

# DESENVOLVIMENTO DE APPS MÓVEIS MULTIPLATAFORMA COM TECNOLOGIAS WEB



**“A mente que se abre a uma nova ideia  
jamais voltará ao seu tamanho original”  
(Albert Einstein)**

# CONTEÚDOS

- **Desenvolvimento Híbrido**
- **Introdução ao React e React Native**
- **Ferramentase e Configuração**
- **Aplicação prática**
- **Componentes Visuais**
- **Implantação e Testes**



# DESENVOLVIMENTO HÍBRIDO

**Tantas plataformas**

**Tantas linguagens**

**Tão pouco tempo**  
**Tão pouco dinheiro**

**Precisamos fazer escolhas...**



# DESAFIO?

**Criar aplicativos**  
**Multiplataforma!**



# TECNOLOGIAS WEB

- **Arquitetura Cliente-Servidor;**
- **Rodam em quase todas as plataformas existentes;**
- **Fáceis de aprender;**
- **Rápido desenvolvimento.**





**Será que estamos pensando o mesmo?**

**Que tal utilizar tecnologias web  
para criar aplicativos móveis  
multiplataforma?**



**Solução perfeita? Não.**

**Nem tudo são flores...**



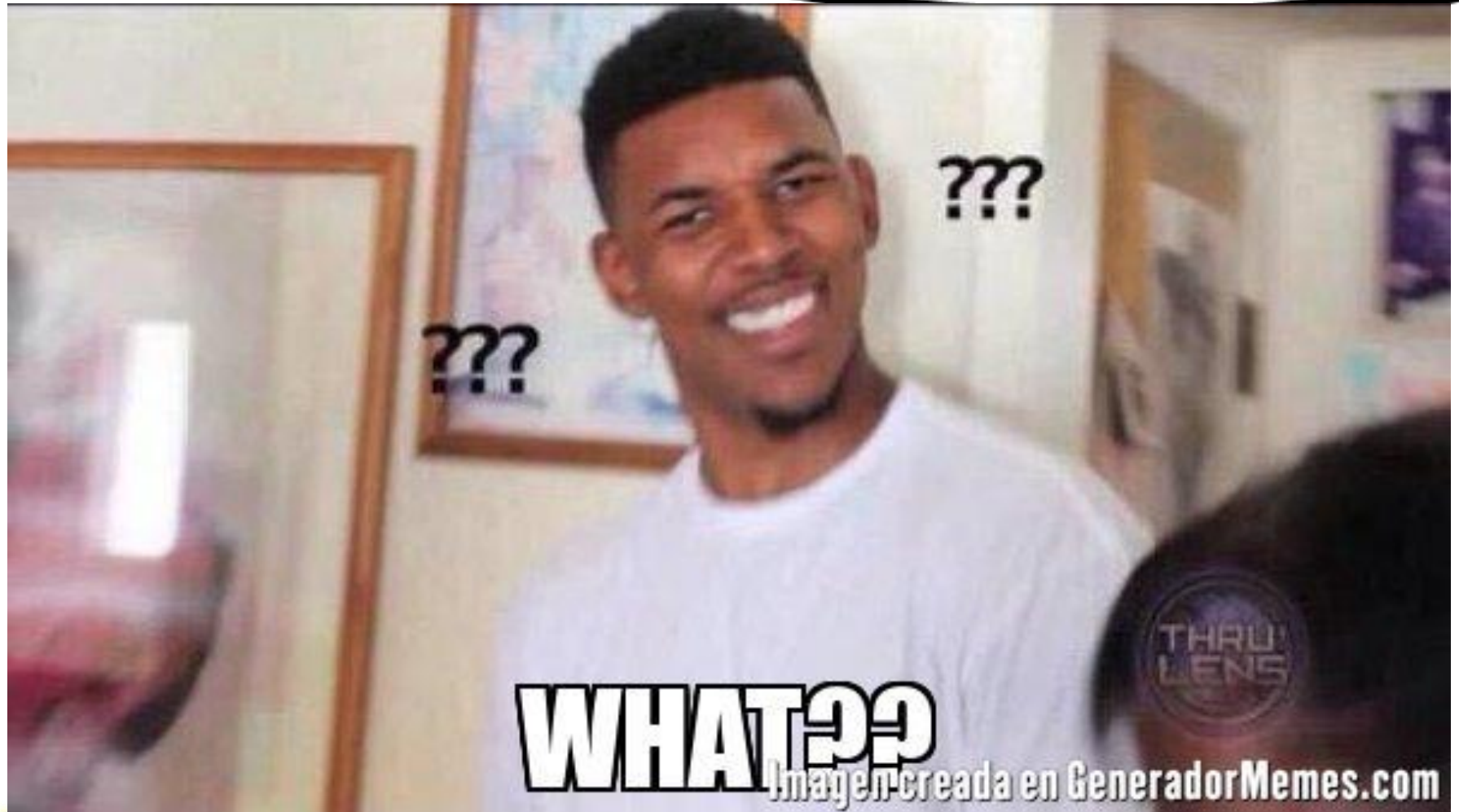
# QUAL A MELHOR OPÇÃO?

- O aplicativo requer o uso de alguma funcionalidade especial? (Câmera, acelerômetro, GPS, armazenamento...)
- Qual o seu orçamento?
- O aplicativo requer conexão com a internet?
- Quantas plataformas você pretende suportar?
- Quais linguagens de programação eu, ou minha equipe, domina?
- A performance é muito importante?
- O tempo é curto?



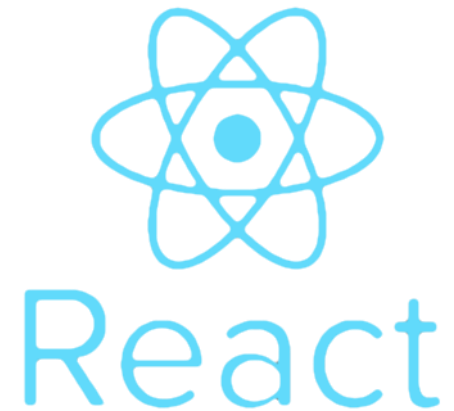


# REACT E REACT NATIVE???



# REACT: CARACTERÍSTICAS(2)

- Código aberto
- Framework de desenvolvimento Web baseado em Javascript
- Foco apenas na definição da UI
- Entrou em cena em maio 2013
- Mantido pelo **Facebook**, **Instagram** e uma comunidade de desenvolvedores e empresas



# REACT: INTRODUÇÃO(3)

- **Componentes**
  - JSX – linguagem de descrição Javascript/HTML
  - Composição de componentes
  - Propriedades de componentes
- **Manipulação de eventos**
- **Ciclo de vida dos componentes**



# REACT: INTRODUÇÃO(4)

- Composição de componentes e modularização

```
import React, { Component } from 'react';  
  
class Title extends Component {  
  render() {  
    return (  
      <h1> Hello World! </h1>  
    );  
  }  
}
```





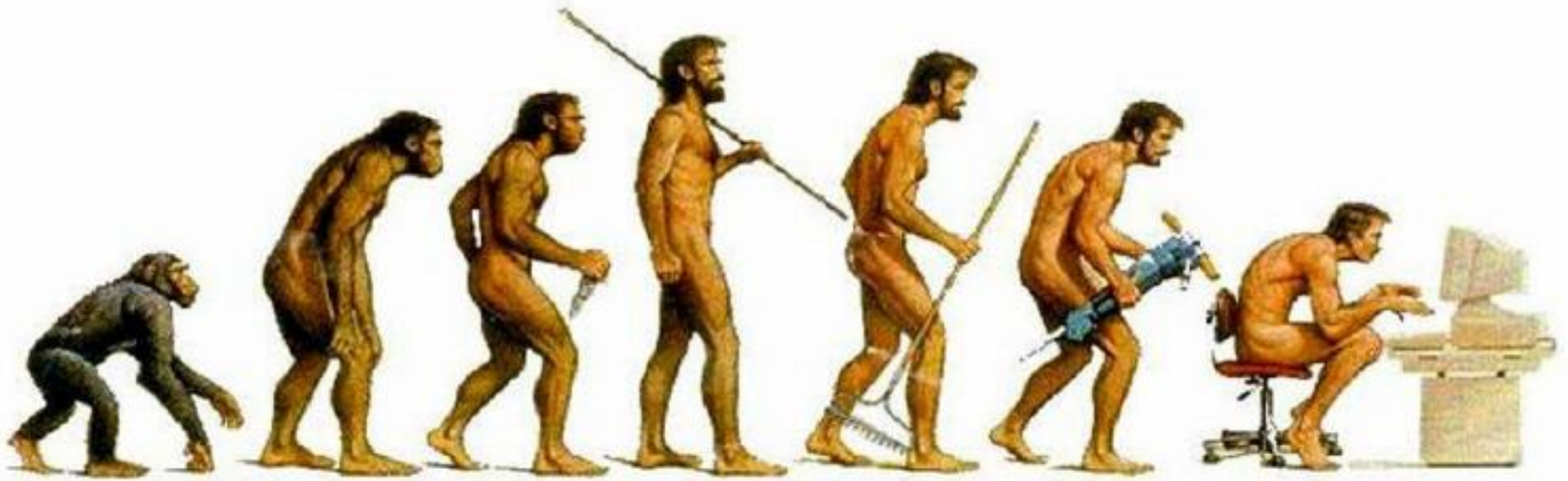
# REACT: INTRODUÇÃO(5)

- Tem suporte a ***props***(propriedades e sua customização) – valores estáticos;
- Para valores dinâmicos utiliza-se o ***state***;
- Permite programar respostas a eventos;
- Tem um ciclo de vida que gira em torno do método ***render()***.





# DESENVOLVIMENTO MOBILE



ANDROID  
2012



IOS  
2014



AVENUE  
2015



IONIC1  
2015



IONIC2  
2016



RN  
2016



# DESENVOLVIMENTO MOBILE

Developer  
Experience

Developer  
Experience

User  
Experience

Web



Native



RN



Web

React Native

Native Mobile

User  
Experience



<https://speakerdeck.com/felipecl/react-native-at-airbnb>

# REACT NATIVE(RN): O QUE É?

- Especialização do React para mobile;
- Uma biblioteca JavaScript para construção de interfaces de usuário;
- Aplicações podem ser renderizadas no iOS e Android;
- Implementa a parte V de uma arquitetura MVC;
- Programação Funcional;
- Quem usa: AirBNB, Facebook, Instagram, Vogue.



# RN: O QUE NÃO É?

- Não é um framework híbrido;
- Não é HTML/DOM, ou CSS;
- Não é um framework de aplicação;
- Não é uma ferramenta cross-platform;
- Não é possível integrar com JQuery.





# RN: CARACTERÍSTICAS

- **React Native** é um projeto desenvolvido pelos engenheiros do Facebook;
- Consiste em uma série de ferramentas que viabilizam a criação de aplicações **móveis nativas** para a plataforma **iOS** e **Android**, utilizando o que há de mais moderno no desenvolvimento.





# RN: CARACTERÍSTICAS

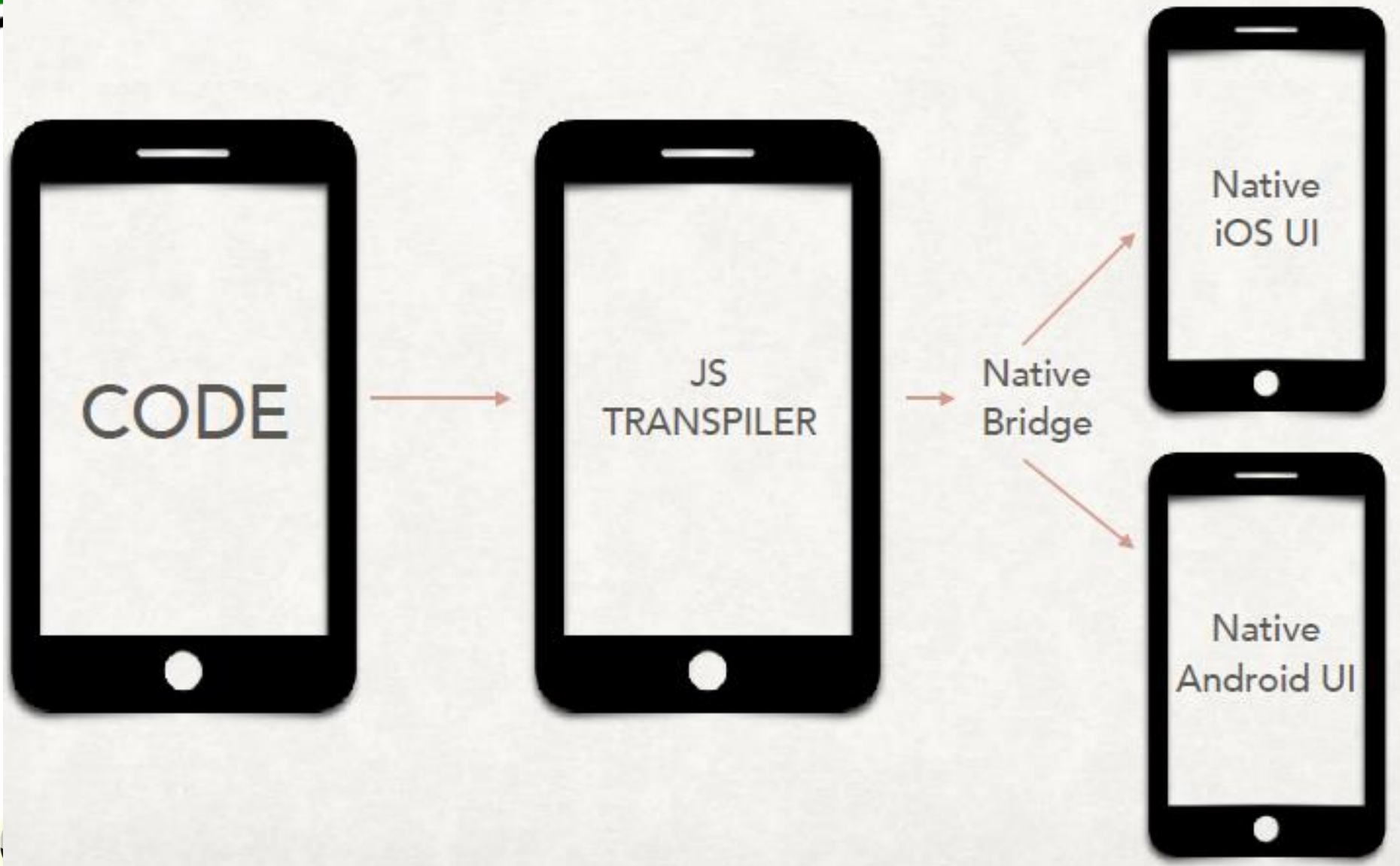
- Não utiliza HTML para processar o aplicativo (div, input, ...), mas possui componentes alternativos, que funcionam de forma parecida.

**View == div**

**Text == p**



# RN: COMO FUNCIONA?

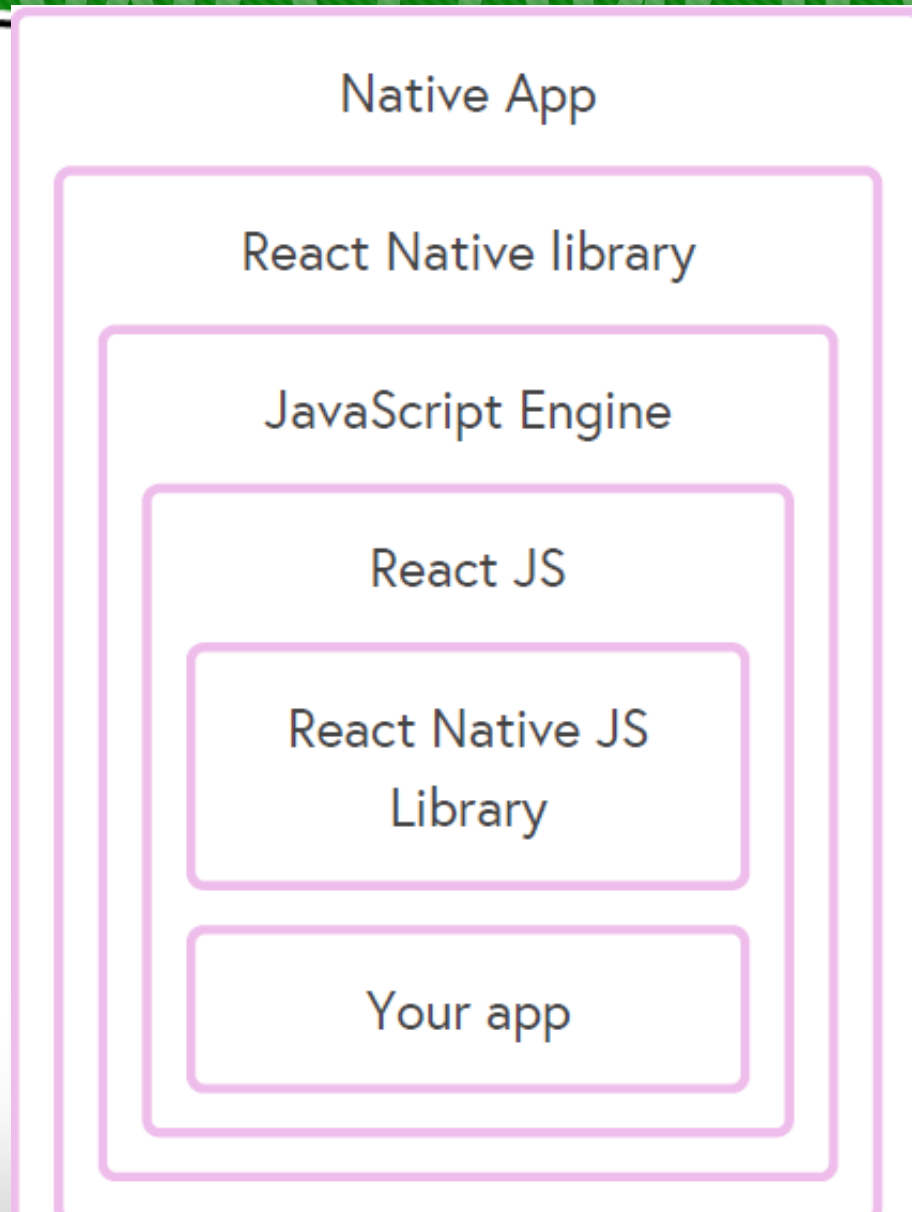


# RN: JS TRANSPILER

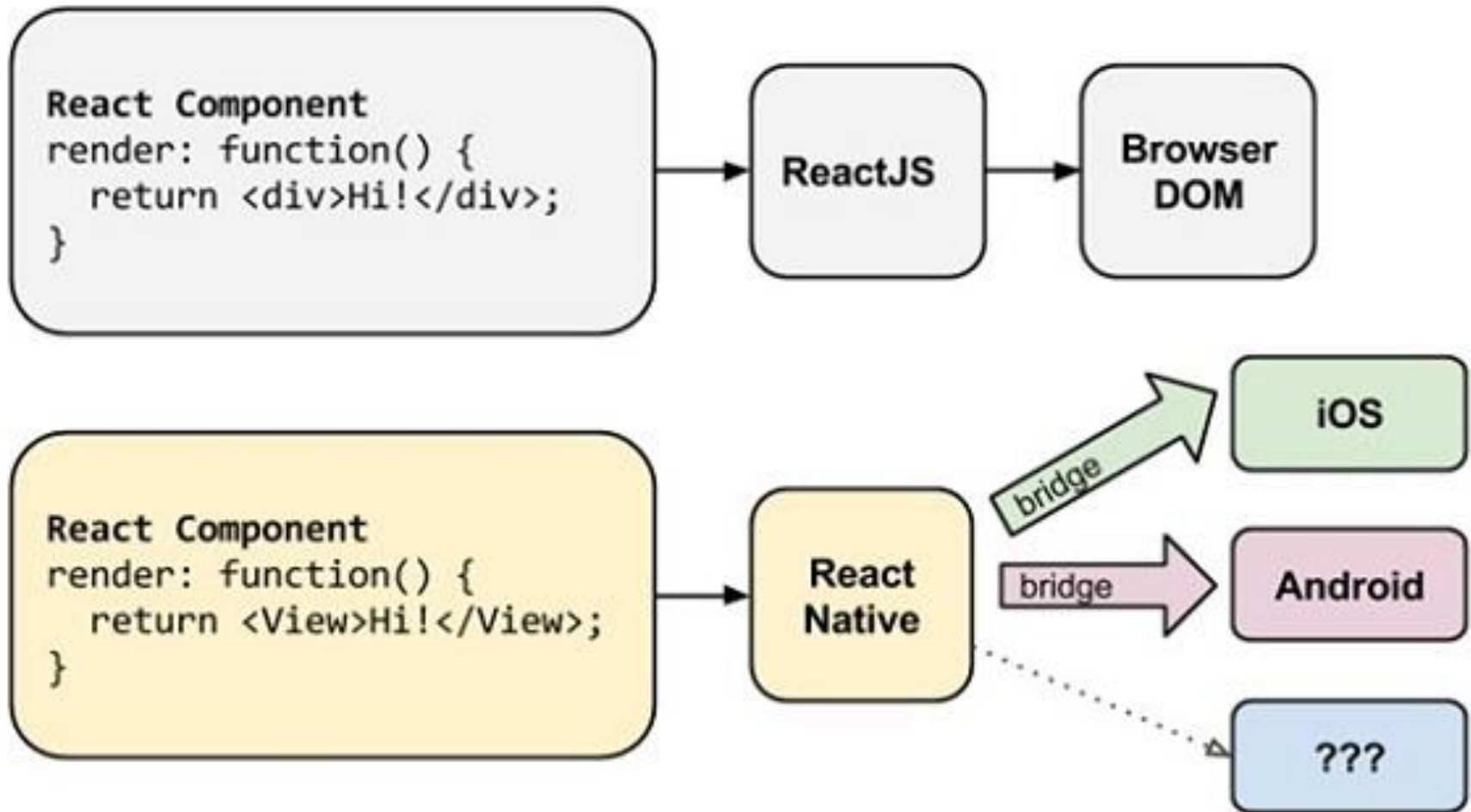
- ***Transpilers*** são uma realidade na programação e em especial no JavaScript;
- De maneira geral, a função de um *transpiler* é *traduzir* um código escrito em linguagem para um código de outra.



# RN: COMO FUNCIONA?



# RN: REACT VS RN





# RN: CARACTERÍSTICAS

- A stack do React Native é poderosa, pois nos permite utilizar **ECMAScript**, **CSS Flexbox**, **JSX**, diversos pacotes do **NPM** e mais ...;
- Permite fazer debug na mesma IDE utilizada para o desenvolvimento nativo com essas plataformas.



# RN: ECMAScript(ES)

- Primeiro se você veio do JavaScript ou já teve contato com essa linguagem não se assuste quando você vê o nome **ECMAScript**;
- O ECMAScript é a evolução do **JavaScript**.



# RN: ECMASCRIPT(ES)

- **O ECMAScript tem como foco:**
  - Ser uma linguagem melhor para construir aplicações complexas;
  - Resolver problemas antigos do JavaScript;
  - Facilitar o desenvolvimento de libraries.



# RN: CSS FLEXBOX

- ***CSS Flexbox Layout Module***
- Mais flexibilidade para desenhar layouts responsivos, sem usar *float*;
- Tem opções para direcionamento e posicionamento dos componentes visuais;
- Composição visual.



# RN: CSS FLEXBOX

flex-start



flex-end



center



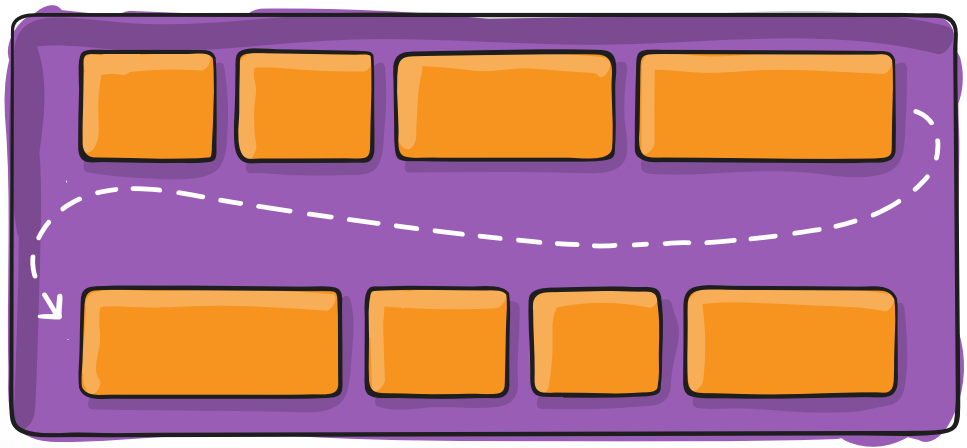
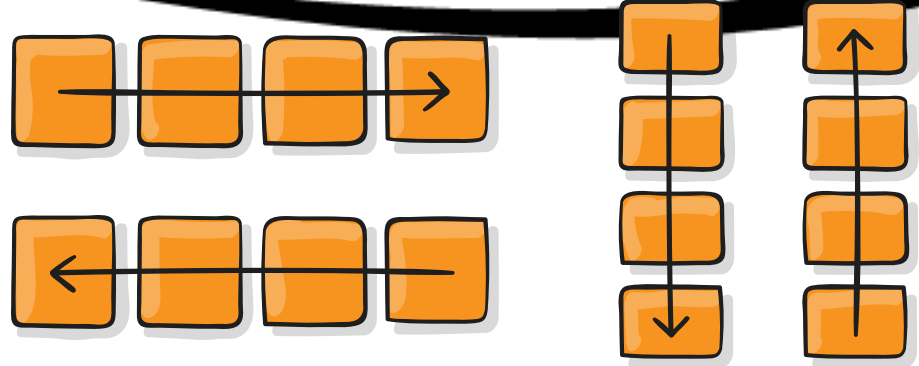
space-between



space-around



space-evenly





# RN: CONFIGURANDO O AMBIENTE

- **NodeJS e NPM**
- **JDK 8**
- **Python 2**
- **React Native CLI(Client)**
- **Yarn**
- **Android Studio**
- **Visual Studio Code**



# RN: CHOCOLATEY NO WINDOWS

- Gerenciador de pacotes para Windows;
  - Para instalar acesse:  
<https://chocolatey.org/install>
- Instalando via chocolatey(como administrador):
  - `choco install -y nodejs.install python2 jdk8`



# RN: INTALANDO O CLIENTE

- Cliente do react-native instalado de forma global;
- Gerencia a instalação do framework react-native;
- Utilizar o npm:

***—npm install -g react-native-cli***



# RN: INSTALANDO O YARN

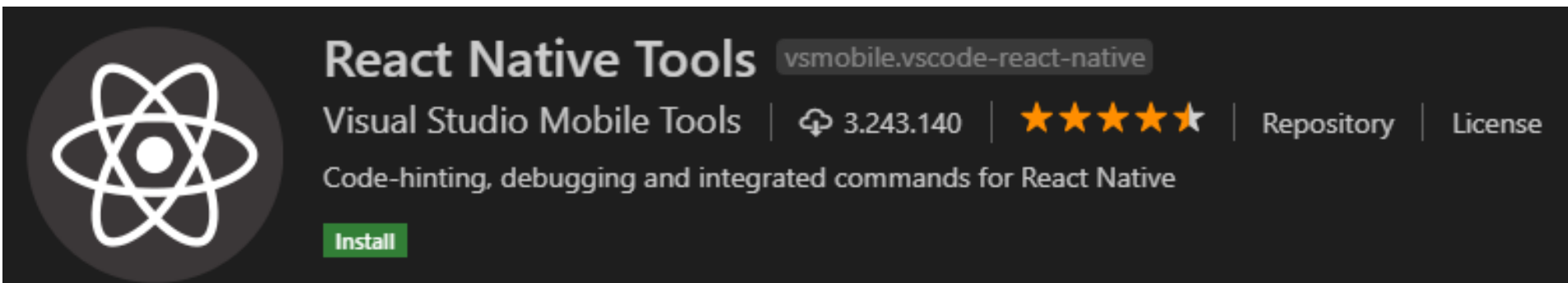
- Gerenciador de dependências e pacotes
- Integra-se ao NodeJS
- Utilizando o chocolatey:  
*—choco install yarn*





# RN: CONFIGURANDO O VSC

- Plugin: *React Native Tools*



The image shows the Visual Studio Marketplace entry for the 'React Native Tools' extension. On the left is the React logo. To its right, the extension name 'React Native Tools' is displayed in large white text, with the identifier 'vsmobile.vscode-react-native' in a smaller font below it. Further right, the text 'Visual Studio Mobile Tools' is shown, followed by a download count of '3.243.140', a five-star rating, and links for 'Repository' and 'License'. Below this, a description reads 'Code-hinting, debugging and integrated commands for React Native'. At the bottom left of the card is a green 'Install' button.

**React Native Tools** vsmobile.vscode-react-native

Visual Studio Mobile Tools | 3.243.140 | ★★★★★ | Repository | License

Code-hinting, debugging and integrated commands for React Native

[Install](#)



# RN: CONFIGURANDO O VSC

- Plugin: *React-Native/React/Redux*



React-Native/React/Redux snippets for es6/es7

[equimper.react-native-react-redux](https://github.com/equimper/react-native-react-redux)

EQuimper | 698.522 | ★★★★★ | Repository

Code snippets for React-Native/React/Redux es6/es7 and flowtype/typescript

Install



# HANDS ON REACT-NATIVE



# RN: CRIANDO A PRIMEIRA APP

- Abrir o *Node.js command prompt*
- Para criarmos um novo projeto:
  - *react-native init NomeProjeto*





# RN: PRIMEIRA APP

```
D:\cloud\gdrive\dsv\wksrn>react-native init OlaMundo
This will walk you through creating a new React Native project in D:\cloud\gdrive\dsv\wksrn\OlaMundo
Using yarn v1.13.0
Installing react-native...
yarn add v1.13.0
info No lockfile found.
[1/4] Resolving packages...
@babel/helper-plugin-utils@^7.0.0
```

14:46

Welcome to React Native!

To get started, edit App.js

Double tap R on your keyboard to reload,  
Shake or press menu button for dev menu



# RN: ESTRUTURA DO PROJETO

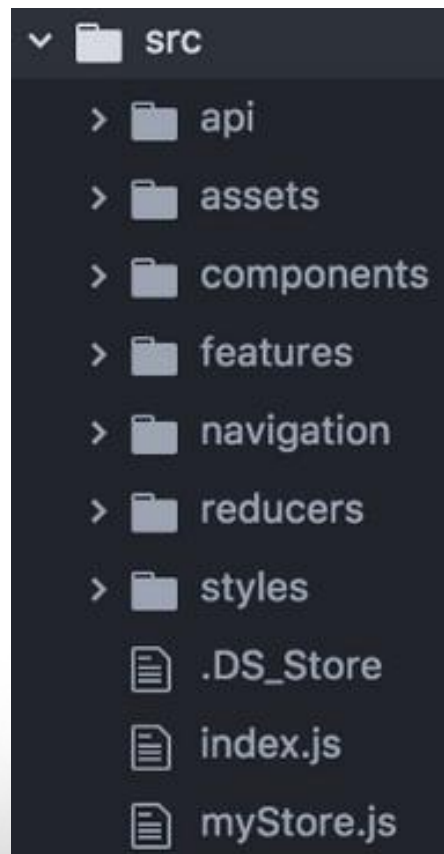
## OLAMUNDO

- \_\_tests\_\_
- android
- ios
- node\_modules
- ≡ .buckconfig
- ≡ .flowconfig
- ◆ .gitattributes
- ◆ .gitignore
- { } .watchmanconfig
- JS** App.js
- { } app.json
- JS** babel.config.js
- JS** index.js
- { } package.json
- 👤 yarn.lock



# RN: CRIANDO A PRIMEIRA APP

- Possibilidade de introduzir arquivos **JSX** e **JS**;
- Arquivos extras são colocados em um diretório **src**.



# RN: CRIANDO DO ZERO

- Importar react e react native;

```
import React, { Component } from 'react';  
import {  
  View,  
  Text,  
  Image  
} from 'react-native';
```





# RN: CRIANDO DO ZERO

- Criar componente para a aplicação;

```
export default class App extends Component {  
  render() {  
    return (  
      <View>  
        { /* o render é usado para retornar as views */}  
        <Text>Olá Mundo!</Text>  
      </View>  
    );  
  }  
}
```



# RN: EXECUTANDO ....

- Navegar até a pasta do projeto:
- Android
  - *react-native run-android*
- IOs
  - *react-native run-ios*
- Antes de executar o comando deixe o dispositivo conectado ou o emulador aberto.



# RN: VIEWS

- Componente container para os demais;
- Equivale a uma div do HTML.

```
<View style={styles.container}>  
  <Text style={styles.welcome}>Welcome to React Native!</Text>  
  <Text style={styles.instructions}>To get started, edit App.js</Text>  
  <Text style={styles.instructions}>{instructions}</Text>  
</View>
```



# RN: PROPRIEDADES(PROPS)

- Propriedades permitem configurar o comportamento dos componentes visuais;
- Basicamente são propriedades CSS(styles);
- A configuração tem um padrão diferente;
- Valores são estáticos.





# RN: CRIANDO COMPONENTES

- Basicamente estender *Component*

```
class MinhaImagem extends Component {  
  render() {  
    |  
    let img = {  
      uri: "https://sujeitoprogramador.com/steve.png"  
    };  
  
    return (  
      <Image source={img} style={{ width: parseInt(this.props.largura), height: parseInt(this.props.altura)}}  
    );  
  }  
}
```

```
<MinhaImagem/>
```



# RN: COMPONENTE + PROPS

```
class MinhaImagem extends Component {  
  render() {  
    |  
    let img = {  
      uri: "https://sujeitoprogramador.com/steve.png"  
    };  
  
    return (  
      <Image source={img} style={{ width: parseInt(this.props.largura), height: parseInt(this.props.altura)}}  
    );  
  }  
}
```

```
<MinhaImagem largura='100' altura='10' />
```



# RN: COMPONENTE + PROPS

```
class MinhaImagem extends Component {  
  render() {  
    |  
    let img = {  
      uri: "https://sujeitoprogramador.com/steve.png"  
    };  
  
    return (  
      <Image source={img} style={{ width: parseInt(this.props.largura), height: parseInt(this.props.altura)}}  
    );  
  }  
}
```

```
<MinhaImagem largura='100' altura='10' />
```



# RN: TEXT

- Representa um texto na tela;

```
<Text>Olá Mundo!</Text>
```



# RN: SINTAXE

- `{}` – **Código JS**
- `{{}}` – **Objetos JS**





# RN: STATES

- States = Estados;
- Permitem definir valores mutáveis;
- Utiliza-se o *this.setState()* para atualizar states.

```
constructor(props) {  
  super(props);  
  this.state = {  
    pizza: 'strogonoff'  
  }  
}
```



# RN: STATES - EXEMPLO

```
export default class App extends Component {  
  
  constructor(props) {  
    super(props);  
    this.state = {  
      pizza: 'stroganoff'  
    }  
  
    var pizzas = ['calabresa', 'gaúcha', 'picanha', 'califórnia'];  
  
    setInterval(() => {  
      this.setState(previousState => {  
        var n = Math.floor(Math.random() * pizzas.length);  
        return {pizza : pizzas[n]}  
      });  
    }, 1000);  
  }  
}
```



# RN: STATES - EXEMPLO

```
render() {  
  return (  
    <View style={{paddingTop: 20}}>  
      {/* o render é usado para retornar as views */}  
      <Text style={{textAlign: 'center', fontSize: 28, fontWeight: 'bold', color: 'red'}}>Menu de Pizza</Text>  
      <Text style={{textAlign: 'center', fontSize: 20, color: 'green'}}>{this.state.pizza}</Text>  
    </View>  
  );  
}
```



# RN: GRUPOS DE ESTILO

- Podem ser aplicados em mais views/componentes;
- Deve ser declarado fora do componente;

```
const styles = StyleSheet.create({
  textoPrincipal: {
    fontSize: 27,
    textAlign: 'center'
  },
  textoColor: {
    color: 'red'
  },
  subTexto: {
    textAlign: 'left',
    fontSize: 15
  }
});
```

```
<Text style={styles.textoPrincipal}>Texto 1</Text>
<Text style={styles.textoColor}>Texto 2</Text>
<Text style={styles.textoPrincipal}>Texto 3</Text>
<Text style={[styles.textoColor,styles.textoPrincipal]}>Texto 4</Text>
<Text style={styles.subTexto}>Texto 4</Text>
```

# RN: TAMANHOS DINÂMICOS E FIXOS

- Utilizando nas views;
- Propriedade *flex*;
- Permite definir tamanho flexível;

```
<View style={{flex: 1}}>  
  <View style={{height: 60, backgroundColor: 'red'}}></View>  
  <View style={{flex: 1, backgroundColor: 'white'}}></View>  
  <View style={{height: 60, backgroundColor: 'green'}}></View>  
</View>
```





# RN: SEPARANDO OS ESTILOS

```
const styles = StyleSheet.create({
  textoPrincipal: {
    fontSize: 27,
    textAlign: 'center'
  },
  textoColor: {
    color: 'red'
  },
  subTexto: {
    textAlign: 'right',
    fontSize: 15
  }
})
```

```
<Text style={styles.textoPrincipal}>Texto 1</Text>
```



# RN: FLEXBOX E ALINHAMENTOS

- Flexbox, ou *Flexible Box Layout*, é um novo modo de layout em CSS3, projetado para criar layouts mais complexos e responsivos;
- Propriedade *flexDirection* combinada com *flex*;
- Direções principais: *column* e *row*;
- Para o alinhamento pode ser utilizado o *justifyContent* e *alignItems*;



# RN: FLEXBOX E ALINHAMENTOS

- Exemplo:

```
flex: 1, flexDirection: 'column', alignItems: 'center'
```



# RN: RECEBENDO DADOS

- Utiliza-se o componente *TextInput*;
- Lembrar de importar TextInput;

```
<TextInput style={styles.input} placeholder="Digite seu nome ..."  
underlineColorAndroid='transparent'  
onChangeText={(textoInput) => this.setState({textoInput})}/>
```



# RN: EVENTOS DE BOTÃO

- Utiliza-se o componente ***Button***;
- Possui a propriedade ***title***;
- Possui um evento ***onPress***;

```
<Button title="Clique aqui!" onPress={this.enviar}/>
```

```
    this.enviar = this.enviar.bind(this);  
  }
```

```
  enviar () {  
    alert('teste de botão');  
  }
```





# RN: CUSTOMIZANDO UM BOTÃO

```
class Botao extends Component {  
  
  constructor(props) {  
    super(props);  
    this.state = {};  
  
    this.styles = StyleSheet.create({  
      botao: {  
        width: 230,  
        height: 50,  
        borderWidth: 2,  
        borderColor: props.cor,  
        borderRadius: 25  
      },  
      botaoArea: {  
        flex: 1,  
        flexDirection: 'row',  
        justifyContent: 'center',  
        alignItems: 'center'  
      },  
      botaoTexto: {  
        fontSize: 20,  
        fontWeight: 'bold',  
        color: props.cor  
      }  
    });  
  }  
}
```



# RN: CUSTOMIZANDO UM BOTÃO

```
render() {  
  return(  
    <TouchableOpacity style={this.styles.botao} onPress={this.props.onPress}>  
      <View style={this.styles.botaoArea}>  
        <Text style={this.styles.botaoTexto}>{this.props.nome}</Text>  
      </View>  
    </TouchableOpacity>  
  );  
}
```



# RN: APP BISCOITO DA SORTE

- Criando nosso primeiro App mais complete.



# RN: APP CRONÔMETRO

- Criando um app de cronômetro.



# RN: SCROLLVIEW

- Permite a rolagem em objetos da view;
- Carrega todos os dados de uma vez.

```
<View style={styles.container}>
  <View style={styles.header}>
    <Text style={styles.texto}>Últimas Notícias</Text>
  </View>
  <ScrollView>
    <View style={styles.box1}></View>
    <View style={styles.box2}></View>
    <View style={styles.box3}></View>
    <View style={styles.box4}></View>
    <View style={styles.box1}></View>
    <View style={styles.box2}></View>
    <View style={styles.box3}></View>
    <View style={styles.box4}></View>
  </ScrollView>
  <View style={styles.header}>
    <Text style={styles.texto}>Sair do App</Text>
  </View>
</View>
```





# RN: FLATLIST

- Similar ao ScrollView, mas não carrega tudo na memória(Lazy).
- Propriedades importantes: *data* e *renderItem*;
- *data* = recebe os dados(array de dado);
- *renderItems* = forma de renderizar os itens. Renderiza conforme navegamos na lista.



# RN: FLATLIST

```
<FlatList data={this.state.listaData} renderItem={({item}) => this.renderItemData(item)}/>
```

```
renderItemData(item) {  
  return (  
    <View style={styles.areaView}>  
      <Text style={styles.texto}>{item.nome}</Text>  
      <Text style={styles.texto}>{item.idade}</Text>  
      <Text style={styles.texto}>{item.email}</Text>  
    </View>  
  );  
}
```

```
listaData: [  
  { key: '1', nome: 'Roberson1', idade: 39, email: 'robersonjfa@gmail.com' },  
  { key: '2', nome: 'Roberson2', idade: 39, email: 'robersonjfa@gmail.com' },  
  { key: '3', nome: 'Roberson3', idade: 39, email: 'robersonjfa@gmail.com' },  
  { key: '4', nome: 'Roberson4', idade: 39, email: 'robersonjfa@gmail.com' },  
  { key: '5', nome: 'Roberson5', idade: 39, email: 'robersonjfa@gmail.com' },  
  { key: '6', nome: 'Roberson6', idade: 39, email: 'robersonjfa@gmail.com' },  
  { key: '7', nome: 'Roberson7', idade: 39, email: 'robersonjfa@gmail.com' },  
  { key: '8', nome: 'Roberson8', idade: 39, email: 'robersonjfa@gmail.com' }  
]
```



# RN: PICKER

- Permite a escolha de uma lista de opções;
- Duas propriedades: *selectedValue* e *onValueChange*;
- Precisamos criar um Picker.Item com as propriedades *key*, *value* e *label*.



# RN: PICKER

```
let pizzasItem = this.state.pizzas.map((v, k) => {
  return <Picker.Item key={k} value={k} label={v.nome} />
});

return (
  <View style={styles.container}>
    <Text style={styles.logo}>Pizzas React</Text>
    <Picker selectedValue={this.state.pizza} onValueChange={(itemValue, itemIndex) =>
      this.setState({ pizza: itemValue })>
      {pizzasItem}
    </Picker>
    <Text style={styles.valorPizza}>R$ {this.state.pizzas[this.state.pizza].valor.toFixed(2)}</Text>
  </View>
);
```



# RN: ATIVIDADE AVALIATIVA 01

- **Crie um novo projeto com as seguintes características:**
  - (2,00) Apresente um cabeçalho e um rodapé;
  - (2,00) Estilos estejam separados;
  - (2,00) Que permita entradas pelo usuário e interação via botões;
  - (2,00) Validar as entradas.
  - (2,00) Exiba uma lista.

