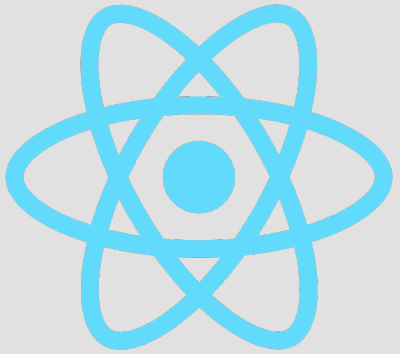
Components e Estado (*state*)

Curso de React Native



Component

- Encapsulamento de código
  - Evita repetição
- Composição de lógica, visualização e estilização
- Possui estado próprio
- É escrito de maneira **declarativa**

Component

```
...
<View>
    <Text>Olá mundo!</Text>
    <Button title="Enviar mensagem"
            onPress={() => null}/>
</View>
...
```

Importando componentes

- Você deve importar um componente no arquivo para que seja possível utilizá-lo
  - Similar ao *#include* em .C ou *import* em Java

Importando componentes

```
import {View,
        Text,
        Button} from 'react-native'

import axios from 'axios'

import MeuBotao from './MeuBotao.js'
```

Components nativos do React Native

- <View> - criação de uma área de visualização
- <Text> - componente para exibição de texto
- <Image> - componente para exibição de imagem
- <Button> - componente para criação de botão
- <TouchableOpacity> - componente para criação de botão com retorno visual

Entre outros!

Components nativos do React Native

- Ler a documentação é **essencial** para entender como cada componente funciona

Documentação dos componentes nativos do react-native

https://facebook.github.io/react-native/

API -> Components

Components nativos do React Native

```
...
<View>
    <Text>Olá mundo!</Text>
    <Button title="Enviar mensagem"
            onPress={() => null}/>
</View>
...
```

- PRÁTICA:

Ver como o código acima fica no React Native

Criando seu próprio componente

- Ao longo do desenvolvimento de um app, você pode precisar um componente que ainda não existe, ou então encapsular um código que está sendo utilizado várias vezes
- Para isso você pode criar seu próprio componente

Criando seu próprio componente

Class Component

```
import React from 'react'

class MeuComponente extends React.Component {

    render(){
        return(
            //Seu componente
        )
    }

}

export default MeuComponente
```

- OBS: o *return()* do componente **precisa** retornar apenas **uma tag <> ... </>**

Criando seu próprio componente

- PRÁTICA:

Criar um componente MeuBotao utilizando TouchableOpacity com um texto dentro, e replicar ele 5 vezes

Estado (*state*)

- É um objeto que representa a situação atual do componente dentro do app
  - Cada componente tem um estado único!
- Contém as *n* variáveis que vão ser alteradas durante a execução do aplicativo para aquele componente
- O React Native controla o estado internamente, mas precisamos avisar para ele quando ele muda!

Inicializando um *state*

```
class MeuComponente extends React.Component {

    state = {
        variavel1: '',
        variavel2: 0,
        variavel3: true,
        ...
        variavel n: undefined
    }

    ...
```

Inicializando um *state*

```
class MeuComponente extends React.Component {

    constructor(){

        state = {
            variavel1: '',
            variavel2: 0,
            variavel3: true,
            ...
            variaveln: undefined
        }

    }

    ...
```

Mudando o estado de uma variável

```
this.setState({

    variavel: //novo valor

})
```

Mudando o estado de uma variável

- PRÁTICA:

Alterar o componente MeuBotao para mudar seu texto quando for pressionado

Mudando o estado de uma variável

- O React espera todos os componentes terem chamado setState() antes de renderizar novamente.
- Isso garante mais performance do aplicativo, porém pode gerar inconsistências.

Mudando o estado de uma variável

```
mudarMensagem(){
  if(this.state.resultado < 18.5){
      this.setState({res:'É menor que 15!'})
  }else if(this.state.resultado>=15 && resultado < 20){
      this.setState({res:'Está entre 15 e 20'})
  }else {
      this.setState({res:'É maior que 20!'})
  }
}

verificarNumero(){
  const resultado= this.state.numero/2
  this.setState({ total:resultado })
  this.mudarMensagem()
}
```

Não vai funcionar!

Mudando o estado de uma variável

```
mudarMensagem(resultado){
  if(resultado<15){
    this.setState({
        res:'É menor que 15!',
        total: resultado
      })
  }else if(resultado>=15 && resultado<20){
    this.setState({
        res:'Está entre 15 e 20',
        total: resultado
      })
  }else {
    this.setState({
        res:'É maior que 20!',
        total: resultado
      })
  }
}

verificarNumero(){
  const resultado= this.state.numero/2
  this.imcMensagem(resultado)
}
```

Function component

- Você também pode escrever componentes como funções (RN 0.59>)
- Dispensa o uso de **classes e hierarquia** para sua criação
  - Utiliza **Hooks** para compensar

```
import React from 'react'
import {TouchableOpacity, Text} from 'react-native'

function FunctionButton() {

    return(
        <TouchableOpacity
            style={{ width: 200, height: 50, backgroundColor: '#65ab65',
                    justifyContent: 'center'}}>
                <Text style={{textAlign: 'center'}}>Me pressione!</Text>
        </TouchableOpacity>
    )

}

export default FunctionButton
```

Function stateless component

```
import React, { useState } from 'react'
import {TouchableOpacity, Text} from 'react-native'

function FunctionButton() {

    const [texto, setText] = useState('Me pressione!')

    return(
        <TouchableOpacity
            style={{ width: 200, height: 50, backgroundColor: '#65ab65',
                     justifyContent: 'center'}}
            onPress={ () => setText('Fui pressionado!') }>
                <Text style={{ textAlign: 'center' }}> {texto} </Text>
        </TouchableOpacity>
    )

}

export default FunctionButton
```

Function component com o uso de Hooks para indicar a existência de um *state*

Class ou function component?

- Qual tipo de component devemos utilizar?
- Você quem escolhe! *(por enquanto)*

Em resumo...

- Componentes são códigos encapsulados, e escritos de maneira declarativa, que evitam a repetição de código
- São a composição de lógica, visualização e estilização dentro de uma **tag <>**
- Possuem estado próprio, que é gerenciado automaticamente pelo React Native
- Para o React Native, tudo é analisado como componente!