**State**

There are two types of data that control a component: props and state. props are set by the parent and they are fixed throughout the lifetime of a component. For data that is going to change, we have to use state.

In general, you should initialize state in the constructor, and then call setState when you want to change it.

For example, let's say we want to make text that blinks all the time. The text itself gets set once when the blinking component gets created, so the text itself is a prop. The "whether the text is currently on or off" changes over time, so that should be kept in state.

**useEffect**

The useEffect Hook allows you to perform side effects in your components.



**Props**

Most components can be customized when they are created, with different parameters. These created parameters are called props, short for properties.

For example, one basic React Native component is the Image. When you create an image, you can use a prop named source to control what image it shows.

****

Your own components can also use props. This lets you make a single component that is used in many different places in your app, with slightly different properties in each place by referring to props in your render function. Here's an example:

****

**StyleSheet**

A StyleSheet is an abstraction similar to CSS StyleSheets

**Methods**

*compose()*

static compose(style1: object, style2: object)

Combines two styles such that style2 will override any styles in style1. If either style is falsy, the other one is returned without allocating an array, saving allocations and maintaining reference equality for PureComponent checks.

Graphical user interface, text

Description automatically generated

*create()*

Creates a StyleSheet style reference from the given object.

*flatten()*

Flattens an array of style objects, into one aggregated style object. Alternatively, this method can be used to lookup IDs, returned by StyleSheet.register.

**Properties**

*absoluteFill:*

A very common pattern is to create overlays with position absolute and zero positioning (position: 'absolute', left: 0, right: 0, top: 0, bottom: 0), so absoluteFill can be used for convenience and to reduce duplication of these repeated styles.

*hairlineWidth:*

This is defined as the width of a thin line on the platform. It can be used as the thickness of a border or division between two elements.