

Entrada de caracteres

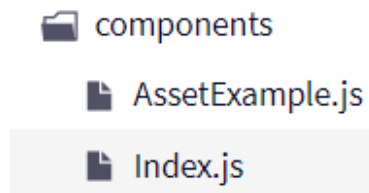
Ewerton J. Silva

ewerton.silva41@etec.sp.gov.br

Criando componente principal

Em um novo projeto dentro da pasta “components” crie um novo arquivo com o nome “Index.js” e insira o código apresentado ao lado.

Este componente vai retornar uma View com um Text dentro.



```
1 import * as React from 'react';
2 import { View, Text, StyleSheet } from 'react-native';
3
4 export default function Index() {
5   return(
6     <View style={styles.container} >
7       <Text style={styles.paragraph} >
8         Exemplo 4
9       </Text>
10    </View>
11  );
12 }
13
14 const styles = StyleSheet.create({
15   container: {
16     flex: 1,
17     justifyContent: 'center',
18     backgroundColor: '#FFCDD2',
19     padding: 8,
20   },
21   paragraph: {
22     margin: 24,
23     fontSize: 26,
24     fontWeight: 'bold',
25     textAlign: 'center',
26     color: '#B71C1C',
27   },
28 });
```

Em “App.js”, vamos importar o componente “Index.js” e utilizá-lo dentro do “App.js”. Em nossas aulas iremos manter sempre o padrão dos slides 2 e 3 sempre que iniciarmos um novo projeto, limpando o código e chamando um componente criado para realizarmos nossos exemplos.

Deixe o código de “App.js” conforme o exemplo ao lado, aqui iremos retornar uma “View” com o componente criado anteriormente.

```
1  import * as React from 'react';
2  import { View, StyleSheet } from 'react-native';
3  import Constants from 'expo-constants';
4
5  import Index from './components/Index';
6
7  export default function App() {
8    return (
9      <View style={styles.container}>
10        <Index />
11      </View>
12    );
13  }
14
15  const styles = StyleSheet.create({
16    container: {
17      flex: 1,
18      justifyContent: 'center',
19      paddingTop: Constants.statusBarHeight,
20      backgroundColor: '#B71C1C',
21      padding: 8,
22    },
23  });
```



Componente para inserção de texto

Agora iremos adicionar um componente que permite a escrita. Realize o import de um “TextInput”

```
1  import * as React from 'react';  
2  import { View, Text, StyleSheet, TextInput } from 'react-native';
```



Componente para inserção de texto

Adicione os componentes apontados ao lado.

Com o “TextInput” não temos a necessidade de abrir e fechar a tag como em um “Text”, observe que apontamos um estilo no momento da declaração deste componente, e apesar de não ser possível visualizá-lo, nenhum erro é apresentado.

```
4  export default function Index() {  
5    return(  
6      <View style={styles.container} >  
7        <Text style={styles.paragraph} >  
8          Exemplo 4  
9        </Text>  
10  
11        {  
12          <Text style={styles.txtSaida} >  
13            Exemplo 4  
14          </Text>  
15          <TextInput style={styles.txtEntrada} />  
16        }  
17      </View>  
18    );  
19  }
```

Estilo para o Text e o TextInput

Insira o código apresentado abaixo para formatação do componentes, no local correspondente.

```
35   txtSaida: {  
36     margin: 24,  
37     fontSize: 22,  
38     fontWeight: 'bold',  
39     textAlign: 'center',  
40     color: '#E53935',  
41   },  
42   txtEntrada: {  
43     borderWidth: 4,  
44     textAlign: 'center',  
45     fontSize: 22,  
46     borderColor: '#B71C1C',  
47     height: 40,  
48     color: '#E53935',  
49     borderRadius: 10,  
50   },
```



Propriedades do TextInput

Experimente alterar os valores da propriedade `TextInput` e visualize os resultados testando o aplicativo pelo Android!

`keyboardType`

Determines which keyboard to open, e.g. `numeric`.

See screenshots of all the types [here](#).

The following values work across platforms:

- `default`
- `number-pad`
- `decimal-pad`
- `numeric`
- `email-address`
- `phone-pad`

<https://reactnative.dev/docs/textinput>



Trabalhando com State

Realize inserção/alteração nas linhas apontadas para definir um “State” que vai armazenar o valor inserido no “TextInput” e na sequência declare-o como saída no “Text” apontado.

```
1 import React, { useState } from 'react';
2 import { View, Text, StyleSheet, TextInput } from 'react-native';
3
4 export default function Index() {
5
6   → const [textoEscrito, setTextoEscrito] = useState('Texto programado');
7
8   return(
9     <View style={styles.container} >
10       <Text style={styles.paragraph} >
11         Exemplo 4
12       </Text>
13
14       <Text style={styles.txtSaida} >
15         → {textoEscrito}
16       </Text>
17
18       <TextInput style={styles.txtEntrada} />
19
20     </View>
21   );
22 }
```


Atualização do valor inserido

Para garantir que o “State” tenha o valor atualizado a cada caractere inserido iremos programar no evento “onChangeText” do “TextInput”.

Aqui inserimos uma arrow function que passa como parâmetro o valor inserido no componente e na sua execução altera o “State” com o valor do parâmetro que foi recebido na entrada de caracteres.

Teste o programa para verificar se o valor é atualizado a cada nova inserção de caractere.

```
<TextInput  
  style={styles.txtEntrada}  
  onChangeText={ (entrada) => setTextoEscrito(entrada)}  
/>
```



Exemplo 4

A cada caractere

A cada caractere

Controlando o momento de exibição do texto

Não precisamos atualizar o valor exibido em tela sempre que um valor é inserido, vamos otimizar para que isso só ocorra quando um botão for acionado, para isso devemos importar um “TouchableOpacity” e criar um novo “State”, este vai armazenar o valor que será exibido ao tocar no botão. Também altere o valor inicial do “State textEscrito”.

```
1  import React, { useState } from 'react';
2  import { View, Text, StyleSheet, TextInput, TouchableOpacity } from 'react-native';
3
4  export default function Index() {
5
6      {const [textoEscrito, setTextoEscrito] = useState('');
7      const [mensagem, setMensagem] = useState('Texto programado');
8
9      return(
```



Inserindo botão para exibição de mensagem

Agora o valor apresentado no “Text” deve ser altero, pois iremos exibir o “State mensagem”.

```
<Text style={styles.txtSaida} >  
  {mensagem}   
</Text>  
  
<TextInput  
  style={styles.txtEntrada}  
  onChangeText={ (entrada) => setTextoEscrito(entrada)}  
>  
  
<TouchableOpacity style={styles.button} onPress={() => setMensagem(textoEscrito)}>  
  <Text style={styles.textButton}> Exibir texto </Text>  
</TouchableOpacity>
```

Inserindo botão para exibição de mensagem

No “TextInput” não é preciso alterar nada, pois ainda é necessário atualizar o “State entrada” a cada caractere inserido, pois é assim que temos acesso ao valor no momento em que clicarmos no botão.

```
<Text style={styles.txtSaida} >
  {mensagem}
</Text>

<TextInput
  style={styles.txtEntrada}
  onChangeText={ (entrada) => setTextoEscrito(entrada)}
/>

<TouchableOpacity style={styles.button} onPress={() => setMensagem(textoEscrito)}>
  <Text style={styles.textButton}> Exibir texto </Text>
</TouchableOpacity>
```

Inserindo botão para exibição de mensagem

Adicione o código apontado abaixo para inserir o botão, observe que no evento “onPress” passamos o “State textoEscrito” para p “State mensagem”, permitindo que o valor exibido em tela seja alterado.

```
<Text style={styles.txtSaida} >
  {mensagem}
</Text>

<TextInput
  style={styles.txtEntrada}
  onChangeText={ (entrada) => setTextoEscrito(entrada)}
/>

<TouchableOpacity style={styles.button} onPress={() => setMensagem(textoEscrito)}>
  <Text style={styles.textButton}> Exibir texto </Text>
</TouchableOpacity>
```



Aplicar estilo

Insira os objetos de estilização apresentados abaixo e teste o programa.

```
62 button: {  
63   backgroundColor: '#E53935',  
64   height: 40,  
65   justifyContent: 'center',  
66   borderRadius: 10,  
67   marginTop: 20,  
68 },  
69 buttonText: {  
70   fontSize: 22,  
71   color: '#FFCDD2',  
72   textAlign: 'center',  
73 },
```

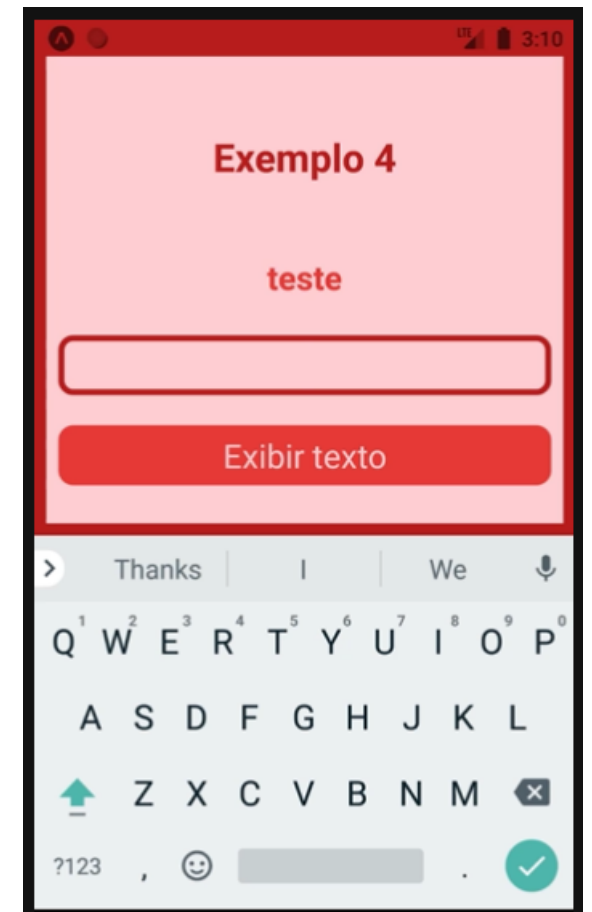


Limpando o valor do TextInput

Para limpar o texto será necessário alterar a estrutura conforme apontado abaixo:

```
<TextInput
  style={styles.txtEntrada}
  onChangeText={ (entrada) => setTextoEscrito(entrada)}
  value={textoEscrito}
/>


<TouchableOpacity
  style={styles.button}
  onPress={() => {
    setMensagem(textoEscrito);
    setTextoEscrito('');
  }}
>
  <Text style={styles.textButton}> Exibir texto </Text>
</TouchableOpacity>
```



Criando função

Para simplificar o aplicativo crie uma função entre a declaração das constantes e o “return” do componente e altere a chamada do evento “onPress” do botão conforme apresentado abaixo:

```
4  export default function Index() {  
5  
6      const [textoEscrito, setTextoEscrito] = useState('');  
7      const [mensagem, setMensagem] = useState('Texto programado');  
8  
9      {  
10         function ExibeTexto() {  
11             setMensagem(textoEscrito);  
12             setTextoEscrito('');  
13         }  
14     }  
15     return(  
16
```



```
<TouchableOpacity style={ styles.button} onPress={() => ExibeTexto() } >  
  <Text style={styles.textButton}> Exibir texto </Text>  
</TouchableOpacity>
```


O código apontado ao lado não precisa ser inserido, serve apenas de exemplo.

Existe outra forma de trabalhar com funções, conforma apontado na imagem ao lado, observe e compare as alterações necessárias para trabalhar definindo a função em uma variável.

Existem exemplos na internet com essa notação.

```
4 export default function Index() {
5
6   const [textoEscrito, setTextoEscrito] = useState('');
7   const [mensagem, setMensagem] = useState('Texto programado');
8
9   const ExibeTexto = () => {
10     setMensagem(textoEscrito);
11     setTextoEscrito('');
12   }
13
14   return(
15     <View style={styles.container} >
16       <Text style={styles.paragraph} >
17         Exemplo 4
18       </Text>
19
20       <Text style={styles.txtSaida} >
21         {mensagem}
22       </Text>
23
24       <TextInput
25         style={styles.txtEntrada}
26         onChangeText={ (entrada) => setTextoEscrito(entrada)}
27         value={textoEscrito}
28       />
29
30       <TouchableOpacity style={ styles.button} onPress={ExibeTexto} >
31         <Text style={styles.textButton}> Exibir texto </Text>
32       </TouchableOpacity>
33
34     </View>
35   );
36 }
```



Atividade

Crie um aplicativo que permita inserir o nome e sobrenome, após clicar em “Exibir texto” o nome completo deve ser apresentado conforme o exemplo ao lado.

Exemplo 4

Inserir o nome e sobrenome

Nome

Sobrenome

This is a mobile application interface with a light pink background and a dark red border. At the top, there's a status bar with icons for signal, battery, and time (3:27). Below the status bar, the title "Exemplo 4" is centered in dark red. Underneath, the instruction "Inserir o nome e sobrenome" is also centered in dark red. The form consists of two input fields: the first is labeled "Nome" and contains the text "Ewerton José"; the second is labeled "Sobrenome" and contains the text "da Silva". At the bottom of the form is a red button with the text "Exibir texto".

Exemplo 4

Ewerton José da Silva

Nome

Sobrenome

This is the same mobile application interface as the previous one, but it shows the result after clicking the "Exibir texto" button. The title "Exemplo 4" remains at the top. The instruction "Inserir o nome e sobrenome" has been replaced by the full name "Ewerton José da Silva" in dark red. The two input fields are now empty, with only their labels "Nome" and "Sobrenome" visible. The red "Exibir texto" button remains at the bottom.

Referências

- <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Fun%C3%A7%C3%B5es>
- <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Operators/function>
- [https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Functions/Arrow functions](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Functions/Arrow_functions)