

Ziel der App

Die App soll Name und Nachname des Benutzers nehmen und diese Lokal Speichern

Entwicklerumgebung einrichten

1. Flutter [Dokumentation](#) öffnen
2. Betriebssystem auswählen und Dokumentation folgen

Wichtig zu merken ist:

- iOS app debugging ist nur über MacOS möglich
- Android App Entwicklung benötigt Android Studio (Auf Windows und MacOS)
- iOS App Entwicklung benötigt XCode auf MacOS (ist nicht auf Windows installierbar)
- Command Line Tools müssen installiert sein (Auf Windows sowie MacOS)
- Bei Android Entwicklung müssen USB Drivers heruntergeladen werden
 - Wichtig ist dabei zu achten, dass man die Richtigen Drivers herunterlädt. Jedes Android gerät benötigt eigene Drivers die nur über die Webseite des Herstellers heruntergeladen werden kann

WICHTIG: Entwickleroptionen auf Andoird bzw. iOS aktivieren (USB Debugging)

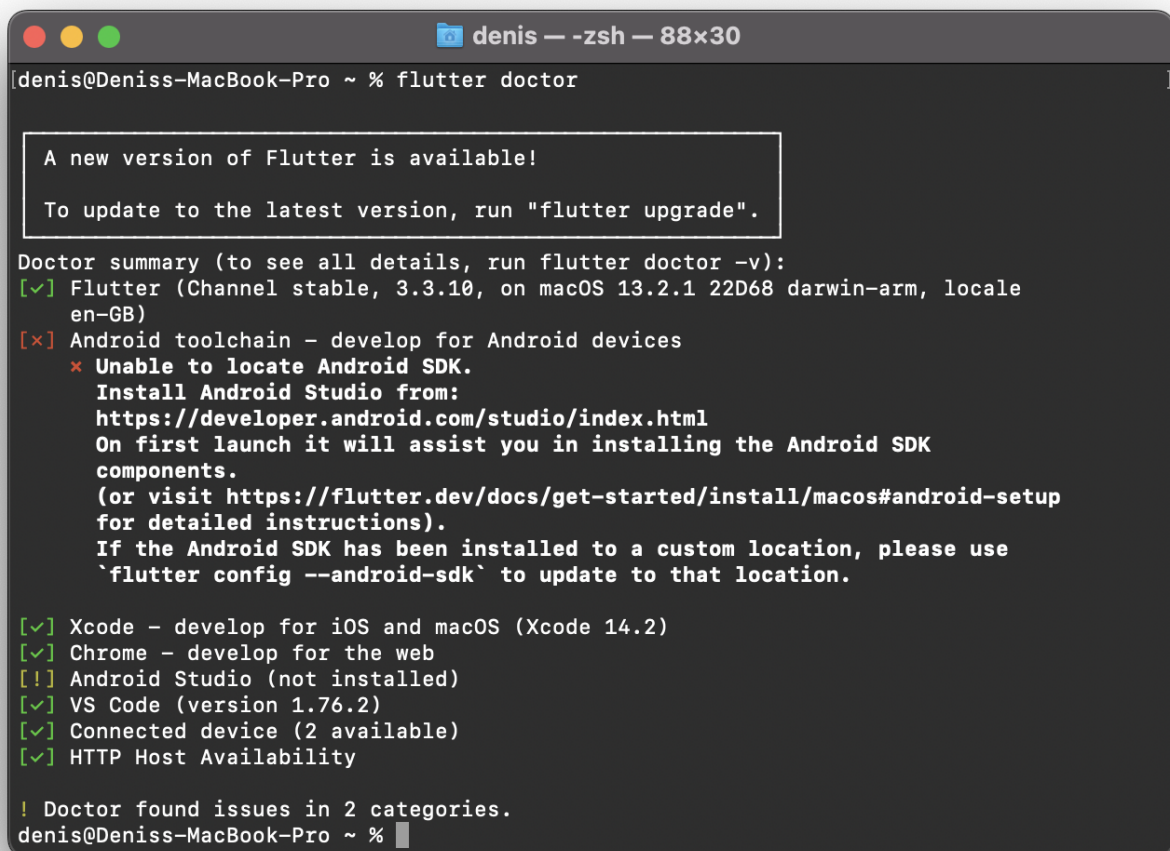
Android: <https://developer.android.com/studio/debug/dev-options>

iOS: <https://developer.apple.com/documentation/xcode/enabling-developer-mode-on-a-device>

iOS benötigt zudem einen Entwickler-Konto

<https://developer.apple.com/> (Oben rechts auf "Account" drücken und mit Apple ID anmelden)

Wenn alles korrekt installiert ist dann sollte im terminal `flutter doctor` einen ähnlichen Output geben:



```
[denis@Deniss-MacBook-Pro ~ % flutter doctor]

A new version of Flutter is available!

To update to the latest version, run "flutter upgrade".

Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.3.10, on macOS 13.2.1 22D68 darwin-arm, locale en-GB)
[✗] Android toolchain - develop for Android devices
    ✗ Unable to locate Android SDK.
      Install Android Studio from:
      https://developer.android.com/studio/index.html
      On first launch it will assist you in installing the Android SDK components.
      (or visit https://flutter.dev/docs/get-started/install/macos#android-setup for detailed instructions).
      If the Android SDK has been installed to a custom location, please use
      `flutter config --android-sdk` to update to that location.
[✓] Xcode - develop for iOS and macOS (Xcode 14.2)
[✓] Chrome - develop for the web
[!] Android Studio (not installed)
[✓] VS Code (version 1.76.2)
[✓] Connected device (2 available)
[✓] HTTP Host Availability

! Doctor found issues in 2 categories.
denis@Deniss-MacBook-Pro ~ %
```

Falls fehler aufscheinen sollten (Im screenshot z.B. fehlt Android Studio), anleitungen im terminal folgen

3. Textbearbeitungsprogramm installieren ([Visual Studio Code](#) ist empfehlenswert)

Flutter projekt starten

1. Terminal in einem beliebigen Ordner öffnen (App Ordner wird später dort gespeichert)
2. Im Terminal `flutter create <app name hier eingeben>` (<> löschen)
3. Jetzt sollte ein Ordner mit dem ausgewählten Namen zu finden sein, diesen in Visual Studio Code öffnen (oder beliebigen IDE)

App bearbeiten

Alle Dateien, bis auf `lib/main.dart`, können momentan ignoriert werden

1. Im Ordner `lib` die Datei `main.dart` öffnen

Die `main.dart` Datei ist wichtig und ist die Hauptdatei einer Flutter App

In `main.dart` findet man schon einen Sourcecode, weil Flutter schon eine Beispiel App generiert und diese kann ohne weiteres ausgeführt werden

Flutter ist ein Framework und basiert sich auf der Programmiersprache "Dart"

"Dart" ähnelt in vielen Bereichen Java und C++, hat aber eine viel einfachere Syntax

```
// Flutter framework wird hier importiert
import 'package:flutter/material.dart';

// Die methode main() wird beim start aufgerufen und führt runApp() aus
// cont MyApp() ist die effektive App die auf dem Bildschirm angezeigt wird
void main() {
  runApp(const MyApp());
}
```

Flutter Apps werden mit verschiedenen Bausteinen (Widgets) zusammengestellt

Sehr empfehlenswert!

Um Widgets in Flutter besser zu verstehen gibt es einen gratis Online-Kurs auf der Flutter Webseite (<https://codelabs.developers.google.com/codelabs/flutter-codelab-first#0>)

Weitere Ressourcen zum Flutter lernen findet man unter (<https://flutter.dev/learn>)

App Beispiel (Rainbow-app)

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

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

class MyApp extends StatelessWidget {
  const MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key});
  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  // void wird benutzt um Funktionen in Dart zu deklarieren (wie in Java)
  // Diese Funktion steigert die Variable _counter jedes mal um einen Wert
  void _incrementCounter() {
    setState(() {
      // Hier findet die Inkrementation statt (_counter = _counter + 1)
      _counter++;

      // Dieses if-Statement kontrolliert ob _counter größer ist als die Farbenliste (colors) und setzt _counter wieder auf 0, damit
      // kein Fehler aufscheint
      if (_counter > colors.length - 1) {
        _counter = 0;
      }
    });
  }

  // List ist ein Array (Liste von Variablen)
  // <MaterialColor> gibt an was für ein Datentyp sich in dieser Liste befindet
  List<MaterialColor> colors = [
    Colors.purple,
    Colors.blue,
    Colors.amber,
    Colors.green,
    Colors.yellow,
    Colors.pink,
    Colors.teal,
    Colors.orange,
    Colors.red,
  ];
}
```

```

// List ist ein Array (Liste von variablen)
// <String> In dieser Liste befinden sich, Strings, also Text
List<String> strings = [
    "Click me UwU",
    "SUS",
    "ok pls stop",
    "Man. stop. pls.",
    "I SAID STOP",
    "Ok fine I warned you",
    "We're no strangers to loooove...",
    "You know the rules and so do I!!!!",
    "Error 69420, sins_johnny.exe stopped working",
];

// Diese Funktion baut die Benutzeroberfläche (muss in jeder Flutter-App mindestens ein Mal vorkommen)
@override
Widget build(BuildContext context) {
    // Alles was nach return steht, beinhaltet alle Widgets, die auf dem Bildschirm aufscheinen sollen
    return Material(
        color: colors[_counter],
        child: Center(
            child: TextButton(
                child: Text(
                    strings[_counter],
                    style: TextStyle(color: Colors.white, fontSize: 32)),
                onPressed: () => {
                    _incrementCounter(),
                },
            ),
        ),
    );
}
}

```

Dies ist ein Beispielcode für eine Flutter-App, die eine Schaltfläche mit einem Text anzeigt, der sich jedes Mal ändert, wenn die Schaltfläche gedrückt wird. Die Hintergrundfarbe ändert sich ebenfalls jedes Mal.

MyHomePage ist ein StatefulWidget ([unterschied zwischen StatefulWidget und StatelessWidget](#)), was bedeutet, dass es sich während der Laufzeit ändern kann. Es enthält eine Schaltfläche, die bei jedem Drücken den Text ändert und die Hintergrundfarbe ändert. Der Text und die Hintergrundfarbe werden aus den Listen colors und strings abgefragt, die jeweils eine Reihe von vordefinierten Werten enthalten.

Die App wird mit dem Befehl "flutter run" (oder **Ctrl** + **F5** in VSCode) gestartet. Der Emulator oder das Gerät zeigt die Schaltfläche und den aktuellen Text mit der entsprechenden Hintergrundfarbe an. Wenn Sie auf die Schaltfläche klicken, wird der Text geändert und die Hintergrundfarbe wechselt entsprechend.

Dies ist ein sehr einfaches Beispiel für eine Flutter-App, die zeigt, wie man ein StatefulWidget verwendet, wie man Text und Hintergrundfarbe ändert und wie man eine Schaltfläche hinzufügt. Es ist ein guter Ausgangspunkt für Anfänger, die lernen möchten, wie man Flutter-Apps entwickelt.

App Besipiel #2 (Text input)

```

import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
    const MyApp({super.key});

    // This widget is the root of your application.

    @override
    Widget build(BuildContext context) {
        return MaterialApp(
            title: 'Input Beispiel',
            theme: ThemeData(
                primarySwatch: Colors.blue,
                home: const MyHomePage(title: 'Input Beispiel'),
            );
        );
    }
}

class MyHomePage extends StatefulWidget {
    const MyHomePage({super.key, required this.title});

    final String title;

    @override
    State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
    TextEditingController text = TextEditingController();

    @override
    Widget build(BuildContext context) {
        return Scaffold(

```

```

    appBar: AppBar(
      title: Text(widget.title),
    ),
    body: Column(
      children: [
        Spacer(),
        Center(
          child: Text(
            '${text.text}',
            style: TextStyle(fontSize: 72),
          ),
        ),
        Spacer(),
        Padding(
          padding: const EdgeInsets.symmetric(horizontal: 20.0),
          child: TextField(
            onChanged: (value) {
              setState(() {});
            },
            decoration: InputDecoration(
              border: OutlineInputBorder(),
              hintText: 'Geben Sie hier text ein',
            ),
            controller: text,
          ),
        ),
        Spacer()
      ],
    ),
  ],
);
}
}

```

Diese Beispiel-App nimmt Input durch ein Textfeld ein und zeigt dies an, wenn sich dieser ändert (onChange auf Zeile 80)