Flutter

60 FPS UI of the Future
Agenda

- **Android** (facts and opinions)
- **Flutter** (facts)
- **Dart** (code and opinions)
- **Flutter** (opinions)
- **Fuchsia** (speculations)
Android History

- **Founded**: Oct 2003
- **2008**: 1.0
- **2011**: 4.0
- **2014**: 5.0
- **2017**: 8.0

- **Holo Design**
- **Material Design**

- **Android View API Design** (View.java)
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 ‘17
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

120 FPS UI of the Future
What is Flutter?

- A mobile app SDK containing
  - complete UI Framework for mobile apps
  - huge Widget catalog
  - Tools
- Allows building beautiful mobile apps
- Platform independent, currently supporting Android, iOS and Fuchsia
- Uses Dart - Easy to learn language by Google
Flutters goals

- Beautiful fluid UIs
- Run same UI on multiple platforms, perfect for brand-first designs
- High-performance apps that feel natural on different platforms
- Be productive
Flutter highlights

- Super performant, 120fps without optimizations
- Fast development - Hot Reload
- Modern, reactive framework like React
Flutter is not yet another Cross-Platform SDK

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

- JavaScript

Flutter

- Native Flutter Code
  - Widget Rendering
  - Platform Channels

App

Platform OS

- OEM Widgets
- Canvas
- Events
- System Services (Location, Bluetooth, Audio, Camera, Sensors, etc.)
What is Dart?

- Java like language - easy to learn
- aimed to replace Javascript (2010)
- DartVM
- Javascript compiler (dart2js)

- Great language compared to Javascript and Java 6
- Missing syntactical sugar from Kotlin
Dart 2.0

- 2.0 is coming soon™
- Will be sound (type safe)
- Will most likely get nullable types
- “new” could become optional
- Language discussions are available in the sdk repository
- [https://github.com/dart-lang/sdk/tree/master/docs](https://github.com/dart-lang/sdk/tree/master/docs)
  - Weekly newsletter (6 weeks in a row)
  - Informal specifications
First Steps - 5 min Setup

- Clone repo and add to $PATH:
  
  $ git clone -b alpha 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
Flutter

Native Flutter Code

Platform Channels

Widget Rendering

App

Platform OS

Canvas

Events

System Services
(Location, Bluetooth, Audio, Camera, Sensors, etc.)
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)
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**
Flutter layered UI Architecture

Flutter:
- Widgets (immutable)
- Custom RenderObjects (* extends RenderObjects)
- Rendering (Layout)
- dart:ui (Canvas)

Android:
- Custom Views (support design library)
- android.view (View, Layout)
- android.graphics (Canvas)
Important Material Widgets

```dart
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?
Mixed responsibilities (Android)

1. Declare View in XML with initial attributes

```xml
<TextView
    android:id="@+id/myText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Hello Droidcon"
    android:textSize="14sp" />
```

2. Mutate View with updated data

```kotlin
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    val myText = findViewById<TextView>(R.id.myText)
    api.getData().subscribe{
        data ->
            // mutate view, append text
            myText.text += data
    }, { e ->
        handleError(e)
    }
}
```

The `TextView`, responsible for **drawing** text, is now also responsible for the **state** of the text!
Widgets on Flutter are immutable

- You can’t change the text of a Widget, a new Widget instance is required
- build is a one way function binding data to immutable Widgets
- setState schedules rebuild of the widget

```dart
// boilerplate
class DroidconWidget extends StatefulWidget {
  @override
  State<Widget> createState() {
    return new DroidconState();
  }
}

class DroidconState extends State<DroidconWidget> {
  var _data = "Hello Droidcon";

  @override
  void initState() {
    api.getData().then((data) {
      // append text, trigger rebuild
      setState(() {
        _data += data;
      });
    });
  }

  @override
  Widget build(BuildContext context) {
    return new Text(_data);
  }
}
```
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
- You don’t have to nest it very deep,
  - extract static parts
  - Split it in multiple build functions
Integration with the OS

Flutter

Native Flutter Code

Widget Rendering

Platform Channels

App

Platform OS

Canvas

Events

System Services (Location, Bluetooth, Audio, Camera, Sensors, etc.)
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

```dart
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

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

        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

<table>
<thead>
<tr>
<th>Plugin</th>
<th>Pub</th>
</tr>
</thead>
<tbody>
<tr>
<td>android_intent</td>
<td>pub v0.0.1</td>
</tr>
<tr>
<td>battery</td>
<td>pub v0.0.1</td>
</tr>
<tr>
<td>connectivity</td>
<td>pub v0.0.1</td>
</tr>
<tr>
<td>device info</td>
<td>pub v0.0.1</td>
</tr>
<tr>
<td>google_sign_in</td>
<td>pub v0.3.1</td>
</tr>
<tr>
<td>image_picker</td>
<td>pub v0.1.1</td>
</tr>
<tr>
<td>local_auth</td>
<td>pub v0.0.1</td>
</tr>
<tr>
<td>package_info</td>
<td>pub v0.0.1</td>
</tr>
<tr>
<td>path_provider</td>
<td>pub v0.2.1+1</td>
</tr>
<tr>
<td>quick_actions</td>
<td>pub v0.0.1</td>
</tr>
<tr>
<td>sensors</td>
<td>pub v0.1.0</td>
</tr>
</tbody>
</table>

### FlutterFire Plugins

<table>
<thead>
<tr>
<th>Plugin</th>
<th>Pub</th>
</tr>
</thead>
<tbody>
<tr>
<td>share</td>
<td>pub v0.2.1</td>
</tr>
<tr>
<td>shared_preferences</td>
<td>pub v0.2.5</td>
</tr>
<tr>
<td>url_launcher</td>
<td>pub v0.4.2+4</td>
</tr>
<tr>
<td>firebase_analytics</td>
<td>pub v0.1.0</td>
</tr>
<tr>
<td>firebase_auth</td>
<td>pub v0.2.0</td>
</tr>
<tr>
<td>firebase_database</td>
<td>pub v0.1.0</td>
</tr>
<tr>
<td>firebase_messaging</td>
<td>pub v0.0.5</td>
</tr>
<tr>
<td>firebase_storage</td>
<td>pub v0.0.5</td>
</tr>
</tbody>
</table>

[github.com/flutter/plugins](https://github.com/flutter/plugins)
Shared code with Dart

- **FlutterView** is required to run dart code. You always need an Activity.
- You can’t run Dart code in a background service.
- You can’t reuse network or parsing logic in your JobScheduler.
- Unclear if this will ever work.
Is flutter production ready?

No, but...

...the Flutter team is very aware of it and working hard to make it production ready.
What’s missing

- Retrofit/OkHttp and a persistent cache
- Google Maps
- Push Notifications (iOS) sometimes give no callback
- No “headless flutter”
Room for improvement

- brackets hell, no DSL
  - workaround 'closing labels'
    in VS Code in IntelliJ maybe?!
- Flatten with variables,
  extract methods
- One missing comma,
  breaks code completion
Openness of Dart/Flutter/Fuchsia

- Everything is open source
- Bug trackers are public and used by Googlers
- Dartlang newsletter inside sdk repository with detailed language decisions for Dart 2.0

- Get help in Gitter [gitter.im/flutter/flutter]
What is Fuchsia?

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

fuchsia.googlesource.com
We are hiring!

grandcentrix.jobs
@grandcentrix

+Albrecht Noll
@UhrArt

+Pascal Welsch
@passsy
Learning Resources

- Official Page: [https://flutter.io](https://flutter.io)
- Widget catalog: [https://flutter.io/widgets](https://flutter.io/widgets)
- Ui Codelab: [https://codelabs.developers.google.com/codelabs/flutter/](https://codelabs.developers.google.com/codelabs/flutter/)
- Firebase Codelab: [https://codelabs.developers.google.com/codelabs/flutter-firebase](https://codelabs.developers.google.com/codelabs/flutter-firebase)
- **Valuable Flutter Links**: [https://github.com/Solido/awesome-flutter](https://github.com/Solido/awesome-flutter)
- Flutter Examples: [https://github.com/nisrulz/flutter-examples](https://github.com/nisrulz/flutter-examples)