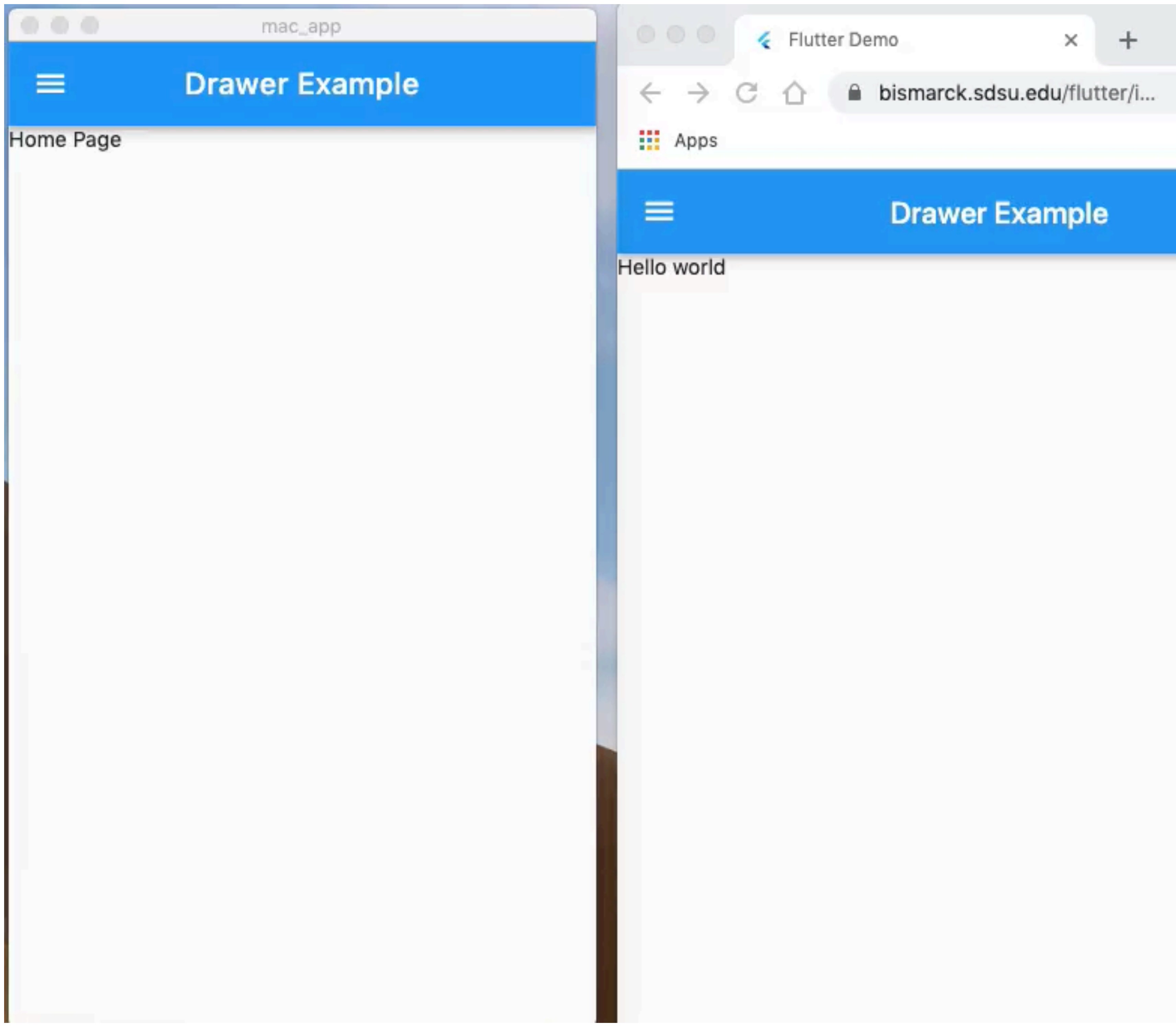
CS 696 Multi-platform Mobile App Development
Fall Semester, 2020
Doc 13 Web & Desktop
Oct 13, 2020

Other Platforms

Flutter also runs on

MacOS	dev channel
Windows	dev channel
Linux	dev channel
Web	beta channel



Flutter Channels

master

Absolute latest cutting edge build

dev

The latest fully-tested build

beta

Branch from master for a new beta release at the beginning of the month,

stable

Roughly once a quarter, a branch that has been stabilized on beta becomes stable

Changing Channels

Current Channel

AI pro 15->flutter channel

```
Flutter channels:
  master
* dev
  beta
  stable
```

Changing Channels

```
flutter channel dev
flutter upgrade
```

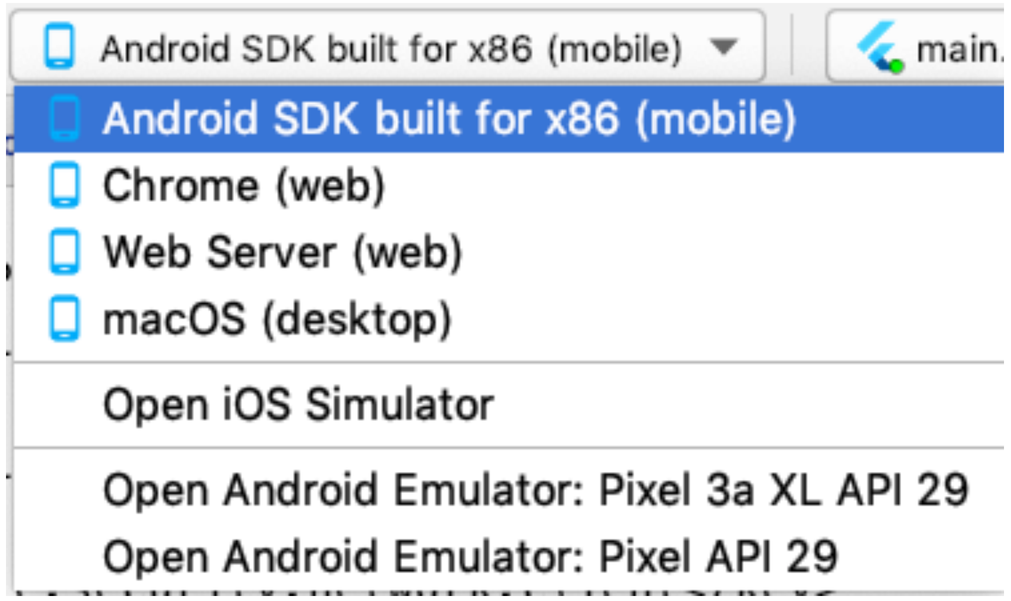
Enabling Other Platforms

After changing to correct channel

```
$ flutter config --enable-windows-desktop
$ flutter config --enable-macos-desktop
$ flutter config --enable-linux-desktop
$ flutter config --enable-web
```

If adding to existing project run

```
flutter create .
```



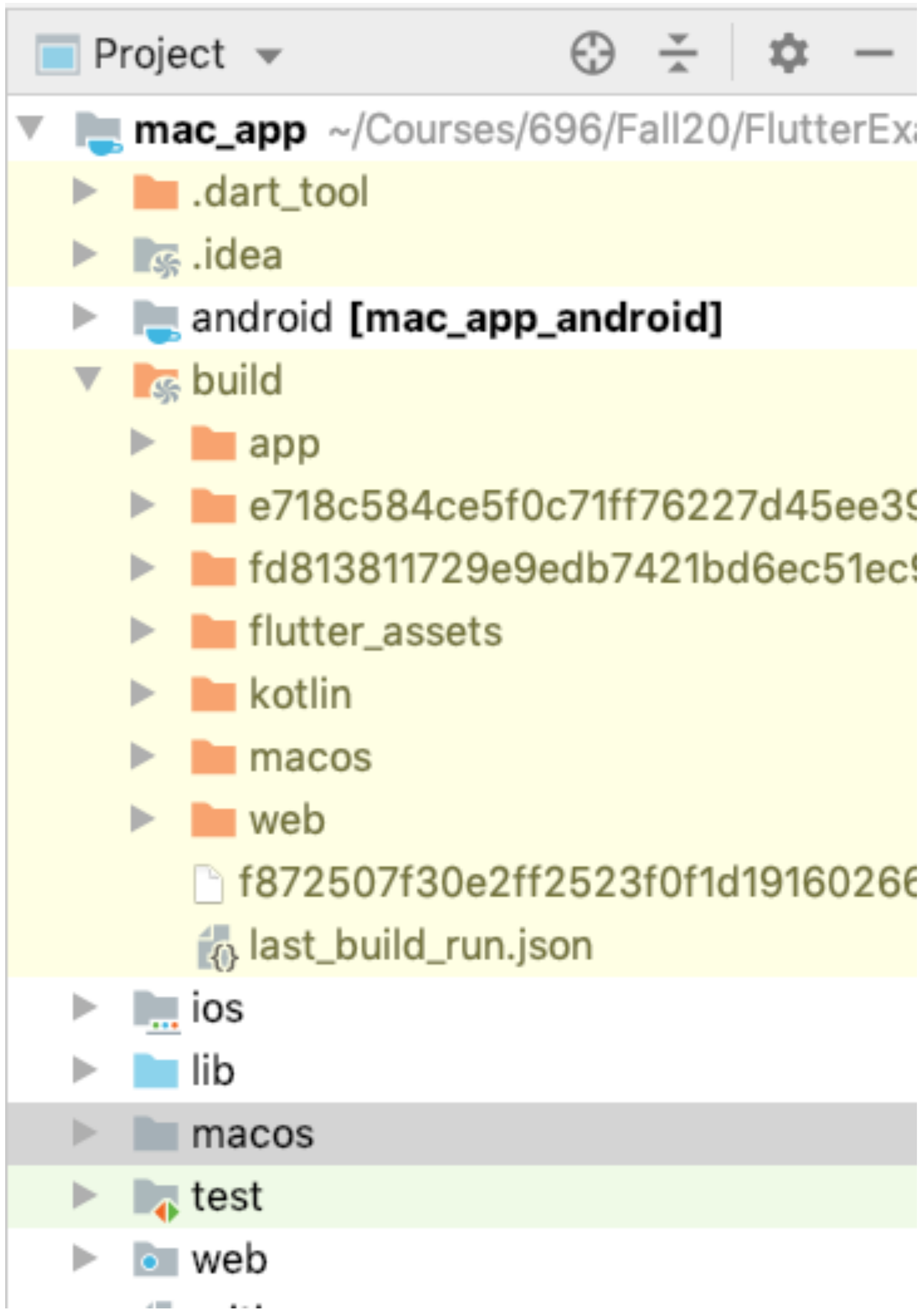
Android SDK built for x86 (mobile)
main.
Android SDK built for x86 (mobile)
Chrome (web)
Web Server (web)
macOS (desktop)
Open iOS Simulator
Open Android Emulator: Pixel 3a XL API 29
Open Android Emulator: Pixel API 29

Release Build

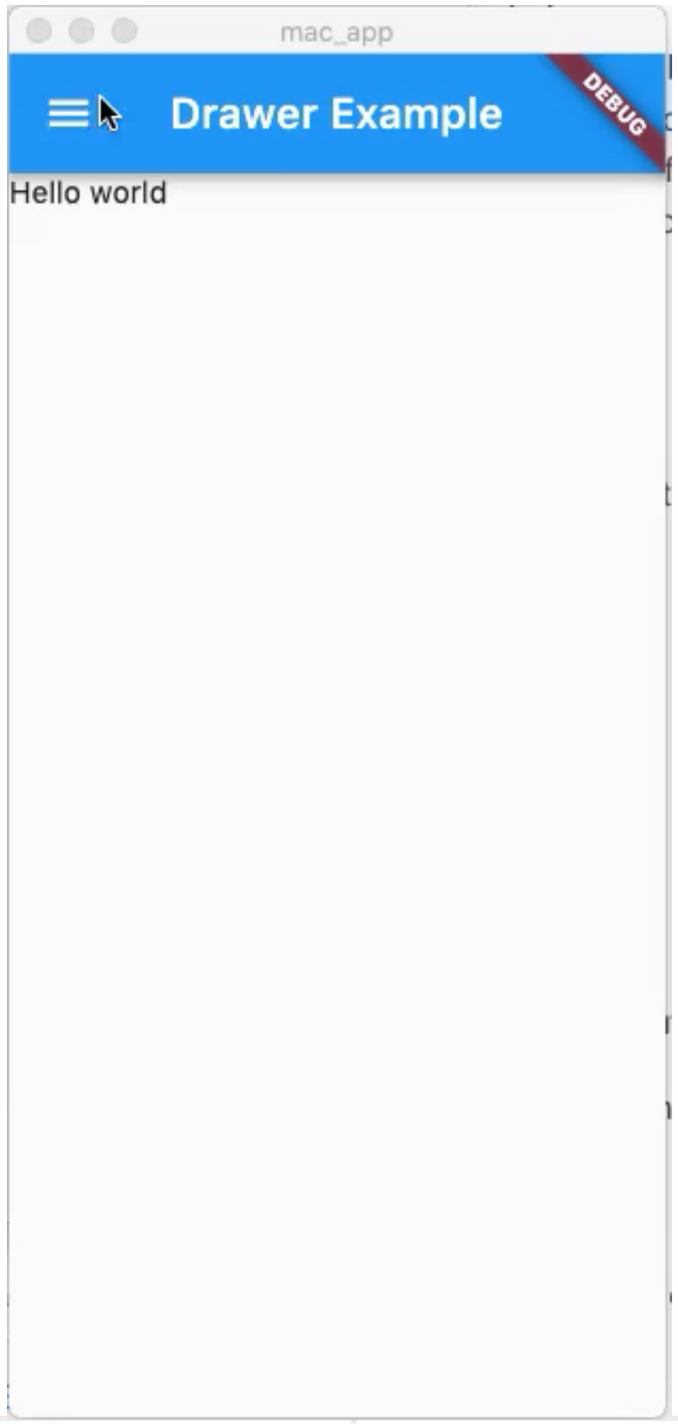
flutter build web
flutter build windows
flutter build macos
flutter build linux

Drawer Example

https://bismarck.sdsu.edu/flutter/instapost/#/



```
mac_app ~/Courses/696/Fall20/FlutterEx
  .dart_tool
  .idea
  android [mac_app_android]
  build
    app
    e718c584ce5f0c71ff76227d45ee3
    fd813811729e9edb7421bd6ec51ec
    flutter_assets
    kotlin
    macos
    web
    f872507f30e2ff2523f0f1d1916026
    last_build_run.json
  ios
  lib
  macos
  test
  web
```



Hello world

Platform Requirements

Windows

- Visual Studio 2019
- Desktop development with C++ workload, all of its default components

MacOS

- Xcode
- CocoaPods if you use plugins

Linux

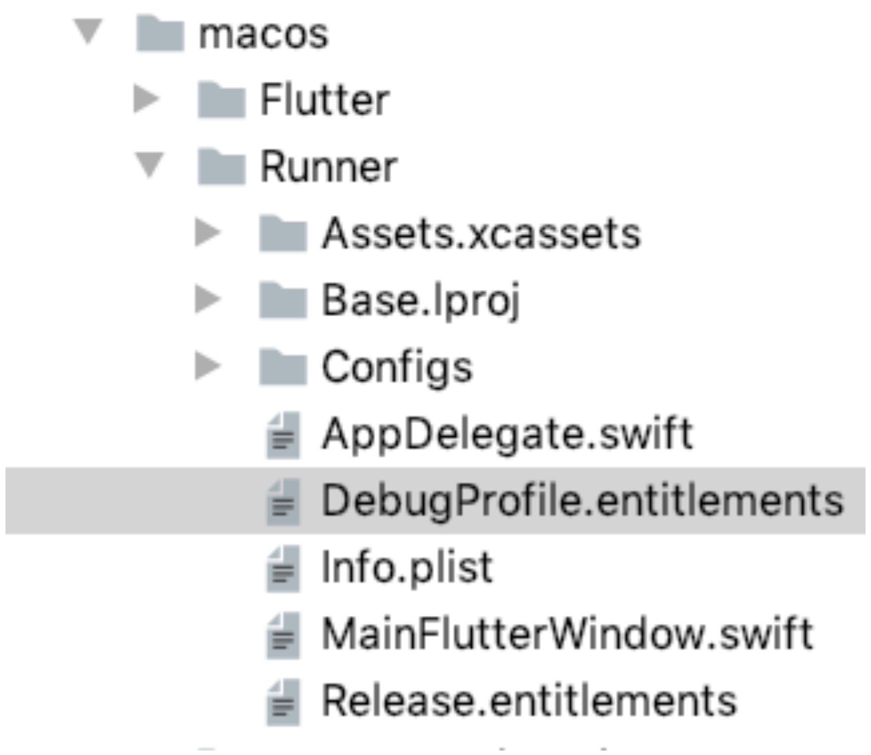
- Clang
- CMake
- GTK development headers
- Ninja build
- pkg-config
- libblkid

Configuring Platforms

Different platforms need different configurations

macOS apps

- Sandboxed

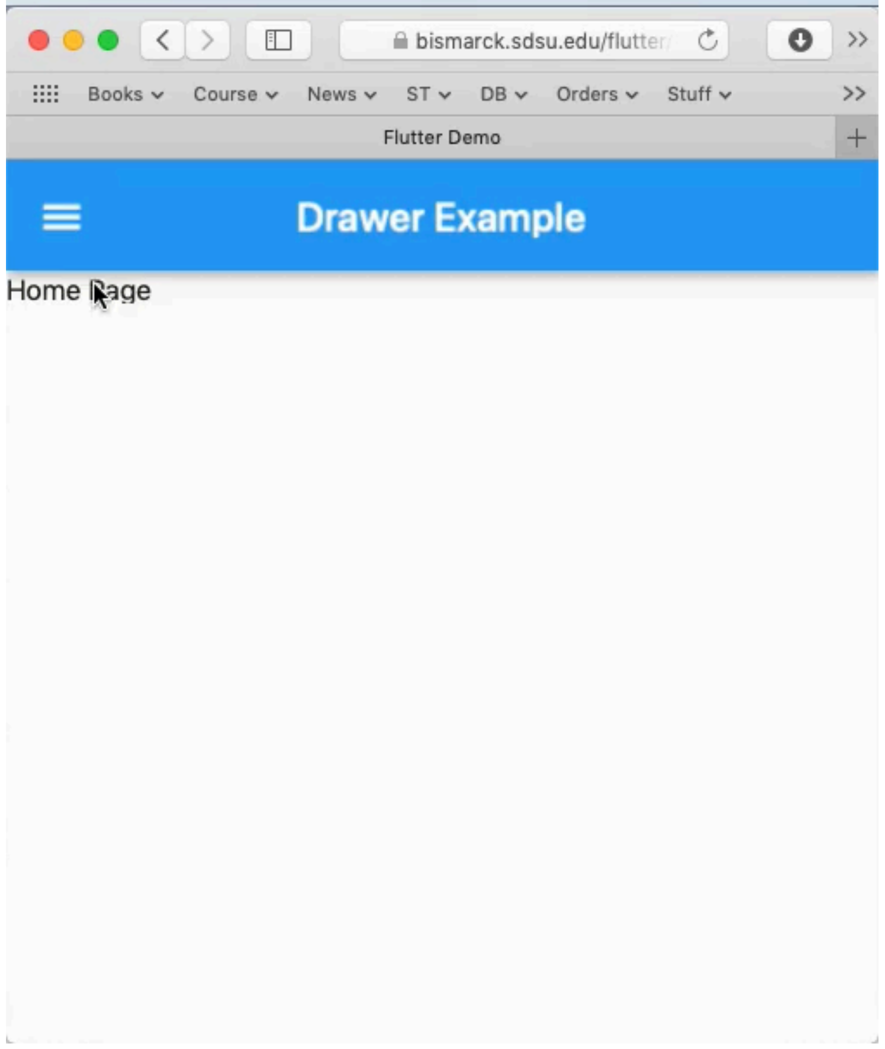


For Hot Restart

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://w
<plist version="1.0">
<dict>
    <key>com.apple.security.app-sandbox</key>
    <true/>
    <key>com.apple.security.cs.allow-jit</key>
    <true/>
    <key>com.apple.security.network.server</key>
    <true/>
    <key>com.apple.security.network.client</key>
    <true/>
</dict>
</plist>
```

For Debugging

For Accessing Bismarck



Changing the "Top Level" Widget

```
class MyHomePage extends StatefulWidget {
  MyHomePage({Key key, this.title}) : super(key: key);
  final String title;

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

class _MyHomePageState extends State<MyHomePage> {
  Widget currentBody = Text('Hello world');

  void setWidget(Function getWidget) {
    setState(() {
      currentBody = getWidget();
    });
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      drawer: DrawerCode(setWidget),
      body: currentBody,
    );
  }
}
```

```
() => CountPage(title: "Count",)

() => Text("Home Page")
```

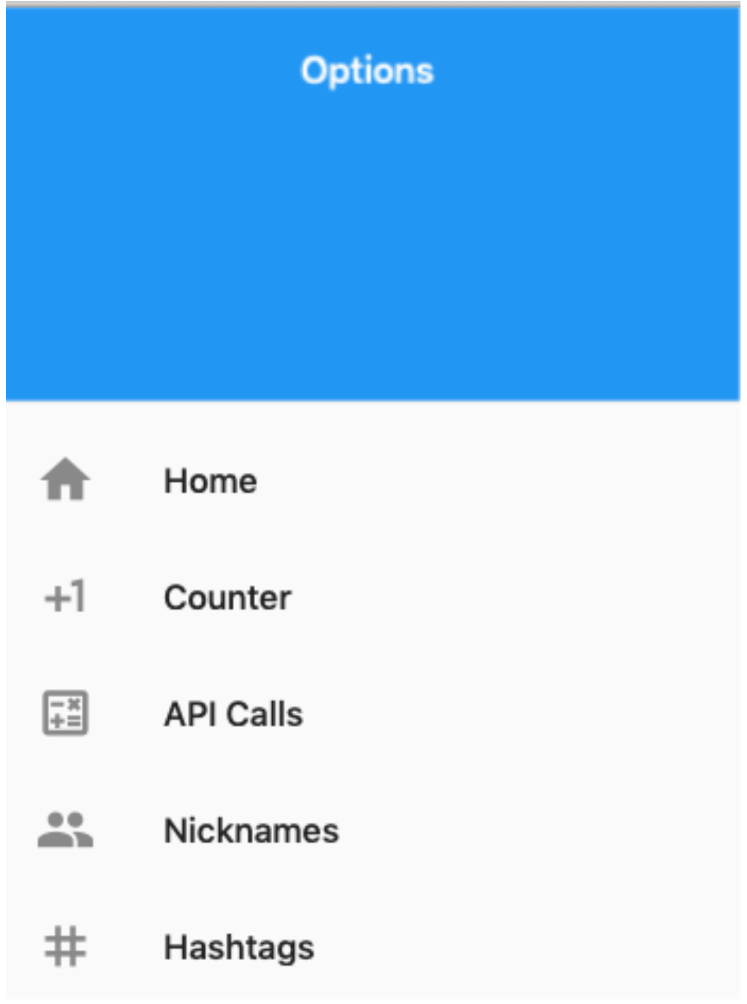
Method to Reduce Repetition

```
class DrawerCode extends StatelessWidget {
  final Function setWidget;

  DrawerCode(this.setWidget);

  Widget drawerTile(BuildContext context, {Icon icon, String title, Function getWidget}) {
    return ListTile(
      leading: icon,
      title: Text(title),
      onTap: () {
        Navigator.pop(context);
        setWidget(getWidget);
      }
    );
  }
```

Close the Drawer

Callback to parent widget to
Change what is being displayed

```
Widget build(BuildContext context) {
  return Drawer(
    child: ListView(
      padding: EdgeInsets.zero,
      children: <Widget>[
        DrawerHeader(
          decoration: BoxDecoration(color: Colors.blue),
          child: Column(
            children: <Widget>[
              Text(
                "Options",
                style: TextStyle(
                    fontSize: 15.0,
                    fontWeight: FontWeight.w500,
                    color: Colors.white),
              ),
            ],
          ),
        ),
        drawerTile(context,
            icon: Icon(Icons.home),
            title: "Home",
            getWidget: () => Text("Home Page")
        ),
        drawerTile(context,
            icon: Icon(Icons.plus_one),
            title: "Counter",
            getWidget: () => CountPage(title: "Count",)
        ),
```



Pass a lambda instead of Widget

So widget is created only when needed

But get different copy each time

Could give them keys so state remains

CountPage

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

class CountPage extends StatefulWidget {
  CountPage({Key key, this.title}) : super(key: key);
  final String title;

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

class _CountPageState extends State<CountPage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
      _counter++;
    });
  }

  @override
  Widget build(BuildContext context) {

    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text(
              'You have pushed the button this many times:',
            ),
            Text(
              '$_counter'
```

typedef

What type of Function?

```
void setWidget(Function getWidget) {
  setState(() {
    currentBody = getWidget();
  });
}
```

```
typedef Widget SetWidgetCallback();
```

```
void setWidget(SetWidgetCallback getWidget) {
  setState(() {
    currentBody = getWidget();
  });
}
```

Now we know what type of function

Desktop & Web Issues

Variable screen sizes

- PWA - Progressive Web Apps

Menus

Multiple Widows

Dealing With Screen Sizes

LayoutBuilder
MediaQuery.of
OrientationBuilder

AspectRatio
CustomSingleChildLayout
CustomMultiChildLayout
FittedBox
FractionallySizedBox

OrientationBuilder

Updates when size changes

Provides orientation

Orientation.landscape

Orientation.portrait

```
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(),
    body: OrientationBuilder(builder: (context, orientation)
```

OrientationBuilder Example

```
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(),
    body: OrientationBuilder(
      builder: (context, orientation) {
        return orientation == Orientation.portrait ? _buildVerticalLayout() : _buildHorizontalLayout();
      },
    ),
  );
}
```

Could be lot of repeated code

MediaQuery.of, MediaQueryData

MediaQuery.of

Returns MediaQueryData

MediaQueryData

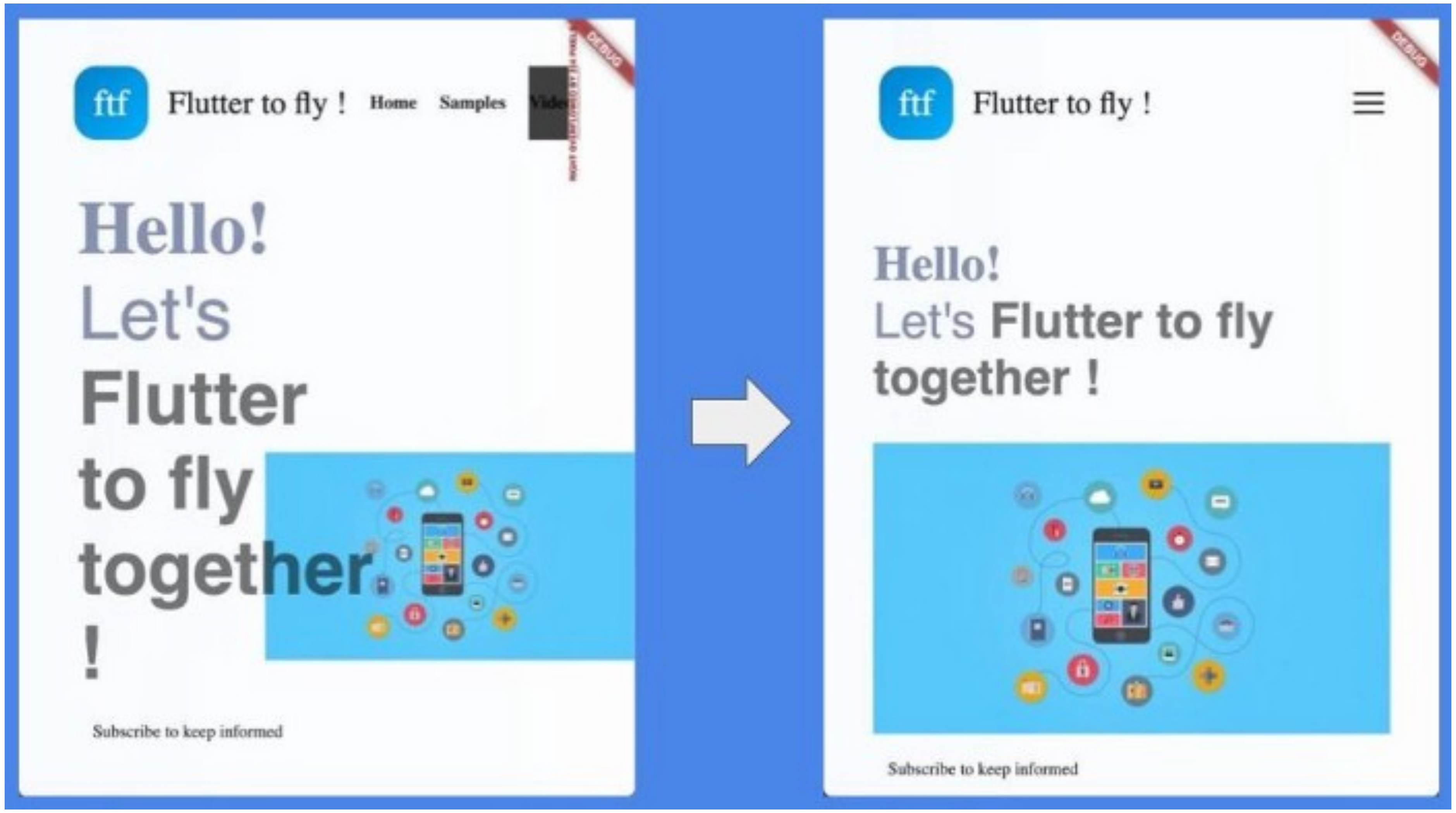
- size
- devicePixelRatio
- textScaleFactor
- platformBrightness
- padding
- viewInsets
- systemGestureInsets
- viewPadding
- alwaysUse24HourFormat
- accessibleNavigation
- invertColors
- highContrast
- disableAnimations
- boldText
- navigationMode

Progressive Web Apps (PWA)

Different Screen Sizes require different layouts

Example - Master-Detail

May need changes in fonts sizes, padding etc.


ftf
Flutter to fly !
Home
Samples
DEBUG
Hello!
Let's Flutter to fly together !
Subscribe to keep informed
ftf
Flutter to fly !
DEBUG
Hello!
Let's Flutter to fly together !
Subscribe to keep informed

Defining Screen Sizes

```
class ResponsiveWidget {

  static bool isLargeScreen(BuildContext context) {
    return MediaQuery.of(context).size.width > 1200;
  }

  static bool isSmallScreen(BuildContext context) {
    return MediaQuery.of(context).size.width < 800;
  }

  static bool isMediumScreen(BuildContext context) {
    return MediaQuery.of(context).size.width > 800 &&
    MediaQuery.of(context).size.width < 1200;
  }
}
```

https://medium.com/flutter-community/making-cross-platform-flutter-landing-page-responsive-7fffe0655970

ResponsiveWidget

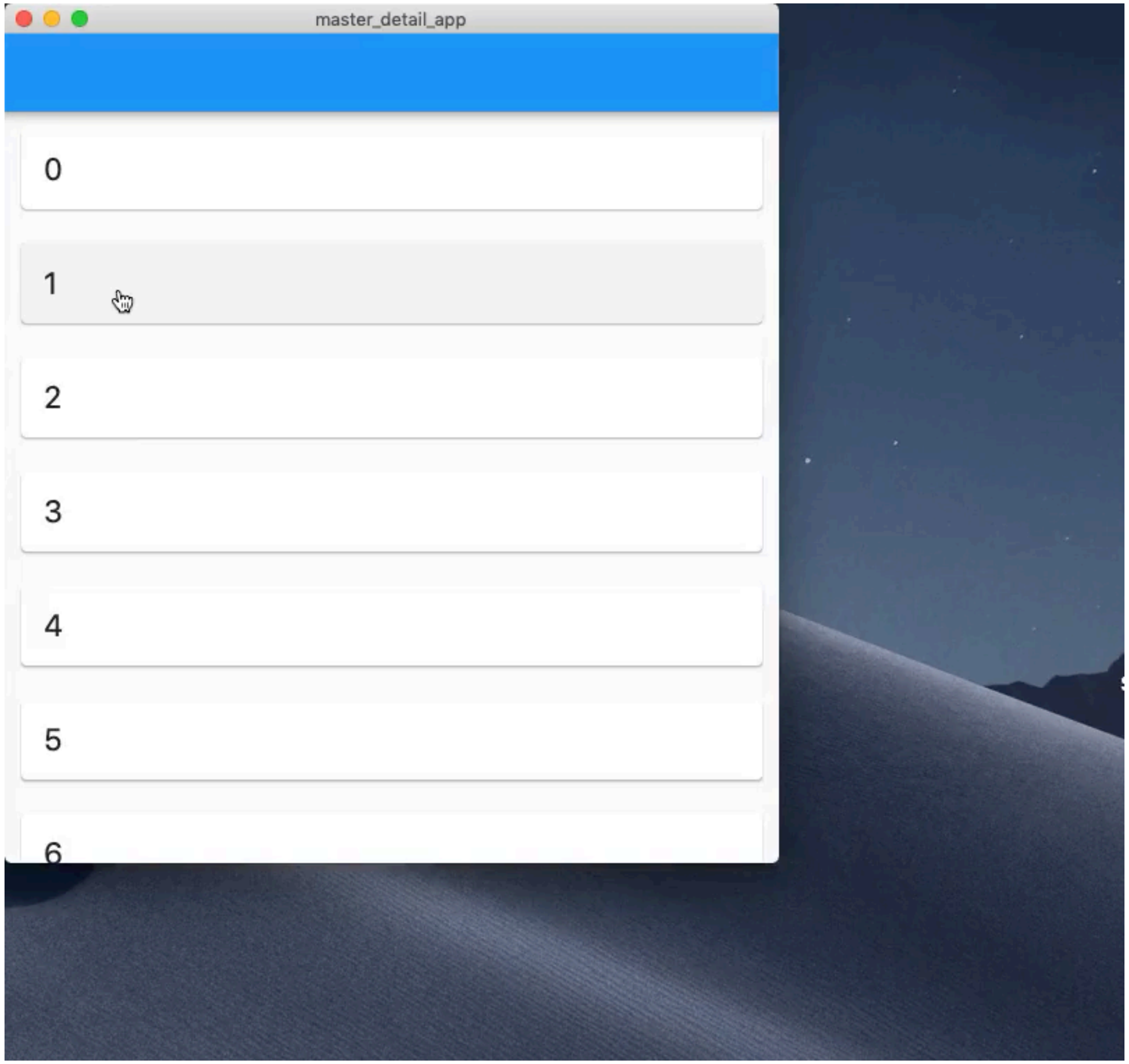
```
final Widget largeScreen;
final Widget mediumScreen;
final Widget smallScreen;

const ResponsiveWidget(
    {Key key, this.largeScreen, this.mediumScreen, this.smallScreen})
    : super(key: key);

@override
Widget build(BuildContext context) {
 return LayoutBuilder(builder: (context, constraints) {
    if (constraints.maxWidth > 1200) {
      return largeScreen;
    } else if (constraints.maxWidth > 800 && constraints.maxWidth < 1200) {
      return mediumScreen ?? largeScreen;
    } else {
      return smallScreen ?? largeScreen;
    }
  });
}
```

Small Variations

```
Text(
   Strings.subscribeButton,
   style: TextStyle(
      color: MyColors.white1,
      fontSize: ResponsiveWidget.isSmallScreen(context)
         ? 12
         : ResponsiveWidget.isMediumScreen(context) ? 12 : 16,
      letterSpacing: 1),
 ),
```

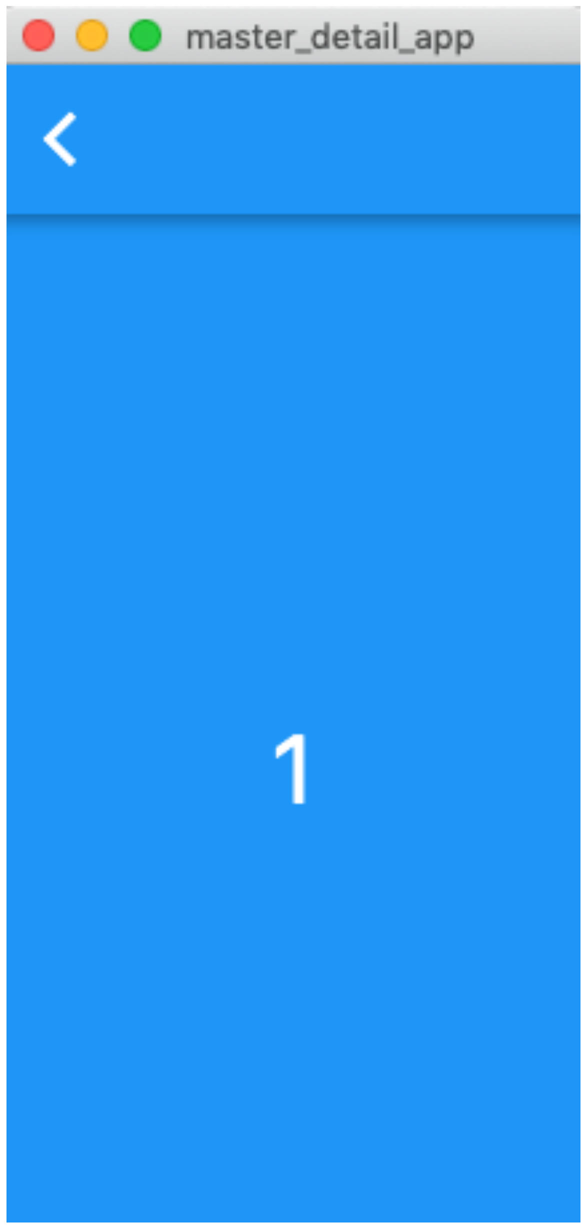
Master-Detail

Example from
Deven Joshi

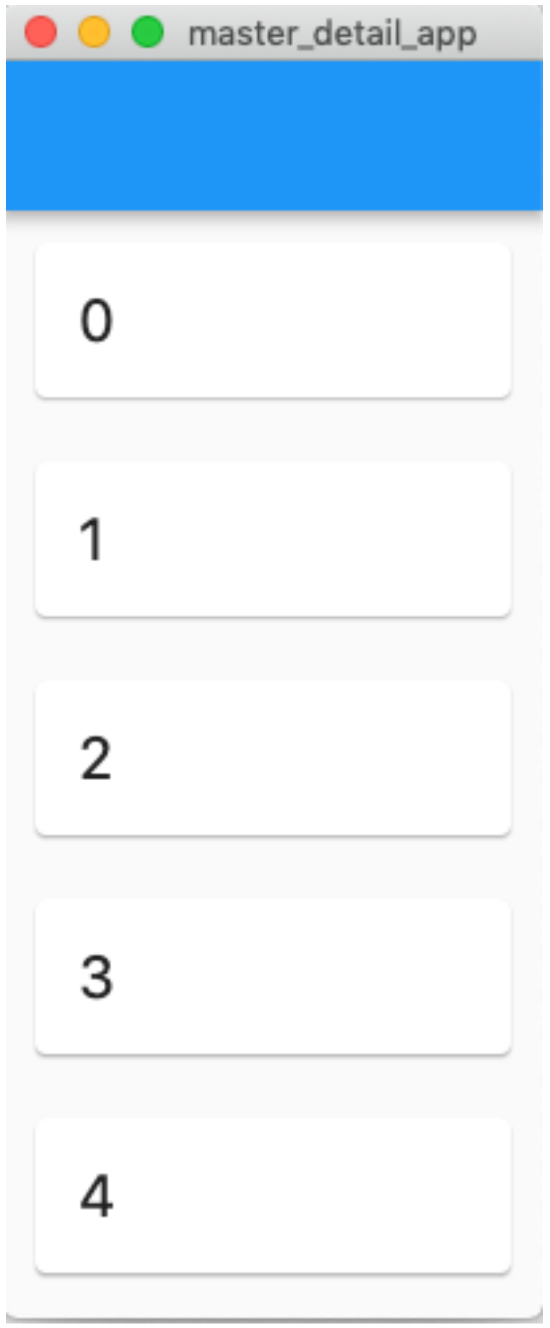


https://medium.com/flutter-community/developing-for-multiple-screen-sizes-and-orientations-in-flutter-fragments-in-flutter-a4c51b849434

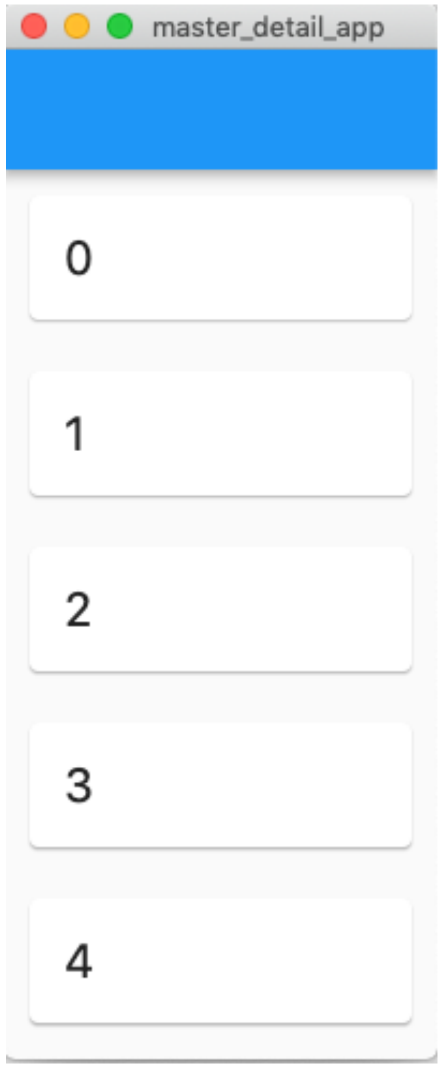
```dart
import 'package:flutter/material.dart';

class DetailWidget extends StatefulWidget {

  final int data;

  DetailWidget(this.data);

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

class _DetailWidgetState extends State<DetailWidget> {
  @override
  Widget build(BuildContext context) {
    return Container(
      color: Colors.blue,
      child: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text(widget.data.toString(), style: TextStyle(fontSize: 36.0, color: Colors.white),),
          ],
        ),
      ),
    );
  }
}
```



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

typedef Null ItemSelectedCallback(int value);

class ListWidget extends StatefulWidget {
  final int count;
  final ItemSelectedCallback onItemSelected;

  ListWidget(
      this.count,
      this.onItemSelected,
      );

  @override
  _ListWidgetState createState() => _ListWidgetState();
}
```

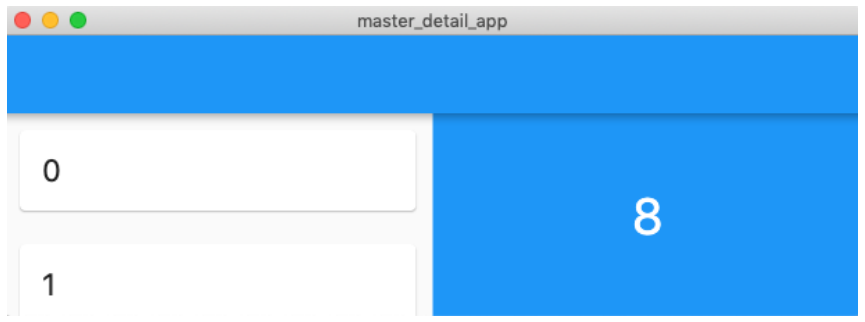


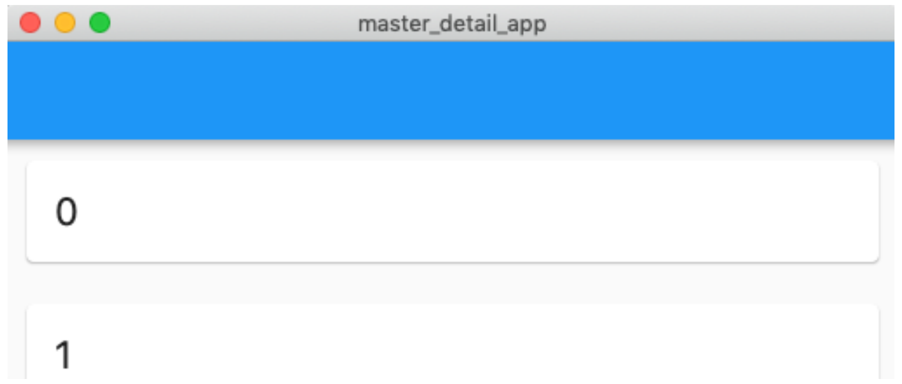
```
class _ListWidgetState extends State<ListWidget> {
  @override
  Widget build(BuildContext context) {
    return ListView.builder(
      itemCount: widget.count,
      itemBuilder: (context, position) {
        return Padding(
          padding: const EdgeInsets.all(8.0),
          child: Card(
            child: InkWell(
              onTap: () {
                widget.onItemSelected(position);
              },
              child: Row(
                children: <Widget>[
                  Padding(
                    padding: const EdgeInsets.all(16.0),
                    child: Text(position.toString(), style: TextStyle(fontSize: 22.0),),
                  ),
                ],
              ),
            ),
          ),
        );
      },
    );
  }
}
```



```
import 'package:flutter/material.dart';
import 'DetailPage.dart';
import 'DetailWidget.dart';
import 'ListWidget.dart';

class MasterDetailPage extends StatefulWidget {
  @override
  _MasterDetailPageState createState() => _MasterDetailPageState();
}
```

```
class _MasterDetailPageState extends State<MasterDetailPage> {
  var selectedValue = 0;
  var isLargeScreen = false;

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: OrientationBuilder(builder: (context, orientation) {
        if (MediaQuery.of(context).size.width > 600) {
          isLargeScreen = true;
        } else {
          isLargeScreen = false;
        }
        return Row(children: <Widget>[
          Expanded(
            child: ListWidget(10, (value) {
              if (isLargeScreen) {
                selectedValue = value;
                setState(() {});
              } else {
                Navigator.push(context, MaterialPageRoute(
                  builder: (context) {
                    return DetailPage(value);
                  },
                ));
              }
            }),
          ),
          isLargeScreen ? Expanded(child: DetailWidget(selectedValue)) : Container(),
        ]);
      }),
    );
  }
}
```





Some Desktop Plugins

url_launcher

shared_preferences
- Web only

path_provider
- Desktop only

file_chooser

menubar

url_launcher

```
class MyApp extends StatelessWidget {
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        body: Center(
          child: RaisedButton(
            onPressed: _launchURL,
            child: Text('Show Flutter homepage'),
          ),
        ),
      ),
    );
  }
}

_launchURL() async {
  const url = 'https://flutter.dev';
  if (await canLaunch(url)) {
    await launch(url);
  } else {
    throw 'Could not launch $url';
  }
}
```

Desktop & Web

Opens in new Browser window

```
dependencies:
  flutter:
    sdk: flutter
  url_launcher: ^5.7.2
```

Menubar

pub spec.yaml

```
menubar:
  git:
    url: https://github.com/google/flutter-desktop-embedding.git
    path: plugins/menubar
```

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

```
menubar.setApplicationMenu([
  menubar.Submenu(label: 'Search', children: [
    menubar.MenuItem(
      label: 'Search ...',
      onClicked: () {
        showDialog<void>(
          context: context,
          builder: (context) =>
              PhotoSearchDialog(photoSearchModel.addSearch),
        );
      },
    ),
  ])
```

File Chooser

```yaml
file_chooser:
  git:
    url: https://github.com/google/flutter-desktop-embedding.git
    path: plugins/file_chooser
```

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

final result = await file_chooser.showSavePanel(
  suggestedFileName: '${photo.id}.jpg',
  allowedFileTypes: const [
    file_chooser.FileTypeFilterGroup(
      label: 'JPGs',
      fileExtensions: ['jpg'],
    )
  ],
);
```

Flutter in Existing Android & iOS Apps

Flutter Engine runs on device

    Runs flutter "app"

Startup costs

Accessing Flutter UI costs

Pre-Warming Costs

	Android	iOS
Memory	42 MB	22 MB
Warming Time	1530 ms	860 ms

First Frame Cost

	Android	iOS
Memory	12 MB	16 MB
Time	320 ms	200 ms

Adding Flutter to Existing Android Project

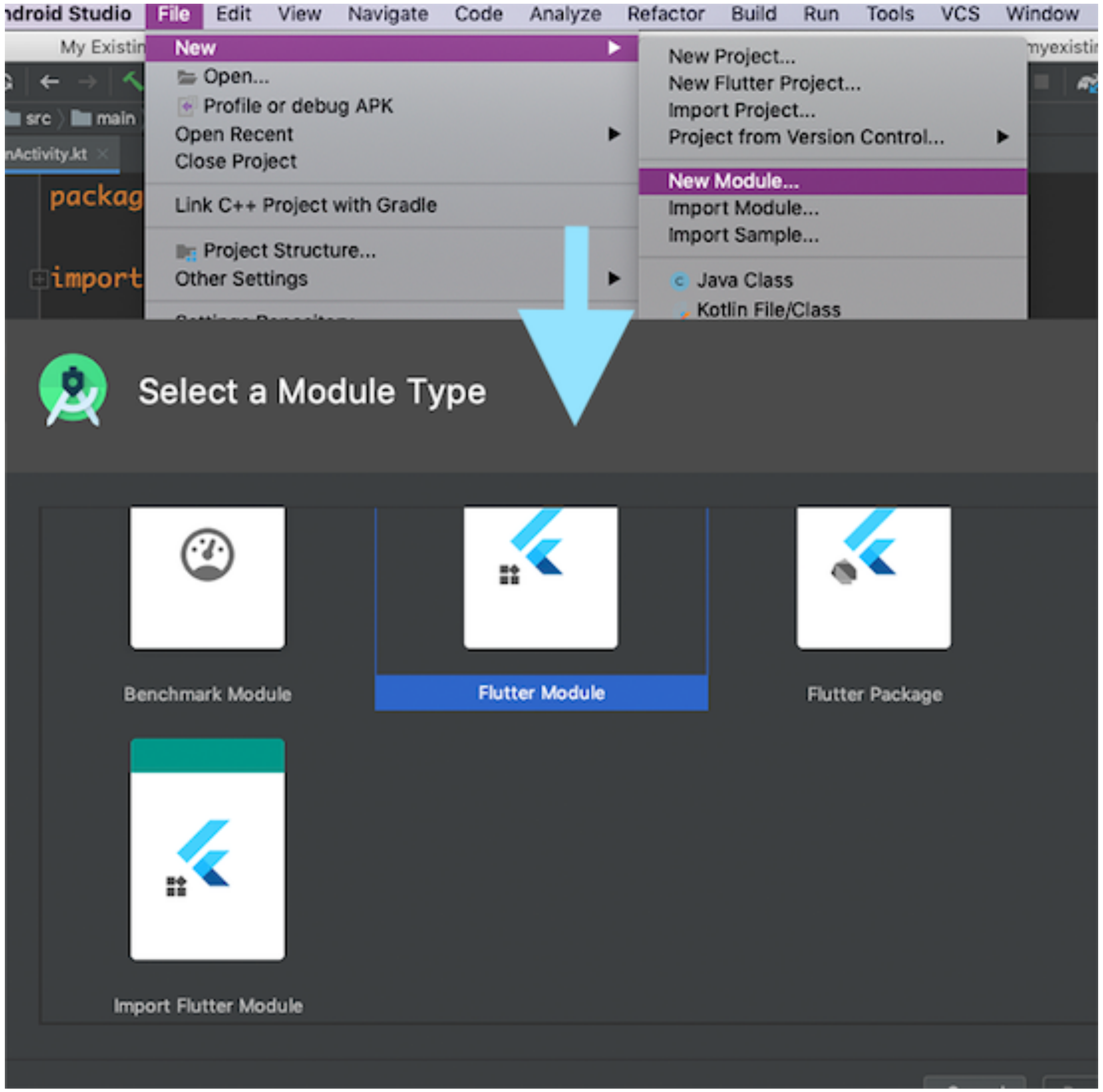
Flutter can be added to existing Android or iOS projects

Android runs on more CPUs that flutter

Restrict the CPUs types in Gradle file

```
android {
  //...
  defaultConfig {
    ndk {
      // Filter for architectures supported by Flutter.
      abiFilters 'armeabi-v7a', 'arm64-v8a', 'x86_64'
    }
  }
}
```

Add New Module in Android Project



Create Flutter app

main()

Routes

Add Flutter Activity to Android Manifest

```
<activity
  android:name="io.flutter.embedding.android.FlutterActivity"
  android:theme="@style/LaunchTheme"
  android:configChanges="orientation|keyboardHidden|keyboard|screenSize|locale|layoutDirection|
fontScale|screenLayout|density|uiMode"
  android:hardwareAccelerated="true"
  android:windowSoftInputMode="adjustResize"
  />
```

Calling the Code

```
myButton.setOnClickListener(new OnClickListener() {
  @Override
  public void onClick(View v) {
    startActivity(
      FlutterActivity.createDefaultIntent(currentActivity)
    );
  }
});
```