

Programação Mobile

React Native - Aula 04

Professor: João Felipe Bragança



Objetivos da Aula

- Entender os Componentes
- Entender os Tipos de Dimensões
- Conceitos de Layout com FlexBox

Propriedade style

- Podemos estilizar todos os Componentes Principais através da propriedade “style”.
- Os nomes dos estilos e seus valores geralmente batem com o CSS, porém utilizam um padrão diferente de escrita, o camelCase.(exemplos: backgroundColor, fontSize, justifyContent, alignItems, etc)
- Para declarar estilos “inline”, deve-se atribuir um par extra de {}.(exemplo: style={{flex: 1, padding: 16}}).

StyleSheet

- StyleSheet é uma abstração similar ao CSS. Com ele podemos criar objetos de estilização fora do componente, ou até mesmo em um arquivo separado, assim como CSS no desenvolvimento Web.
- Para utilizar, precisamos primeiramente importar.
`Import { StyleSheet } from "react-native"`
- Para criar nosso objeto, utilizamos

```
StyleSheet.create({  
  nome_estile: {  
    estilo1: valor,  
    estilo2: valor  
  }  
})
```

Inline x StyleSheet

```
export default function App() {  
  return (  
    <View style={styles.container}>  
      <Text style={styles.texto}>Hello World!!!</Text>  
    </View>  
  );  
}
```

- Estilos Inline são ideias para testes rápidos
- StyleSheet oferece melhor legibilidade e proporciona facilidade na hora da manutenção
- Projetos grande utilizam muitas linhas de código, separar os estilos melhora significativamente o desempenho da equipe, principalmente quando é preciso fazer manutenção

```
export default function App() {  
  return (  
    <View  
      style={{  
        flex: 1,  
        backgroundColor: "#e8e8e8",  
        justifyContent: "center",  
        alignItems: "center",  
      }}  
    >  
      <Text  
        style={{  
          fontSize: 20,  
          fontWeight: "bold",  
          color: "#00F",  
          letterSpacing: 5,  
        }}  
      >  
        Hello World!!!  
      </Text>  
    </View>  
  );  
}
```

Dimensões

- Existem 3 formas de declarar dimensões, fixa, flex e porcentagem
- As dimensões fixas e porcentagem são aplicadas as propriedades “width” e “height”
- Em React Native, não utilizamos unidades de medida, elas são densidade de pixels por padrão (dp).
- A dimensão flex, como o nome já diz, utiliza flex.
- Na prática, utilizamos uma mescla dos 3 tipos, depende apenas do layout.
- Atenção ao utilizar medidas fixas, seu aplicativo vai rodar em diversos tamanhos de tela, medidas fixas podem prejudicar seu layout.

Dimensão Fixo

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

const FixedDimensionsBasics = () => {
  return (
    <View style={{flex:1, justifyContent:"center",
    alignItems: "center"}}>
      <View style={{
        width: 50, height: 50, backgroundColor:
        'powderblue'
      }} />
      <View style={{
        width: 100, height: 100, backgroundColor:
        'skyblue'
      }} />
      <View style={{
        width: 150, height: 150, backgroundColor:
        'steelblue'
      }} />
    </View>
  );
};

export default FixedDimensionsBasics;
```

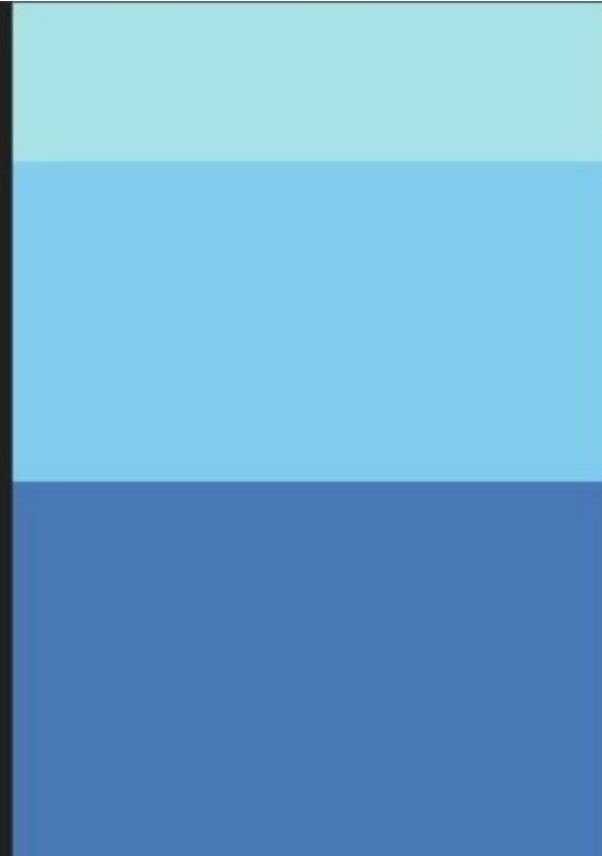


Dimensão Flex

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

const FlexDimensionsBasics = () => {
  return (
    // Try removing the 'flex: 1' on the parent View.
    // The parent will not have dimensions, so the
    // children can't expand.
    // What if you add 'height: 300' instead of 'flex:
    1'?
    <View style={{ flex: 1 }}>
      <View style={{ flex: 1, backgroundColor:
        'powderblue' }} />
      <View style={{ flex: 2, backgroundColor:
        'skyblue' }} />
      <View style={{ flex: 3, backgroundColor:
        'steelblue' }} />
    </View>
  );
};

export default FlexDimensionsBasics;
```

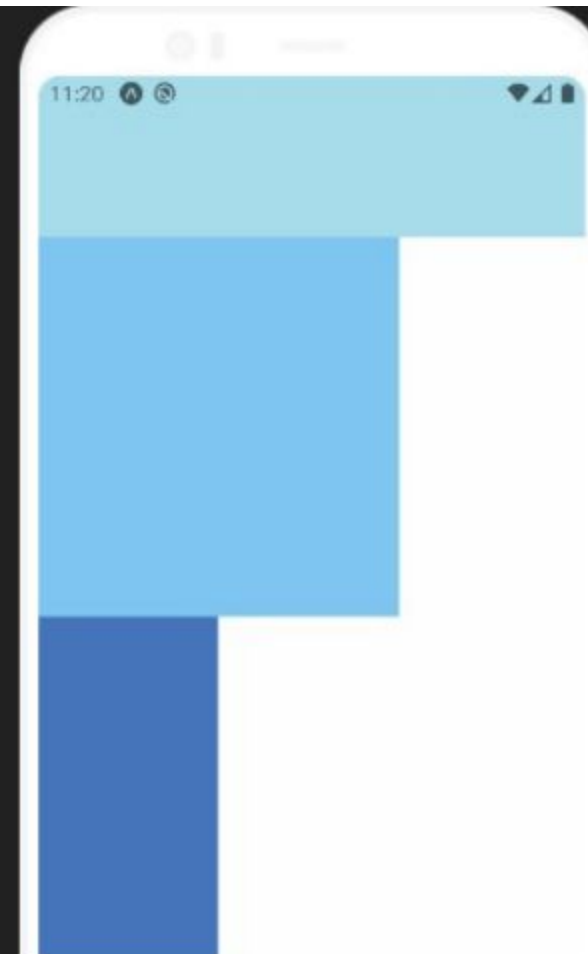


Porcentagem

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

const PercentageDimensionsBasics = () => {
  // Try removing the 'height: '100%' on the parent
  // View.
  // The parent will not have dimensions, so the
  // children can't expand.
  return (
    <View style={{ height: '100%' }}>
      <View style={{
        height: '15%', backgroundColor: 'powderblue'
      }} />
      <View style={{
        width: '66%', height: '35%', backgroundColor:
        'skyblue'
      }} />
      <View style={{
        width: '33%', height: '50%', backgroundColor:
        'steelblue'
      }} />
    </View>
  );
};

export default PercentageDimensionsBasics;
```



Layout com FlexBox

- O FlexBox foi feito para oferecer um layout consistente em diversos tamanhos de tela.
- Geralmente utilizamos uma combinação de `flexDirection`, `alignItems` e `justifyContent` para posicionar um layout responsivo.
- Flexbox funciona quase igual na web, mas com alguma exceções. Por exemplo, o `flexDirection` padrão é “column” em vez de “row” e `alignContent` é “flex-start” em vez de “stretch”

Propriedade Flex

- Flex irá definir como seus itens vão preencher o espaço disponível.
- Na imagem, temos 3 Views com flex diferentes que somam 6.
- Cada View vai ocupar $\text{flex}/6$ do espaço.

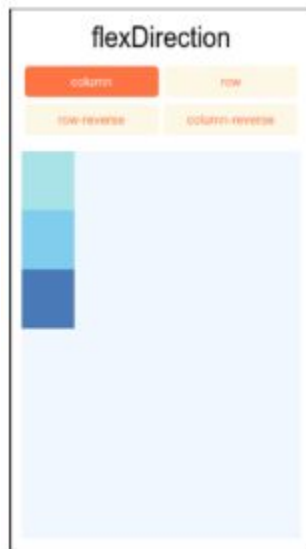
```
import React from "react";
import { StyleSheet, Text, View } from "react-native";

const Flex = () => {
  return (
    <View style={styles.container, {
      // Try setting 'flexDirection' to 'row'.
      flexDirection: "column"
    }}>
      <View style={{ flex: 1, backgroundColor: "red"
    }} />
      <View style={{ flex: 2, backgroundColor:
    "darkorange" }} />
      <View style={{ flex: 3, backgroundColor: "green"
    }} />
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 20,
  },
});
```



flexDirection




justifyContent

- justifyContent descreve como alinhar os filhos dentro do eixo principal de seu contêiner. Lembre-se, a direção do eixo principal é definido em flexDirection
- Temos os valores “flex-start”, que alinha no início do container, “flex-end” alinha no final, “center” centraliza, “space-between” que distribui os elementos uniformemente, distribuindo o espaço restante entre os filhos, “space-around” que distribui uniformemente os elementos, distribuindo o espaço restante em torno dos filhos e “space-evenly” que distribui espaços iguais.

justifyContent


justifyContent

flex-start	flex-end
center	space-between
space-around	space-evenly




justifyContent

flex-start	flex-end
center	space-between
space-around	space-evenly



justifyContent

flex-start	flex-end
center	space-between
space-around	space-evenly



justifyContent

justifyContent


flex-start	flex-end
center	space-between
space-around	space-evenly



A visual representation of the 'space-between' justifyContent property. It shows a light blue rectangular container. On the left side, there are four colored squares (teal, light blue, medium blue, and dark blue) stacked vertically. The 'space-between' property is highlighted in orange in the control panel above.

justifyContent

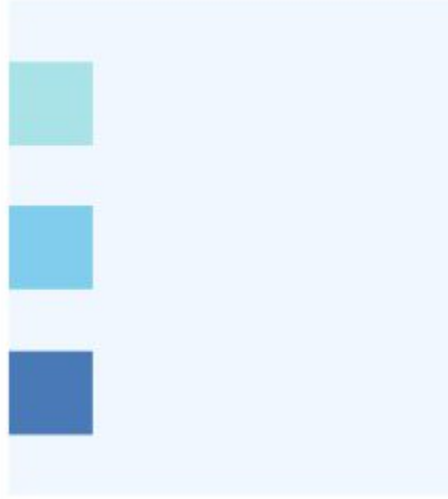
flex-start	flex-end
center	space-between
space-around	space-evenly



A visual representation of the 'space-around' justifyContent property. It shows a light blue rectangular container. On the left side, there are four colored squares (teal, light blue, medium blue, and dark blue) stacked vertically. The 'space-around' property is highlighted in orange in the control panel above.

justifyContent

flex-start	flex-end
center	space-between
space-around	space-evenly



A visual representation of the 'space-evenly' justifyContent property. It shows a light blue rectangular container. On the left side, there are four colored squares (teal, light blue, medium blue, and dark blue) stacked vertically. The 'space-evenly' property is highlighted in orange in the control panel above.

alignItems

- alignItems é muito parecido com o justifyContent, porém ele faz o alinhamento no eixo cruzado.
- Os valores mais usadas são “flex-start”, “flex-end” e “center”, que já vimos no justifyContent.
- “stretch” é valor padrão, ele preenche todo o espaço do eixo cruzado, caso não tenha um tamanho definido.


alignItems

alignItems

stretch flex-start

flex-end center

baseline

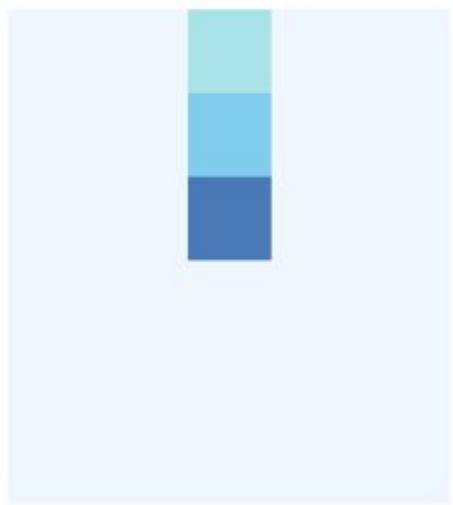


alignItems

stretch flex-start

flex-end center

baseline




alignItems

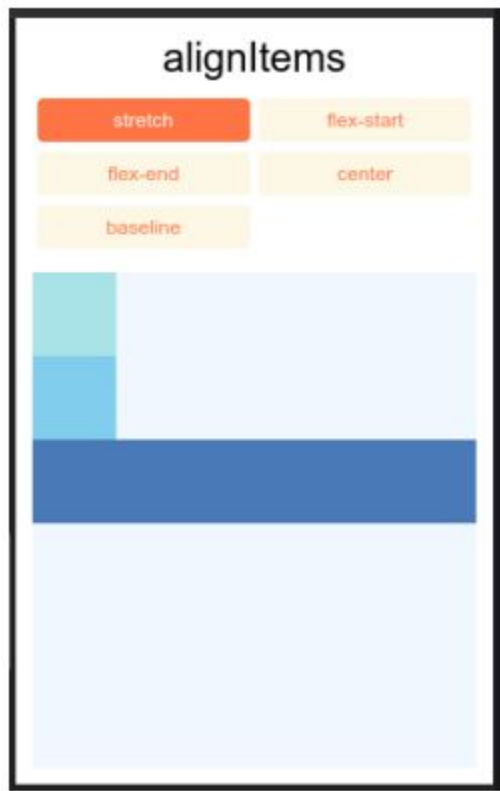
stretch flex-start

flex-end center

baseline



stretch



- Só funciona quando o elemento não tem um width configurado, ou seja, das 3 Views, apenas a última não tinha um width definido.
- Stretch é o valor padrão, então não tem necessidade de declarar nada.


alignSelf

- alignSelf tem exatamente o mesmo efeito que o alignItems, porém o comportamento é aplicado diretamente no próprio elemento, em vez de nos elementos filhos de um container.
- Com alignSelf, você consegue aplicar a propriedade em apenas um elemento, em alignItems a propriedade é aplicada em todos os elementos juntos, de uma vez só.

alignSelf

alignSelf

stretch	flex-start
flex-end	center
baseline	



The diagram shows a light blue rectangular container. On the left side, there is a vertical stack of three colored squares: a teal square at the top, a light blue square in the middle, and a dark blue square at the bottom. The squares are aligned to the top-left corner of the container, demonstrating the 'flex-start' alignment.

alignSelf

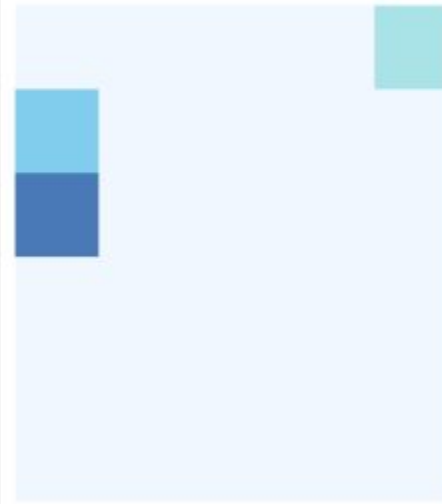
stretch	flex-start
flex-end	center
baseline	



The diagram shows a light blue rectangular container. On the left side, there is a vertical stack of three colored squares: a teal square at the top, a light blue square in the middle, and a dark blue square at the bottom. The squares are aligned to the horizontal center of the container, demonstrating the 'center' alignment.

alignSelf

stretch	flex-start
flex-end	center
baseline	

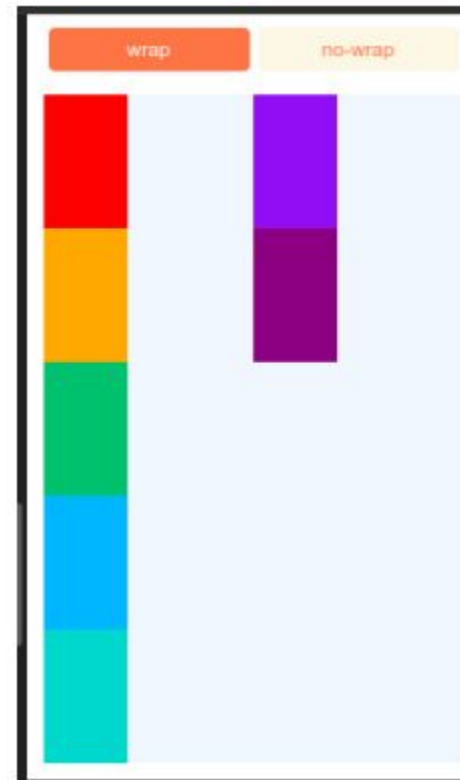
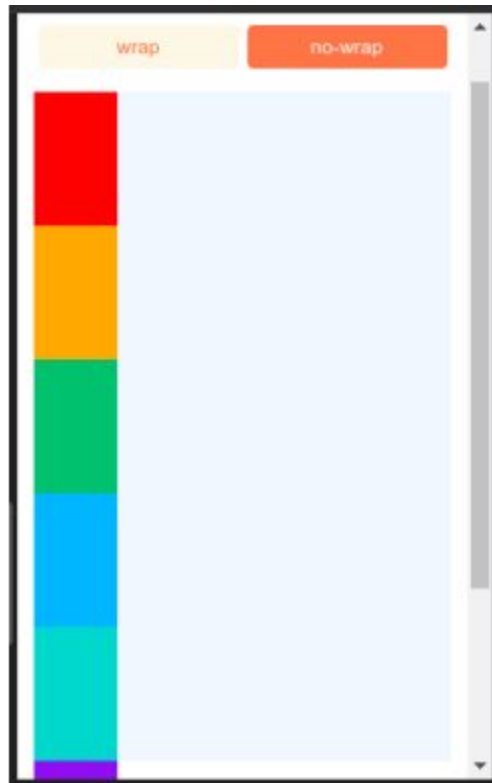


The diagram shows a light blue rectangular container. On the left side, there is a vertical stack of three colored squares: a teal square at the top, a light blue square in the middle, and a dark blue square at the bottom. The squares are aligned to the bottom-right corner of the container, demonstrating the 'flex-end' alignment.

flexWrap

- É uma propriedade aplicada aos containers e que definem o que vai acontecer quando o tamanho dos elementos filhos excederem o tamanho do eixo principal.
- Por padrão, os filhos são forçados a uma única linha, o que pode alterar suas medidas.
- flexWrap aceita “wrap” e “no-wrap”, que é o valor padrão

flexWrap




alignContent

- Quando flexWrap está “wrap”, podemos controlar como a exibição é feita.
- Aceita os mesmo valores que o justifyContent, exceto “space-evenly”

alignContent

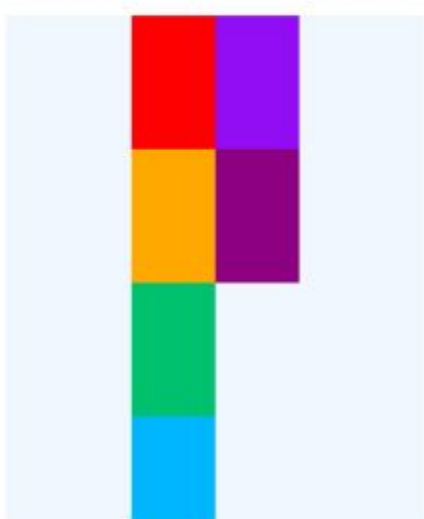
alignContent

flex-start	flex-end
stretch	center
space-between	space-around




alignContent

flex-start	flex-end
stretch	center
space-between	space-around

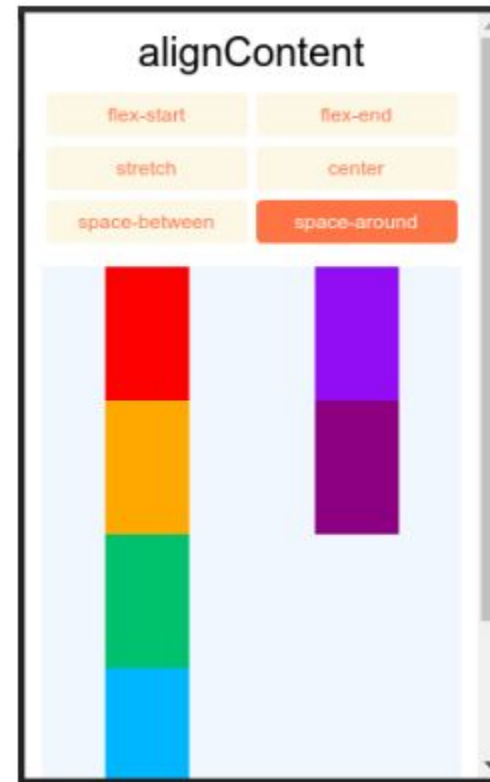
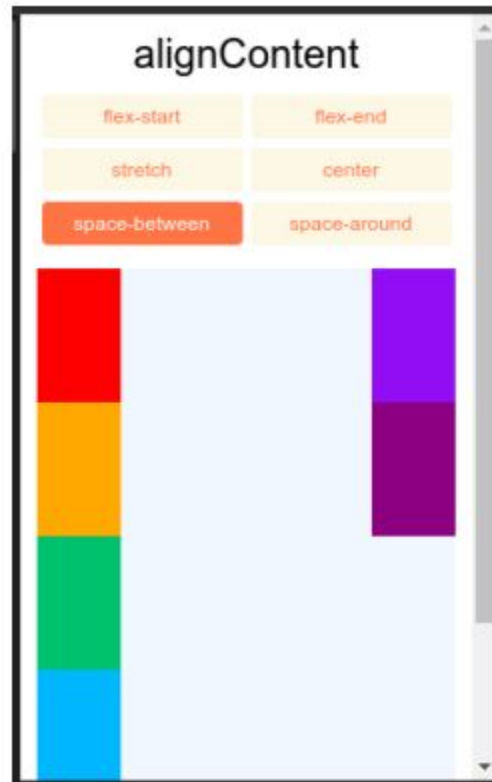


alignContent

flex-start	flex-end
stretch	center
space-between	space-around



alignContent



Aprenda Mais

- Vimos aqui a base para criação de qualquer layout, porém existem muito mais detalhes e que podem ser encontrados na documentação oficial.
- Links para consulta:
- <https://reactnative.dev/docs/style>
- <https://reactnative.dev/docs/height-and-width>
- <https://reactnative.dev/docs/flexbox>

Aprenda Mais Jogando



<https://flexboxfroggy.com/>

FLEXBOX FROGGY

Level 1 of 24

Welcome to Flexbox Froggy, a game where you help Froggy and friends by writing CSS code! Guide this frog to the lilypad on the right by using the **justify-content** property, which aligns items horizontally and accepts the following values:

- **flex-start**: Items align to the left side of the container.
- **flex-end**: Items align to the right side of the container.
- **center**: Items align at the center of the container.
- **space-between**: Items display with equal spacing between them.
- **space-around**: Items display with equal spacing around them.

For example, **justify-content: flex-end;** will move the frog to the right.

```
1 #pond {  
2   display: flex;  
3   justify-content: flex-end  
4 }  
5  
6  
7  
8  
9  
10
```

Next

SPONSOR Vultr — Save up to 90% on bandwidth when you make the switch to Vultr and leave Big Tech baggage behind.

Flexbox Froggy is created by [Codepip](#) • [YouTube](#) • [Twitter](#) • [GitHub](#) • [Settings](#)

Want to learn CSS grid? Play [Grid Garden](#).





Obrigado(a)!