53 45 52 45 49 20 46 49 45 4c 20
41 4f 53 20 50 52 45 43 45 49 54
4f 53 20 44 41 20 48 4f 4e 52 41
20 45 20 44 41 20 43 49 c3 8a 4e
43 49 41 2c 20 50 52 4f 4d 4f 56
45 4e 44 4f 20 4f 20 55 53 4f 20
45 20 4f 20 44 45 53 45 4e 56 4f
4c 56 49 4d 45 4e 54 4f 20 44 41
20 49 4e 46 4f 52 4d c3 81 54 49
43 41 20 45 4d 20 42 45 4e 45 46
c3 8d 43 49 4f 20 44 4f 20 43 49
44 41 44 c3 83 4f 20 45 20 44 41
20 53 4f 43 49 45 44 41 44 45 2e

RESIDÊNCIA DE

SOFTWARE

CAPACITAR
TREINAR
EMPREGAR
TRANSFORMAR

Firjan SENAI

serratec
Parque Tecnológico Região Serrana

FrontEnd Mobile
Conceitos de Layout com FlexBox

Objetivos

01 O que é style e StyleSheet

02 Entender os tipos de Dimensões

03 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 }})

StyleSheet


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

export default function App() {
  return (
    <View style={styles.container}>
      <Text>Estilo via StyleSheet</Text>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#fff',
    alignItems: 'center',
    justifyContent: 'center',
  },
});
```

Inline x StyleSheet

- Observem a diferença de legibilidade do código quando extraímos os estilos inline.

- É lógico que o “css” não sumiu, ele apenas está em um lugar que faz mais sentido.

- Em aplicativos grandes, com milhares de linhas de código, deixar o código de estilo separado terá impacto significativo no desempenho da equipe.

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

```
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

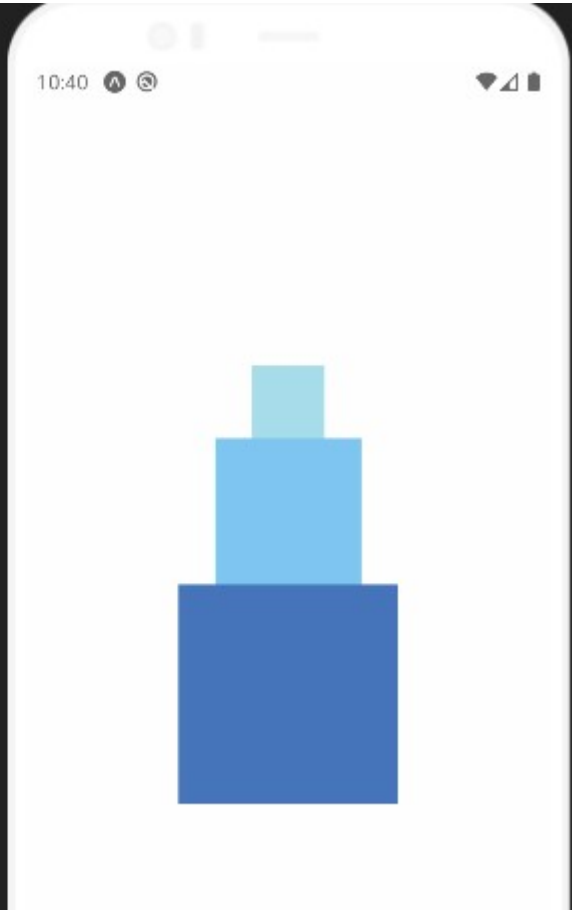
- Existem 3 formas de declarar dimensões, fixa, flex e porcentagem
- As dimensões fixas e porcentagem são aplicadas as propriedades “width” e “height”
- Em React Native, não utilizamos unidades de medida, elas são densidade de pixels por padrão (dp).
- A dimensão flex, como o nome já diz, utiliza flex.
- 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 Fixa

```
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;
```

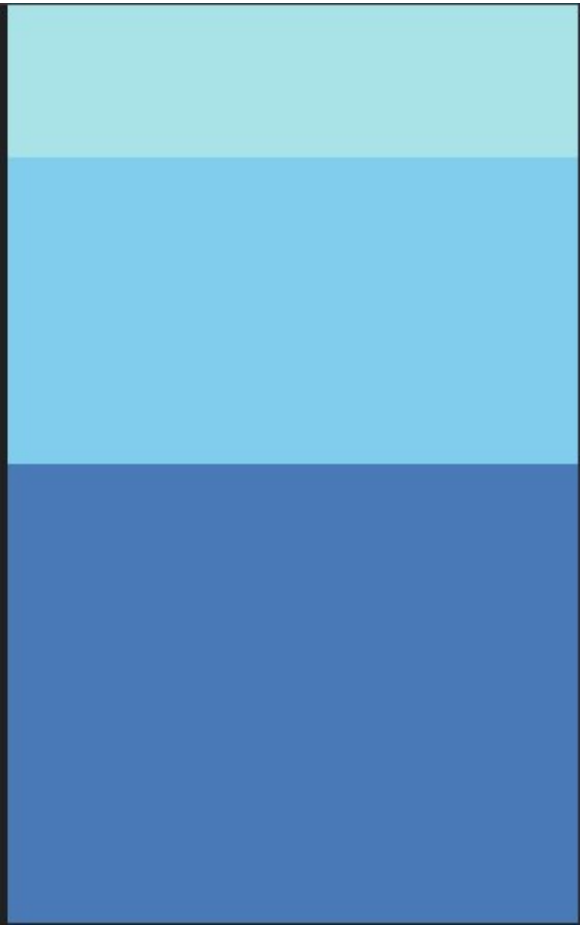


Dimensão Flex


```jsx
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;
```

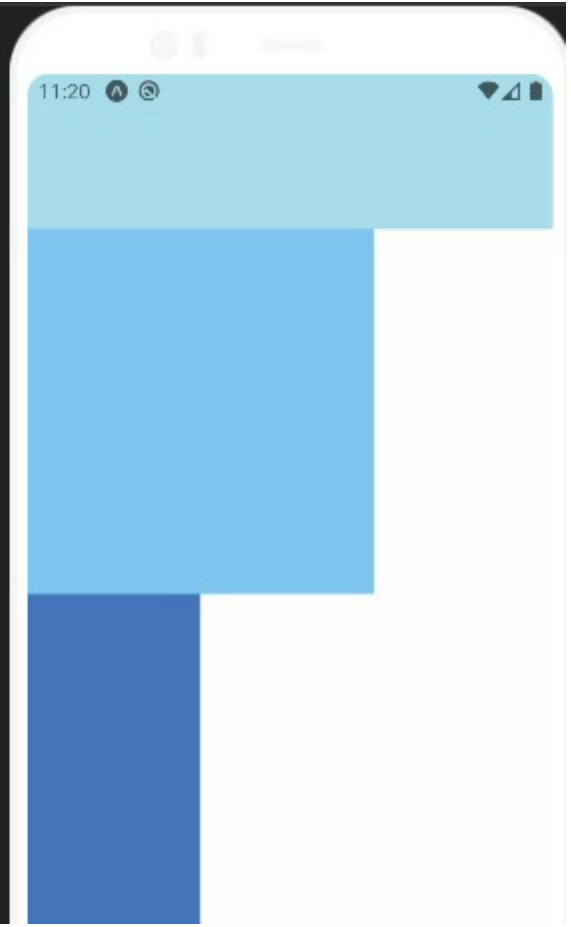


Porcentagem


```jsx
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

Firjan SENAI | RESIDÊNCIA DE SOFTWARE | serratec Parque Tecnológico Região Serrana

- 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 excessões. Por exemplo, o flexDirection padrão é “column” em vez de “row” e alignContent é “flex-start” em vez de “stretch”

Propriedade flex

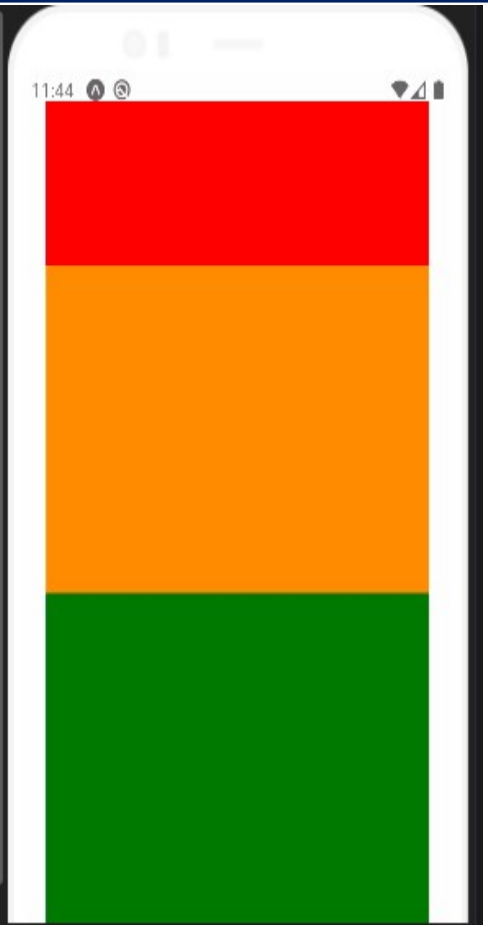
- Flex irá definir como seus itens vão preencher o espaço disponível.
- Na imagem, temos 3 Views com flex diferentes que somam 6. Cada View vai ocupar flex/6 do espaço.
- Flexbox funciona quase igual na web, mas com alguma excessões. Por exemplo, o flexDirection padrão é “column” em vez de “row” e alignContent é “flex-start” em vez de “stretch”

```
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,
  },
});

export default Flex;
```



FlexDirection

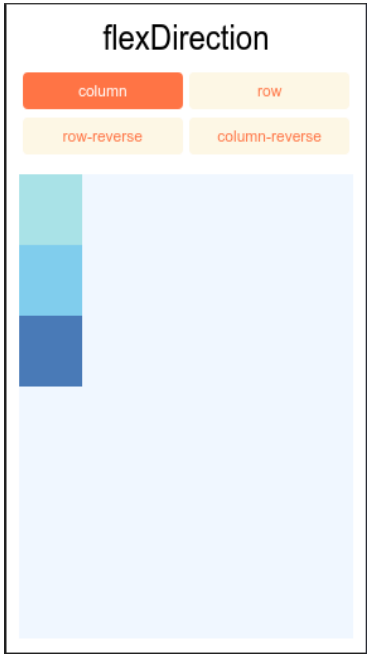
- O FlexBox foi feito para oferecer um layout consistente em diversos tamanhos de tela. Isso também é conhecido como eixo principal. O eixo cruzado é o eixo perpendicular ao eixo principal

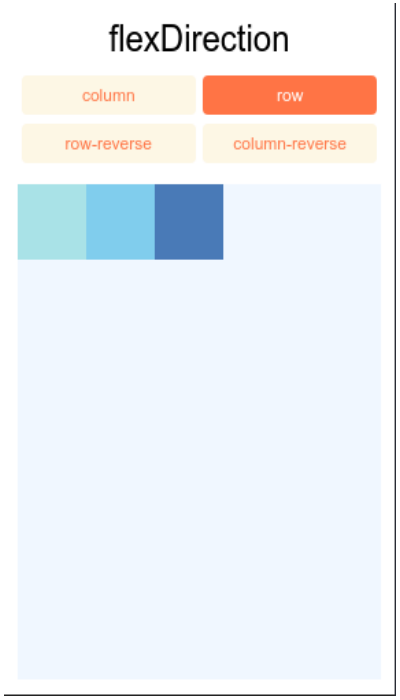
- Os elementos podem ser dispostos em coluna (“column”), em coluna invertida (“column-reverse”), em linha (“row”) e em linha invertida (row-reverse)

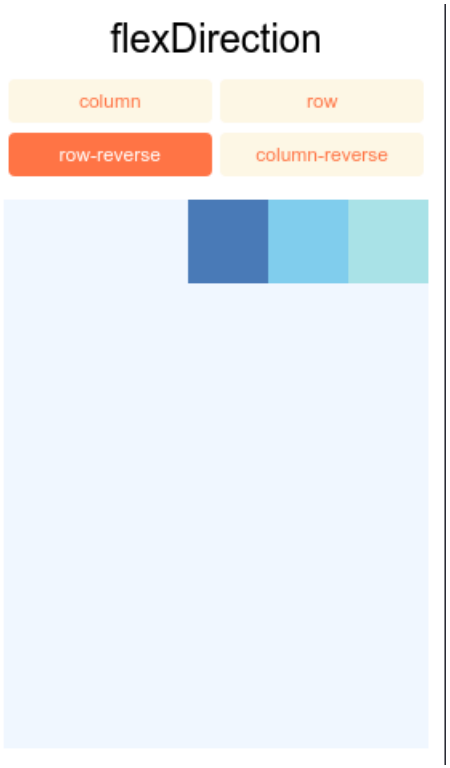
- Flexbox funciona quase igual na web, mas com alguma excessões. Por exemplo, o flexDirection padrão é “column” em vez de “row” e alignContent é “flex-start” em vez de “stretch”

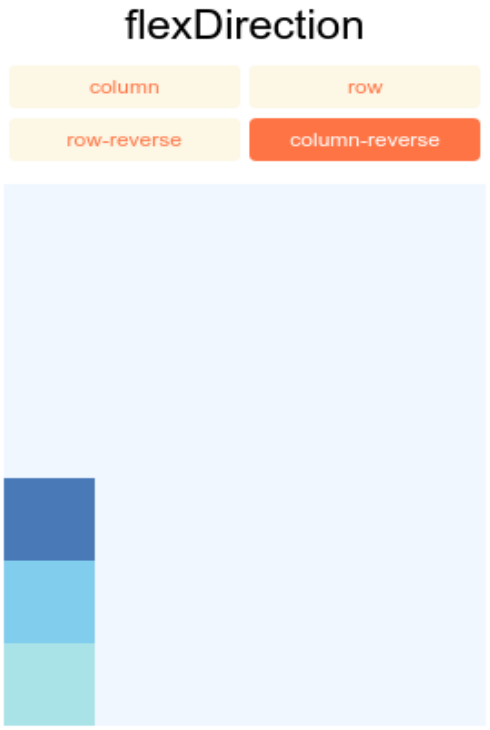
flexDirection

Firjan SENAI | RESIDÊNCIA DE SOFTWARE | serratec Parque Tecnológico Região Serrana









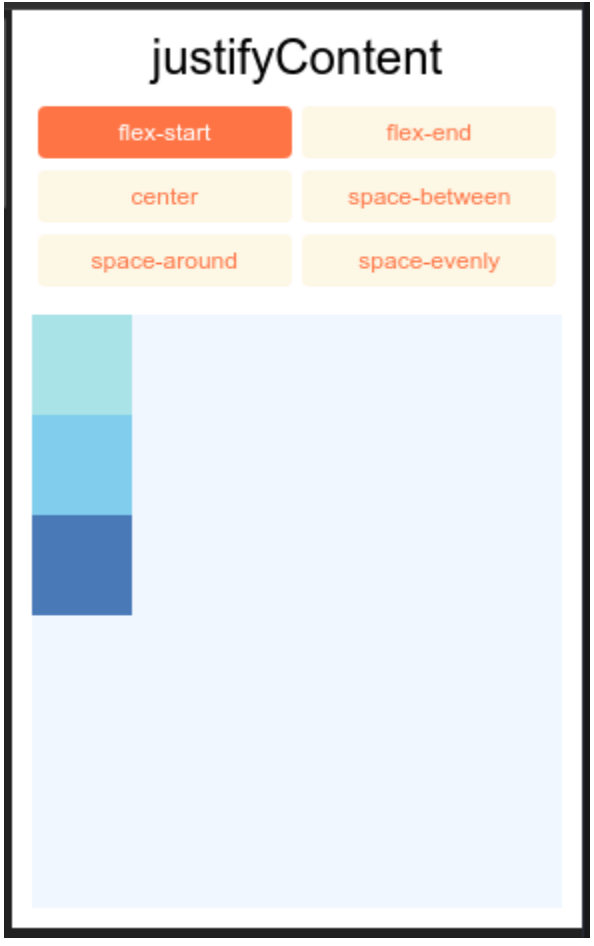
justifyContent

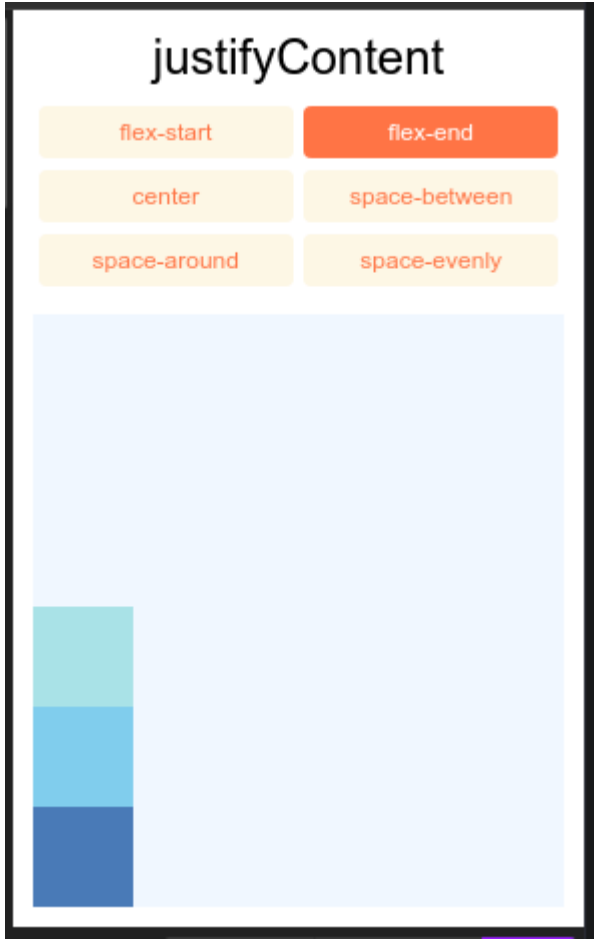
- justifyContent descreve como alinhar os filhos dentro do eixo principal de seu contêiner. Lembre-se, a direção do eixo principal é defino em flexDirection

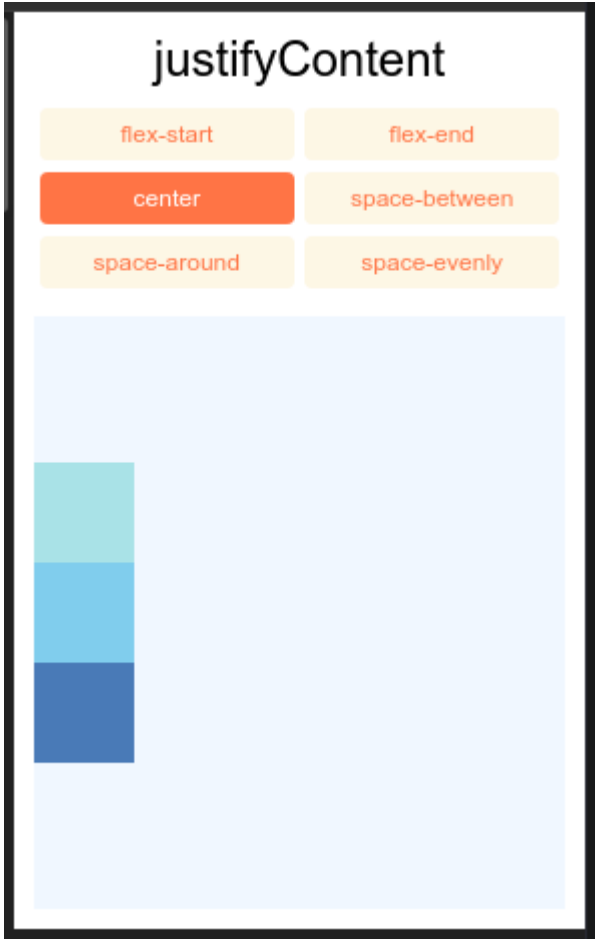
- Temos os valores “flex-start”, que alinha no início do container, “flex-end” alinha no final, “center” centraliza, “space-between” que distribui os elementos uniformimente, distribuindo o espaço restante entre os filhos, “space-around” que distribui uniformemente os elementos, distribuindo o espaço restante em torno dos filhos e “space-evently” que distribui espaços iguais.

Start, End e Center

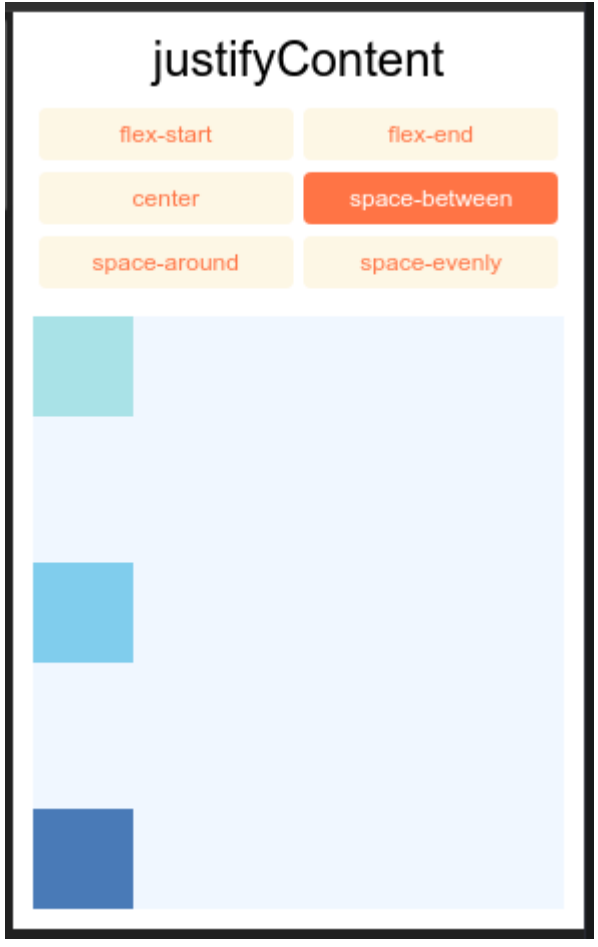
Firjan SENAI | RESIDÊNCIA DE SOFTWARE | serratec Parque Tecnológico Região Serrana

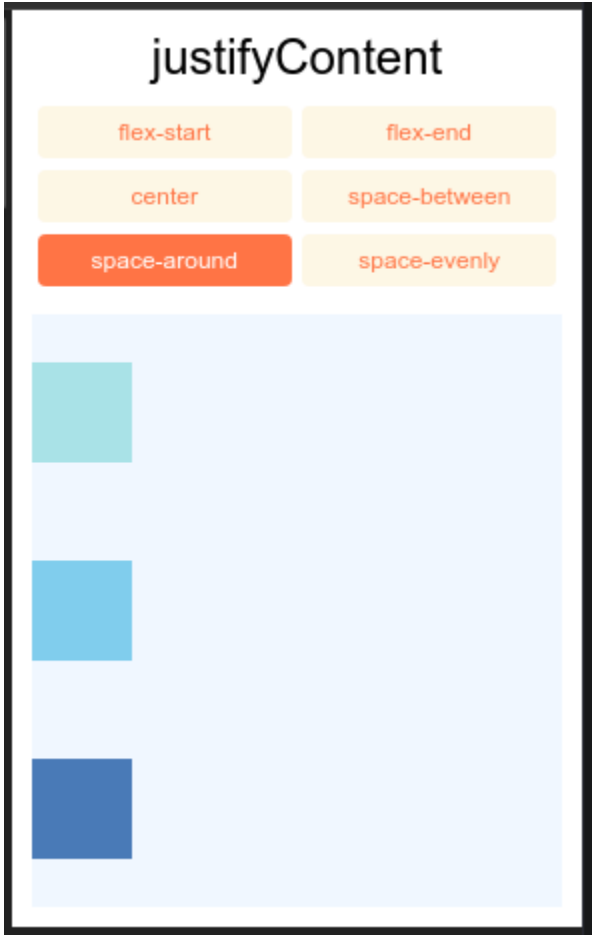


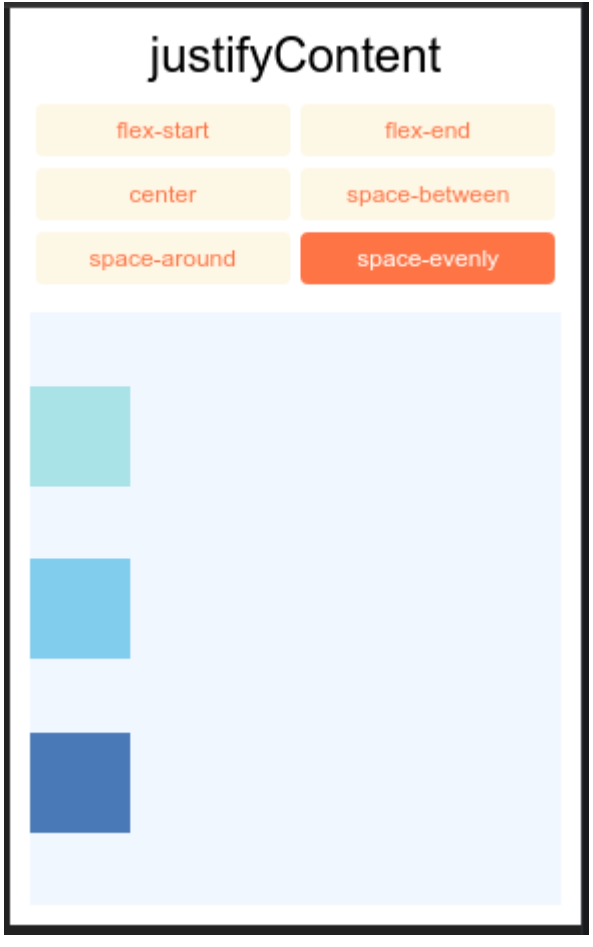




Spaces


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

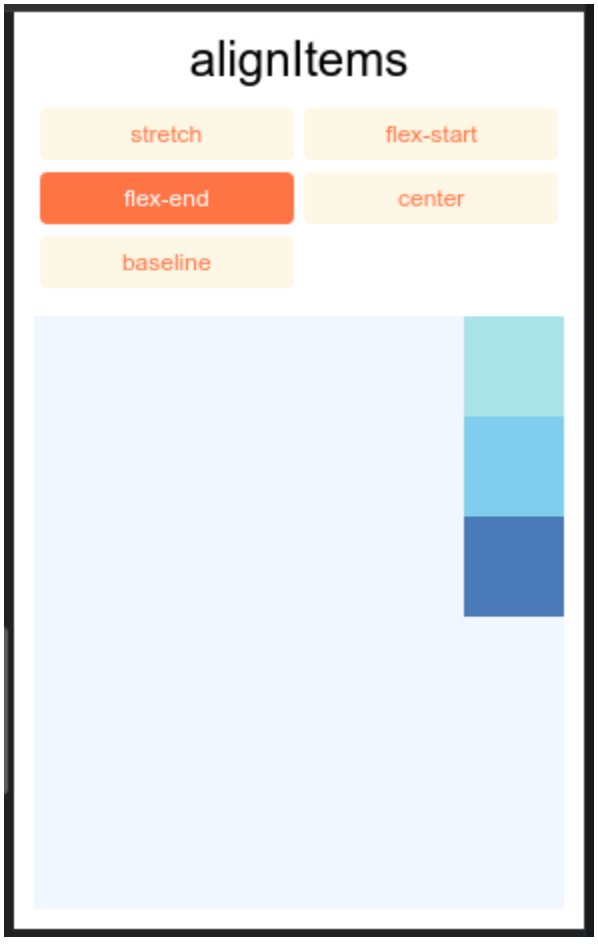
alignItems

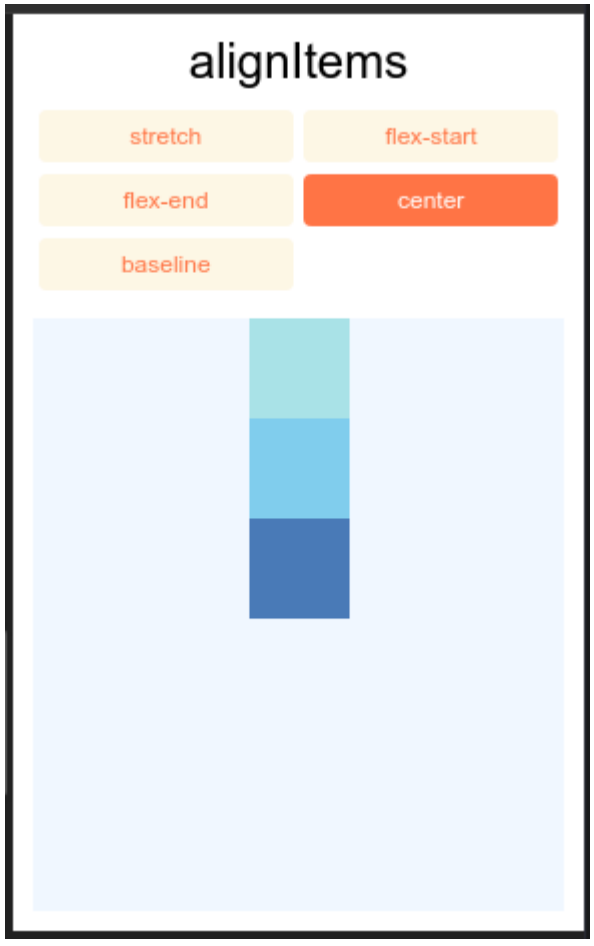
- AlignItems é muito parecido com o justifyContent, porém ele faz o alinhamento no eixo cruzado.

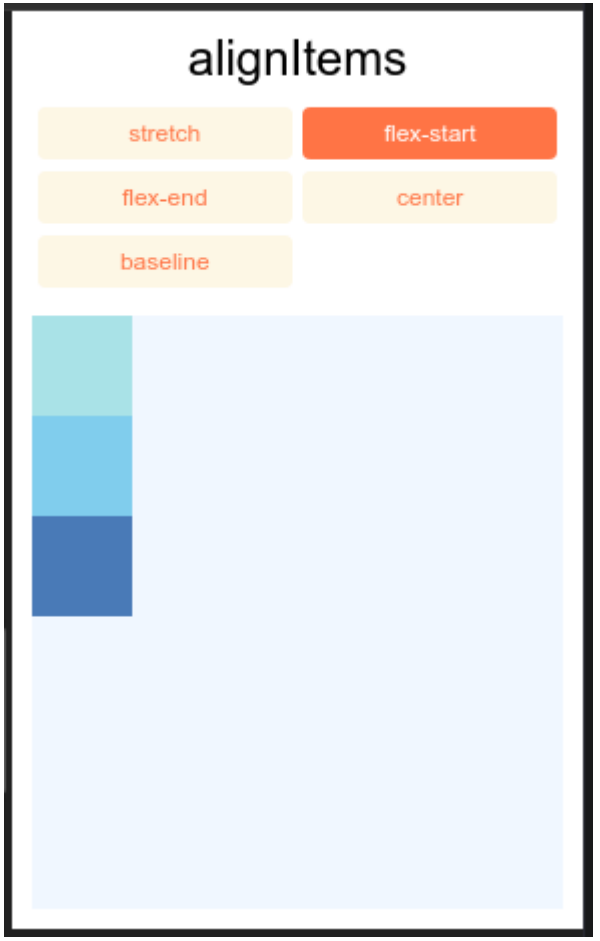
- Os valores mais usadas são “flex-start”, “flex-end” e “center”, que já vimos no justifyContent, temos os valores “stretch” e o “baseline”.

- “strech” é valor padrão, ele preenche todo o height do eixo cruzado.

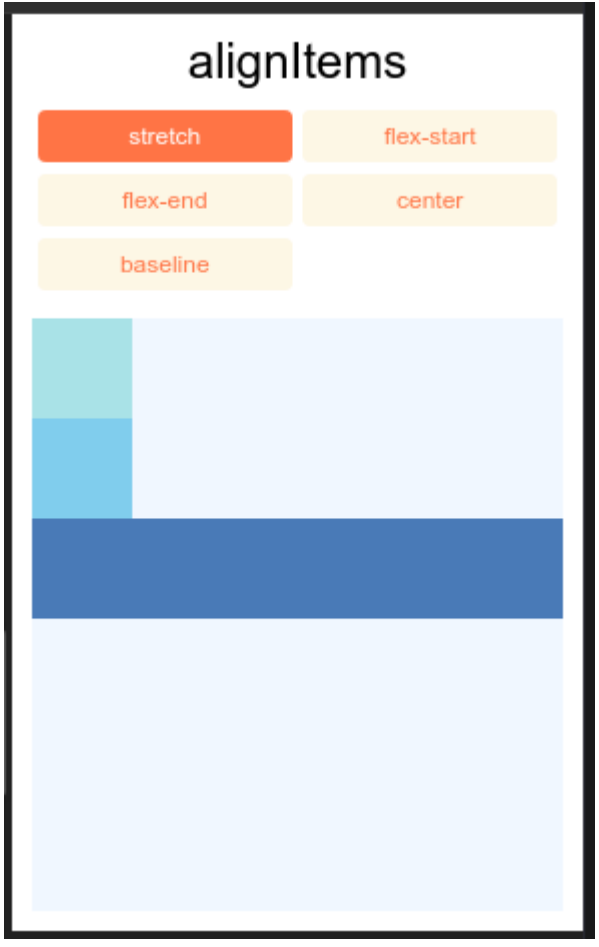
Start, center e end







Stretch



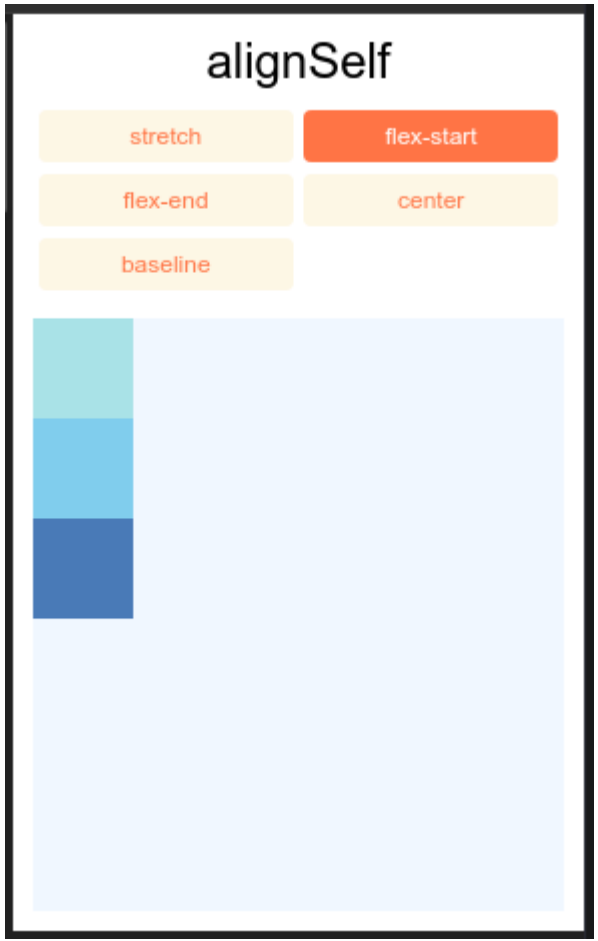
- Só funciona quando o elemento não tem um width configurado, ou seja, das 3 Views, apenas a última não tinha um width definido.
- Stretch é o valor padrão, então não tem necessidade de declarar nada.

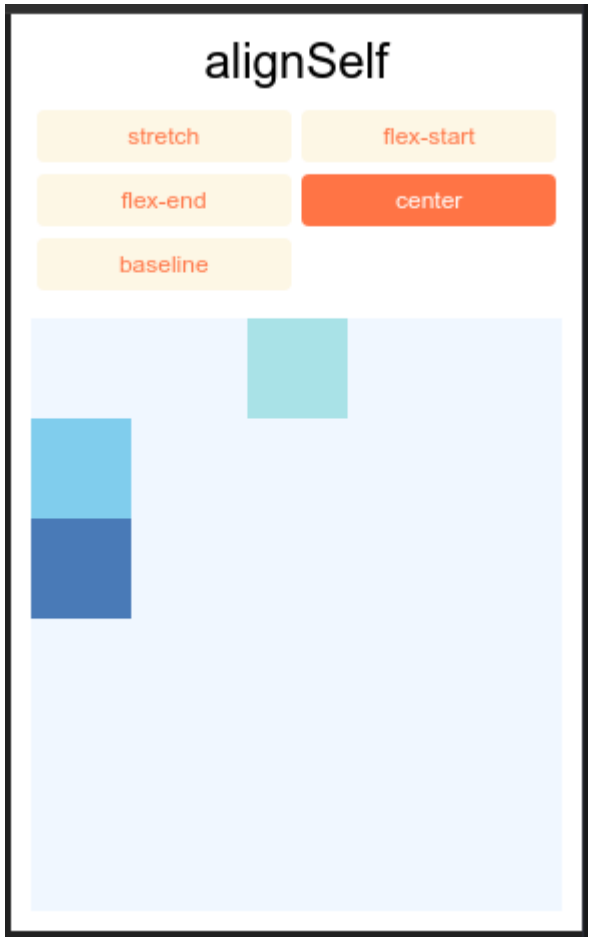
alignSelf

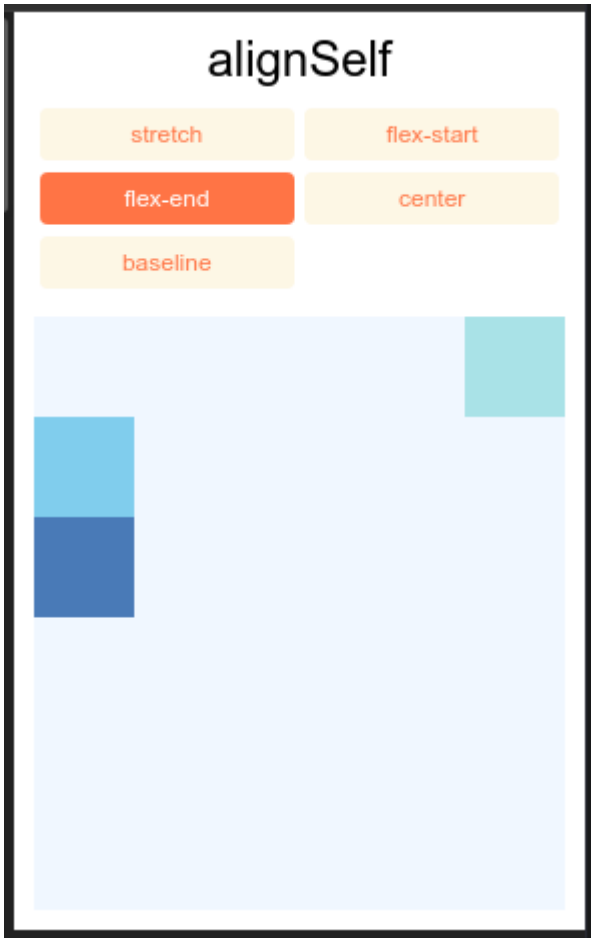
- alignSelf tem o exatamente o mesmo efeito que o alignItems, porém o comportamento é aplicado diretamente no próprio elemento, em vez de nos elementos filhos de um container.

- Com alignSelf, você consegue aplicar a propriedade em apenas um elemento, em alignItems a proprieda é aplicada em todos os elementos juntos, de uma vez só.

alignSelf


alignSelf
stretch
flex-start
flex-end
center
baseline


alignSelf
stretch
flex-start
flex-end
center
baseline

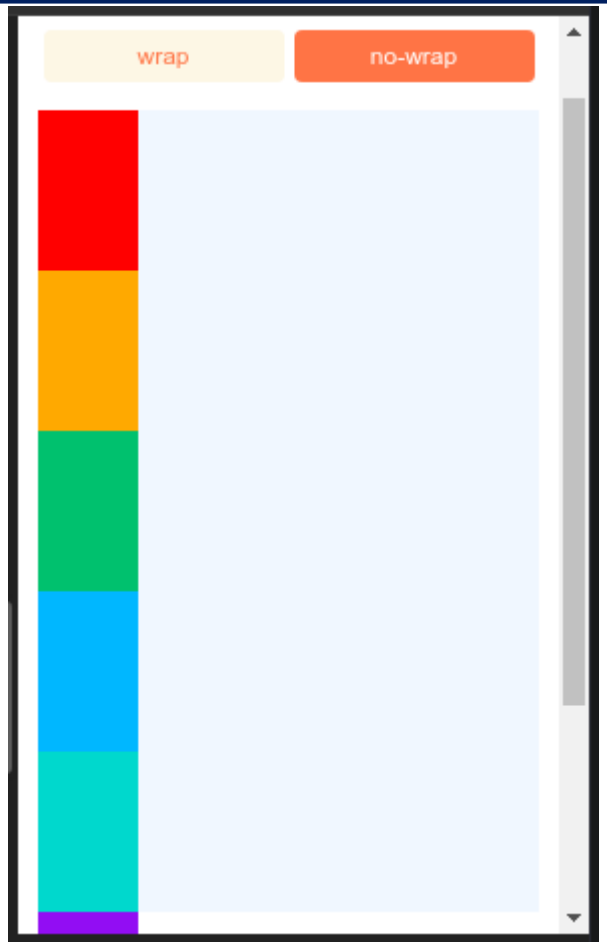

alignSelf
stretch
flex-start
flex-end
center
baseline

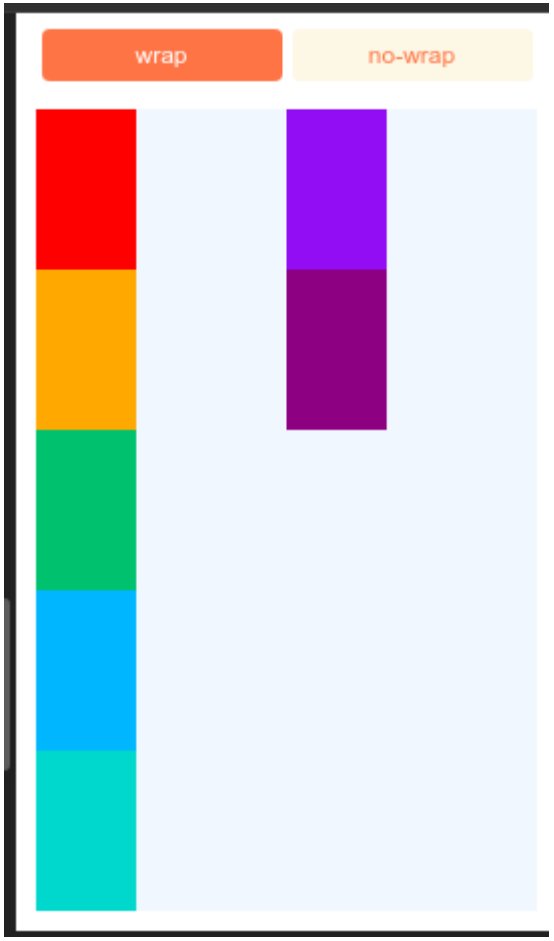
- É uma propriedade aplicada aos containers e que definem o que vai acontecer quando o tamanho dos elementos filhos excederem o tamanho do eixo principal.

- 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=”wrap”



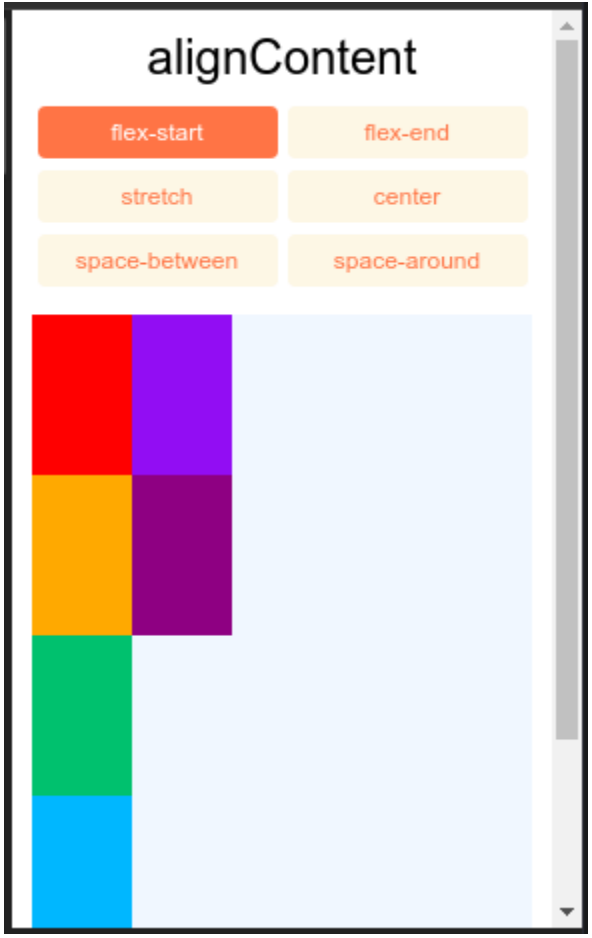


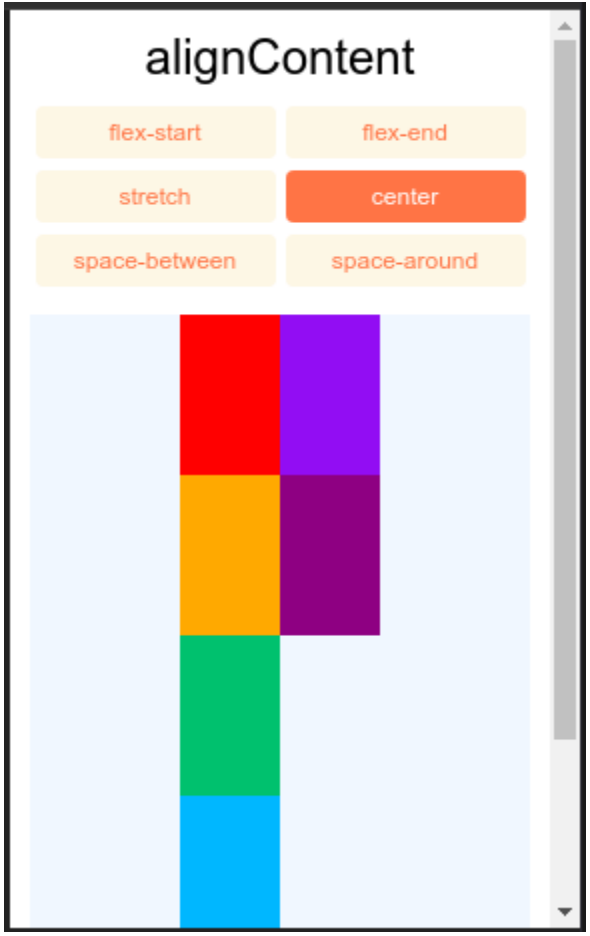
Align Content

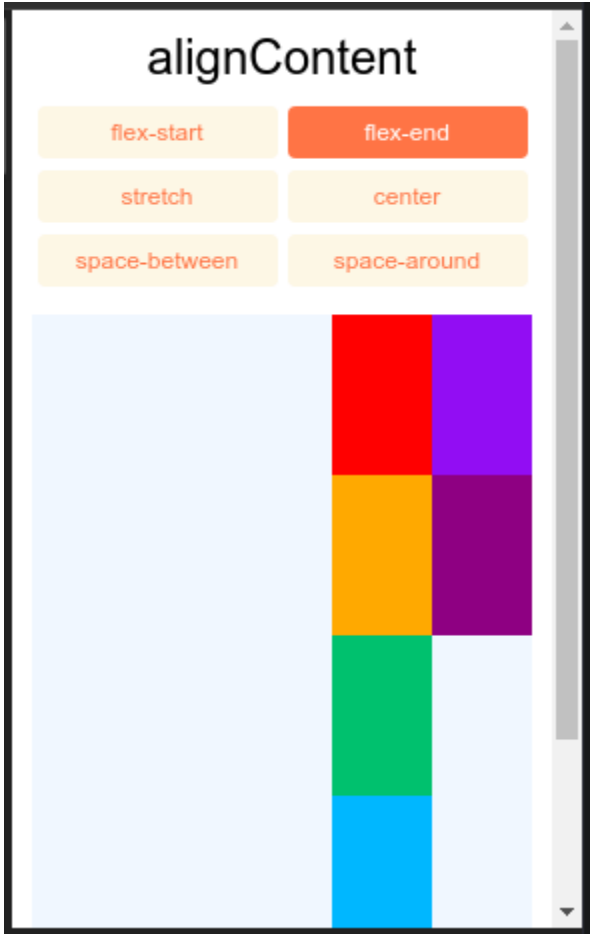
- Quando flexWrap está “wrap”, podemos controlar como a exibição é feita.

- Aceita os mesmo valores que o justifyContent, exceto “space-evently”

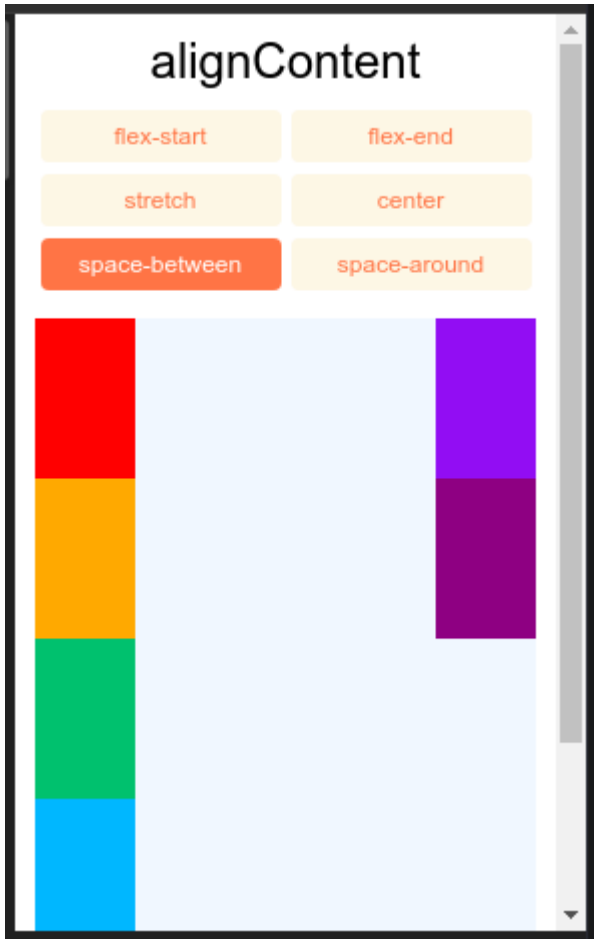
Align Content

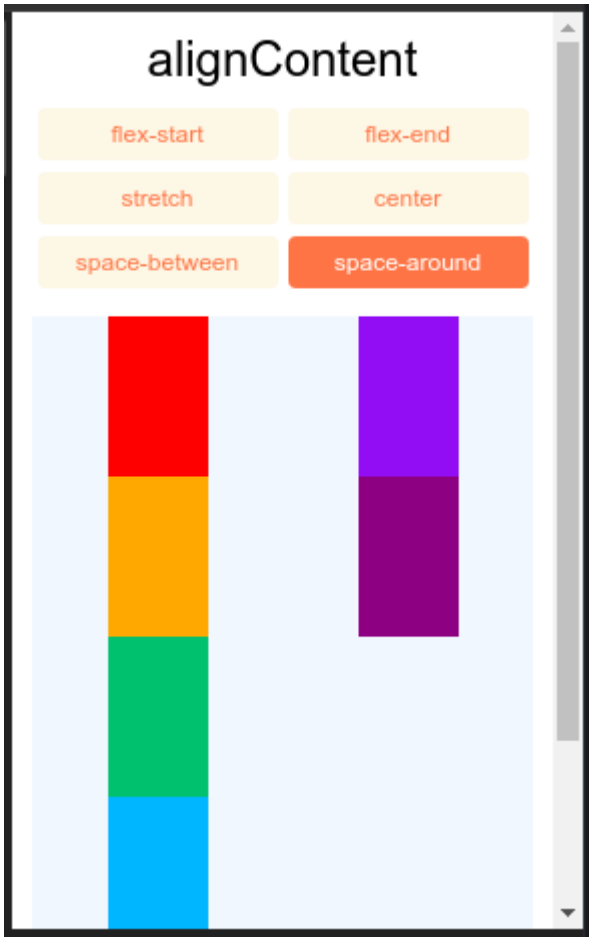






Align Content





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

- Pratique flexBox jogando =)

http://flexboxfroggy.com/