

React Native

• • •

Konrad Kotelczuk

<https://github.com/kkotelczuk/rn-pb>

Konrad Kotelczuk



Front End developer w Leocode



Wolontariusz Hacklag



Wiking



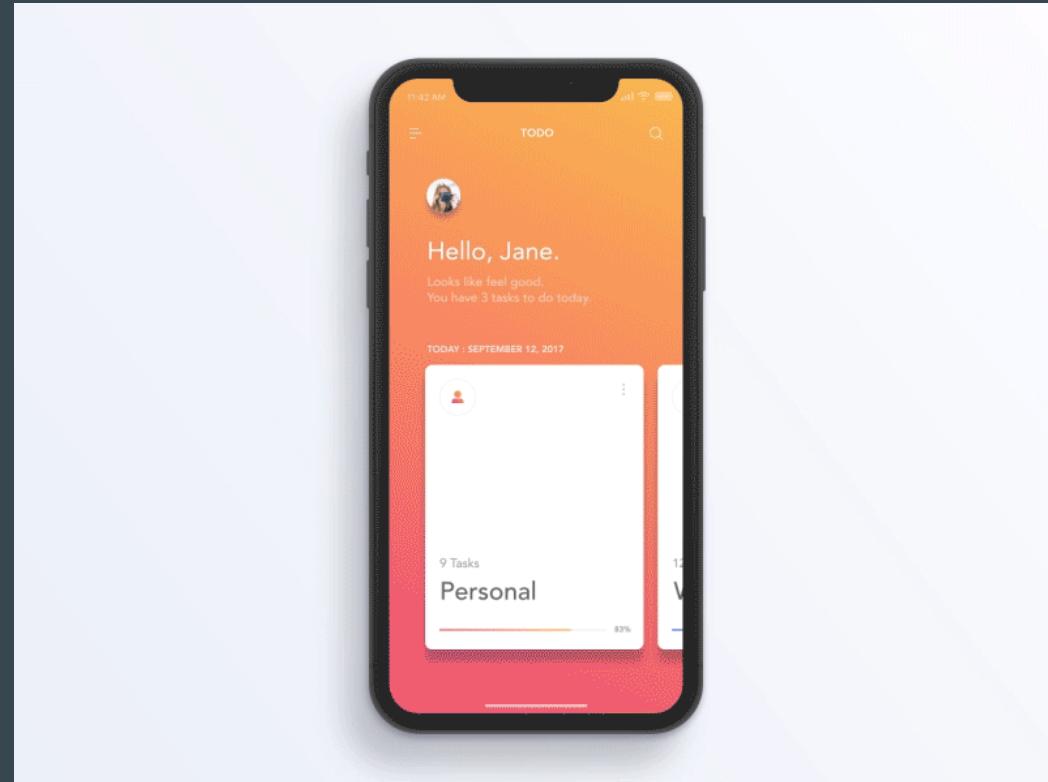
Agenda

- A co to ten React Native
- Wymagania techniczne
- Budowanie aplikacji
- UI w React Native
- Komunikacja z API
- Jak działa React Native
- React Native i Firebase
- Notyfikacje

Aplikacje Mobilne

- iOS

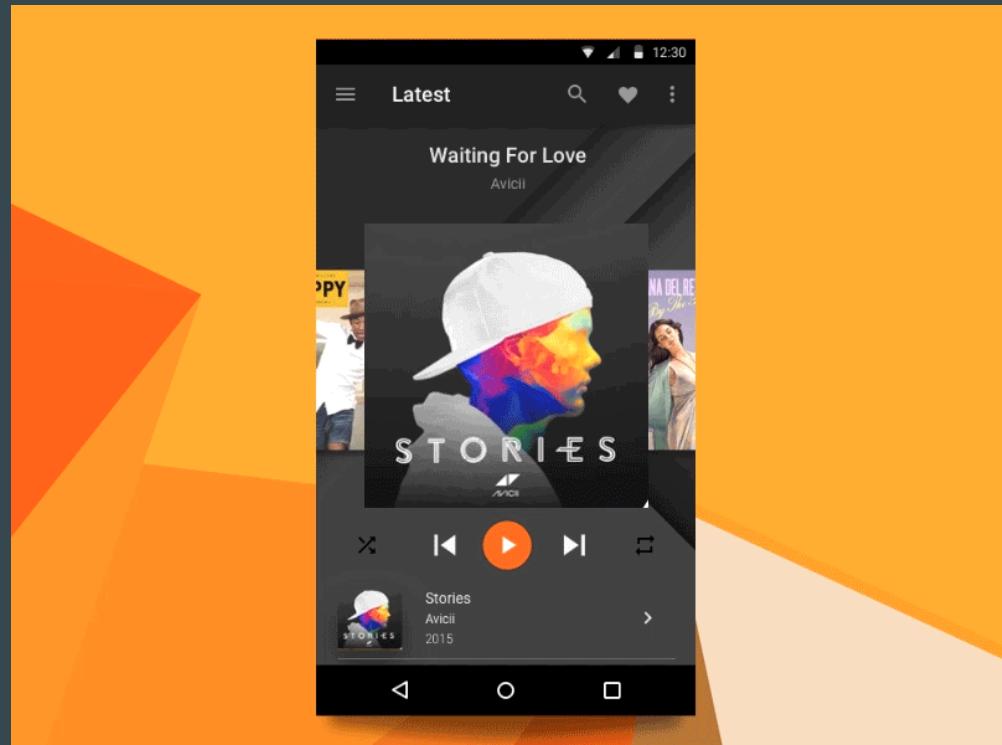
- płynne
- ładne
- notch
- stabilne
- drogie
- iphone X*



Aplikacje Mobilne

- Android

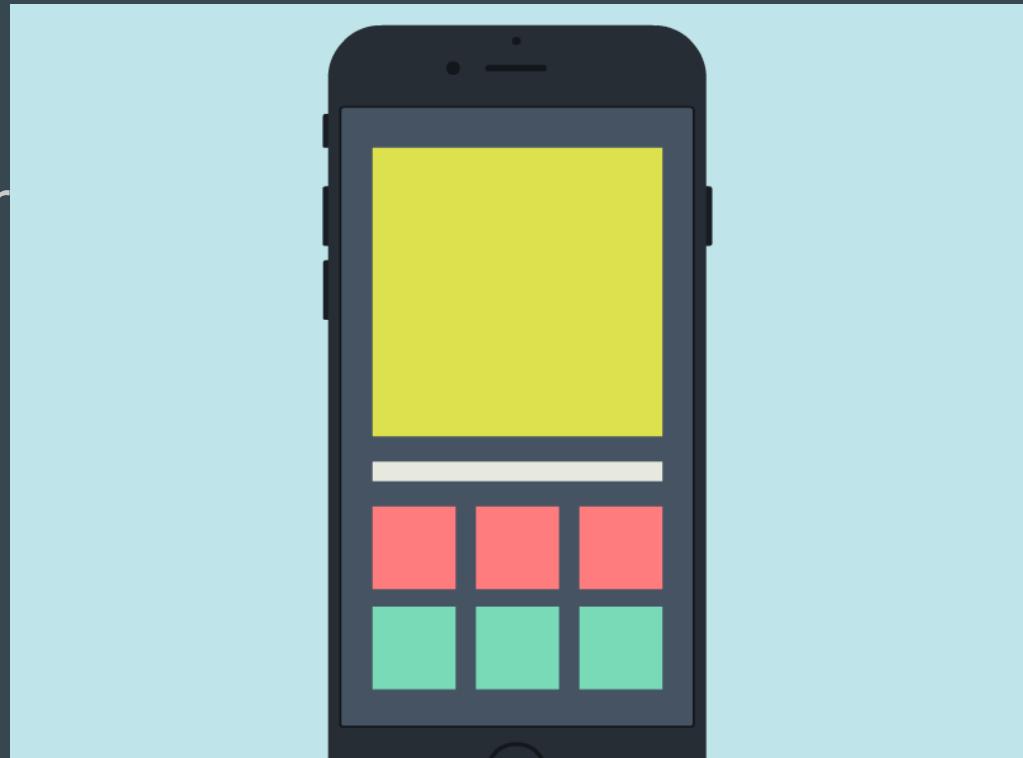
- material design
- wolne
- miljony urządzeń
- xiaomi lepsze
- brzydsze
- nawigacja u dołu
 - lub fizyczna
 - lub gesty
 - lub niewiadomo co



Aplikacje Mobilne

- Windows

- i inne dziwactwa
- znikomy udział w r
- nikt o to nie dba
- brzydkie jak noc
- Nokia?
- To nie działa



Aplikacje Mobilne

iOS => Swift / Objective C

Android => Kotlin / Java

Windows i inne nieistotne technologie => C# itp

Aplikacje hybrydowe

React Native => **JavaScript** + React

Ionic => **JavaScript** + Angular + Cordova

Xamarin => C#

Inne potworki np. Flutter => np. Dart

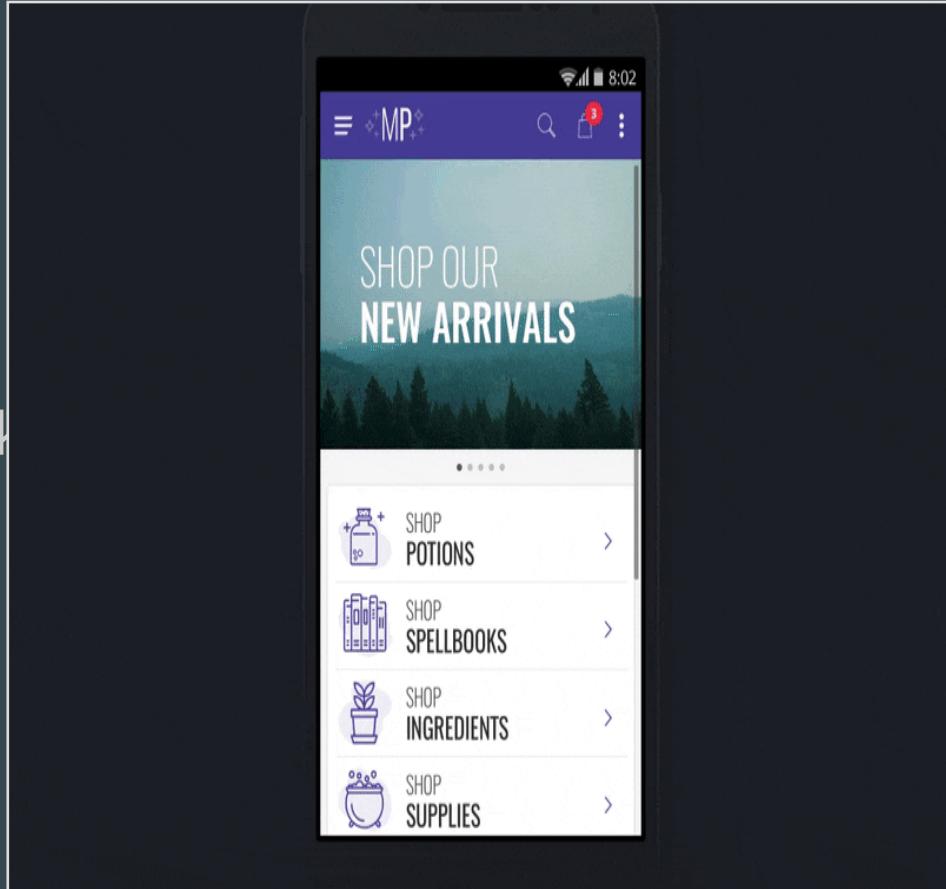
PWA a Native Feel

PWA:

strona internetowa offline

oparta na silniku przeglądarki

ograniczona przez API przeglądarki



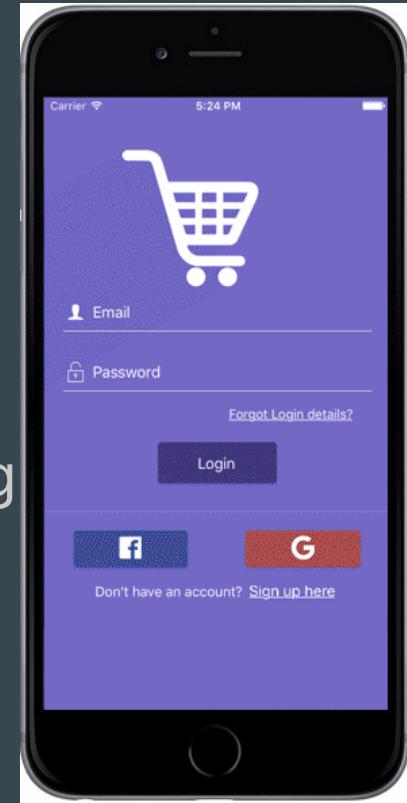
PWA a Native Feel

Native Feel:

płynne animacje

przeciąganie elementów do dołu i na boki

wykorzystanie natywnych elementów nawig



React Native

**Framework do tworzenia aplikacji mobilnych w języku
JavaScript na różne platformy**

Native UI - Material dla Andorid, UIKit dla iOS

Native Components

Native App

React w React Native

React - biblioteka renderująca UI

Virtual Dom / Fiber - algorytm decydujący co ma zostać wyrenderowane

JSX - notacja, pozwalająca pisać HTML w JavaScript

State / Props - zarządzanie komponentami

Do czego użyć React Native?

MVP - Minimum Viable Product

PoC - Proof of Concept

Chat bot / QR codes / BLE / Camera / API viewer

Wymagania

MacOs / Linux / Windows

Jakiś telefon z Androidem lub iOS'em, ale nie za stary

Wiadro cierpliwości

Trochę zacięcia

Nerwy ze stali

Android Studio / Xcode

Symulator a Emulator

Symulator - obrazuje zachowanie software'u w danym środowisku => Xcode

Emulator - replikuje system środowiska => Android Studio

A czy telefon jest potrzebny?

TAK!

Wersja aplikacji

Debug - dostęp do narzędzi developerskich, konsoli, hot reload, source mapy, komunikatów o błędach i ostrzeżeniach, bundler JS

Release - wersja aplikacji przygotowana na konkretną platformę, błąd == zawieszenie lub zamknięcie aplikacji, mniejszy rozmiar, nie potrzebuje bundler'a JS

Expo

Zestaw narzędzi ułatwiających używanie React Native

- + Pozwala budować aplikację na iOS z Windowsa
- + Upraszcza proces uruchamiania i budowania aplikacji
- + Przyspiesza development, gdyż posiada szereg wbudowanych najczęściej używanych bibliotek
- Nie pozwala używać natywnego kodu
- Używa starych wersji bibliotek
- 3rd part library
- Powoduje problemy w utrzymaniu kodu

Tworzymy pierwszą aplikację

I hop do konsoli!

Native Components

`<View>` === `<div>`

`<Text>` === `<p>` / `<div>` / ``

`<Button>` === `<button>`

`<Image>` === ``

React Native posiada blisko 40 wbudowanych komponentów

Native Components Props

style === style

onPress === onClick

source === src

Każdy komponent ma swoją listę Props'ów

Native Components Styling

Flexbox style guide

<https://flexboxfroggy.com/>

React Native PSy nr. 1

<https://facebook.github.io/react-native/docs/getting-started>

If you are familiar with native development, you will likely want to use React Native CLI. It requires Android Studio to get started. If you already have one of these tools installed, you should be up and running within a few minutes. If they are not installed, you should expect to spend about 15 minutes installing and configuring them.

[Expo CLI Quickstart](#)

[React Native CLI Quickstart](#)

Follow these instructions if you need to build native code in your project. For example, if you want to add native components to your application, or if you want to integrate React Native into an existing application, or if you "ejected" from [Expo](#) or Create React Native.

Wykład II

LIVE

breakyourownnews.com

BREAKING NEWS

REACT NATIVE DLA WINDOWS!

17:15

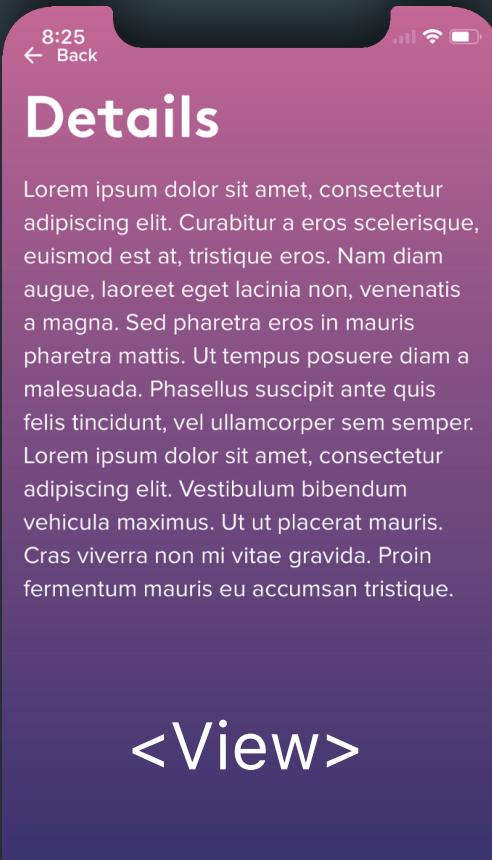
CZY TO NAPRAWDE MOŻLIWE? TAK! JUŻ TERAZ! SZOK I NIE DOWIERZANIE!

React Native for Windows

- Komponenty napisane w C++
- Pisanie aplikacji dla Windows 10, 8 i 7 przy pomocy RN
- Obecna wersja to jest **Dramat!**

[https://github.com/microsoft/react-native-windows/blob/master/vnext/docs/
GettingStarted.md](https://github.com/microsoft/react-native-windows/blob/master/vnext/docs/GettingStarted.md)

Podstawowe komponenty w RN



Podstawowe komponenty w RN - ScrollView

- Niezbędne jest ustalenie wysokości: {flex: 1}
- Renderuje WSZYSTKIE komponenty na raz
- Problemy wydajnościowe

Podstawowe komponenty w RN - FlatList

- Lazy rendering
- Stworzony do renderowania list komponentów
- Wspiera natywne zachowania list obu platform jak pull to refresh

ScrollView vs FlatList

- ScrollView
 - renderuje child components tak jak są
 - wspiera “długie” widoki
- FlatList
 - wymaga przygotowania danych do renderowania
 - Lazy rendering
 - wspiera cross platformowe zachowania list

Podstawowe komponenty w RN - Text

- Służy do wyświetlania tekstu
- Posiada własne propsy do stylowania tekstu
- Wszystkie elementy wewnętrz `<Text>` tracą właściwości Flexbox
- `<Text>` jest również interaktywny

Podstawowe komponenty w RN - Button

- Podstawowy przycisk
- Wygląd jest zależny od platformy
- Minimalne opcje modyfikacji

Podstawowe komponenty w RN - TouchableOpacity

- Wrapper pozwalający reagować widokowi na akcje związane z dotykiem
- Pełne możliwości dopasowania wyglądu i zachowania
- Działa podobnie na wszystkich platformach - w przeciwieństwie do <Button>

Podstawowe komponenty w RN - Image

- Komponent służy do wyświetlania różnych typów obrazów
- React Native nie wspiera natywnie SVG
- React Native dla Androida, nie wspiera natywnie Gifów...
- ...ale jest na to rozwiązanie

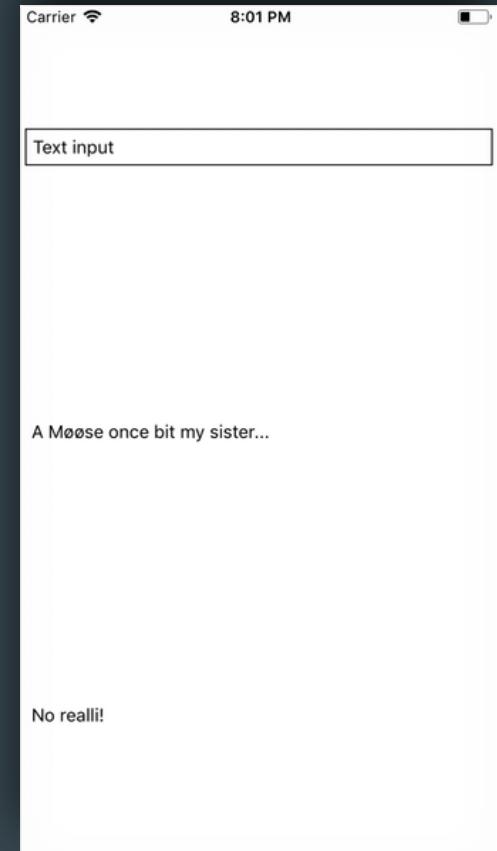


Podstawowe komponenty w RN - TextInput

- Komponent pozwalający na wprowadzanie danych przez użytkownika
- Posiada szereg opcji zwiększających accessibility
- Pozwala kontrolować wygląd i zachowanie klawiatury

Nie do końca podstawowy komponent w RN

- KeyboardAvoidingView - komponent pozwalający na “zwijanie” się widoku na potrzeby pokazania klawiatury na ekranie



Podsumowanie - komponenty

- <View> | <SafeAreaView>
- <ScrollView>
- <FlatList>
- <Text>
- <Image>
- <Button> | <TouchableOpacity>
- <TextInput>

React Native pozostałe komponenty

- Rozszerzają możliwości podstawowych
- Leżą u podstawa podstawowych
- Są używane znacznie rzadziej



React Native Style

- Nazwy atrybutów CSS piszemy camelCase'em
 - **background-colour -> backgroundColor**
- W RN nie działa kaskadowość stylów
- Style mogą być zwykłym obiektem
- Style można łączyć przy pomocy tablic, dzięki temu w obrębie danego komponentu mogą po sobie dziedziczyć

React Native Style

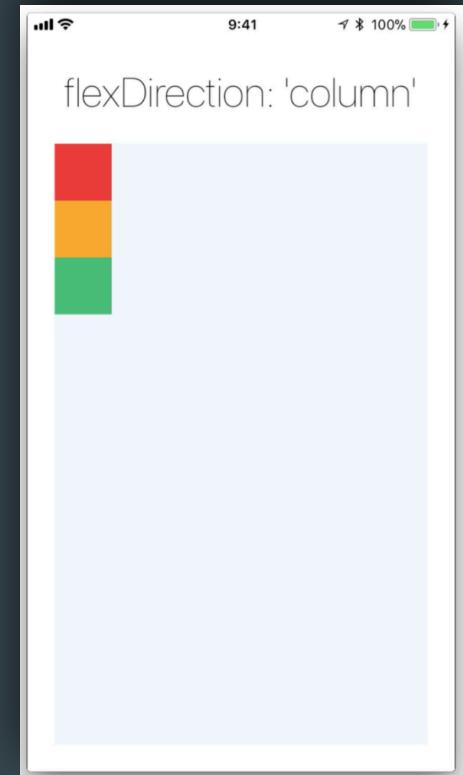
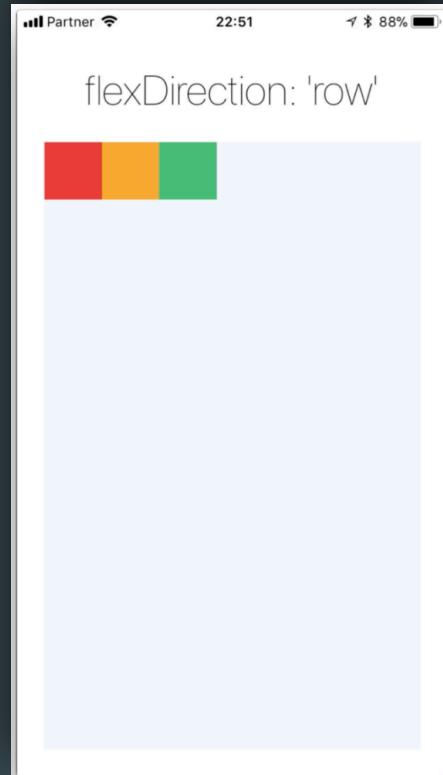
- StyleSheet API - porządkuje kod
 - Teoretycznie: wpływa na szybkość działania aplikacji
- Nie używamy Pixeli, ale po prostu jednostek
- Aplikacja nie zna wymiarów urządzenia
 - Dimensions.get('window').<height | width>
- Flexbox!

Flexbox

- Nie ma potrzeby używania: `display: 'flex'` po prostu nie ma innej opcji

Flexbox - flexDirection

- Domyślny kierunek
to : column



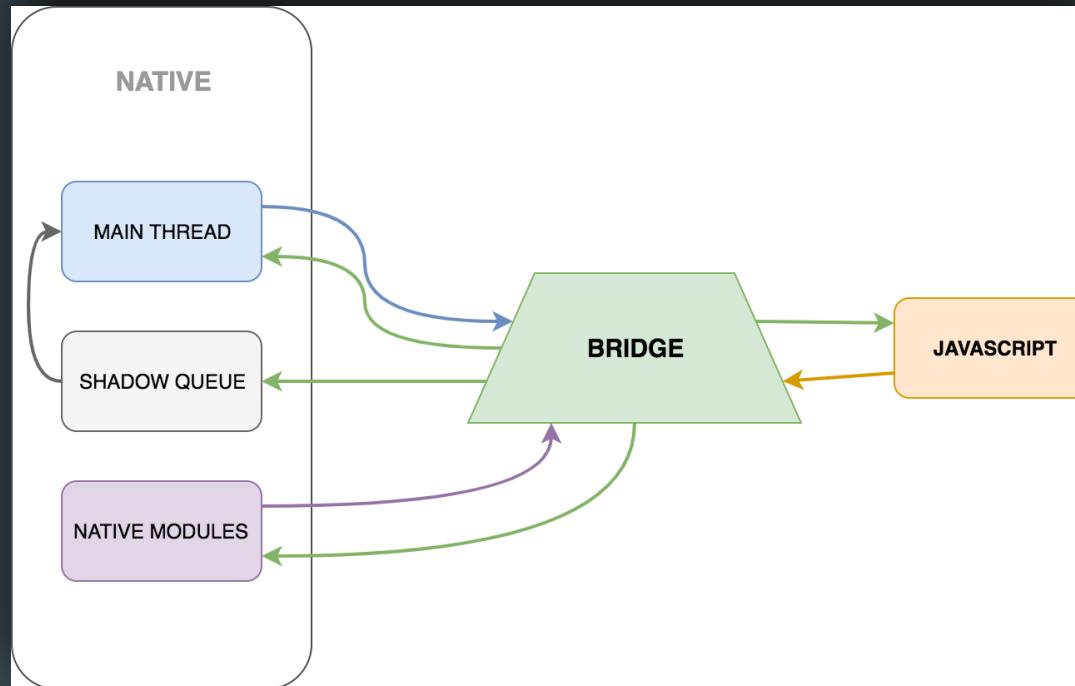
Flexbox

- `flex: <number>` definiuje jak elementy będą walczyć o dostępną przestrzeń na ekranie, zazwyczaj używamy `flex: 1`
- `justifyContent` określa rozłożenie elementów wzdłuż głównej osi (`flexDirection`)
- `alignItems` określa położenie elementów na osi poprzecznej

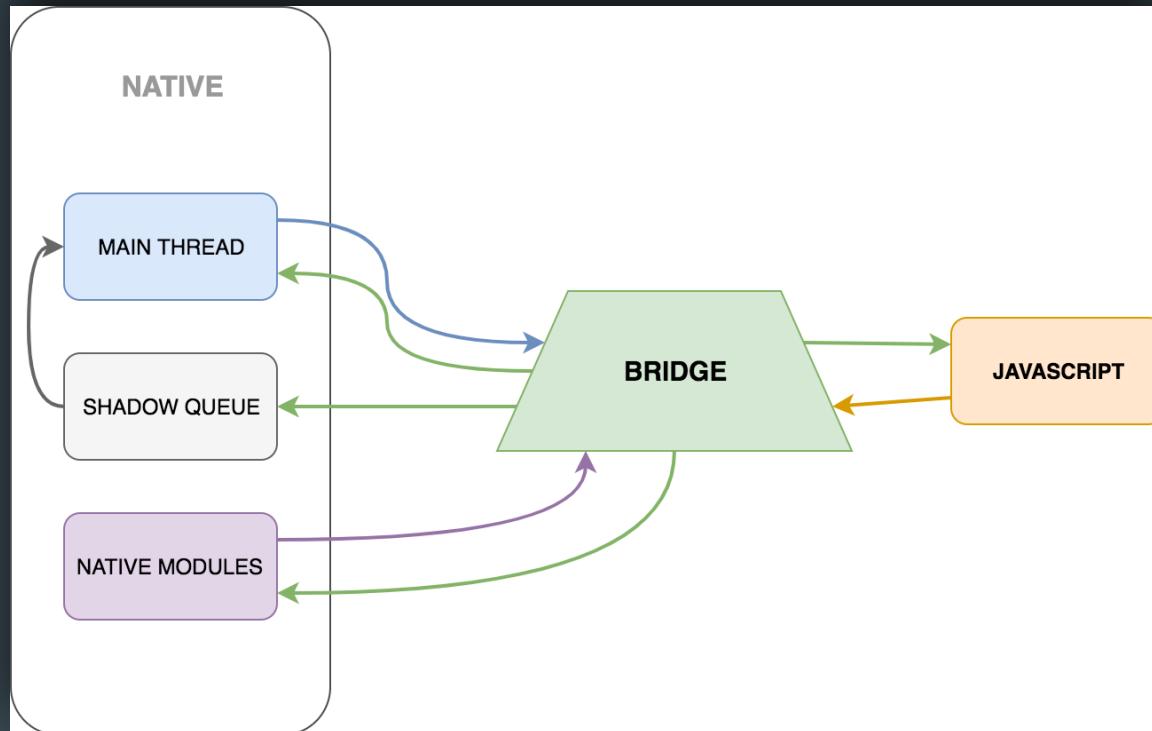
Podsumowanie Style

- Do nazw atrybutów używamy **camelCase**
- Zamiast pixeli używamy jednostek
- Nie ma kaskadowego dziedziczenia
- Style łączymy przy pomocy tablic
- Styl może być plain obiektem lub stworzonym przy pomocy **StyleSheet API**
- Jedyną opcją pozycjonowania elementów jest **flexbox**

Architektura aplikacji React Native



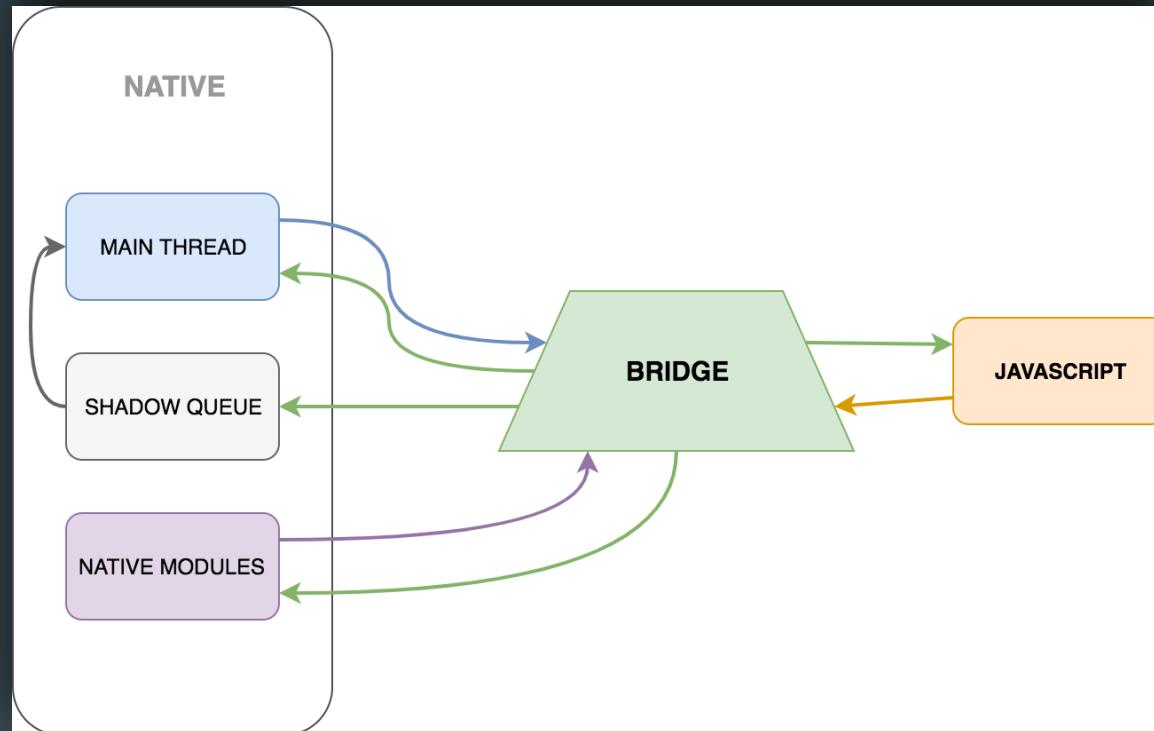
Architektura aplikacji React Native



Main Thread

- domyślny wątek aplikacji

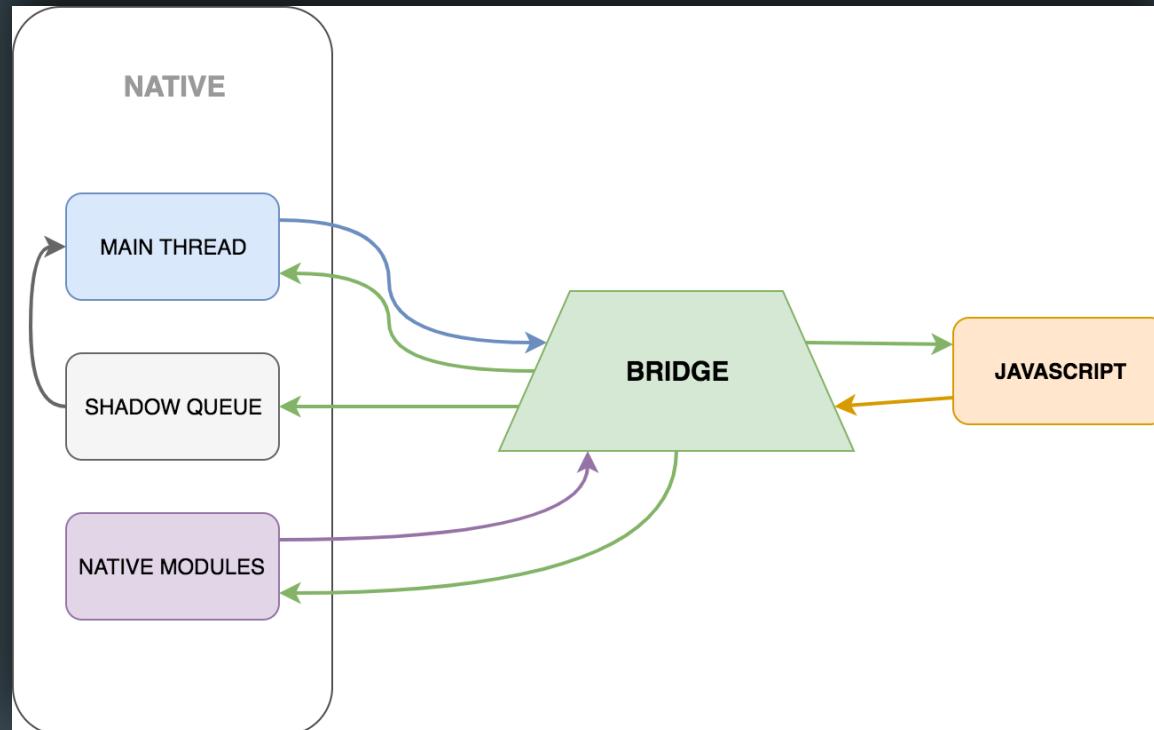
Architektura aplikacji React Native



Shadow Queue

- informacje o layout
- zna shadow nodes

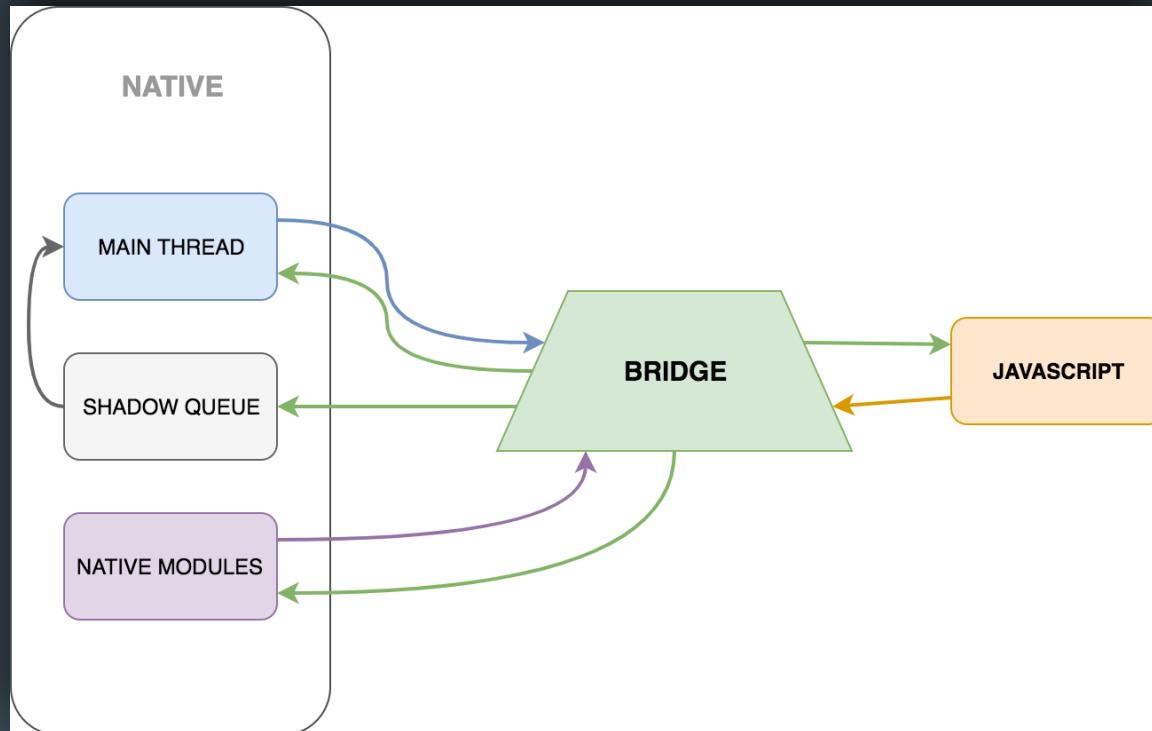
Architektura aplikacji React Native



Native Modules

- każdy moduł ma swój wątek
- network, images, localStorage etc

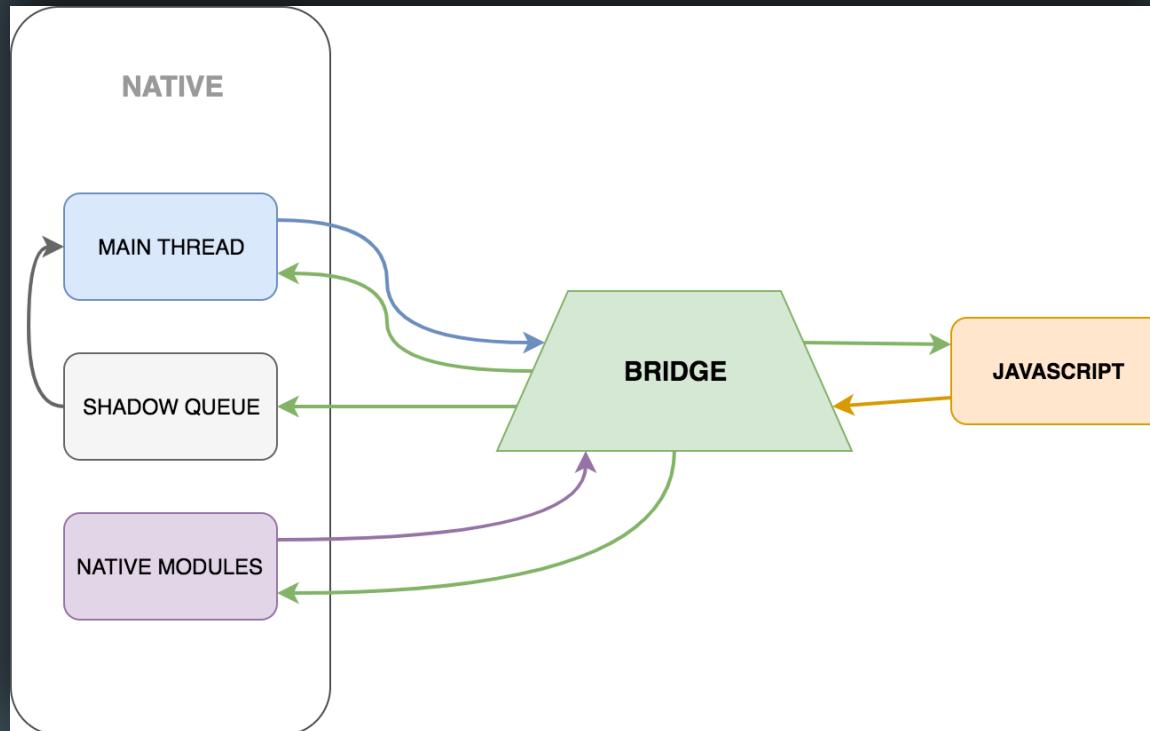
Architektura aplikacji React Native



JavaScript

- Instancja JS VM
- właściwy runtime aplikacji

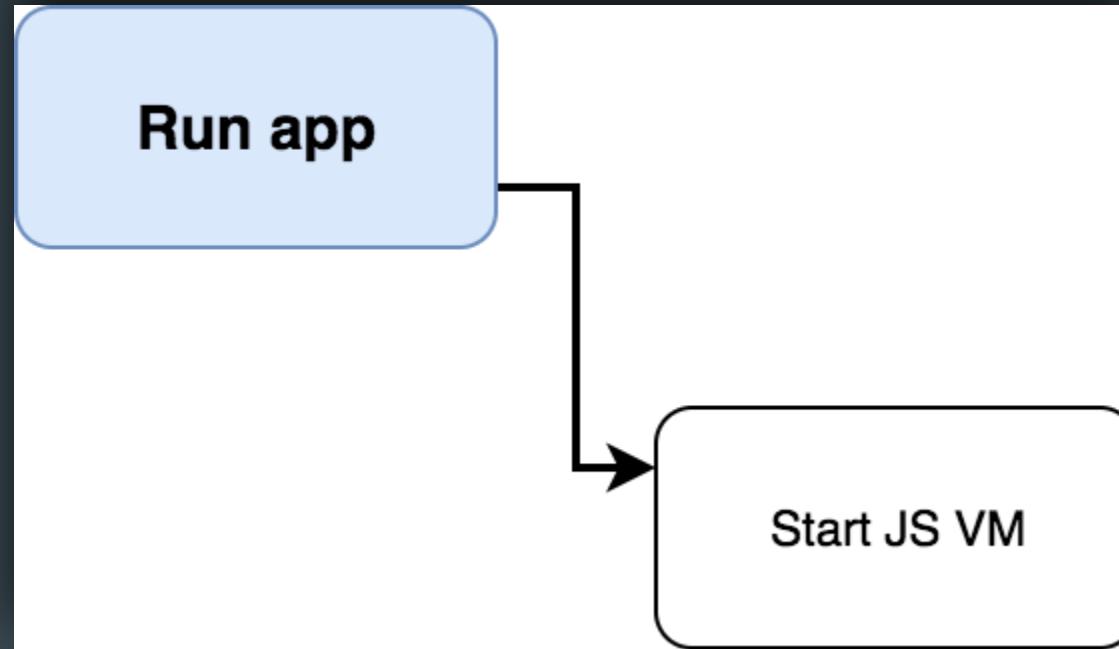
Architektura aplikacji React Native



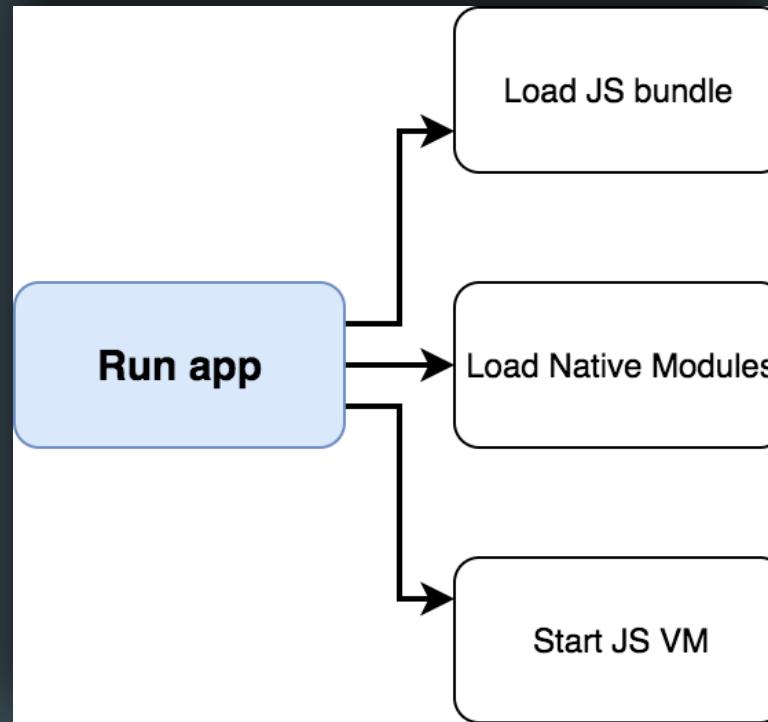
Bridge

- async
- batched
- serialized

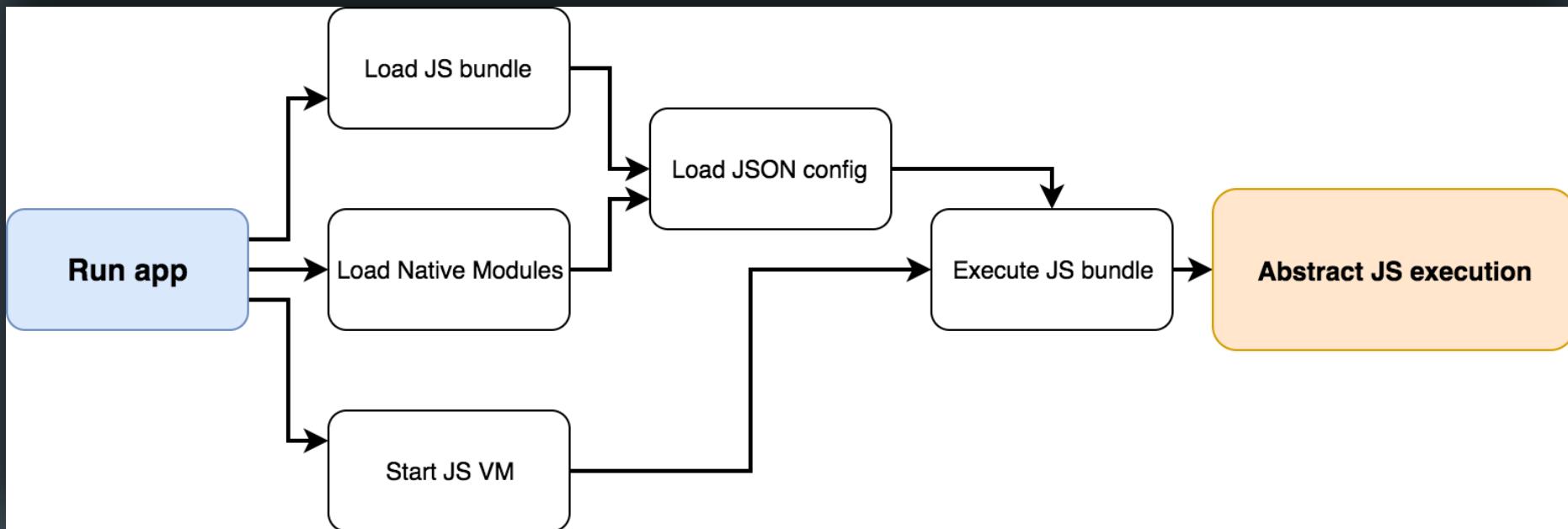
Run React Native App



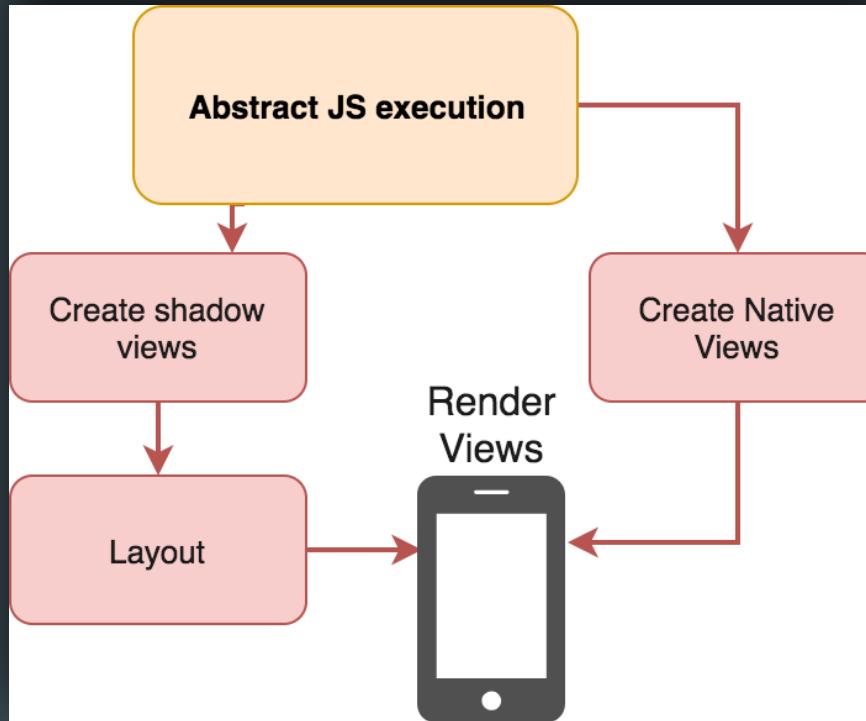
Run React Native App



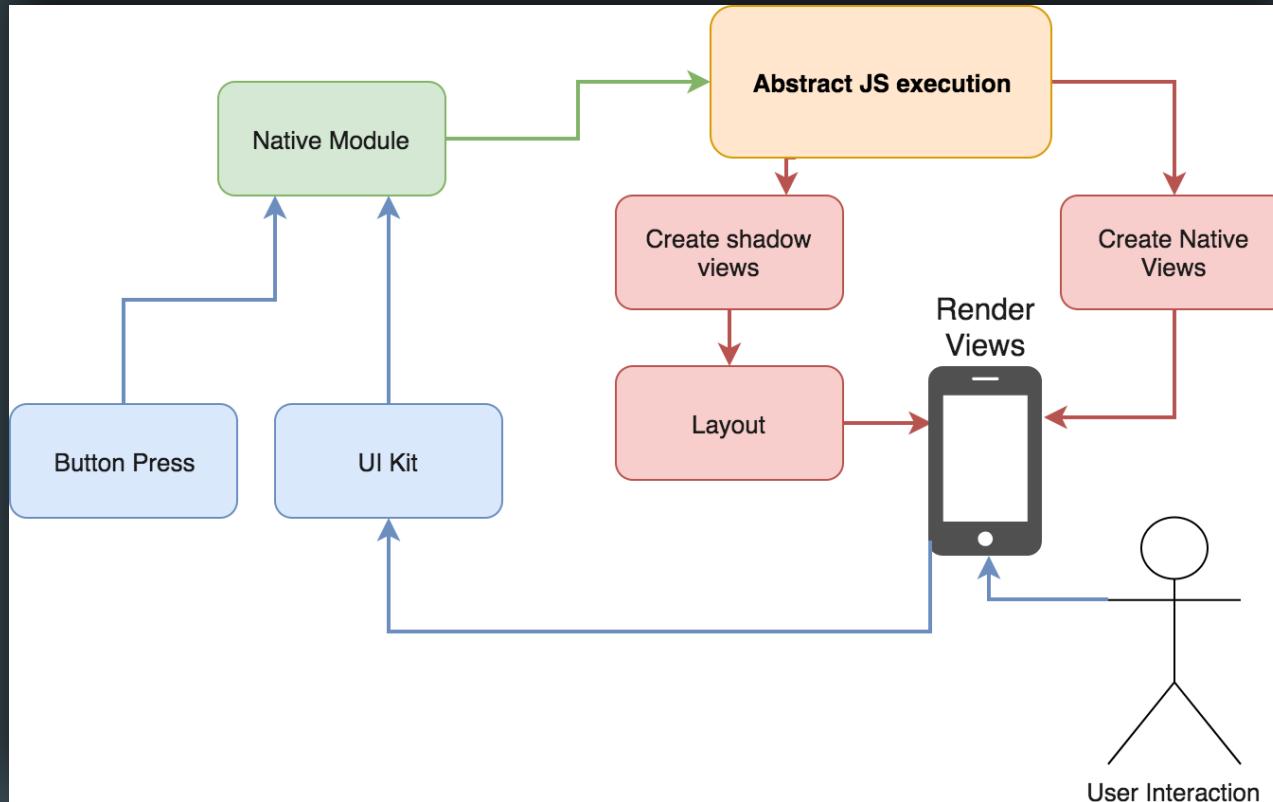
Run React Native App



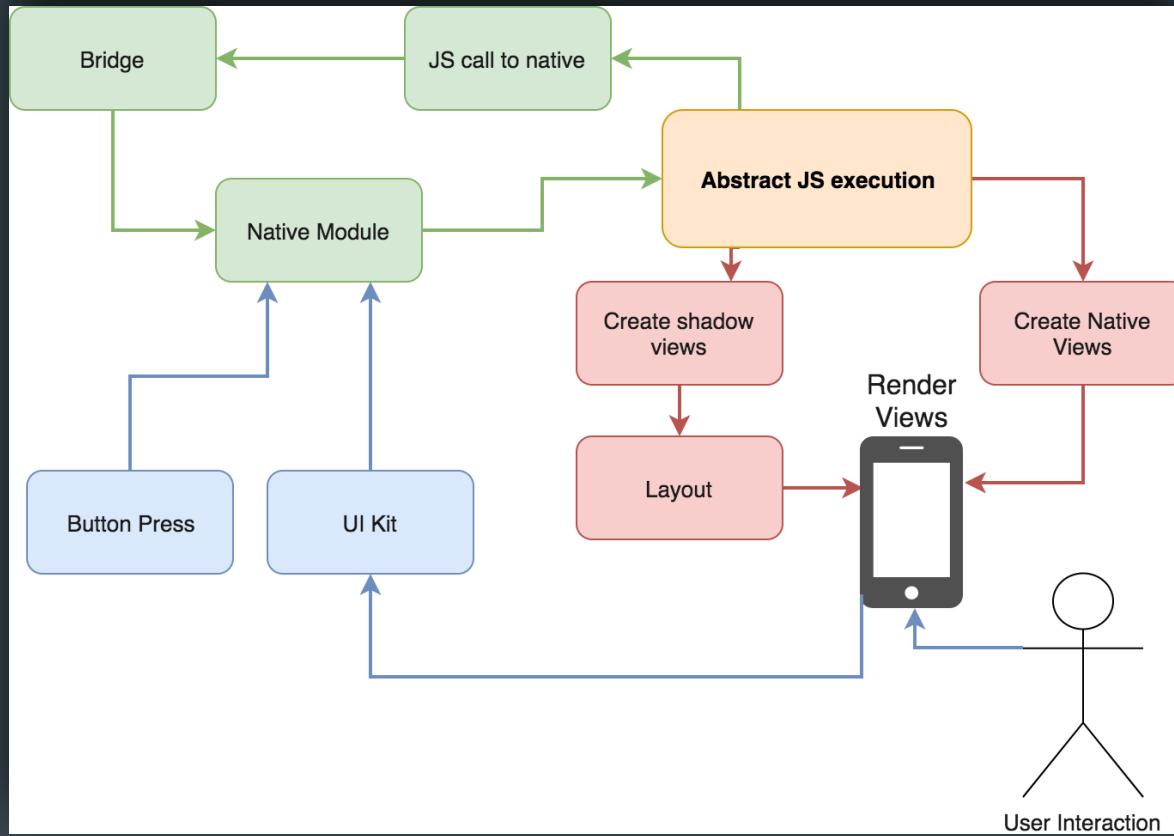
React Native app lifecycle



React Native app lifecycle



React Native app lifecycle



Podsumowanie architektury aplikacji

- Uruchomiony jest jeden watek JS
- Bridge serializuje dane
- Bridge komunikuje JS'a z natywnymi komponentami
- React Native uruchamia wirtualny silnik JS'a, który bundluje na bieżąco zmiany
- JS zarządza logiką aplikacji, natywne moduły odpowiadają za bezpośrednią komunikację z użytkownikiem