Flutter

120

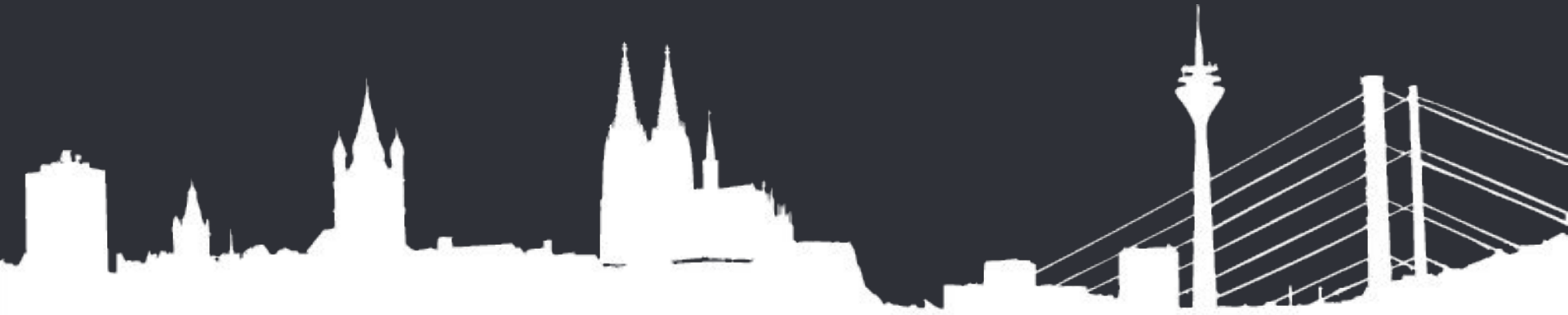
~~60~~ FPS UI of the Future

We are hiring!

grandcentrix.net
@grandcentrix

+Albrecht Noll
@UhrArt

+Pascal Welsch
@passsy



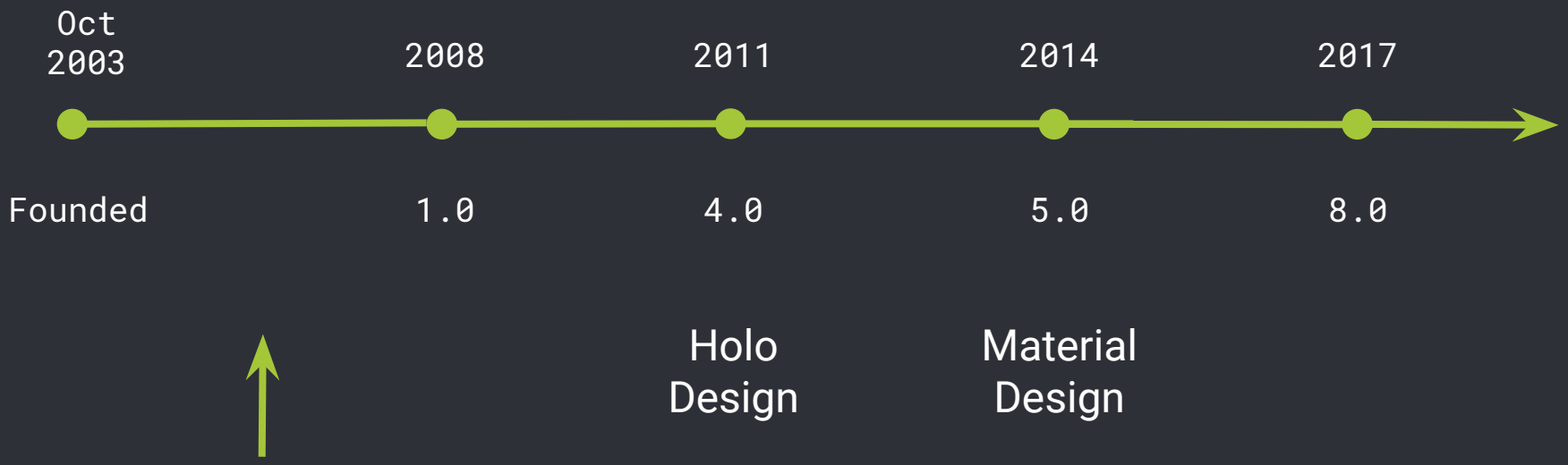
Flutter

Now in beta

Agenda

- Android - why UI sucks
- Flutter - Architecture
- Dart - History and Dart 2.0
- Flutter - UI (Widget)
- Flutter - OS connection (Platform channels)
- Fuchsia - Lookout
- Flutter - Live Coding

Android History



UI Bugfixes and Improvements

- Project Butter
- RecyclerView
- Design support library
- Instant Run
- Databinding in XML layouts
- Vector Drawables
- ...and thousands small fixes every release

My smartphone is lagging

- Every Android user ‘18

Android UI Framework

- >10 years old
- The Java API hasn’t seen major changes
- No architectural changes, we are still using `android.view` to render our UIs

- Feels old
  - XML still “best practice”
  - No virtual dom

The entire UI architecture is wrong from the start.

Erik Hellman
@ErikHellman

Flutter

Now in beta

What is Flutter?

- A multi-platform mobile app SDK for Android, iOS and Fuchsia
- Uses Dart - compiles to efficient ARM code
- Rich Widget catalog
- Modern, React inspired View-Framework

flutter.io

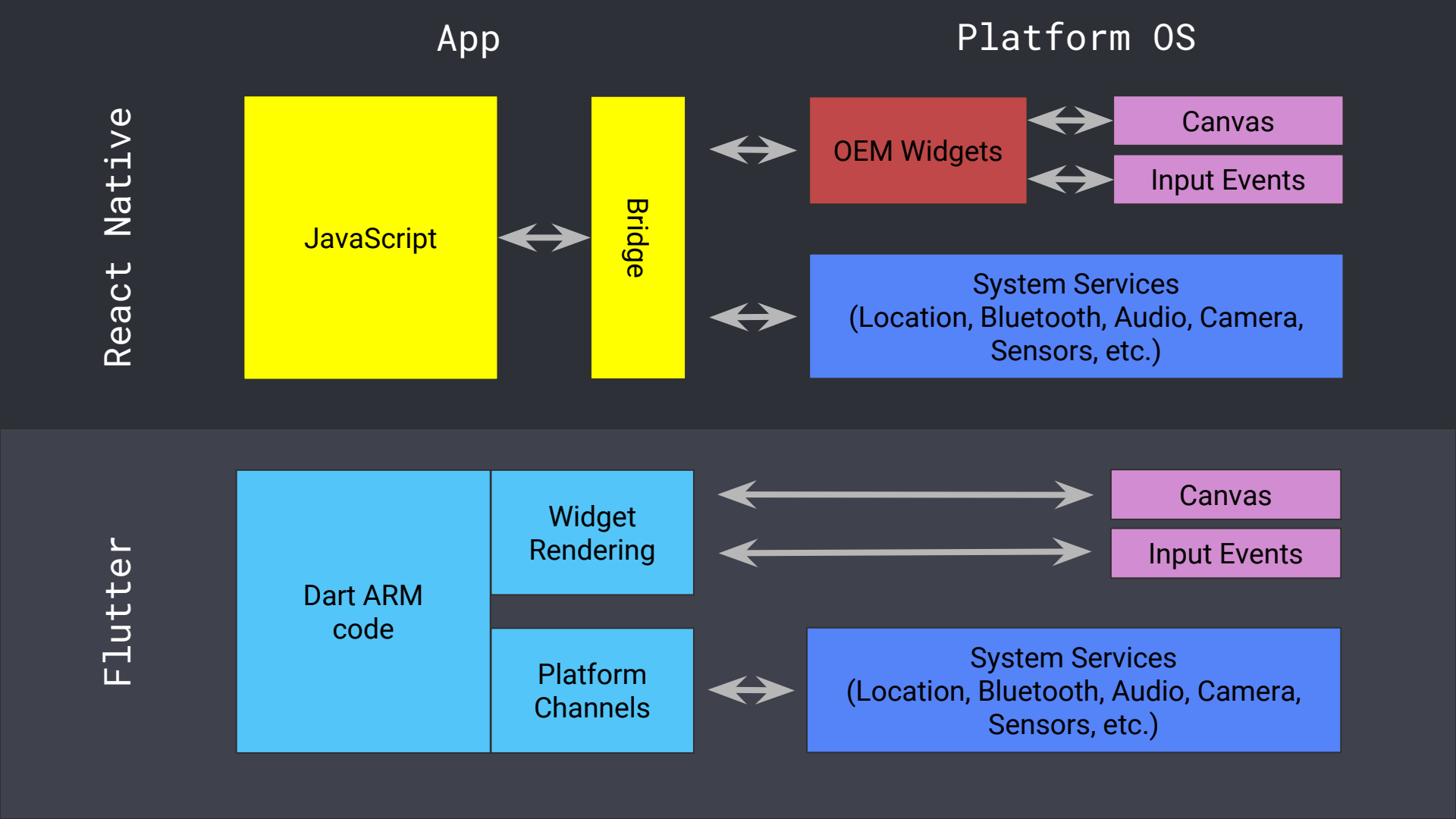
Flutters goals

- Beautiful fluid UIs
- high-performance apps that feel natural on different platforms
- Be productive
- Run same UI on multiple platforms, perfect for brand-first designs (optional)

flutter.io

Flutter is not yet another Cross-Platform SDK

- Engine is shipped in apk (∾7.5 Mb)
- Doesn’t use OEM widgets
- Ships SDK with the app, no fragmentation or compatibility issues
- Compared to React Native:
  No bridge needed, direct drawing to platform canvas


App
Platform OS
React Native
JavaScript
Bridge
OEM Widgets
Canvas
Input Events
System Services
(Location, Bluetooth, Audio, Camera,
Sensors, etc.)
Flutter
Dart ARM
code
Widget
Rendering
Platform
Channels
Canvas
Input Events
System Services
(Location, Bluetooth, Audio, Camera,
Sensors, etc.)

Dart?

dartlang.org

Dart History

No new language features

2011

Goal: Replace JavaScript in Chrome

2013

Dart 1.0

JIT only

2014

Dart will not land in Chrome!

-> **Move to AOT**
JavaScript compilation;
Introduce Strong mode;
async/await

2015

Roadmap: Dart 2.0 announced for early 2016

😂

Compile to ARM

Make Dart and tools multiplatform

Web, mobile, Dart VM

2018

Dart 2.0 dev preview

Dart 2.0 - a reboot

Become the best language for client-side code

- Leaf Petersen (DartConf '18)

Darts strengths?

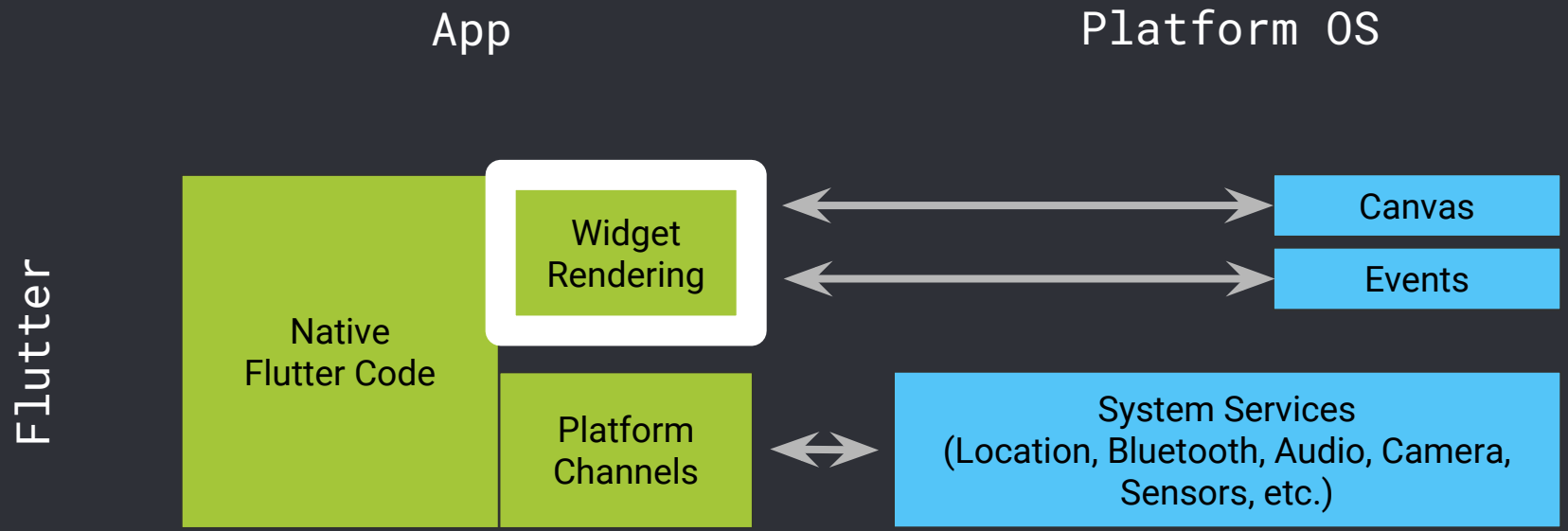
dartlang.org

- Java like language - easy to learn
- Supports JIT and AOT compilation
  - JIT -> fast development
  - AOT -> fast release builds
- Reactive - Futures and Streams are built-in
- Finally reached 2.0, syntactic sugar will follow soon™
  - optional “new”, Nullable types
- Compiles to native ARM code

Flutter

Rendering

Widget Rendering

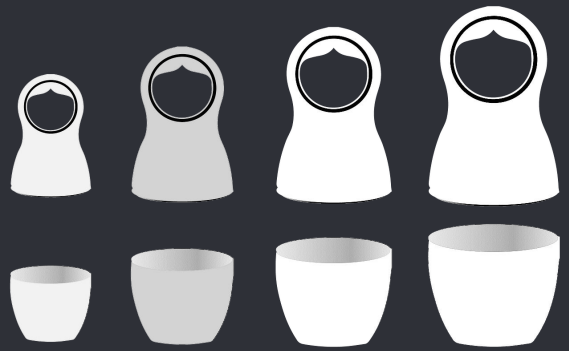


What are Widgets?

- Widgets are immutable declarations of parts of the UI
- Like a `<div/>`
- a structural **element**
  (e.g. button, menu)
- a stylistic **element**
  (themes, styles, fonts)
- an aspect of layout
  (padding, center)

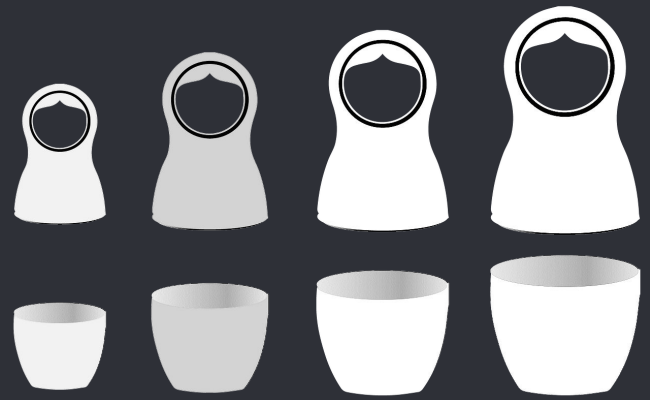
```
class PaddedText extends StatelessWidget {

 final String _data;

 PaddedText(this._data, {Key key})
      : super(key: key);

 @override
 Widget build(BuildContext context) {
   return new Padding(
        padding: const EdgeInsets.all(4.0),
        child: new Text(_data,
          style:
            const TextStyle(fontSize: 18.0))
   );
 }
}
```

Everything is a Widget



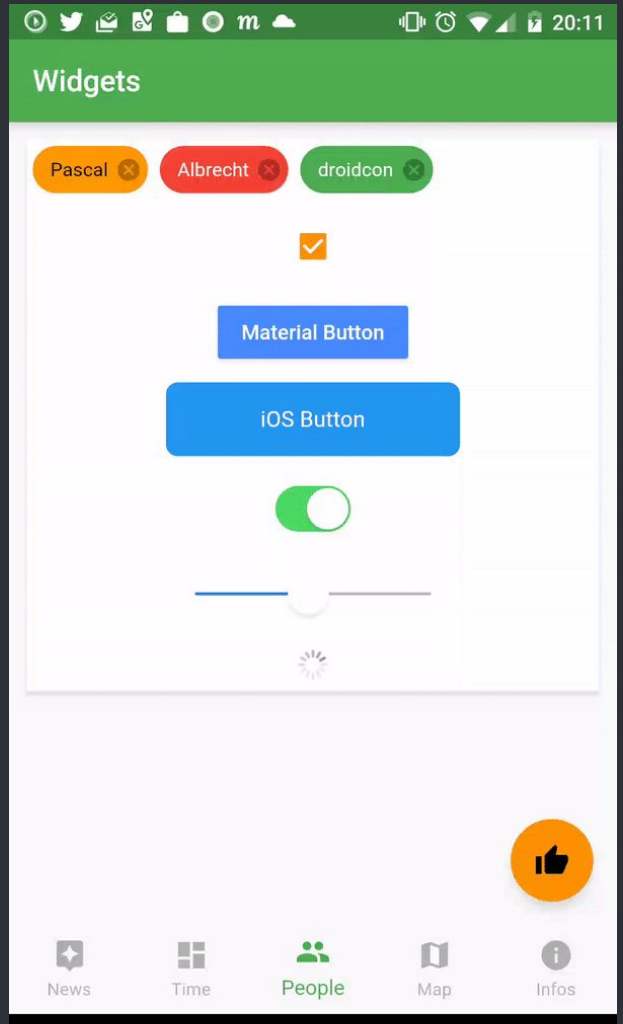
Everything is a Widget

- Application itself is a widget
- Hierarchically stacked
- inherit parent properties
- **Composition > inheritance**

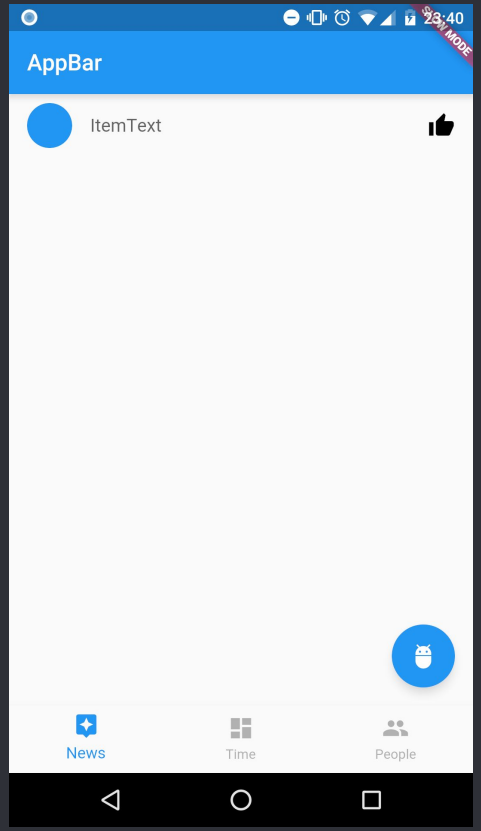


Existing Widgets

- **Material Guidelines** fully covered by Material Package
- **Human Interface Guidelines** iOS covered by Cupertino Package
- Premium Flutter Documentation



Important Material Widgets




```
new Scaffold(
 appBar: new AppBar(title: new Text('AppBar')),
 body: new ListView(
   children: <Widget>[
     new ListTile(
       leading: new CircleAvatar(),
       title: new Text('ItemText'),
       trailing: new Icon(Icons.thumb_up),
     )
   ],
 ),
 floatingActionButton: new FloatingActionButton(
     child: new Icon(Icons.adb),
     onPressed: () { /* do nothing */ }
 ),
 bottomNavigationBar: new BottomNavigationBar(
   items: [
     new BottomNavigationBarItem(
         icon: new Icon(Icons.assistant),
         title: new Text("News")),
     ...
],),);
```

Why do we want immutable Widgets?

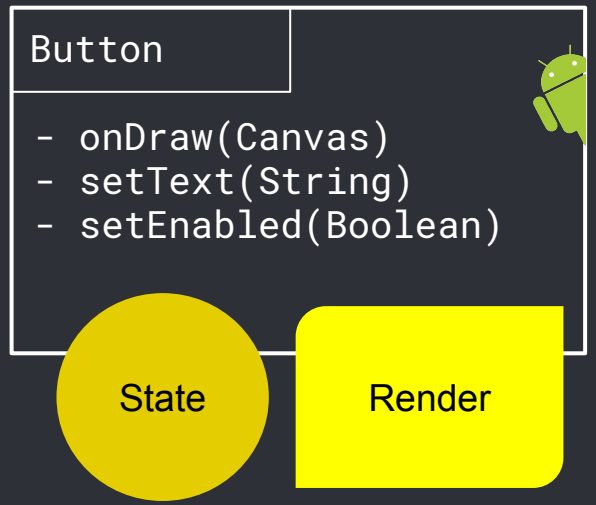
Mutation (evil)

```
val myText = new Button(context)
// initialize
myText.text = “Enabled”

fun disableButton() {
    myText.text = “Disabled”
    myText.isEnabled = false
}

fun enableButton() {
    myText.isEnabled = false
}

// Q: What’s the button text after
// calling enableButton()?
```

```
// A: undefined, depends on call order
```

!

Mutable View



Only the Button knows about its state

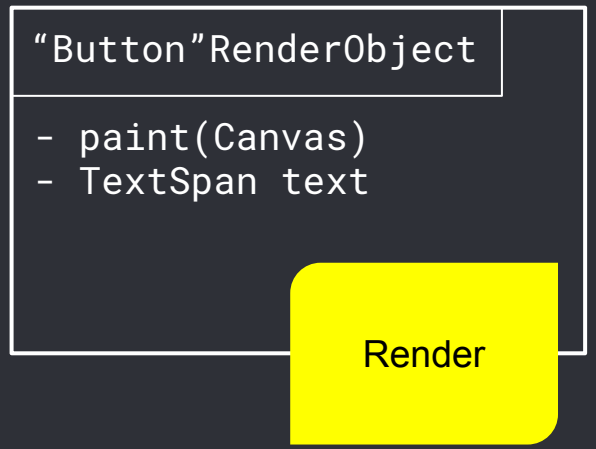
Two responsibilities
- drawing
- owning state

Declaration

State

```
var _enabled = true;

@override
Widget build(BuildContext context) {
 // Configure RenderObject for each possible state
 return RaisedButton(
   onPressed: _enabled ? () {...} : null,
   child: Text(_enabled ? "Enabled" : "Disabled"),
 );
}

void disableButton() {
  // change state
  setState((){ _enabled = false; });
}

void enableButton() {
  setState((){ _enabled = true; });
}
```

Mutable RenderObject




```
// Q: What’s the button text after
// calling enableButton()?

// A: “Enabled”
```

Reactive layer

View Layer

```dart
var _enabled = true;

@override
Widget build(BuildContext context) {
 // Configure RenderObject for each possible state
 return RaisedButton(
   onPressed: _enabled ? () {...} : null,
   child: Text(_enabled ? "Enabled" :
"Disabled"),
 );
}

void disableButton() {
  // change state
  setState((){ _enabled = false; });
}

void enableButton() {
  setState((){ _enabled = true; });
}
```

State

Widget tree

Element tree

Render tree

"Button"RenderObject

- paint(Canvas)
- TextSpan text

Render

Render

Render Object

Widget

Diff

Element

Update

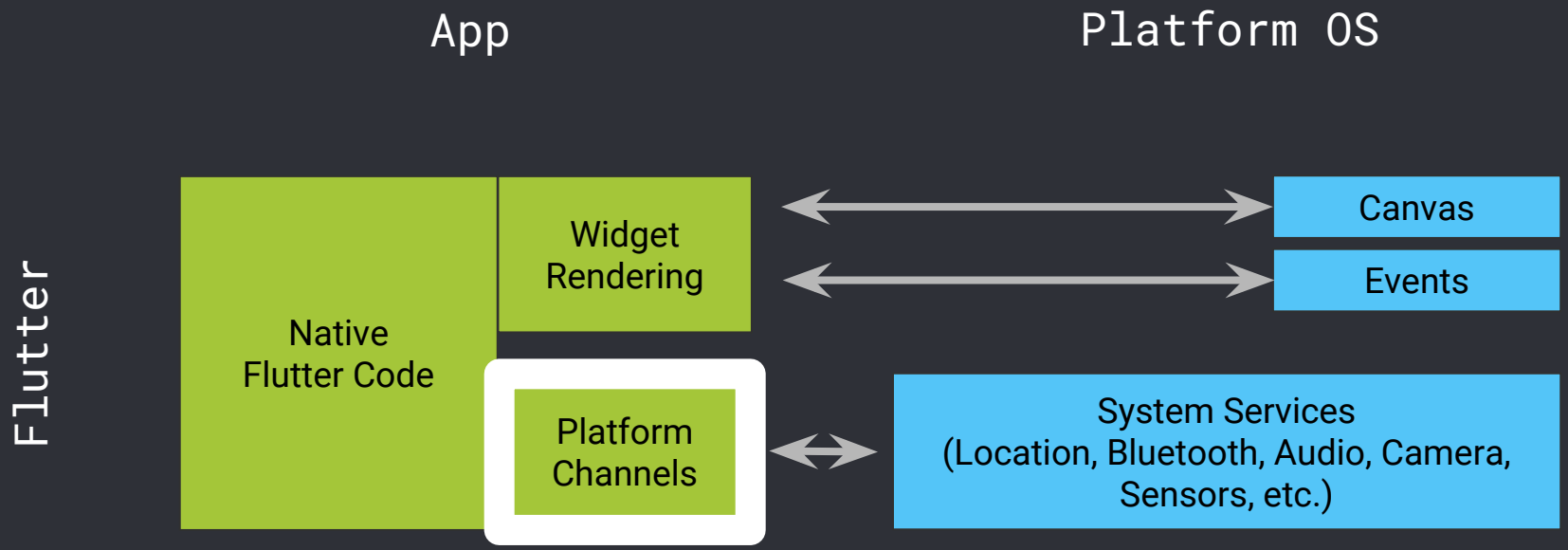
Build function

- For smooth animations it may be called for every frame (remember: 120FPS!)
- Flutter diffs the result with the previous build result to minimize updates of RenderObjects
- You don’t have to nest it very deep,
  - extract static parts
  - Split it in multiple build functions

Flutter

OS Integration

Integration with the OS



Communication between Android and Flutter

- `FlutterView` (`extends SurfaceView`) is placed fullscreen in your Activity.
- Plugins can be initialized which register a `MethodChannel` on the `FlutterView`.
- These `MethodChannel` are invoked by the plugins Dart API

SharedPrefs Plugin example

Dart part of plugin

```
static const MethodChannel methodChannel =
    const MethodChannel('samples.flutter.io/battery');

String batteryLevel;
try {
      final int result =
              await methodChannel.invokeMethod('getBatteryLevel');
      batteryLevel = 'Battery level: $result%.';
} on PlatformException {
      batteryLevel = "Failed to get battery level.";
}
```

SharedPrefs Plugin example

Android Kotlin part of plugin

```
val msgHandler: MethodCallHandler = MethodCallHandler { call, result ->
   if (call.method == "getBatteryLevel") {
       val level: Int = getBatteryLevelFromAndroid()

       if (level != -1) {
           result.success(level)
       } else {
           result.error("UNAVAILABLE", "Battery level not available.", null)
       }
   } else {
       result.notImplemented()
   }
}


MethodChannel(flutterView, "samples.flutter.io/battery").setMethodCallHandler(msgHandler)
```

Plugins

- Communication is contract based, can’t be type safe
  - Method name is `String`
  - Method args are named and dynamic (`Map<String, dynamic>`)
- `MethodChannel` work in both directions

Official Plugins

- Essential plugins
- Firebase plugins
- Android focused plugins

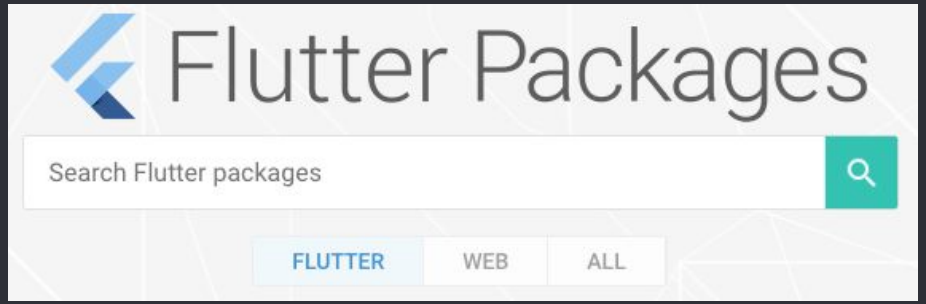
github.com/flutter/plugins

android_alarm_manager	pub v0.0.5
android_intent	pub v0.1.1
battery	pub v0.1.1
connectivity	pub v0.2.1
device info	pub v0.1.1
google_sign_in	pub v2.1.0
image_picker	pub v0.2.1
local_auth	pub v0.1.1
package_info	pub v0.2.0
path_provider	pub v0.3.1
quick_actions	pub v0.1.1
sensors	pub v0.2.1
share	pub v0.3.1
shared_preferences	pub v0.3.2

url_launcher	pub v2.0.1
video_player	pub v0.2.1
FlutterFire Plugins	
firebase_admob	pub v0.3.1
firebase_analytics	pub v0.2.3
firebase_auth	pub v0.4.5
cloud_firestore	pub v0.2.9
firebase_core	pub v0.0.6
firebase_database	pub v0.3.5
firebase_messaging	pub v0.1.3
firebase_storage	pub v0.1.4

Flutter Packages

- Pub - package manager
- over 1000 packages:
  - SQLite, GraphQL, Maps ...
- Participation appreciated



Flutter

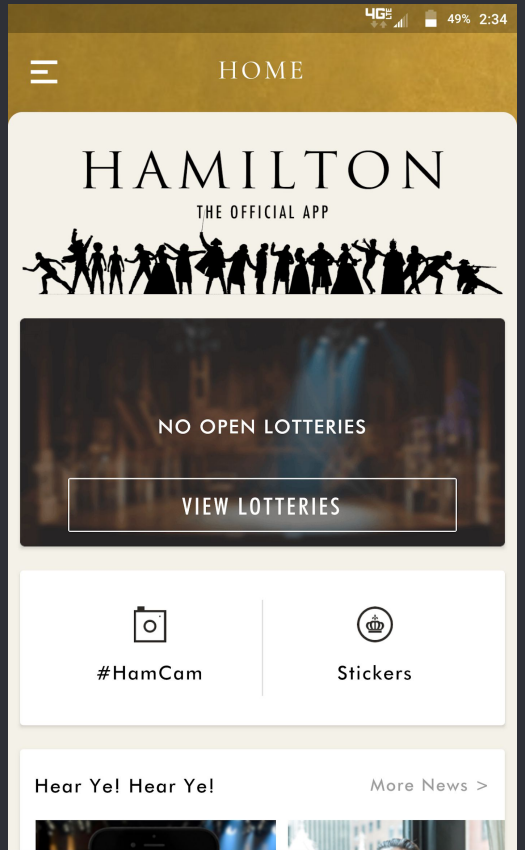
Now in beta

Is flutter production ready?

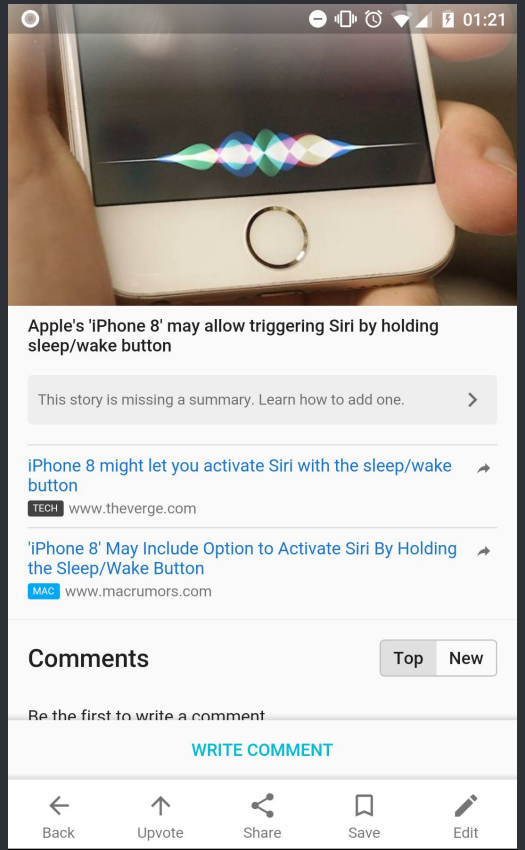
Pretty much...

Flutter in Production

Hamilton



Newsvoice



What's missing

- Retrofit/OkHttp and a persistent cache
- Google Maps
  - third-party approach with flutter widgets https://github.com/apptreesoftware/flutter_map
- Push Notifications (iOS) sometimes give no callback

Room for improvement

- Dart 2.0 syntax sugar
- Better integration into existing iOS and Android Apps
- More iOS related packages/plugins

Shared code with Dart?

- `FlutterView` is required to run dart code. You always need a context.
- Theoretically you can run Dart code in a background service, but its hacky
- “headless flutter” in early prototype on Android https://github.com/flutter/plugins/tree/master/packages/android_alarm_manager

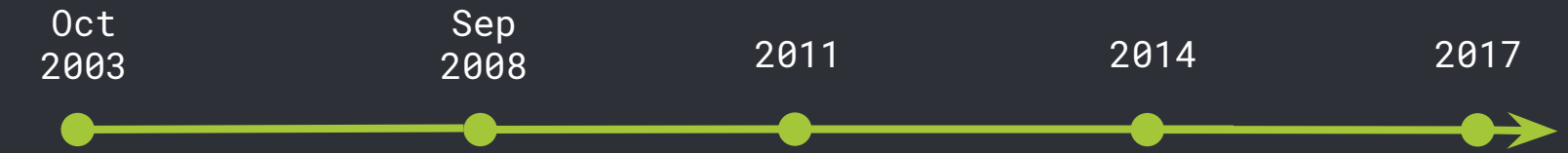
What is Fuchsia?

- Open-source OS by Google
- /ˈfjuːʃə/
- No Linux kernel - Google Kernel called Magenta
- Sky Engine with Vulkan
- Works on PixelBook
- Languages:
  - Dart, C++, Go, C, Python
  - No Java
- **Flutter Apps are “native” apps**

fuchsia.googlesource.com

Fuchsia Roadmap

Android



Openness of Dart/Flutter/Fuchsia

- Everything is open source
- Bug trackers are public and used by Googlers
- Getting things merged is pretty fast

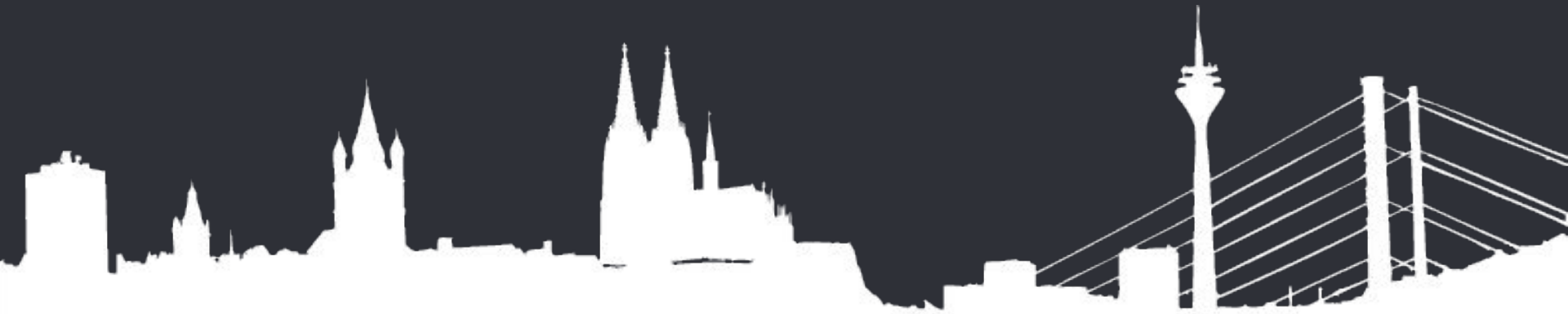
- Get help in Gitter gitter.im/flutter/flutter

We are hiring!

grandcentrix.jobs
@grandcentrix

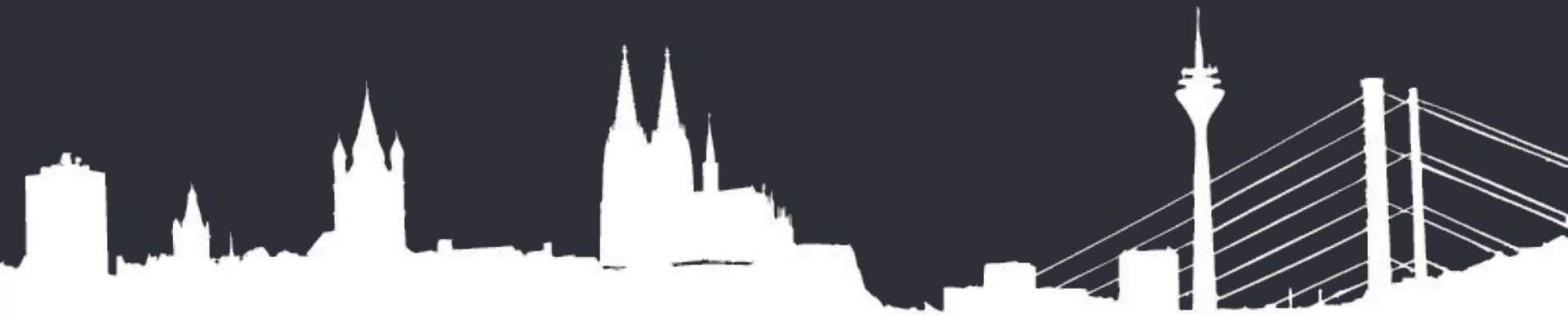
+Albrecht Noll
@UhrArt

+Pascal Welsch
@passsy



Slides: welsch.link/flutter-cgn

Next: LIVE CODING



First Steps - 5 min Setup

- Clone repo and add to $PATH:

```
$ git clone -b beta https://github.com/flutter/flutter.git
$ export PATH=`pwd`/flutter/bin:$PATH
```

- Run flutter doctor and do the suggested tasks

```
$ flutter doctor
```

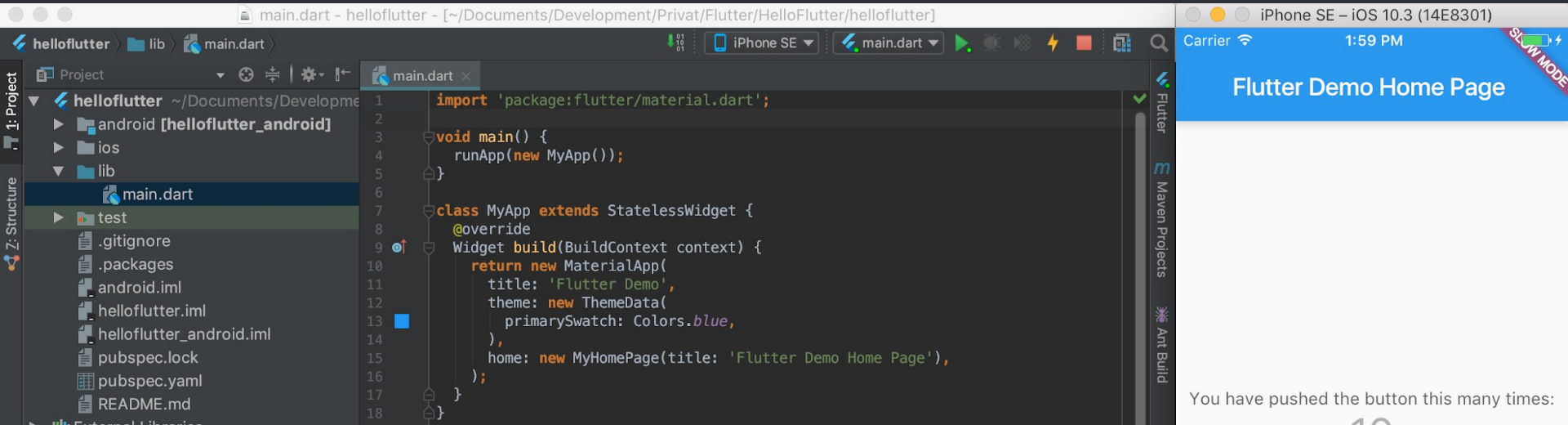
- Start developing

First Steps - Hello Flutter

- Create a new project

```
$ flutter create myapp
```

- Or use the Project Wizard in IntelliJ IDEA



Learning Resources

- Official Page: https://flutter.io
- Flutter Weekly https://flutterweekly.net/
- Dart Bootstrap: https://www.dartlang.org/guides/language/language-tour
- Widget catalog: https://flutter.io/widgets
- Ui Codelab: https://codelabs.developers.google.com/codelabs/flutter/
- Firebase Codelab: https://codelabs.developers.google.com/codelabs/flutter-firebase
- **Valuable Flutter Links**: https://github.com/Solido/awesome-flutter
- Flutter Examples: https://github.com/nisrulz/flutter-examples