Curso de React Native - Semana 4: Estilos y Layout

Estructura de Carpetas y Archivos

Organizar los archivos y carpetas de manera eficiente es crucial en cualquier proyecto de React Native. Una estructura típica incluye directorios separados para componentes, pantallas, servicios (APIs), y recursos (imágenes, estilos).

CSS en React Native

A diferencia del CSS tradicional en desarrollo web, React Native utiliza un sistema de estilos en JavaScript. Los estilos se definen como objetos JavaScript y se asignan a los componentes de React Native mediante la propiedad `style`.

Ejemplo:

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

const MiComponente = () => {
  return (
    <View style={styles.container}>
      <Text style={styles.texto}>Hola, React Native!</Text>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#fff',
  },
  texto: {
    color: 'blue',
    fontSize: 20,
  },
});

export default MiComponente;
```

En este ejemplo, `StyleSheet.create` se utiliza para definir estilos, donde `container` y `texto` son estilos personalizados aplicados a los componentes `View` y `Text`.

Curso de React Native - Semana 4: Estilos y Layout

Flexbox para Layouts

Flexbox es un método de layout que permite organizar elementos de forma predecible en diferentes tamaños de pantalla y dispositivos. React Native utiliza Flexbox para el diseño de interfaces.

Propiedades comunes de Flexbox:

- `flexDirection`: Define la dirección principal del layout (columna o fila).
- `justifyContent`: Alinea los elementos en el eje principal.
- `alignItems`: Alinea los elementos en el eje transversal.

Ejemplo de Flexbox:

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
    backgroundColor: '#fff',
  },
  box: {
    width: 50,
    height: 50,
    backgroundColor: 'skyblue',
  },
});
```

En este ejemplo, `flexDirection` está establecido en 'row', lo que hace que los hijos de `container` se alineen horizontalmente. `justifyContent` y `alignItems` se utilizan para alinear los elementos dentro del contenedor.

Curso de React Native - Semana 4: Estilos y Layout

Ejemplos de Estilos y Layouts

A continuación, se muestra un ejemplo que combina estilos y layouts utilizando Flexbox en React Native:

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

const App = () => {
  return (
    <View style={styles.container}>
      <View style={styles.box}>
        <Text>Box 1</Text>
      </View>
      <View style={[styles.box, { backgroundColor: 'powderblue' }]}>
        <Text>Box 2</Text>
      </View>
      <View style={[styles.box, { backgroundColor: 'skyblue' }]}>
        <Text>Box 3</Text>
      </View>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    flexDirection: 'column',
    justifyContent: 'center',
    alignItems: 'stretch',
  },
  box: {
    height: 100,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: 'lightgray',
    marginBottom: 10,
  },
});

export default App;
```

En este código, hay tres cajas (componentes `View`) dentro de un contenedor. Cada caja tiene su propio estilo, y el contenedor utiliza Flexbox para posicionar y alinear estas cajas.