

# A JS APPROACH INTO MOBILE APP DEVELOPMENT AND REACT NATIVE

Nikos Kleidis

Frontend developer @ Logicea



@nikoskleidis





# LOGICEA

A PEOPLE-FIRST TECHNOLOGY COMPANY

**BASED IN ATHENS**

**HERAKLIO BRANCH OPENED THIS SUMMER**

Always looking for new passionate people on

- Frontend dev - we just love React
- Backend dev - Java and Kotlin are our flavours
- Designer - Getting creative with UI/UX
- DevOps - Making sure everything runs smoothly



# I HAVE A DREAM!

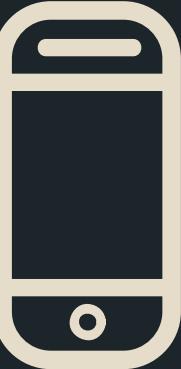
We will build a mobile app that will  
{dream.idea}!

And we will make {dream.money}\$\$\$\$

But I am just a web developer...



# AS A WEB DEVELOPER



## NOT SO SKILLED ON SOME AREAS

I don't know Java or Kotlin

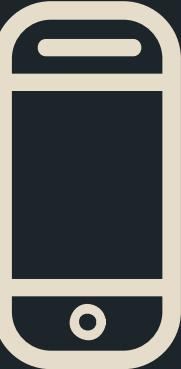
I don't know C# or Swift

I don't know how the inner mechanics of the mobiles

I don't know how to talk to hardware



# AS A WEB DEVELOPER



## BUT I AM SKILLED ON SOME OTHER

I know how to build user interfaces

I am experienced on great tools - eg. Chrome console

I know the cross platform pains - Internet explorer

I can be creative with animations



**Give me a sec to check what is out there...**



# PWA

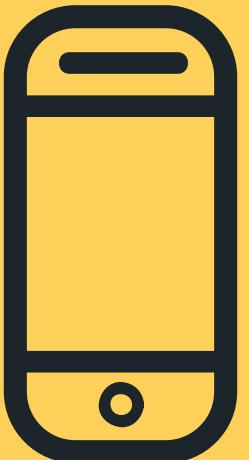
PROGRESSIVE WEB APP



**PWA.**



# PWA PROS



## FAST

Accelerated page load

## INTEGRATED

Feels more app like

## DEVELOPMENT SAVINGS

Runs on both iOS and Android

## RELIABLE

Can provide offline access

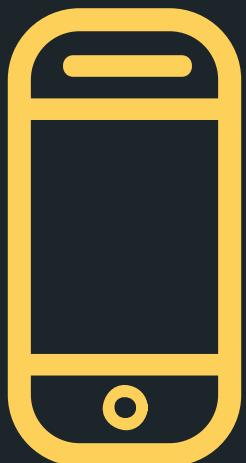
## ENGAGING

Can send push notifications

## SELF-UPDATES

Doesn't go through the stores

# PWA CONS



## LESS RESPONSIVE

Than native apps

## NOT DISCOVERABLE

No presence on the stores

## HIGH BATTERY USAGE

Drains a device battery faster than a native app

## UNEXPECTED BEHAVIOUR

Deletes user cached files after a few weeks of inactivity

## LIMITED HARDWARE ACCESS

Hardware features like Bluetooth, NFC and Face ID are not available or limited

# HYBRID

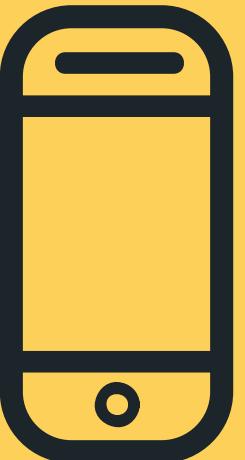
A LITTLE BIT OF BOTH WORLDS

HYBRID



NATIVE+HTML

# HYBRID PROS



## HARDWARE ACCESS

Using third party wrappers

## SEARCHABLE

Store presence

## DEVELOPMENT SAVINGS

Runs on both iOS and Android

## DEVELOPER EXPERIENCE

Most of the testing is done on a browser

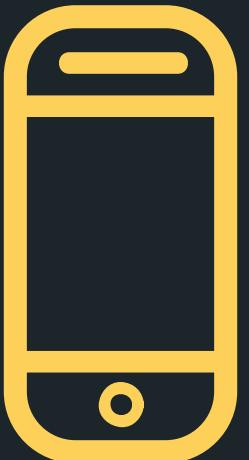
## EXPECTED VISUAL BEHAVIOUR

WebView under the hood

## EASIER UPDATE PROCESS

HTML, CSS updates can be updated on the fly

# HYBRID CONS



## LESS RESPONSIVE

Than native apps

## SLOWER

Building on top of WebView comes with a cost

## DIFFERENT DESIGN

Android and iOS require different approaches when it comes to meeting the user needs

## NEED FOR NATIVE APP DEVELOPERS

Platform specific problems require platform specific experts!

# NATIVE-LIKE

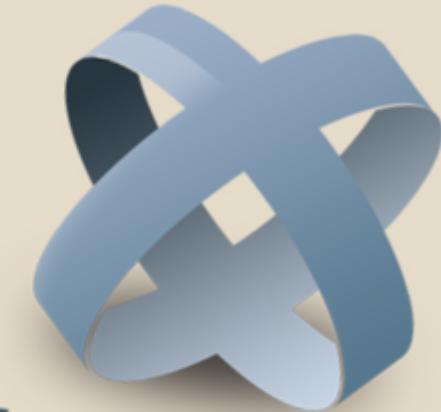
NO WEBVIEW!



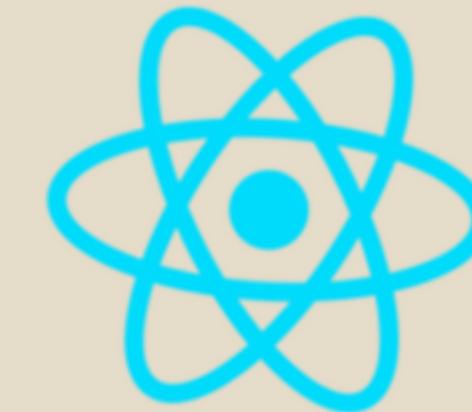
Tabris



NativeScript



titanium™



React Native

# NATIVE-LIKE APPS

## NO WEB VIEW NEEDED

Apps render native components depending on the platform they are deployed in

## CUSTOM COMPONENTS

Each framework exposes its own components that devs use like HTML elements

## THE JS WE KNOW

Most of the application logic is written in JS. Platform specific code is limited

## A BRIDGE IS REQUIRED

JS code needs to communicate with native code. A bridge is required for this purpose

```
function App () {  
  return (  
    <View style={styles.app}>  
      <View style={styles.header}>  
        <Image  
          accessibilityLabel="React logo"  
          source={{ uri: logoUri }}  
          resizeMode="contain"  
          style={styles.headerLogo} />  
    </View>  
  )  
}
```

# React Native



## POWERED BY FACEBOOK

#deleteFacebook

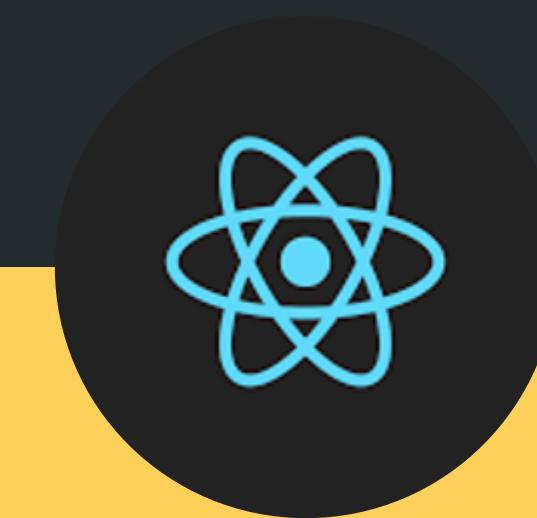
Started in an internal hackathon in 2013. Released to the public at March 2015



## STRONG COMMUNITY

GitHub stats

- > 2k Contributors
- > 80k Stars
- > 275k Used by
- > 18k Commits



## POPULAR APPS

Instagram

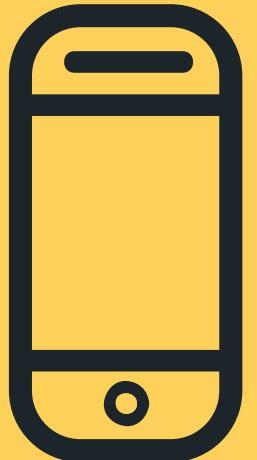
Skype

Walmart

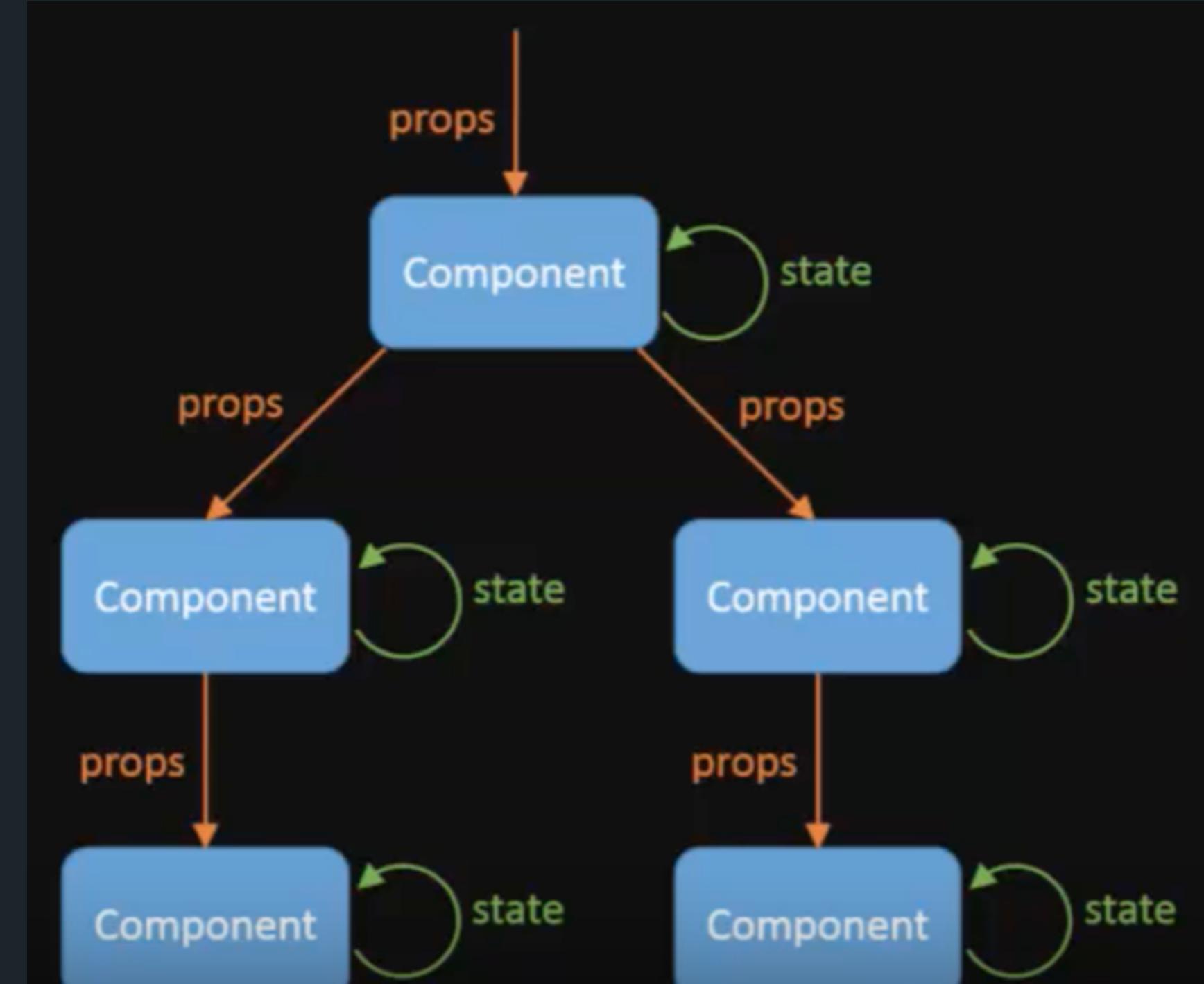
Bloomberg

Baidu

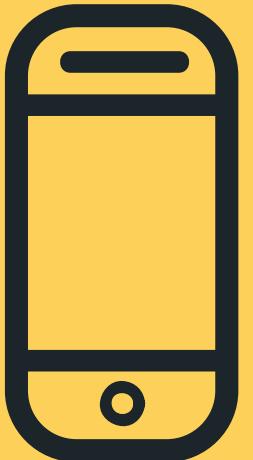
# REACT NATIVE DATA FLOW



Unidirectional data flow  
 $ui = fn(state)$



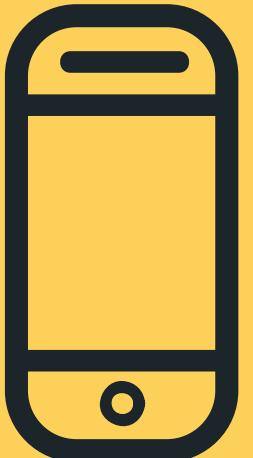
# REACT NATIVE ELEMENTS



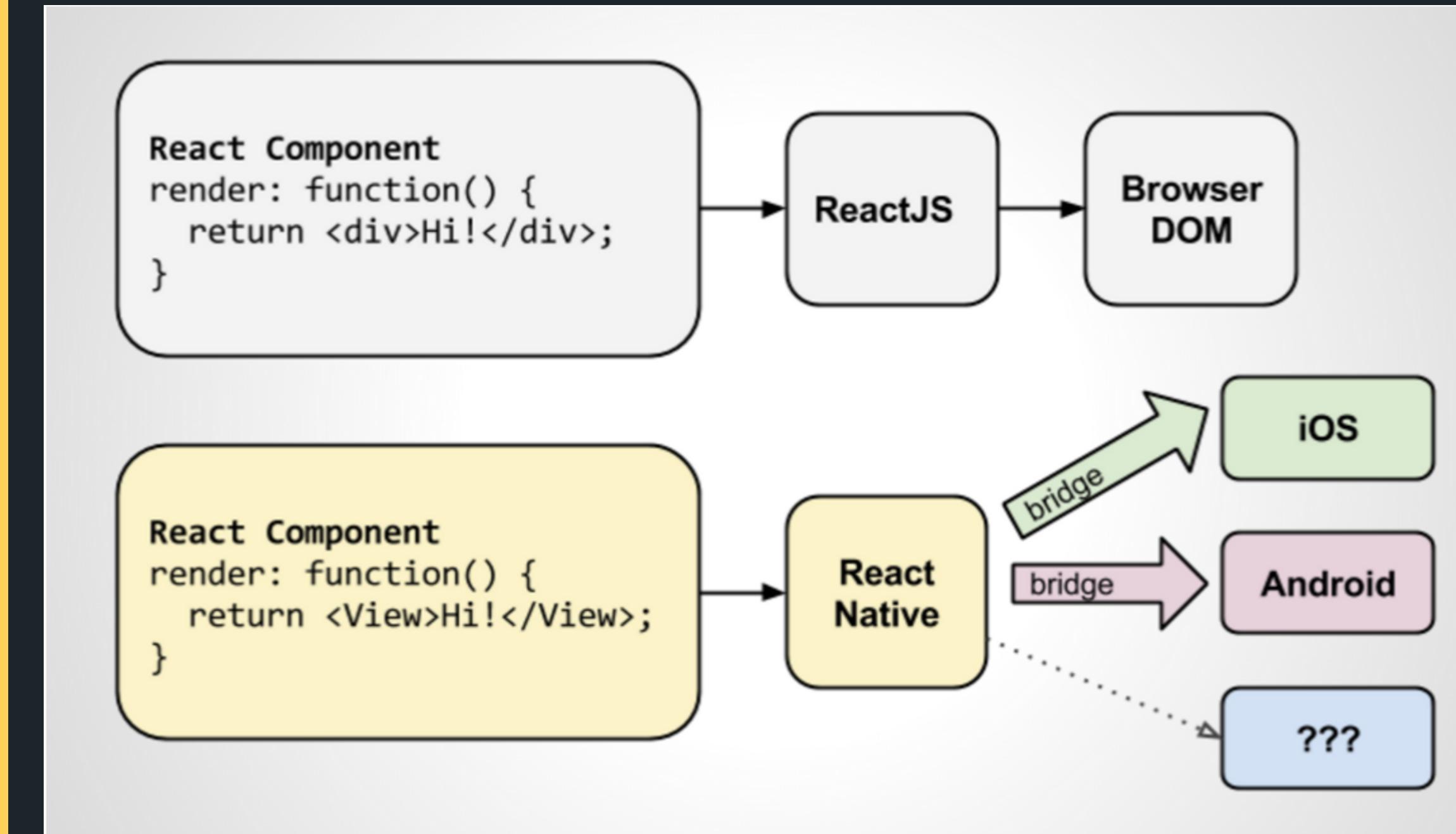
Elements are similar to HTML

- <div>
- <img>
- <span>, <p>
- <ul>, <ol>
- <View>
- <Image>
- <Text>
- <ListView>

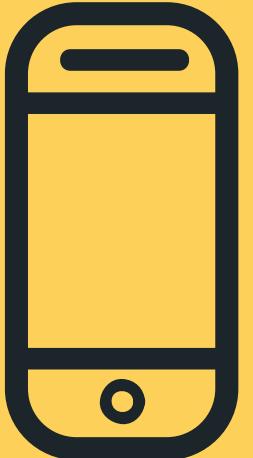
# REACT NATIVE RENDERING



Depends on the platform



# REACT NATIVE JSX

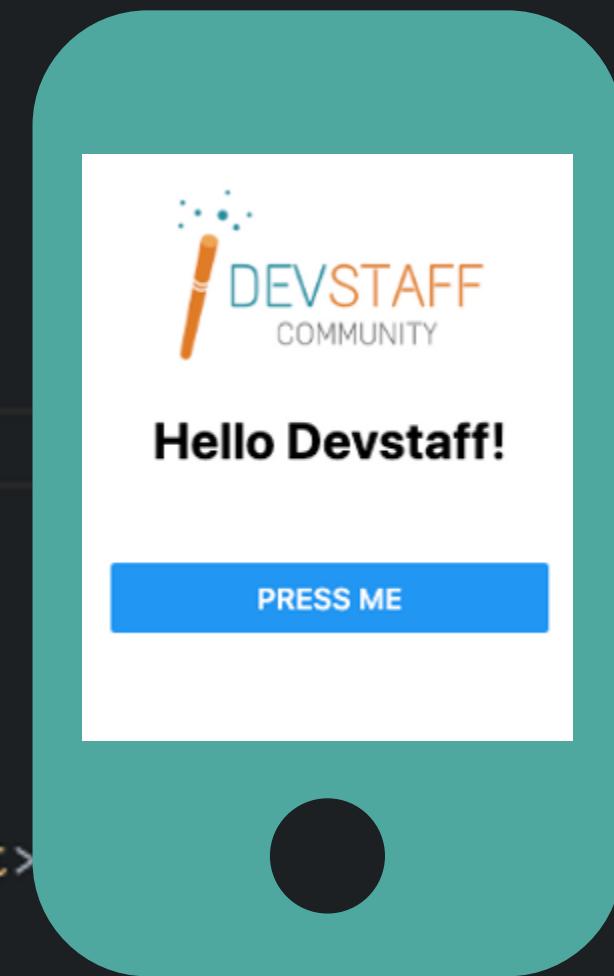


## Example

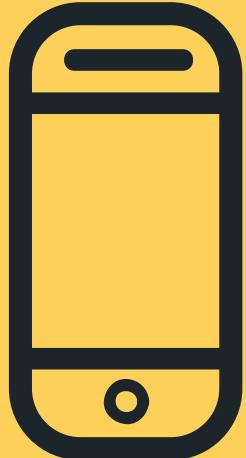
```
import React from "react";
import { Button, Image, StyleSheet, Text, View } from "react-native";

const logoUri = `data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAxMAAAG0CAYAAABJ...`;
```

```
function App () {
  return (
    <View style={styles.app}>
      <View style={styles.header}>
        <Image
          source={{ uri: logoUri }}
          resizeMode="contain"
          style={styles.logo}
        />
        <Text style={styles.title}>Hello Devstaff!</Text>
      </View>
      <Button onPress={() => {}} title="Press me" />
    </View>
  );
}
```



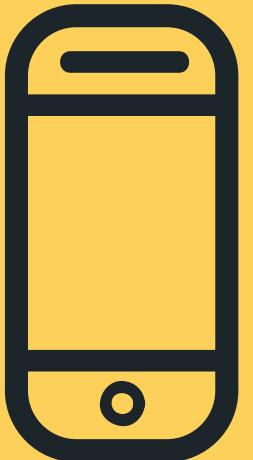
# REACT NATIVE STYLING



Based on Flexbox

```
22 const styles = StyleSheet.create({
23   app: {
24     marginHorizontal: "auto",
25     maxWidth: 500,
26     flex: 1,
27     flexDirection: "column",
28     justifyContent: 'flex-start'
29   },
30   logo: {
31     height: 80
32   },
33   header: {
34     padding: 20
35   },
36   title: {
37     fontWeight: "bold",
38     fontSize: "1.5rem",
39     marginVertical: "1em",
40     textAlign: "center"
```

# REACT NATIVE NETWORK



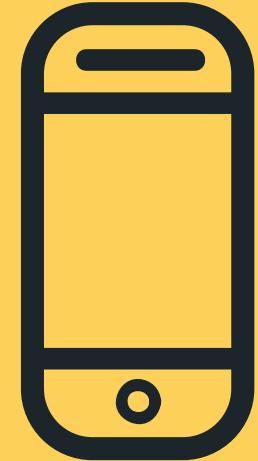
## New fetch API supported

```
async function getMoviesFromApi() {  
  try {  
    let response = await fetch(  
      'https://facebook.github.io/react-native/movies.json',  
    );  
    let responseJson = await response.json();  
    return responseJson.movies;  
  } catch (error) {  
    console.error(error);  
  }  
}
```

→ Raw XHR implemented if required

→ WebSockets are supported

# REACT NATIVE STORAGE



## Async storage

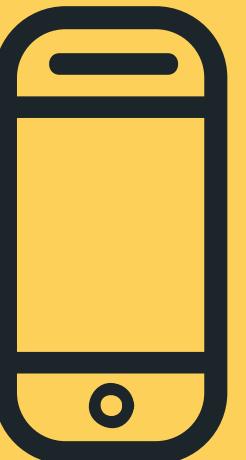
@react-native-community/async-storage

```
const storeData = async () => {
  try {
    await AsyncStorage.setItem('@devstaff_topic', 'Mobile app dev')
  } catch (e) {
    // saving error
  }
}
```

→ Is asynchronous

→ Data is stored on the device

# REACT NATIVE DEBUGGING



## React Dev Tools

The screenshot shows the React Developer Tools window. At the top, there's a search bar and two checkboxes: "Highlight Updates" and "Highlight Search". Below the search bar is a tree view of the rendered UI components. A blue selection bar highlights a specific component: `<SectionList ItemSeparatorComponent=ItemSeparator()`. To the right of the tree view, the component's props are listed in a detailed pane. The props include various configuration options like `enableEmptySections: true`, `getItem: getItem()`, and `sections: Array[2]`. At the bottom of the tools window, there's a "React Native Style Editor" section with a color picker set to `#eeeeee`.

Props

- ItemSeparatorComponent: `ItemSeparator()`
- automaticallyAdjustContentInsets: `false`
- contentContainerStyle: `{...}`
- data: `Array[0]`
- disableVirtualization: `false`
- ✓ enableEmptySections: `true`
- getItem: `getItem()`
- getItemCount: `getItemCount()`
- horizontal: `false`
- initialNumToRender: `10`
- itemShouldUpdate: `_itemShouldUpdate()`
- keyExtractor: `keyExtractor()`
- keyboardDismissMode: `"on-drag"`
- keyboardShouldPersistTaps: `"handled"`
- legacyImplementation: `false`
- maxToRenderPerBatch: `10`
- onEndReachedThreshold: `2`
- renderItem: `fn()`
- renderScrollIndicator: `renderScrollIndicator()`
- renderSectionHeader: `renderSectionHeader()`
- scrollEventThrottle: `50`
- sections: `Array[2]`
- ✓ stickySectionHeadersEnabled: `true`
- style: `27`
- updateCellsBatchingPeriod: `50`
- windowSize: `21`

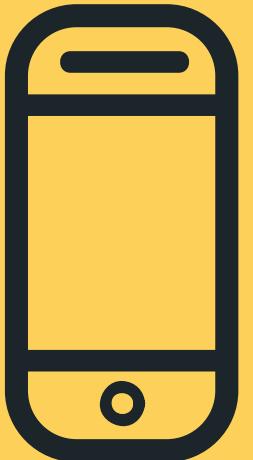
React Native Style Editor

backgroundColor : #eeeeee

→ Can view rendered tree

→ Inspect element's props and state

# REACT NATIVE DEBUGGING



## Chrome Dev Tools

```
Console was cleared
Running application "ReactNativeDebugDemo" with appParams: {"initialProps":{}, "rootTag":1}. infoLog.js:17
__DEV__ === true, development-level warning are ON, performance optimizations are OFF
⚠️ Remote debugger is in a background tab which may cause apps to perform slowly. Fix this by foregrounding the tab (or opening it in a separate window). YellowBox.js:71
value1 0
value2 0
result 0
value1 2
value2 2
result 4
```

→ Can console.log

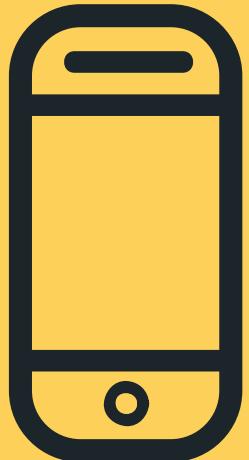
→ Use debugger and pause execution

# REACT NATIVE SYSTEM REQUIREMENTS



- node.js and npm installed
- minimum JDK 8
- Xcode and Xcode Command Line Tools  
for iOS development
- Android Studio with built-in emulator  
for Android development

# REACT NATIVE INSTALLATION



1

```
npm install -g react-native-cli
```

2

```
react-native init DevstaffDemo &&  
cd DevstaffDemo
```

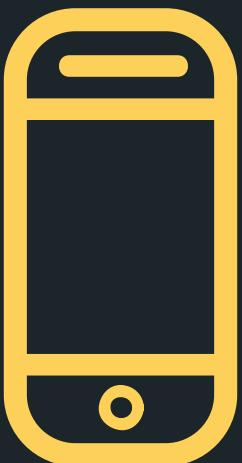
3

```
Open Android emulator and  
react-native run-android  
or  
react-native run-ios — simulator="iPhone X"
```

# REACT NATIVE PITFALLS



# REACT NATIVE PITFALLS



## NOT 100% NATIVE

Performance just isn't like that of native apps

Simple apps would have no problems

Not suitable for heavy animations like games

## NAVIGATION SOLUTIONS

React Navigation is the dominant solution. Deals with platform specific issues. eg header, drawers etc

NOT very customisable

Limited good alternatives

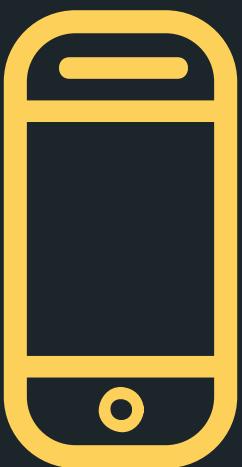
## UPDATE PROCESS

Version to version update is a bit of pain.

Attempts to make it easier:

<https://react-native-community.github.io/upgrade-helper/>

# REACT NATIVE PITFALLS



## CROSS-PLATFORM DOES NOT MEAN SINGLE-PLATFORM

There are platform specific issues to address  
eg elements rendering differently in Android and iOS

```
import {Platform, StyleSheet} from 'react-native';

const styles = StyleSheet.create({
  height: Platform.OS === 'ios' ? 200 : 100,
});
```

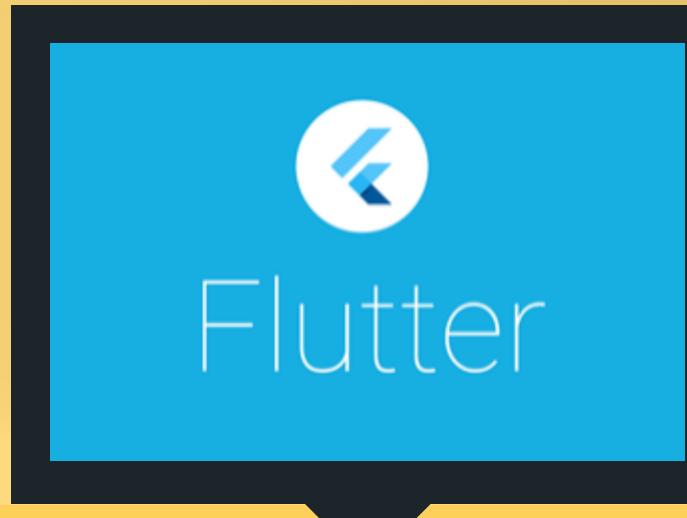
## NEED FOR NATIVE APP DEVELOPERS

Platform specific problems require platform specific experts!

## LARGE APP SIZE

The app size will increase greatly if you use too many third-party libraries and native components.

# Inspired by React

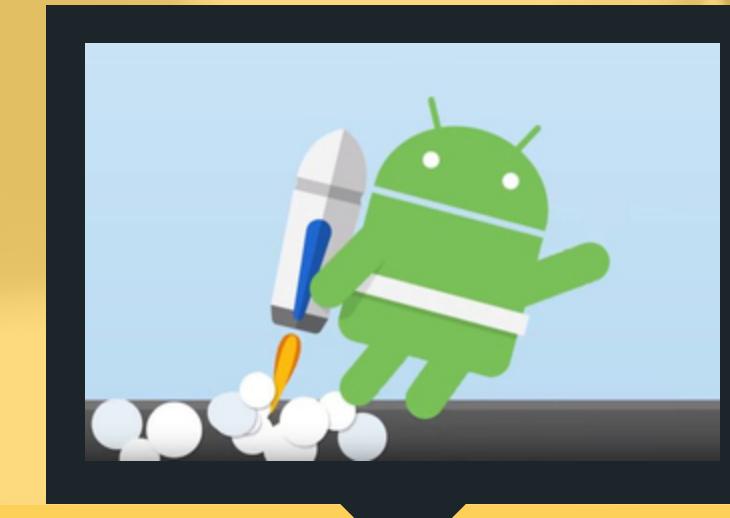


## GOOGLE FLUTTER

UI Framework for cross  
platform apps  
  
Dart used as main language

---

Unidirectional Data Flow  
Hot Module Reload

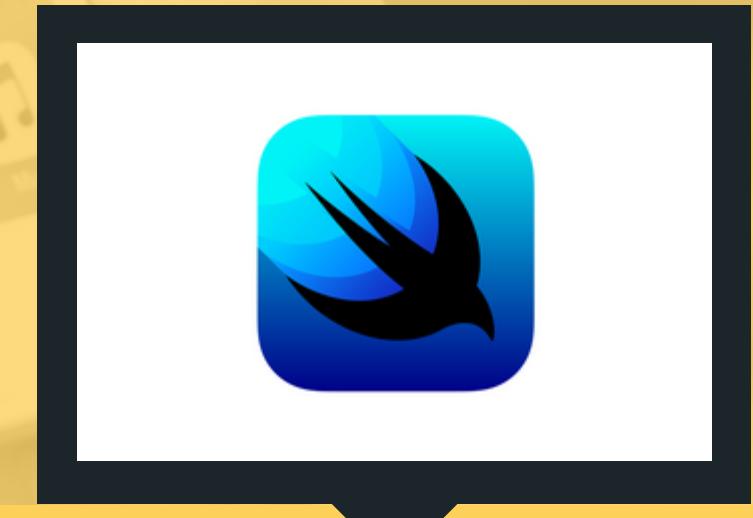


## JETPACK COMPOSE

UI framework for Android apps  
Kotlin used as main language

---

Unidirectional Data Flow  
Event handling similar to React



## SWIFT UI

UI framework for iOS apps  
Swift used as main language

---

Unidirectional Data Flow

# THANK YOU FOR LISTENING

