React Native Module – State

Peter Kassenaar –
info@kassenaar.com

State in React Native

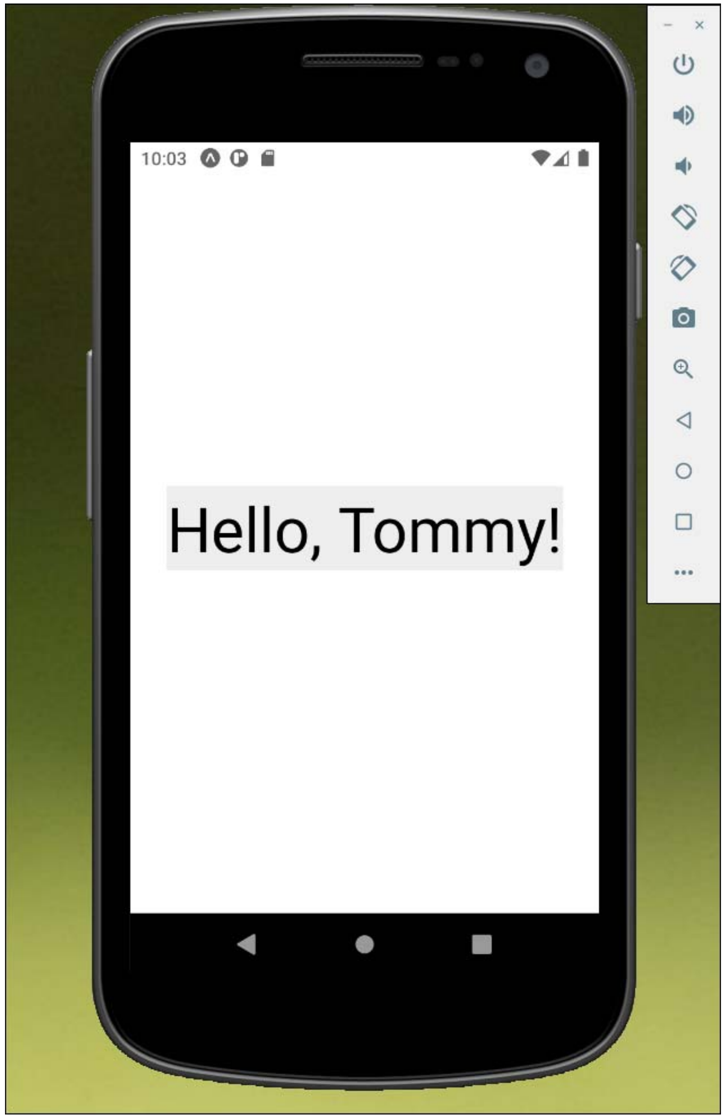
Working with data in your components

State in React Native…

- …works basically the same as in React for web
- We use Functional Components and React Hooks only here.
- Used for dynamic content displayed on the screen

```
export default function App() {
    // state in React Native works like in React Web
    const [name, setName] = useState('Tommy')
    return (
        <View style={styles.container}>
            <Text style={styles.heading}>Hello, {name}!</Text>
        </View>
    );
}
```

Result



We can update the state as normal, via `set[<someVariable>]` functions. For instance, using a a `<Button />` component and functions inside the component

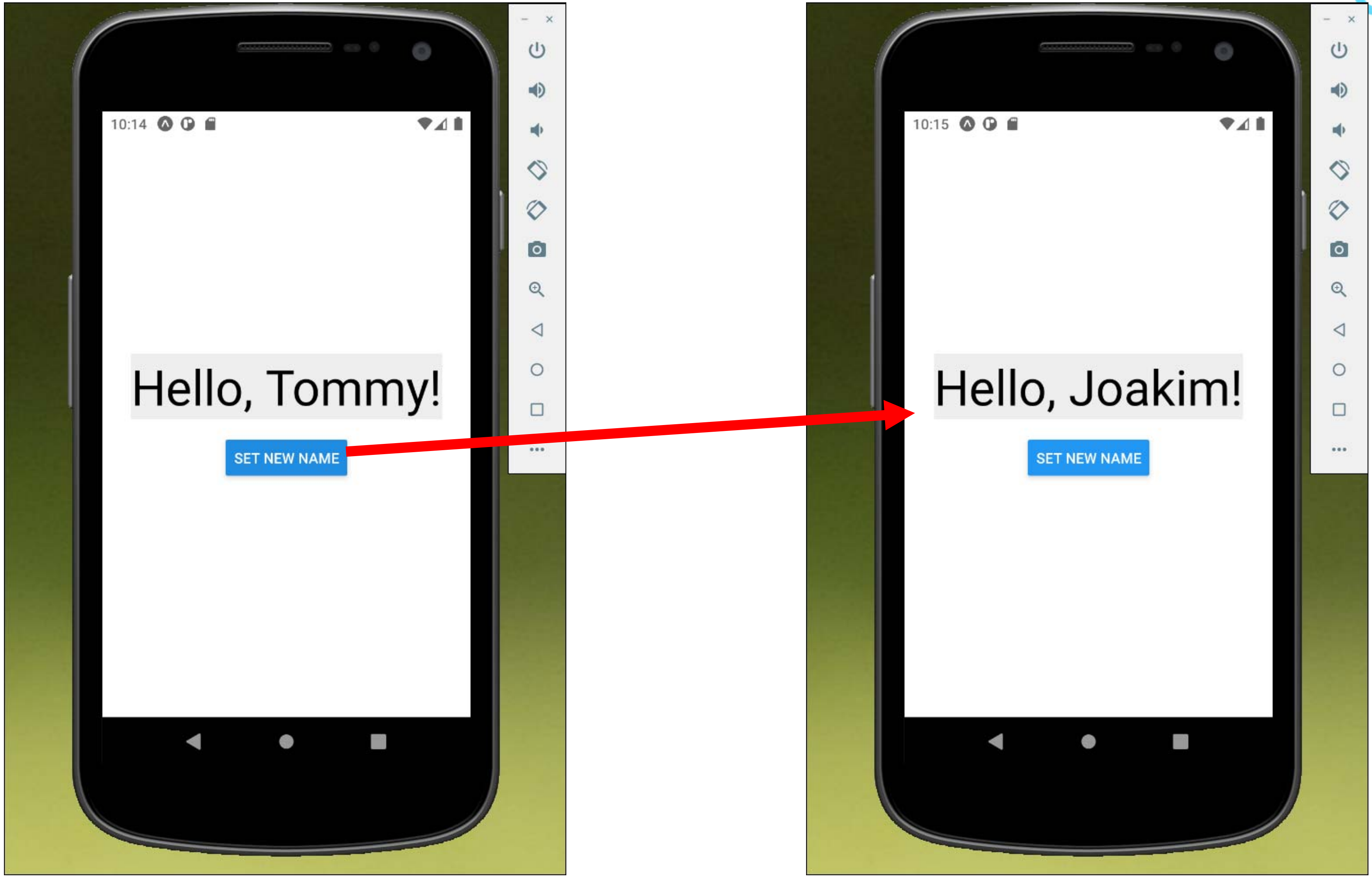
Updating the state

- 1. Create a function that updates the state
- 2. Create an event that triggers the function
  - Here: a <Button/> component
  - Use the onPress property
  - Buttons DON'T have a style property.

```
const [name, setName] = useState('Tommy')

const clickHandler = () => {
    setName('Joakim')
}
return (
    <View style={styles.container}>
        <Text style={styles.heading}>Hello, {name}!</Text>
        <View style={styles.buttonContainer}>
            <Button title="Set new name" onPress={clickHandler}/>
        </View>
    </View>
);
```

Result



Of course, this also works with objects

```
const [person, setPerson] = useState({
    name: 'Hannes',
    age: 32
})
```

```
const personHandler = () => {
    setPerson({
        name: 'Par',
        age: 41
    })
}
```

```
<Button title="Set new person" onPress={personHandler}/>
```

```
<Text>I am {person.name}, and I am { person.age} years old.</Text>
```

Styling the button

- Default buttons come with default styling
- If you use an unstyled component, for instance `<Pressable />`, you can style it per your needs

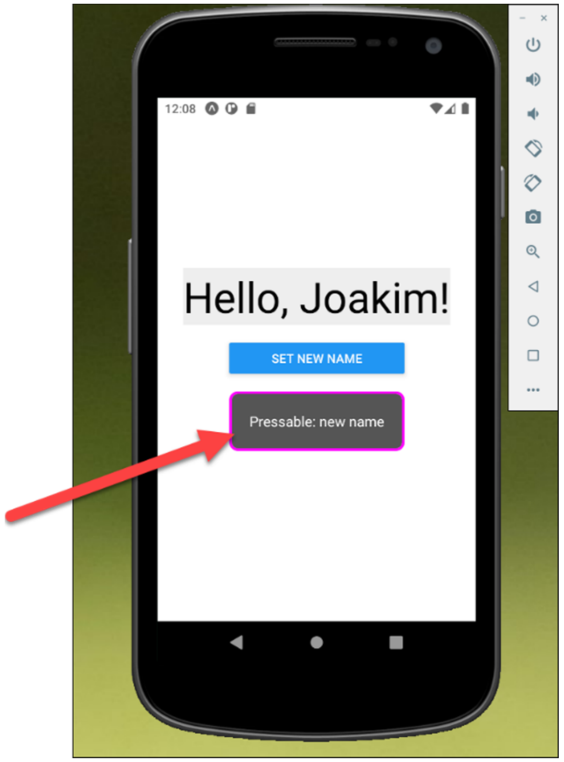
“Pressable is a Core Component wrapper that can detect various stages of press interactions on any of its defined children.

https://reactnative.dev/docs/pressable

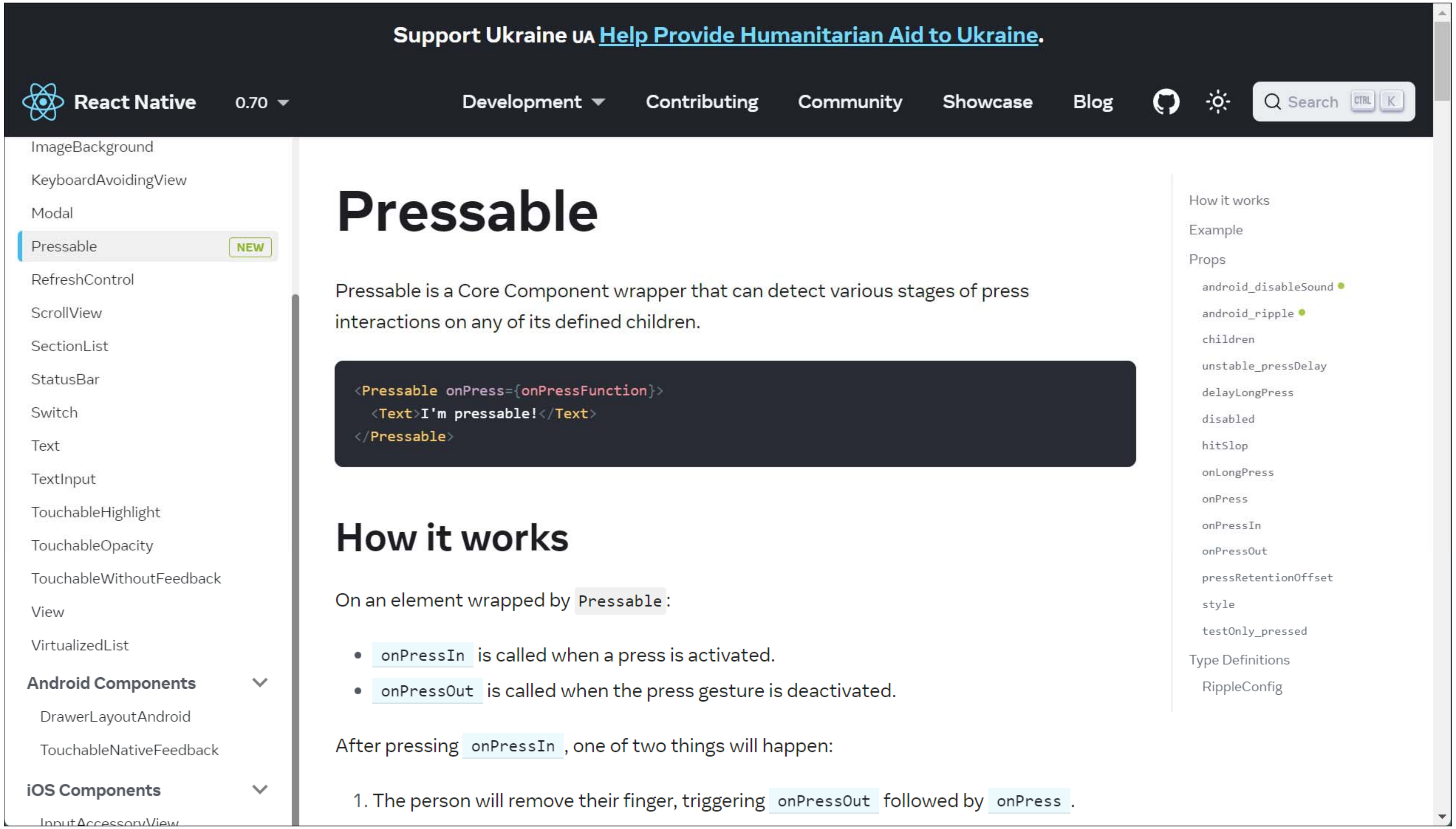
Example <Pressable>

```
<Pressable style={styles.button}
           onPress={clickHandler}
           android_ripple={styles.ripple}>
    <Text style={styles.buttonLabel}>Pressable: new name</Text>
</Pressable>
```

```
button: {
    marginTop: 20,
    padding: 20,
    borderRadius: 10,
    borderWidth: 3,
    borderColor: 'BFFF00C3',
    backgroundColor: '#555'
},
buttonLabel: {
    color: '#fff',
    fontSize: 16,
},
ripple: {
    color: 'red'
}
```



More info on <Pressable>



Support Ukraine UA Help Provide Humanitarian Aid to Ukraine.

Pressable

Pressable is a Core Component wrapper that can detect various stages of press interactions on any of its defined children.

```
<Pressable onPress={onPressFunction}>
  <Text>I'm pressable!</Text>
</Pressable>
```

How it works

On an element wrapped by `Pressable`:

- `onPressIn` is called when a press is activated.
- `onPressOut` is called when the press gesture is deactivated.

After pressing `onPressIn`, one of two things will happen:

1. The person will remove their finger, triggering `onPressOut` followed by `onPress`.

https://reactnative.dev/docs/pressable

Using TextInput

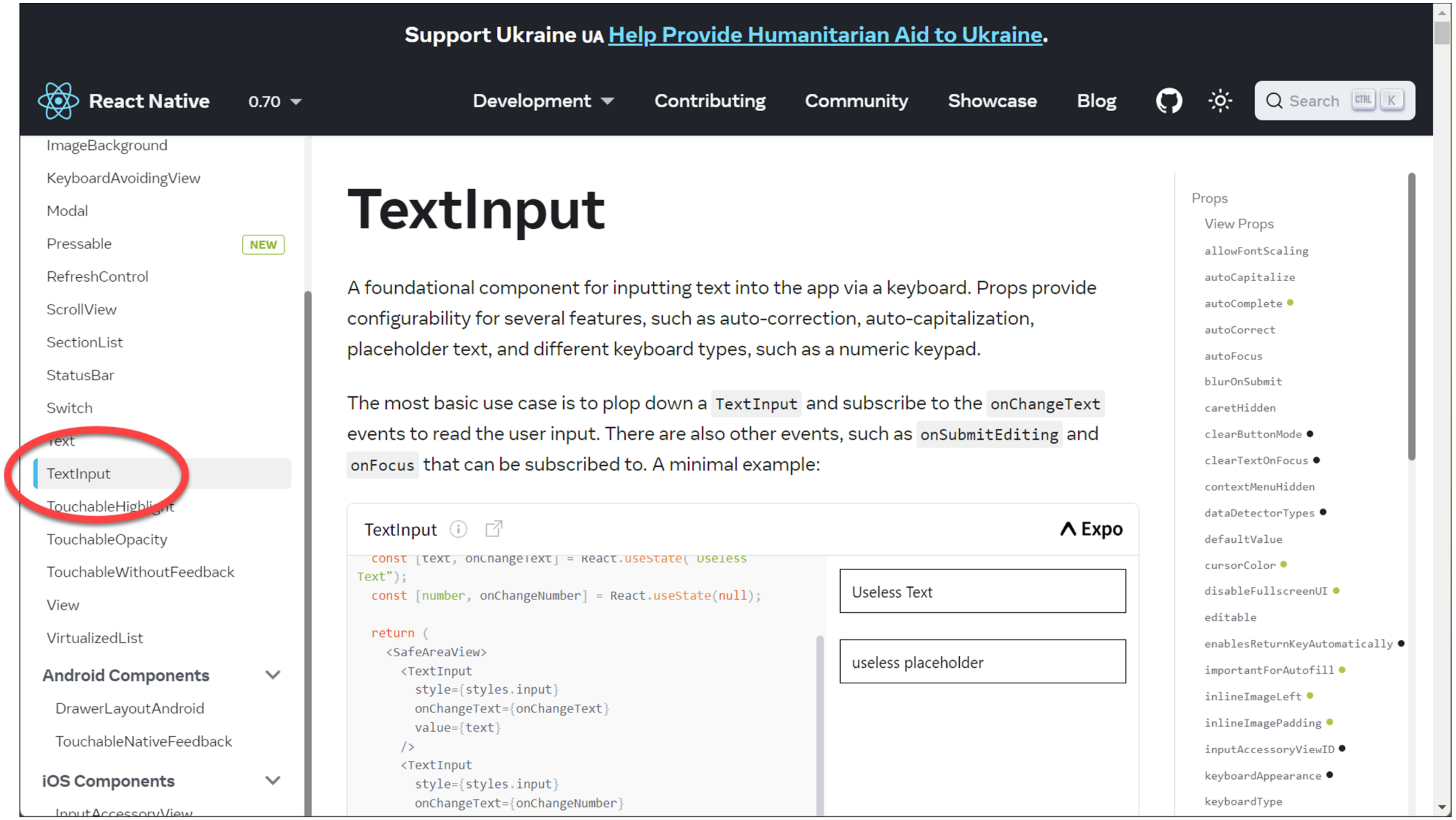
Getting data from your users, using a textfield

Getting data from users: `<TextInput />`

"A foundational component for inputting text into the app via a keyboard. Props provide configurability for several features, such as auto-correction, auto-capitalization, placeholder text, and different keyboard types, such as a numeric keypad."

https://reactnative.dev/docs/textinput

reactnative.dev/docs/textinput



Support Ukraine UA Help Provide Humanitarian Aid to Ukraine.

TextInput

A foundational component for inputting text into the app via a keyboard. Props provide configurability for several features, such as auto-correction, auto-capitalization, placeholder text, and different keyboard types, such as a numeric keypad.

The most basic use case is to plop down a `TextInput` and subscribe to the `onChangeText` events to read the user input. There are also other events, such as `onSubmitEditing` and `onFocus` that can be subscribed to. A minimal example:

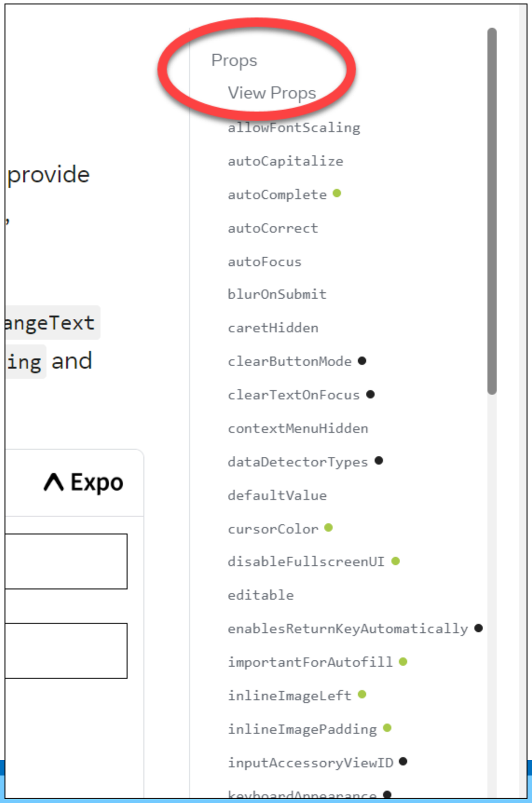
TextInput Expo

```
  const [text, onChangeText] = React.useState("Useless
Text");
  const [number, onChangeNumber] = React.useState(null);

  return (
    <SafeAreaView>
      <TextInput
        style={styles.input}
        onChangeText={onChangeText}
        value={text}
      />
      <TextInput
        style={styles.input}
        onChangeText={onChangeNumber}
```

Features of `<TextInput />`

- Also: NO default styling (so you actually can’t see it on your screen!) so set a `style=“…”` property.
- Lots of properties and methods. For instance:
  - `placeholder`
  - `multiline`
  - `keyboardType`
  - `defaultValue`
  - `onChangeText()`
  - `onFocus()`, `onBlur()`, etc.



provide
,
angeText
ing and
Expo

Props
View Props
allowFontScaling
autoCapitalize
autoComplete ●
autoCorrect
autoFocus
blurOnSubmit
caretHidden
clearButtonMode ●
clearTextOnFocus ●
contextMenuHidden
dataDetectorTypes ●
defaultValue
cursorColor ●
disableFullscreenUI ●
editable
enablesReturnKeyAutomatically ●
importantForAutofill ●
inlineImageLeft ●
inlineImagePadding ●
inputAccessoryViewID ●
keyboardAppearance ●

https://reactnative.dev/docs/textinput

Example <TextInput />

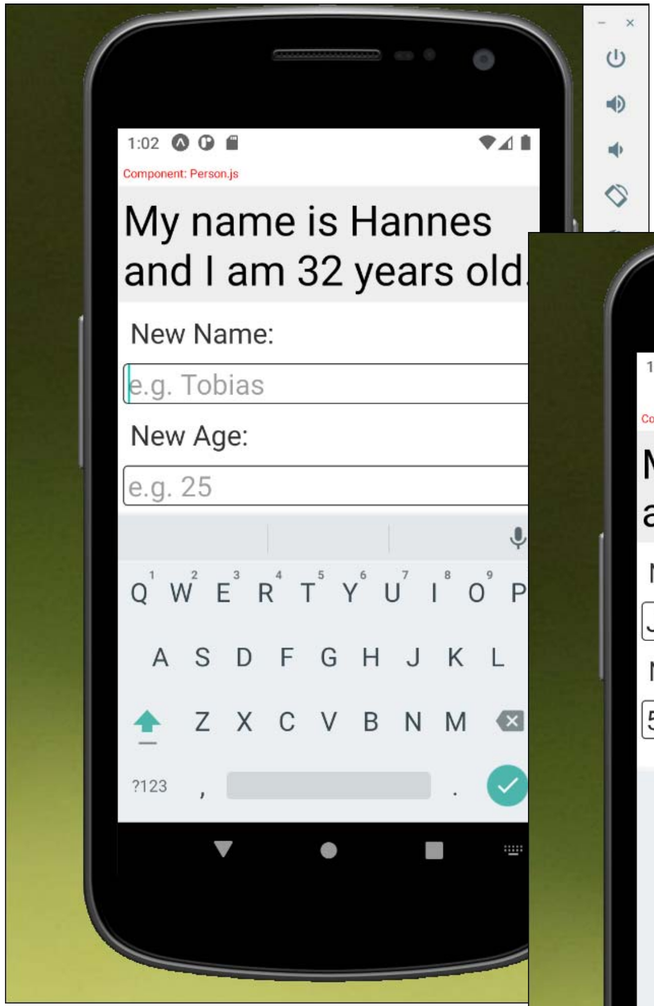
```
const Person = () => {
    // 0. Data/state in person component
    const [person, setPerson] = useState({
        name: 'Hannes',
        age: 32
    })

    const updatePerson = (prop, val) =>{
        setPerson({
            ...person,
            [prop]: val
        })
    }

    return (
        <View>
            <Text style={styles.heading}>My name is {person.name} and I am {person.age} years old.</Text>
            <Text style={styles.person}>New Name:</Text>
            <TextInput style={styles.input}
                       placeholder="e.g. Tobias"
                       onChangeText={(val) => updatePerson('name', val)}
            />
            <Text style={styles.person}>New Age:</Text>
            <TextInput style={styles.input}
                       placeholder="e.g. 25"
                       keyboardType="numeric"
                       onChangeText={(val) => updatePerson('age',  +val)}
            />
        </View>
    );
}
```

Example styles and output

```
const styles = StyleSheet.create({
    component: {
        fontSize: 10,
        color: 'red',
        margin: 4
    },
    heading: {
        fontSize: 36,
        backgroundColor: '#eee',
        padding: 4,
    },
    person: {
        color: '#333',
        fontSize: 24,
        alignItems: 'center',
        justifyContent: 'center',
        padding: 10
    },
    input: {
        borderStyle: "solid",
        borderWidth: 1,
        borderColor: '#333',
        borderRadius: 4,
        marginHorizontal: 4,
        paddingHorizontal: 4,
        fontSize: 24
    }
})
```


1:02

Component: Person.js

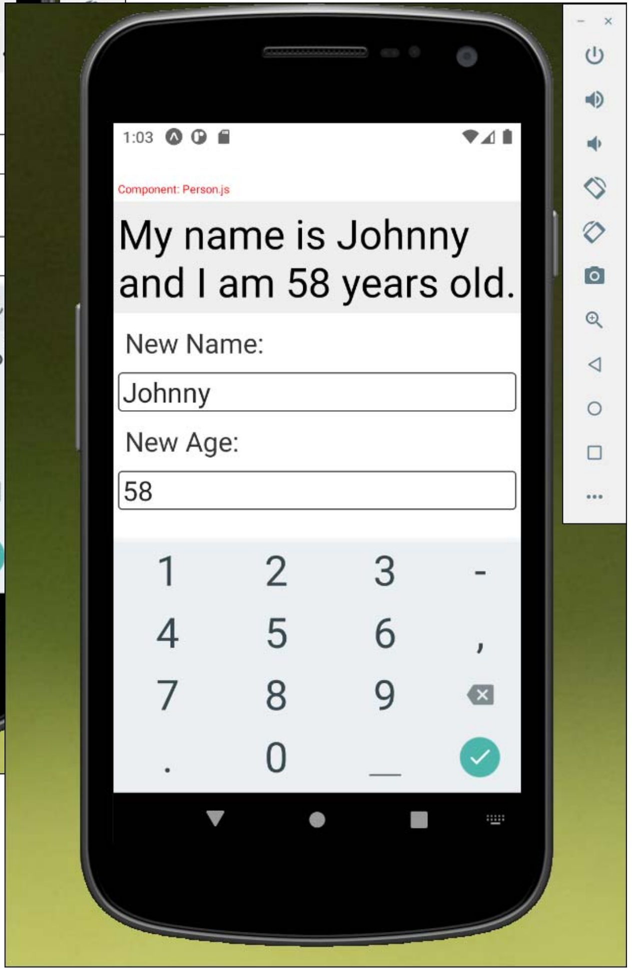
My name is Hannes and I am 32 years old.

New Name:

e.g. Tobias

New Age:

e.g. 25


1:03

Component: Person.js

My name is Johnny and I am 58 years old.

New Name:

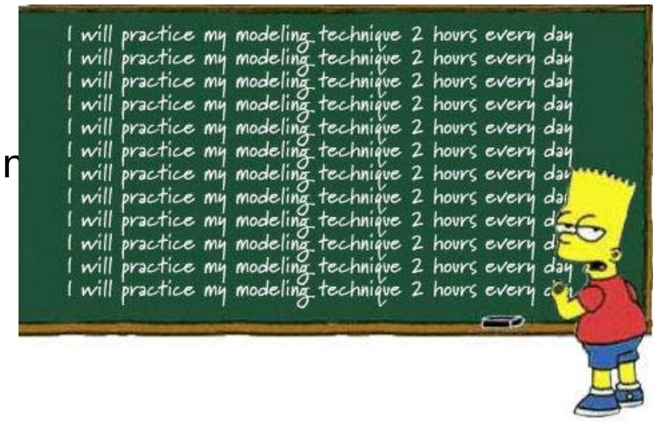
Johnny

New Age:

58

Workshop

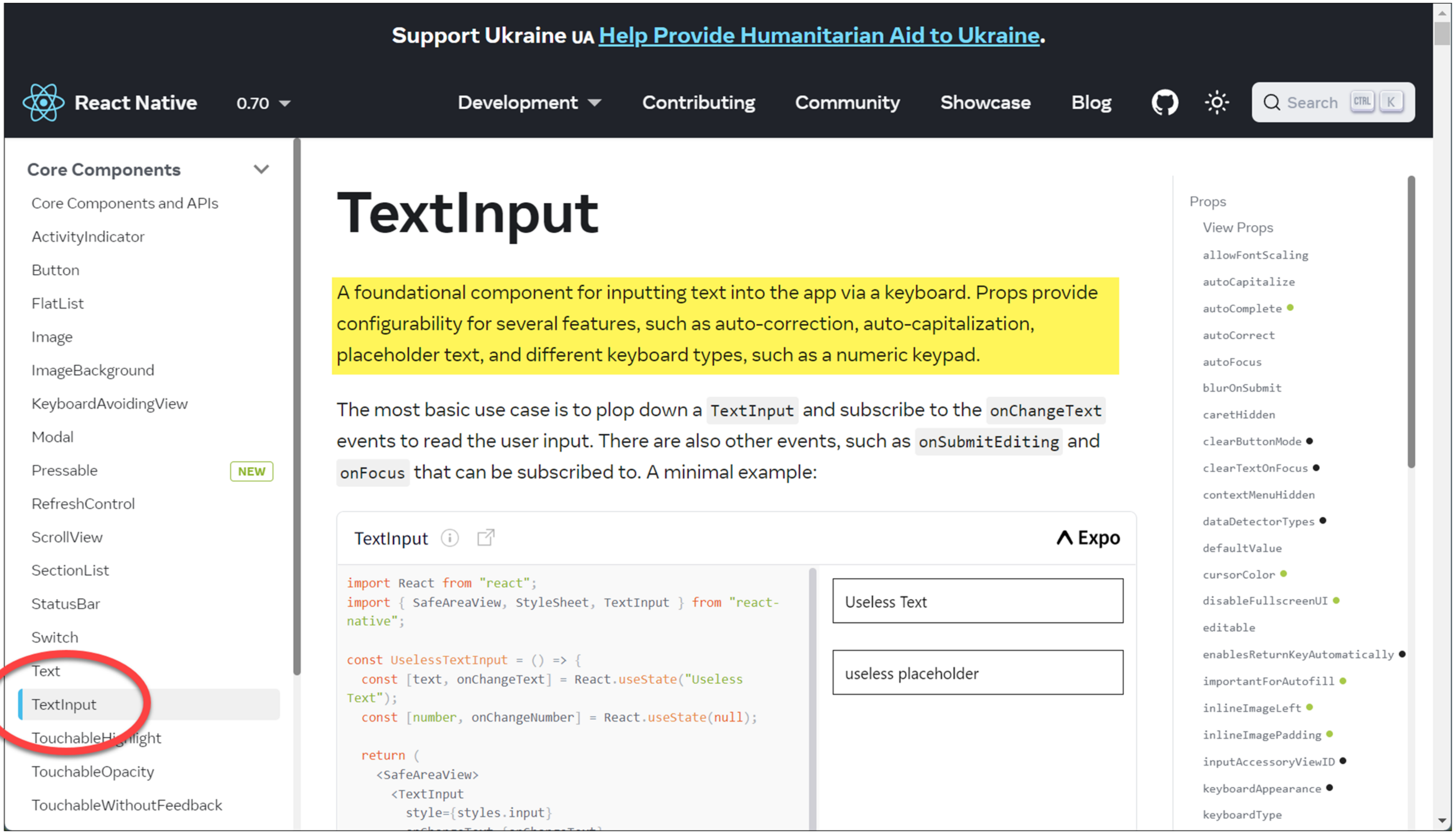
- Create a new Component with your favorite `Car | Person |` Food, etc. Add this to the state,
- Show the properties of your object on the screen
- Make it possible to update properties using `<TextInput />`
- Set correct styling for properties and inputfield
- Set correct type of keyboard
- Optional:
  - 1. DON't update text on every keystroke, but create a button that updates the object/properties when pressed.
  - 2. Add validations to update method
- Example `../40-state-textinput`


I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every day
I will practice my modeling technique 2 hours every

More info

More info on the topics in this module

TextInput: https://reactnative.dev/docs/textinput



TextInput

A foundational component for inputting text into the app via a keyboard. Props provide configurability for several features, such as auto-correction, auto-capitalization, placeholder text, and different keyboard types, such as a numeric keypad.

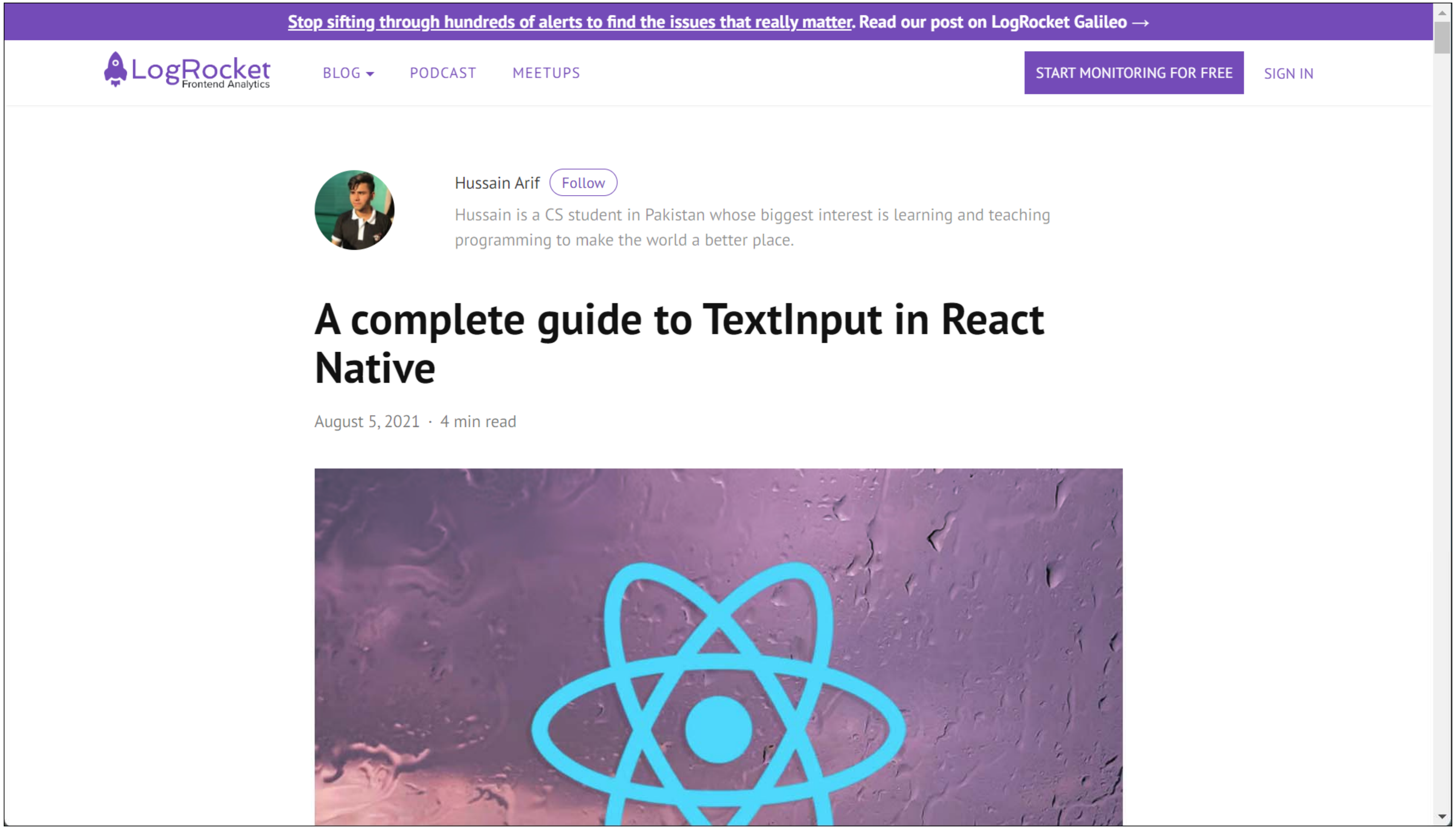
The most basic use case is to plop down a `TextInput` and subscribe to the `onChangeText` events to read the user input. There are also other events, such as `onSubmitEditing` and `onFocus` that can be subscribed to. A minimal example:

```
import React from "react";
import { SafeAreaView, StyleSheet, TextInput } from "react-
native";

const UselessTextInput = () => {
  const [text, onChangeText] = React.useState("Useless
Text");
  const [number, onChangeNumber] = React.useState(null);

  return (
    <SafeAreaView>
      <TextInput
        style={styles.input}
```

Blog on <TextInput />



Stop sifting through hundreds of alerts to find the issues that really matter. Read our post on LogRocket Galileo →

Hussain Arif

Hussain is a CS student in Pakistan whose biggest interest is learning and teaching programming to make the world a better place.

A complete guide to TextInput in React Native

August 5, 2021 · 4 min read

https://blog.logrocket.com/complete-guide-textinput-react-native/