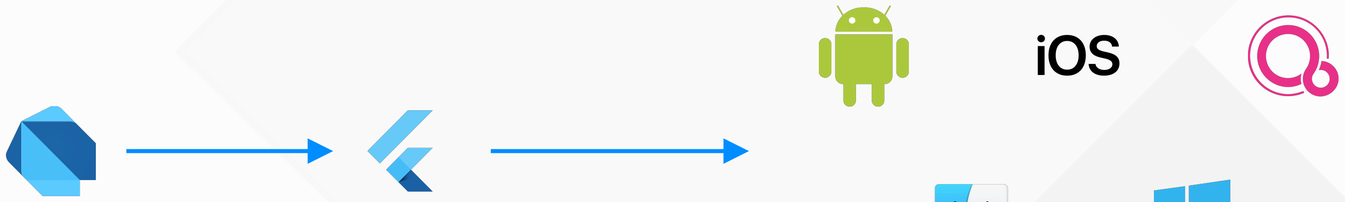
cross platform development.

Florian Pürschel Samir Wadi Larissa Rothmann Dominik Liesfeld Julia Kasanzewa

Was ist cross platform development ?

Die Idee hinter cross platform development ist, dass man eine Software plattformunabhängig realisiert. Das heißt, dass die Software unabhängig von dem zu Grunde liegenden Betriebssystem lauffähig ist.

Was ist cross platform development ?



Dart

Flutter

verschiedene Betriebssysteme

Entstehung

Es lässt sich nicht exakt zurückverfolgen, was die erste cross platform development Umgebung war.

Adobe Flash wird in vielen Medien als das ‚ultimative Urzeit cross platform tool‘ bezeichnet.

Entstehung

Adobe Flash wurde 1997 von Macromedia als Macromedia Flash Version 1 veröffentlich.

Adobe Flash ermöglicht die Programmierung und Darstellung von interaktiven Inhalten über mehrere Plattformen.

HTML5 hat Adobe Flash 2014 vom Funktionsumfang entgültig abgelöst.

Entstehung

Cross platform development ermöglicht Entwicklern kostengünstiger und effizienter Software zu entwickeln, was einen großen Mehrwert für Forschungseinrichtungen und Unternehmen bietet.

Entwickler können mit ihrer „Muttersprache“ in den meisten Fällen programmieren und müssen sich nicht auf plattform abhängige Sprachen einstellen. *(in den meisten Fällen)*

Kategorien

Es gibt drei verschiedene Arten von cross platform Applikationen.

1

WEB APPLIKATIONEN

2

NATIVE APPLIKATIONEN

3

HYBRIDE APPLIKATIONEN

Web Applikationen

WebView Applikationen sind eine Kombination aus nativen Applikationen und Webseiten. Sie werden mit HTML, CSS und JavaScript geschrieben und simulieren mittels einer WebView eine Applikation.

Web Applikationen

Vorteile: Die meisten Web Applikationen werden mit HTML, CSS und JavaScript realisiert und laufen Betriebssystem unabhängig in einem Browser. Durch diese Technologien ist die Entwicklung sehr einfach.

Web Applikationen

Nachteile: WebView Applikationen haben ohne Erweiterung keinen Zugriff auf Hardware Komponenten und softwareseitig auf Push Notifications, Systeminteraktivität etc.

Native Applikationen

Native Applikationen werden für eine spezifische Plattform entwickelt in der jeweiligen systemtypischen Programmiersprache wie zum Beispiel Java, Kotlin, Swift, Objective-C.

Sie werden direkt auf dem System installiert und ausgeführt.

Native Applikationen

Vorteile: Native Applikationen sind sehr schnell, da sie für das jeweilige System spezifisch entwickelt werden.

Schnellerer und leichterer Zugriff auf Dienstprogramme und Hardware Komponenten.

Können Lokal eine große Datenmenge speichern.

Native Applikationen

Nachteile: Native Applikationen haben einen hohen Entwicklungsaufwand und Kostenaufwand, da für die Entwicklung auf mehreren Plattformen mehrere Entwicklerteams benötigt werden.

Die nötigen Programmiersprachen sind schwerer erlernbar als die Webtechnologien.

Hybride Applikationen

Hybride Applikationen werden mit einer für das Framework spezifizierten Sprache geschrieben. Durch das Framework wird der geschriebene Code in native Elemente übersetzt. Bei hybriden Applikationen lassen sich Web Technologien und native Technologien besonders einfach verbinden.

Hybride Applikationen

Vorteile: Durch die Übersetzung des Quellcodes in native Komponenten wird in der Applikation ein natives Look and Feel erzeugt. Außerdem kann die Performance durch verwendete Engines besser sein als bei nativen Applikationen.

Hybride Applikationen

Nachteile: Sehr großer Datenaufwand bei der Erstellung. Eingeschränkte Individualität gegenüber klassischer Web Technologien. Sie benötigen vergleichsweise viel Arbeitsspeicher.

Zusammenfassung

	1 WEB APPLIKATIONEN	2 NATIVE APPLIKATIONEN	3 HYBRIDE APPLIKATIONEN	4 WEBSEITE
Kosten	€ €	€ € € €	€ € €	€
Performance	★	★ ★ ★	★ ★	nicht vergleichbar
Plattformunabhängigkeit	★	✖	★ ★	★ ★ ★
Gerätefunktionen	★	★ ★ ★	★ ★	nicht vergleichbar
AppStore	✓	✓	✓	✖
Offline-Fähigkeit	✓	✓	✓	✖

Frameworks

Electron

Electron wird seit **2013** von **GitHub inc**. entwickelt und ist **Open Source**.

Auf der Basis von **Chromium** und **node.js** werden cross platform development Applikationen mittels **HTML, CSS, JavaScript** entwickelt.

Bekannte, unter Electron, laufende Apps sind u.a. Skype, Discord und Atom.

Frameworks

NativeScript

NativeScript ist ein seit **2015** von Telerik by Progress entwickeltes **Open Source Framework** zur Erstellung von **iOS** oder **Android** Anwendungen. Die grafische Oberfläche wird mittels XML definiert und anschließend in die jeweilige Betriebssystem spezifische Sprache übersetzt. Die Logik wird mittels **JavaScript** und **TypeScript** sowie Angular implementiert.

Frameworks

Apache Cordova

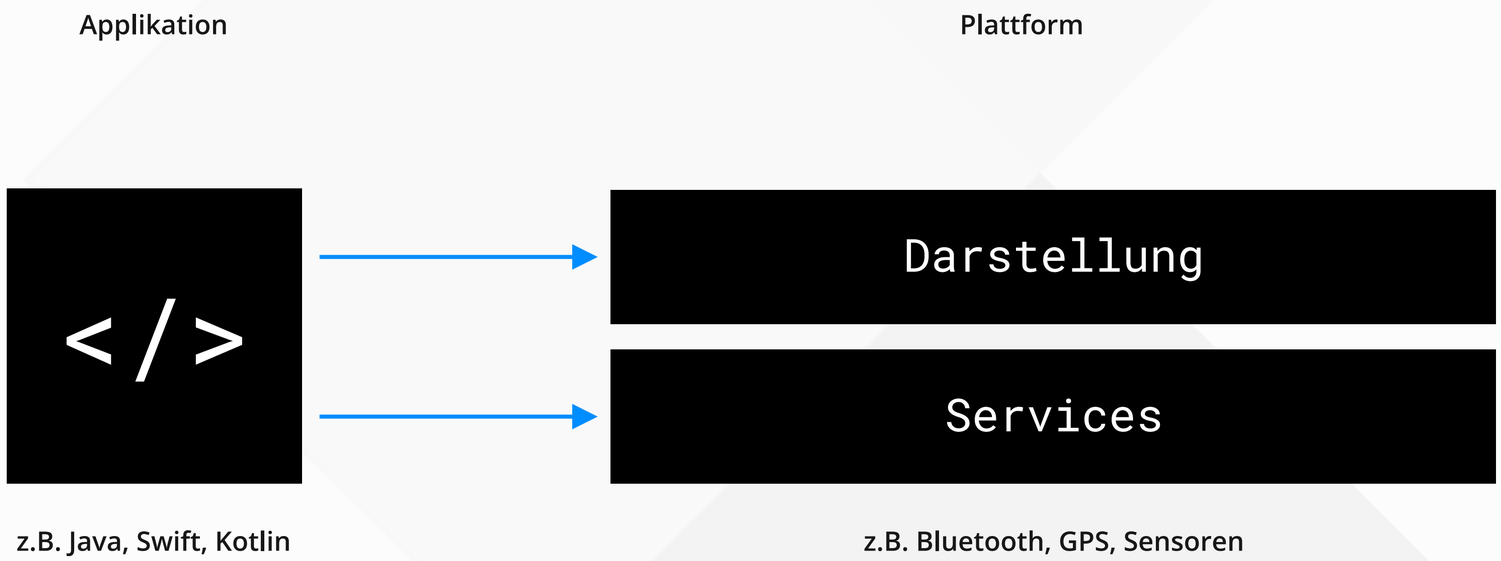
Apache Cordova ist ein im **Jahre 2009** von Nitobi entwickeltes Framework zur Programmierung von Mobilen Anwendungen mittels **HTML, CSS** und **JavaScript.** Apache Cordova wurde 2011 von Adobe gekauft und als Adobe PhoneGap vermarktet. Später wurde Cordova als Open Source Projekt veröffentlicht. Cordova erweitert JavaScript um Funktionen zur **Interaktion mit dem Betriebssystem.**

Frameworks

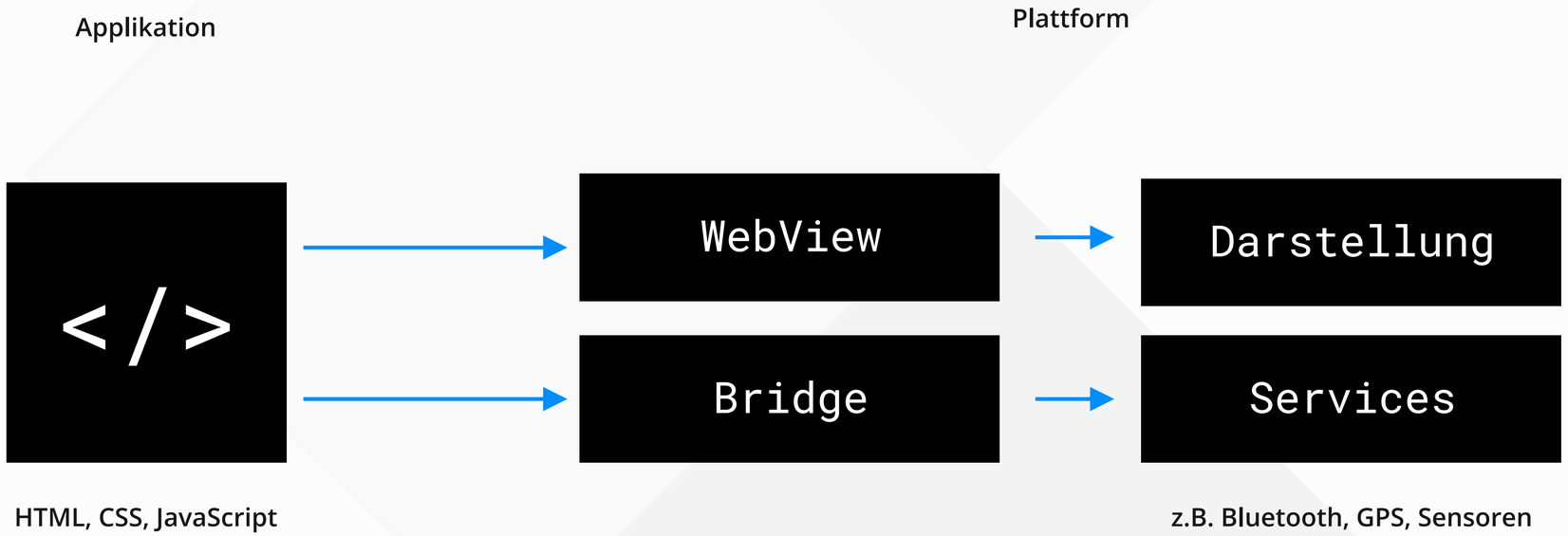
ionic

IONIC ist ein seit **2013** entwickeltes auf **Angular** basierendes Framework zur Programmierung von Applikationen auf jeder Plattform. IONIC basiert auf Cordova und erweitert dieses Framework speziell um weitere **Frontend Technologien** wie zum Beispiel Gestensteuerung und UI Elemente. IONIC selbst ist ein npm-Modul und läuft über node.js.

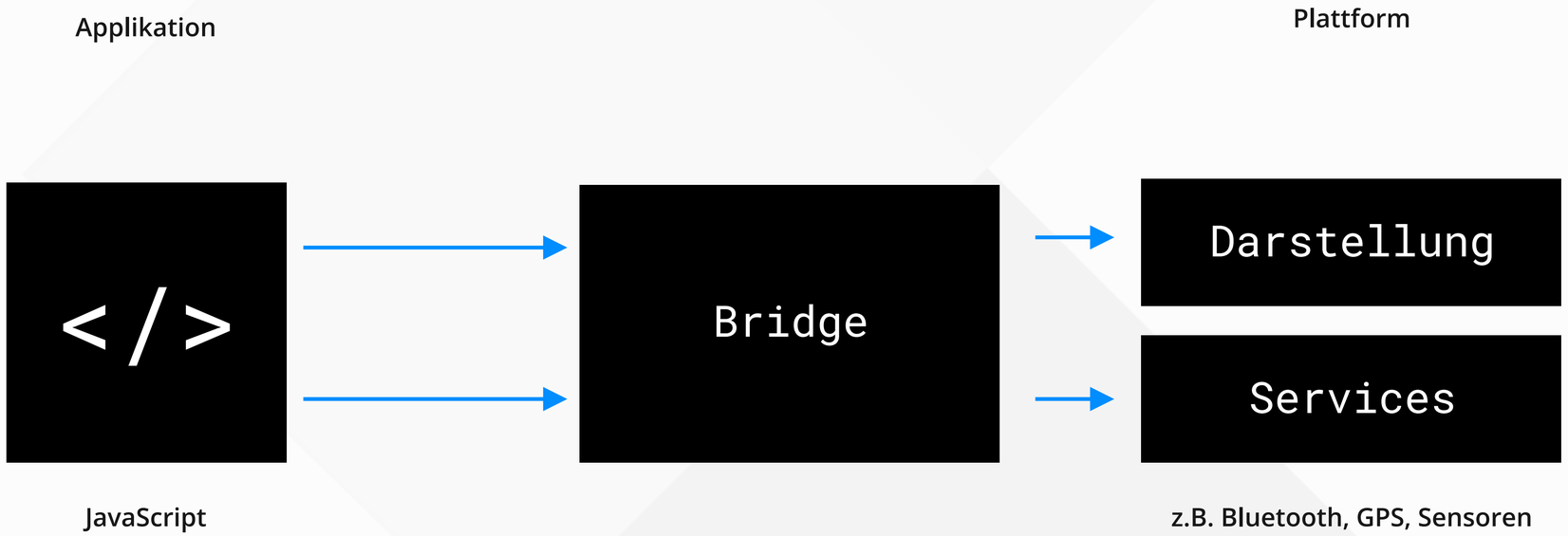
Wie sind Native Applikationen aufgebaut?



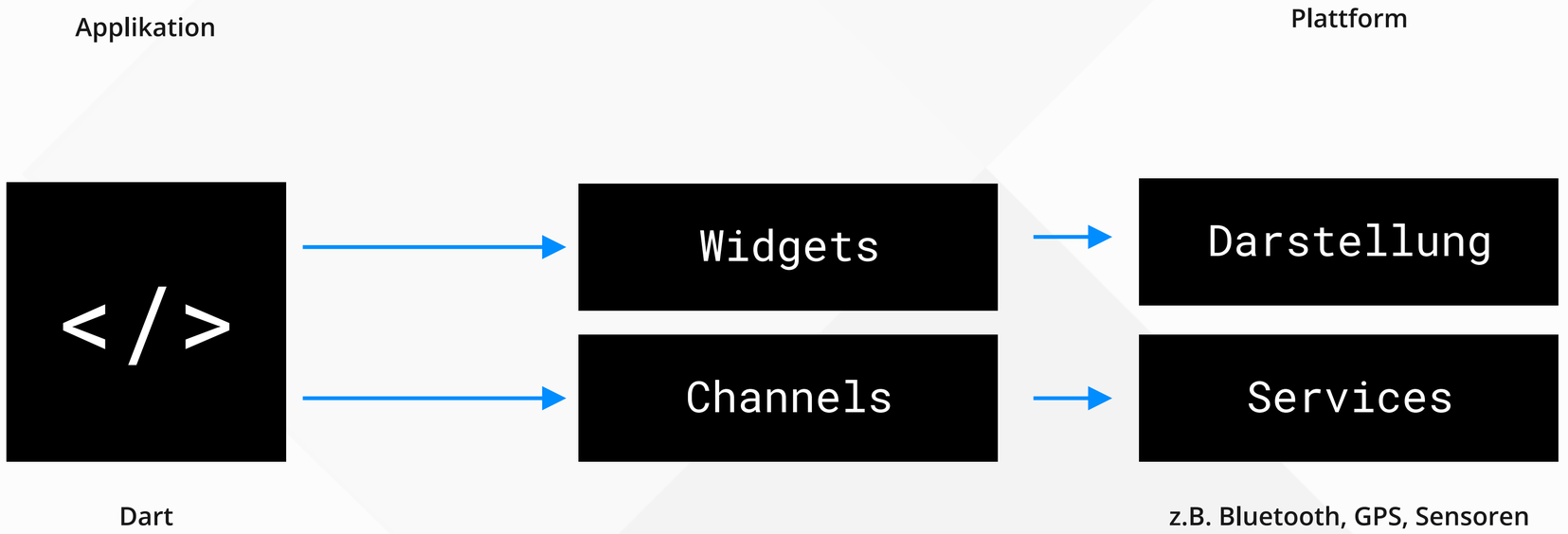
Wie sind Web Applikationen aufgebaut?

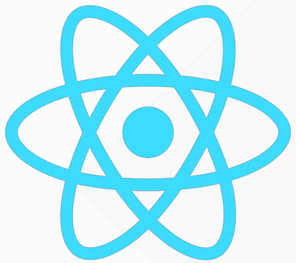


Wie sind React Native Applikationen aufgebaut?



Wie sind Flutter Applikationen aufgebaut?





React Native

vs.

Flutter

		
	React Native	Flutter
Release	2015	2018
Entwickler	Facebook	Google
Darstellungssprache	proprietär (ähnlich XML)	Dart
Programmiersprache	JavaScript	Dart
Popularität (SO Questions *15.06.2019*)	69.678	37.204
Popularität p.a.	17.419	37.204
Kosten	Kostenlos	Kostenlos

Performance

React Native

Erreicht nativnahe Performance mit bis zu 60 Frames per Second.

Flutter

Kann durch die Skia-2d-Engine auch bis zu 60 Frames per Second und mehr erreichen und ist damit in bestimmten Tätigkeiten schneller als native Applikationen.

Developer Experience

React Native

Da React Native auf React aufbaut, ist der Einstieg für erfahrene Entwickler sehr leicht. Außerdem wird React Native mit JavaScript geschrieben, was den Umstieg umso einfacher macht.

Flutter

Da Flutter von Google entwickelt wird, ist die Integration in Android Services ein wenig einfacher als bei React Native. Es gibt allerdings auf Grund des Alters noch vergleichweise wenig Bibliotheken/Erweiterungen für Flutter.

Developer Experience

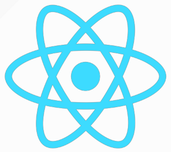
Flutter

```
01 import ‚package:flutter/material.dart‘
02
03 void main() => runApp(HelloFlutter());
04
05 class HelloFlutter extends StatelessWidget {
06   @override
07   Widget build(BuildContext context) {
08     return MaterialApp(
09       home: Scaffold(
10         body: Row(
11           children: [
12             Text(‚Welcome to Flutter.‘),
13             Text(‚Hello World‘)
14           ],
15         ),
16       ),
17     );
18   }
19 }
```

React native

```
01 import React, { Component } from „react“;
02 import { Text, View } from „react-native“;
03
04 class HelloReactNative extends Component {
05   render() {
06     return (
07       <View>
08         <Text>Welcome to React Native!</Text>
09         <Text>Hello World</Text>
10       </View>
11     );
12   }
13 }
```

Zusammenfassung



baut auf React auf

bietet schnelle Performance

liefert plattformnahes Look and Feel

große Community

kompakter Code

viele Open Source Ressourcen

komplett neuer Ansatz

schneller als React Native

teilweise schneller als Native

liefert plattformnahes Look and Feel

stetig wachsende Community

Quellen

01 https://techbeacon.com/app-dev-testing/web-native-mobile-app-frameworks-how-sort-through-choices

02 https://medium.com/flutter-community/react-native-or-flutter-which-should-i-choose-48567ae2e5e1

03 https://hackernoon.com/getting-started-with-cross-platform-app-development-in-2019-dd2bf7f6161b

04 https://www.cabotsolutions.com/react-native-vs-flutter-which-is-better-for-cross-platform-mobile-app-development

05 https://medium.com/coding-blocks/part-1-the-road-to-cross-platform-frameworks-d6a193b9ce2d

06 https://www.theseus.fi/bitstream/handle/10024/146232/thesis.pdf?sequence=1

07 https://subscription.packtpub.com/book/application_development/9781789344967/1/ch01lvl1sec12/comparing-flutter-to-existing-frameworks

08 https://medium.com/flutter-community/react-native-or-flutter-which-should-i-choose-part-two-3950ac273492


LIVE

Kurzanleitung React Native App

1. Den Expo Client auf dem Handy installieren.

   AppStore: https://apps.apple.com/app/apple-store/id982107779

   PlayStore: https://play.google.com/store/apps/details?id=host.exp.exponent&referrer=www
2. Sicherstellen, dass Node.js global auf dem Rechner installiert ist
3. `npm install create-react-native-app`
4. neues ReactNative Projekt mit `create-react-native-app` *`ProjectName`* erzeugen
5. Projekt Attribute und Template wählen
6. ReactNative fragt ob Expo-cli über npm installiert werden soll. Wenn die Installation fehlschlägt alternativ über yarn versuchen
7. Expo Client öffnet sich im Browser in der GUI
8. QR Code scannen (bei Android mit dem in der Expo App integrierten QR-Code scanner / bei iOS über die iPhone Kamera im Foto modus - Push benachrichtigung mit „In Expo öffnen“ sollte erscheinen
9. JavaScript Code in App.js beliebig verändern