Desenvolvimento para Dispositivos Móveis

Aula 09 - Estilo (Parte 2)

Material Didático do Instituto Metrópole Digital - IMD

Apresentação

Nesta aula, continuaremos o assunto de estilização. Em especial, trataremos de um conhecimento muito importante introduzido na aula passada: o *flex*.

Objetivos

- Conhecer como trabalhar com direcionamento de elementos utilizando *flex*
- Conhecer como justificar o conteúdo utilizando *flex*
- Conhecer como trabalhar com alinhamento de itens

Flex e Flex Direction

Link do video da aula: https://youtu.be/ukXa8XiSFIU

Nesta aula, trabalharemos, com um exemplo prático, o uso do *Flex*, mais especificamente *Flex Direction*.

Alinhamento de elementos

O *Flex Direction* controla como serão dispostos na tela os elementos que possuem tal propriedade. Para essa propriedade podem ser atribuídos os seguintes valores:

- ***colunm:*** é o valor padrão e vai organizar os elementos de cima para baixo.
- ***row:*** alinha os elementos da esquerda para a direita
- ***colunm-reverse:*** é semelhante ao *colunm*, mas organiza os elementos de forma reversa. Ou seja, os elementos serão organizados de baixo para cima.
- ***row-reverse:*** semelhante ao *row*, porém, alinha os elementos da direita para a esquerda.

Prática de *Flex Direction*

Acompanhando a aula, teste essa funcionalidade implementando o código abaixo:

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

export default function App() {
  return (
    <SafeAreaView style={styles.container}>
      <View style={[styles.box, styles.cor1]}><Text style={{ color:
'white', fontSize: 30 }}>1</Text></View>
      <View style={[styles.box, styles.cor2]}><Text style={{ color:
```

```
'white', fontSize: 30 }}>2</Text></View>
      <View style={[styles.box, styles.cor3]}><Text style={{ color:
'white', fontSize: 30 }}>3</Text></View>
    </SafeAreaView>
  );
}

const styles = StyleSheet.create({

  container: {
    flex: 1,
    flexDirection: 'row-reverse'
  },
  box: {
    width: 100,
    height: 100
  },
  cor1: {
    backgroundColor: '#00F'
  },
  cor2: {
    backgroundColor: '#05F'
  },
  cor3: {
    backgroundColor: '#0AF'
  },

})
```

Layout Direction

Link do video da aula: https://youtu.be/s5gPtOFiDhQ

Nesta aula, veremos mais uma funcionalidade adquirida com o uso do *Flex*: o *Layout Direction*.

Direção do *Layout*

O layout possui uma borda *start*, que representa o início e uma referência *end*, que representa o fim. Por padrão, o layout inicia com *start* na borda esquerda e o *end* na borda direita, isto é, LTR *(Left to Right)*.

Existem duas possibilidades de direção:

- ***LTR*** (valor padrão): os elementos são dispostos da esquerda para a direita.
- ***RTL***: os elementos são dispostos da direita para a esquerda.

Acompanhando a aula, adicione em seu objeto de estilização a direção desejada. Veja o exemplo abaixo:

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

export default function App() {
  return (
    <SafeAreaView style={styles.container}>
      <View style={[styles.box, styles.cor1]}><Text style={{ color:
'white', fontSize: 30 }}>1</Text></View>
      <View style={[styles.box, styles.cor2]}><Text style={{ color:
'white', fontSize: 30 }}>2</Text></View>
      <View style={[styles.box, styles.cor3]}><Text style={{ color:
'white', fontSize: 30 }}>3</Text></View>
    </SafeAreaView>
  );
}

const styles = StyleSheet.create({

  container: {
    flex: 1,
    flexDirection: 'column',
    direction: 'rtl'
  },
  box: {
    width: 100,
    height: 100
  },
  cor1: {
    backgroundColor: '#00F'
  },
  cor2: {
    backgroundColor: '#05F'
  },
  cor3: {
    backgroundColor: '#0AF'
  },
```

```
})
```

Align items

Link do video da aula: https://youtu.be/DUDPbEYgOsg

Nesta aula veremos outras duas importantes propriedades do *flex*: *JustifyContent* e *AlignItems*

Justificando o conteúdo

Essa propriedade diz respeito a como os elementos serão distribuídos dentro da direção principal. Existem algumas formas de distribuição, são elas:

- ***flex-start*** (versão padrão): aAlinha os elementos do *container* a partir do início do eixo principal do *container*.
- ***flex-end***: alinha os elemento do *container* a partir do final do eixo principal do *container*.
- ***center:*** alinha os elementos do *container* no centro do eixo principal do *container*.
- ***space-between:*** distribui de maneira que o primeiro elemento fica na borda inicial do eixo principal, o último elemento fica na borda final e o restante dos itens são distribuídos de maneira uniforme no eixo principal.
- ***space-around:*** semelhante ao *space-between*, porém, vai existir uma margem no início e no final do eixo principal, sendo o restante dos elementos distribuídos de maneira uniforme. Essas margens no início e no fim do eixo têm metade do tamanho do espaço entre dois elementos do *container*.
- ***space-evenly:*** distribui os elementos de maneira uniforme no eixo principal, sendo exatamente iguais tanto os espaços entre os itens, quanto os espaços nas margens do início e fim do eixo principal do *container*.

Alinhamento dos itens

O *AlignItems* é semelhante ao *JustifyContent*, a diferença é que ao invés de aplicar-se ao eixo principal, se aplica ao eixo secundário.

As formas de alinhamento estão descritas abaixo:

- ***flex-start***: alinha os elementos a partir do início do eixo secundário do *container*.
- ***flex-end***: alinha os elementos a partir do final do eixo secundário do *container*.
- ***center***: alinha os elementos no centro do eixo secundário do *container*.

Acompanhando a aula, teste essas funcionalidades modificando o objeto de estilização abaixo:

```
const styles = StyleSheet.create({

  container: {
    flex: 1,
    flexDirection: 'column',
    direction: 'ltr',
    justifyContent: 'center',
    alignItems: 'center'
  },
  box: {
    width: 100,
    height: 100
  },
  cor1: {
    backgroundColor: '#00F'
  },
  cor2: {
    backgroundColor: '#05F'
  },
  cor3: {
    backgroundColor: '#0AF'
  },

})
```

Wrap e Align Contents

Link do video da aula: https://youtu.be/43OdEQOf0KU

Dando sequência ao conteúdo do *flex*, vamos ver uma propriedade útil para quando os elementos excederem o tamanho do *container* principal.

Wrap

Por padrão, os elementos são forçados a ficar em uma única linha (ou coluna). Caso os elementos excedam o tamanho do *container* principal, por padrão, não há quebra de linha (ou coluna) e o usuário não poderá visualizar o conteúdo.

Para resolver essa problemática, temos a propriedade *Wrap*, que pode assumir dois valores. São eles:

- ***nowrap (valor padrão)***: não há quebra de linha (ou coluna) e os elementos excedentes não poderão ser visualizados.
- ***wrap***: há quebra de linha (ou coluna) e os elementos que excederem o *container* principal serão postos na linha (ou coluna) seguinte.

Align Contents

Dado que houve uma quebra de linha (ou coluna), a propriedade *Align Contents* especifíca como isso será disposto na tela. Os valores possíveis são:

- ***flex-start (valor padrão)***: Alinha as linhas (ou colunas) quebradas no início do eixo principal do *container*.
- ***flex-end***: alinha as linhas (ou colunas) quebradas no final do eixo principal do *container*.
- ***center***: alinha as linhas (ou colunas) quebradas no centro do eixo principal do *container*.
- ***space-between*** : distribui as linhas (ou colunas) quebradas de maneira uniforme no eixo principal, sem deixar margem entre a borda e as linhas/colunas iniciais e finais.
- ***space-around***: distribui as linhas (ou colunas) quebradas de maneira uniforme no eixo principal, deixando uma margem entre a borda, a primeira e última linha/coluna.

Acompanhe a aula e ao final seu código deve ficar como se vê abaixo:

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

export default function App() {
  return (
    <SafeAreaView style={styles.container}>
      <View style={[styles.box, styles.cor1]}><Text style={{ color:
'white', fontSize: 30 }}>1</Text></View>
      <View style={[styles.box, styles.cor2]}><Text style={{ color:
'white', fontSize: 30 }}>2</Text></View>
      <View style={[styles.box, styles.cor3]}><Text style={{ color:
'white', fontSize: 30 }}>3</Text></View>
      <View style={[styles.box, styles.cor1]}><Text style={{ color:
'white', fontSize: 30 }}>1</Text></View>
      <View style={[styles.box, styles.cor2]}><Text style={{ color:
'white', fontSize: 30 }}>2</Text></View>
      <View style={[styles.box, styles.cor3]}><Text style={{ color:
'white', fontSize: 30 }}>3</Text></View>
      <View style={[styles.box, styles.cor1]}><Text style={{ color:
'white', fontSize: 30 }}>1</Text></View>
```

```
      <View style={[styles.box, styles.cor2]}><Text style={{ color:
'white', fontSize: 30 }}>2</Text></View>
      <View style={[styles.box, styles.cor3]}><Text style={{ color:
'white', fontSize: 30 }}>3</Text></View>
    </SafeAreaView>
  );
}

const styles = StyleSheet.create({

  container: {
    flex: 1,
    flexDirection: 'row',
    direction: 'ltr',
    justifyContent: 'flex-start',
    alignItems: 'flex-start',
    flexWrap: 'wrap',
    alignContent: 'space-around'

  },
  box: {
    width: 60,
    height: 60
  },
  cor1: {
    backgroundColor: '#00F'
  },
  cor2: {
    backgroundColor: '#05F'
  },
  cor3: {
    backgroundColor: '#0AF'
  },

})
```

Resumo

Nesta aula, vimos algumas propriedades do *flex*, sendo elas referentes às dimensões e distribuições dos elementos em tela. Ademais, vimos como funcionam os valores que podem ser aplicados a essas propriedades.