Paradigmas de Programação
React Native

Componentes

Function Component e Stateful Component

Gil Eduardo de Andrade

CRIAÇÃO DE COMPONENTES

Introdução:

- Como visto nas primeiras aulas, o React é uma biblioteca JavaScript voltada a criação de interfaces e estruturada via componentes;
- O React possibilita que os desenvolvedores criem aplicações através da utilização e criação de vários componentes;

CRIAÇÃO DE COMPONENTES

Introdução:

- Neste contexto, o React possibilita a criação de dois tipos de componentes:
  - *Functional Component* ou *Stateless Functional component*: são componentes que não possuem estado interno (*state*) – componentes baseados numa função;
  - *Class Component* ou *Stateful Component*: são componentes que possuem estado interno que podem ser alterado ao longo da execução da aplicação – componentes baseados numa classe;

CRIAÇÃO DE COMPONENTES

Organização dos Diretórios:

- Por questões de boas práticas de programação, a primeira etapa para criação dos componentes é criarmos os diretórios que irão receber componentes, imagens, estilos, etc.;
- Sendo assim, criamos um diretório *"src"* e dentro dele adicionamos os outros diretórios: *"components"*, *"pages"*, *"img"*, *"styles"*, *"util"*;

CRIAÇÃO DE COMPONENTES

Organização dos Diretórios:

- A organização apresentada e nomes dos diretórios são o padrão normalmente encontrado. Abaixo uma rápida descrição:
  - *components*: componentes criados da aplicação;
  - *pages* ou *screens*: telas da aplicação;
  - *img*: imagens da aplicação;
  - *styles*: arquivos de estilo da aplicação;
  - *util*: outras classes e funcionalidades gerais;

CRIAÇÃO DE COMPONENTES

***Functional Component*:**

- São criados no formato de função, podendo ou não utilizar as *props*. Veja os exemplos a seguir:

```
const FunctionalComponent = () => {
    return (
        <View> </View>
    );
}
```

Componente Funcional que não possui propriedades (***props***). Utiliza o conceito de Arrow Function. Como não é uma classe, não possui método render(), e assim já retorna o que deve ser renderizado.

```
const FunctionalComponent = (props) => {
    return (
        <View> props.valor </View>
    );
}
```

Componente Funcional que possui propriedades (***props***). No exemplo é considerada uma propriedade *valor*. Observe que ***não usamos o this.props*** porque não trata-se de uma classe.

```
export default FunctionalComponent;
```

Exporta o *functional* para que possa ser importado por outro componente.

CRIAÇÃO DE COMPONENTES

***Class/Stateful Component*:**

- São criados no formato de classe e possuem estado interno (*state*). Veja o exemplo a seguir:

```
class ClassComponent extends React.Component {

    constructor(props) {
      super(props);

      this.state = {};
    }

    render() {
        return (
            <View></View>
        )
    }
}
```

Por se tratar de uma classe possui um método construtor que recebe como parâmetro suas propriedades (*props*). Como visto em aulas anteriores o estado (*state*) do componente é definido, por padrão, dentro do método construtor. Componentes de estado possuem um método *render()* onde são definidos o *layout* e os componentes que serão renderizados.

```
export default ClassComponent;
```

Exporta o *class* para que possa ser importado por outro componente

Componentes na Prática

Continuando Desenvolvimento: App

www.gileduardo.com.br/ifpr/pp_rn/downloads/pp_rn_exapp06.zip

FUNCTIONAL: APP EXEMPLO

Calculadora *(App.js)*

```
import Display from './src/components/Display';
render() {
  return (
    <View style={ styles.container }>
      <View style={ styles.title }>
        <Text style={ styles.text } > Calculadora </Text>
      </View>
      <Display valor={ this.state.display } />
      <Keyboard onPress={ (val) => this.onPress(val) } />
    </View>
  );
}
```

Importa o componente *Display* do diretório *"src/components"*

Invoca o componente *Display* e passa como propriedade *(props)* para ele o *state display* da classe *App.js*, que contém os dados que deverão ser exibidos dentro do componente invocado *<Display>*. O state é passado via propriedade *valor.*

(Display.js)

```
const Display = (props) => {
    return (
        <View style={ styles.display }>
            <Text style={ styles.text } > { props.valor } </Text>
        </View>
    );
}
```

O componente *<Display>* renderiza o *props valor* recebido da classe *App.js* quando o mesmo foi invocado.

FUNCTIONAL: APP EXEMPLO

Calculadora *(App.js)*

```
import Keyboard from './src/components/Keyboard';
```

Importa o componente ***Keyboard*** do diretório ***"src/components"***

```
render() {
  return (
    <View style={ styles.container }>
      <View style={ styles.title }>
        <Text style={ styles.text } > Calculadora </Text>
      </View>
      <Display valor={ this.state.display } />
      <Keyboard onPress={ (val) => this.onPress(val) } />
    </View>
  );
}
```

Invoca o componente ***Keyboard*** e passa como propriedade ***(props)*** para ele o ***método onPress()*** da classe ***App.js***, que contém a rotina que deve ser executada dentro do componente invocado ***<Keyboard>*** quando um dos botões for pressionado. O método ***onPress()*** é passado via propriedade ***onPress***.

(Keyboard.js)

```
<View style={ styles.button }>
    <Button
        onPress={ () => { this.props.onPress(10) } }
        title='CC'
        color="green"
    />
</View>
```

O componente ***<Keyboard>*** renderiza os ***Buttons*** da calculadora e define que o método a ser executado quando estes forem pressionados é o ***props onPress*** recebido.

CRIAÇÃO DE COMPONENTES

Exemplos Utilizados no Documento

http://www.gileduardo.com.br/ifpr/pp_rn/downloads/pp_rn_exdoc06.zip

Código-fonte do App Exemplo: Calculadora

http://www.gileduardo.com.br/ifpr/pp_rn/downloads/pp_rn_exapp06.zip

Exercício sobre o Conteúdo

http://www.gileduardo.com.br/ifpr/pp_rn/downloads/pratica06.pdf