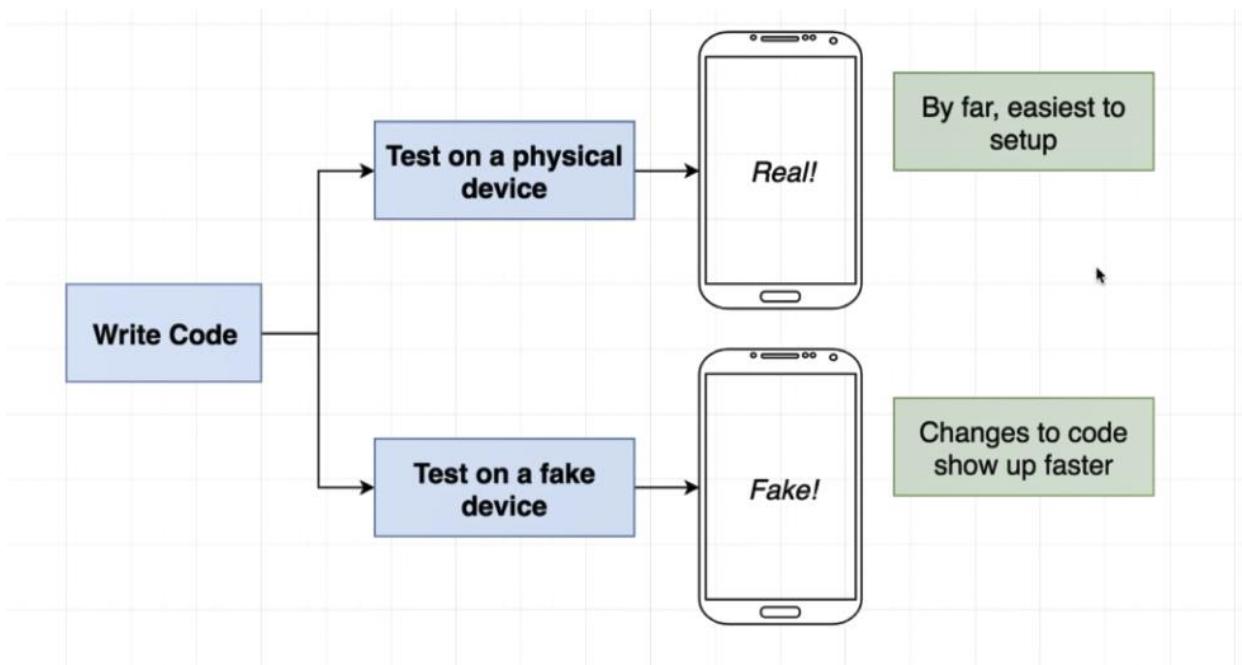


React Native

1. Getting Started



Physical Device Setup

Download the 'rn-starter.zip' file in the lecture before this one. Extract into a workspace directory

In your terminal, change into the 'rn-starter' folder that was just created

In your terminal, run '**npm install**'

Error? Make sure you have node installed + updated. nodejs.org/en/download/

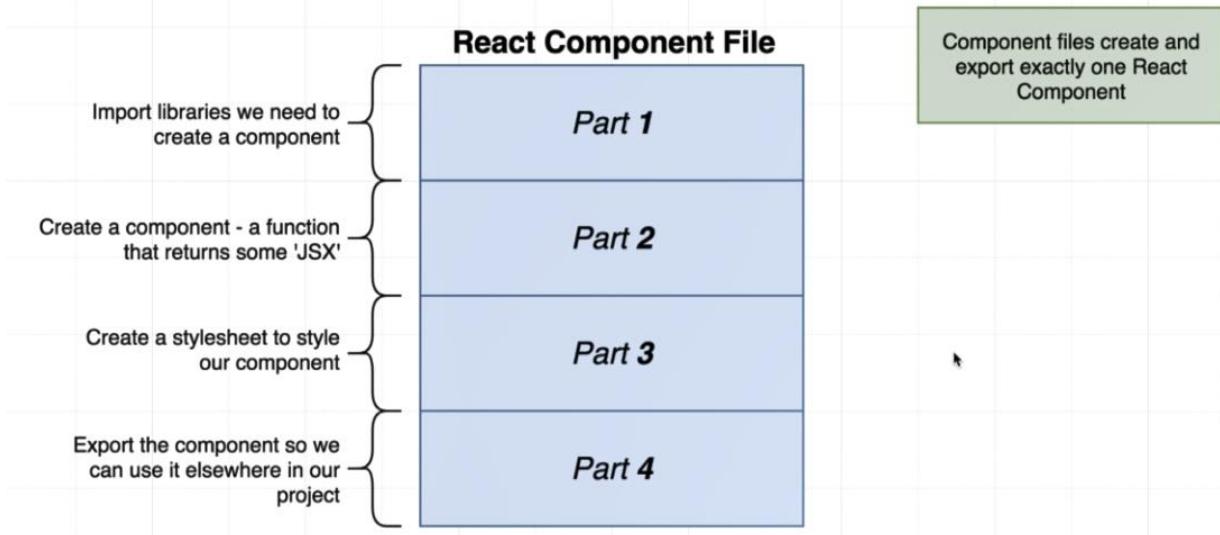
After installing, run '**npm start**'. This opens the React Native bundler, which gets your code ready to be ran on a mobile device

Install the '**expo**' app on your mobile device from the App Store or Google Play Store

Scan the QR code from the React Native Bundler on your phone

2. Working with Content

1. Overview of React Components



The screenshot shows a code editor interface with two tabs: 'HomeScreen.js' and 'ComponentsScreen.js'. The 'ComponentsScreen.js' tab is active, displaying the following code:

```
src > screens > ComponentsScreen.js > ...
1 import React from 'react';
2 import { Text, StyleSheet } from 'react-native';
3
4 const ComponentsScreen = () => {
5   return <Text style={styles.textStyle}>This is the Components Screen.</Text>;
6 };
7
8 const styles = StyleSheet.create({
9   textStyle: {
10     fontSize: 30
11   }
12 });
13
14 export default ComponentsScreen;
```

The code defines a component named 'ComponentsScreen' that returns a single text element. It uses a local stylesheet named 'styles' to set the font size to 30. The component is exported at the bottom.

2. Showing a Custom Component

The screenshot shows a code editor interface with two tabs open: 'ComponentsScreen.js' and 'App.js'. The 'App.js' tab is active, displaying the following code:

```
1 import { createStackNavigator, createAppContainer } from 'react-navigation';
2 import HomeScreen from './src/screens/HomeScreen';
3 import ComponentsScreen from './src/screens/ComponentsScreen';
4
5 const navigator = createStackNavigator(
6   {
7     Home: HomeScreen,
8     Components: ComponentsScreen
9   },
10  {
11    initialRouteName: 'Components',
12    defaultNavigationOptions: {
13      title: 'App'
14    }
15  }
16);
17
18 export default createAppContainer(navigator)
19
```

Below the code editor is a status bar with the following information: Ln 10, Col 4, Spaces: 2, UTF-8, LF, JavaScript, Prettier: ✓, 0 errors, 1 warning.

To the right of the code editor is a mobile application simulator window titled 'iPhone XS - 12.1'. The screen displays the text 'This is the components screen' under the title 'App'.

3. Common Questions and Answers

Questions	Answers
What's that 'Text' thing?	A 'primitive' React element. Used to show some basic content on the screen.
What's that HTML looking stuff?	JSX! It is a 'dialect' of Javascript that tells React what content we want to show
What's that 'appNavigator' in the 'App.js' file?	Its a tool from a library called 'React Navigation' that is used to show different screens to the user
How's that 'styles' thing work?	The 'StyleSheet.create()' function validates a set of styling rules that we pass into it. We can use it or pass styling directly into an element

Use this link to show the JSX format: <https://babeljs.io/repl>

```
const ComponentsScreen = () => {
  return React.createElement(Text, {
    style: styles.textStyle
  }, "This is the Components Screen. Hello World");
};
```

4. Rules of JSX

Rules of JSX

We can assemble different JSX elements like normal HTML

We configure different JSX elements using 'props'

We can refer to JS variables inside of a JSX block by using curly braces

We can assign JSX elements to a variable, then show that variable inside of a JSX block

The screenshot shows a code editor on the left and an iPhone simulator on the right. The code editor displays the file `ComponentsScreen.js` with the following content:

```
JS ComponentsScreen.js ✘ JS App.js
src > screens > JS ComponentsScreen.js > ComponentsScreen
1 import React from 'react';
2 import { Text, StyleSheet, View } from 'react-native';
3
4 const ComponentsScreen = () => {
5   const greeting = 'hi there';
6
7   return (
8     <View>
9       <Text style={styles.textStyle}>This is the components screen</Text>
10      <Text>{greeting}</Text>
11    </View>
12  );
13};
14
15 const styles = StyleSheet.create({
16   textStyle: {
17     fontSize: 30
18   }
19});
```

The iPhone simulator shows the application running with the text "This is the components screen" and "hi there" displayed on the screen.

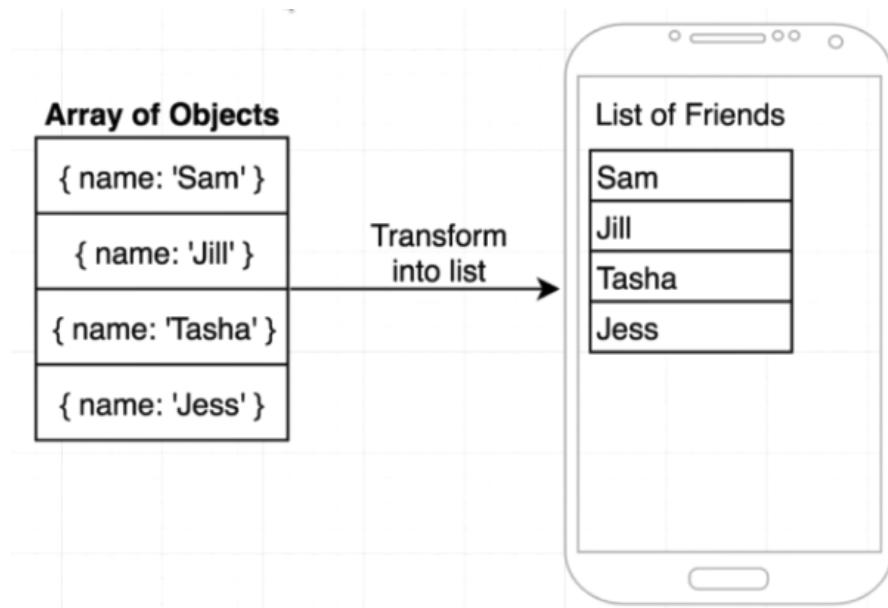
The screenshot shows a code editor and a mobile application preview side-by-side. The code editor on the left contains `App.js` with the following content:

```
JS ComponentsScreen.js x JS App.js
src > screens > JS ComponentsScreen.js > ComponentsScreen
1 import React from 'react';
2 import { Text, StyleSheet, View } from 'react-native';
3
4 const ComponentsScreen = () => {
5   const greeting = <Text>Hello to you!</Text>;
6
7   return (
8     <View>
9       <Text style={styles.textStyle}>This is the components
10      {greeting}
11    </View>
12  );
13};
14
15 const styles = StyleSheet.create({
16   textStyle: {
17     fontSize: 30
18   }
19});
```

The mobile application preview on the right shows the text "This is the components screen" above "Hello to you!".

3. List Building - With Style

1. The FlatList Element



FlatList Element

Turns an array into a list of elements

We are required to pass in a 'prop' of 'data' - the array of data that we are going to create a bunch of elements out of

Also required to pass in a 'renderItem' prop - function that will turn each individual item into an element

If you are coming from React on the web, you might be used to 'mapping' an array of data to build a list - FlatList is better with RN

2. Rendering a FlatList

The screenshot shows a code editor interface with a sidebar labeled "EXPLORER" on the left. The "src" folder contains several files: HomeScreen.js, App.js, ComponentsScreen.js, and ListScreen.js (which is currently selected). The "RN-STARTER" folder contains .expo, assets, node_modules, and a src folder which further contains screens, .gitignore, .watchmanconfig, App.js, app.json, babel.config.js, package-lock.json, and package.json.

The main editor area displays the content of ListScreen.js:

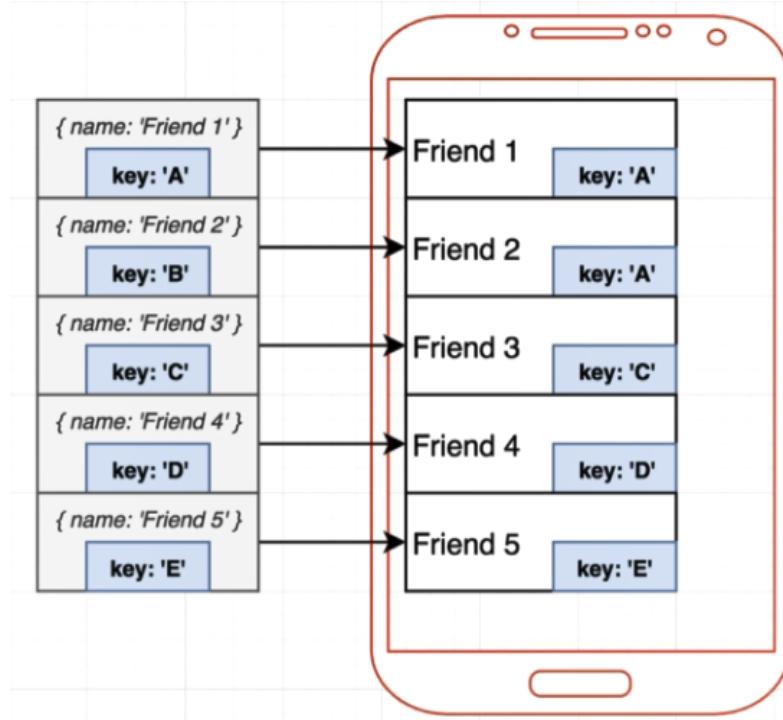
```
JS HomeScreen.js JS App.js JS ComponentsScreen.js JS ListScreen.js ●

OPEN EDITOR... 1 UNSAVED
src > screens > JS ListScreen.js ...
JS HomeScreen.js sr...
JS App.js
JS ComponentsScre...
● JS ListScreen.js src\sc...
RN-STARTER
> .expo
> assets
> node_modules
src
> screens
JS ComponentsScree...
JS HomeScreen.js
JS ListScreen.js
.gitignore
{.watchmanconfig
JS App.js
{} app.json
JS babel.config.js
{} package-lock.json
{} package.json

JS HomeScreen.js      JS App.js      JS ComponentsScreen.js      JS ListScreen.js ●

1 import React from 'react';
2 import { Text, View, StyleSheet, FlatList } from 'react-native';
3
4 const ListScreen = () => {
5   const friends = [
6     { name: 'Friend #1' },
7     { name: 'Friend #2' },
8     { name: 'Friend #3' },
9     { name: 'Friend #4' },
10    { name: 'Friend #5' },
11  ];
12  return (
13    <FlatList
14      data={friends}
15      renderItem={({ item }) => {
16        return <Text>{item.name}</Text>
17      }}
18    />
19  );
20};
21
22 const styles = StyleSheet.create({});
23
24 export default ListScreen;
```

3. Solving the Key Issue



```
src > screens > JS ListScreen.js > ...
1  import React from 'react';
2  import { Text, View, StyleSheet, FlatList } from 'react-native';
3
4  const ListScreen = () => {
5    const friends = [
6      { name: 'Friend #1' },
7      { name: 'Friend #2' },
8      { name: 'Friend #3' },
9      { name: 'Friend #4' },
10     { name: 'Friend #5' },
11   ];
12   return (
13     <FlatList
14       keyExtractor={friend => friend.name}
15       data={friends}
16       renderItem={({ item }) => {
17         return <Text>{item.name}</Text>
18       }}
19     />
20   );
21 };
22
23 const styles = StyleSheet.create({});
```

```
24
25 export default ListScreen;
```

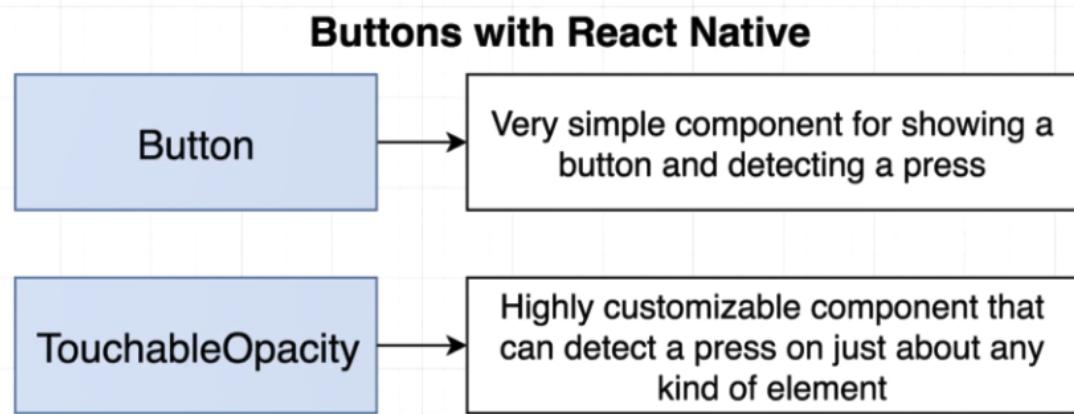
{ name: 'Friend #1' , key: '1' } we can use unique key for each items.

4. A Few Props Around FlatList

```
src > screens > JS ListScreen.js > ...
1  import React from 'react';
2  import { Text, View, StyleSheet, FlatList } from 'react-native';
3
4  const ListScreen = () => {
5    const friends = [
6      { name: 'Friend #1', age: 24 },
7      { name: 'Friend #2', age: 21 },
8      { name: 'Friend #3', age: 29 },
9      { name: 'Friend #4', age: 21 },
10     { name: 'Friend #5', age: 27 },
11     { name: 'Friend #6', age: 23 },
12     { name: 'Friend #7', age: 23 },
13     { name: 'Friend #8', age: 28 },
14     { name: 'Friend #9', age: 20 },
15   ];
16  return (
17    <FlatList
18      //horizontal
19      keyExtractor = {friend => friend.name}
20      data={friends}
21      renderItem={({ item }) => {
22        return (
23          <Text style={ styles.textStyle}>
24            | Name: {item.name} - Age: {item.age}
25          </Text>
26        );
27      }}
28    />
29  );
30};
31
32 const styles = StyleSheet.create({
33   textStyle: {
34     marginHorizontal: 20,
35     marginVertical: 20
36   }
37 });
38
39 export default ListScreen;
```

ShowsHorizontalScrollIndicator?

4. Navigating Users Between Screens



1. Button in Action

```
JS App.js          JS HomeScreen.js ×  
src > screens > JS HomeScreen.js > ...  
1  import React from 'react';  
2  import { Text, StyleSheet, Button, View } from 'react-native';  
3  
4  const HomeScreen = () => {  
5    return (  
6      <View>  
7        <Text style={styles.text}>Hello World!</Text>  
8        <Button  
9          title="Go to Components Demo"  
10         onPress={() => console.log('Button Pressed')}  
11       />  
12     </View>  
13   );;  
14 };  
15  
16  const styles = StyleSheet.create({  
17    text: {  
18      fontSize: 30,  
19      padding: 20  
20    }  
21  });  
22  
23  export default HomeScreen;  
24
```

2. Touchable Opacity in Action

```
src > screens > JS HomeScreen.js > [x] styles
```

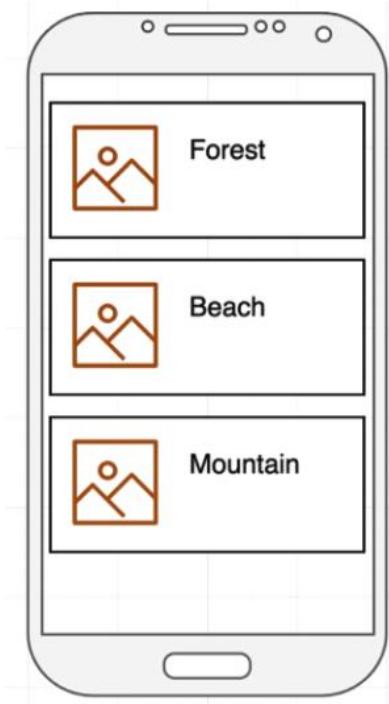
```
1 import React from 'react';
2 import { Text, StyleSheet, Button, View, TouchableOpacity } from 'react-native';
3
4 const HomeScreen = () => {
5   return (
6     <View style={styles.viewStyle}>
7       <Text style={styles.text}>Hello World!</Text>
8       <Button
9         title="Go to Components Demo"
10        onPress={() => console.log('Button Pressed')}
11      />
12      <TouchableOpacity onPress={() => console.log('List Pressed')}>
13        <Text>Go to List Demo</Text>
14      </TouchableOpacity>
15    </View>
16  );
17};
18
19 const styles = StyleSheet.create({
20   text: {
21     fontSize: 30,
22     padding: 20
23   },
24   viewStyle: {
25     marginHorizontal: 15,
26     marginVertical: 15
27   }
28 });
29
30 export default HomeScreen;
31
```

3. Navigating with React Navigation

```
const HomeScreen = (props) => {
//console.log(props.navigation.navigate);
return (
<View style={styles.viewStyle}>
  <Text style={styles.text}>Hello World!</Text>
  <Button
    title="Go to Components Demo"
    onPress={() => props.navigation.navigate('Components')}
  />
  <TouchableOpacity onPress={() => props.navigation.navigate('List')}>
    <Text>Go to List Demo</Text>
  </TouchableOpacity>
</View>
);
};
```

```
const HomeScreen = ({ navigation }) => {
  return (
    <View style={styles.viewStyle}>
      <Text style={styles.text}>Hello World!</Text>
      <Button
        title="Go to Components Demo"
        onPress={() => navigation.navigate('Components')}
      />
      <Button
        title="Go to List Demo"
        onPress={() => navigation.navigate('List')}
      />
    </View>
  );
};
```

5. Building Reusable Components



Same group of elements, repeated three times

We could repeat the same JSX three times over, or we could create a *separate component*

Create ImageScreen.js and add button for HomeScreen to Navigate to the Image Screen.

```

EXPLORER JS App.js JS HomeScreen.js JS ImageScreen.js ...
OPEN EDITOR... 2 UNSAVED GROUP 1
JS App.js
JS HomeScreen.js sr...
● JS ImageScreen.js sr...
GROUP 2
● JS ImageDetail.js src...
RN-STARTER >.expo
> assets
> node_modules
src < components
JS ImageDetail.js
< screens
JS ComponentsScreen.js
JS HomeScreen.js
JS ImageScreen.js
JS ListScreen.js
.gitignore
{} .watchmanconfig
JS App.js
{} app.json
JS babel.config.js
{} package-lock.json
{} package.json

```

ImageScreen.js

```

1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3 import ImageDetail from '../components/ImageDetail'
4
5 const ImageScreen = () => {
6   return(
7     <View style={styles.viewStyle}>
8       <ImageDetail />
9       <ImageDetail />
10      <ImageDetail />
11      <ImageDetail />
12      <ImageDetail />
13     </View>
14   );
15 };
16
17 const styles = StyleSheet.create({
18   viewStyle: {
19     marginVertical: 30,
20     marginHorizontal: 30
21   }
22 });
23
24 export default ImageScreen;

```

ImageDetail.js

```

src > components > JS ImageDetail.js ...
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 const ImageDetail = () => {
5   return(
6     <Text>Image Detail</Text>
7   );
8 }
9
10 const styles = StyleSheet.create({});
11
12 export default ImageDetail;
13

```

1. Communicating from Parent to Child

ImageScreen.js

```

src > screens > JS ImageScreen.js ...
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3 import ImageDetail from '../components/ImageDetail'
4
5 const ImageScreen = () => {
6   return(
7     <View style={styles.viewStyle}>
8       <ImageDetail title="Forest" />
9       <ImageDetail title="Beach"/>
10      <ImageDetail title="Mountain"/>
11     </View>
12   );
13 };
14
15 const styles = StyleSheet.create({
16   viewStyle: {
17     marginVertical: 30,
18     marginHorizontal: 30
19   }
20 });
21
22 export default ImageScreen;

```

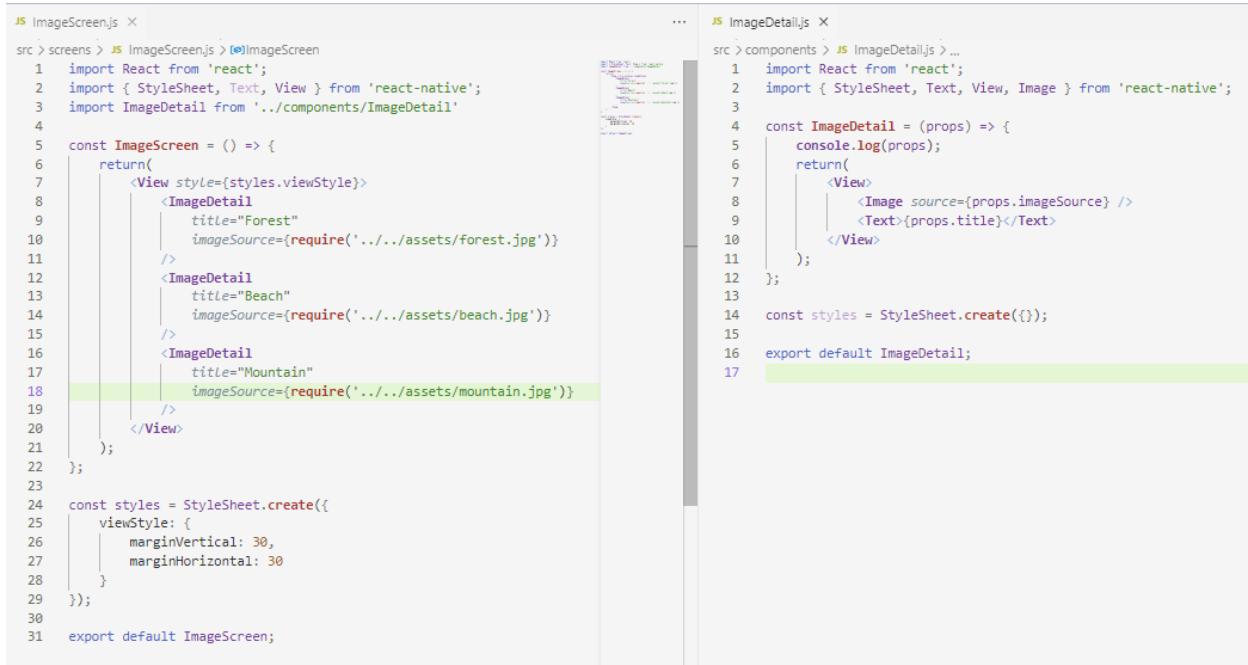
ImageDetail.js

```

src > components > JS ImageDetail.js ...
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3
4 const ImageDetail = (props) => {
5   console.log(props);
6   return(
7     <Text>{props.title}</Text>
8   );
9 }
10
11 const styles = StyleSheet.create({});
12
13 export default ImageDetail;
14

```

2. Passing Images as Props

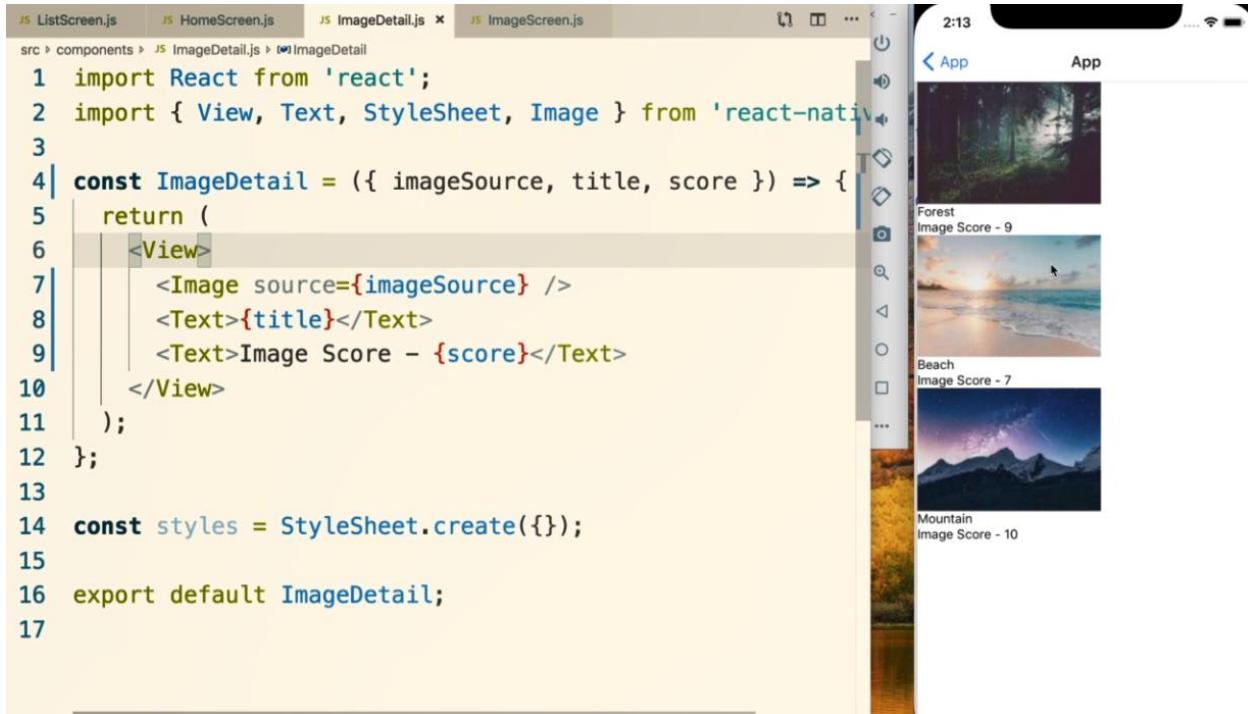


```
JS ImageScreen.js ...
src > screens > JS ImageScreen.js > [o]ImageScreen
1 import React from 'react';
2 import { StyleSheet, Text, View } from 'react-native';
3 import ImageDetail from '../components/ImageDetail'
4
5 const ImageScreen = () => {
6   return(
7     <View style={styles.viewStyle}>
8       <ImageDetail
9         title="Forest"
10        imageSource={require('../assets/forest.jpg')}
11      />
12      <ImageDetail
13        title="Beach"
14        imageSource={require('../assets/beach.jpg')}
15      />
16      <ImageDetail
17        title="Mountain"
18        imageSource={require('../assets/mountain.jpg')}
19      />
20    </View>
21  );
22}
23
24 const styles = StyleSheet.create({
25   viewStyle: {
26     marginVertical: 30,
27     marginHorizontal: 30
28   }
29 });
30
31 export default ImageScreen;
```

```
JS ImageDetail.js ...
src > components > JS ImageDetail.js > ...
1 import React from 'react';
2 import { StyleSheet, Text, View, Image } from 'react-native';
3
4 const ImageDetail = (props) => {
5   console.log(props);
6   return(
7     <View>
8       <Image source={props.imageSource} />
9       <Text>{props.title}</Text>
10    </View>
11  );
12}
13
14 const styles = StyleSheet.create({});
```

```
16 export default ImageDetail;
```

```
17
```



```
JS ListScreen.js JS HomeScreen.js JS ImageDetail.js > JS ImageScreen.js
src > components > JS ImageDetail.js > [o]ImageDetail
1 import React from 'react';
2 import { View, Text, StyleSheet, Image } from 'react-native';
3
4 const ImageDetail = ({ imageSource, title, score }) => {
5   return (
6     <View>
7       <Image source={imageSource} />
8       <Text>{title}</Text>
9       <Text>Image Score - {score}</Text>
10    </View>
11  );
12}
13
14 const styles = StyleSheet.create({});
```

```
15
16 export default ImageDetail;
```

```
17
```

2:13

App

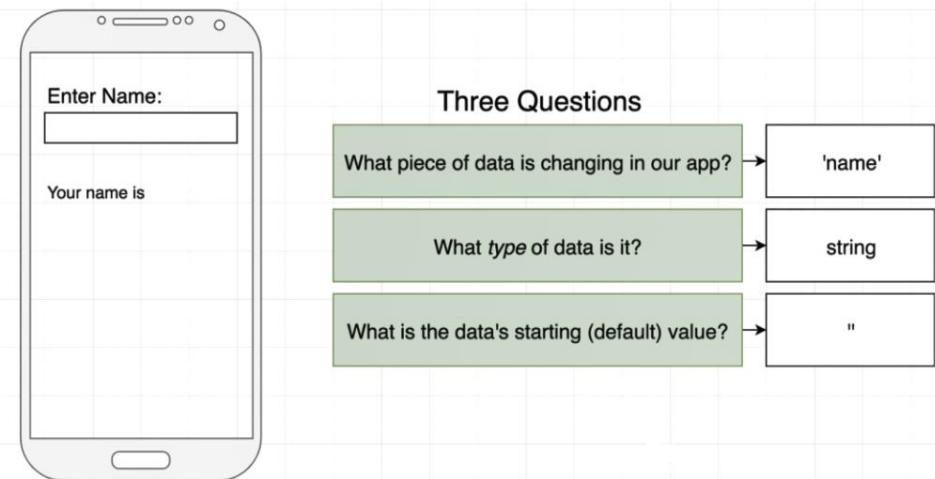
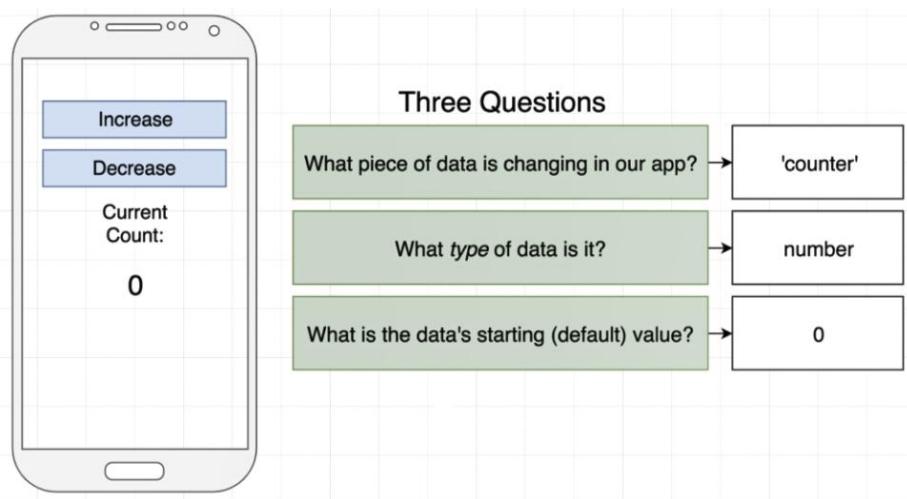
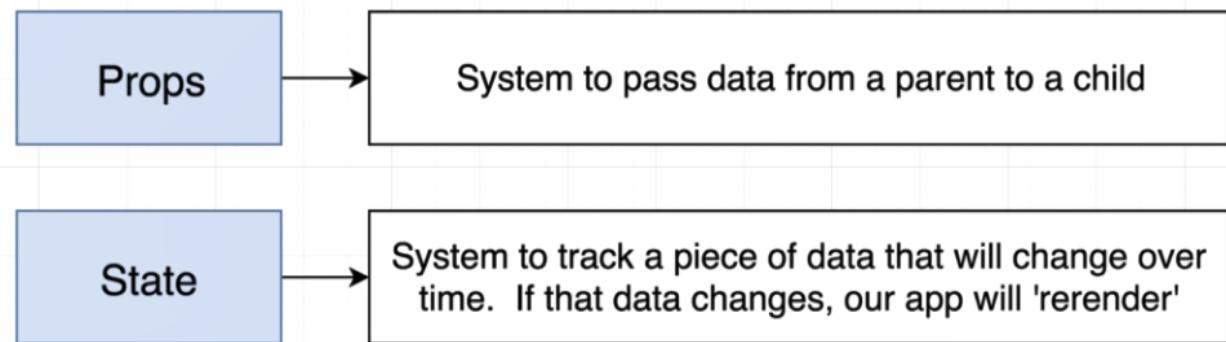
App

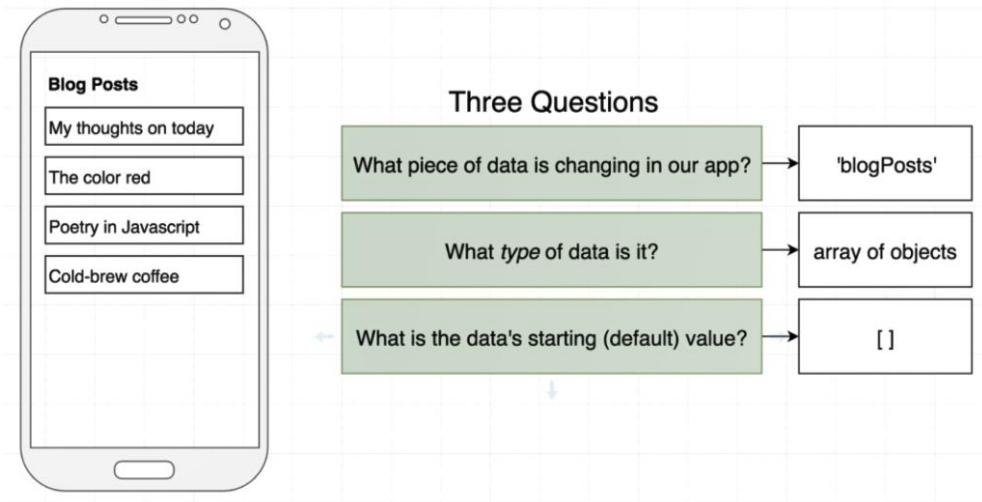
Forest
Image Score - 9

Beach
Image Score - 7

Mountain
Image Score - 10

6. State Management in React Components





Create a CounterScreen.js File in screens folder.

```

JS ImageScreen.js      JS CounterScreen.js X   JS ListScreen.js      JS HomeScreen.js
src > screens > JS CounterScreen.js > [o] styles
1  import React from 'react';
2  import { Text, View, StyleSheet, Button } from 'react-native';
3
4  const CounterScreen = () => {
5
6    let counter = 0;
7
8    return (
9      <View style={styles.viewStyle}>
10     <Button
11       title="Increase"
12       onPress={() => {
13         counter++;
14         console.log(counter);
15       }}
16     <Button
17       title="Decrease"
18       onPress={() => {
19         counter--;
20         console.log(counter);
21       }}
22     <Text>Current Count: {counter}</Text>
23   </View>
24 );
25 };
26
27 const styles = StyleSheet.create([
28   viewStyle: {
29     padding: 15
30   }
31 ]);
32
33 export default CounterScreen;

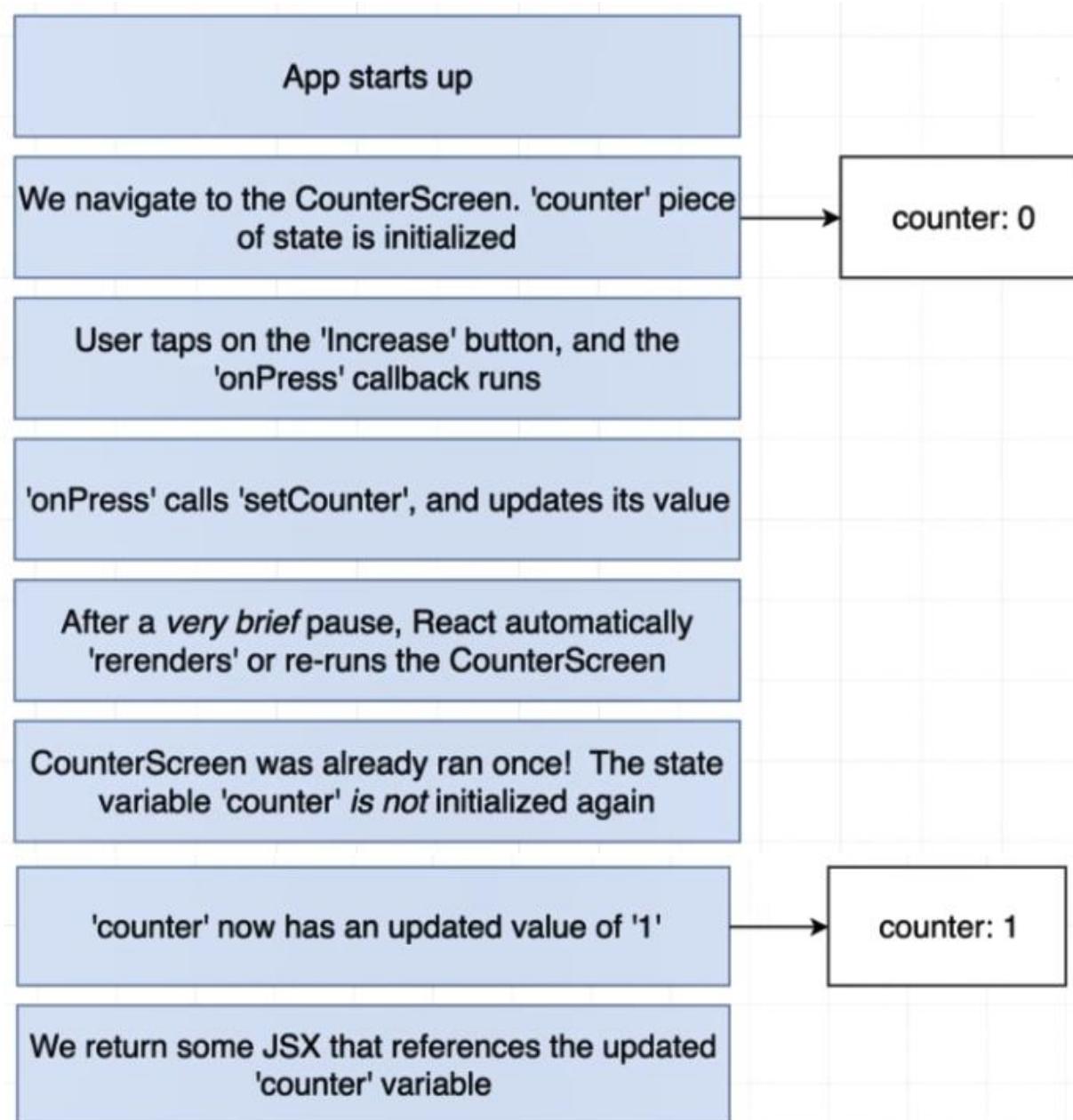
```

When press button, counter is up or down in console log. But not showing in App.

1. State in Action

We have to import useState from react Library. After it's worked well.

```
src > screens > JS CounterScreen.js > ...
1  import React, { useState } from 'react';
2  import { Text, View, StyleSheet, Button } from 'react-native';
3
4  const CounterScreen = () => {
5      const [counter, setCounter] = useState(0);
6      return (
7          <View style={styles.viewStyle}>
8              <Button title="Increase" onPress={() => {
9                  setCounter(counter + 1);
10             }} />
11              <Button title="Decrease" onPress={() => {
12                  setCounter(counter - 1);
13             }} />
14              <Text>Current Count: {counter}</Text>
15          </View>
16      );
17  };
18
19  const styles = StyleSheet.create({
20      viewStyle: {
21          padding: 15
22      }
23 });
24
25  export default CounterScreen;
```



Few Notes on State

We are using function-based state in a functional component. React also has class-based components that have access to state

We **never** directly modify a state variable. React doesn't detect this change!
Only use the 'setter' function

We can name the state variable anything we wish

We can track any kind of data that changes over time - a number, string, array of objects, etc

When a component is rerendered, *all of its children get rerendered too*

A state variable can be passed to a child component! At that point, the state variable is now being used as props.

2. Generating Random Colors



Three Questions

- | | | |
|--|---|------------------|
| What piece of data is changing in our app? | → | colors |
| What <i>type</i> of data is it? | → | array of strings |
| What is the data's starting (default) value? | → | [] |

```

JS ColorScreen.js × JS App.js
src > screens > JS ColorScreen.js > ...
1  import React from 'react';
2  import { View, Text, StyleSheet, Button } from 'react-native';
3
4  const ColorScreen = () => {
5    return (
6      <View style={ styles.viewStyle }>
7        <Text>ColorScreen</Text>
8        <Button title="Add a Color">
9          {/* <View style={{ height: 100, width: 100, backgroundColor: 'rgb(0, 255, 0)' }}/> */}
10         <View style={{ height: 100, width: 100, backgroundColor: randomRgb() }}/>
11       </View>
12     );
13   };
14
15  const randomRgb = () => {
16    const red = Math.floor(Math.random() * 256);
17    const green = Math.floor(Math.random() * 256);
18    const blue = Math.floor(Math.random() * 256);
19
20    return `rgb(${red}, ${green}, ${blue})`;
21    /* The grave accent (`) is a diacritical mark used in many written Languages. */
22  };
23
24
25  const styles = StyleSheet.create({
26    viewStyle: {
27      padding: 15
28    }
29  });
30
31  export default ColorScreen;

```

```

src > screens > JS ColorScreen.js > [x] styles
1  import React, { useState } from 'react';
2  import { View, Text, StyleSheet, Button } from 'react-native';
3
4  const ColorScreen = () => {
5    const [colors, setColors] = useState([]);
6    console.log(colors);
7    return (
8      <View style={ styles.viewStyle }>
9        <Text>ColorScreen</Text>
10       <Button title="Add a Color"
11         onPress={() => {
12           setColors([...colors, randomRgb()]);
13         }}
14       >
15         {/* <View style={{ height: 100, width: 100, backgroundColor: 'rgb(0, 255, 0)' }}/> */}
16         <View style={{ height: 100, width: 100, backgroundColor: randomRgb() }}/>
17       </View>
18     );
19   };

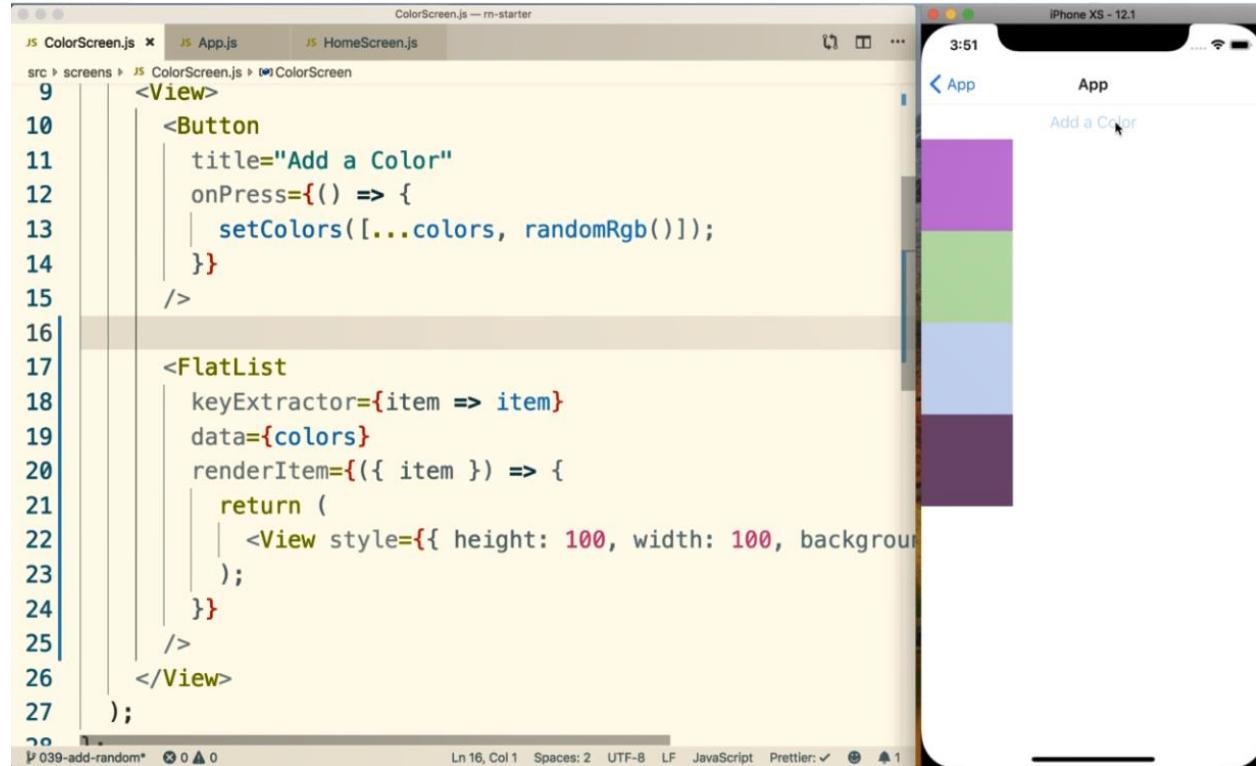
```

Press “Add a Color” button. Then change the color on view area.

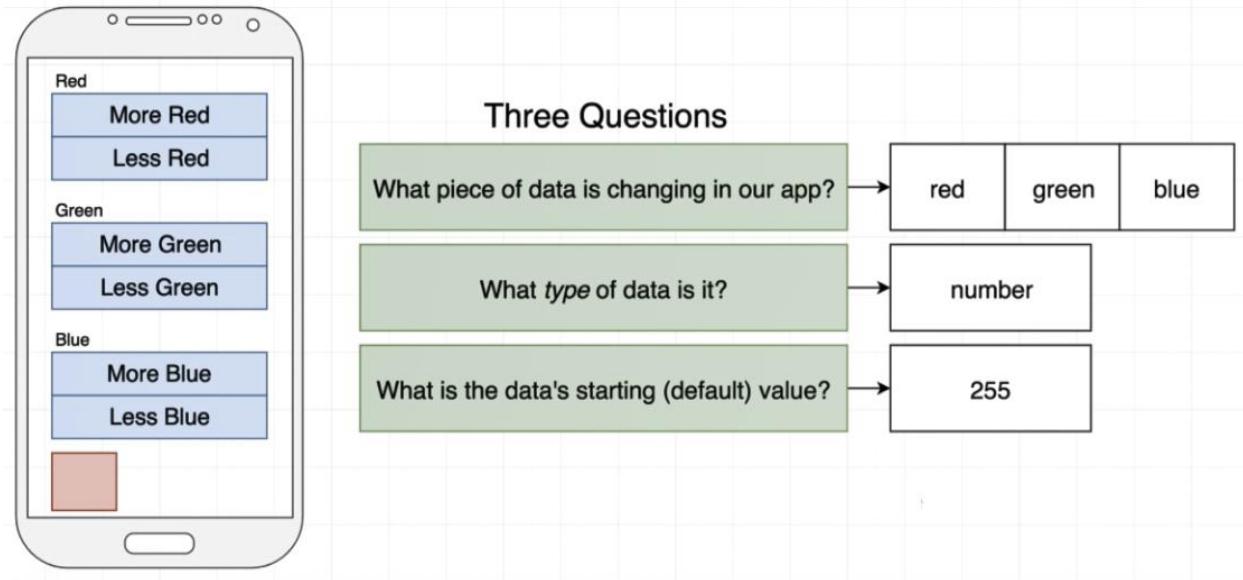
3. Showing Colors with a FlatList

```
src > screens > JS ColorScreen.js > [randomRgb]
1  import React, { useState } from 'react';
2  import { View, Text, StyleSheet, Button, FlatList } from 'react-native';
3
4  const ColorScreen = () => {
5      const [colors, setColors] = useState([]);
6      return (
7          <View style={ styles.viewStyle }>
8              <Text>ColorScreen</Text>
9              <Button title="Add a Color"
10                  onPress={() => {
11                      setColors([...colors, randomRgb()]);
12                  }}
13              />
14              {/* <View style={{ height: 100, width: 100, backgroundColor: 'rgb(0, 255, 0)' }}/> */}
15              <FlatList
16                  keyExtractor={item => item}
17                  data={colors}
18                  renderItem={({ item }) => {
19                      return <View style={{ height: 100, width: 100, backgroundColor: item }}/>;
20                  }}
21              />
22          </View>
23      );
24  };
--
```

We can see the colors in FlatList.



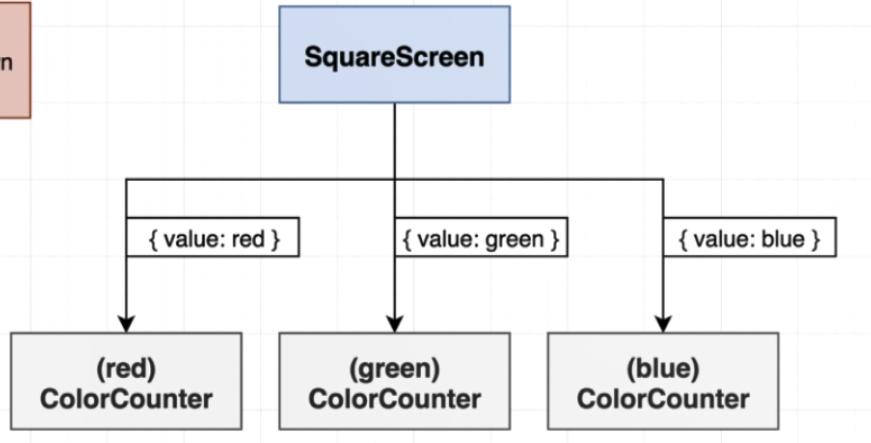
4. Reusable Color Adjusters



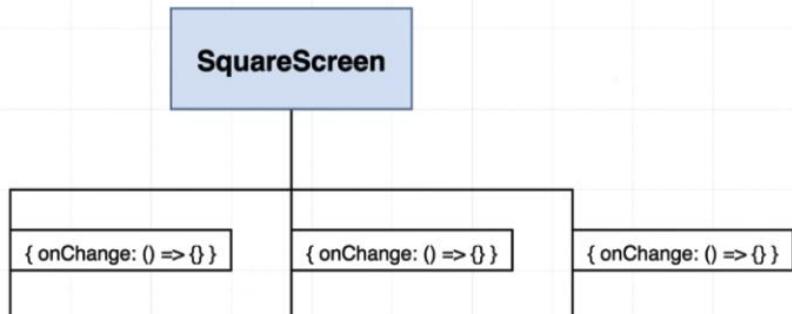
Create SquareScreen.js in screens directory and ColorCounter.js in components directory.

If a child needs to **read** a state value, the parent can pass it down as a prop

Note: ColorCounter doesn't need to read state values! This is just for your info



If a child needs to **change** a state value, the parent can pass down a **callback function to change the state value** as a prop



ColorCounter.js

```
JS SquareScreen.js ● JS ColorCounter.js X JS HomeScreen.js
src > components > JS ColorCounter.js > ...
1  import React, { useState } from 'react';
2  import { View, Button, Text, StyleSheet } from 'react-native';
3
4  const ColorCounter = ({ color, onIncrease, onDecrease }) => {
5      return (
6          <View>
7              <Text>{color}</Text>
8              <Button onPress={() => onIncrease()} title={`Increase ${color}`} />
9              <Button onPress={() => onDecrease()} title={`Decrease ${color}`} />
10         </View>
11     );
12 };
13
14 const styles = StyleSheet.create({});
15
16 export default ColorCounter;
```

SquareScreen.js

```
JS SquareScreen.js ● JS ColorCounter.js X JS HomeScreen.js
src > screens > JS SquareScreen.js > ...
1  import React, { useState } from 'react';
2  import { View, StyleSheet } from 'react-native';
3  import ColorCounter from '../components/ColorCounter';
4
5  const COLOR_INCREMENT = 15;
6
7  const SquareScreen = () => {
8      const [red, setRed] = useState(0);
9      const [green, setGreen] = useState(0);
10     const [blue, setBlue] = useState(0);
11     console.log(red);
12     return (
13         <View style={ styles.viewStyle }>
14             <ColorCounter
15                 color="Red"
16                 onIncrease={() => setRed(red + COLOR_INCREMENT)}
17                 onDecrease={() => setRed(red - COLOR_INCREMENT)}>
18             />
19             <ColorCounter
20                 color="Green"
21                 onIncrease={() => setGreen(green + COLOR_INCREMENT)}
22                 onDecrease={() => setGreen(green - COLOR_INCREMENT)}>
23             />
24             <ColorCounter
25                 color="Blue"
26                 onIncrease={() => setBlue(blue + COLOR_INCREMENT)}
27                 onDecrease={() => setBlue(blue - COLOR_INCREMENT)}>
28             />
29             <View style={{ height: 100, width: 100, backgroundColor: `rgb(${red}, ${green}, ${blue})` }}>
30         </View>
31     );
32 }
```

How to use if function in render part

```
return (
  <View style={ styles.viewStyle }>
    <ColorCounter
      color="Red"
      onIncrease={() => {
        if(red + COLOR_INCREMENT > 255){
          return;
        }
        setRed(red + COLOR_INCREMENT)
      }}
      onDecrease={() => setRed(red - COLOR_INCREMENT)}
    />
```

How to use switch case in React Native

```
const setColor = (color, change) => {
  switch (color) {
    case 'red':
      if (red + change > 255 || red + change < 0) {
        return;
      } else {
        setRed(red + change);
      }
      return;
    case 'green':
      if (green + change > 255 || green + change < 0) {
        return;
      } else {
        setRed(green + change);
      }
      return;
    case 'blue':
      if (blue + change > 255 || blue + change < 0) {
        return;
      } else {
        setRed(blue + change);
      }
      return;
  }
};
```

5. Validating State Changes

```
5  const COLOR_INCREMENT = 15;
6  const SquareScreen = () => {
7    const [red, setRed] = useState(0);
8    const [green, setGreen] = useState(0);
9    const [blue, setBlue] = useState(0);
10   //Validator
11   const setColor = (color, change) => {
12     switch (color) {
13       case 'red':
14         red + change > 255 || red + change < 0 ? null : setRed(red + change);
15         return;
16       case 'green':
17         green + change > 255 || green + change < 0 ? null : setGreen(green + change);
18         return;
19       case 'blue':
20         blue + change > 255 || blue + change < 0 ? null : setBlue(blue + change);
21         return;
22       default:
23         return;
24     }
25   };

```

```
26   return (
27     <View style={styles.viewStyle}>
28       <ColorCounter
29         color="Red"
30         onIncrease={() => setColor('red', COLOR_INCREMENT)}
31         onDecrease={() => setColor('red', -1 * COLOR_INCREMENT)}
32       />
33       <ColorCounter
34         color="Green"
35         onIncrease={() => setColor('green', COLOR_INCREMENT)}
36         onDecrease={() => setColor('green', -1 * COLOR_INCREMENT)}
37       />
38       <ColorCounter
39         color="Blue"
40         onIncrease={() => setColor('blue', COLOR_INCREMENT)}
41         onDecrease={() => setColor('blue', -1 * COLOR_INCREMENT)}
42       />
43       <View style={{ height: 100, width: 100, backgroundColor: `rgb(${red}, ${green}, ${blue})` }} />
44     </View>
45   );
46 };

```

One line if statement :

```
red + change > 255 || red + change < 0 ? null : setRed(red + change);
```

Quick Thoughts

App works right now - we could leave it as-is! But we could make it *slightly better...*

We have three *separate* pieces of state

For this app, these three pieces of state are extremely related

There is a precise set of well-known ways in which we update these values

This makes our state a great candidate for being managed by a '*reducer*'

FunctionThatManagesChangesTOAnObject ?!?!?

Real fancy name

Function that gets called with two objects

Argument #1 - object that has *all of our state in it*

```
const state = { red: 0, green: 0, blue: 0 }  
state.red = state.red + 10
```

Argument #2 - object that describes the update we want to make

```
{ colorToChange: 'red', amount: 15 }
```

We look at Argument #2 and use it to decide how to change Argument #1

Two technicalities - (1) We never change Argument #1 *directly*. (2) We must always return a value to be used as Argument #1

Note :

6. Creating a Reducer

```
JS SquareScreen.js X JS ColorCounter.js JS HomeScreen.js
src > screens > JS SquareScreen.js > [e] SquareScreen
1 √ import React, { useReducer } from 'react';
2   import { View, StyleSheet } from 'react-native';
3   import ColorCounter from '../components/ColorCounter';
4
5   const COLOR_INCREMENT = 15;
6
7   const reducer = (state, action) => {
8     //state === { red: number, green: number, blue: number }
9     //action === { colorToChange: 'red' || 'green' || 'blue', amount: 15 || -15 }
10
11  switch (action.colorToChange) {
12    case 'red':
13      return { ...state, red: state.red + action.amount };
14    case 'green':
15      return { ...state, green: state.green + action.amount };
16    case 'blue':
17      return { ...state, blue: state.blue + action.amount };
18    default:
19      return state;
20  }
21
22};

23
```

```
24 const SquareScreen = () => {
25   const [state, dispatch] = useReducer(reducer, { red: 0, green: 0, blue: 0 });
26   const { red, green, blue } = state;
27   return (
28     <View style={styles.viewStyle}>
29       <ColorCounter
30         color="Red"
31         onIncrease={() => dispatch({ colorToChange: 'red', amount: COLOR_INCREMENT })}
32         onDecrease={() => dispatch({ colorToChange: 'red', amount: -1 * COLOR_INCREMENT })}
33       />
34       <ColorCounter
35         color="Green"
36         onIncrease={() => dispatch({ colorToChange: 'green', amount: COLOR_INCREMENT })}
37         onDecrease={() => dispatch({ colorToChange: 'green', amount: -1 * COLOR_INCREMENT })}
38       />
39       <ColorCounter
40         color="Blue"
41         onIncrease={() => dispatch({ colorToChange: 'blue', amount: COLOR_INCREMENT })}
42         onDecrease={() => dispatch({ colorToChange: 'blue', amount: -1 * COLOR_INCREMENT })}
43       />
44       <View style={{ height: 100, width: 100, backgroundColor: `rgb(${state.red}, ${state.green}, ${state.blue})` }}>
45     </View>
46   );
47 };

48
```

7. Restoring Validation

```
7  const reducer = (state, action) => {
8    //state === { red: number, green: number, blue: number }
9    //action === { colorToChange: 'red' || 'green' || 'blue', amount: 15 || -15 }
10
11   switch (action.colorToChange) {
12     case 'red':
13       return state.red + action.amount > 255 || state.red + action.amount < 0
14       ? state
15       : { ...state, red: state.red + action.amount };
16     case 'green':
17       return state.green + action.amount > 255 || state.green + action.amount < 0
18       ? state
19       : { ...state, green: state.green + action.amount };
20     case 'blue':
21       return state.blue + action.amount > 255 || state.blue + action.amount < 0
22       ? state
23       : { ...state, blue: state.blue + action.amount };
24     default:
25       return state;
26   }
27
28 };
29
```

8. Community Convention in Reducers

Our Action Object Had..

{ colorToChange: 'red', amount: 15 }

Usually, by convention, we'll instead use:

{ type: 'change_red', payload: 15 }

type

String that describes the exact change operation we want to make

payload

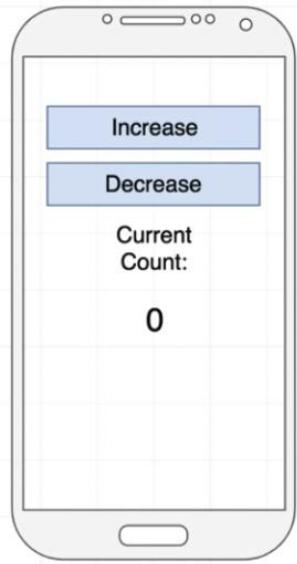
Some data that is critical to the change operation

After change Community Convention type.

```
src > screens > JS SquareScreen.js > [x] SquareScreen
1  import React, { useReducer } from 'react';
2  import { View, StyleSheet } from 'react-native';
3  import ColorCounter from '../components/ColorCounter';
4
5  const COLOR_INCREMENT = 15;
6
7  const reducer = (state, action) => {
8      //state === { red: number, green: number, blue: number }
9      //action === { type: 'change_red' || 'change_green' || 'change_blue', payload: 15 || -15 }
10
11     switch (action.type) {
12         case 'change_red':
13             return state.red + action.payload > 255 || state.red + action.payload < 0
14             ? state
15             : { ...state, red: state.red + action.payload };
16         case 'change_green':
17             return state.green + action.payload > 255 || state.green + action.payload < 0
18             ? state
19             : { ...state, green: state.green + action.payload };
20         case 'change_blue':
21             return state.blue + action.payload > 255 || state.blue + action.payload < 0
22             ? state
23             : { ...state, blue: state.blue + action.payload };
24         default:
25             return state;
26     }
27 }
28};
```

```
30  const SquareScreen = () => {
31      const [state, dispatch] = useReducer(reducer, { red: 0, green: 0, blue: 0 });
32      const { red, green, blue } = state;
33      return (
34          <View style={styles.viewStyle}>
35              <ColorCounter
36                  color="Red"
37                  onIncrease={() => dispatch({ type: 'change_red', payload: COLOR_INCREMENT })}
38                  onDecrease={() => dispatch({ type: 'change_red', payload: -1 * COLOR_INCREMENT })}>
39          />
40          <ColorCounter
41                  color="Green"
42                  onIncrease={() => dispatch({ type: 'change_green', payload: COLOR_INCREMENT })}
43                  onDecrease={() => dispatch({ type: 'change_green', payload: -1 * COLOR_INCREMENT })}>
44          />
45          <ColorCounter
46                  color="Blue"
47                  onIncrease={() => dispatch({ type: 'change_blue', payload: COLOR_INCREMENT })}
48                  onDecrease={() => dispatch({ type: 'change_blue', payload: -1 * COLOR_INCREMENT })}>
49          />
50          <View style={{ height: 100, width: 100, backgroundColor: `rgb(${state.red}, ${state.green}, ${state.blue})` }}>
51          </View>
52      );
53  };
```

9. Exercise Solution



Change the CounterScreen to manage its state with
'useReducer' instead of 'useState'

You won't have any more references to 'useState'! Delete
them all!

```
src > screens > JS CounterScreen.js > ...
1  import React, { useReducer } from 'react';
2  import { Text, View, StyleSheet, Button } from 'react-native';
3
4  const reducer = (state, action) => {
5    // state === { count: number }
6    // action === { type: 'increment' || 'decrement', payload: 1 }
7    switch (action.type) {
8      case 'increment':
9        return { ...state, count: state.count + action.payload };
10     case 'decrement':
11        return { ...state, count: state.count - action.payload };
12     default:
13        return state;
14    }
15  };
16
17 const CounterScreen = () => {
18  const [state, dispatch] = useReducer(reducer, { count: 0 });
19  //dispatch = callMyReducer
20
21  return (
22    <View style={styles.viewStyle}>
23      <Button title="Increase" onPress={() => {
24        dispatch({ type: 'increment', payload: 1 })
25      }} />
26      <Button title="decrease" onPress={() => {
27        dispatch({ type: 'decrement', payload: 1 })
28      }} />
29      <Text>Current Count: {state.count}</Text>
30    </View>
31  );
32};
```

10. Handling Text Input



Show a text input

When a user types in the input, show the same text immediately underneath the input



Validate the user's input, show a message if it doesn't meet some criteria

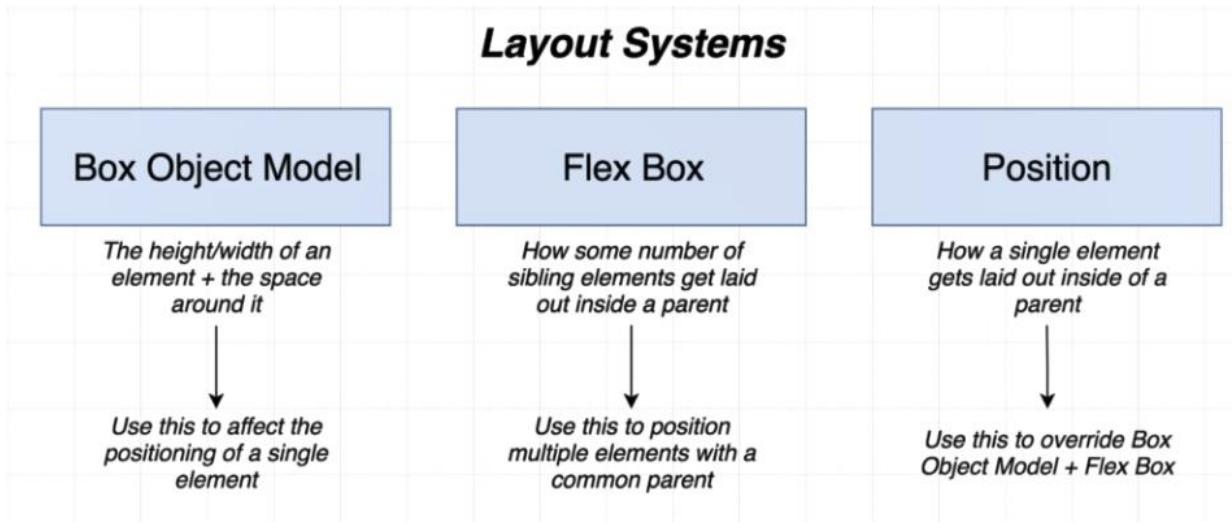
When a user types in the input, show the same text immediately underneath the input

JS TextScreen.js X

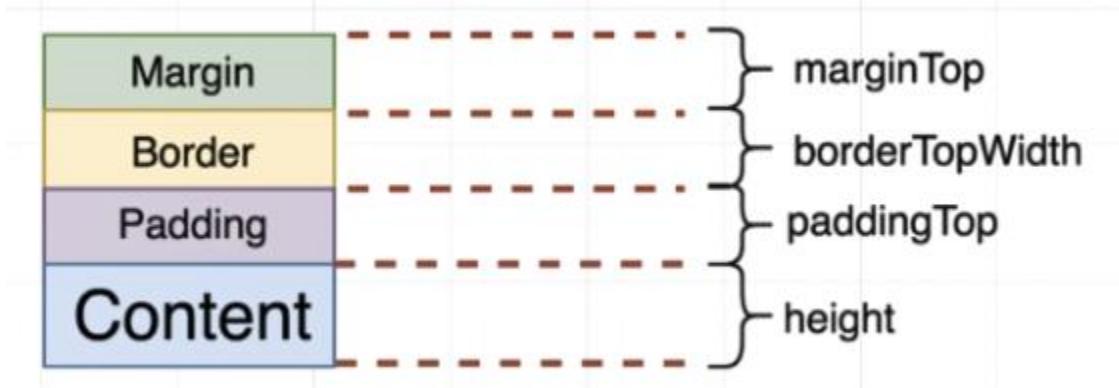
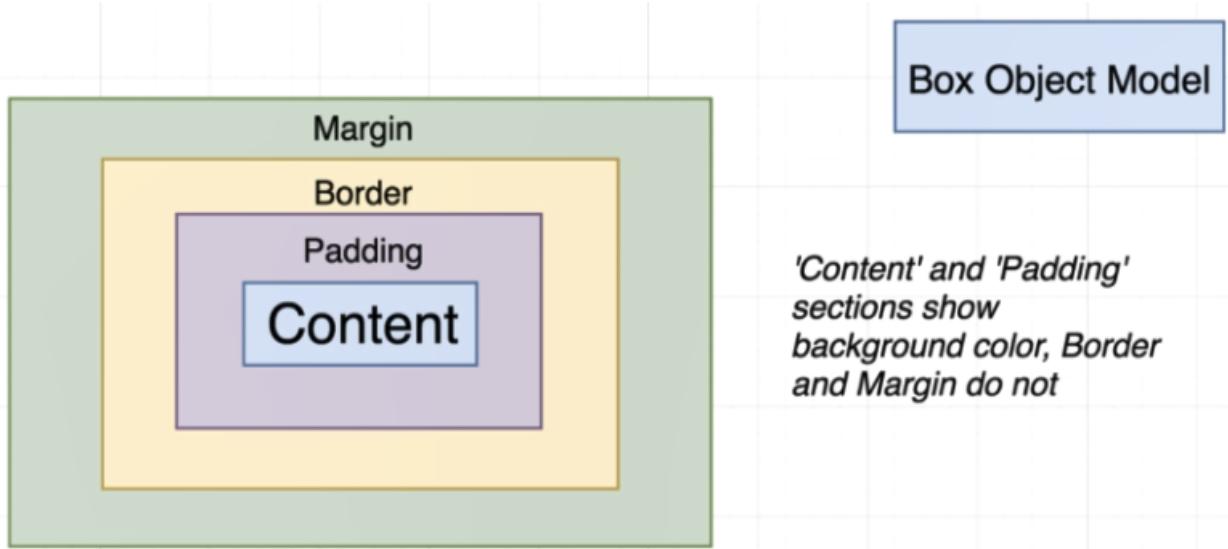
```
src > screens > JS TextScreen.js > ...
1  import React,{ useState } from 'react';
2  import { View, Text, StyleSheet, TextInput } from 'react-native';
3
4  const TextScreen = () => {
5      const [name, setName] = useState('');
6      return (
7          <View style={styles.viewStyle}>
8              <Text>Enter Name: </Text>
9              <TextInput
10                  style={styles.input}
11                  autoCapitalize="none"
12                  autoCorrect={false}
13                  value={name}
14                  onChangeText={(newValue) => setName(newValue)}
15              />
16              <Text>My name is {name}</Text>
17          </View>
18      );
19  }
20
21  const styles = StyleSheet.create({
22      input: {
23          marginVertical: 15,
24          borderColor: 'black',
25          borderWidth: 1
26      },
27      viewStyle: {
28          margin: 15
29      }
30  });
31
32  export default TextScreen;
```

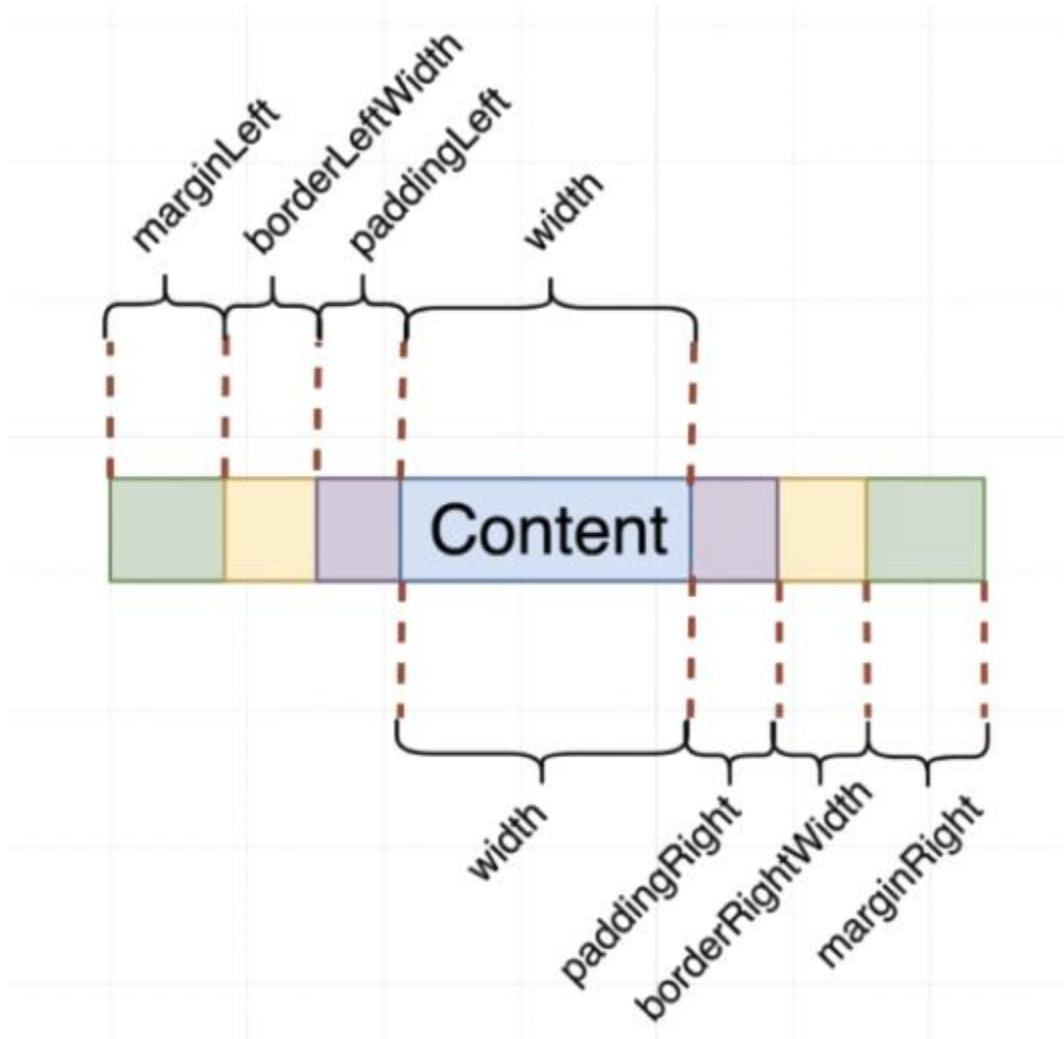
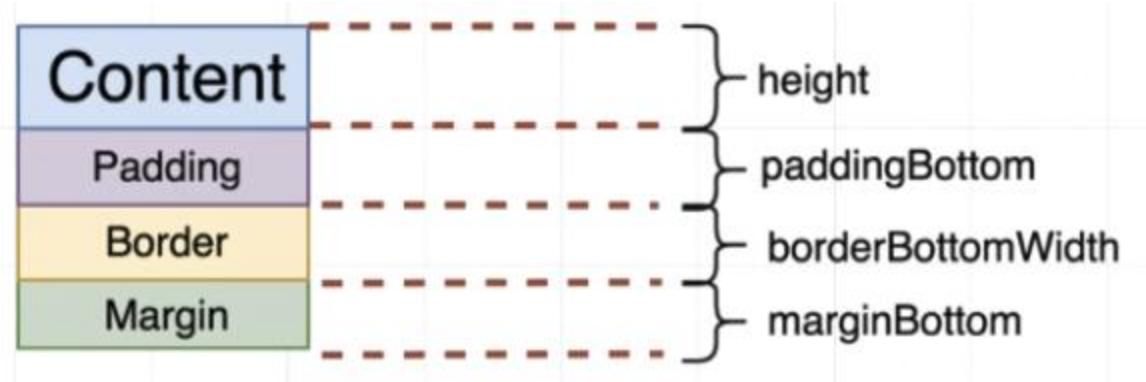
```
JS TextScreen.js ●
src > screens > JS TextScreen.js > ...
1  import React,{ useState } from 'react';
2  import { View, Text, StyleSheet, TextInput } from 'react-native';
3
4  const TextScreen = () => {
5      const [password, setPassword] = useState('');
6      return (
7          <View style={styles.viewStyle}>
8              <Text>Enter password: </Text>
9              <TextInput
10                  style={styles.input}
11                  autoCapitalize="none"
12                  autoCorrect={false}
13                  value={password}
14                  onChangeText={(newValue) => setPassword(newValue)}
15              />
16              {password.length < 4 ? <Text>Password must be 4 characters</Text> : null}
17          </View>
18      );
19  }
20
21  const styles = StyleSheet.create({
22      input: {
23          marginVertical: 15,
24          borderColor: 'black',
25          borderWidth: 1
26      },
27      viewStyle: {
28          margin: 15
29      }
30  });
31
32  export default TextScreen;
```

7. How to Handle Screen Layout



1. Basics of Box Object Model





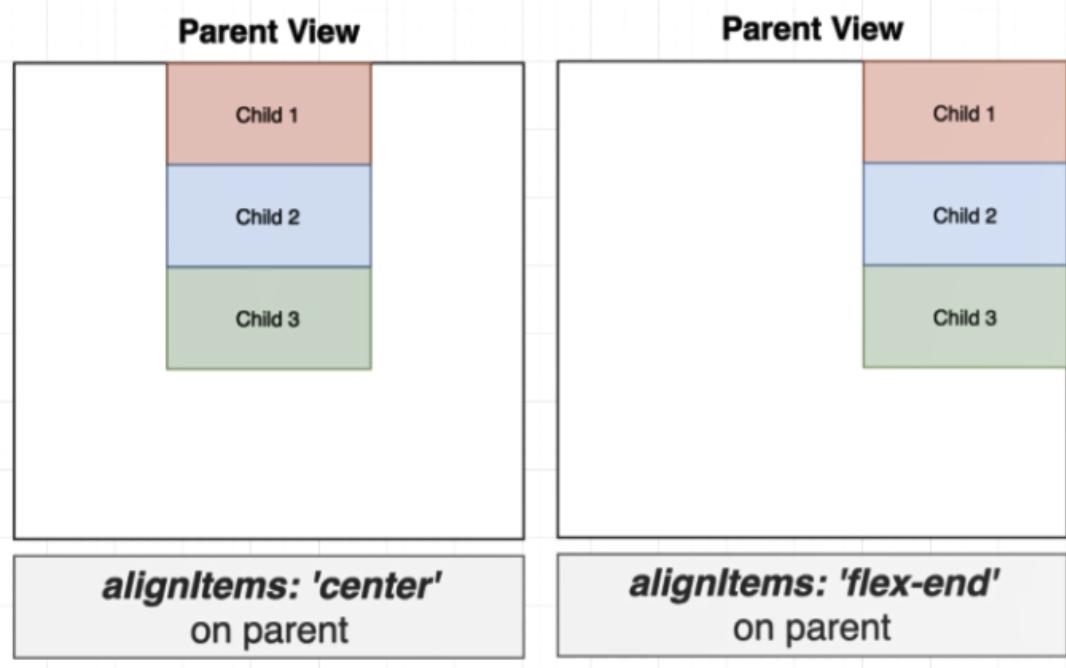
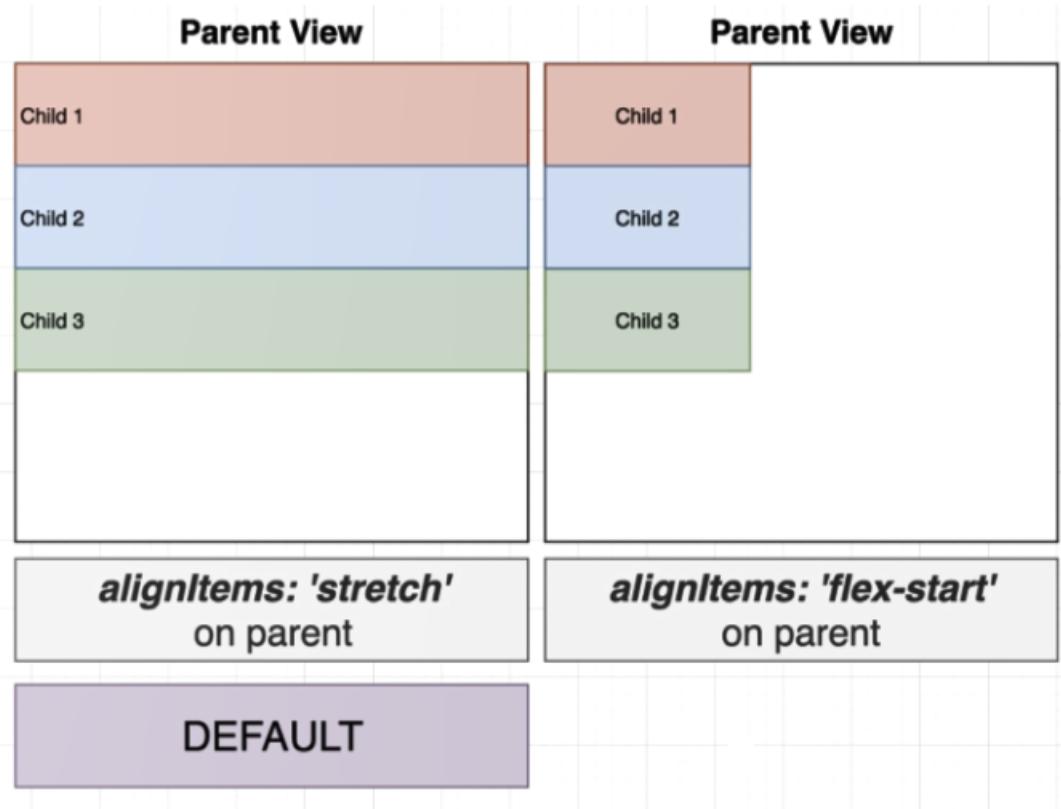
Shortcuts

margin	Set the margin on all sides
marginVertical	Set the margin on top and bottom
marginHorizontal	Set the margin on left and right
padding	Set the padding on all sides
paddingVertical	Set the padding on top and bottom
paddingHorizontal	Set the padding on left and right
borderWidth	Set border width on all sides

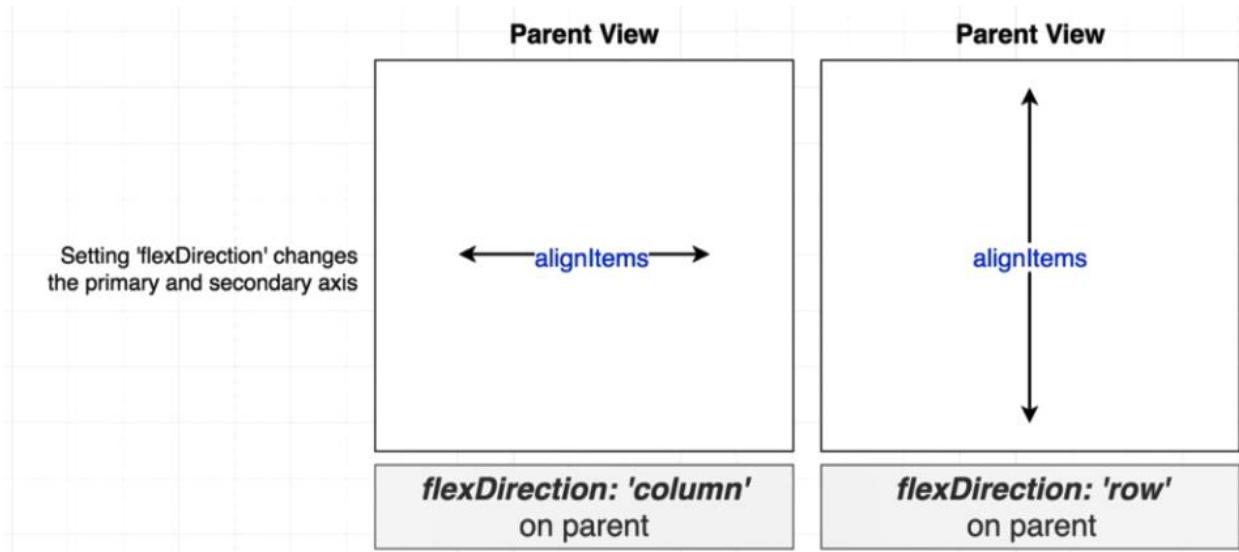
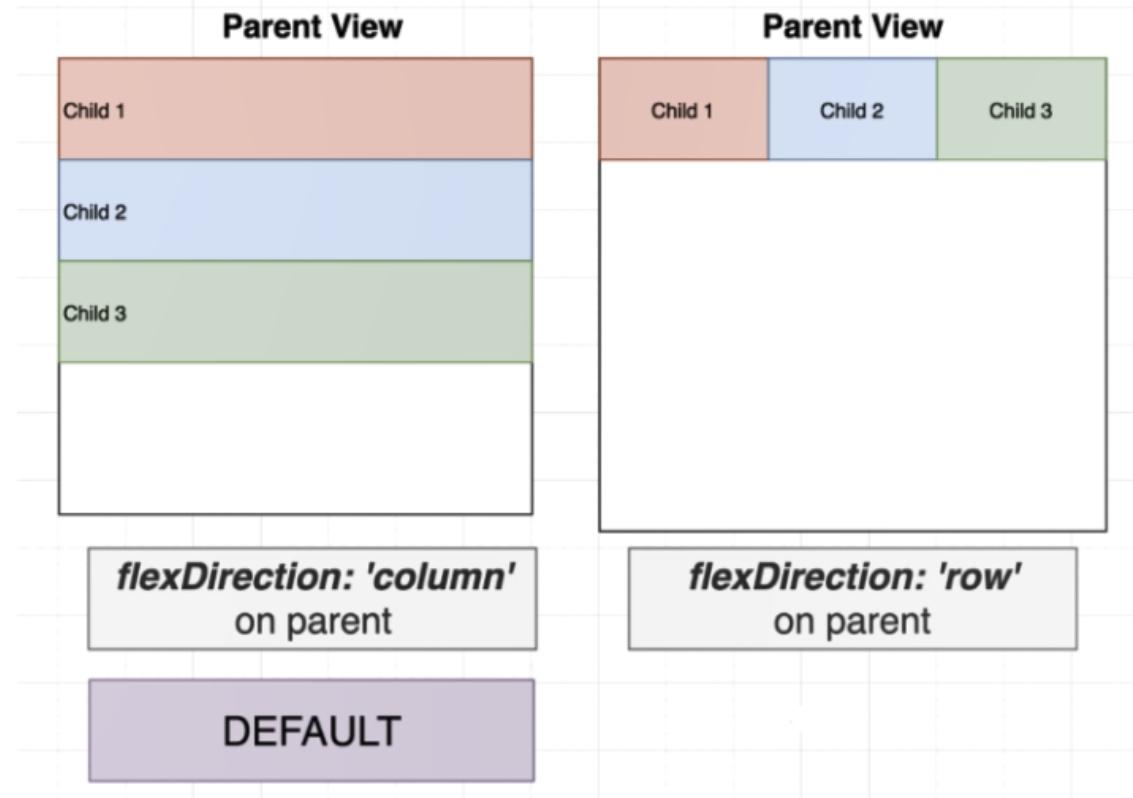
JS BoxScreen.js •

```
src > screens > JS BoxScreen.js > ...
1  import React from 'react';
2  import { Text, View, StyleSheet } from 'react-native';
3
4  const BoxScreen = () => {
5    return (
6      <View style={styles.viewStyle}>
7        <Text style={styles.textStyle}>BoxScreen</Text>
8      </View>
9    );
10 };
11
12 const styles = StyleSheet.create({
13   viewStyle: {
14     borderWidth: 1,
15     borderColor: 'black'
16   },
17   textStyle: {
18     borderWidth: 1,
19     borderColor: 'red',
20     margin: 70
21   }
22 });
23
24 export default BoxScreen;
```

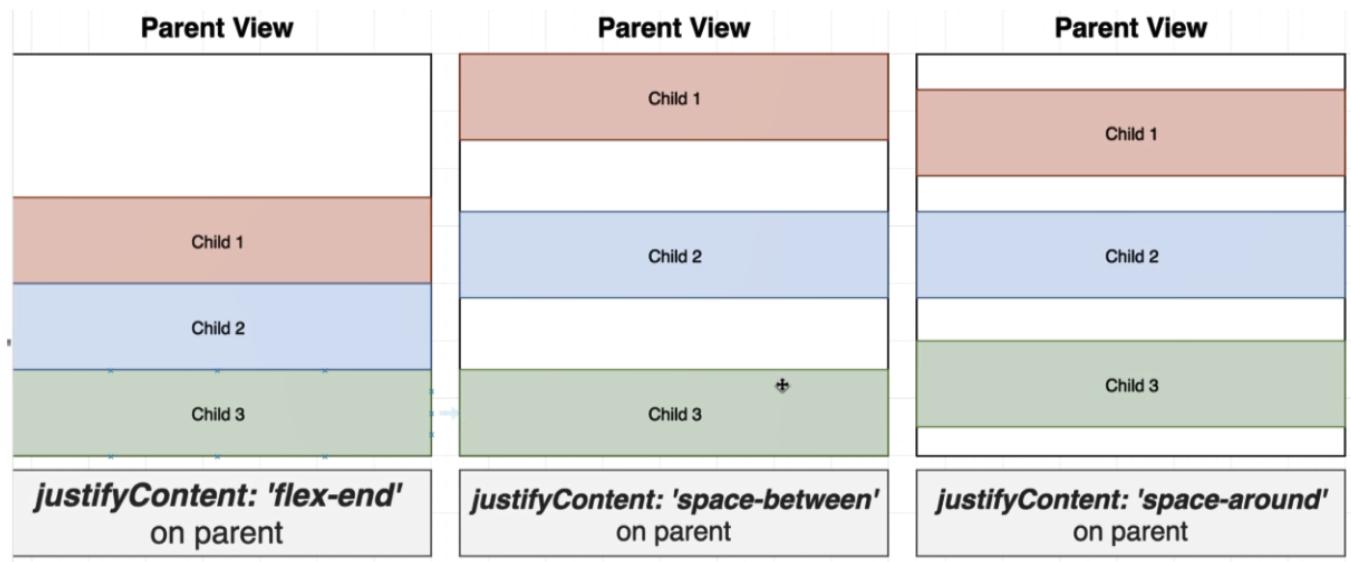
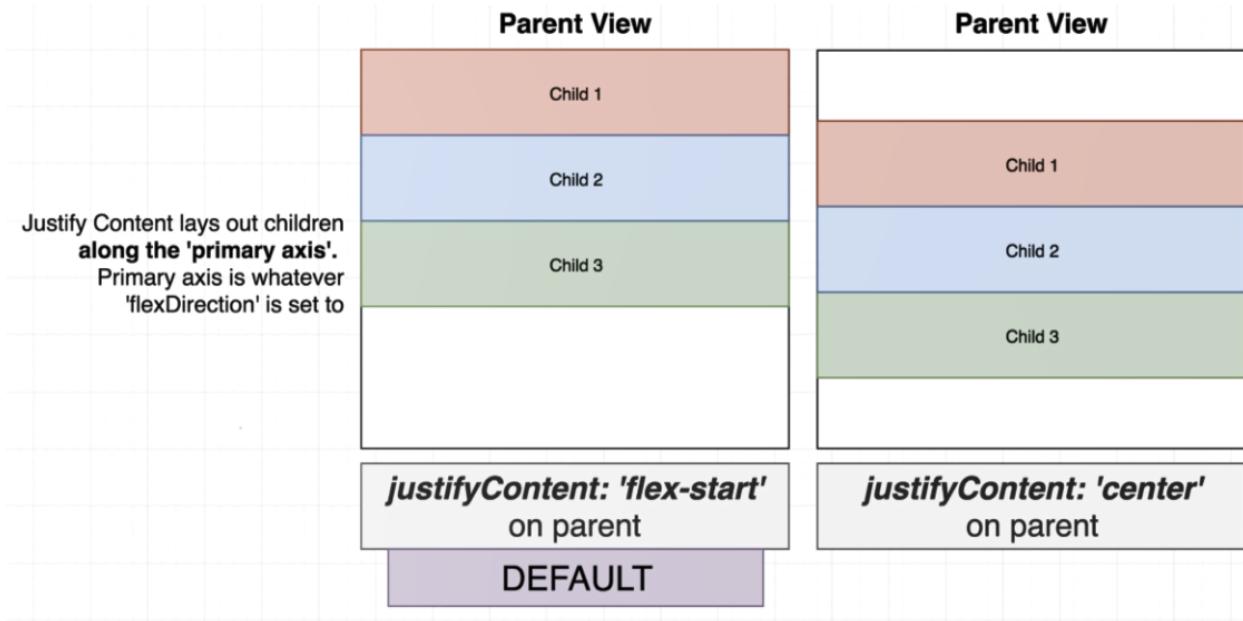
2. Align Items with Flex



3. Flex Direction



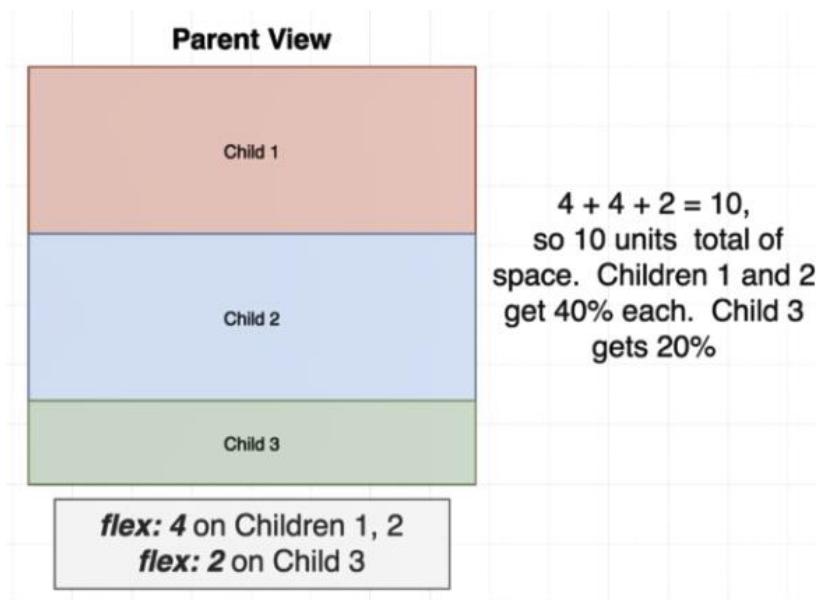
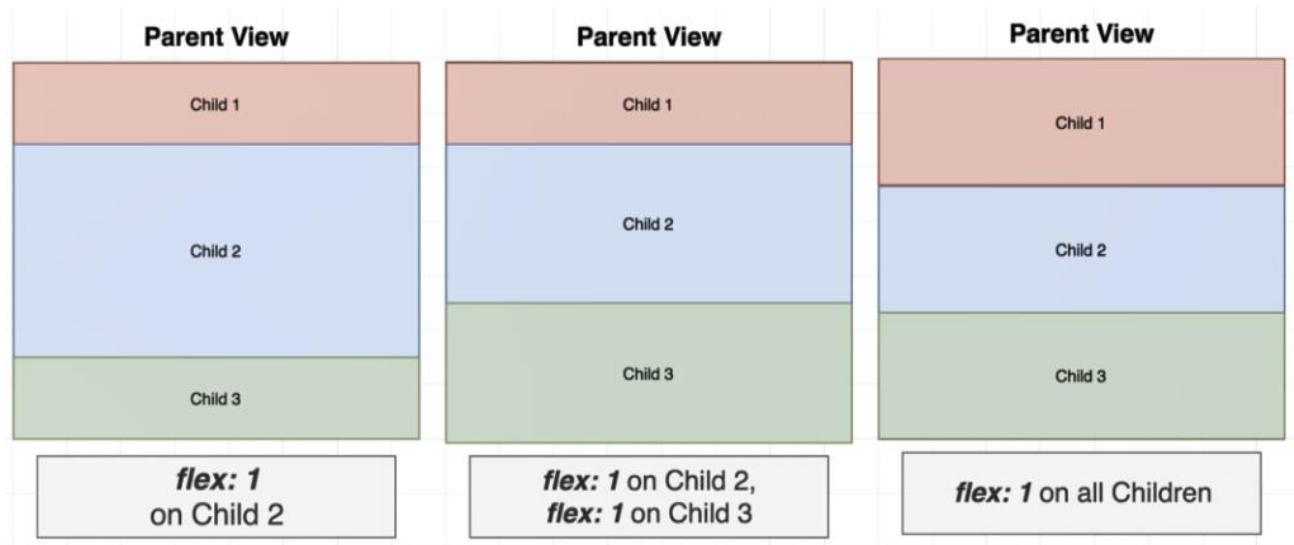
4. Justify Content



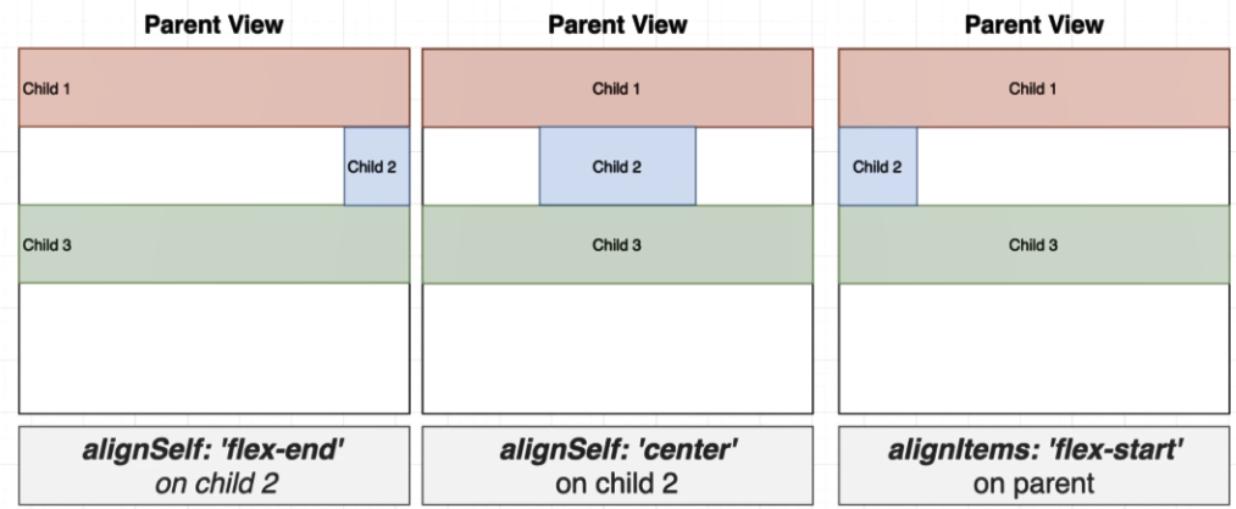
5. Flex Values

Parent	Child
alignItems	flex
justifyContent	alignSelf
flexDirection	

Flex makes a child in a parent try to take up as much space as possible.



6. Align Self on Children

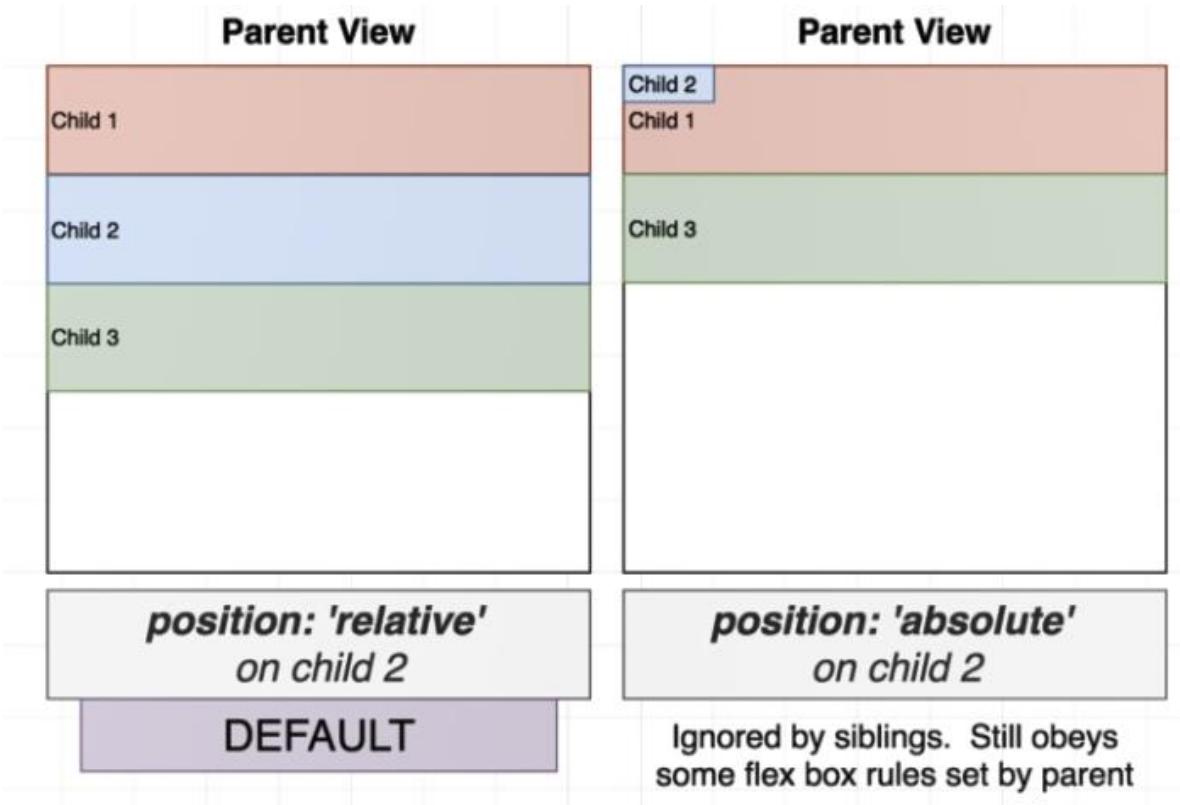


A screenshot of a code editor showing a file named `BoxScreen.js`. The code defines a component with the following styles:

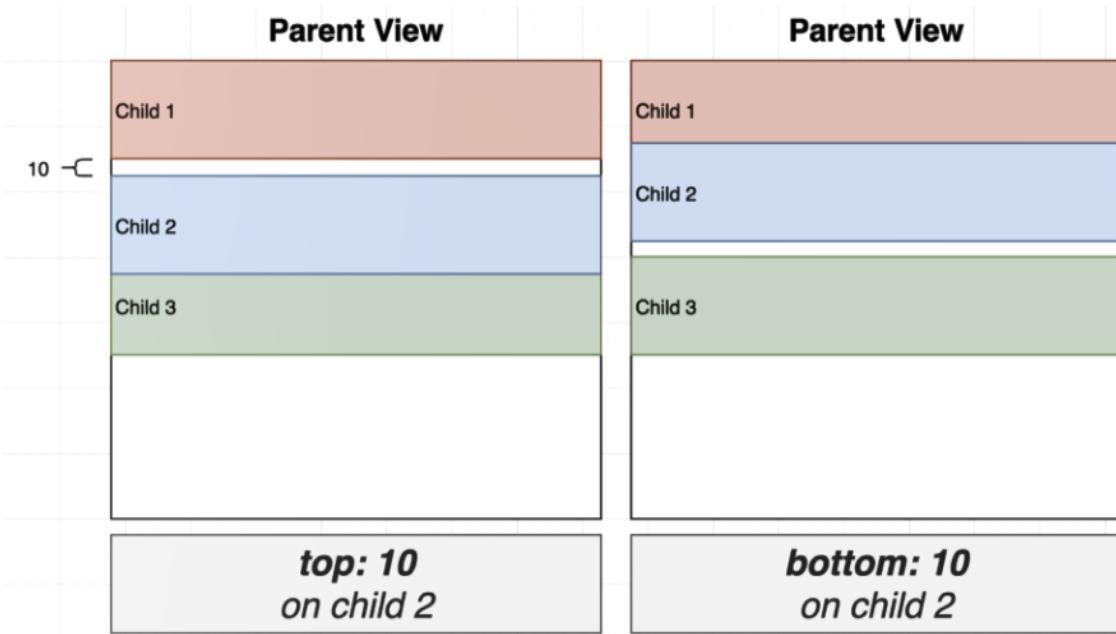
```
JS BoxScreen.js x
src > screens > JS BoxScreen.js > styles
17 |   borderColor: 'black',
18 |   height: 200,
19 |   alignItems: 'center'
20 | },
21 |   textOneStyle: {
22 |     borderWidth: 3,
23 |     borderColor: 'red'
24 | },
25 |   textTwoStyle: {
26 |     borderWidth: 3,
27 |     borderColor: 'red',
28 |     alignSelf: 'stretch'
29 | },
30 |   textThreeStyle: {
31 |     borderWidth: 3,
32 |     borderColor: 'red'
33 | }
34 });
35 
```

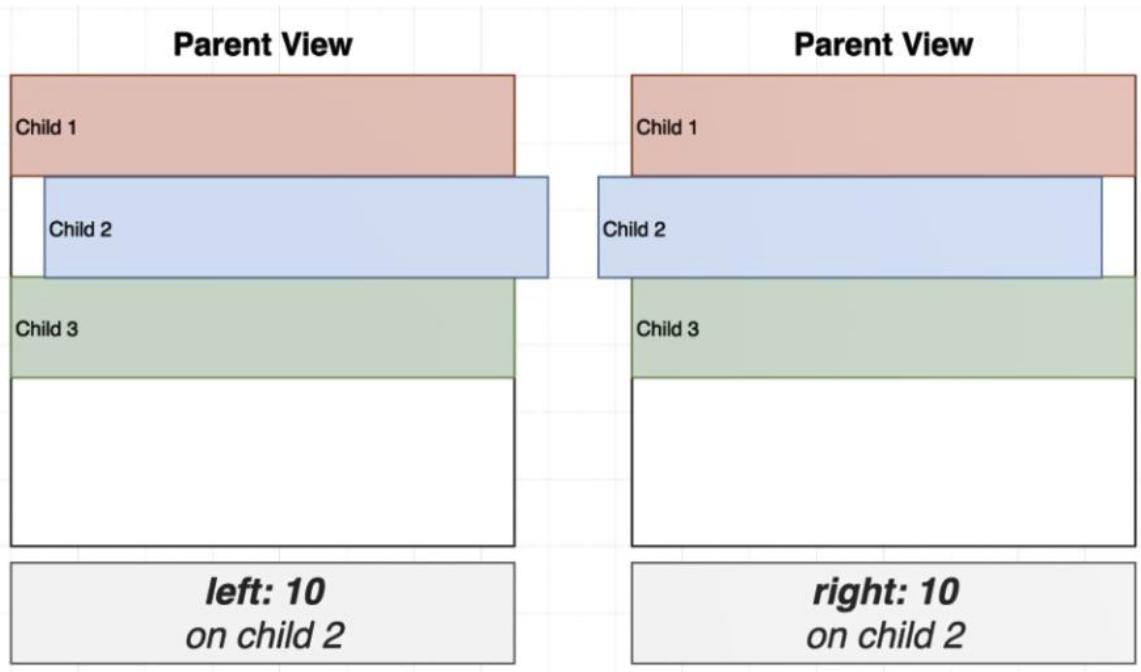
The `alignItems: 'center'` style is applied to the parent view. The `alignSelf: 'stretch'` style is applied to the second child, causing it to expand vertically to fill the parent's height. The code editor shows line numbers 17 through 35, and the status bar indicates "Ln 26, Col 20". To the right, an iPhone X simulator displays the resulting UI with three red-bordered boxes labeled "Child #1", "Child #2", and "Child #3" arranged vertically.

7. The Position Property



8. Top, Bottom, Left, Right





9. Absolute Fill Objects

The screenshot shows a code editor window for `BoxScreen.js` and an adjacent iPhone X simulator window.

Code Editor (BoxScreen.js):

```

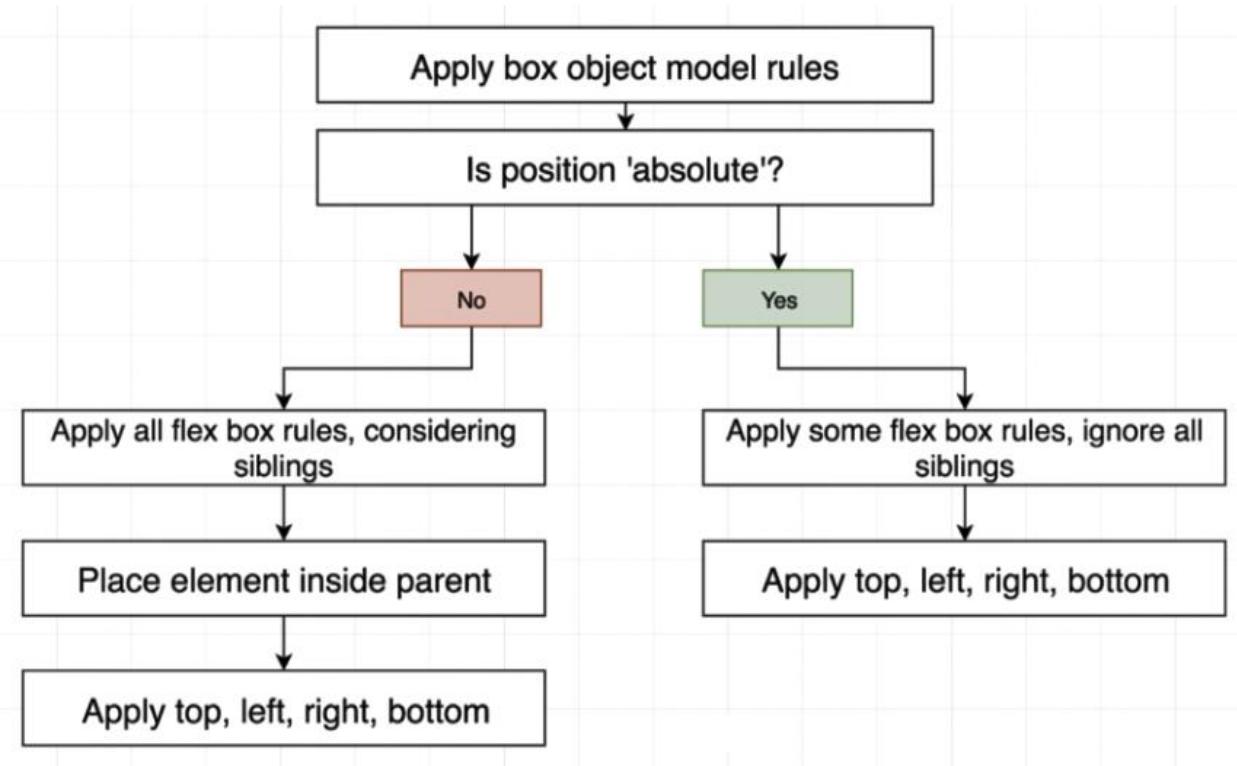
JS BoxScreen.js x
src > screens > JS BoxScreen.js > styles
20 |   textOneStyle: {
21 |     borderWidth: 3,
22 |     borderColor: 'red'
23 |   },
24 |   textTwoStyle: {
25 |     borderWidth: 3,
26 |     borderColor: 'red',
27 |     fontSize: 18,
28 |     position: 'absolute',
29 |     top: 0,
30 |     right: 0,
31 |     bottom: 0,
32 |     left: 0
33 |   },
34 |   textThreeStyle: {
35 |     borderWidth: 3,
36 |     borderColor: 'red'
37 |   }
38 });

```

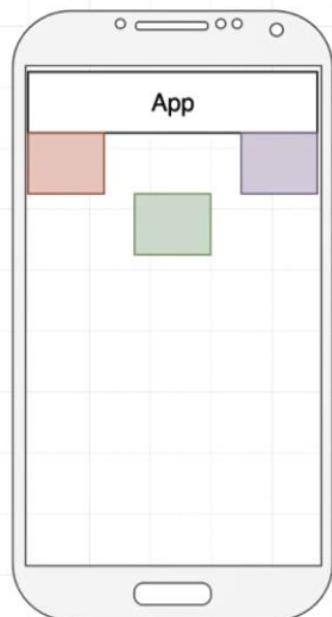
iPhone X Simulator:

The simulator displays a vertical stack of three red-bordered boxes labeled 'Child #2', 'Child #3', and 'Child #4'. The middle box ('Child #3') has a white center area.

10. Applying Layout Systems



11. Exercise Solution



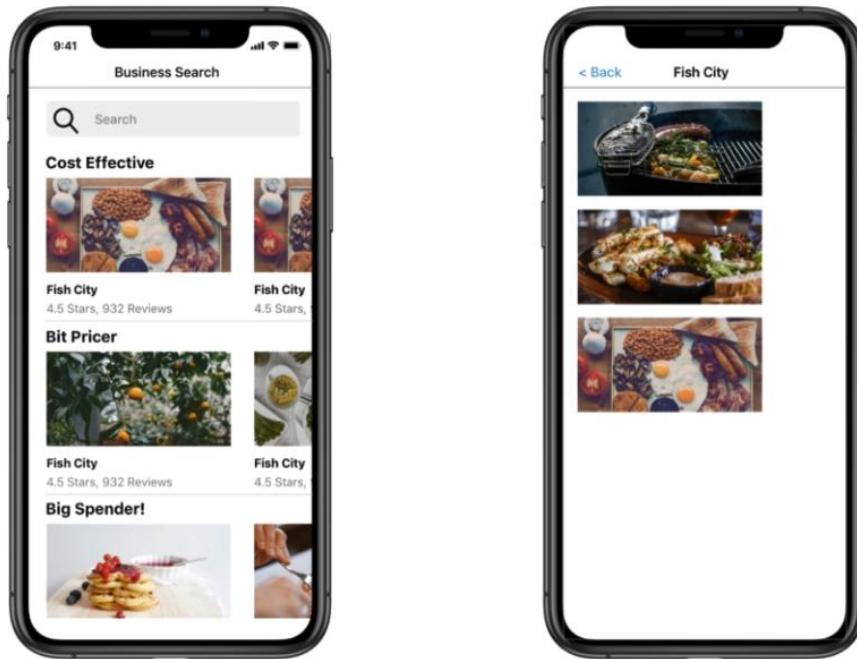
- Show three different view elements with the given layout
- Set the 'height' and 'width' properties of the view elements, otherwise they will collapse and be invisible!
- Try to get that green element into its position using two different techniques

JS BoxScreen.js X

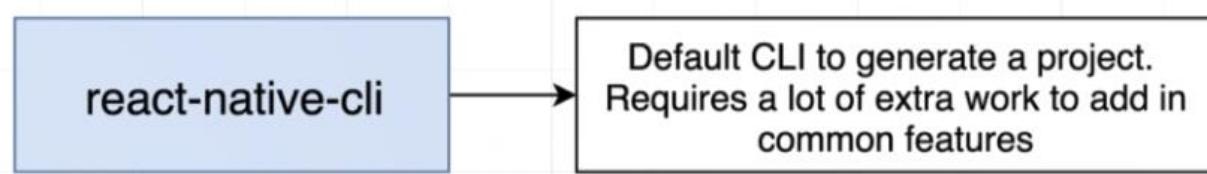
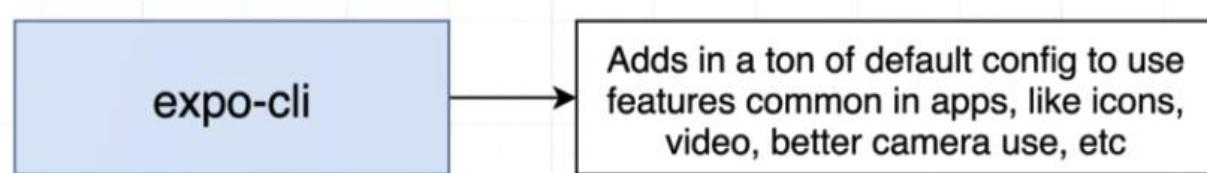
```
src > screens > JS BoxScreen.js > [x] styles
1  import React from 'react';
2  import { View, StyleSheet } from 'react-native';
3
4  const BoxScreen = () => {
5      return (
6          <View style={styles.parentStyle}>
7              <View style={styles.viewOneStyle} />
8              <View style={styles.viewTwoStyle} />
9              <View style={styles.viewThreeStyle} />
10         </View>
11     );
12 }
```

```
14  const styles = StyleSheet.create({
15      parentStyle: {
16          borderWidth: 3,
17          borderColor: 'black',
18          height: 100,
19          flexDirection: 'row',
20          justifyContent: 'space-between'
21      },
22      viewOneStyle: {
23          height: 50,
24          width: 50,
25          backgroundColor: 'red'
26      },
27      viewTwoStyle: {
28          height: 50,
29          width: 50,
30          backgroundColor: 'blue',
31          // marginTop: 50
32          alignSelf: 'flex-end'
33          // top: 50
34      },
35      viewThreeStyle: {
36          height: 50,
37          width: 50,
38          backgroundColor: 'green'
39      }
40  });
41 });
42
43 export default BoxScreen;
```

8. Putting It All Together - Restaurant Search App



1. Project Generation



```
npx expo-cli init food
```

Creates a new project called 'food'

2. React Navigation



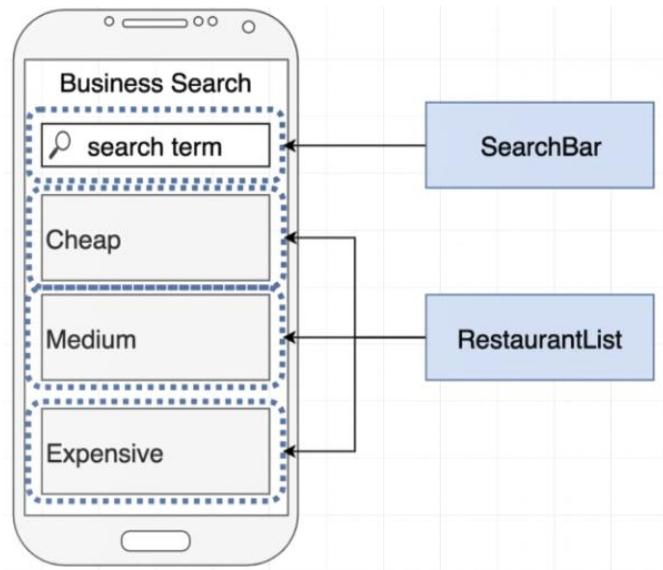
How to install React Navigation: **npm install react-navigation**

React Navigation requires two additional dependencies to work correctly. Be sure to also run this command in your terminal:

npx expo-cli install react-native-gesture-handler react-native-reanimated

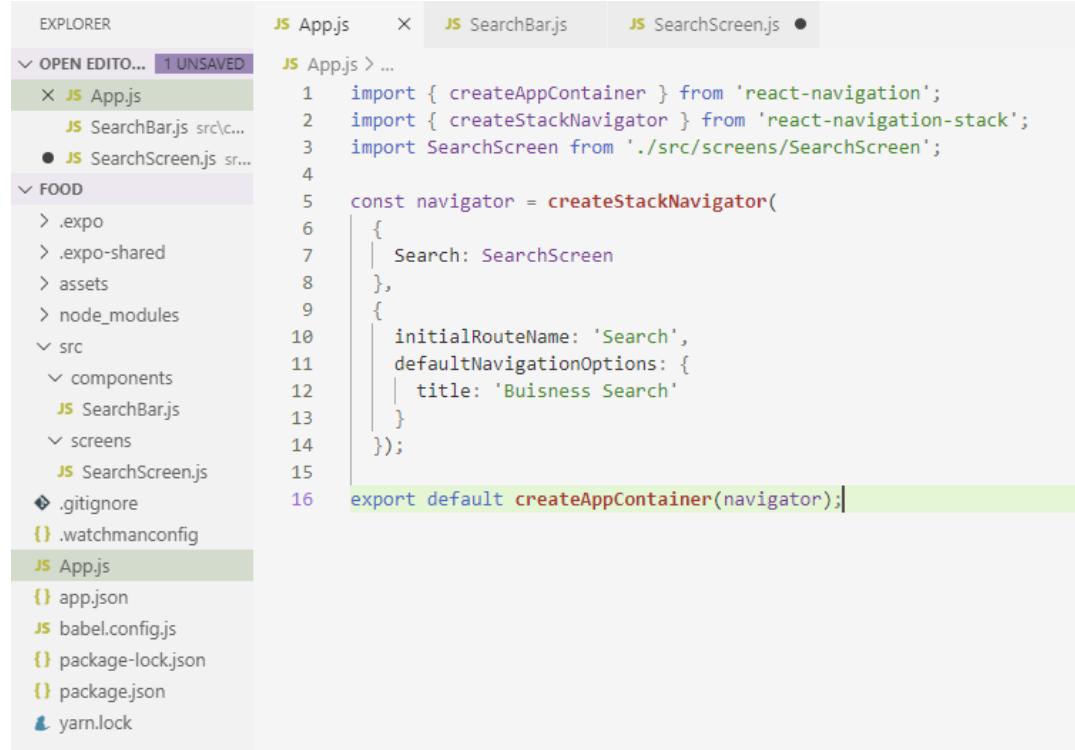
yarn add react-navigation-stack

<https://reactnavigation.org/docs/en/stack-navigator.html>



3. Starting the SearchBar

App.js



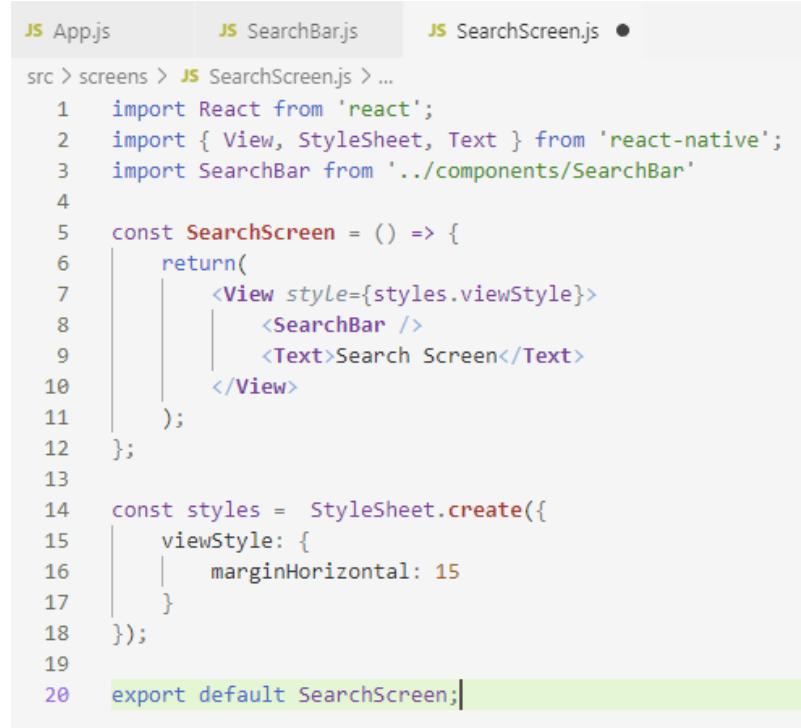
```
JS App.js      X JS SearchBar.js    JS SearchScreen.js ●

JS App.js > ...
1 import { createAppContainer } from 'react-navigation';
2 import { createStackNavigator } from 'react-navigation-stack';
3 import SearchScreen from './src/screens/SearchScreen';
4
5 const navigator = createStackNavigator(
6   {
7     Search: SearchScreen
8   },
9   {
10    initialRouteName: 'Search',
11    defaultNavigationOptions: {
12      title: 'Buisness Search'
13    }
14  });
15
16 export default createAppContainer(navigator);
```

EXPLORER

- OPEN EDITOR... | 1 UNSAVED
 - X JS App.js
 - JS SearchBar.js src\c...
 - JS SearchScreen.js sr...
- FOOD
 - > .expo
 - > .expo-shared
 - > assets
 - > node_modules
 - ✓ src
 - ✓ components
 - JS SearchBar.js
 - ✓ screens
 - JS SearchScreen.js
 - ❖ .gitignore
 - { .watchmanconfig
 - JS App.js
 - { app.json
 - JS babel.config.js
 - { package-lock.json
 - { package.json
 - ↳ yarn.lock

SearchScreen.js



```
JS App.js      JS SearchBar.js    JS SearchScreen.js ●

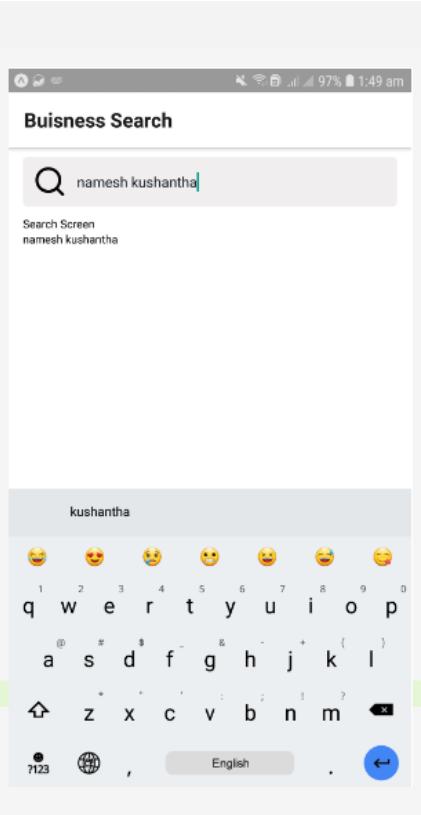
src > screens > JS SearchScreen.js > ...
1 import React from 'react';
2 import { View, StyleSheet, Text } from 'react-native';
3 import SearchBar from '../components/SearchBar'
4
5 const SearchScreen = () => {
6   return(
7     <View style={styles.viewStyle}>
8       <SearchBar />
9       <Text>Search Screen</Text>
10      </View>
11    );
12 };
13
14 const styles = StyleSheet.create({
15   viewStyle: {
16     marginHorizontal: 15
17   }
18 });
19
20 export default SearchScreen;
```

SearchBar.js

```
JS App.js          JS SearchBar.js X   JS SearchScreen.js ●  
src > components > JS SearchBar.js > [e] styles  
1  import React from 'react';  
2  import { View, StyleSheet, TextInput } from 'react-native';  
3  import { Feather, FontAwesome } from '@expo/vector-icons';  
4  
5  const SearchBar = () => {  
6      return(  
7          <View style={styles.backgroundStyle}>  
8              <Feather name="search" style={styles.iconStyle} />  
9              {/* <FontAwesome name="Search" /> */}  
10             <TextInput style={styles.inputStyle}  
11                 placeholder="Search"  
12             />  
13         </View>  
14     );  
15 };  
  
17  const styles = StyleSheet.create({  
18      backgroundStyle:{  
19          backgroundColor: '#F0EEEE',  
20          height: 50,  
21          borderRadius: 5,  
22          flexDirection: 'row',  
23          marginTop: 10,  
24          marginBottom: 10  
25      },  
26      inputStyle: {  
27          flex: 1,  
28          fontSize: 18,  
29          color: '#17202A'  
30      },  
31      iconStyle: {  
32          fontSize: 35,  
33          alignSelf: 'center',  
34          marginHorizontal: 10  
35      }  
36  });  
37  
38  export default SearchBar;
```

4. Managing State

<https://github.com/expo/vector-icons>



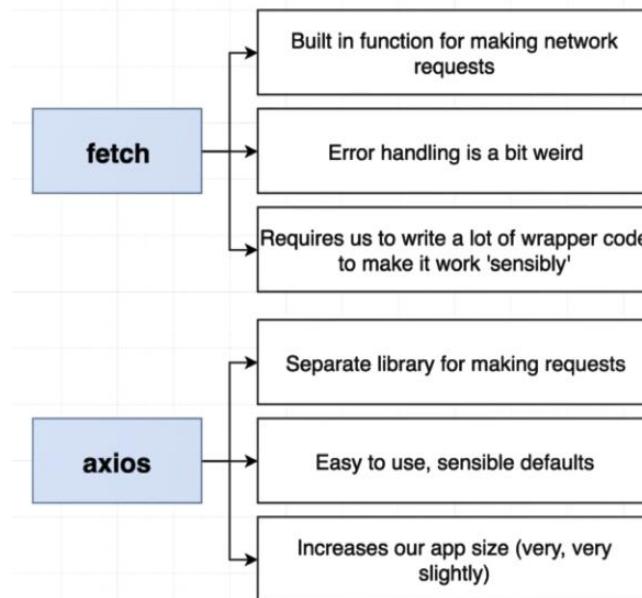
```
JS App.js JS SearchBar.js JS SearchScreen.js
src > components > JS SearchBar.js > [e] styles
1 import React from 'react';
2 import { View, StyleSheet, TextInput } from 'react-native';
3 import { Feather } from '@expo/vector-icons';
4
5 const SearchBar = ({ term, onTermChange, onTermSubmit }) => {
6   return(
7     <View style={styles.backgroundStyle}>
8       <Feather name="search" style={styles.iconStyle} />
9       {/* <FontAwesome name="Search" /> */}
10      <TextInput
11        autoCapitalize="none"
12        autoCorrect={false}
13        style={styles.inputStyle}
14        placeholder="Search"
15        value={term}
16        onChangeText={newTerm => onTermChange(newTerm)}
17        onEndEditing={() => onTermSubmit()}
18      />
19    </View>
20  );
21}
22
23 const styles = StyleSheet.create({
24   backgroundStyle:[
25     backgroundColor: '#F0EEEE',
26     height: 50,
27     borderRadius: 5,
28     flexDirection: 'row',
29     marginTop: 10,
30     marginBottom: 10
31   ]
32 });

kushantha
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
?123 , English . ←
```

```
JS App.js JS SearchBar.js JS SearchScreen.js
src > screens > JS SearchScreen.js > ...
1 import React, { useState } from 'react';
2 import { View, StyleSheet, Text } from 'react-native';
3 import SearchBar from '../components/SearchBar'
4
5 const SearchScreen = () => {
6   const [term, setTerm] = useState('');
7   return(
8     <View style={styles.viewStyle}>
9       <SearchBar
10         term={term}
11         onTermChange={newTerm => setTerm(newTerm)}
12         onTermSubmit={() => console.log('term was submitted')}
13       />
14       <Text>Search Screen</Text>
15       <Text>{term}</Text>
16     </View>
17   );
18 }
```

Draw a Relationship Diagram for this code. [C8 – V13]

9. Using Outside API's



We use axios for this example: **npm install axios**

And create the `yelp.js` file in `api` directory.

A screenshot of the VS Code interface showing the file structure and the content of the `yelp.js` file.

EXPLORER: OPEN EDITO... 1 UNSAVED
src > api > **yelp.js**
FOODPROJECT .expo .expo-shared assets node_modules src api yelp.js
components SearchBar.js screens SearchScreen.js .gitignore .watchmanconfig App.js

JS yelp.js

```
1 import axios from 'axios';
2
3 export default axios.create({
4   baseURL: 'https://api.yelp.com/v3/businesses',
5   headers: {
6     Authorization:
7       'Bearer 12cmhf2ezRl6ZkcHWNYKoDpiaro1zlqUjwkA7nVxnWzryiTWFk35
8 });
9 })
```

Api Key:

I2cmhf2ezRl6ZkcHWNYKoDpiaro1zlqUjwkA7nVxnWzryiTWFk35PJucoLhjjFY9ECD
8GTBGaHBg5yv5YDLiszKQx8EMvm30ply0UWoHQOnYFjLozpYnZOx-UsXXYx

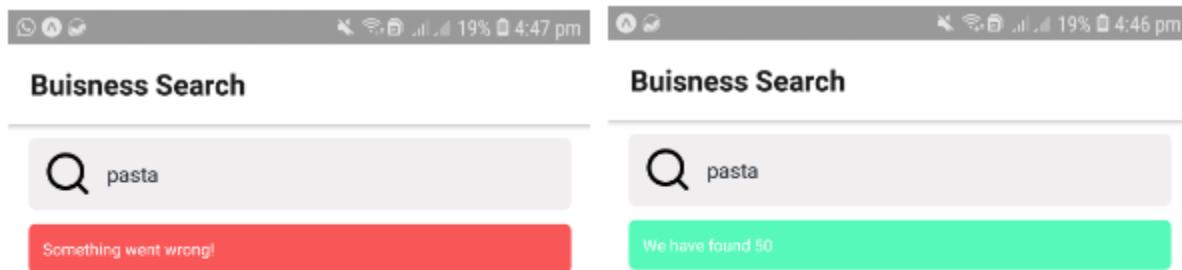
1. Making the Request

```
JS SearchScreen.js X
src > screens > JS SearchScreen.js > [o] SearchScreen > [o] searchApi
1  import React, { useState } from 'react';
2  import { View, StyleSheet, Text } from 'react-native';
3  import SearchBar from '../components/SearchBar';
4  import yelp from '../api/yelp';
5
6  const SearchScreen = () => {
7    const [term, setTerm] = useState('');
8    const [results, setResults] = useState([]);
9    const [errorMessage, setErrorMessage] = useState('');
10
11   const searchApi = async () => {
12     try {
13       const response = await yelp.get('/search', {
14         params: {
15           limit: 50,
16           term,
17           location: 'san jose'
18         }
19       });
20       setResults(response.data.businesses);
21     } catch (err) {
22       setErrorMessage('Something went wrong!')
23     }
24   };
25
```

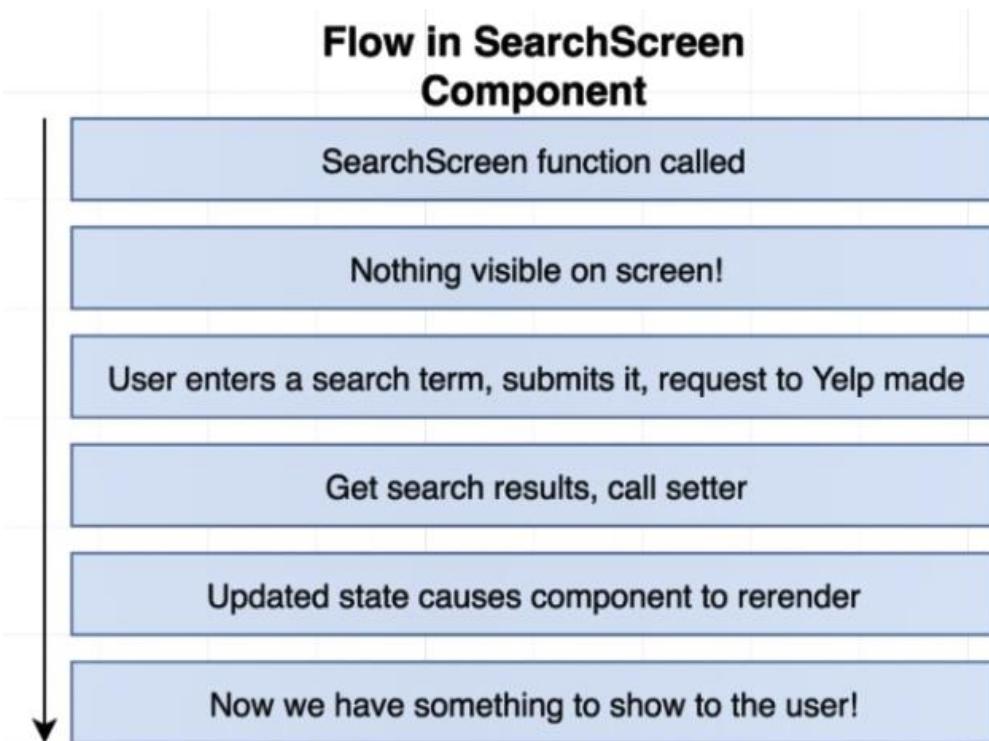
```
26   return (
27     <View style={styles.viewStyle}>
28       <SearchBar
29         term={term}
30         onTermChange={newTerm => setTerm(newTerm)}
31         onTermSubmit={() => searchApi()}
32       />
33       {errorMessage
34        ? <View style={styles.errorStyle}>
35          <Text style={styles.textStyle}>{errorMessage}</Text>
36        </View>
37        : null
38      }
39      {results.length > 1
40       ? <View style={styles.resultsStyle}>
41         <Text style={styles.textStyle}>We have found {results.length}</Text>
42       </View>
43       : null
44     }
45   </View>
46 );
47
48 };
```

```
50 const styles = StyleSheet.create({
51   viewStyle: {
52     marginHorizontal: 15
53   },
54   errorStyle: {
55     backgroundColor: '#F95757',
56     height: 35,
57     borderRadius: 5,
58     flexDirection: 'row',
59     marginBottom: 10
60   },
61   textStyle: {
62     margin: 10,
63     color: 'white'
64   },
65   resultsStyle: {
66     backgroundColor: '#57F9B9',
67     height: 35,
68     borderRadius: 5,
69     flexDirection: 'row',
70     marginBottom: 10
71   },
72 });
73
74 export default SearchScreen;
```

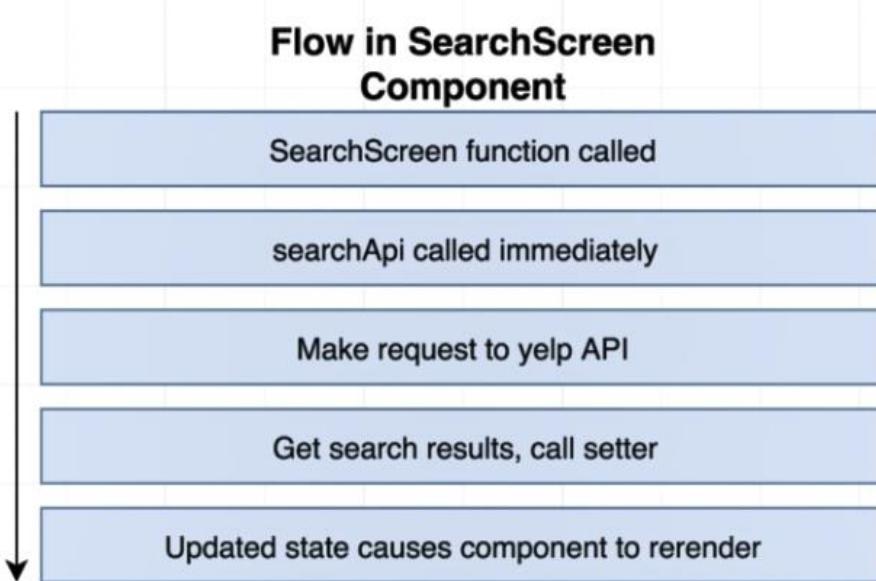
Results:



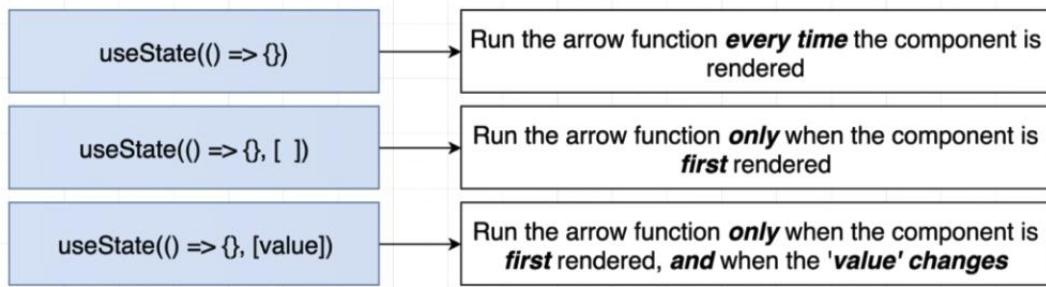
2. Running an Initial Search



How to create a running an initial search



useState's Second Argument



JS SearchScreen.js ● **JS** yelp.js

```
src > screens > JS SearchScreen.js > [S] SearchScreen
1  import React, { useState, useEffect } from 'react';
2  import { View, StyleSheet, Text } from 'react-native';
3  import SearchBar from '../components/SearchBar';
4  import yelp from '../api/yelp';

5
6  const SearchScreen = () => {
7      const [term, setTerm] = useState('');
8      const [results, setResults] = useState([]);
9      const [errorMessage, setErrorMessage] = useState('');

10
11     const searchApi = async (searchTerm) => {
12         try {
13             const response = await yelp.get('/search', {
14                 params: {
15                     limit: 50,
16                     term: searchTerm,
17                     location: 'san jose'
18                 }
19             });
20             setResults(response.data.businesses);
21         } catch (err) {
22             setErrorMessage('Something went wrong!');
23         }
24     };

25
26     useEffect(() => {
27         searchApi('pasta');
28     }, []);
29 }
```

```
30     return (
31       <View style={styles.viewStyle}>
32         <SearchBar
33           term={term}
34           onTermChange={newTerm => setTerm(newTerm)}
35           onTermSubmit={() => searchApi(term)}
36         />
```