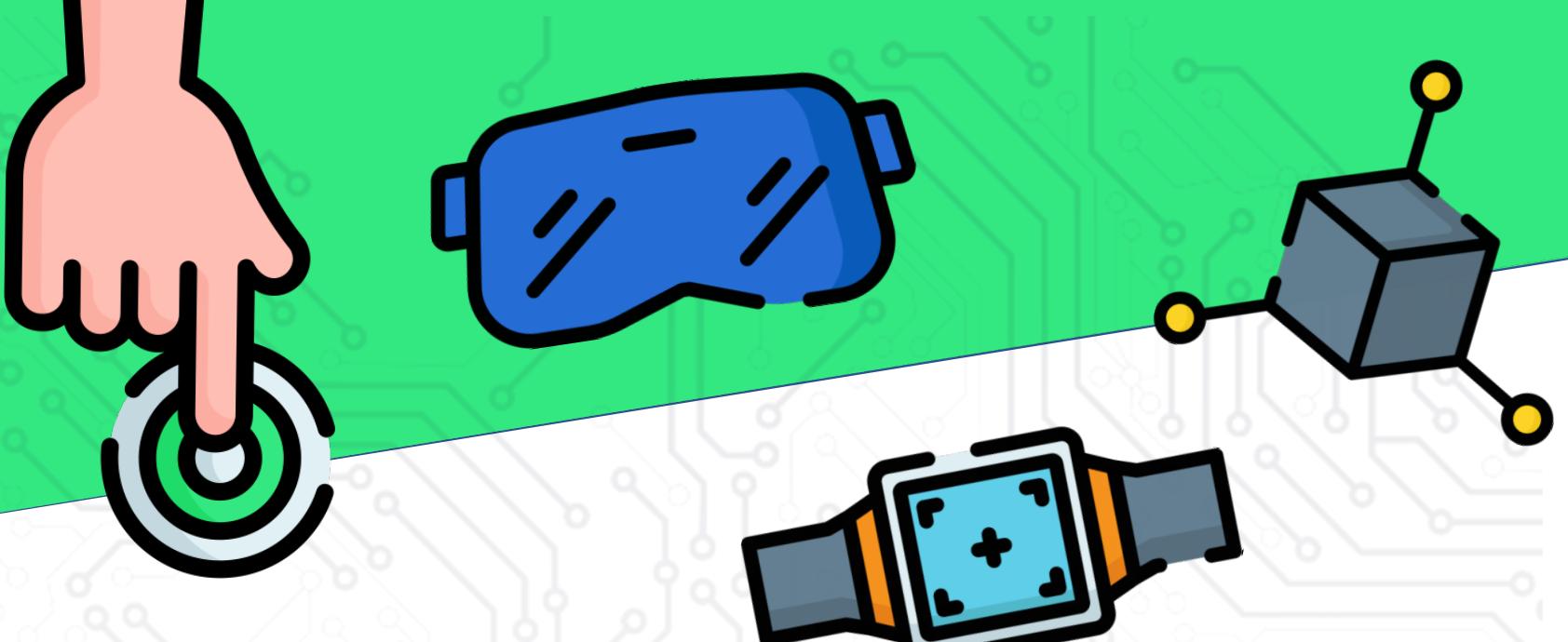


**CLIENT  
TECHNOLOGY  
DAYS**



# Flutter Workshop

Denise & Dominic & Oliver & Stefan



# Speakers



**Oliver Gepp**

Lead Software Architect  
Focus iOS  
FG Mobile Camp &  
Education



**Denise Scherzinger**

Professional Software  
Engineer  
Focus Mobile  
FG Mobile Camp &  
Education



**Stefan Diegas**

Professional Software  
Engineer  
Focus Mobile  
FG Mobile Camp



**Dominic Mülhaupt**

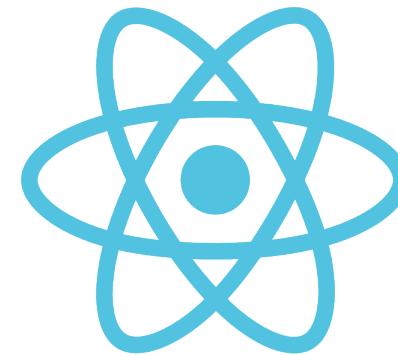
Advanced Software  
Engineer  
Focus Mobile  
FG Mobile Camp

# Agenda

- Flutter Introduction
- Dart
- Widgets
- Break 
- Navigation
- State Management
- REST



Flutter



React Native



ionic



Xamarin



NativeScript



PhoneGap

# What is Flutter?

- Native Apps for iOS & Android
- Web & Desktop are coming
- Open-Source, backed by Google
- Alpha Release: Mid 2017
- Stable Release: End 2018



# What is Dart?



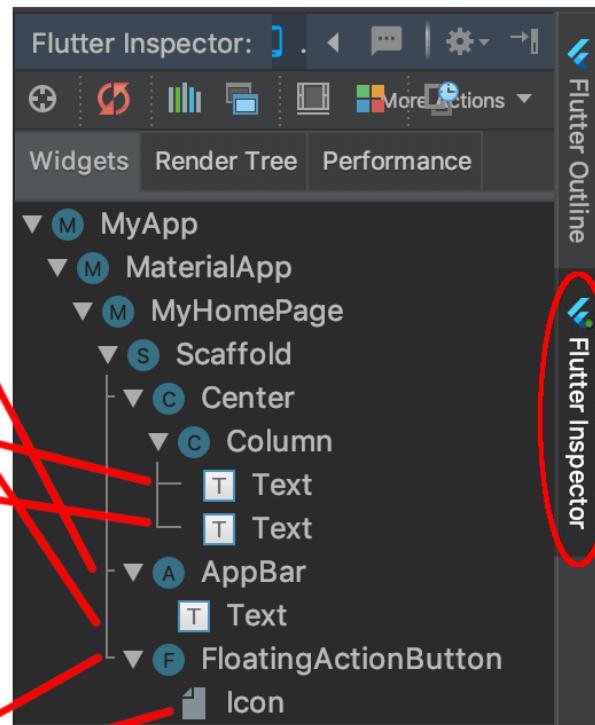
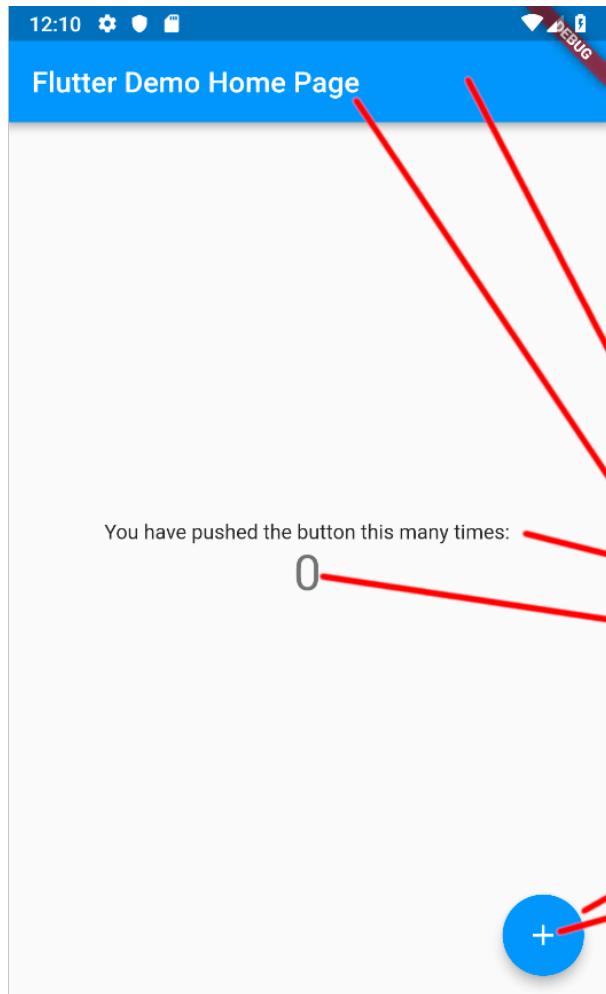
# Dart

- Object-oriented
- Optimized for UI
- Open-Source, backed by Google
- Supports JIT & AOT compilation

How do I build the interface?

```
1 @override
2 Widget build(BuildContext context) {
3     return MaterialApp(
4         home: Scaffold(
5             appBar: AppBar(
6                 title: Text("Exploring Widgets"),
7             ),
8             body: myWidget(),
9         ),
10    );
11 }
```

Everything is a Widget



# Hot Reloading!



# How is Flutter native?

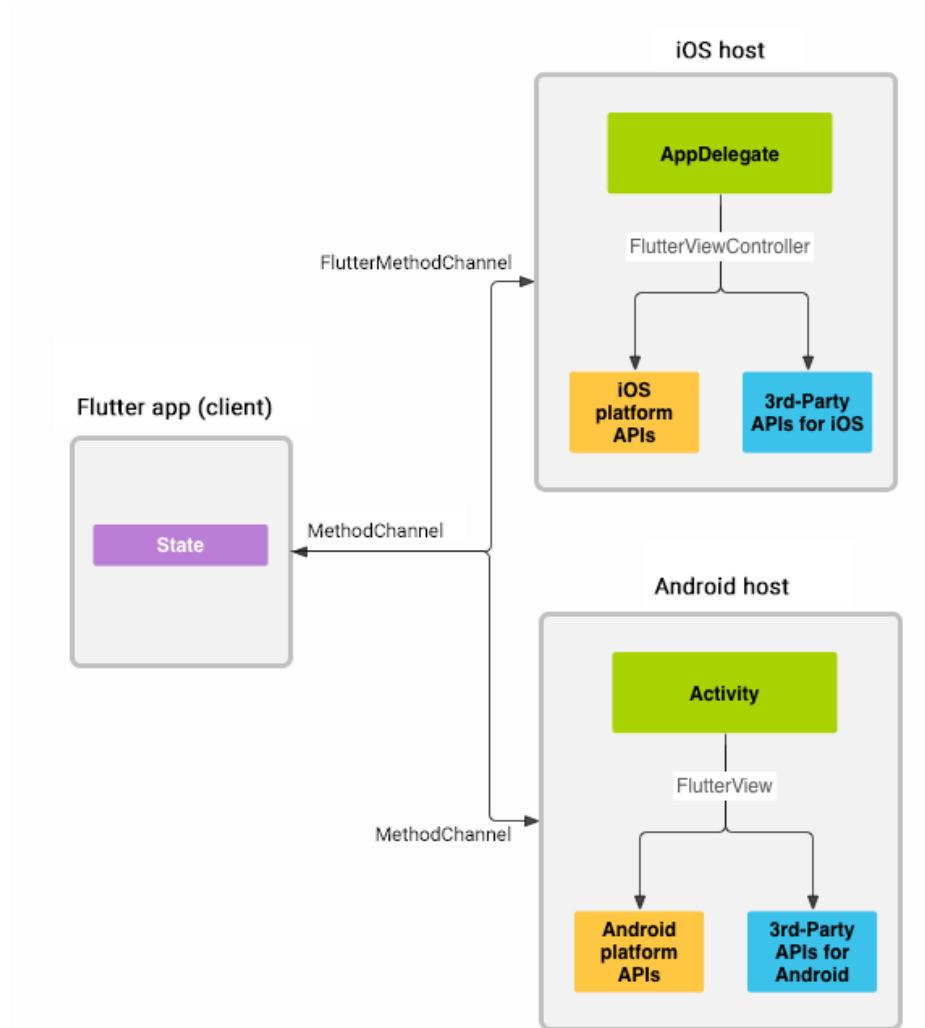
# UI

- Rendering using Skia
- No OEM Widgets
  - Look is independent of OS version

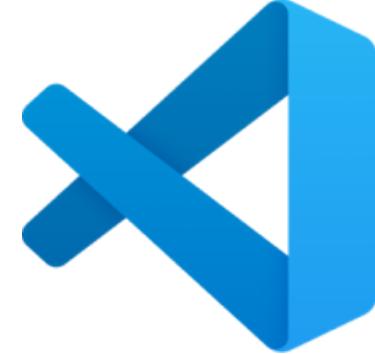
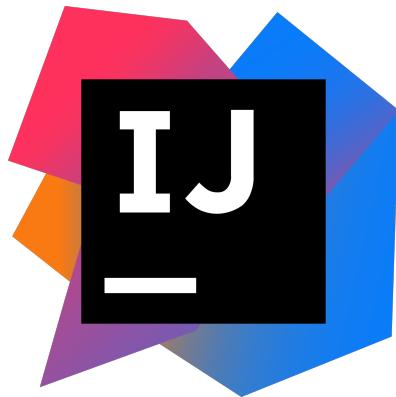


# Services

- Basic [API](#) provided by Flutter
- Integrating APIs is easy
  - Client sends a message
  - Host listens for messages and executes native code



What tools can I use?



Flutter Dev Tools



Dart Editor + flutter CLI

Is Flutter production-ready?

# Yes!

But...

- Ecosystem smaller, however growing fast
- Dart is new to most developers
- Some cute and some nasty bugs
- New minor version every ~1 month

# Showcase



## Alibaba

Alibaba, the world's biggest online commerce company, used Flutter to create a beautiful app experience for iOS and Android on their Xianyu app, which has 50M+ downloads.

[Download ▾](#)



## Google Ads

The Google Ads app helps you keep your ad campaigns running smoothly – no matter where your business takes you.

[Download ▾](#)



## AppTree

AppTree provides an enterprise app platform for brands like McDonalds, Stanford, Wayfair & Fermilab.

[Developer website](#) [Download ▾](#)



## Reflectly

A beautiful journal and mindfulness app driven by artificial intelligence.

[Learn more](#) [Download ▾](#)



## Hamilton Musical

Official app of the hit Broadway musical, Hamilton. Includes daily lotteries, exclusive news and videos, a trivia game, merchandise store, and more.

[Learn more](#) [Download ▾](#)



## Google Greentea

An internal customer management app used widely by Google sales that provides rich visualizations for tracking sales targets.



## Abbey Road Studios

Abbey Road's Topline app helps artists record songs. You can share files, sing over imported tracks, add lyrics and more.

[Website](#) [Download ▾](#)

## Tencent 腾讯

## Tencent

NOW Live is Tencent's video streaming service with tens of millions of MAU.

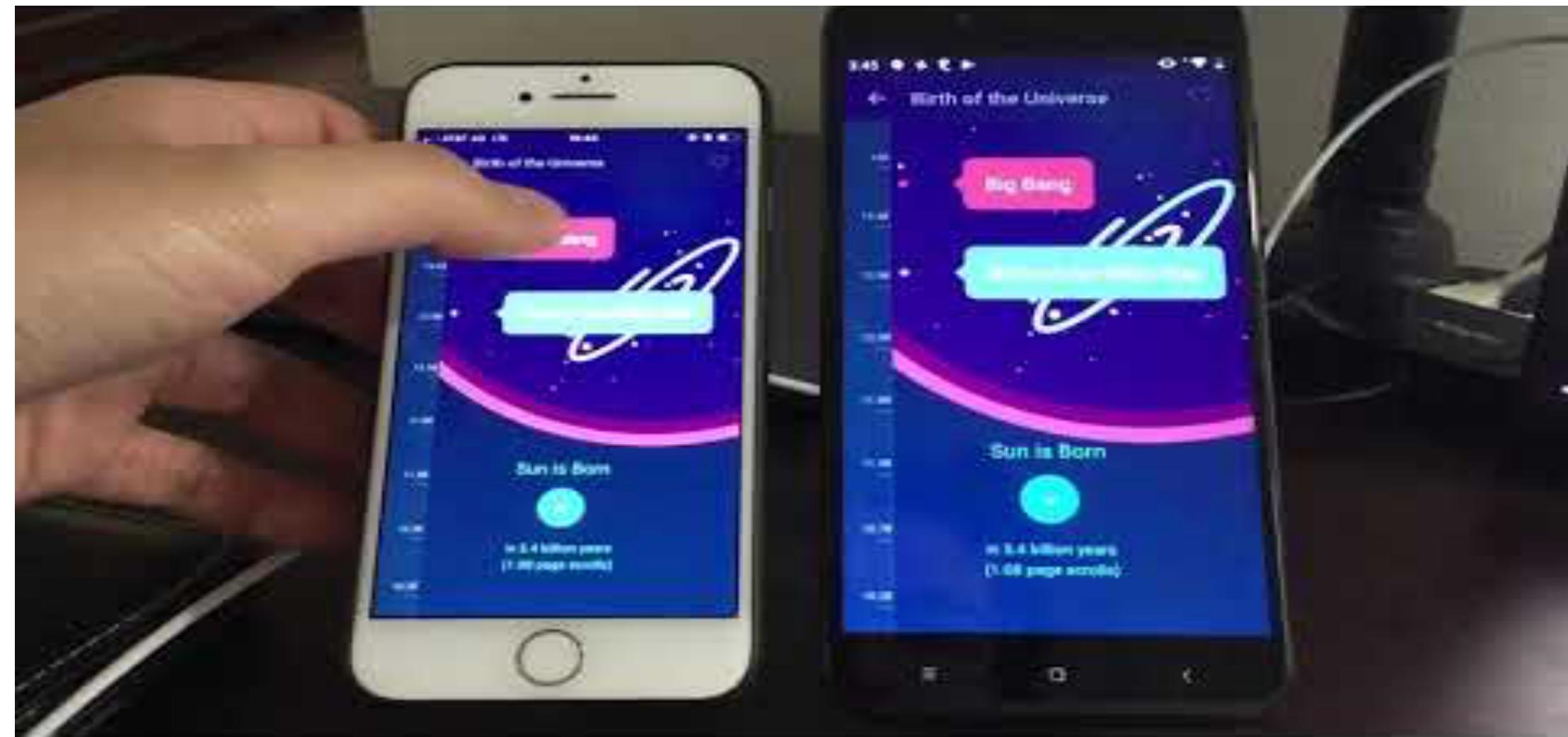
[Developer website](#) [Download ▾](#)



## JD Finance

JD Finance is a leading digital technology company. The company covers fintech, digital enterprise services and urban computing.

[Developer website](#) [Download ▾](#)



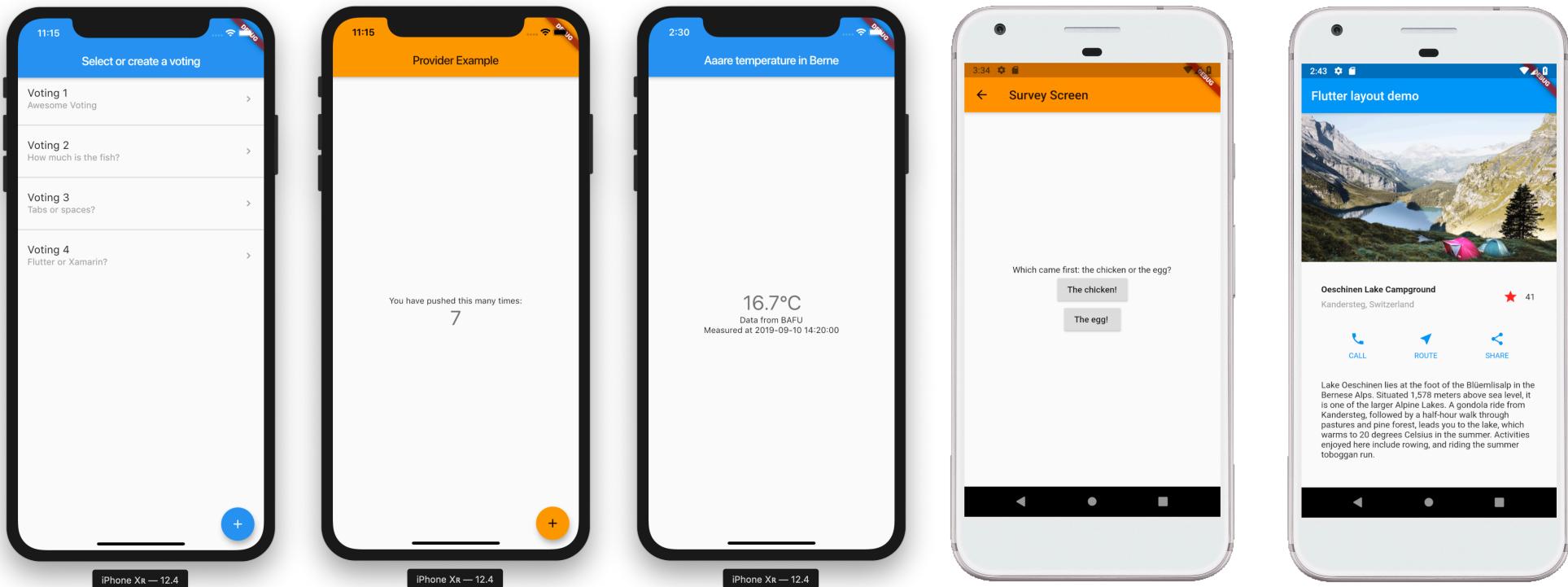
# The History of Everything



When can we actually start coding?

# Hands-On

Git Repo: [https://github.com/dominicmh/flutter\\_workshop/](https://github.com/dominicmh/flutter_workshop/)



Dart

# Dart

- Dart SDK comes with standalone Dart VM

```
$ dart main.dart
```

- Package manager: Pub (<https://pub.dev/>)
- DartPad: run Dart code in browser <https://dartpad.dartlang.org/>

# Dart

- Strongly typed language, supports type inference
- All data types are objects and inherit from *Object* class  
→ default value is *null*
- Only single inheritance, but multiple interfaces can be implemented
- Abstraction possible with abstract classes and interfaces

# Dart

- Entry point: main() method
- No private/public keywords, \_ is used instead
- String interpolation and string concatenation with adjacent string literals

```
1 class Tweet {  
2     String _username;  
3     String _content;  
4  
5     Tweet(this._username, this._content);  
6  
7     printTweet() {  
8         print('--- Tweet ---'  
9             '\n Username: ${_username}'  
10            '\n Content: ${_content}');  
11    }  
12 }  
13  
14 main() {  
15     Tweet tweet = new Tweet("trumpet", "I am great!");  
16     tweet.printTweet();  
17 }  
18
```

# Dart

- Shorthand constructor without body
- Named constructors can be used to implement multiple constructors
  - provides namespace to constructor instead of parameter count

```
1 class Point {  
2   num x, y;  
3  
4   // Shorthand constructor  
5   Point(this.x, this.y);  
6  
7   // Named constructor  
8   Point.origin() {  
9     x = 0;  
10    y = 0;  
11  }  
12 }
```

# Dart

- Cascade notation (..) can be used to make a sequence of operations on the same object (method calls, accessing fields, ... )
- It invokes the method, discards the result and instead returns the instance each time

```
1 querySelector('#confirm')
2   ..text = 'Confirm'
3   ..classes.add('important')
4   ..onClick.listen((e) => window.alert('Confirmed!'));
```



```
1 var button = querySelector('#confirm');
2 button.text = 'Confirm';
3 button.classes.add('important');
4 button.onClick.listen((e) => window.alert('Confirmed!'));
```



# Dart

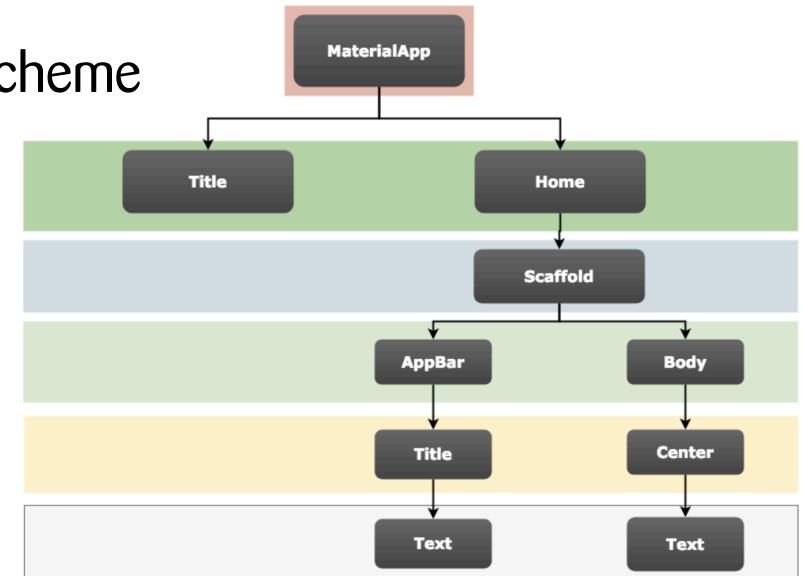
- Optional parameters (either positional or named)

```
1 // named optional parameters, with @required annotation
2 Scrollbar({Key key, @required Widget child})
3
4 // positional optional parameters, with default value
5 readFile(String name, [String mode, String charset = 'utf-8']) {...}
```

# Widgets

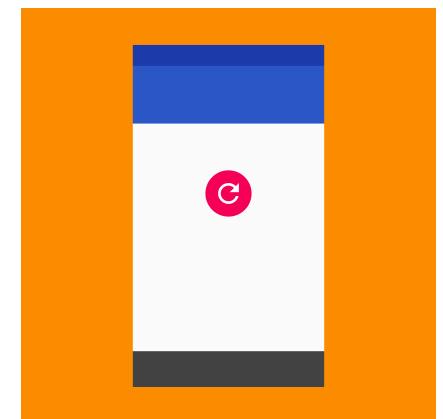
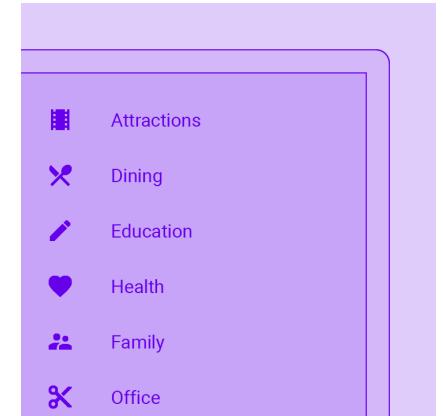
# Widgets

- Widgets are basic building blocks of a flutter app
- Each widget is a mutable declaration of part of user interface
- Everything is a widget
  - Structural elements like button, menu
  - Stylistic elements that propagate a font or color scheme
  - Layout related elements like padding
  - ...
  - Or a composition of existing widgets

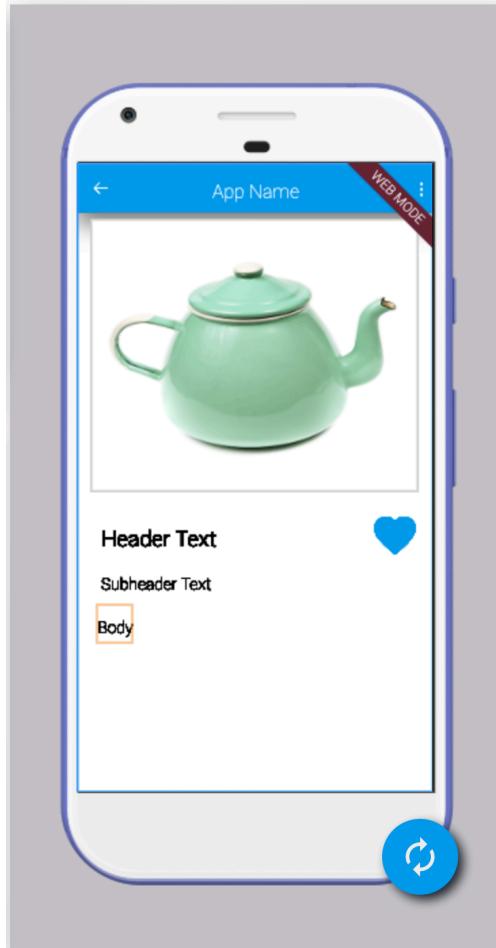


# Catalogue

- There are Widgets for almost everything
  - Accessibility
  - Animation and Motion
  - Assets, Images and Icons
  - Async
  - Input
  - Interaction Models
  - Layouts
  - Painting
  - Scrolling
  - Styling
  - And more ...
- <https://flutter.dev/docs/development/ui/widgets>



# Exercise 1



- Navigate to <http://mutisya.com>
- By drag and drop try to create this News App on the left
- Have a look at the generated Code



Time: 10 minutes

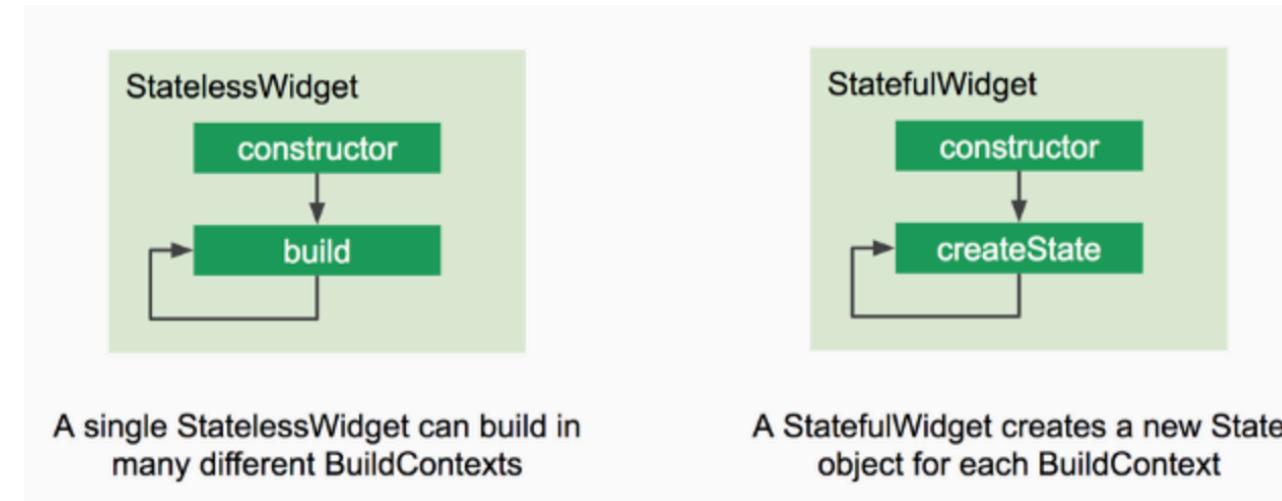
# Solution

```
override  
Widget build(BuildContext context) {  
  return new Scaffold(  
    appBar: new AppBar(  
      title: new Text('App Name'),  
    ),  
    body:  
    new Center(  
      child:  
      new Column(  
        mainAxisAlignment: MainAxisAlignment.start,  
        mainAxisSize: MainAxisSize.min,  
        crossAxisAlignment: CrossAxisAlignment.center,  
        children: <Widget>[  
          new Image.asset(  
            'images/name.jpg',  
            width: 240.0,  
            height: 200.0,  
          ),  
  
          new Expanded(  
            child:  
            new Column(  
              mainAxisAlignment: MainAxisAlignment.start,  
              mainAxisSize: MainAxisSize.min,  
              crossAxisAlignment: CrossAxisAlignment.center,  
              children: <Widget>[  
                new Row(  
                  mainAxisAlignment: MainAxisAlignment.start,  
                  mainAxisSize: MainAxisSize.min,  
                  crossAxisAlignment: CrossAxisAlignment.baseline,  
                  children: <Widget>[  
                    new Expanded(  
                      child:  
                      new Column(  
                        mainAxisAlignment: MainAxisAlignment.start,  
                        mainAxisSize: MainAxisSize.min,  
                        crossAxisAlignment: CrossAxisAlignment.start,  
                        children: <Widget>[  
                          new Text(  
                            "Header Text",  
                            style: new TextStyle(fontSize:16.0,  
                                color: const Color(0xFF000000),  
                                fontWeight: FontWeight.w300,  
                                fontFamily: "Roboto")  
                        ),  
                      ],  
                    ],  
                  ],  
                ],  
              ],  
            ],  
          ],  
        ],  
      ),  
    ),  
  );  
}
```

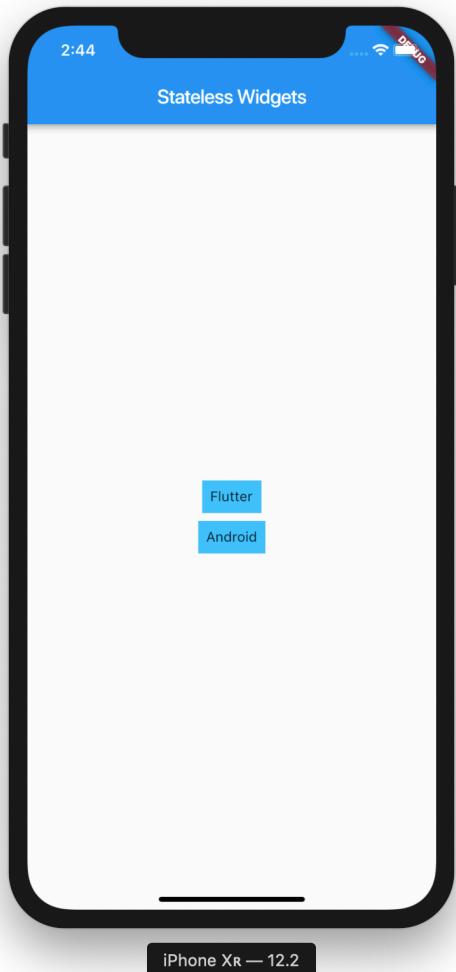
- Many Column and Row Widgets -> important in Flutter
- Uncommon structure of the Code

# Stateless Widgets

- A stateless widget is a widget that is composed of children and doesn't contain any mutual state
- The *build(...)* function is only called once
- Can't track data over time, or trigger rebuild on their own



# Example



```
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(new DemoApp());
5 }
6
7 class DemoApp extends StatelessWidget {
8   @override
9   Widget build(BuildContext context) {
10     return MaterialApp(
11       title: 'Learning Flutter',
12       home: Scaffold(
13         appBar: AppBar(
14           title: Text('Stateless Widgets'),
15         ),
16         body: Center(
17           child: Column(
18             mainAxisAlignment: MainAxisAlignment.center,
19             children: [
20               DecoratedBox(
21                 decoration: BoxDecoration(color: Colors.lightBlueAccent),
22                 child: Padding(
23                   padding: const EdgeInsets.all(8.0),
24                   child: Text('Flutter'),
25                 ),
26               ),
27               SizedBox(height: 8.0),
28               DecoratedBox(
29                 decoration: BoxDecoration(color: Colors.lightBlueAccent),
30                 child: Padding(
31                   padding: const EdgeInsets.all(8.0),
32                   child: Text('Android'),
33                 ),
34               ),
35             ],
36           ),
37         ),
38       );
39     }
40   }
41 }
```

# Exercise 2

- Read the code from last page carefully and understand its concept.
- Eliminate duplicated code and create a new Stateless Widget "TechnologyName" that takes two arguments for name of technology and background color.
- Time is limited, so only build it in your head ☺.
- Compare your solution with the one on the next page



Time: 5 minutes

# Solution:

```
1 import 'package:flutter/material.dart';
2
3 class TechnologyName extends StatelessWidget {
4   final String name;
5   final MaterialAccentColor backgroundColor;
6
7   const TechnologyName(this.name, this.backgroundColor);
8
9   @override
10  Widget build(BuildContext context) {
11    return DecoratedBox(
12      decoration: BoxDecoration(color: backgroundColor),
13      child: Padding(
14        padding: const EdgeInsets.all(8.0),
15        child: Text(name),
16      ),
17    );
18  }
19 }
```

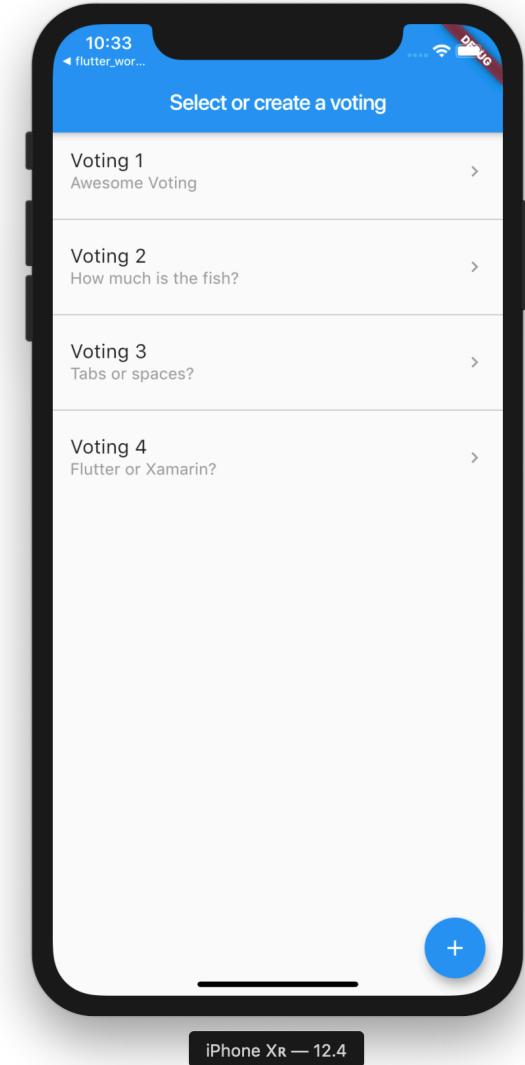
```
1 import 'package:flutter/material.dart';
2 import 'technology_name.dart';
3
4 void main() {
5   runApp(new DemoApp());
6 }
7
8 class DemoApp extends StatelessWidget {
9   @override
10  Widget build(BuildContext context) {
11    return MaterialApp(
12      title: 'Learning Flutter',
13      home: Scaffold(
14        appBar: AppBar(
15          title: Text('Stateless Widgets'),
16        ),
17        body: Center(
18          child: Column(
19            mainAxisAlignment: MainAxisAlignment.center,
20            children: [
21              TechnologyName('Flutter', Colors.orangeAccent),
22              SizedBox(height: 8.0),
23              TechnologyName('Android', Colors.blueAccent)
24            ],
25          ),
26        ),
27      ),
28    );
29  }
30 }
```

# Exercise 3

- Build the UI on the right ☺
  - Open project from ListView folder (Task) in our repo
  - Add a [FloatingActionButton](#) and call existing function on tap
  - Add VotingListPage widget that is already part of the project as the body of the Scaffold widget
  - Add missing code in VotingListPage widget. Data is provided by existing VotingModel class
  - Solution can be found [here](#)



Time: 15 minutes



# Stateful Widgets

- A stateful widget is a widget that is composed of children and has a mutual state
  - Mutual state: Properties that change over time like the string value of a textfield.
- Can track data over time, or trigger rebuild on their own
- When the widget's state changes, the state object calls `setState(...)`, telling the framework to redraw the widget.

# Basic Syntax

```
class FavoriteWidget extends StatefulWidget {
  @override
  _FavoriteWidgetState createState() => _FavoriteWidgetState();
}

class _FavoriteWidgetState extends State<FavoriteWidget> {
  bool _isFavorite = true;

  void _toggleFavorite() {
  }

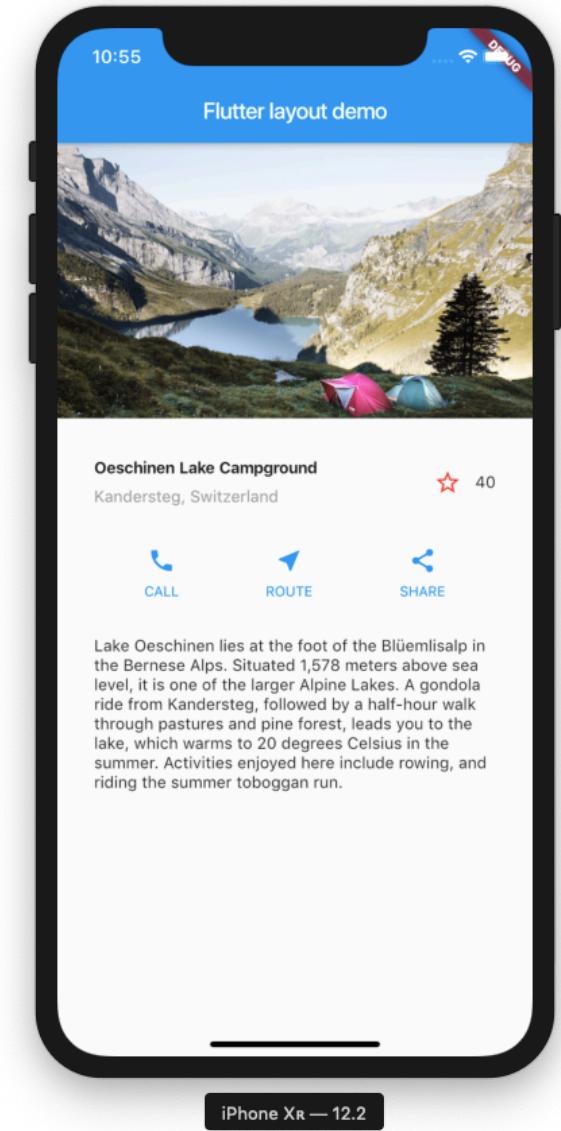
  @override
  Widget build(BuildContext context) {
    return Row(
      );
  }
}
```

# Exercise 4

- Replacing Stateless Widgets
  - Checkout code [here](#)
  - Replace two stateless widgets – the solid red star and the counter next to it with one stateful widget
  - First of all create a subclass of StatefulWidget that defines the Widget
  - Then create a Subclass of State
  - Solution can be found [here](#)



Time: 15 minutes



# Navigation

# Navigation

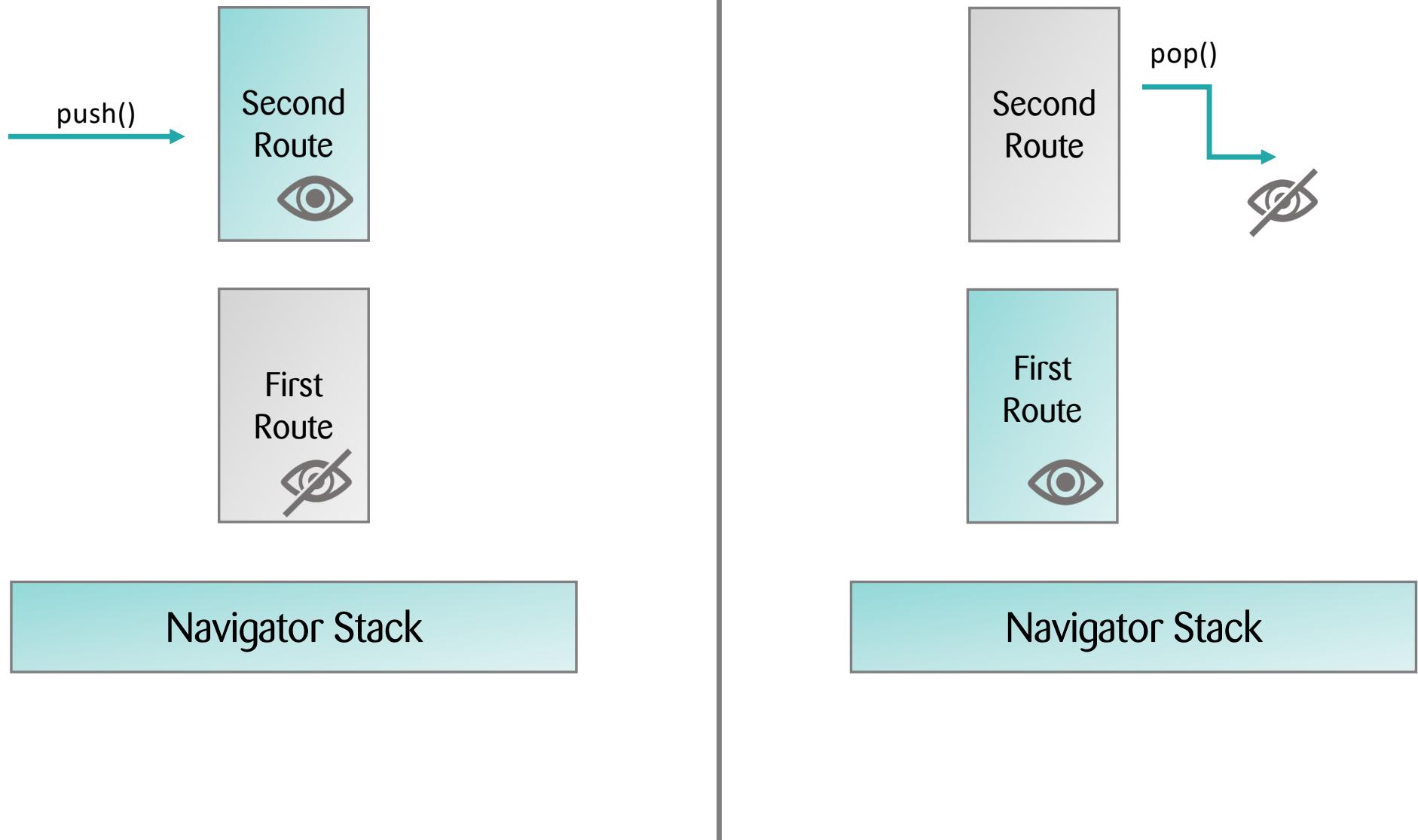
- Screens and pages are called *Routes*
- Managed using *Navigator* widget
- *Navigator* handles stack (*Overlay*) of *Route* objects and provides methods for managing the stack
- Most recent pages are visually placed on top of older ones



Activity



iOS ViewController

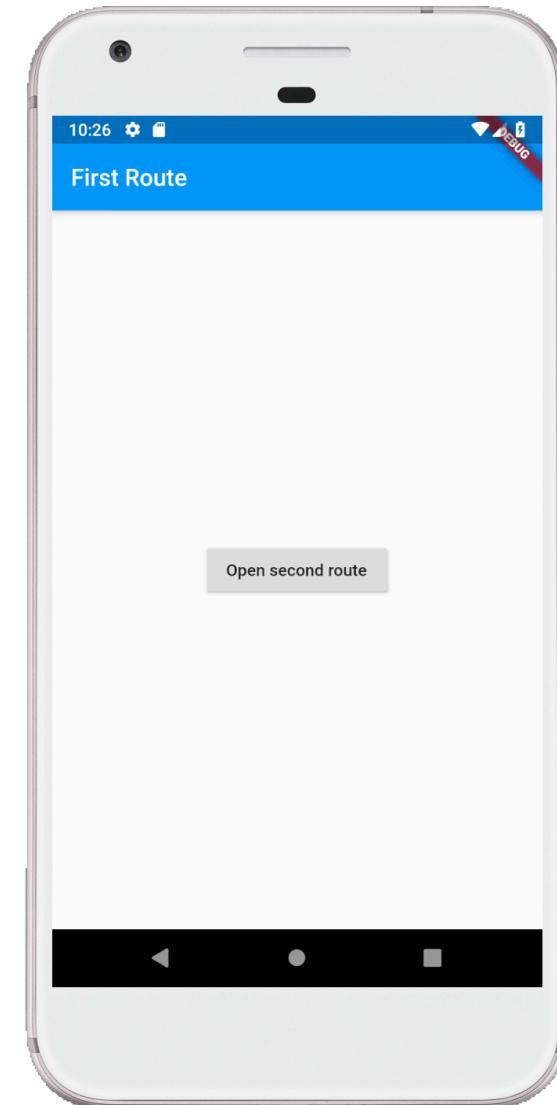


# Exercise 5

- Go to Navigation/Exercise1/Task
- Use the Navigator widget to switch between FirstRoute and SecondRoute



Time: 5 minutes



# Navigation

---

```
1 Navigator.push(  
2     context,  
3     MaterialPageRoute(builder: (context) => SecondRoute()),  
4 );
```

Problem: Duplicated code, routing logic distributed throughout code base

Solution: Using **named routes**, encapsulating routing logic

# Navigation

- Define route table (*routes* property) in `MaterialApp`
- Set *initialRoute* instead of *home*
- Navigate to routes using `Navigator.pushNamed()`

```
1 MaterialApp(  
2   initialRoute: '/',  
3   routes: {  
4     // When navigating to the "/" route, build the FirstScreen widget.  
5     '/': (context) => FirstScreen(),  
6     // When navigating to the "/second" route, build the SecondScreen widget.  
7     '/second': (context) => SecondScreen(),  
8   },  
9 );
```

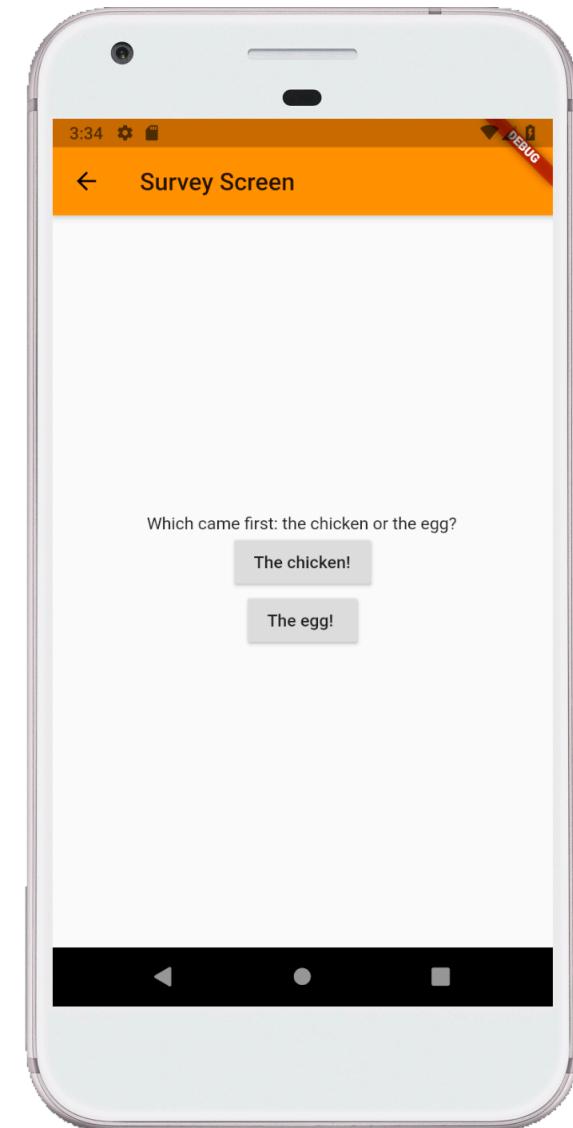
```
1 // Within the 'FirstScreen' widget  
2 onPressed: () {  
3   Navigator.pushNamed(context, '/second');  
4 }
```

# Exercise 6

- Go to Navigation/Exercise2/Task
- Use named routes to navigate from HomeScreen to SurveyScreen
- Display result from SurveyScreen in Dialog

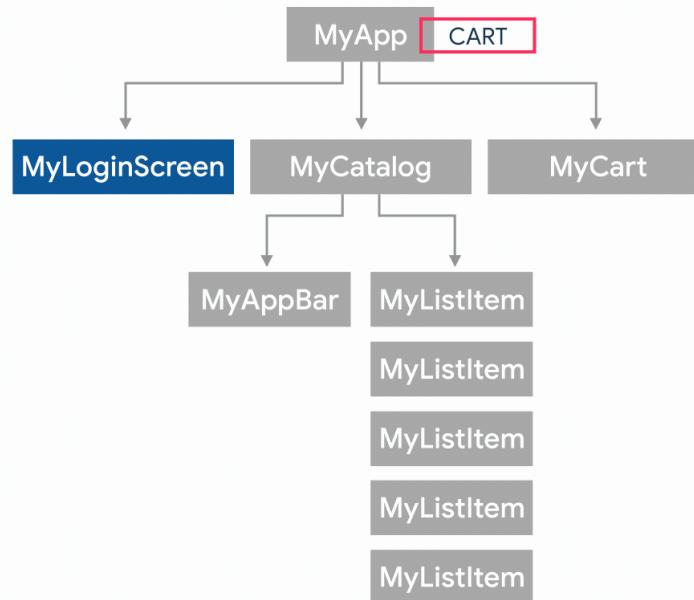
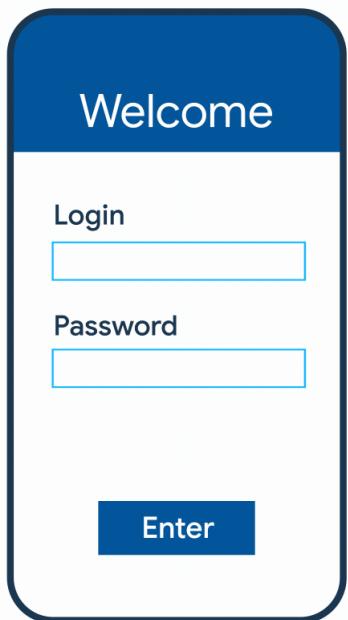


Time: 15 minutes

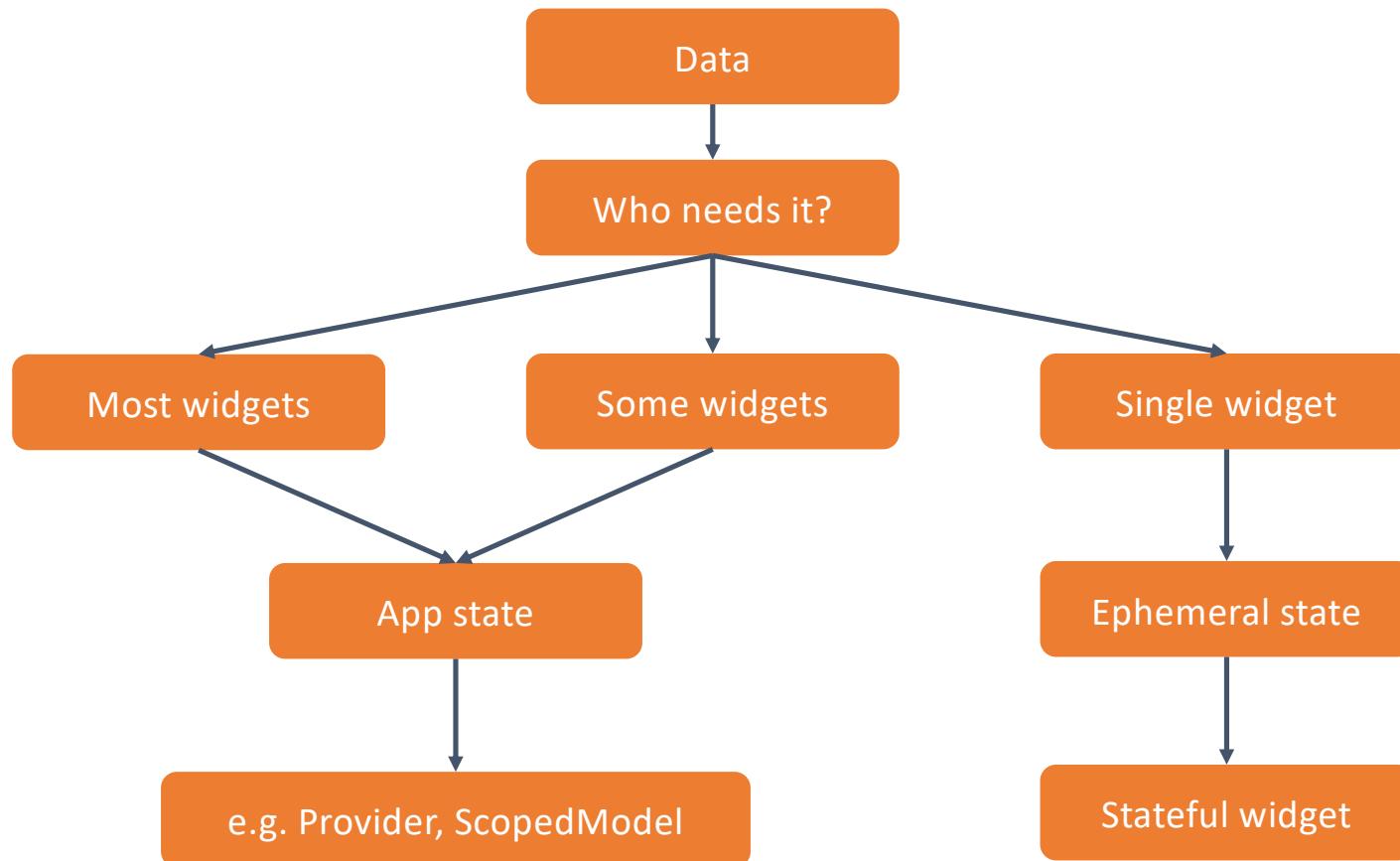


# State Management

# State Management



# App state versus widget state



# Provider

- A mixture between dependency injection and state management, built with widgets for widgets.
- With Provider you get
  - forced uni-directional data-flow
  - testability/composability
  - robustness, as it is harder to forget to handle the update scenario of a model/widget
- <https://pub.dev/packages/provider>



# Exercise 7

- Read [article](#) in flutter documentation about state management
- Change flutter project in folder *Statemanagement* from a stateful widget to a stateless widget that uses a provider
  - See existing code for further details



Time: 20 minutes

**REST**

# HTTP Theory

- Most apps are fetching data from the internet, luckily Flutter provides a http package 😊

# HTTP Basics 1

- To use HTTP Calls we need to import the http package

```
dependencies:  
    http: <latest_version>
```

- Make a network Request

```
Future<http.Response> fetchComment() {  
    return http.get('https://jsonplaceholder.typicode.com/comments/1');  
}
```

# HTTP Basics 2

- Create a Comment class

```
class Comment {  
    final int userId;  
    final int id;  
    final String title;  
    final String body;  
  
    Comment({this.userId, this.id, this.title, this.body});  
  
    factory Comment.fromJson(Map<String, dynamic> json) {  
        return Comment(  
            userId: json['userId'],  
            id: json['id'],  
            title: json['title'],  
            body: json['body'],  
        );  
    }  
}
```

# HTTP Basics 3

- Convert the HTTP Response to a Comment Object

```
Future<Comment> fetchComment() async {
    final response =
        await http.get('https://jsonplaceholder.typicode.com/comments/1');

    if (response.statusCode == 200) {
        // If server returns an OK response, parse the JSON.
        return Comment.fromJson(json.decode(response.body));
    } else {
        // If that response was not OK, throw an error.
        throw Exception('Failed to load Comment');
    }
}
```

# HTTP Basics 4

- Fetch the data and display it on screen, for example with FutureBuilder

```
FutureBuilder<Comment>(
    future: fetchComment(),
    builder: (context, snapshot) {
        if (snapshot.hasData) {
            return Text(snapshot.data.title);
        } else if (snapshot.hasError) {
            return Text("${snapshot.error}");
        }

        //By default, show a loading spinner.
        return CircularProgressIndicator();
    },
);
```

# Exercise 8

- Let's check the temperature of the Aare
  - Checkout code [here](#)
  - Check the AareService, there are multiple comments guiding you through the task
  - Solution can be found [here](#)



Time: 15 minutes

