

Flutter Fundamentals

Short recap day #1



Peter Kassenaar –
info@kassenaar.com

Agenda - details



- [Introduction](#) – overview of the Flutter landscape
- Flutter [tooling](#) – installation
- Hello World –the [structure and architecture](#) of Flutter apps.
- A [Dart Primer](#)
- Zooming in on Flutter:
 - Components – [Stateless](#) vs. [Stateful](#)

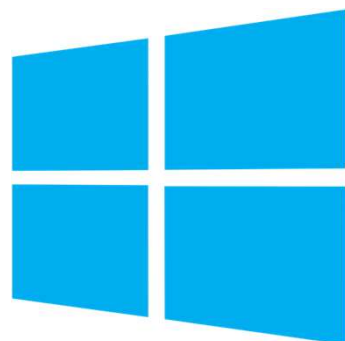


Agenda – cont'd

- Widgets - introduction
- The Flutter layout system – `Scaffold()` and more
- Using Images and assets
- More layout widgets
 - `Button()`, `Icon()`, `Container()`, `Padding()`
 - `Row()`, `Column()`, properties, children and more
- Working with data
 - `ListView()`, `Card()`, Designing layouts



*"Flutter is **Google's UI toolkit** for building beautiful, **natively compiled** applications for mobile, web, and desktop from a **single codebase.**"*



Installation – recommended order



1. Install Visual Studio
2. Install IntelliJ
3. Install Flutter plug-in for IntelliJ
4. Install Flutter SDK
5. Update Windows PATH variable
6. Run `flutter doctor`, fix any possible problems

Default code – recognize the structure



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

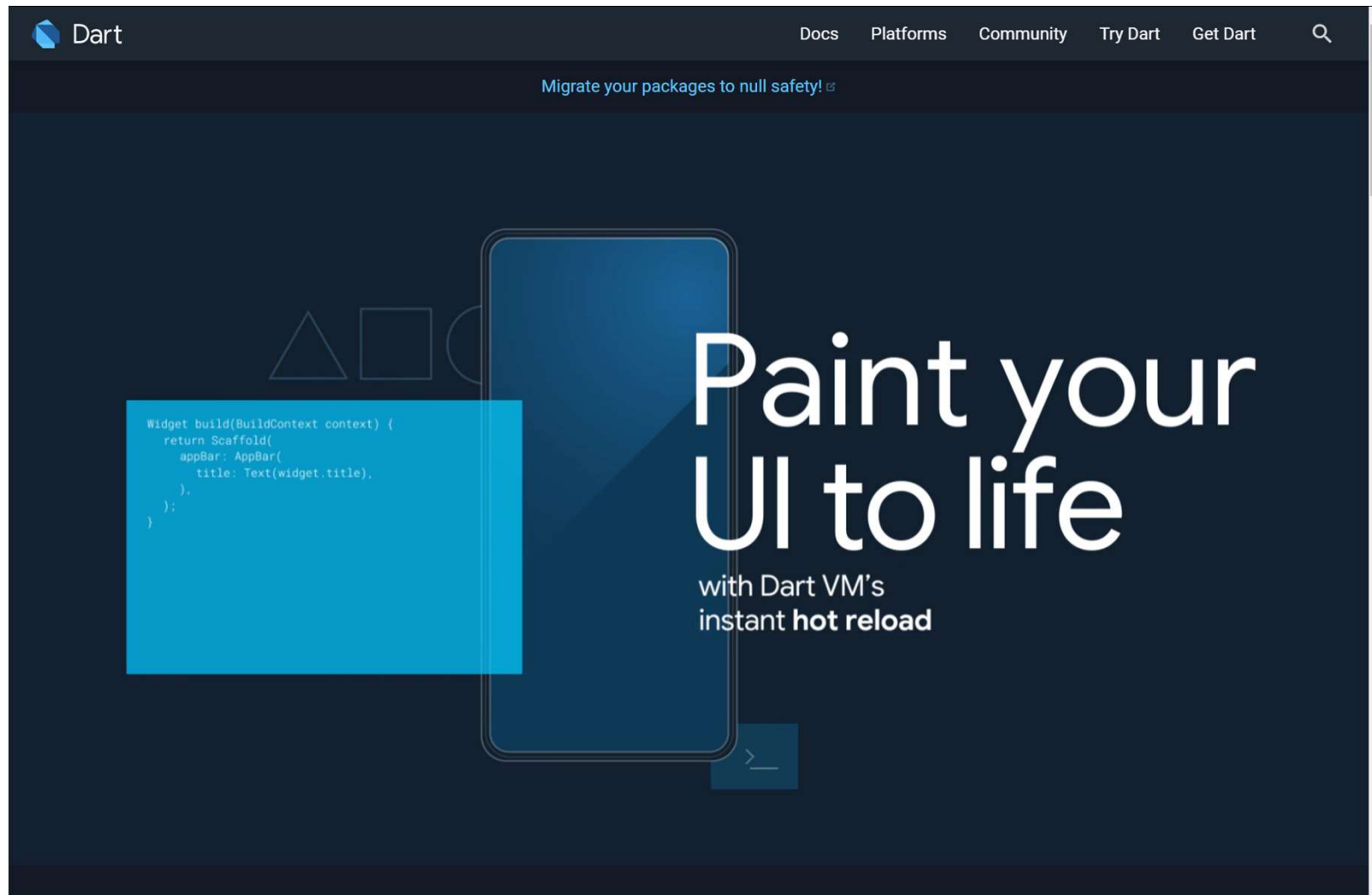
void main() {
  runApp(MyApp());
}

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,
        visualDensity: VisualDensity.adaptivePlatformDensity,
      ),
      home: MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}
...
```

(Yours might be slightly different,
due to updates)

Study the default code. It has [useful comments](#).

dart.dev

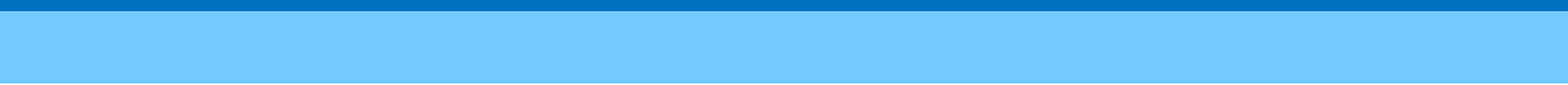


Flutter == Dart in action



- Important widgets
 - Scaffold()
 - AppBar()
 - Themes, fonts & colors
 - Image()
 - Icon()
 - Row(), Column()
 - ListView()
 - Container()
 - Expanded()

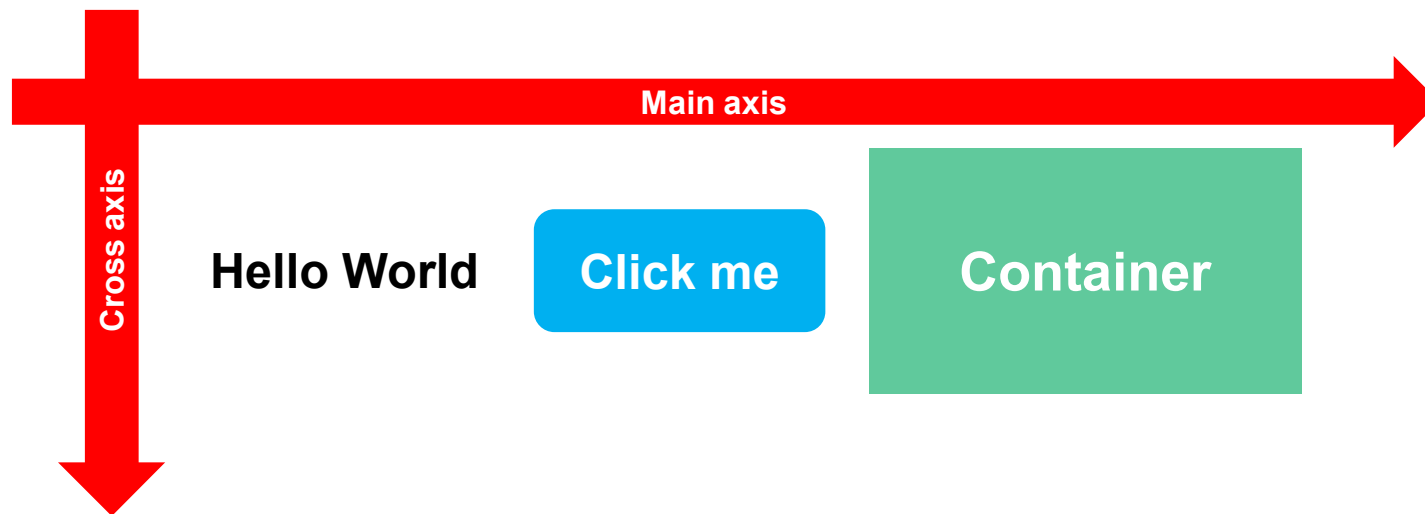
*“Every Flutter App
is composed as a
tree of widgets”*





Alignment in Rows/Columns

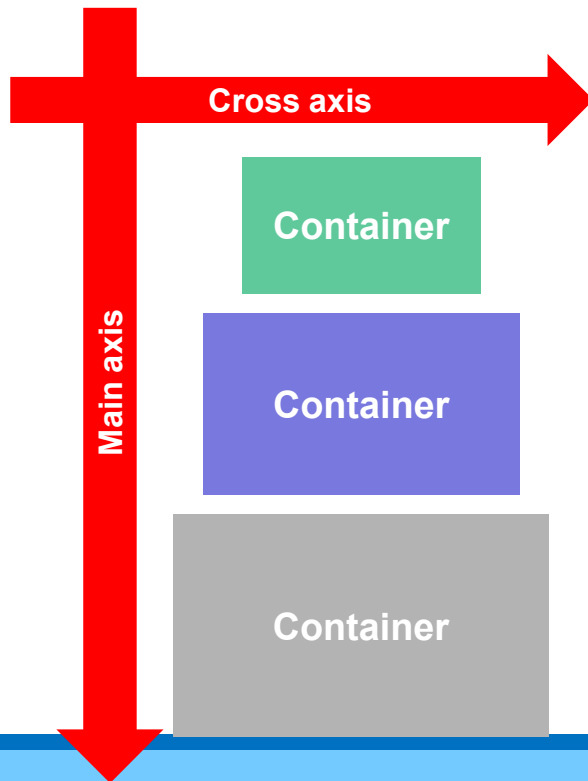
- In Rows:
 - Use `MainAxisAlignment` for horizontal layout
 - Use `CrossAxisAlignment` for vertical layout



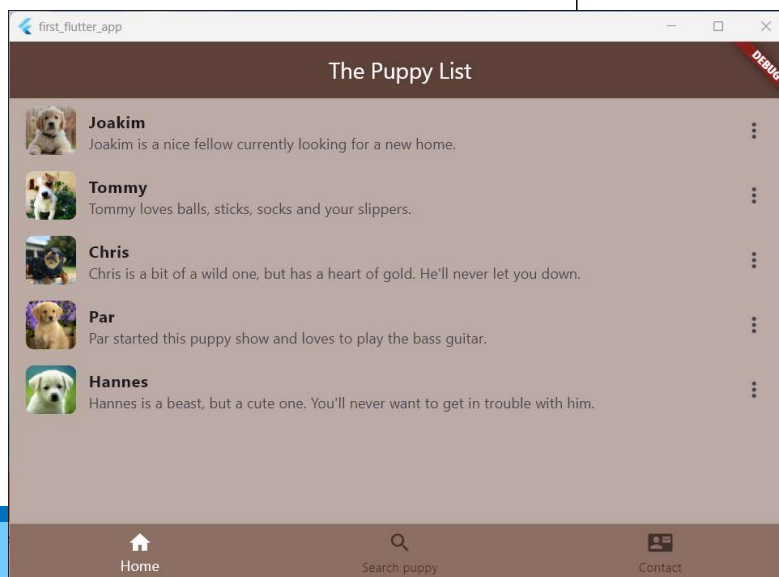
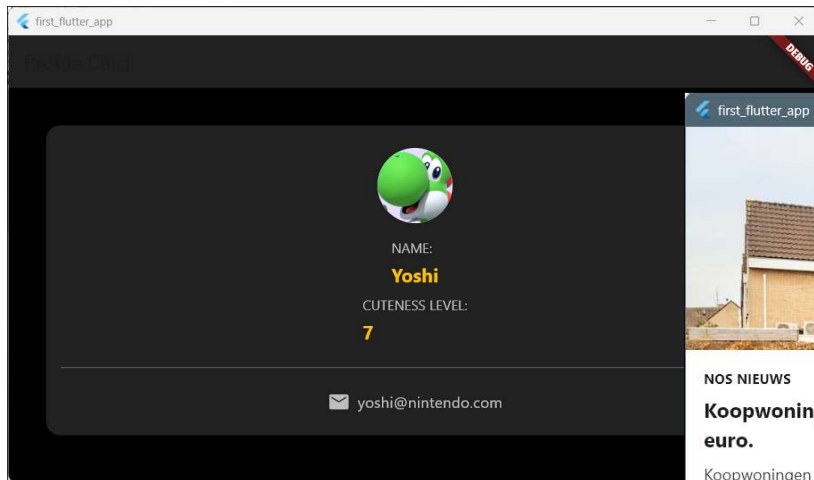
Alignment



- In Columns – opposite to Rows:
 - Use `MainAxisAlignment` for vertical layout
 - Use `CrossAxisAlignment` for horizontal layout



Creating various layouts by combining Widgets

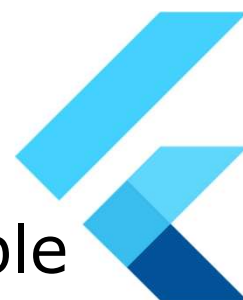




Questions:

- Using `const` or not in code?
- Use `const` in situations where the following is true:
 - The object is entirely immutable.
 - Its parameters (if any) are *also compile-time constants*.
 - You want to *avoid recreating the object unnecessarily* every time a widget rebuilds.
- E.g: consider this snippet:

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



- Removing the keyword `const` is less optimal!
- It still works because `Placeholder()` is immutable by design.
- However, removing `const` makes a difference in performance!
 - Without `const`, a *new instance* of `Placeholder()` will be created every time this widget rebuilds.
 - With `const`, the *same compile-time constant instance is reused* across rebuilds, improving performance by avoiding unnecessary object creation!

```
Widget build(BuildContext context) {  
    return Placeholder(); // also works!  
}
```

Dart Extension Methods:

Announcing Dart 3.7! Find out about updates to the language, analyzer, pub.dev, and more, in the [blog post](#).

Dart Overview Docs Community Try Dart Get Dart

Classes & objects

- Classes
- Constructors
- Methods
- Extend a class
- Mixins
- Enums
- Extension methods**
- Extension types
- Callable objects
- Class modifiers
- Concurrency
- Null safety
- Core libraries
- Effective Dart
- Packages
- Development
- Interoperability
- Tools & techniques

Extension methods

Extension methods add functionality to existing libraries. You might use extension methods without even knowing it. For example, when you use code completion in an IDE, it suggests extension methods alongside regular methods.

If watching videos helps you learn, check out this overview of extension methods.

Dart extension methods

Extension Methods

[Later bekij...](#) [Delen](#)

[Bekijken op YouTube](#)

[Dart extension methods](#)

Overview

When you're using someone else's API or when you implement a library that's widely used, it's often impractical or impossible to change the API. But you might still want to add some functionality.

Contents

- Overview
- Using extension methods
 - Static types and dynamic
 - API conflicts
- Implementing extension methods
 - Unnamed extensions
- Implementing generic extensions
- Resources

dart.dev uses cookies from Google to deliver and enhance the quality of its services and to analyze traffic. [Learn more](#) [OK, got it](#)

<https://dart.dev/language/extension-methods>



other

Questions?



Today:

- State management in various ways
 - Stateful widgets – using models/custom classes
 - passing parameters, passing functions
- Communicating with external API's
 - http / other methods
- More state management
 - bloc pattern
- (gRPC)