Cheatsheets / **Learn React Native**

codecademy

Styling Components

StyleSheet

`StyleSheet` is an abstraction that replaces CSS by accepting CSS styling rules using a two-dimensional JavaScript object.

```
import { StyleSheet } from 'react-native';

const styles = StyleSheet.create({
  paragraph: {
    fontSize: 16,
  },
  label: {
    fontSize: 11,
    textTransform: 'uppercase'
  }
});

<Text style={styles.paragraph}>My paragraph</Text>
<Text style={styles.label}>My label</Text>
```

style Property

Components can be styled using the `style={}` property, which accepts objects as inline-styling, style definitions created by `StyleSheet`, or an array of objects/definitions to compose styling.

```
<Text style={styles.paragraph} />
<Text style={{ fontSize: 16 }} />
<Text style={[styles.paragraph, { color: 'red' }]} />
```

Using `StyleSheet` Definitions

Splitting styling properties from the render method using `StyleSheet` definitions makes the rendering method more readable.

```
// Using inline styling
const AwesomeBox = () => (
  <View style={{ width: 100, height: 100,
backgroundColor: 'red' }} />
);

// Using the StyleSheet API
const AwesomeBox = () => (
  <View style={styles.box} />
);

const styles = StyleSheet.create({
  box: {
    width: 100,
    height: 100,
    backgroundColor: 'red'
  },
});
```

Dynamic Styling

You can make component styling dynamic by adding JavaScript logic for the `style={}` prop or providing inline styles to override single properties.

```
// Applies the `selected` style on top of
the `paragraph` style if props.isActive
is truthy

function Item(props) {
  return (
    <Text style={[styles.paragraph,
props.isActive && styles.selected]} />
  );
}
```

Flex in React Native

Layouts are defined with Flex-like rules to account for a wide variety of screen sizes. The major difference between Flex on web and Flex in React Native is that a parent element with `display: flex` is not required.

```
<View style={{ flexDirection: 'row' }}>
  <View style={{ flex: 1 }} />
  <View style={{ flex: 1 }} />
  <View style={{ flex: 1 }} />
</View>
```

codecademy

flexDirection

The `flexDirection` style property determines the direction and order in which child elements are laid out, which could be `row`, `row-reverse`, `column`, or `column-reverse`.

```
<View style={{ flexDirection: 'row' }}>
  <View style={{ flex: 1 }} />
  <View style={{ flex: 1 }} />
  <View style={{ flex: 1 }} />
</View>
```

justifyContent

The `justifyContent` style property determines how child elements are positioned in the parent container, which could be `center`, `flex-start`, `flex-end`, `space-around`, `space-between`, or `space-evenly`.

```
<View style={{ flexDirection: 'row' ,
justifyContent: 'flex-start' }}>
  <View style={{ flex: 1 }} />
  <View style={{ flex: 1 }} />
  <View style={{ flex: 1 }} />
</View>
```

Dimensions in React Native

All dimensions are unitless by default, and represent density-independent pixels.

```
<View style={{ width: 50, height: 50,
backgroundColor: 'powderblue' }} />
```

Print Share