React Native Components & Styling: Complete Beginner Guide

Subtitle: Master the building blocks of mobile apps and design visually appealing screens.

Website Name: haas.dev
Website Link: https://dev-roast-app.vercel.app

Introduction

Components and styling are the heart of React Native apps. Understanding how to create reusable components and style them correctly allows you to build consistent and attractive mobile applications. This guide explains core components, layout techniques, and styling best practices for beginners.

Step 1: Core Components

React Native comes with prebuilt components:

- **View:** Container for other components, like a `div` in HTML.
- **Text:** Display text content.
- **Image:** Show images from local assets or URLs.
- **ScrollView:** Make content scrollable vertically or horizontally.
- **TextInput:** Take user input.
- **Button / TouchableOpacity / Pressable:** Handle taps and gestures.

Example:

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

export default function App() {
  return (
    <View style={{ padding: 20 }}>
      <Text>Hello, React Native!</Text>
      <Button title="Click Me" onPress={() => alert('Button pressed')} />
    </View>
  );
}
```

Step 2: Styling Basics

React Native uses **JavaScript objects** for styling. CSS is not supported directly.

- Use `StyleSheet.create()` for reusable styles.
- Styles are camelCased (`backgroundColor`, `fontSize`).
- Common style properties:
  - Layout: `flex`, `flexDirection`, `justifyContent`, `alignItems`
  - Text: `fontSize`, `color`, `fontWeight`
  - Spacing: `margin`, `padding`
  - Background: `backgroundColor`, `borderRadius`

Example:

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

export default function App() {
  return (
    <View style={styles.container}>
      <Text style={styles.header}>Hello React Native</Text>
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: '#f2f2f2',
  },
  header: {
    fontSize: 24,
    fontWeight: 'bold',
    color: '#333',
  },
});
```

Step 3: Layout with Flexbox

Step 3: Layout with Flexbox

React Native uses **Flexbox** for layout:

- Default `flexDirection` is `column`.
- `justifyContent` aligns items vertically (main axis).
- `alignItems` aligns items horizontally (cross axis).

Flexbox Example:

```
<View style={{ flex: 1, flexDirection: 'row', justifyContent: 'space-around', al
  <View style={{ width: 50, height: 50, backgroundColor: 'red' }} />
  <View style={{ width: 50, height: 50, backgroundColor: 'blue' }} />
  <View style={{ width: 50, height: 50, backgroundColor: 'green' }} />
</View>
```

Exercise: Create a layout with **three colored boxes** spaced evenly horizontally.

Step 4: Styling Tips for Beginners

- Use **consistent spacing and colors** across the app.
- Avoid inline styles for large projects; prefer **StyleSheet**.
- Use **percentage width or flex** instead of fixed pixels for responsiveness.
- Test on both **iOS and Android** — some styles render differently.
- Leverage **React Native community libraries** for buttons, cards, and UI kits.

Step 5: Creating Reusable Components

- Encapsulate repeated UI into a component.
- Pass **props** to customize content or behavior.

Example: Custom Button Component

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

export default function CustomButton({ title, onPress }) {
  return (
    <TouchableOpacity style={styles.button} onPress={onPress}>
```

```
      <Text style={styles.text}>{title}</Text>
    </TouchableOpacity>
  );
}

const styles = StyleSheet.create({
  button: {
    backgroundColor: '#007bff',
    padding: 12,
    borderRadius: 8,
  },
  text: {
    color: '#fff',
    fontWeight: 'bold',
    textAlign: 'center',
  },
});
```

Exercise: Build a reusable **Card component** that takes a title and description as props and styles it with padding, border, and shadow.

Key Takeaways

- Master **View, Text, Image, ScrollView, and TextInput** to build app screens.
- Styling in React Native is **JavaScript-based**, use `StyleSheet` for maintainability.
- **Flexbox layout** is essential for responsive design.
- **Reusable components** improve scalability and code organization.
- Consistency in design and testing on both platforms is crucial.
