Paradigmas de Programação
React Native

FLEBOX: Layout React Native

justify-content / align-items / flex-direction

Gil Eduardo de Andrade

FLEBOX

Introdução:

- O Flexbox (*CSS Flexible Box Layout Model*) tem como objetivo organizar os elementos de um contêiner (/ <View>) quando o layout criado para uma aplicação necessita ser visualizado em diferentes dispositivos e formatos (resoluções) de tela;

FLEBOX

Introdução:

- A ideia por trás do Flexbox é relativamente simples, nela os elemento adicionados a um determinado contêiner podem ser posicionados em qualquer direção e conter dimensões flexíveis, o que permite que os mesmos sejam adaptáveis ao tamanho de tela do dispositivo;

FLEBOX

Propriedades:

- O Flexbox possui três (principais) propriedades:
  - ***justifyContent***: efetua o alinhamento horizontal dos itens;
  - ***alignItems***: efetua o alinhamento vertical dos itens;
  - ***flexDirection***: define a direção com que os itens serão colocados dentro contêiner;

FLEBOX

Propriedade – *justifyContent* (horizontal):

- Aceita os seguintes valores :
  - ***flex-start***: os itens são alinhados no início (topo ou esquerda) do contêiner;
  - ***flex-end***: os itens são alinhados no final (rodapé ou direita) do contêiner;
  - ***center***: os itens são alinhados no centro do contêiner;
  - ***space-between***: os itens são exibidos com espaçamento igual entre eles;
  - ***space-around***: os itens são exibidos com espaçamento igual no seu em torno;

FLEBOX

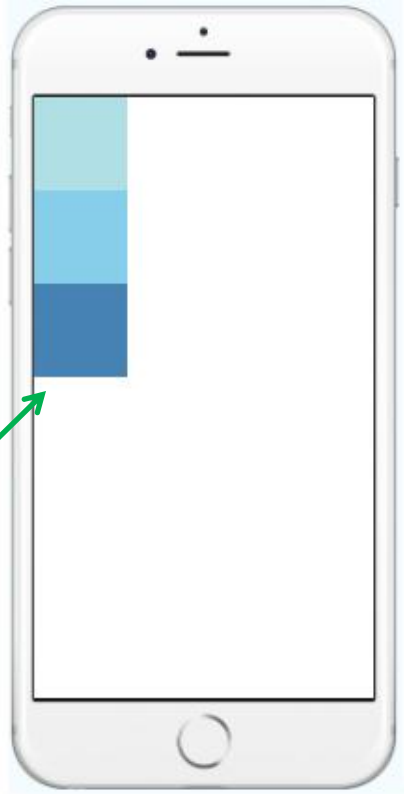
Propriedade – *justifyContent*: 'flex-start'

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

Observe que o layout possui uma propriedade adicional *flex: 1*, ela é utilizada para indicar que o contêiner <View> irá utilizar todo o espaço da tela do dispositivo.

```
<View style={ {flex:1, justifyContent: 'flex-start'} }>
  <View style={{width: 50, height: 50, backgroundColor: 'powderblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'skyblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'steelblue'}} />
</View>
```

Observe que o alinhamento dos itens foi vertical, isso ocorre porque, por padrão, o Flexbox insere um item abaixo do outros, *em coluna*.



Download Código-exemplo

FLEBOX

Propriedade – *justifyContent*: 'flex-start'

Se utilizarmos a propriedade ***flexDirection*** com o valor ***row*** é possível mudar a direção com que os itens são inseridos, nesse caso na horizontal ou ***em linha***.

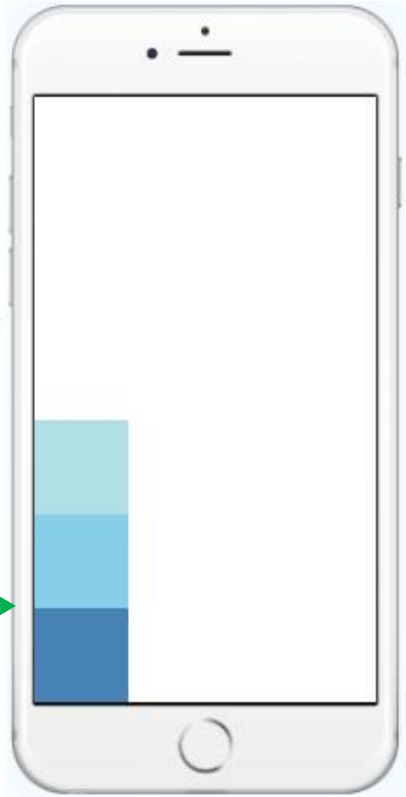
```
<View style={ {flex:1, flexDirection: 'row', justifyContent: 'flex-start'} }>
  <View style={{width: 50, height: 50, backgroundColor: 'powderblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'skyblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'steelblue'}} />
</View>
```

FLEBOX

Propriedade – *justifyContent*: ‘flex-end’

```
<View style={ {flex:1, justifyContent: 'flex-end'} }>
  <View style={{width: 50, height: 50, backgroundColor: 'powderblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'skyblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'steelblue'}} />
</View>
```

Os itens são alinhados no final do contêiner

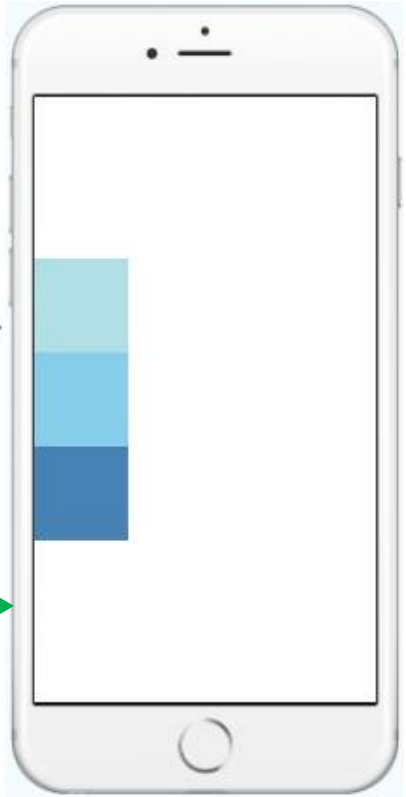


FLEBOX

Propriedade – *justifyContent*: ‘center’

```
<View style={ {flex:1, justifyContent: 'center'} }>
  <View style={{width: 50, height: 50, backgroundColor: 'powderblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'skyblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'steelblue'}} />
</View>
```

Os itens são alinhados no centro do contêiner

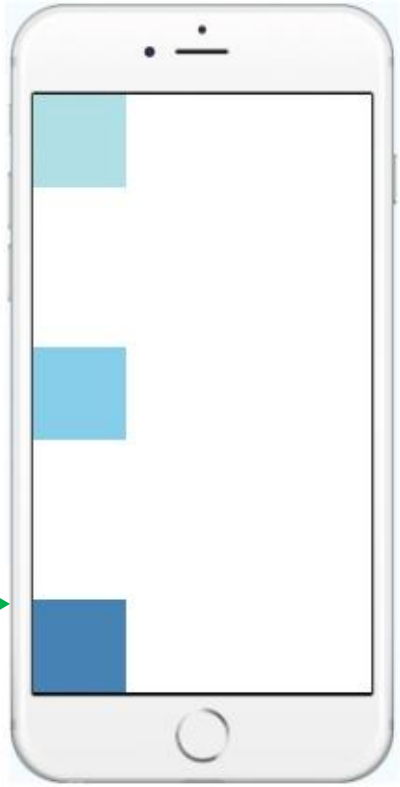


FLEBOX

Propriedade – *justifyContent*: ‘space-between’

```
<View>
  <View style={{width: 50, height: 50, backgroundColor: 'powderblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'skyblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'steelblue'}} />
</View>
```

Os itens são alinhados com espaçamento igual entre eles

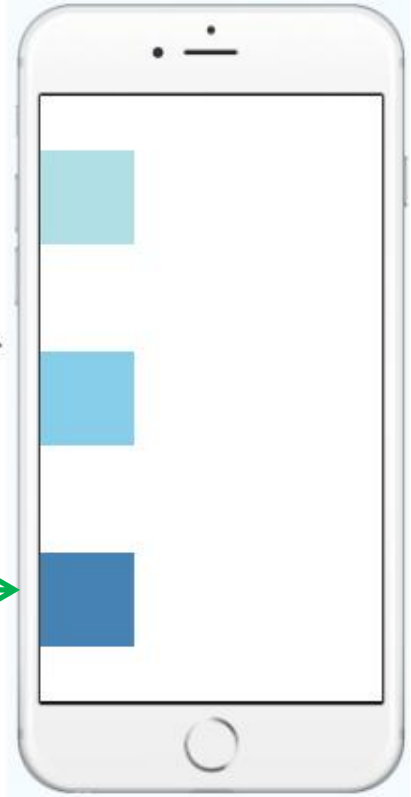


FLEBOX

Propriedade – *justifyContent*: ‘space-around’

```
<View style={ {flex:1, justifyContent: 'space-around'} }>
  <View style={{width: 50, height: 50, backgroundColor: 'powderblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'skyblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'steelblue'}} />
</View>
```

Os itens são alinhados com espaçamento igual e sua volta

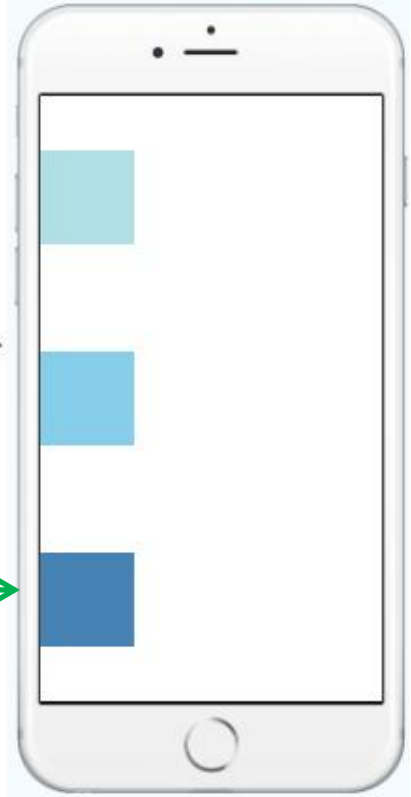


FLEBOX

Propriedade – *justifyContent*: ‘space-around’

```
<View style={ {flex:1, justifyContent: 'space-around'} }>
  <View style={{width: 50, height: 50, backgroundColor: 'powderblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'skyblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'steelblue'}} />
</View>
```

Os itens são alinhados com espaçamento igual e sua volta



FLEBOX

Propriedade – *alignItems* (vertical):

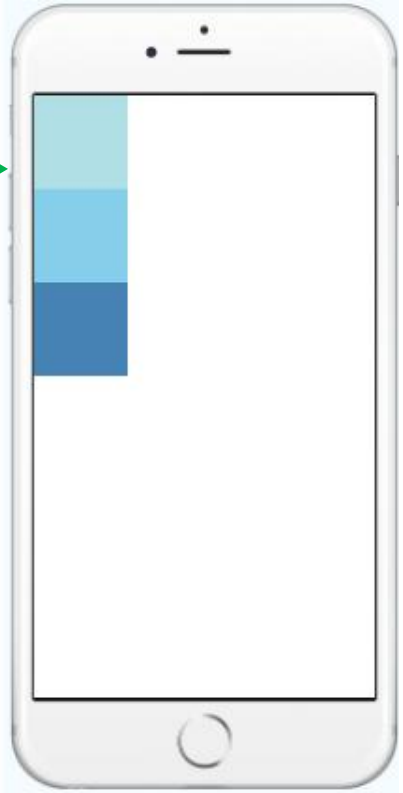
- Aceita os seguintes valores :
  - ***flex-start***: os itens são alinhados na parte superior (topo) do contêiner;
  - ***flex-end***: os itens são alinhados na parte inferior (rodapé) do contêiner;
  - ***center***: os itens são alinhados no cento (vertical) do contêiner;
  - ***baseline***: os itens são alinhados na base do contêiner;
  - ***stretch***: os itens são esticados para preencher todo o contêiner;

FLEBOX

Propriedade – *alignItems*: ‘flex-start’

Os itens são alinhados no início do contêiner

```
<View style={ {flex:1, alignItems: 'flex-start'} }>
  <View style={{width: 50, height: 50, backgroundColor: 'powderblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'skyblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'steelblue'}} />
</View>
```

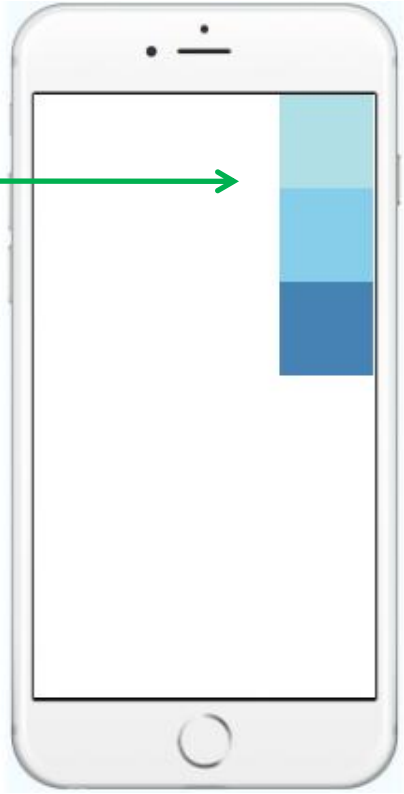


FLEBOX

Propriedade – *alignItems*: ‘flex-end’

Os itens são alinhados no final do contêiner

```
<View style={ {flex:1, alignItems: 'flex-end'} }>
  <View style={{width: 50, height: 50, backgroundColor: 'powderblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'skyblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'steelblue'}} />
</View>
```

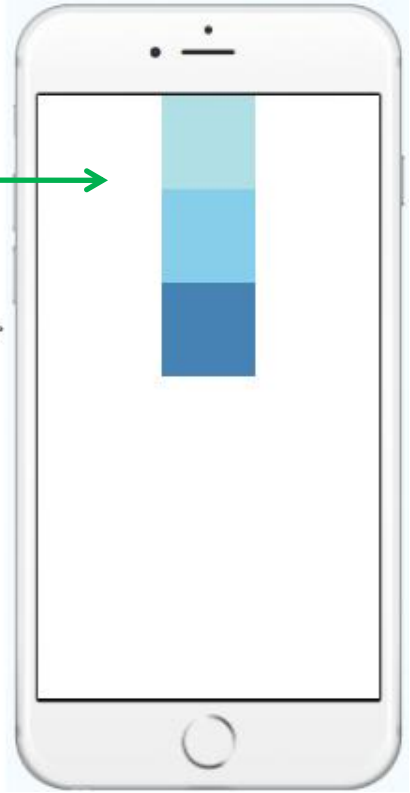


FLEBOX

Propriedade – *alignItems*: ‘center’

Os itens são alinhados no centro no contêiner

```
<View style={ {flex:1, alignItems: 'center'} }>
  <View style={{width: 50, height: 50, backgroundColor: 'powderblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'skyblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'steelblue'}} />
</View>
```

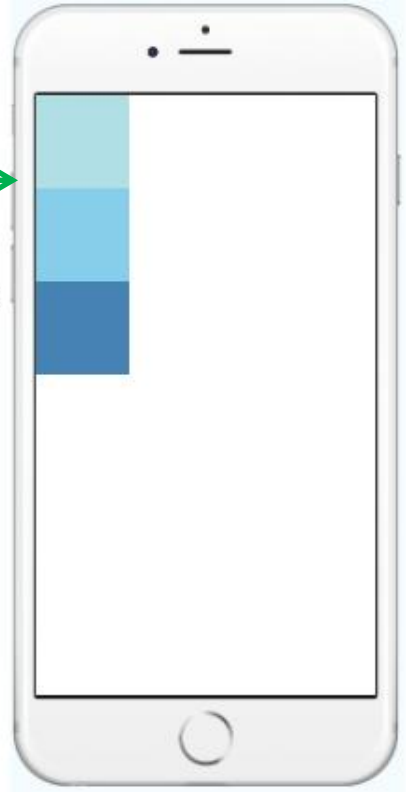


FLEBOX

Propriedade – *alignItems*: ‘baseline’

Os itens são alinhados na base do contêiner

```
<View style={ {flex:1, alignItems: 'baseline'} }>
  <View style={{width: 50, height: 50, backgroundColor: 'powderblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'skyblue'}} />
  <View style={{width: 50, height: 50, backgroundColor: 'steelblue'}} />
</View>
```

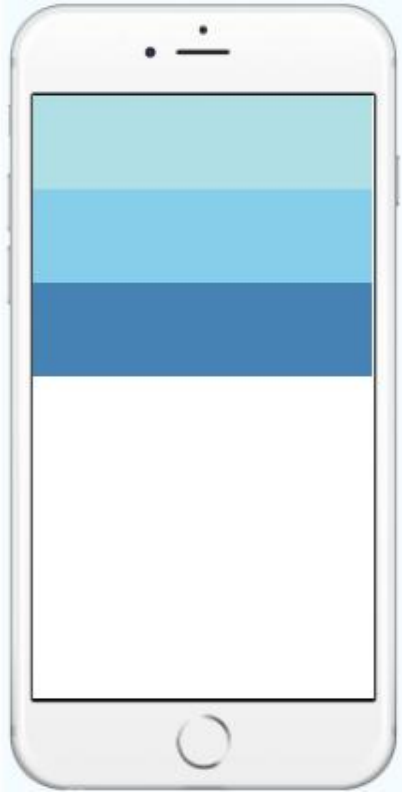


FLEBOX

Propriedade – *alignItems*: ‘stretch’

Observe que para facilitar a compreensão do parâmetro *stretch* foi removida a largura dos quadriláteros, para que assim eles preenchessem, horizontalmente todo o contêiner principal.

```
<View style={ {flex:1, alignItems: 'stretch'} }>
  <View style={{height: 50, backgroundColor: 'powderblue'}} />
  <View style={{height: 50, backgroundColor: 'skyblue'}} />
  <View style={{height: 50, backgroundColor: 'steelblue'}} />
</View>
```



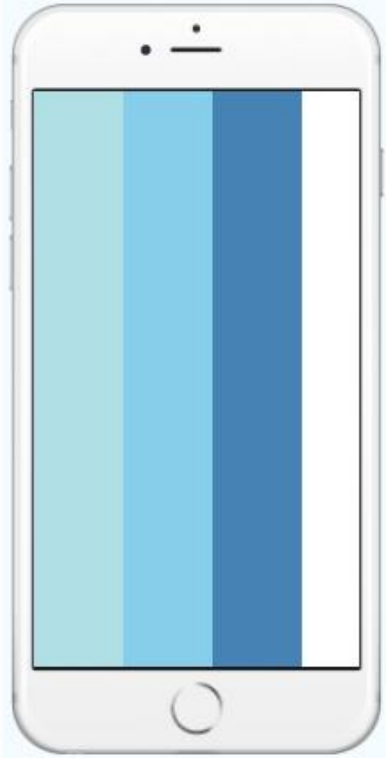
Os itens foram esticados para preencher todo o contêiner na horizontal.

FLEBOX

Propriedade – *alignItems*: ‘stretch’

Aqui é apresentado o uso do ***stretch*** com o ***flexDirection*** configurado como ***‘row’*** . Além disso, foi definida apenas a largura, diferente do exemplo anterior que configurou a altura.

```
<View style={ {flex:1, flexDirection:'row', alignItems: 'stretch'} }>
  <View style={{width: 50, backgroundColor: 'powderblue'}} />
  <View style={{width: 50, backgroundColor: 'skyblue'}} />
  <View style={{width: 50, backgroundColor: 'steelblue'}} />
</View>
```



Os itens foram esticados para preencher todo o contêiner na vertical.

FLEBOX

Propriedade – *flexDirection* (direção):

- Aceita os seguintes valores :
  - ***row***: os itens são colocados seguindo o padrão de direção da escrita (esquerda para direita);
  - ***row-reverse***: os itens são colocados seguindo o padrão inverso de direção da escrita (direita para esquerda);
  - ***column***: os itens são colocados de cima para baixo;
  - ***column-reverse***: os itens são colocados de baixo para cima;

FLEBOX

Propriedade – *flexDirection*: 'row'

Os itens são alinhados em linha, da esquerda para direita.

```
<View style={ {flex:1, flexDirection:'row'} }>
  <View style={{width: 50, height:50, backgroundColor: 'powderblue'}} />
  <View style={{width: 50, height:50, backgroundColor: 'skyblue'}} />
  <View style={{width: 50, height:50, backgroundColor: 'steelblue'}} />
</View>
```

FLEBOX

Propriedade – *flexDirection*: ‘row-reverse’

Os itens são alinhados em linha, da direita para esquerda.

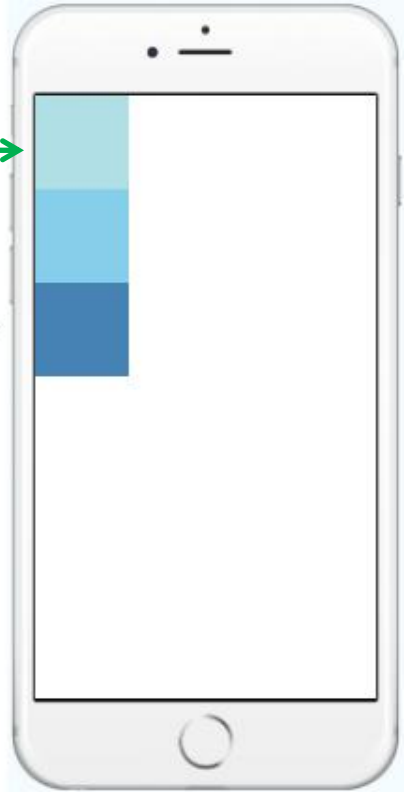
```
<View style={ {flex:1, flexDirection:'row-reverse'} }>
  <View style={{width: 50, height:50, backgroundColor: 'powderblue'}} />
  <View style={{width: 50, height:50, backgroundColor: 'skyblue'}} />
  <View style={{width: 50, height:50, backgroundColor: 'steelblue'}} />
</View>
```

FLEBOX

Propriedade – *flexDirection*: ‘column’

Os itens são alinhados de cima para baixo.

```
<View style={ {flex:1, flexDirection:'column'} }>
  <View style={{width: 50, height:50, backgroundColor: 'powderblue'}} />
  <View style={{width: 50, height:50, backgroundColor: 'skyblue'}} />
  <View style={{width: 50, height:50, backgroundColor: 'steelblue'}} />
</View>
```

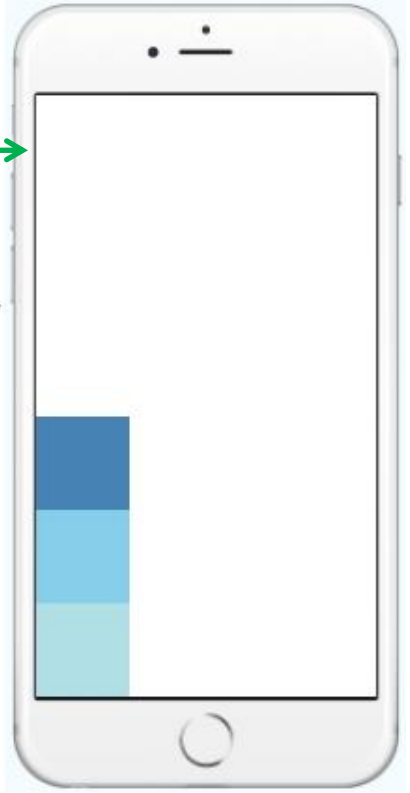


FLEBOX

Propriedade – *flexDirection*: ‘column-reverse’

Os itens são alinhados de baixo para cima.

```
<View style={ {flex:1, flexDirection:'column-reverse'} }>
  <View style={{width: 50, height:50, backgroundColor: 'powderblue'}} />
  <View style={{width: 50, height:50, backgroundColor: 'skyblue'}} />
  <View style={{width: 50, height:50, backgroundColor: 'steelblue'}} />
</View>
```



FLEXBOX na Prática

Criando Layout: App

www.gileduardo.com.br/ifpr/pp_rn/downloads/pp_rn_exapp04.zip

www.gileduardo.com.br/ifpr/pp_rn/downloads/pp_rn_exdoc04.zip

https://flexboxfroggy.com/

LAYOUT: APP EXEMPLO

Calculadora (Código)

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

export default class App extends React.Component {

  render() {
    return (
      <View style={styles.container}>

      </View>
    );
  }
}

const styles = StyleSheet.create({

  container: {
    flex: 1,
    backgroundColor: '#000',
    borderWidth: 3,
    borderColor: '#FFF',
  },
});
```

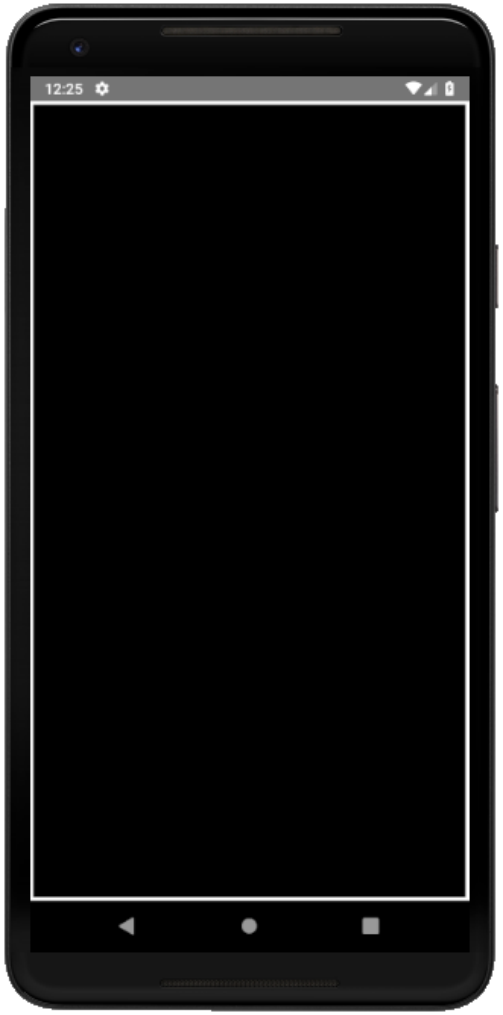
Todo componente *React* possui, obrigatoriamente, um método ***render()***, responsável por desenhar algo na tela .

Assim como acontece no HTML utilizamos um contêiner para receber todos os componentes que serão desenhados. Assim como em outras linguagens uma função possui um único retorno, e para que seja possível devolver textos, imagens, botões etc, agregamos tudo isso a um contêiner principal ***<View>***

Assim como acontece com HTML é possível definirmos estilos aos componentes, nos mesmos moldes do CSS. Contudo, no React Native as propriedades foram renomeadas retirando-se o hífen e colocando o segundo nome em caixa alta. Por exemplo: ***border-color*** se tornou ***borderColor***.

LAYOUT: APP EXEMPLO

Calculadora (Código + Execução)




```
const styles = StyleSheet.create({

  container: {
    flex: 1,
    backgroundColor: '#000',
    borderWidth: 3,
    borderColor: '#FFF',
  },
});
```

Define que todo o contêiner será preenchido. Como trata-se da *<View>* principal, define que toda a tela do smartphones receberá o estilo que está sendo definido

Define a *cor de fundo* do contêiner como preta, usando padrão RGB.

Define a *cor da borda* como branca, usando o padrão RGB.

Define a *espessura (3) da borda* que será colocada no contêiner.

LAYOUT: APP EXEMPLO

Calculadora (Código)

```
render() {
  return (
    <View style={ styles.container }>
      <View style={ styles.title }>
        <Text style={ styles.text } > Calculadora </Text>
      </View>
    </View>
  );
}
title: {
  flexDirection: 'row',
  justifyContent: 'center',
  height: 60,
  margin: 3,
  borderWidth: 3,
  borderColor: 'white',
},
text: {
  color: 'white',
  fontSize: 36,
  fontWeight: '900',
},
```

Um novo container (*<View>*) foi adicionado como filho do contêiner principal da aplicação. A esse segundo contêiner foi adicionado um componente filho do tipo *<Text>* que permite exibir o texto "Calculadora" na tela do dispositivo .

Dois novos estilos foram definidos: um para o segundo contêiner <View> *(title)* e outro para o componente <Text> *(text)*.

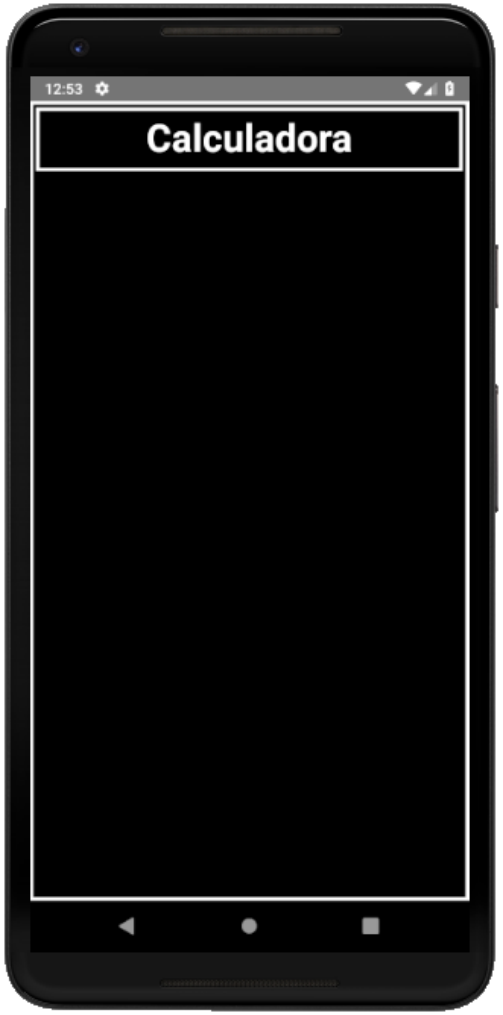
Indica que os componentes dentro do contêiner serão adicionados e exibidos *em linha (horizontal)*, um ao lado do outro.

Indica que o *alinhamento* dos componentes dentro do contêiner será *central*.

Indica que *altura* do contêiner é *60*.

LAYOUT: APP EXEMPLO

Calculadora (Código + Execução)




```
title: {
  flexDirection: 'row',
  justifyContent: 'center',
  height: 60,
  margin: 3,
  borderWidth: 3,
  borderColor: 'white',
},
text: {
  color: 'white',
  fontSize: 36,
  fontWeight: '900',
},
```

Define a *margem externa (3)* que o componente terá em relação aos outros componentes que estão a sua volta. Observe que há um pequeno espaço entre a borda da <View> principal do componente e a borda da <View> filho que contém o componente <Text>.

Define, respectivamente, a *espessura da borda (3)* do contêiner e a sua cor *(branca)*.

Define, respectivamente, a *cor da texto* "Calculadora", o *tamanho da fonte* e a *espessura da fonte*.

LAYOUT: APP EXEMPLO

Calculadora (Código)

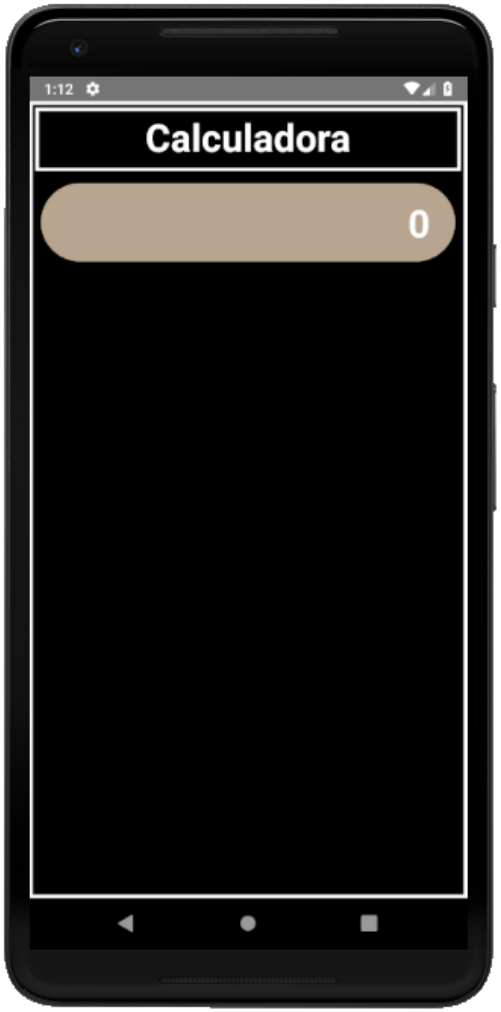
```
render() {
  return (
    <View style={ styles.container }>
      <View style={ styles.title }>
        <Text style={ styles.text } > Calculadora </Text>
      </View>
      <View style={ styles.display }>
        <Text style={ styles.text } > 0 </Text>
      </View>
    </View>
  );
}
display: {
  flexDirection: 'row',
  justifyContent: 'flex-end',
  alignItems: 'center',
  height: 80,
  paddingRight: 15,
  margin: 5,
  borderRadius: 180,
  borderColor: 'black',
  backgroundColor: '#B8A58F',
},
```

Um novo container (*<View>*) foi adicionado como filho do contêiner principal da aplicação. Ele também recebeu como filho um componente do tipo *<Text>*. Esse contêiner será utilizado como display (visor) da calculadora.

Indica, respectivamente, que os componentes dentro do contêiner serão adicionados e exibidos *em linha (horizontal)*, um ao lado do outro, que os componentes serão *inseridos da direita para esquerda*, que o *alinhamento* deles será *central* e que a *altura* do contêiner será de *80*.

LAYOUT: APP EXEMPLO

Calculadora (Código + Execução)




```
display: {
  flexDirection: 'row',
  justifyContent: 'flex-end',
  alignItems: 'center',
  height: 80,
  paddingRight: 15,
  margin: 5,
  borderRadius: 180,
  borderColor: 'black',
  backgroundColor: '#B8A58F',
},
```

Define a *margem interna direita (3)* para os componentes que serão inseridos no contêiner. Observe que o número "0", exibido pelo componente *<Text>*, está levemente deslocado para esquerda, ou seja, não está colado na borda direita do contêiner ao qual foi adicionado.

Define uma *curvatura (180)* para o contêiner, permitindo que suas bordas sejam arredondadas.

LAYOUT: APP EXEMPLO

Calculadora (Código)

```
<View style={ [styles.title, styles.line] }>
  <View style={ styles.button }>
    <Button
      onPress={ () => { alert('Botão Limpar') } }
      title='CC'
      color="green"
    />
  </View>
  <View style={ styles.button }>
    <Button
      onPress={ () => { alert('Botão Raiz') } }
      title='√'
      color="#143414"
    />
  </View>
line: {
  height: 48,
  margin: 5,
  marginTop: 15,
  borderColor: 'black',
},
button: {
  flex: 1,
  margin: 3,
}
```

Um novo container (*<View>*), para recepção da primeira linha de botões, foi adicionado ao contêiner principal da aplicação. Observe que ele está utilizando dois estilos (*um array*) ao mesmo tempo. Isso possibilita reaproveitar outros estilos já criados, o que é definido nos últimos estilos se sobrepõe aos anteriores caso aja redefinição de algum parâmetro.

Dois componentes do tipo ***<Button>*** foram adicionados, como filhos de um componente ***<View>***. Isso deve ser feito para que seja possível utilizar as propriedades do ***flexbox*** sobre eles, nesse caso ***flex: 1***. Como cada botão possui um contêiner com flex: 1 o espaço horizontal deles é dividido igualmente, pois cada um possui uma parte de duas.

Quando usamos um <Button> precisamos, obrigatoriamente, definir duas propriedades, ***"onPress()"*** que indica o que deve ser executado quando o botão for pressionado e ***"title"***, que indica o texto que deve ser exibido por ele.

LAYOUT: APP EXEMPLO

Calculadora (Código + Execução)


```
line: {
  height: 48,
  margin: 5,
  marginTop: 15,
  borderColor: 'black',
},
button: {
  flex: 1,
  margin: 3,
}
```

Define, respectivamente, a altura do contêiner que contém os botões, a ***margem externa (5)***, ***margem externa do topo (15)*** e a cor da borda.

A propriedade ***flex*** permite configurar o percentual total que cada item de um contêiner irá ocupar. Nesse exemplo cada botão possui ***flex: 1***, como são 4 botões temos o contêiner dividido em ***4 partes***, cada qual preenchida por um botão, ou seja, cada um com ***25% do total***. Para compreender melhor veja a última linha de botões da calculadora no próximo slide.

LAYOUT: APP EXEMPLO

Calculadora (Código + Execução)


```
<View style={ [styles.button, {flex: 1}] }>
  <Button
    onPress={ () => { alert('Botão 0') } }
    title='0'
    color="#143414"
  />
</View>
<View style={ [styles.button, , {flex: 3}] }>
  <Button
    onPress={ () => { alert('Botão =') } }
    title='='
    color="green"
  />
</View>
```

Como o *botão "0"* possui a propriedade *flex: 1* e o *botão "="* possui a propriedade *flex: 3*, temos o contêiner dividido em *4 partes*, onde o *botão "0"* ocupa ¼ do total, ou *25%*, e o *botão "="* ocupa ¾ do total, ou *75%*. Isso ocorre na horizontal porque o contêiner principal (*<View>*) no qual estão inseridos possui seu estilo (*style.title*) configurado com *flexDirection: 'row'*.

FLEXBOX E Estilos

Exemplos Utilizados no Documento

http://www.gileduardo.com.br/ifpr/pp_rn/downloads/pp_rn_exdoc04.zip

Código-fonte do App Exemplo: Calculadora

http://www.gileduardo.com.br/ifpr/pp_rn/downloads/pp_rn_exapp04.zip

Exercício sobre o Conteúdo

http://www.gileduardo.com.br/ifpr/pp_rn/downloads/pratica04.pdf