Paradigmas de Programação
React Native

Props / State

Propriedade e Estado do Componente

Gil Eduardo de Andrade

PROPS / STATE

Introdução – props:

- Os componentes, em *React Native*, podem receber dados ou herdar características através de uma funcionalidade denominada ***props***;
- A ***props*** ou ***propriedades*** são valores recebidos pelos componentes e ***estáticos*** em seu contexto;
- As ***props*** podem ser vistas como parâmetros recebidos pelo componente e que permitem efetuar algum tipo de configuração;

PROPS / STATE

Introdução – props:

- A ***props*** de um componente pode ser acessada de maneira global dentro do seu contexto, ou seja, por qualquer método criado dentro da classe;
- Ela pode ser vista como um ***array*** de valores onde utilizamos ***"."*** para indicar qual propriedade recebida queremos acessar;

PROPS / STATE

Exemplo – props:

```
<Button
  onPress={ () => { alert('Botão √') } }
  title='√'
  color="#143414"
/>
```

Se analisarmos parte da codificação utilizada na aula anterior para criar a interface da calculadora, podemos visualizar a utilização de ***props*** ao criarmos os componentes ***<Button>***. Observe que valores de configuração para o componente <Button> são passados pelas propriedades (props): ***"onPress()"***, ***"title"*** e ***"color"*** .

Ao receber esses valores o componente <Button> pode acessá-los internamente via ***props*** que armazena todos os dados passados num formato parecido com um array. Para especificar, dentro da codificação do componente, qual valor do props deseja-se obter basta utilizar o nome reservado ***props*** seguindo de ponto e do nome da propriedade passado. Exemplo: ***this.props.title*** retorna ***'√'*** . Na próxima aula (criação de componentes) veremos em mais detalhes o conceito de ***props***.

A propriedade ***title*** define o texto exibido pelo ***Button***, a propriedade ***color*** a sua cor (nesse caso verde escuro) e a propriedade ***onPress()*** permite definir qual método deve ser executado quando o botão é pressionado, ou seja, é possível passar um método via ***props*** a um componente .

PROPS / STATE

Introdução – state:

- Os componentes, em *React Native*, possuem valores que podem ser armazenados internamente, esse armazenamento é feito via ***state***;
- O ***state*** representa o estado interno de um componente, ele armazena valores dinâmicos, que são alterados ao longo da execução da aplicação;
- O ***state*** pode ser encarado, de modo geral, como os ***atributos*** de uma classe;

PROPS / STATE

Exemplo – state:

- A declaração do ***state*** ocorre dentro do construtor do componente, seguindo o formato apresentado a seguir:

```
constructor(props) {
  super(props);

  this.state = {

  };
}
```

Os dados (display, valor e operação) são armazenados como atributos de um objeto *{}* dentro do *state*.

```
this.state = {
  display: '0',
  valor: [null, null, null],
  operacao: null,
};
```

PROPS / STATE

Introdução – state:

- Para acessarmos os valores armazenados no ***state*** utilizamos ***"."*** para indicar qual atributo queremos acessar. Exemplo: ***this.state.display***;
- Para alterarmos o(s) valor(es) do(s) atributo(s) do ***state*** utilizamos o método ***setState()***, passando um objeto como parâmetro;

PROPS / STATE

Exemplo – state:

- A utilização do método do ***setState()*** segue o formato apresentado a seguir:

```
this.setState({
  display: calc,
  valor: [null, null, aux],
  operacao: null,
});
```

Podemos utilizar o *setState()* de duas maneiras: *(1)* para modificar todos os atributos de estado do componente, passando como parâmetro um objeto com todo os novos valores de estado.

```
this.setState({ operacao: 'sub' });
```

***(2)* para modificar apenas um o mais atributos de estado do componente, passando como parâmetro um objeto com um ou mais novos valores de estado. Nesse caso os outros estados tem seu valor mantido.**

Props/State na Prática

Continuando Desenvolvimento: App

www.gileduardo.com.br/ifpr/pp_rn/downloads/pp_rn_exapp05.zip

LAYOUT: APP EXEMPLO

Calculadora (Código)

```
<View style={ styles.container }>
  <View style={ styles.title }>
    <Text style={ styles.text } > Calculadora </Text>
  </View>
  <View style={ styles.display }>
    <Text style={ styles.text } > { this.state.display } </Text>
  </View>
  <View style={ [styles.title, styles.line] }>
    <View style={ styles.button }>
      <Button
        onPress={ () => { this.onPress(10) } }
        title='CC'
        color="green"
      />
    </View>
```

O *state display* é utilizado para armazenar os dados que devem ser apresentados no visor da calculadora. Isso é comum no *React* porque logo após invocarmos o método *setState()* e para *modificar* o valor do *display,* o *React* automaticamente *executa* método *render()* , redesenhando o componente e *atualizando* o valor apresentado no *visor* da calculadora.

Um novo método, denominado *onPress()* foi criado dentro do componente Calculadora, ele é utilizado por todos os botões quando os mesmos são pressionados. Ele recebe como parâmetro um *id* que lhe permite identificar qual botão foi pressionado no momento e gerou sua chamada.

LAYOUT: APP EXEMPLO

Calculadora (Código)

```
onPress(id) {

  switch(id) {
      // Número
      default:
              let aux = this.state.valor;
              let val;

              // Primeiro Valor
              if(this.state.operacao == null) {
                val = (aux[0] * 10) + id;
                aux[0] = val;
              }
              // Segundo Valor
              else {
                val = (aux[1] * 10) + id;
                aux[1] = id;
              }

              this.setState({
                display: val,
                valor: aux,
              });
              break;
    }
```

Quando um número é pressionado, o *state* ***valor*** (array com tamnaho três) é armazenado numa variável denominada ***aux***.

Para determinar se trata-se do primeiro valor digitado pelo usuário, ou do segundo, é verificado se a operação já foi definida, ou seja, se o *state* ***operacao*** é igual a ***null***.

Se o *state* ***operacao*** é igual a ***null*** então trata-se do primeiro valor, que fica armazenado na posição ***"0"*** do array.

Se o *state* ***operacao*** é diferente de ***null*** então trata-se do segundo valor, que fica armazenado na posição ***"1"*** do array.

Ao final o *state* ***display*** é atualizado via método ***setStat()***, que invoca o método ***render()*** e atualiza o valor mostrado no visor da calculadora.

FLEXBOX E Estilos

Exemplos Utilizados no Documento

http://www.gileduardo.com.br/ifpr/pp_rn/downloads/pp_rn_exdoc05.zip

Código-fonte do App Exemplo: Calculadora

http://www.gileduardo.com.br/ifpr/pp_rn/downloads/pp_rn_exapp05.zip

Exercício sobre o Conteúdo

http://www.gileduardo.com.br/ifpr/pp_rn/downloads/pratica05.pdf