Hello world

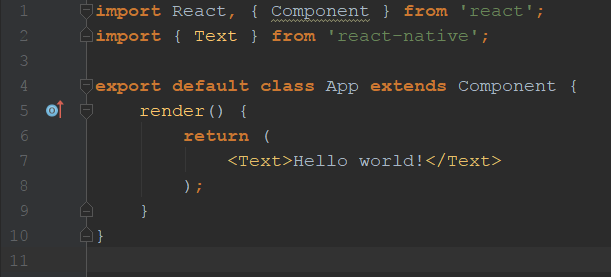
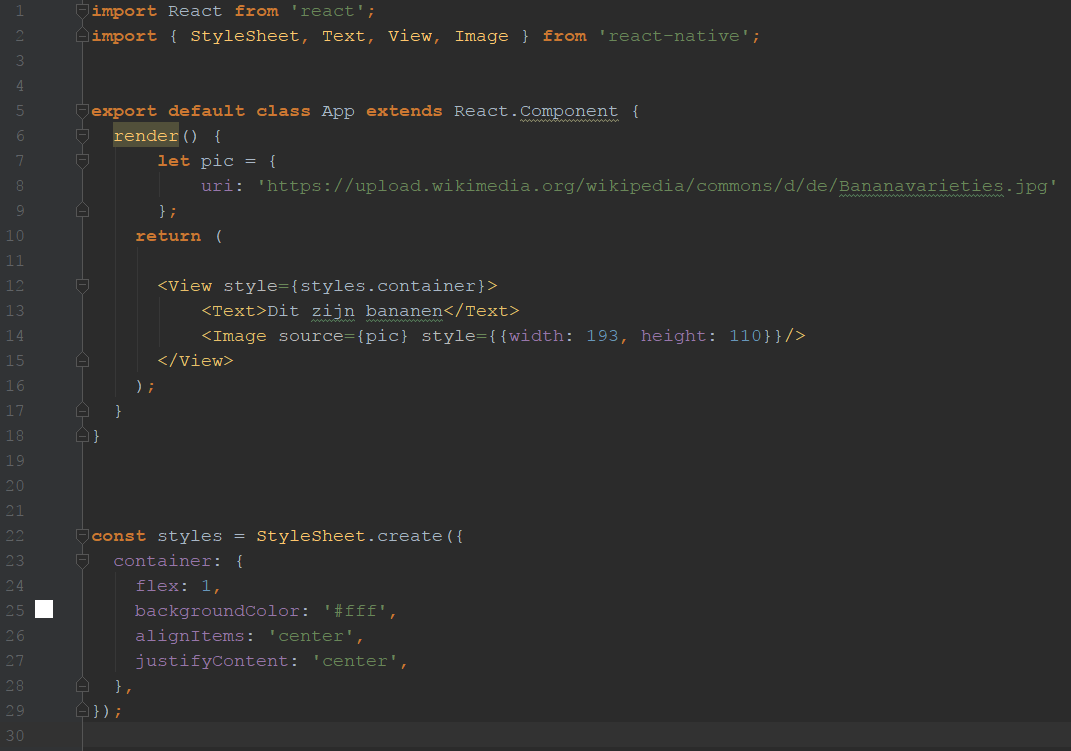
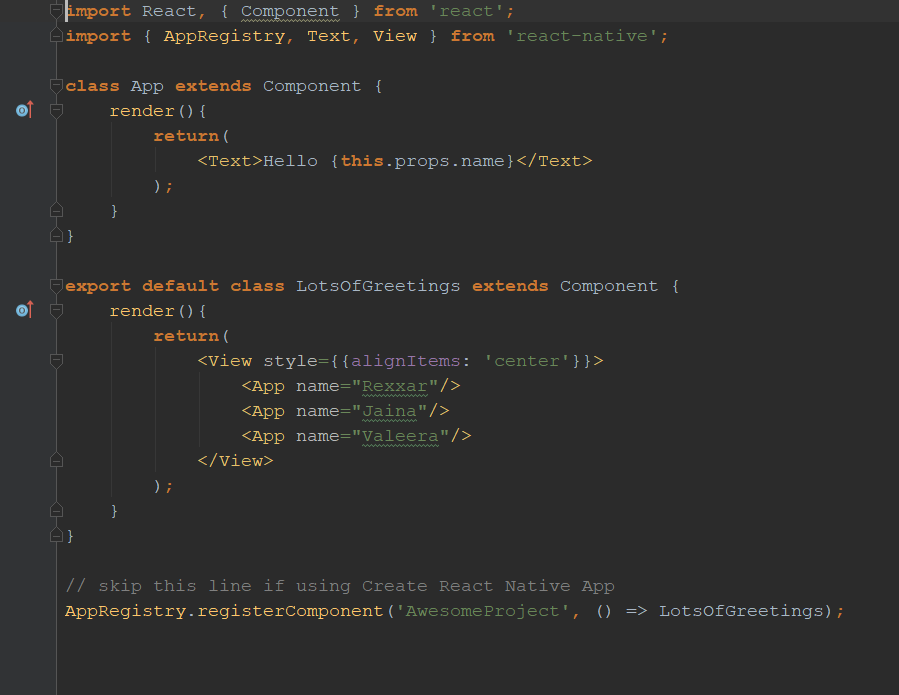


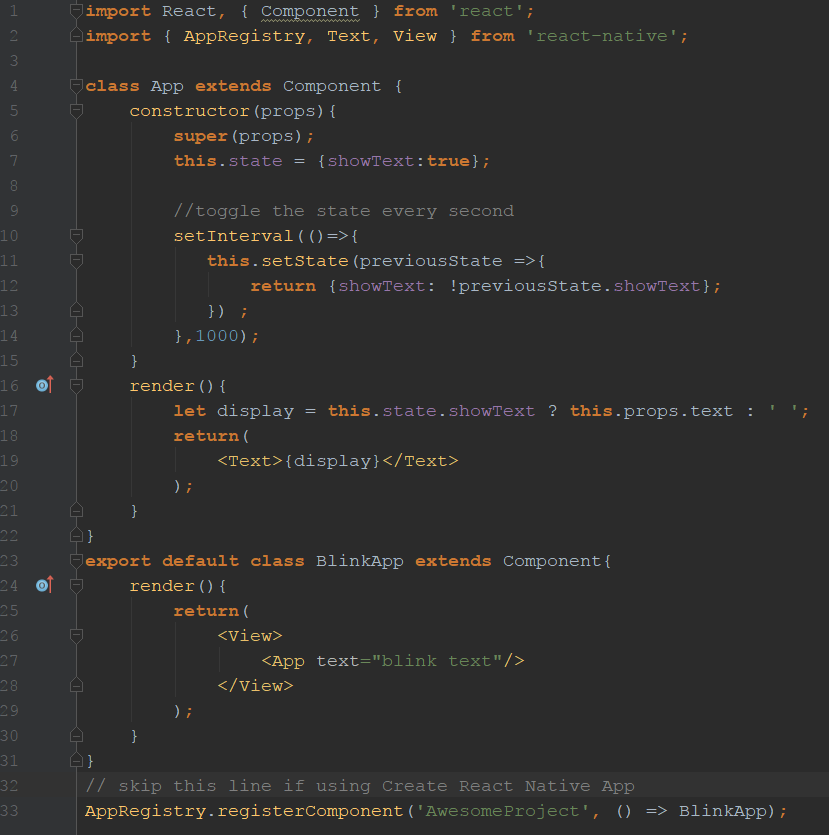
Image with prop source



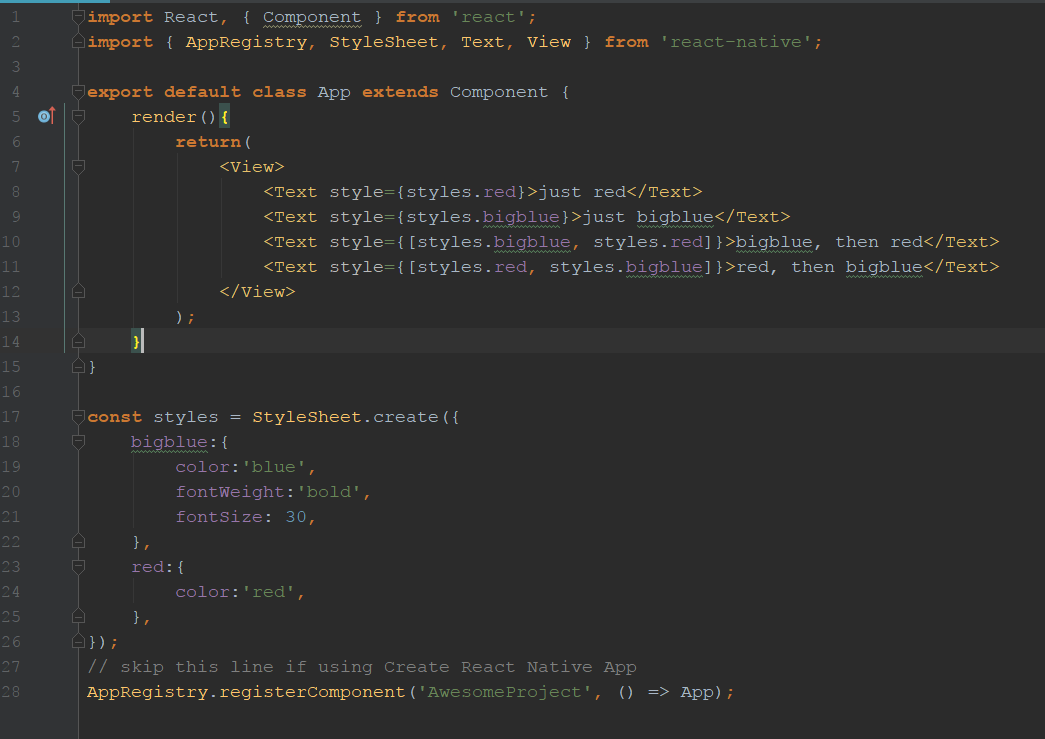
# My own component with props



# State: blink

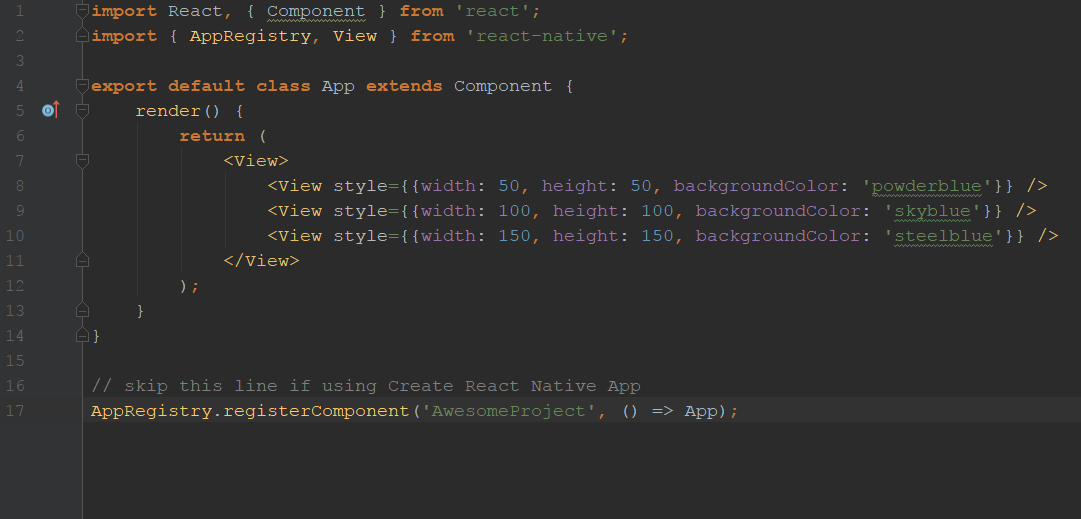


# Style

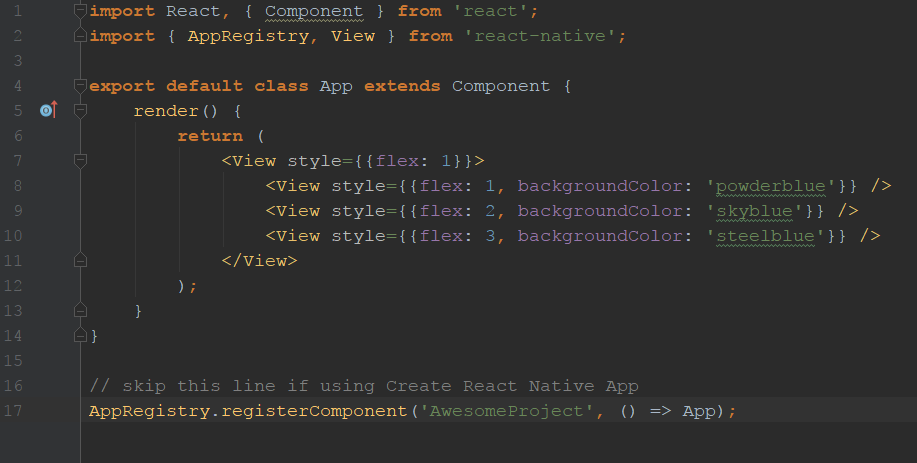


# Width & Height

## fixed

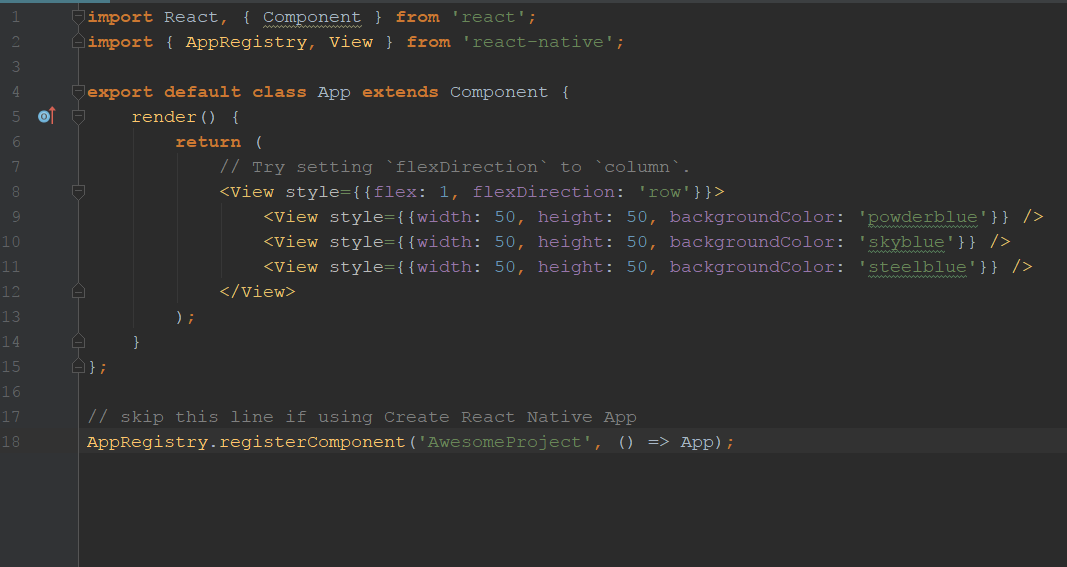


## Flex

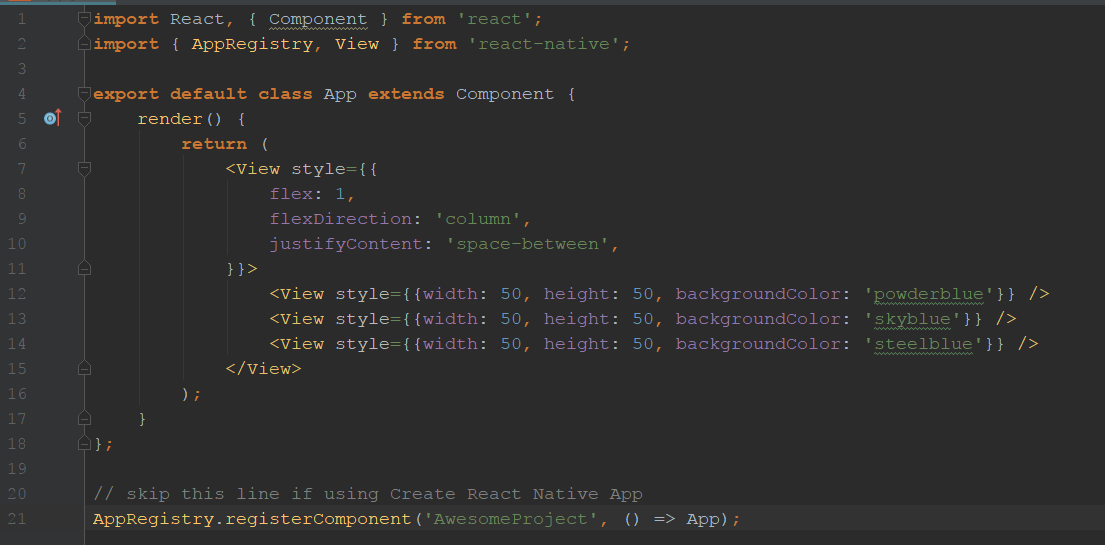


# Layout with Flexbox

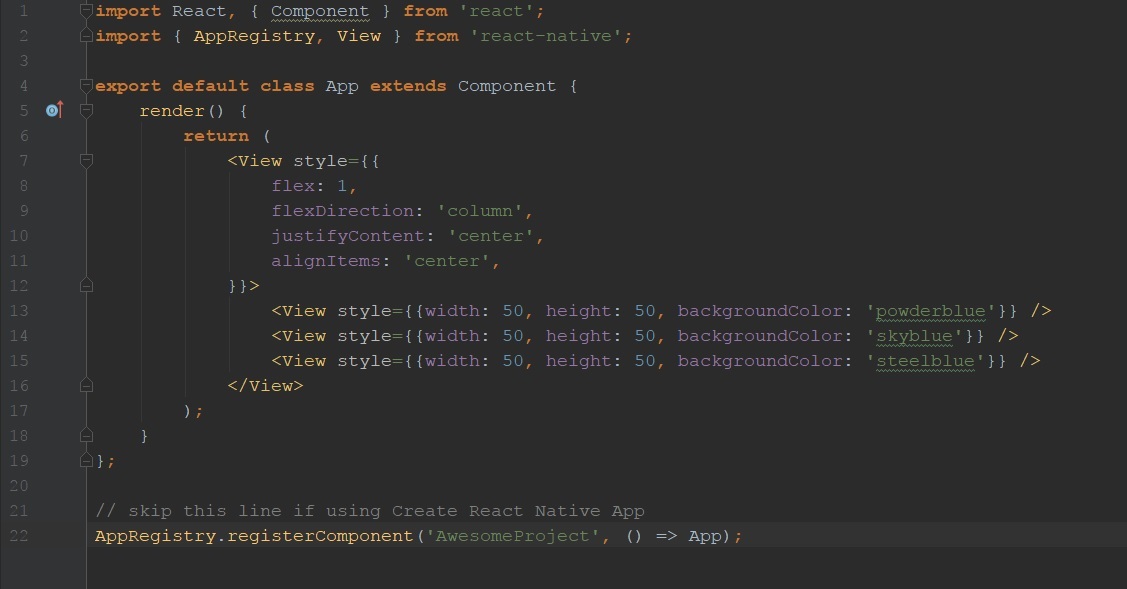
## Flex Direction



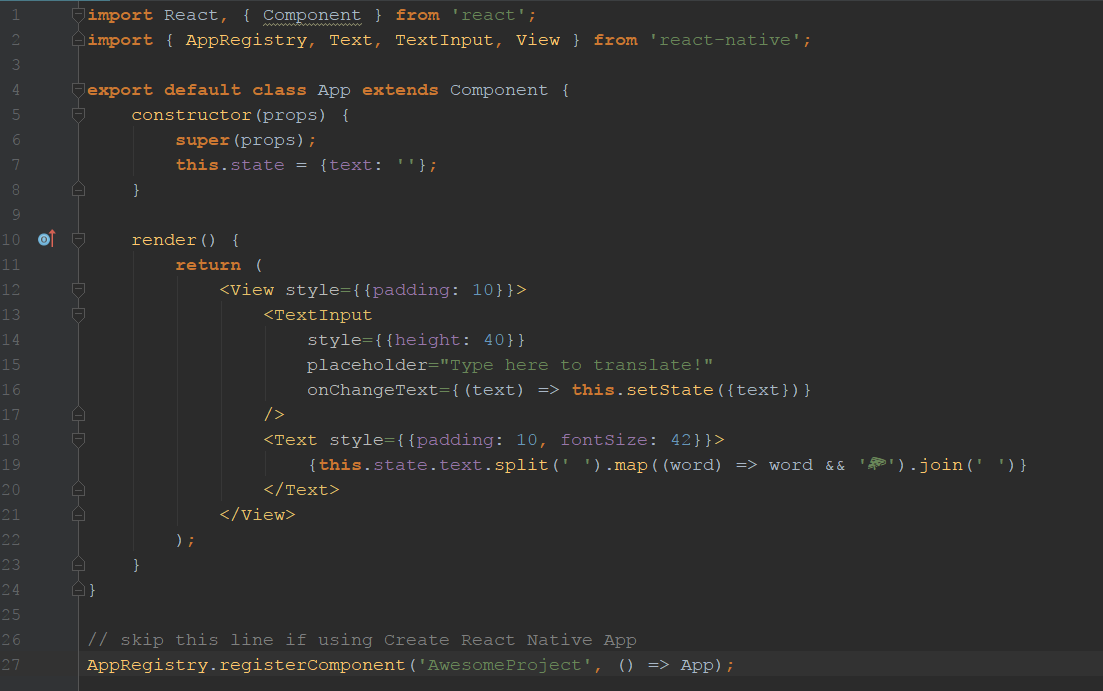
## Justify Content



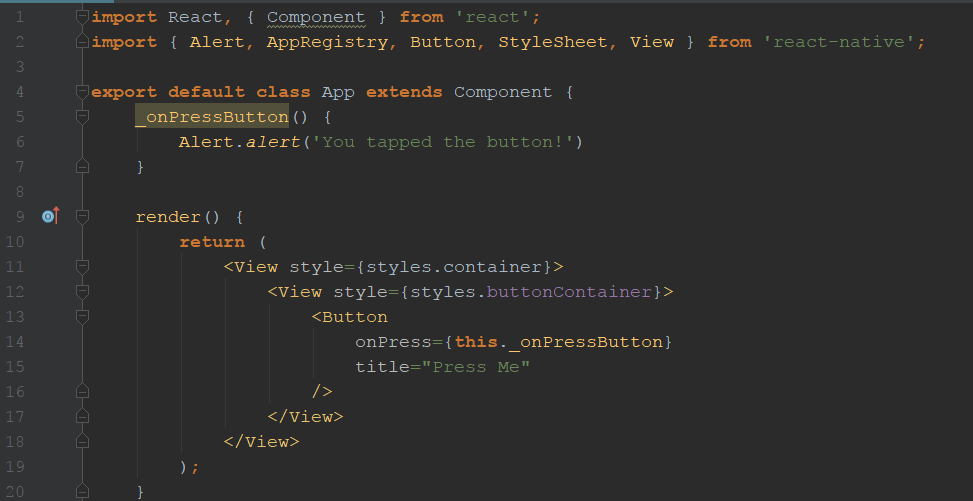
## Align Items



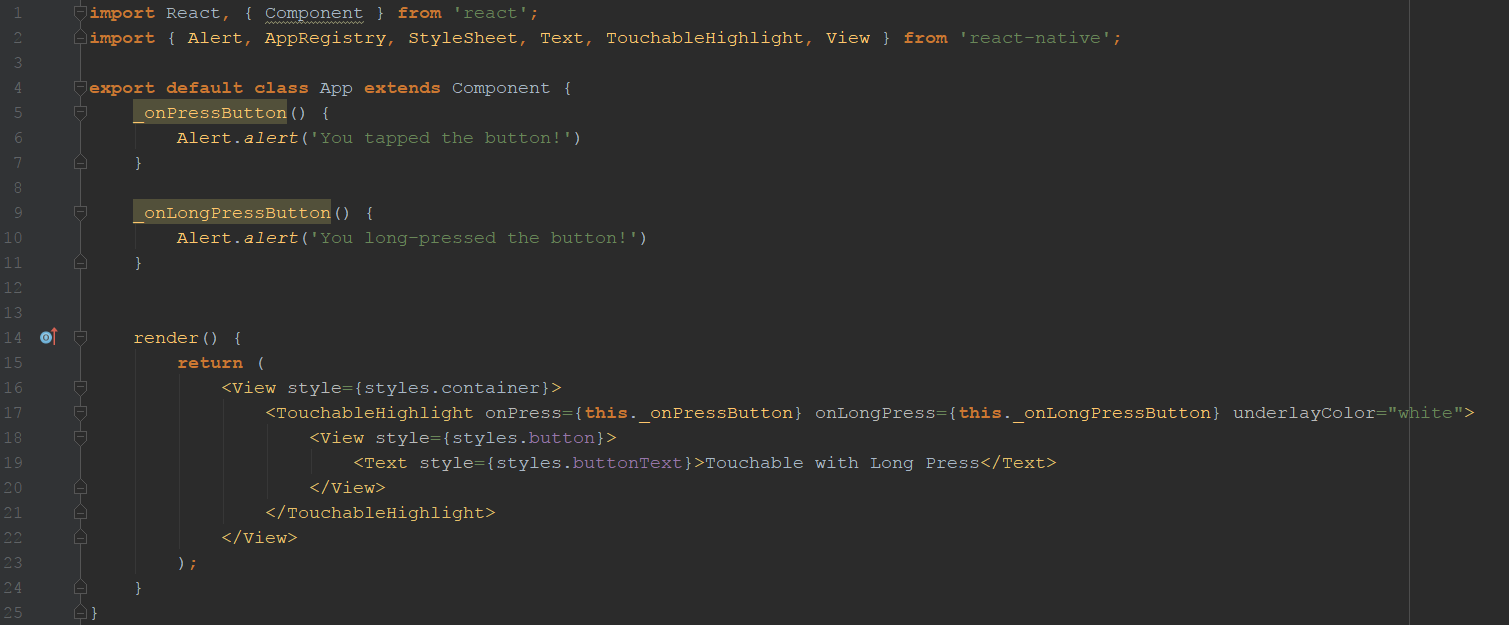
# Handling Text Input



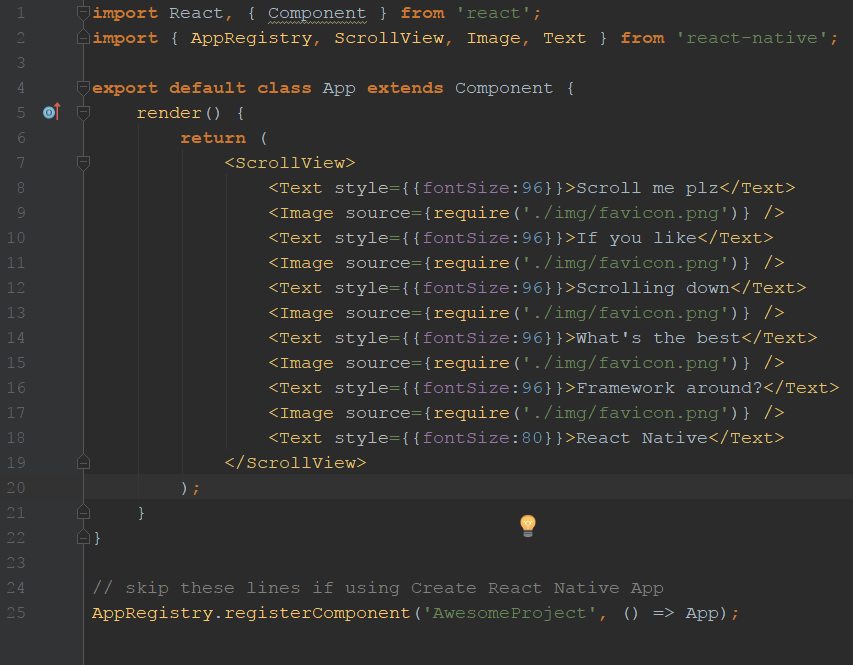
# Buttons



# Touchables – long press



# Scrollview



# List Views

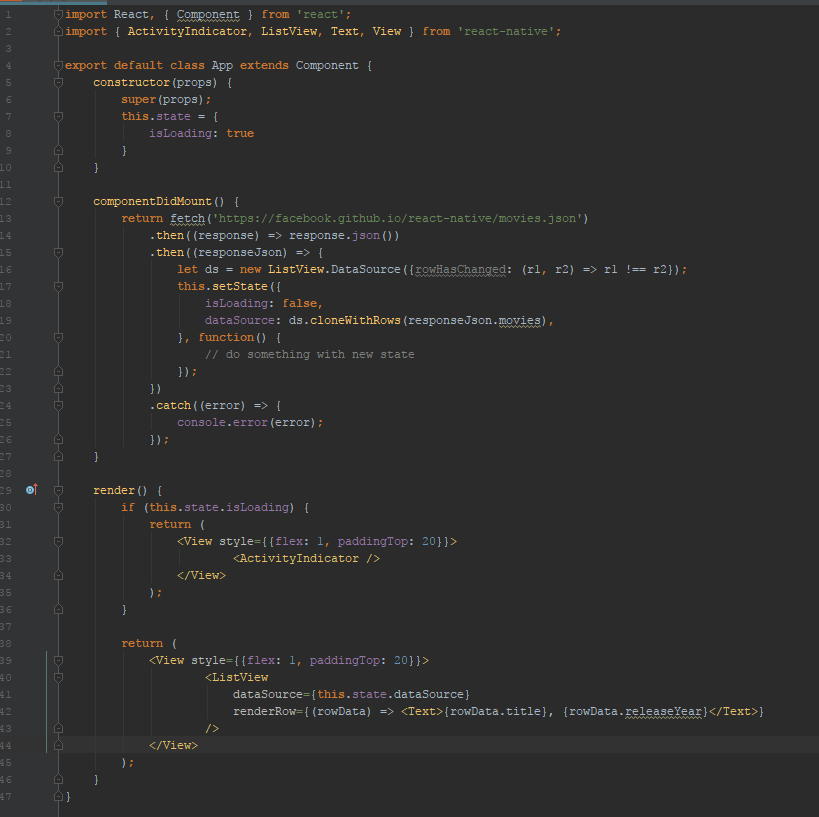
## Flatlist



## SectionList



# Networking



# CrashCourse

## Props aanroepen

In App een property doorgeven en in Component1 weergeven.

App.js

<Component1 message="Hello World"></Component1>

Component1.js

<Text>{**this**.props.message}</Text>

## State

Component1.js

constructor(){  
 **super**();  
 **this**.state = {  
 name:'Andreea'  
 }  
}

<Text>{**this**.state.name}</Text>

### Werken met Boolean in state

**export default class** Component1 **extends** Component{  
 constructor(){  
 **super**();  
 **this**.state = {  
 name:'Andreea',  
 showName:**true** }  
 }  
 render(){  
 **let** name = **this**.state.showName ? **this**.state.name : 'no name';  
 **return**(  
 <View>  
 <Text>{**this**.props.message}</Text>  
 <Text>{name}</Text>  
 </View>  
 );  
 }  
}

### Met state werken en static defaultProps om message te weergeven

**export default class** Component1 **extends** Component{  
 constructor(props){  
 **super**(props);  
 **this**.state = {  
 name:'Andreea',  
 showName:**true**,  
 message: **this**.props.message  
 }  
 }  
  
 **static** *defaultProps* = {  
 message:'Hi there'  
 }  
  
 render(){  
 **let** name = **this**.state.showName ? **this**.state.name : 'no name';  
 **return**(  
 <View>  
 <Text>{**this**.state.message}</Text>  
 <Text>{name}</Text>  
 </View>  
 );  
 }  
}

App.js

<Component1></Component1>

## Debug

React-native log-android

## Styling

## inline

<Text style={{color:'red'}}>Hello</Text>

## Stylesheet

<View style={styles.container}>

**const** styles = StyleSheet.create({  
 container: {  
 flex: 1,  
 backgroundColor: '#fff',  
 alignItems: 'center',  
 justifyContent: 'center',  
 },  
});

## Flexbox

**export default class** Component2 **extends** React.Component {  
 render() {  
 **return** (  
  
 <View style={styles.container}>  
 <View style={styles.v1}>  
 <Text>First</Text>  
 </View>  
  
 <View style={styles.v2}>  
 <Text>Second</Text>  
 </View>  
  
 <View style={styles.v3}>  
 <Text>Third</Text>  
 </View>  
 </View>  
  
 );  
 }

**const** styles = StyleSheet.create({  
 container: {  
 flexDirection:'row',  
 height:100  
 },  
 v1: {  
 flex:1,  
 backgroundColor:'red',  
 padding:10  
 },  
 v2: {  
 flex:1,  
 backgroundColor:'yellow',  
 padding:10  
 },  
 v3: {  
 flex:1,  
 backgroundColor:'black',  
 padding:10  
 },  
  
});

# Touchable highlight

onPress(){  
 console.log('Area Pressed');  
}

<TouchableHighlight style={styles.v1} onPress={**this**.onPress} underlayColor="blue">  
 <View >  
 <Text>First</Text>  
 </View>  
</TouchableHighlight>

# Touchable opacity

onPress2(){  
 console.log('Area2 Pressed');  
}

<TouchableOpacity style={styles.v2} onPress={**this**.onPress2}>  
 <View >  
 <Text>Second</Text>  
 </View>  
</TouchableOpacity>

# TextInput

constructor(){  
 **super**();  
 **this**.state = {  
 textValue:'hello'  
 }  
}  
  
onChangeText(value){  
 **this**.setState({  
 textValue:value  
 })  
}  
  
onSubmit(){  
 console.log('Input submitted');  
}

render() {  
 **return** (  
 <View style={styles.container}>  
 <TextInput  
 placeholder="Enter text"  
 value={**this**.state.textValue}  
 onChangeText={(value)=>**this**.onChangeText(value)}  
 onSubmitEditing={**this**.onSubmit}  
 />  
 <Text>{**this**.state.textValue}</Text>  
 </View>  
 );  
 }  
}

# Switch

onSwitchChange(value){  
 **this**.setState({  
 switchValue:value  
 })  
}

<Switch  
 value={**this**.state.switchValue}  
 onValueChange={(value)=>**this**.onSwitchChange(value)}  
/>

# ListView

**const** users = [  
 {name:'John'},  
 {name:'Brad'},  
 {name:'Steve'},  
 {name:'Janet'}  
]

**export default class** Component4 **extends** React.Component {  
 constructor(){  
 **super**();  
 **const** ds = **new** ListView.DataSource({rowHasChanged: (r1,r2) => r1 !== r2})  
 **this**.state = {  
 userDataSource: ds.cloneWithRows(users),  
 };  
 }  
  
 renderRow(user, sectionId, rowId, highlightRow){  
 **return**(  
 <View style={styles.row}>  
 <Text style={styles.rowText}>{user.name}</Text>  
 </View>)  
 }  
  
 render() {  
 **return** (  
 <ListView  
 dataSource={**this**.state.userDataSource}  
 renderRow={**this**.renderRow.bind(**this**)}  
 />  
 );  
 }  
}

# ListView + Json

<https://jsonplaceholder.typicode.com/users>

**export default class** Component5 **extends** React.Component {  
 constructor(){  
 **super**();  
 **const** ds = **new** ListView.DataSource({rowHasChanged: (r1,r2) => r1 !== r2})  
 **this**.state = {  
 userDataSource: ds,  
 };  
 }  
  
 componentDidMount(){  
 **this**.fetchUsers();  
 }  
  
 //populate listview  
 fetchUsers(){  
 fetch('https://jsonplaceholder.typicode.com/users')  
 .then((response)=>response.json())  
 .then((response)=>{  
 **this**.setState({  
 userDataSource:**this**.state.userDataSource.cloneWithRows(response)  
 });  
 });  
 }  
  
 renderRow(user, sectionId, rowId, highlightRow){  
 **return**(  
 <View style={styles.row}>  
 <Text style={styles.rowText}>{user.name}</Text>  
 </View>)  
 }  
  
 render() {  
 **return** (  
 <ListView  
 dataSource={**this**.state.userDataSource}  
 renderRow={**this**.renderRow.bind(**this**)}  
 />  
 );  
 }  
}

# Navigator

App.js

**export default class** App **extends** React.Component {  
 renderScene(route, navigator){  
 **switch**(route.id){  
 **case** 'component5':  
 **return** (<Component5 navigator={navigator} title="component5"/>)  
 **case** 'component6':  
 **return** (<Component6 user={route.user} navigator={navigator} title="component6"/>)  
 }  
 }  
 render() {  
 **return** (  
 <View>  
 <Navigator  
 initialRoute={{id:'component5'}}  
 renderScene={**this**.renderScene}  
 configureScreen={(route, routeStack)=>Navigator.SceneConfigs.FloatFromBottom}  
 />  
 </View>  
 );  
 }  
}

Component5

**export default class** Component5 **extends** React.Component {  
 constructor(){  
 **super**();  
 **const** ds = **new** ListView.DataSource({rowHasChanged: (r1,r2) => r1 !== r2})  
 **this**.state = {  
 userDataSource: ds,  
 };  
 }  
  
 componentDidMount(){  
 **this**.fetchUsers();  
 }  
  
 //populate listview  
 fetchUsers(){  
 fetch('https://jsonplaceholder.typicode.com/users')  
 .then((response)=>response.json())  
 .then((response)=>{  
 **this**.setState({  
 userDataSource:**this**.state.userDataSource.cloneWithRows(response)  
 });  
 });  
 }  
  
 onPress(user){  
 **this**.props.navigator.push({  
 //userdetails  
 id:'component6',  
 user: user  
 });  
 }  
  
 renderRow(user, sectionId, rowId, highlightRow){  
 **return**(  
 <TouchableHighlight onPress={()=>{**this**.onPress(user)}}>  
 <View style={styles.row}>  
 <Text style={styles.rowText}>{user.name}: {user.email}</Text>  
 </View>  
 </TouchableHighlight>  
 )  
  
 }  
  
 render() {  
 **return** (  
 <ListView  
 dataSource={**this**.state.userDataSource}  
 renderRow={**this**.renderRow.bind(**this**)}  
 />  
 );  
 }  
}

Component6

**export default class** Component6 **extends** React.Component {  
 constructor(props){  
 **super**(props);  
 **this**.state = {  
 name: **this**.props.user.name,  
 email: **this**.props.user.email  
 }  
 }  
 onPress(){  
 **this**.props.navigator.push({  
 id:'component5'  
 });  
 }  
 render() {  
 **return** (  
 <View>  
 <Text>{**this**.state.name}</Text>  
 <Text>{**this**.state.name}</Text>  
 <Button  
 onPress={**this**.onPress.bind(**this**)}  
 title="Go Back"  
 />  
 </View>  
 );  
 }  
}

# Edu

## propTypes

Component1.propTypes = {  
 message: React.PropTypes.string  
}