

DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MÓVILES

Universidad de Sonsonate

Estilización, diseño responsivo y
renderización de listas en React Native



Objetivo general

Comprender y aplicar estilos en React Native utilizando propiedades comunes y StyleSheet, adaptando el diseño a diferentes tamaños de pantalla, y renderizando elementos de forma eficiente mediante listas.

Estilización en React Native

En React Native, los estilos se escriben en **JavaScript**, pero siguen reglas parecidas a **CSS**, aunque con algunas diferencias:

- **CamelCase** en lugar de guiones (`backgroundColor` en lugar de `background-color`).
- No todos los estilos de CSS existen aquí.
- Los valores numéricos no llevan “px” (se asume que son píxeles).

Formas de aplicar estilos

- Estilos en línea
- `StyleSheet.create()`

Formas de aplicar Estilos

Estilos en línea (Inline Styles)

- Estilos dentro del **JSX** como objetos.
- Útiles para pruebas rápidas.
- Difíciles de reutilizar.

Código de ejemplo

JS App.js U X

JS App.js > App

```
1  import { Text, View } from 'react-native'
2
3  export default function App() {
4    return(
5      <View style={{marginTop: 300, marginLeft: 70}}>
6        <Text style={{fontSize: 50}}>Hola, mundo!</Text>
7      </View>
8    )
9  }
```

Formas de aplicar Estilos

StyleSheet

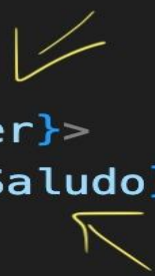
- Centraliza estilos en un objeto.
- Mejor rendimiento.
- Más limpio y **reutilizable**.

Código de ejemplo

JS App.js U X

JS App.js > ...

```
1  import { StyleSheet, Text, View } from 'react-native'
2
3  export default function App() {
4    return (
5      <View style={styles.container}>
6        <Text style={styles.txtSaludo}>Hola, mundo!</Text>
7      </View>
8    )
9  }
10 const styles = StyleSheet.create({
11   container: {
12     marginTop: 300,
13     marginLeft: 70
14   },
15   txtSaludo: {
16     fontSize: 50,
17     color: 'blue',
18   }
19 })
20
```



Aplicación de Colores

Formas de definir colores:

- Nombres: 'red'
- Hex: '#ff6600'
- RGB: 'rgb(255, 0, 0)'
- RGBA: 'rgba(0,0,0,0.5)'

jsx

```
texto: {  
  color: '#ff6600',  
  backgroundColor: 'black'  
}
```


Tipografía

Propiedades más comunes

- **fontSize** → Tamaño.
- **fontWeight** → Grosor (**bold**, **normal**, 100-900).
- **fontStyle** → Cursiva (**italic**).
- **textAlign** → Alineación (**left**, **center**, **right**).

jsx

```
texto: {  
  fontSize: 24,  
  fontWeight: 'bold',  
  textAlign: 'center',  
  color: 'darkblue'  
}
```

Bordes y redondeados

Propiedades más comunes

- `borderWidth`: grosor.
- `borderColor`: color.
- `borderRadius`: esquina redondeadas.

jsx

```
caja: {  
  borderWidth: 2,  
  borderColor: 'black',  
  borderRadius: 10,  
  padding: 10  
}
```

Sombras

- En React Native las sombras son importantes para dar profundidad en las vistas:
- En iOS se usa `shadowColor`, `shadowOpacity`, `shadowOffset`, `shadowRadius`.
- En Android solo usar `elevation`.

LOGIN

Ejemplo aplicando Sombras a una vista

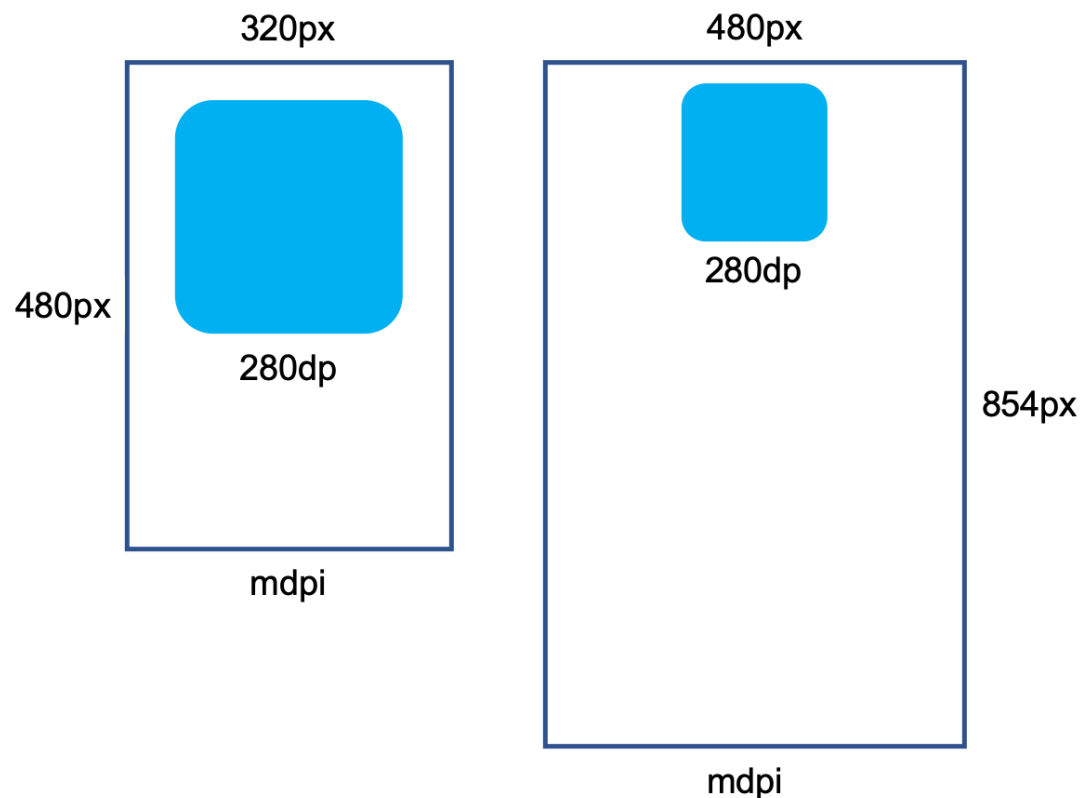


```
import React from 'react';
import { View, StyleSheet } from 'react-native';

export default function App() {
  return <View style={styles.box} />;
}

const styles = StyleSheet.create({
  box: {
    width: 100,
    height: 100,
    backgroundColor: 'white',
    shadowColor: '#000',
    shadowOpacity: 0.3,
    shadowOffset: { width: 2, height: 2 },
    shadowRadius: 4,
    elevation: 5,
  },
});
```

Dimensiones y porcentajes



- Se usa el módulo `Dimensions` para obtener el tamaño de pantalla y adaptar tamaños.
- También se puede usar porcentajes ('50%') para que elementos escalen proporcionalmente.
- Esto es clave para interfaces responsivas.

Ejemplo con Dimensions

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

const { width, height } = Dimensions.get('window');

export default function App() {
  return <View style={styles.box} />;
}

const styles = StyleSheet.create({
  box: {
    width: width * 0.5,    // 50% ancho pantalla
    height: height * 0.2, // 20% alto pantalla
    backgroundColor: 'orange',
  },
});
```

Ejemplo con Porcentajes

```
StyleSheet.create({  
  btn: {  
    width: '80%',  
    height: '30%',  
  }  
});
```



Uso de SafeAreaView

Es útil para evitar que el contenido se esconda detrás de notches o barras de estado.

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

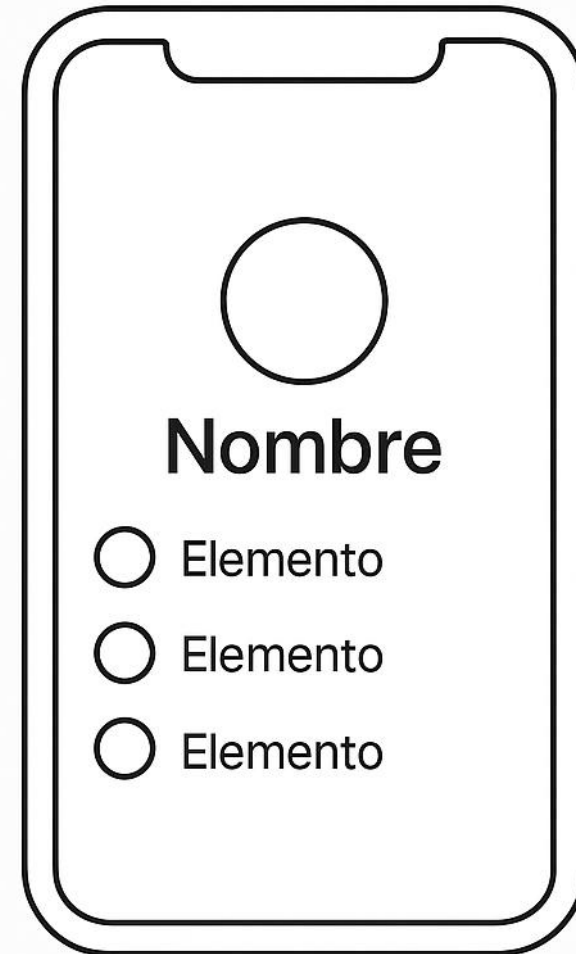
export default function App() {
  return (
    <SafeAreaView style={{ flex: 1, backgroundColor: 'lightyellow' }}>
      <Text>Contenido seguro</Text>
    </SafeAreaView>
  );
}
```


REALIZANDO EJEMPLO

EJEMPLO PRÁCTICO INTEGRANDO TODO LO ANTERIOR...

Renderizado de listas en React Native

**Renderizado de
listas: de lo
simple a lo
eficiente**



En React Native, existen dos formas principales de mostrar listas de datos:

1. Usando `.map()`

- Ideal para listas **pequeñas** o estáticas.
- Sintaxis sencilla: recorre un arreglo y devuelve componentes.
- No optimizado para rendimiento en listas grandes.

2. Usando `FlatList`

- Diseñado para listas **largas o dinámicas**.
- Solo renderiza lo que se ve en pantalla → **mejor rendimiento**.
- Incluye scroll y optimizaciones internas.

Ejemplo de listas usando .map()

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

export default function App() {
  const frutas = ['🍏 Manzana', '🍇 Uva', '🍊 Naranja'];

  return (
    <View style={styles.container}>
      {frutas.map((fruta, index) => (
        <Text key={index} style={styles.item}>
          {fruta}
        </Text>
      )))}
    </View>
  );
}

const styles = StyleSheet.create({
  container: { padding: 20 },
  item: { fontSize: 18, marginBottom: 8 },
});
```

Ejemplo de listas usando FlatList

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

export default function App() {
  const frutas = ['🍏 Manzana', '🍇 Uva', '🍊 Naranja'];

  return (
    <View style={styles.container}>
      <FlatList
        data={frutas}
        renderItem={({ item }) => (
          <Text style={styles.item}>{item}</Text>
        )}
        keyExtractor={(item, index) => index.toString()}
      />
    </View>
  );
}

const styles = StyleSheet.create({
  container: { padding: 20 },
  item: { fontSize: 18, marginBottom: 8 },
});
```

Realizando Ejemplo

- EJEMPLO PRÁCTICO INTEGRANDO LISTAS AL EJEMLO ANTERIOR

Ejercicio



Juan Pérez

juan.perez@email.com

Editar Perfil

PlayLists

Musica para trabajar

Musica Pop

Musica para ejercicios

Musica con guitarra

Musica Clasica

Gracias por su atención