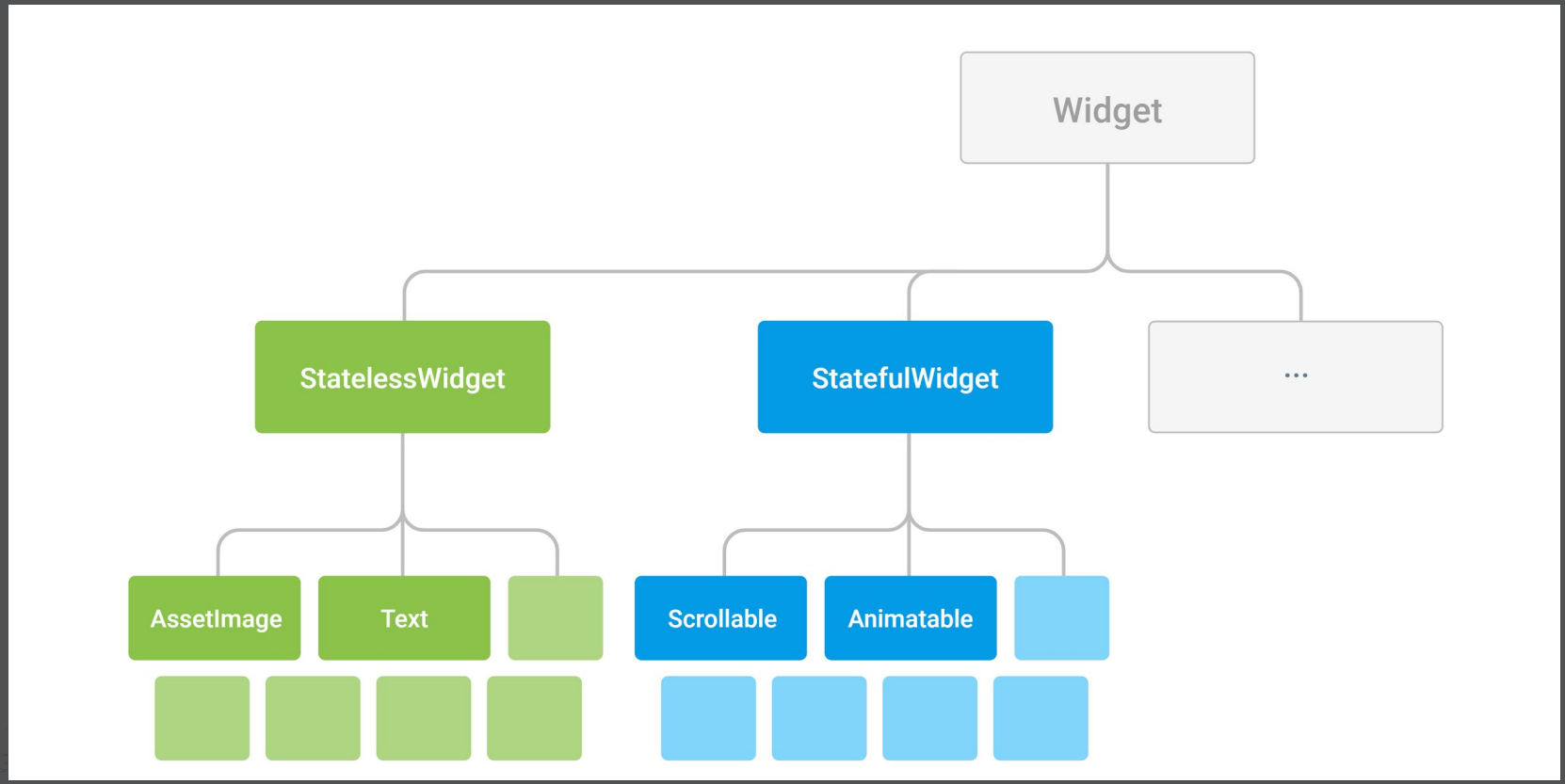
(and views)





It’s all widgets!

Widgets


Widget
StatelessWidget
StatefulWidget
...
AssetImage
Text
Scrollable
Animatable

Stateless Widgets

```
class HelloWorldWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Center(
        child: Text("Hello World", textDirection: TextDirection.ltr,)
    );
  }
}
```

Stateless Widgets

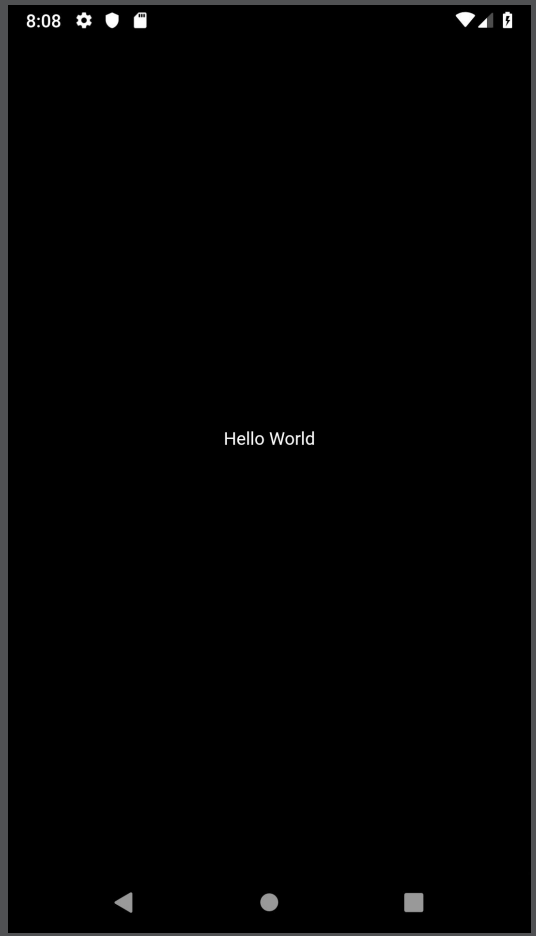
```
class HelloWorldWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Center(
        child: Text("Hello World", textDirection: TextDirection.ltr,)
    );
  }
}
```

Stateless Widgets

```
class HelloWorldWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Center(
        child: Text("Hello World", textDirection: TextDirection.ltr,)
    );
  }
}
```

Stateless Widgets

```
void main() =>
    runApp(HelloWorldWidget());
```



Stateful Widgets

```
class CounterWidget extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    return _CounterWidgetState();
  }
}
```

Stateful Widgets

```
class CounterWidget extends StatefulWidget {
  @override
  State<StatefulWidget> createState() {
    return _CounterWidgetState();
  }
}
```

Stateful Widgets

```
class _CounterWidgetState extends State<CounterWidget> {
  int _counter = 0;
  void _increaseCounter() { … }
  @override
  Widget build(BuildContext context) { … }
}
```

Stateful Widgets

```
class _CounterWidgetState extends State<CounterWidget> {
  int _counter = 0;
  void _increaseCounter() {
    setState(() {
      _counter = _counter + 1;
    });
  }
  @override
  Widget build(BuildContext context) { … }
}
```

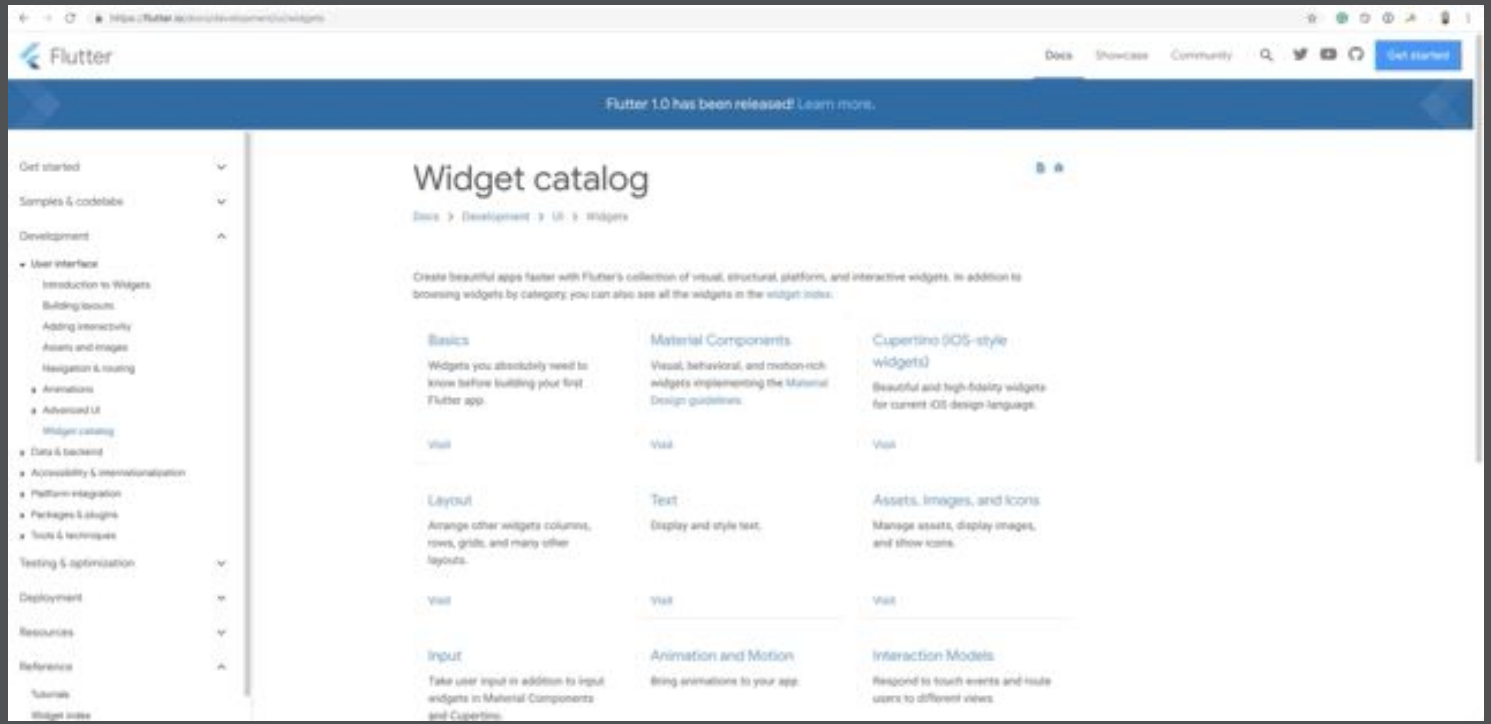
Stateful Widgets

```
class _CounterWidgetState extends State<CounterWidget> {
  void _increaseCounter() { ... }
  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      textDirection: TextDirection.ltr,
      children: <Widget>[
        Text(_counter.toString()),
        RaisedButton(onPressed: _increaseCounter,
                     child: Text("Increment")),
      ],
    );
  }
}
```

Stateful Widgets

```
class _CounterWidgetState extends State<CounterWidget> {
  void _increaseCounter() { ... }
  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      textDirection: TextDirection.ltr,
      children: <Widget>[
        Text(_counter.toString()),
        RaisedButton(onPressed: _increaseCounter,
                     child: Text("Increment")),
      ],
    );
  }
}
```

Widget Catalog



https://flutter.io/docs/development/ui/widgets/catalog

Stateful Widget Lifecycle

How do we do X in Flutter?

Linear Layout

Row / Column

```
class SampleWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
      color: Colors.yellow,
      child: Row(
        children: <Widget>[
          Icon(Icons.star, size: 50),
          Icon(Icons.star, size: 100),
          Icon(Icons.star, size: 50)
        ],
      ),
    );
  }
}
```

Row / Column

```
class SampleWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
      color: Colors.yellow,
      child: Row(
        children: <Widget>[
          Icon(Icons.star, size: 50),
          Icon(Icons.star, size: 100),
          Icon(Icons.star, size: 50)
        ],
      ),
    );
  }
}
```

Row / Column

```
class SampleWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
      color: Colors.yellow,
      child: Row(
        children: <Widget>[
          Icon(Icons.star, size: 50),
          Icon(Icons.star, size: 100),
          Icon(Icons.star, size: 50)
        ],
      ),
    );
  }
}
```

Row / Column

```
class SampleWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
      color: Colors.yellow,
      child: Row(
        children: <Widget>[
          Icon(Icons.star, size: 50),
          Icon(Icons.star, size: 100),
          Icon(Icons.star, size: 50)
        ],
      ),
    );
  }
}
```

Row / Column

```
class SampleWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
      color: Colors.yellow,
      child: Column(
        children: <Widget>[
          Icon(Icons.star, size: 50),
          Icon(Icons.star, size: 100),
          Icon(Icons.star, size: 50)
        ],
      ),
    );
  }
}
```

Row / Column

```
class SampleWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
      color: Colors.yellow,
      child: Row(
        children: <Widget>[
          Icon(Icons.star, size: 50),
          Icon(Icons.star, size: 100),
          Icon(Icons.star, size: 50)
        ],
      ),
    );
  }
}
```

Row / Column

```
class SampleWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container(
      color: Colors.yellow,
      child: Row(
        mainAxisSize: MainAxisSize.min,
        children: <Widget>[
          Icon(Icons.star, size: 50),
          Icon(Icons.star, size: 100),
          Icon(Icons.star, size: 50)
        ],
      ),
    );
  }
}
```

Row / Column

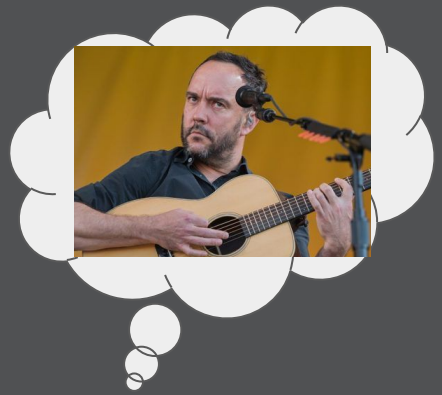
```
class SampleWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
      ...
      child: Row(
        mainAxisSize: MainAxisSize.max,
        mainAxisAlignment: MainAxisAlignment.end,
        children: <Widget>[
          Icon(Icons.star, size: 50),
          Icon(Icons.star, size: 100),
          Icon(Icons.star, size: 50)
        ],
      ),
      ...
  }
}
```

Row / Column

```
class SampleWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
      ...
      child: Row(
        mainAxisSize: MainAxisSize.max,
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          Icon(Icons.star, size: 50),
          Icon(Icons.star, size: 100),
          Icon(Icons.star, size: 50)
        ],
      ),
      ...
  }
}
```

Row / Column

```
class SampleWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
      ...
      child: Row(
        mainAxisSize: MainAxisSize.max,
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: <Widget>[
          Icon(Icons.star, size: 50),
          Icon(Icons.star, size: 100),
          Icon(Icons.star, size: 50)
        ],
      ),
      ...
  }
}
```



Row / Column

```
class SampleWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
      ...
      child: Row(
        mainAxisSize: MainAxisSize.max,
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: <Widget>[
          Icon(Icons.star, size: 50),
          Icon(Icons.star, size: 100),
          Icon(Icons.star, size: 50)
        ],
      ),
      ...
  }
}
```

Row / Column

```
class SampleWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
      ...
      child: Row(
        mainAxisSize: MainAxisSize.max,
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        crossAxisAlignment: CrossAxisAlignment.end,
        children: <Widget>[
          Icon(Icons.star, size: 50),
          Icon(Icons.star, size: 100),
          Icon(Icons.star, size: 50)
        ],
      ),
      ...
  }
}
```

Row / Column

```
class SampleWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
      ...
      child: Row(
        mainAxisSize: MainAxisSize.max,
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        crossAxisAlignment: CrossAxisAlignment.stretch,
        children: <Widget>[
          Icon(Icons.star, size: 50),
          Icon(Icons.star, size: 100),
          Icon(Icons.star, size: 50)
        ],
      ),
      ...
  }
}
```

Row / Column

```
class SampleWidget extends StatelessWidget {
...
      child: Row(
        children: <Widget>[
          Expanded(
              flex: 1,
              child: Container(
                  color: Colors.blue, child: Icon(Icons.star, size: 50)))
          Expanded(
              flex: 1,
              child: Container(
                  color: Colors.green, child: Icon(Icons.star, size: 100)))
        ]),
...
}
```

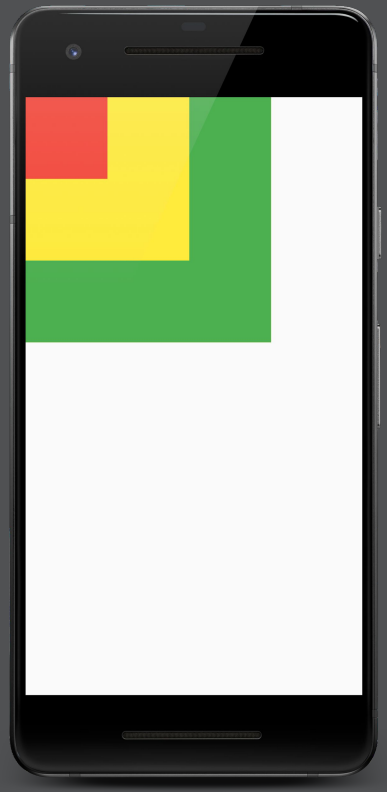
Row / Column

```
class SampleWidget extends StatelessWidget {
...
      child: Row(
        children: <Widget>[
          Expanded(
              flex: 1,
              child: Container(
                  color: Colors.blue, child: Icon(Icons.star, size: 50)))
          Expanded(
              flex: 3,
              child: Container(
                  color: Colors.green, child: Icon(Icons.star, size: 100)))
        ]),
...
}
```

Frame Layout

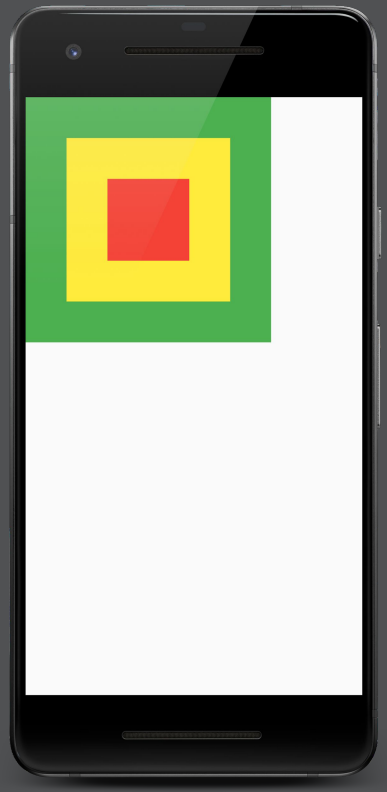
Stack

```
class SampleWidget extends StatelessWidget {
      ...
      child: Stack(
        children: <Widget>[
          SizedBox(
            height: 300, width: 300, child: Container(
                color: Colors.green)),
          SizedBox(
            height: 200, width: 200, child: Container(
                color: Colors.yellow)),
          SizedBox(
            height: 100, width: 100, child: Container(
                color: Colors.red)),
        ],
      ...
}
```



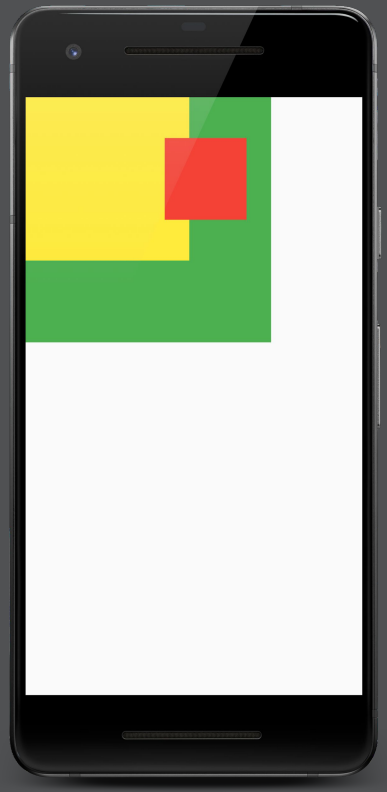
Stack

```
class SampleWidget extends StatelessWidget {
      ...
      child: Stack(
        alignment: Alignment.center,
        children: <Widget>[
          SizedBox(
            height: 300, width: 300, child: Container(
                color: Colors.green)),
          SizedBox(
            height: 200, width: 200, child: Container(
                color: Colors.yellow)),
          SizedBox(
            height: 100, width: 100, child: Container(
                color: Colors.red)),
        ],
      ...
}
```



Stack

```
class SampleWidget extends StatelessWidget {
      ...
      child: Stack(
        children: <Widget>[
          SizedBox(
            height: 300, width: 300, child: Container(
                color: Colors.green)),
          SizedBox(
            height: 200, width: 200, child: Container(
                color: Colors.yellow)),
          Positioned( top: 50, left: 170,
            child: SizedBox( height: 100, width: 100,
                child: Container(color: Colors.red)),
          )],
      ...
}
```

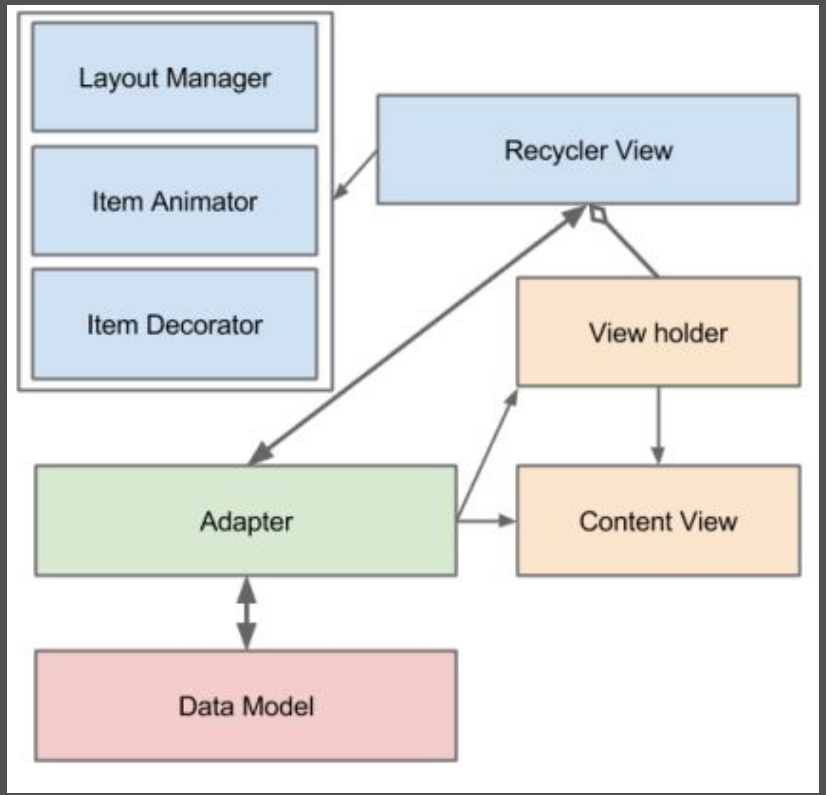


ImageView

ImageView

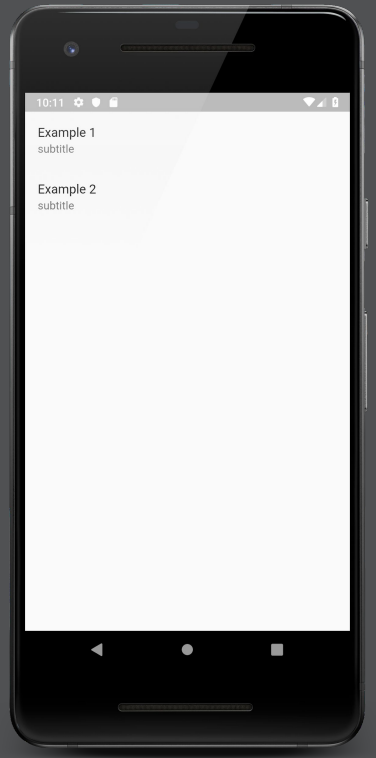
RecyclerView

Recycler View



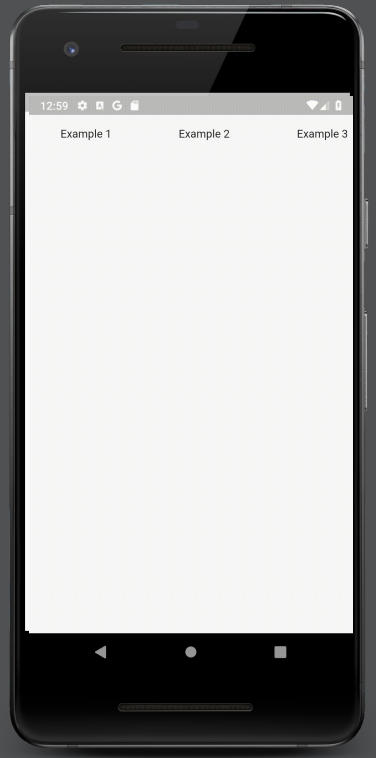
ListView

```
ListView(
  children: <Widget>[
    ListTile(title: Text("Example 1"), subtitle: Text("subtitle")),
    ListTile(title: Text("Example 2"), subtitle: Text("subtitle"))
  ],
);
```



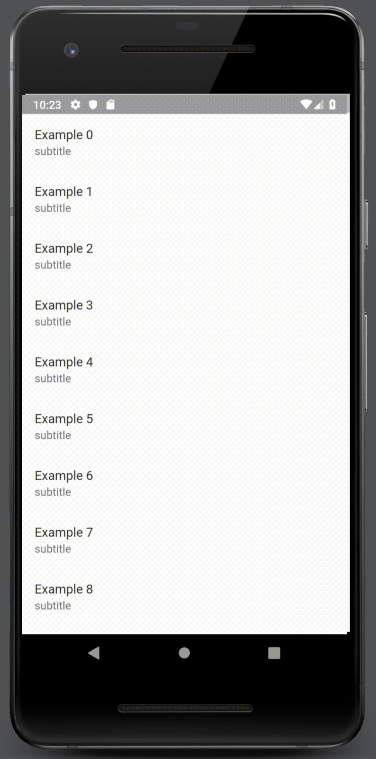
ListView

```
ListView(
      scrollDirection: Axis.horizontal,
      padding: EdgeInsets.all(40),
      children: <Widget>[
        Container(width: 150, child: Text("Example 1")),
        Container(width: 150, child: Text("Example 2")),
        Container(width: 150, child: Text("Example 3")),
        Container(width: 150, child: Text("Example 4")),
        Container(width: 150, child: Text("Example 5"))
      ],
    );
```



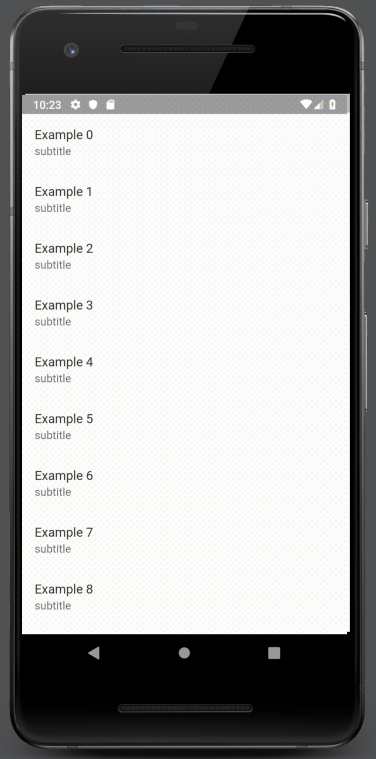
ListView

```
return ListView.builder(
        itemCount: items.length,
        itemBuilder: (context, index) {
          return ListTile(
               title: Text("Example $index"),
               subtitle: Text("subtitle"));
        });
```



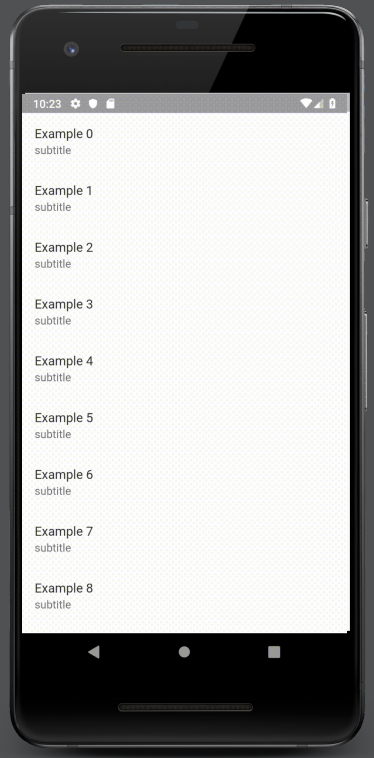
ListView

```
return ListView.builder(
        controller: _scrollController,
        itemCount: items.length,
        itemBuilder: (context, index) {
          return ListTile(
               title: Text("Example $index"),
               subtitle: Text("subtitle"));
        });
```



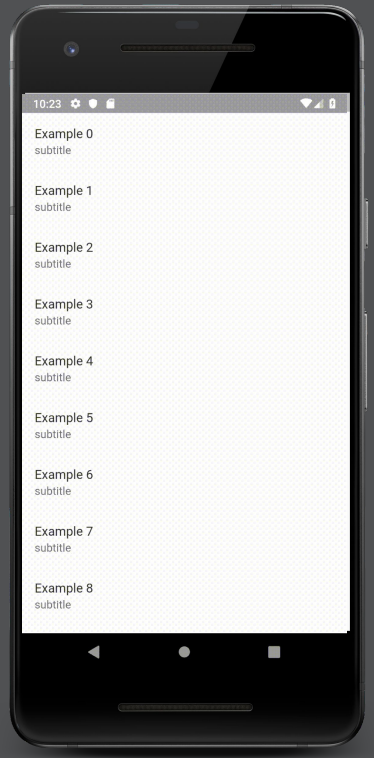
ListView

```
ScrollController _scrollController = new ScrollController();

 @override
  void initState() {
    super.initState();
    _scrollController.addListener(() {
      if (_scrollController.position.pixels ==
          _scrollController.position.maxScrollExtent) {
        _getMoreData();
      }
    });
  }
```



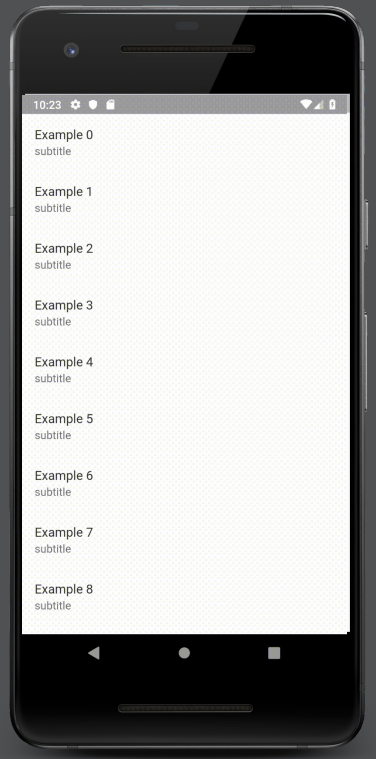
ListView

```
ScrollController _scrollController = new ScrollController();

 @override
  void initState() {
    super.initState();
    _scrollController.addListener(() {
      if (_scrollController.position.pixels ==
          _scrollController.position.maxScrollExtent) {
        _getMoreData();
      }
    });
  }
```



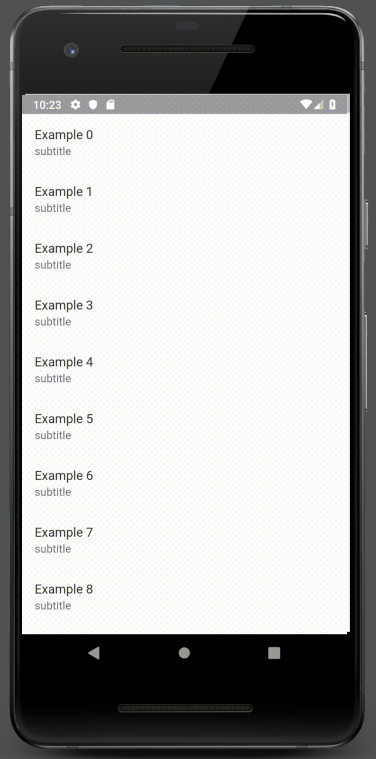
ListView

```
_getMoreData() {
    List<int> newEntries = List.generate(10, (i) => i + 10);
    setState(() {
      items.addAll(newEntries);
    });
  }
```



ListView

```
_getMoreData() {
    List<int> newEntries = List.generate(10, (i) => i + 10);
    setState(() {
      items.addAll(newEntries);
    });
  }
```



Coordinator Layout

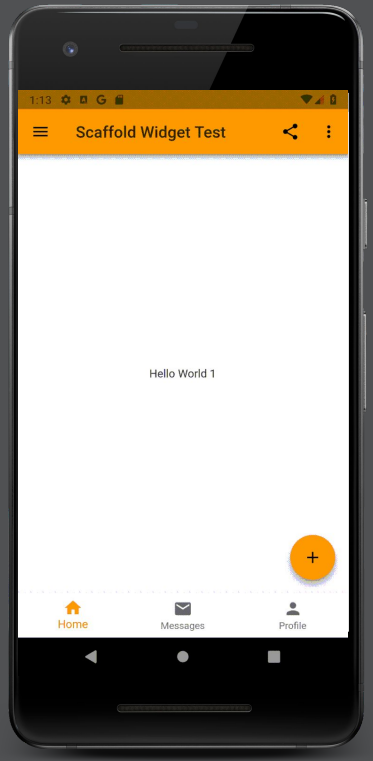
Coordinator Layout

Toolbar

(and Menu, Drawer, Bottom Nav, FAB)

Toolbar (and Menu, Drawer, Bottom Nav, FAB)

```
class ScaffoldExampleWidgetState
    extends State<ScaffoldExampleWidget> {
  ...
  @override
  Widget build(BuildContext context) {
    return Scaffold( ... ));
  }
}
```

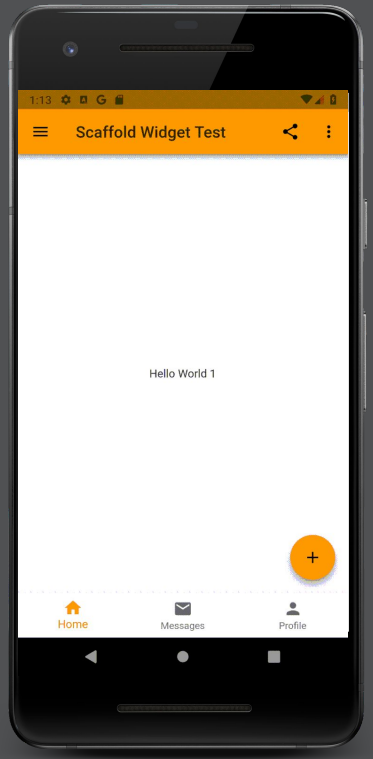


Toolbar (and Menu, Drawer, Bottom Nav, FAB)

```
Scaffold(
    floatingActionButton:
        FloatingActionButton(child: Icon(Icons.add),
                onPressed: () {}),
    appBar: AppBar(
        title: Text("Scaffold Widget Test"),
        actions: <Widget>[...],
    ),
    drawer: Drawer(... ),
    bottomNavigationBar: BottomNavigationBar(...),
    body: _children[_currentIndex],
);
```

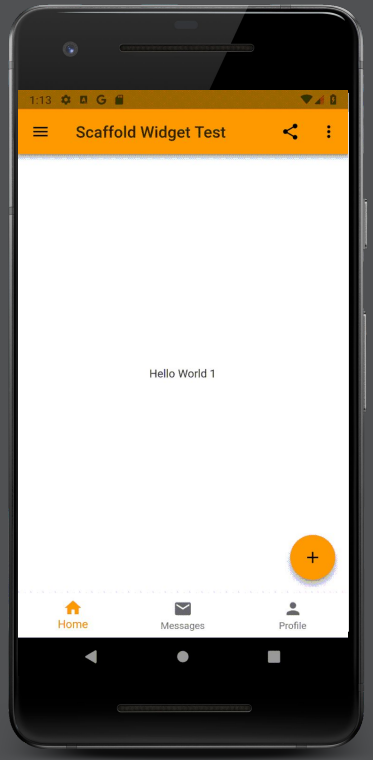
Toolbar (and Menu, Drawer, Bottom Nav, FAB)

```
final List<Widget> _children = [
   Center(child: Text("Hello World 1")),
   Center(child: Text("Hello World 2")),
   Center(child: Text("Hello World 3"))
 ];

void onTabTapped(int index) {
    setState(() { _currentIndex = index; });
 }
bottomNavigationBar: BottomNavigationBar(
          onTap: onTabTapped,
          currentIndex: _currentIndex,
          items: [ BottomNavigationBarItem(...)])),
body: _children[_currentIndex],
```



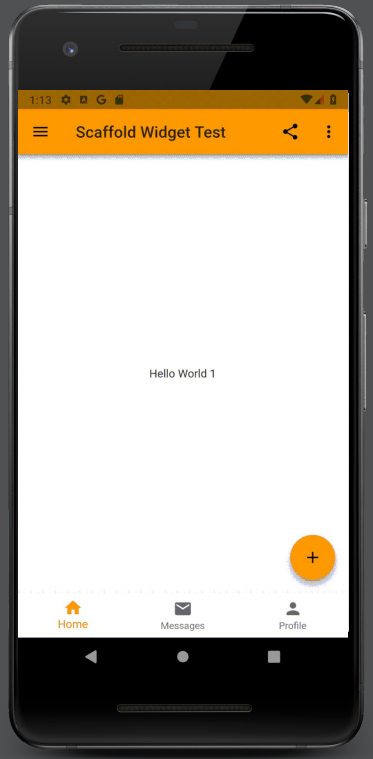
Toolbar (and Menu, Drawer, Bottom Nav, FAB)

```
final List<Widget> _children = [
   Center(child: Text("Hello World 1")),
   Center(child: Text("Hello World 2")),
   Center(child: Text("Hello World 3"))
 ];

void onTabTapped(int index) {
    setState(() { _currentIndex = index; });
 }
bottomNavigationBar: BottomNavigationBar(
          onTap: onTabTapped,
          currentIndex: _currentIndex,
          items: [ BottomNavigationBarItem(...)])),
body: _children[_currentIndex],
```



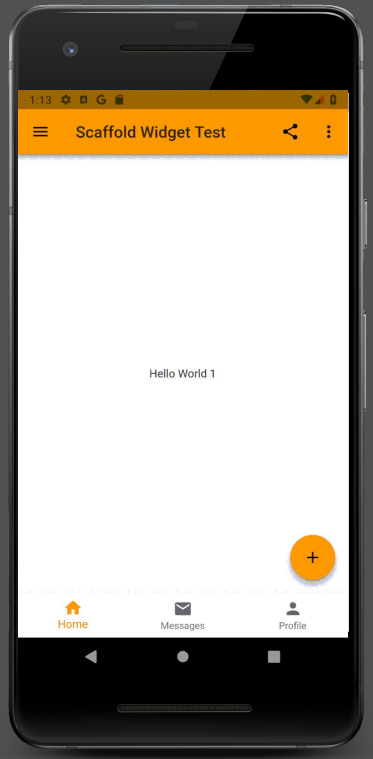
Toolbar (and Menu, Drawer, Bottom Nav, FAB)

```
final List<Widget> _children = [
   Center(child: Text("Hello World 1")),
   Center(child: Text("Hello World 2")),
   Center(child: Text("Hello World 3"))
 ];

void onTabTapped(int index) {
    setState(() { _currentIndex = index; });
 }
bottomNavigationBar: BottomNavigationBar(
          onTap: onTabTapped,
          currentIndex: _currentIndex,
          items: [ BottomNavigationBarItem(...)])),
body: _children[_currentIndex],
```



Toolbar (and Menu, Drawer, Bottom Nav, FAB)

```
final List<Widget> _children = [
   Center(child: Text("Hello World 1")),
   Center(child: Text("Hello World 2")),
   Center(child: Text("Hello World 3"))
 ];

void onTabTapped(int index) {
    setState(() { _currentIndex = index; });
 }
bottomNavigationBar: BottomNavigationBar(
          onTap: onTabTapped,
          currentIndex: _currentIndex,
          items: [ BottomNavigationBarItem(...)])),
body: _children[_currentIndex],
```



ViewPager

Hero Image

Maps

Maps

pubspec.yaml

```
dependencies:
  google_maps_flutter: ^0.0.3+3
```

Maps

Android > app > src > main > AndroidManifest.xml

```
<meta-data android:name="com.google.android.geo.API_KEY"
           android:value="YOUR API KEY HERE"/>
```

Maps

```
class MapsState extends State<SampleMapsWidget> {

  GoogleMapController mapController;

  void _onMapCreated(GoogleMapController controller) {
      mapController = controller;
  }

  @override
  Widget build(BuildContext context) {
    return GoogleMap(
      onMapCreated: _onMapCreated,
    );
  }
}
```

Maps

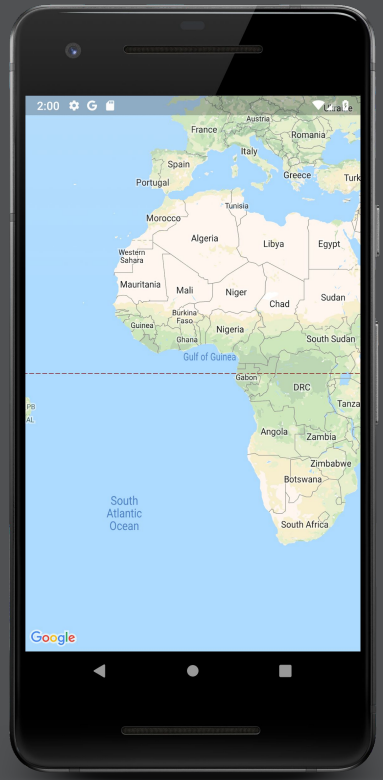
```
class MapsState extends State<SampleMapsWidget> {

  GoogleMapController mapController;

  void _onMapCreated(GoogleMapController controller) {
      mapController = controller;
  }

  @override
  Widget build(BuildContext context) {
    return GoogleMap(
      onMapCreated: _onMapCreated,
    );
  }
}
```

Maps

```
class MapsState extends State<SampleMapsWidget> {

  GoogleMapController mapController;

  void _onMapCreated(GoogleMapController controller) {
      mapController = controller;
  }

  @override
  Widget build(BuildContext context) {
    return GoogleMap(
      onMapCreated: _onMapCreated,
    );
  }
}
```

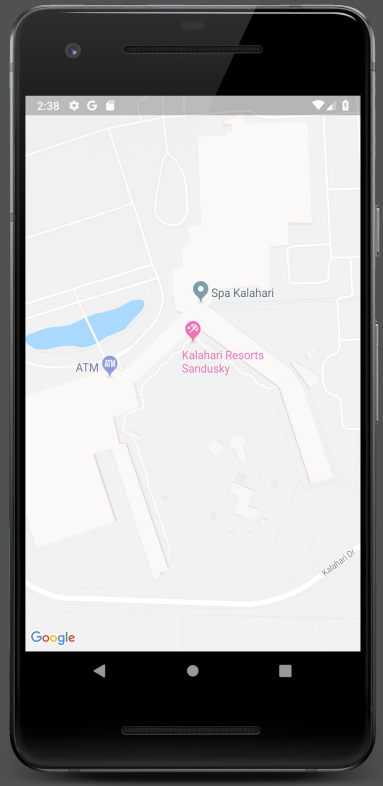
Maps

```
class MapsState extends State<SampleMapsWidget> {

  GoogleMapController mapController;

  void _onMapCreated(GoogleMapController controller) {
      mapController = controller;
  }

  @override
  Widget build(BuildContext context) {
    return GoogleMap(
      onMapCreated: _onMapCreated,
    );
  }
}
```



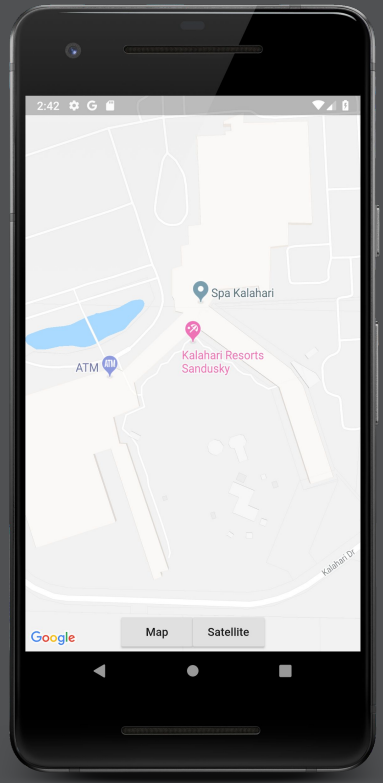
Maps

```
GoogleMap(options: GoogleMapOptions(
                cameraPosition: CameraPosition(
                     target: LatLng(41.383357, -82.641631),
                     zoom: 17.0)),
           onMapCreated: _onMapCreated,
         )
```



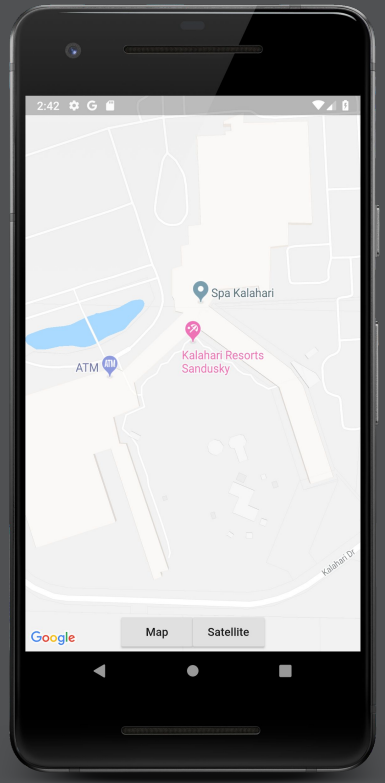
Maps

```
Stack(
      children: <Widget>[
        GoogleMap( ... ),
        Align(
          alignment: Alignment.bottomCenter,
          child: Row(
            mainAxisSize: MainAxisSize.min,
            children: <Widget>[
              RaisedButton(child: Text("Map")),
              RaisedButton(child: Text("Satellite"))
            ],
          ),
        )
      ],
    );
```



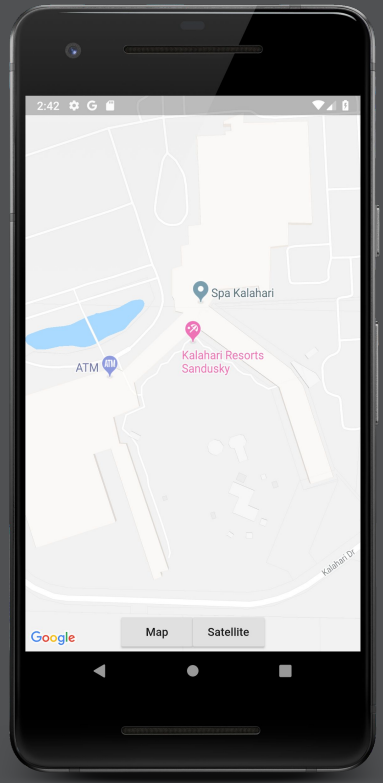
Maps

```
Stack(
      children: <Widget>[
        GoogleMap( ... ),
        Align(
          alignment: Alignment.bottomCenter,
          child: Row(
            mainAxisSize: MainAxisSize.min,
            children: <Widget>[
              RaisedButton(child: Text("Map")),
              RaisedButton(child: Text("Satellite"))
            ],
          ),
        )
      ],
    );
```



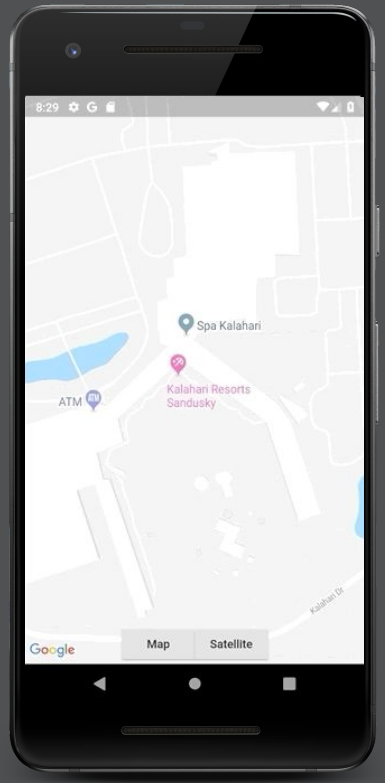
Maps

```
RaisedButton(
    child: Text("Map"),
    onPressed: (){
      mapController.updateMapOptions(GoogleMapOptions(
          mapType: MapType.normal
      ));
    })
  RaisedButton(
    child: Text("Satellite"),
    onPressed: (){
    mapController.updateMapOptions(GoogleMapOptions(
        mapType: MapType.satellite
    ));
  })
```



Maps

```
RaisedButton(
    child: Text("Map"),
    onPressed: (){
      mapController.updateMapOptions(GoogleMapOptions(
          mapType: MapType.normal
      ));
    })
  RaisedButton(
    child: Text("Satellite"),
    onPressed: (){
    mapController.updateMapOptions(GoogleMapOptions(
        mapType: MapType.satellite
    ));
  })
```



Activity Lifecycle

Flutter Stateful Widget Lifecycle

```
class MyLifecycleWatcherState extends State<SampleWidget> with WidgetsBindingObserver {
  AppLifecycleState _lastLifecycleState;

  @override
  void initState() {
    super.initState();
    WidgetsBinding.instance.addObserver(this);
  }

  @override
  void dispose() {
    WidgetsBinding.instance.removeObserver(this);
    super.dispose();
  }
  …
}
```

Flutter Stateful Widget Lifecycle

```
class MyLifecycleWatcherState extends State<SampleWidget> with WidgetsBindingObserver {
  AppLifecycleState _lastLifecycleState;

  @override
  void initState() {
    super.initState();
    WidgetsBinding.instance.addObserver(this);
  }

  @override
  void dispose() {
    WidgetsBinding.instance.removeObserver(this);
    super.dispose();
  }
  ...
}
```

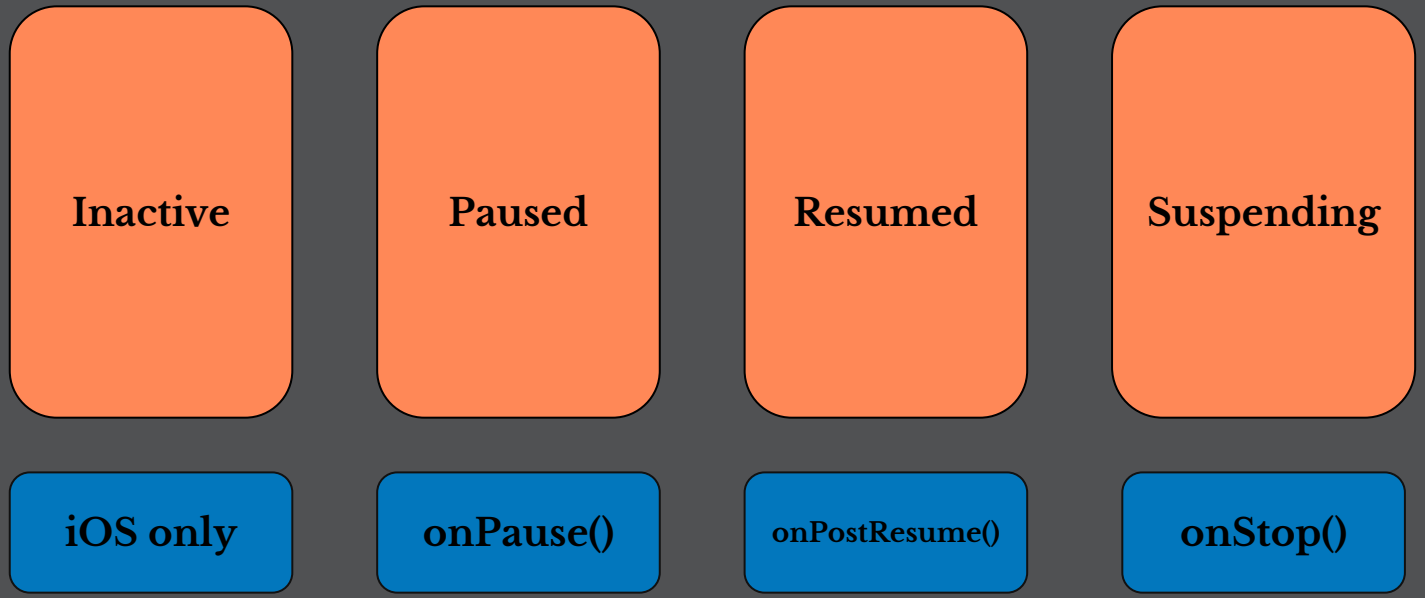
Flutter Stateful Widget Lifecycle

```
class MyLifecycleWatcherState extends State<SampleWidget> with WidgetsBindingObserver {
  AppLifecycleState _lastLifecycleState;

  @override
  void initState() {
    super.initState();
    WidgetsBinding.instance.addObserver(this);
  }

  @override
  void dispose() {
    WidgetsBinding.instance.removeObserver(this);
    super.dispose();
  }
  ...
}
```

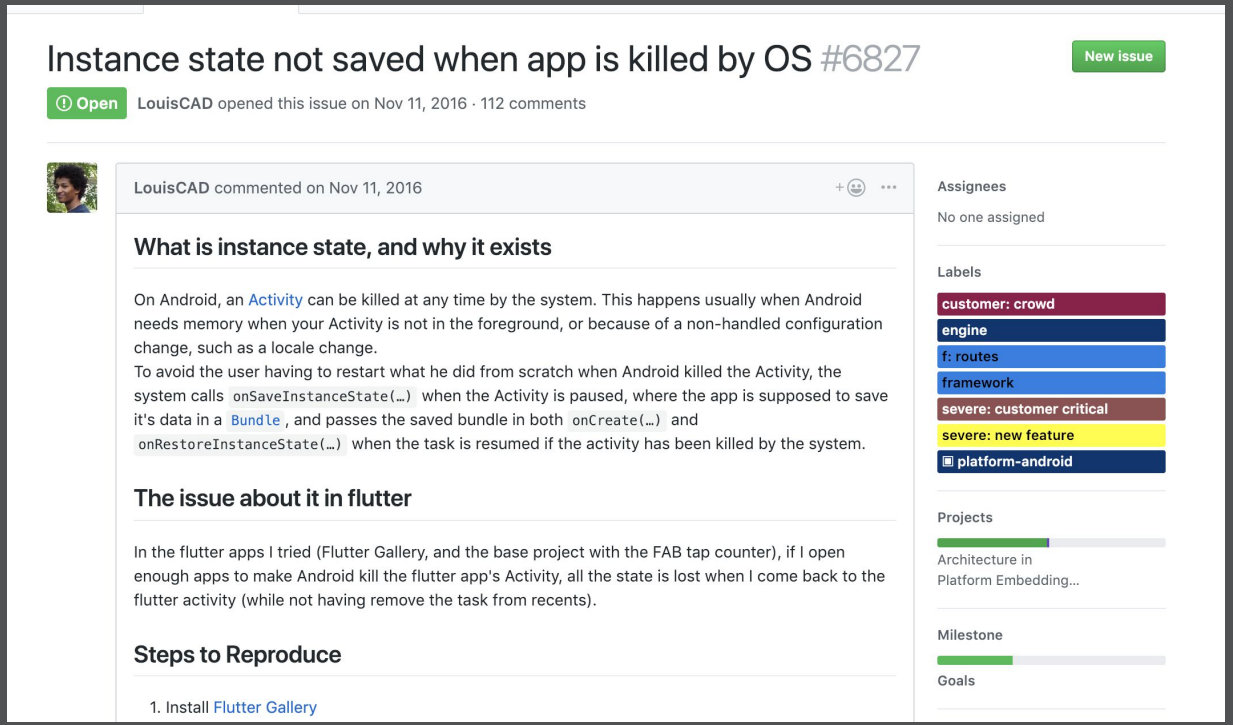
Flutter Stateful Widget Lifecycle

```
class MyLifecycleWatcherState extends State<SampleWidget> with WidgetsBindingObserver {
  AppLifecycleState _lastLifecycleState;
  …
  @override
  void didChangeAppLifecycleState(AppLifecycleState state) {
    setState(() {
      _lastLifecycleState = state;
    });
  }
}
```

Flutter Stateful Widget Lifecycle



Persisting State



Instance state not saved when app is killed by OS #6827

New issue

Open LouisCAD opened this issue on Nov 11, 2016 · 112 comments

LouisCAD commented on Nov 11, 2016

What is instance state, and why it exists

On Android, an Activity can be killed at any time by the system. This happens usually when Android needs memory when your Activity is not in the foreground, or because of a non-handled configuration change, such as a locale change.
To avoid the user having to restart what he did from scratch when Android killed the Activity, the system calls `onSaveInstanceState(…)` when the Activity is paused, where the app is supposed to save it's data in a `Bundle`, and passes the saved bundle in both `onCreate(…)` and `onRestoreInstanceState(…)` when the task is resumed if the activity has been killed by the system.

The issue about it in flutter

In the flutter apps I tried (Flutter Gallery, and the base project with the FAB tap counter), if I open enough apps to make Android kill the flutter app's Activity, all the state is lost when I come back to the flutter activity (while not having remove the task from recents).

Steps to Reproduce

1. Install Flutter Gallery

Assignees

No one assigned

Labels

customer: crowd
engine
f: routes
framework
severe: customer critical
severe: new feature
platform-android

Projects

Architecture in Platform Embedding...

Milestone

Goals

https://github.com/flutter/flutter/issues/6827

Orientation Change

Orientation Change

```
class OrientationChangeExampleWidgetState extends State<OrientationChangeExampleWidget> {
  int _counter = 0;
  _incrementCounter(){ ... }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      floatingActionButton: FloatingActionButton(
          child: Icon(Icons.add),
          onPressed: () => _incrementCounter()
      ),
      body: OrientationBuilder(
        builder: (context, orientation){
          return orientation == Orientation.portrait ? buildPortraitWidget() : buildLandscapeWidget();
        },
      ),
    );
  }
 ...
}
```

Orientation Change

```
class OrientationChangeExampleWidgetState extends State<OrientationChangeExampleWidget> {
  int _counter = 0;
  _incrementCounter(){ ... }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      floatingActionButton: FloatingActionButton(
          child: Icon(Icons.add),
          onPressed: () => _incrementCounter()
      ),
      body: OrientationBuilder(
        builder: (context, orientation){
          return orientation == Orientation.portrait ? buildPortraitWidget() : buildLandscapeWidget();
        },
      ),
    );
  }
 ...
}
```

Orientation Change

```
class OrientationChangeExampleWidgetState extends State<OrientationChangeExampleWidget> {
  int _counter = 0;
  _incrementCounter(){ ... }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      floatingActionButton: FloatingActionButton(
          child: Icon(Icons.add),
          onPressed: () => _incrementCounter()
      ),
      body: OrientationBuilder(
        builder: (context, orientation){
          return orientation == Orientation.portrait ? buildPortraitWidget() : buildLandscapeWidget();
        },
      ),
    );
  }
 ...
}
```

Orientation Change

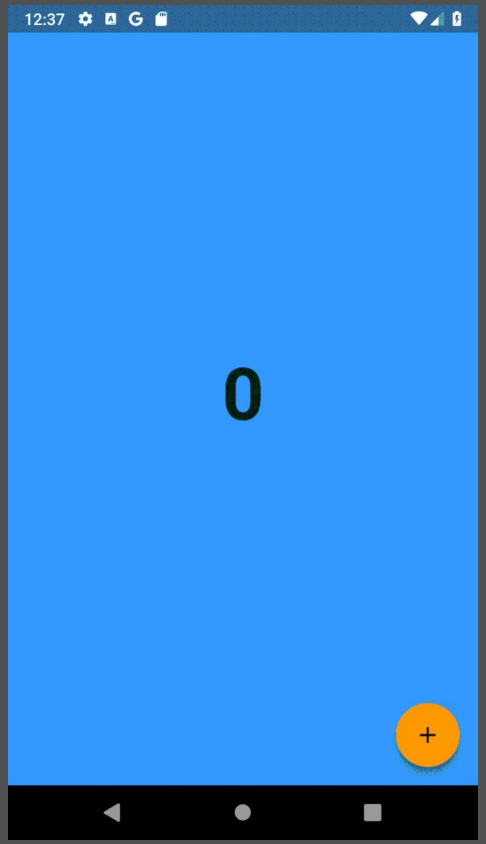
```
buildPortraitWidget() {
    return Container(
      color: Colors.blue,
      alignment: Alignment.center,
      child: Text(_counter.toString(),
        style: TextStyle(fontSize: 64, fontWeight: FontWeight.bold)),
    );
  }

  buildLandscapeWidget() {
    return Container(
      color: Colors.green,
      alignment: Alignment.center,
      child: Text(_counter.toString(),
          style: TextStyle(fontSize: 64, fontWeight: FontWeight.bold)),
    );
  }
```

Orientation Change

```
buildPortraitWidget() {
    return Container(
      color: Colors.blue,
      alignment: Alignment.center,
      child: Text(_counter.toString(),
        style: TextStyle(fontSize: 64, fontWeight: FontWeight.bold)),
    );
  }

  buildLandscapeWidget() {
    return Container(
      color: Colors.green,
      alignment: Alignment.center,
      child: Text(_counter.toString(),
          style: TextStyle(fontSize: 64, fontWeight: FontWeight.bold)),
    );
  }
```

Orientation Change

```
buildPortraitWidget() {
    return Container(
      color: Colors.blue,
      alignment: Alignment.center,
      child: Text(_counter.toString(),
        style: TextStyle(fontSize: 64, fontWeight: FontWeight.bold)),
    );
  }

  buildLandscapeWidget() {
    return Container(
      color: Colors.green,
      alignment: Alignment.center,
      child: Text(_counter.toString(),
          style: TextStyle(fontSize: 64, fontWeight: FontWeight.bold)),
    );
  }
```



AsyncTask

(or Handlers, Threads, RxJava, Coroutines)

Dart Async Library

- Dart executes in a single “thread” (called an **Isolate**).
- You can safely execute I/O bound operations using **Futures** and **async/await**.
- If you need CPU intensive work, offload that work to an **Isolate**.

Isolates vs Threads

- Isolates don’t share memory.
- Communication between Isolates are via messages.
- Processor time shared

- Threads can share memory via heap.
- Communication between Threads are via shared memory or messages.
- Processor time shared

Async Example

```
import 'package:http/http.dart' as http;

Future<List<Movie>> upcomingMovies() async{
    var url = Uri.https(MOVIE_DB_BASE_URL, '/3/movie/upcoming',
        { 'api_key': API_KEY,
          'language': 'en-US'
        });

    var response = await http.get(url);

    var body = json.decode(response.body);

    return List<Movie>.from(body.map((movie) => Movie.fromJson(movie)));
}
```

Async Example

```dart
import 'package:http/http.dart' as http;

Future<List<Movie>> upcomingMovies() async{
    var url = Uri.https(MOVIE_DB_BASE_URL, '/3/movie/upcoming',
        { 'api_key': API_KEY,
          'language': 'en-US'
        });

    var response = await http.get(url);

    var body = json.decode(response.body);

    return List<Movie>.from(body.map((movie) => Movie.fromJson(movie)));
}
```

Async Example

```
import 'package:http/http.dart' as http;

Future<List<Movie>> upcomingMovies() async{
    var url = Uri.https(MOVIE_DB_BASE_URL, '/3/movie/upcoming',
        { 'api_key': API_KEY,
          'language': 'en-US'
        });

    var response = await http.get(url);

    var body = json.decode(response.body);

    return List<Movie>.from(body.map((movie) => Movie.fromJson(movie)));
}
```

Isolate Example

```
import 'package:http/http.dart' as http;
import 'package:flutter/foundation.dart';

static List<Movie> _computeMovies(dynamic body) =>
    List<Movie>.from(body.map((movie) => Movie.fromJson(movie)));

Future<List<Movie>> upcomingMovies() async{
    ...
    var body = json.decode(response.body);

    return compute(_computeMovies, body['results']);
}
```

Isolate Example

```
import 'package:http/http.dart' as http;
import 'package:flutter/foundation.dart';

static List<Movie> _computeMovies(dynamic body) =>
    List<Movie>.from(body.map((movie) => Movie.fromJson(movie)));

Future<List<Movie>> upcomingMovies() async{
    ...
    var body = json.decode(response.body);

    return compute(_computeMovies, body['results']);
}
```

Intents

Intents

- Starting activities.
- Asking the OS for an application.
- Starting Services
- Sending messages to other Android Components.

Intents

- Starting activities.
- Asking the OS for an application.
- Starting Services
- Sending messages to other Android Components.

Navigation

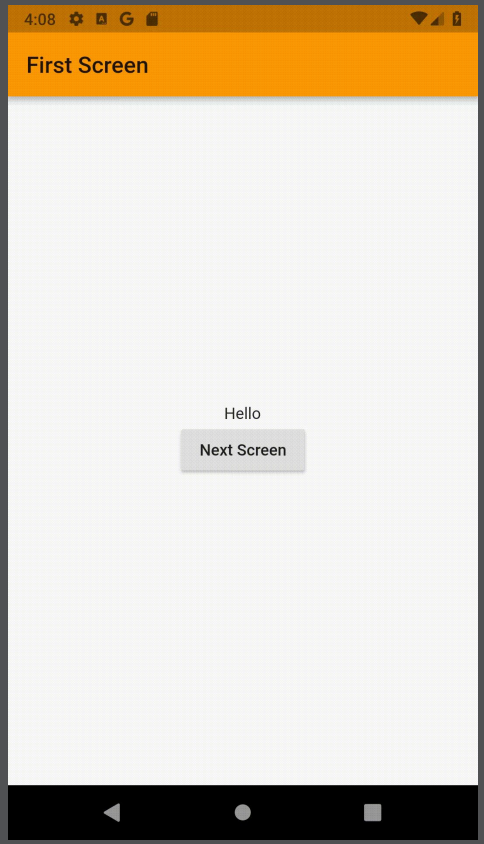
```
class FirstScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(...
            RaisedButton(child: Text("Next Screen"),
              onPressed: () => _goToNextScreen(context),
            )
          ]),
    );
  }

  _goToNextScreen(BuildContext context) =>
    Navigator.push(context,
      MaterialPageRoute(builder: (context) => SecondScreen()),
    );
}
```

Navigation

```
class FirstScreen extends StatelessWidget {
  ...
  _goToNextScreen(BuildContext context) =>
    Navigator.push(context,
      MaterialPageRoute(builder: (context) => SecondScreen()),
    );
}
```

Navigation

```
class FirstScreen extends StatelessWidget {
  ...
  _goToNextScreen(BuildContext context) =>
    Navigator.push(context,
      MaterialPageRoute(builder: (context) => SecondScreen()),
    );
}
```



Navigation

```
class App extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      initialRoute: '/',
      routes: {
        '/': (context) => FirstScreen(),
        '/second': (context) => SecondScreen(),
      },
    );
  }
}
```

Navigation

```
Navigator.pushNamed(context, '/second');

Navigator.pop(context);
```

startActivityForResult

startActivityForResult

```
Widget build(BuildContext context) { ... }

_goToResultScreen(BuildContext context) async {
  final result = await Navigator.push(context,
          MaterialPageRoute(builder: (context) => SecondScreen()));

  Scaffold.of(context)
    ..removeCurrentSnackBar()
    ..showSnackBar(SnackBar(content: Text("$result")));
}
```

startActivityForResult

```
Widget build(BuildContext context) { ... }

_goToResultScreen(BuildContext context) async {
  final result = await Navigator.push(context,
          MaterialPageRoute(builder: (context) => SecondScreen()));

  Scaffold.of(context)
    ..removeCurrentSnackBar()
    ..showSnackBar(SnackBar(content: Text("$result")));
}
```

startActivityForResult

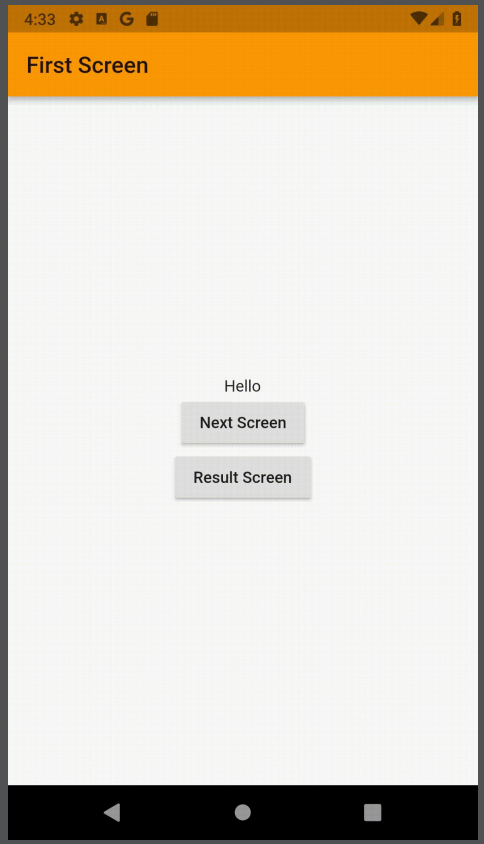
```
class SecondScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    ...
            RaisedButton(
              child: Text("Send Back Result"),
              onPressed: () => _goBack(context))
    ...
  }

  _goBack(BuildContext context) =>
    Navigator.pop(context, "I'm done!");
}
```

startActivityForResult

```
class SecondScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) { ... }

  _goBack(BuildContext context) =>
    Navigator.pop(context, "I'm done!");
}
```

startActivityForResult

```
class SecondScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    ...
            RaisedButton(
              child: Text("Send Back Result"),
              onPressed: () => _goBack(context))
    ...
  }

  _goBack(BuildContext context) =>
    Navigator.pop(context, "I'm done!");
}
```



Implicit Intents

Implicit Intents

```kotlin
// Kotlin
val i = Intent(Intent.ACTION_VIEW)
i.data = Uri.parse("https://www.imdb.com/name/nm0796117")
i.addCategory("android.intent.category.BROWSABLE")
```

Implicit Intents

pubspec.yaml

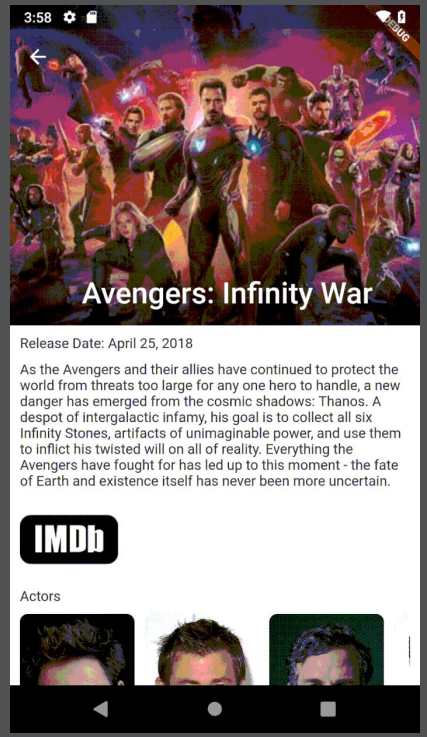
```
dependencies:
  android_intent: ^0.2.1
```

Implicit Intents

```
import 'package:android_intent/android_intent.dart';


AndroidIntent intent = new AndroidIntent(
    action: 'action_view',
    data: url
);
await intent.launch();
```

Implicit Intents

```
import 'package:android_intent/android_intent.dart';


AndroidIntent intent = new AndroidIntent(
    action: 'action_view',
    data: url
);
await intent.launch();
```



Services

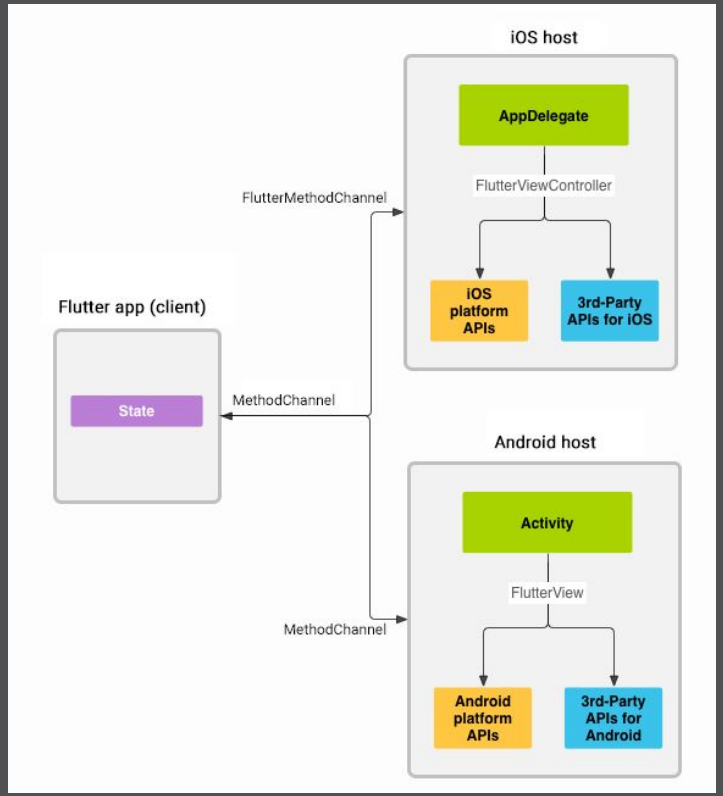
Aw, Snap!

Something went wrong

Learn more

Send feedback

Method Channel



Method Channel

```
class _MyBatteryState extends State<MyBatteryWidget> {
  static const platform = const MethodChannel('com.myotive.example/battery');

  // Get battery level.
}
```

Method Channel

```
Future<void> _getBatteryLevel() async {
    String batteryLevel;
    try {
      final int result = await platform.invokeMethod('getBatteryLevel');
      batteryLevel = 'Battery level at $result % .';
    } on PlatformException catch (e) {
      batteryLevel = "Failed to get battery level: '${e.message}'.";
    }
    setState(() {
      _batteryLevel = batteryLevel;
    });
}
```

Method Channel

```
Future<void> _getBatteryLevel() async {
    String batteryLevel;
    try {
      final int result = await platform.invokeMethod('getBatteryLevel');
      batteryLevel = 'Battery level at $result % .';
    } on PlatformException catch (e) {
      batteryLevel = "Failed to get battery level: '${e.message}'.";
    }
    setState(() {
      _batteryLevel = batteryLevel;
    });
}
```

Method Channel

android > app > src > main > kotlin > <Your Domain> > MainActivity.kt

```
class MainActivity() : FlutterActivity() {
    private val CHANNEL = "com.myotive.example/battery"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        GeneratedPluginRegistrant.registerWith(this)
        MethodChannel(flutterView, CHANNEL).setMethodCallHandler { call, result ->
            ...
        }
    }
    ...
}
```

Method Channel

android > app > src > main > kotlin > <Your Domain> > MainActivity.kt

```
class MainActivity() : FlutterActivity() {
    private val CHANNEL = "com.myotive.example/battery"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        GeneratedPluginRegistrant.registerWith(this)
        MethodChannel(flutterView, CHANNEL).setMethodCallHandler { call, result ->
            ...
        }
    }
    ...
}
```

Method Channel

android > app > src > main > kotlin > <Your Domain> > MainActivity.kt

```
class MainActivity() : FlutterActivity() {
    private val CHANNEL = "com.myotive.example/battery"

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        GeneratedPluginRegistrant.registerWith(this)
        MethodChannel(flutterView, CHANNEL).setMethodCallHandler { call, result ->
            ...
        }
    }
    ...
}
```

Method Channel

android > app > src > main > kotlin > <Your Domain> > MainActivity.kt

```
MethodChannel(flutterView, CHANNEL).setMethodCallHandler { call, result ->
    when {
        call.method == "getBatteryLevel" -> {
            val batteryLevel = getBatteryLevel()
            if (batteryLevel != -1) {
                result.success(batteryLevel)
            } else {
                result.error("UNAVAILABLE", "Battery level not available", null)
            }
        }
        else -> result.notImplemented()
    }
}
```

Method Channel

Dart	Android
null	null
bool	java.lang.Boolean
int	java.lang.Integer
Int (if 32 bits not enough)	java.lang.Long
double	java.lang.Double
String	java.lang.String
Uint8List	byte[]
Int32List	int[]
Int64List	long[]
Float64List	double[]
List	java.util.ArrayList
Map	java.util.HashMap

https://flutter.io/docs/development/platform-integration/platform-channels#platform-channel-data-types-support-and-codecs

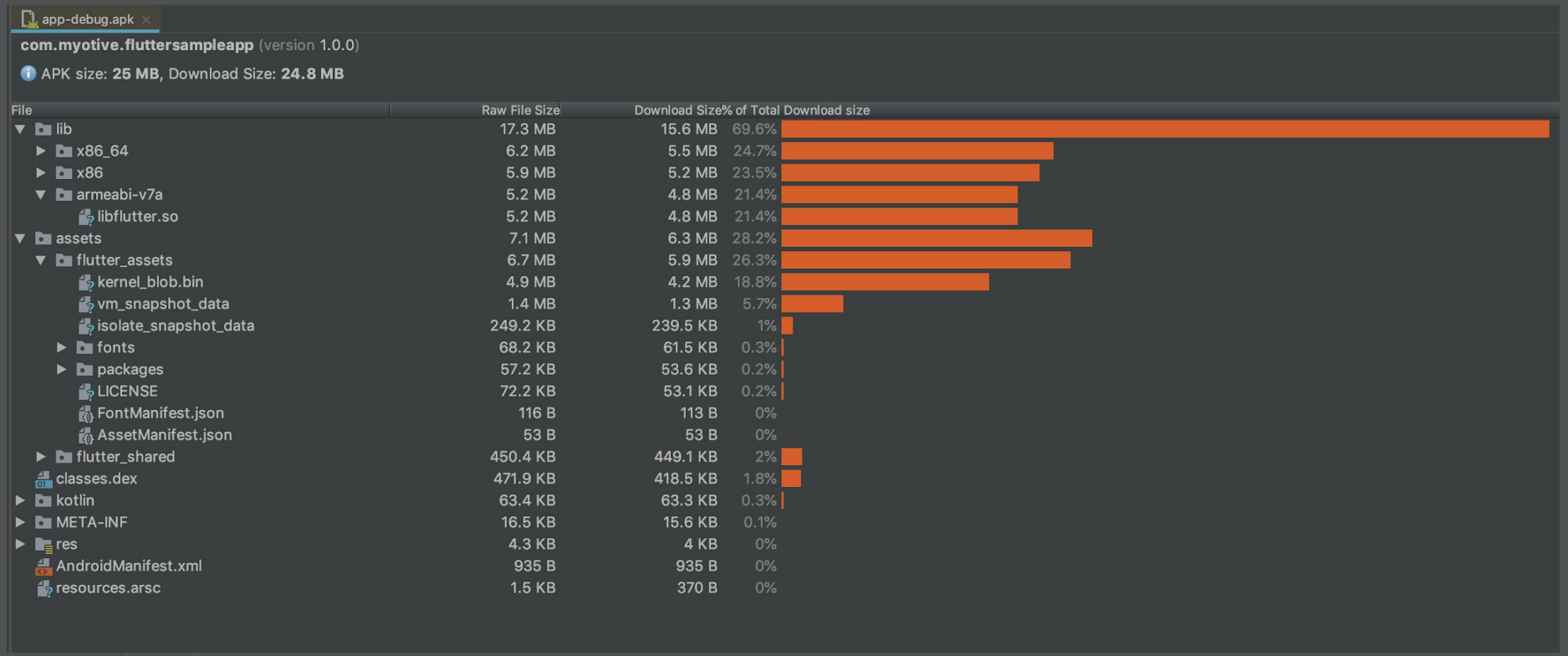

APK

Building an APK

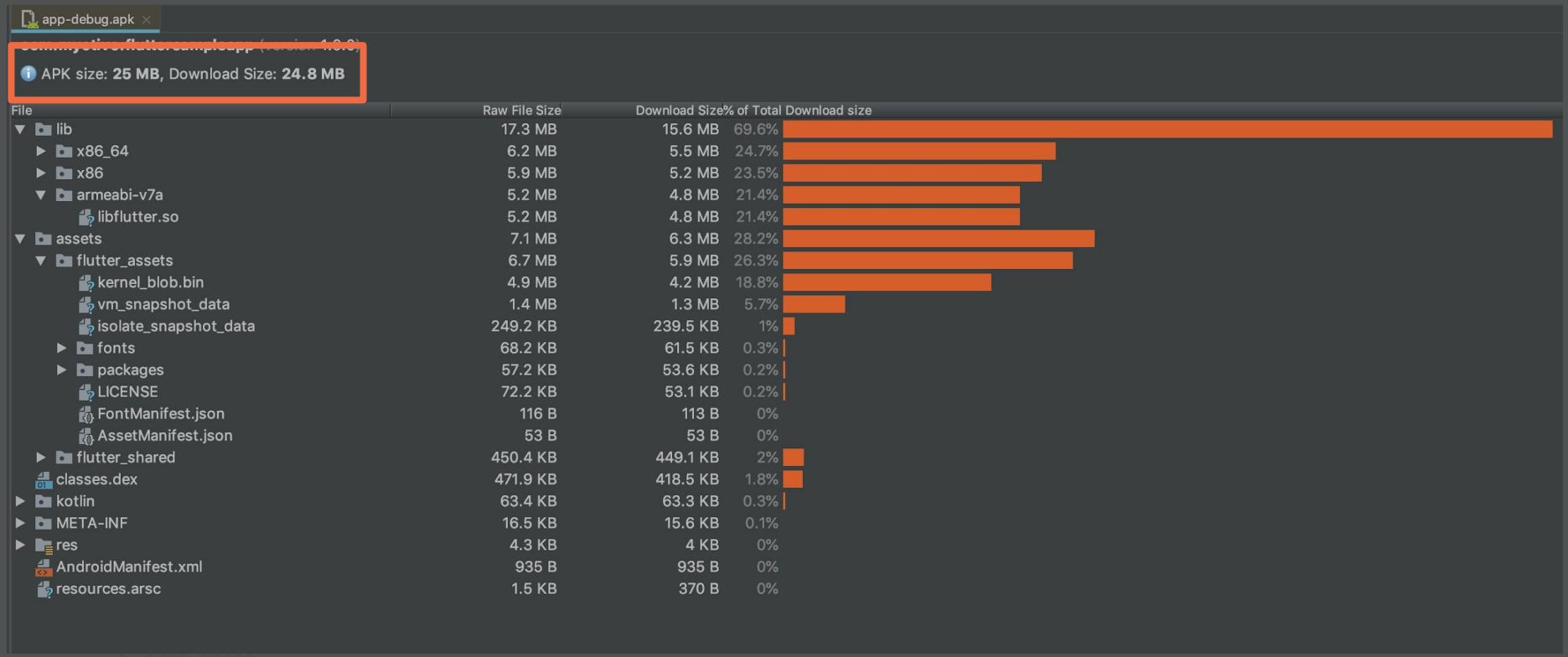
- You must update the build.gradle to not use the debug keystore!!
- If proguarding, add rules to keep Flutter specific classes.
- You are responsible for updating your app's launcher icon.
- flutter build apk

https://flutter.io/docs/deployment/android

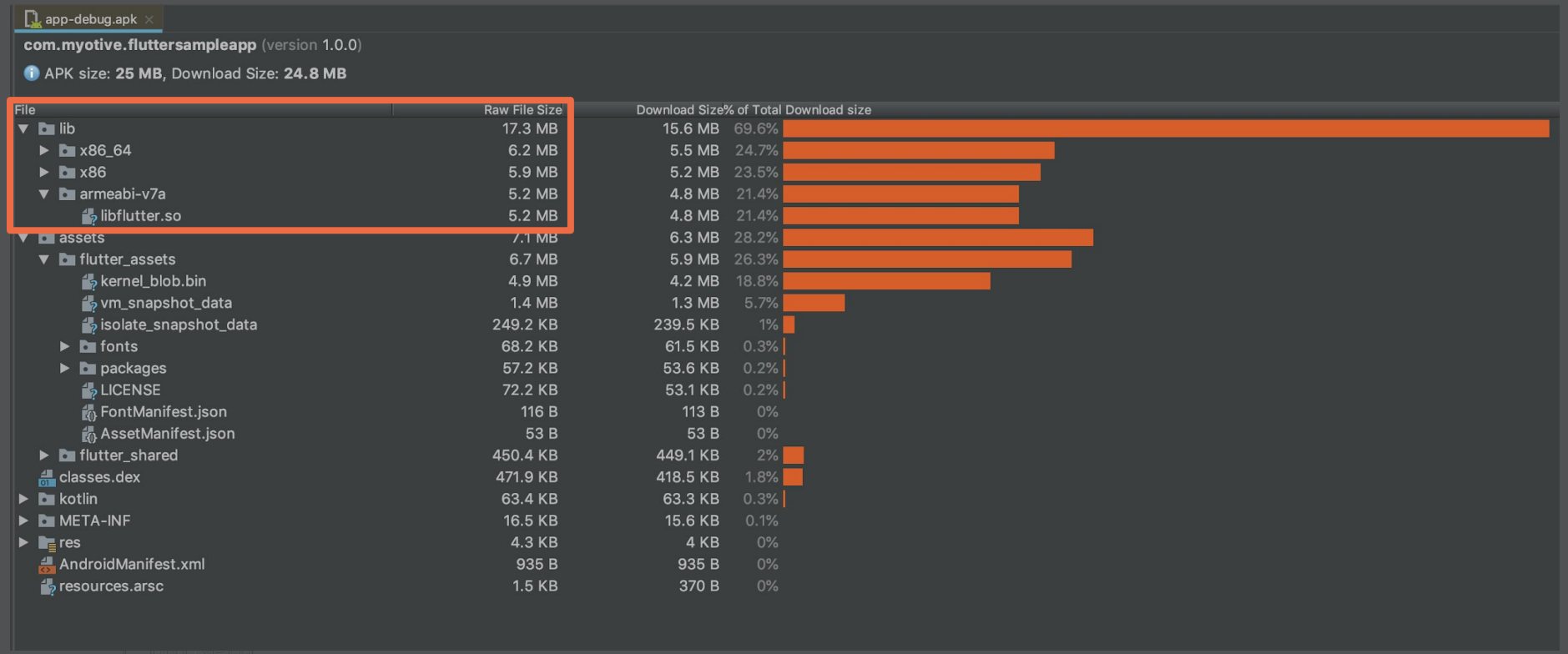
APK - Debug Version



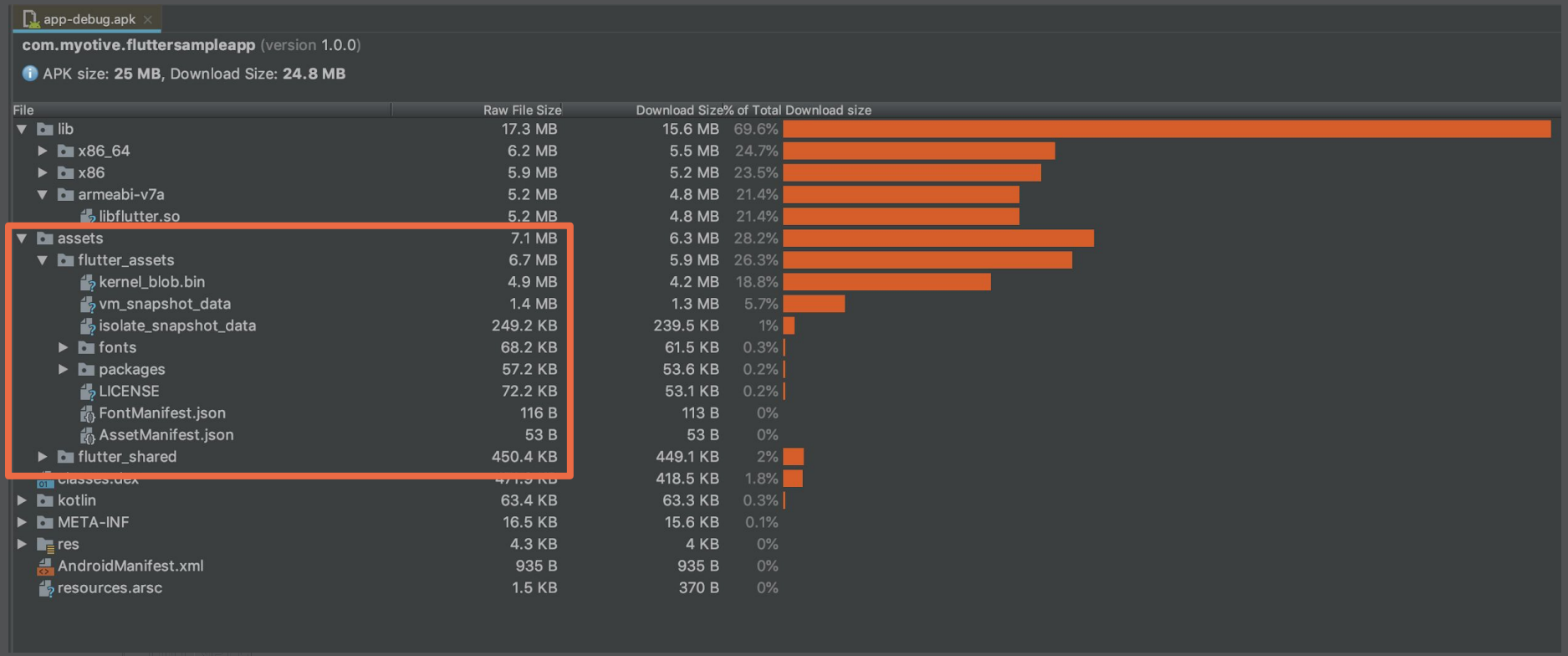
APK - Debug Version



APK - Debug Version

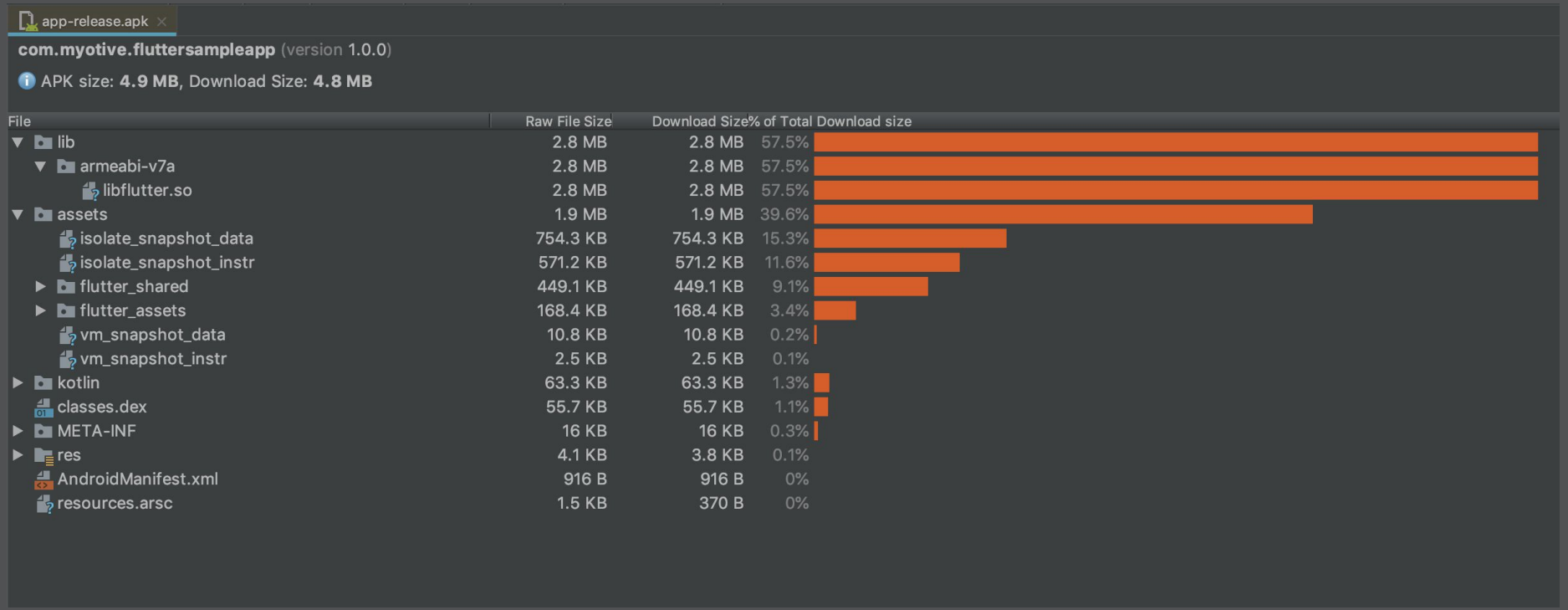


APK - Debug Version

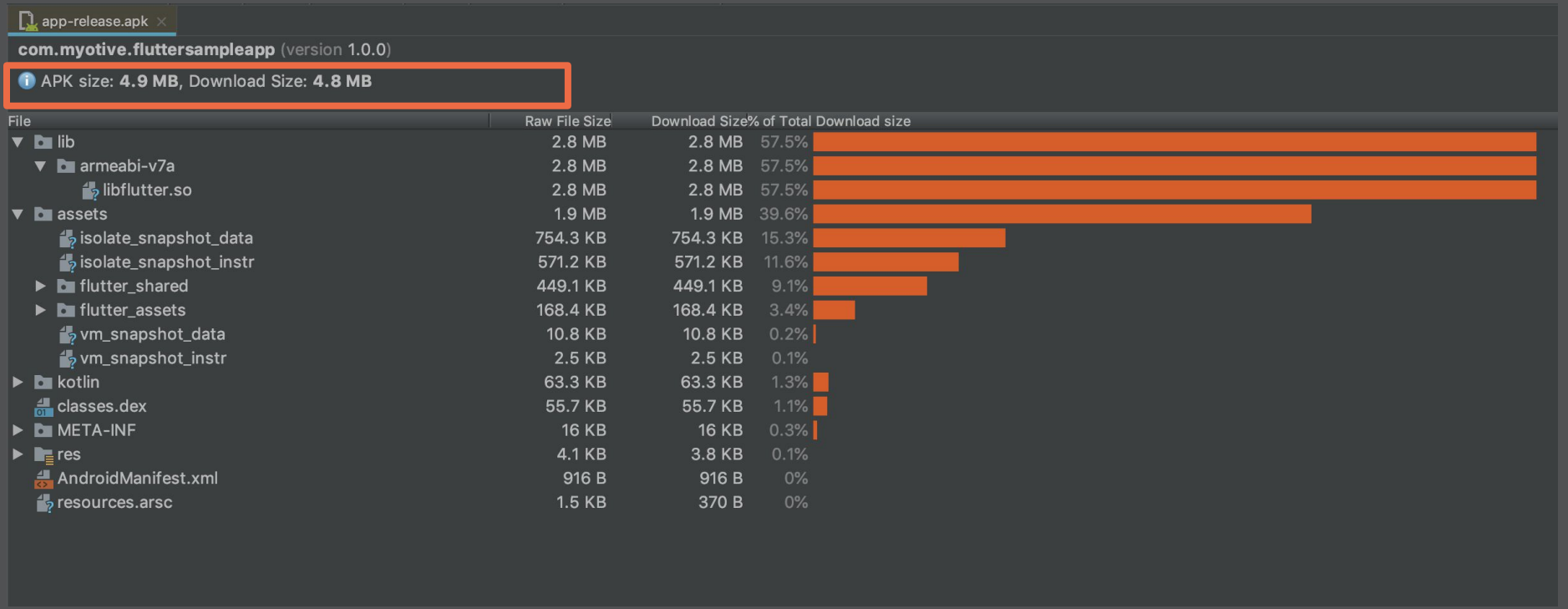


APK - Debug Version

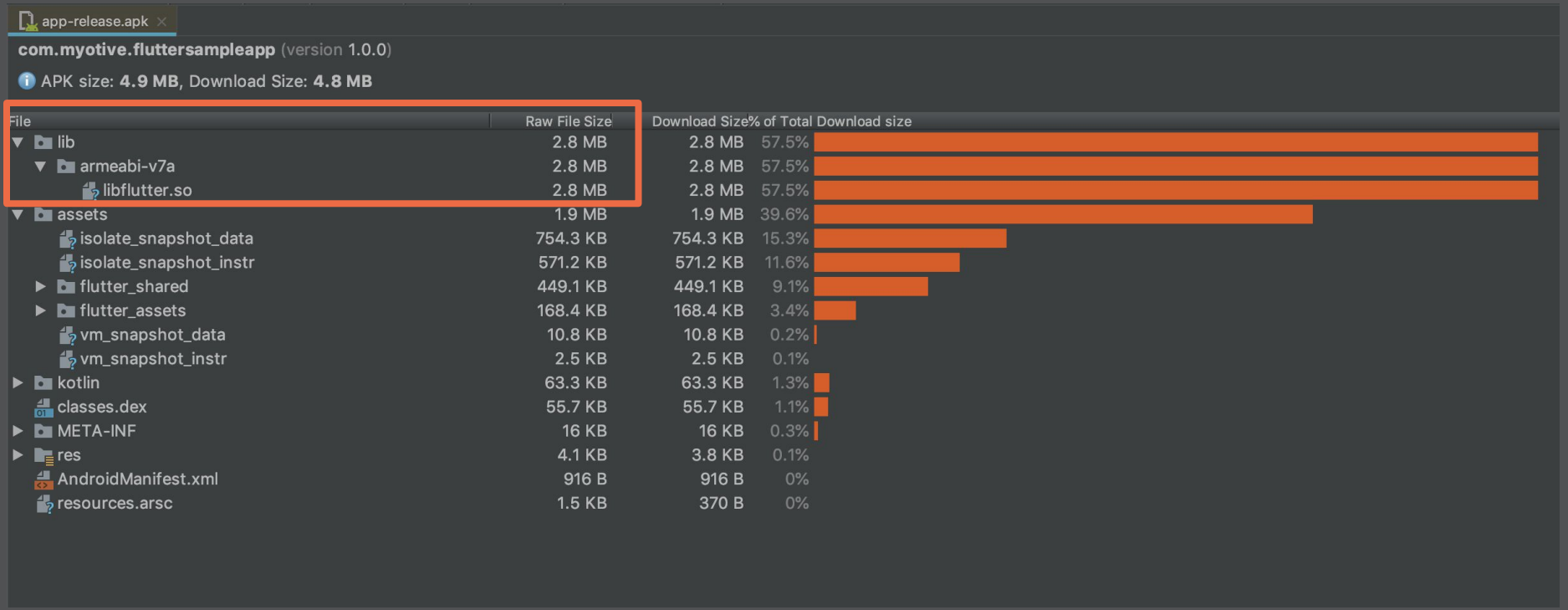
APK - Release Version



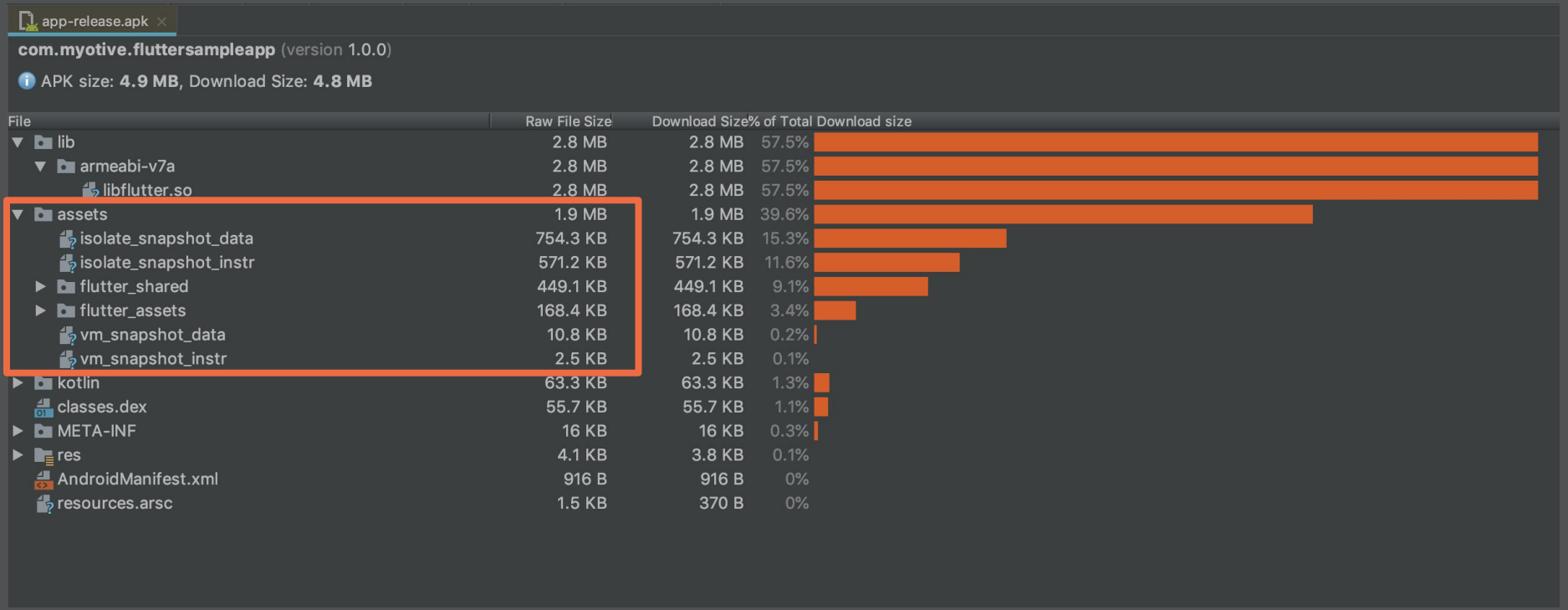
APK - Release Version



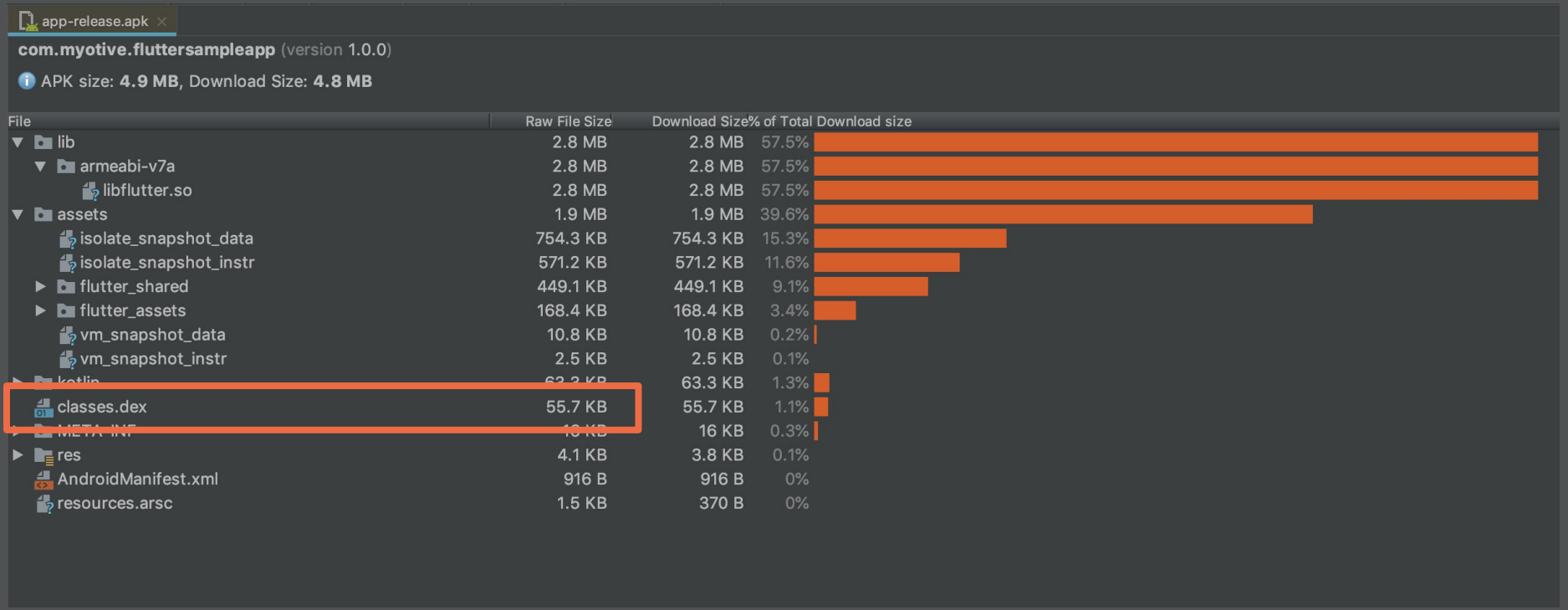
APK - Release Version



APK - Release Version



APK - Release Version



Conclusion

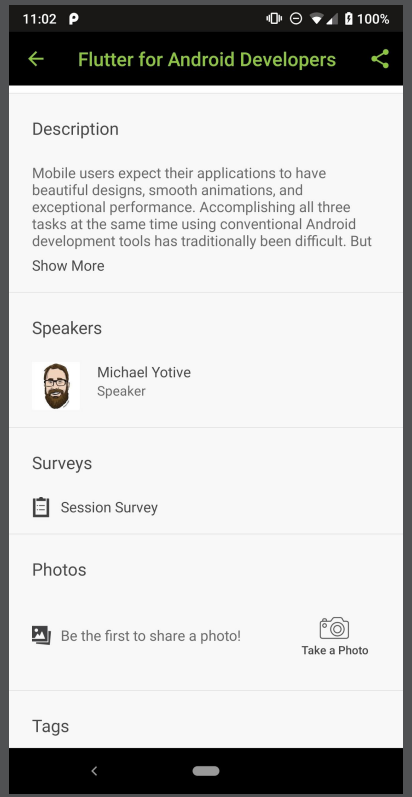
Recap

- Flutter is great for UI platform (not just for mobile).
- Don’t think about Flutter as being a replacement for Android.
- Flutter is a great tool for working with your fellow iOS counterparts.

Resources

- Udacity Flutter Course (free)
  https://mena.udacity.com/course/build-native-mobile-apps-with-flutter--ud905
- Flutter For Android Developers
  https://flutter.io/docs/get-started/flutter-for/android-devs
- It's all Widgets
  https://itsallwidgets.com/
- GitHub
  https://www.github.com/myotive/flutter_movie_db

Session Survey



Thank You!

www.bignerdranch.com