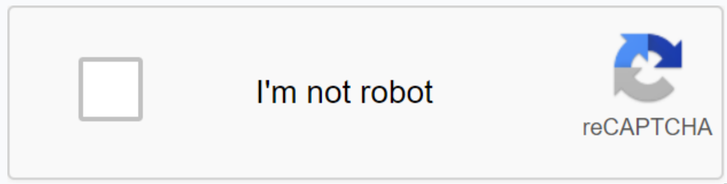

Continue

Form input react native elements

Forms are a very interactive part of our website/product UI. Feedback, surveys, data collection, etc. If you don't get it right, you can unwittingly give your users a bad experience that can lead to a drastic reduction in using your product. In this post we will demonstrate how to build different performant React Native forms with native React native Native UI components. Prerequisites Prerequisites Previous knowledge of React Native will be useful, but not mandatory You should have Xcode or Android Studio installed to the simulator/emulator You must have node 10+ installed It is worthy to note that we will not cover the styling for the youth examples we will build in this project. This will allow us to focus on functionality. However, all styles will be available in the project reward for your use. At the beginning we are going to start with some installations for those using React Native for the first time. If this is your first time with React Native, be sure to install the required packages and follow these steps accordingly: Install the React Native CLI tool using npm as follows: npm -g react native cli afterwards, continue and run these React Native Commands to create and start a new project. Create the animations project and start the server: react-initiative initiative cd pickerSample & npx react-inative run-ios// Run the iOS simulator//OR cd pickerSample &; npx react-native run-Android/Android the Android emulator if you have installed Xcode or Android Studio then the commands above will run the project and you should show the simulator on the screen as follows: In my case, I use Xcode. Reak native forms – voter The React Native picker component is the equivalent of a dropdown in regular JavaScript. It is basically used to deliver a list of multiple selections where users then select only one option from the list. The functionality comes in handy when asking a question with different answers. Say, for example, in payment you want to ask a user to choose which currency they want to pay, it makes sense to deliver as many currencies as possible, depending on the range of your user base. Let's see how we can achieve it using the React Native Picker Component! Click here to see the full demo with network requests in the root directory of the sample project we created create a new src/components/Picker.js file and define certain form fields for demonstration purposes: import React, {useState} from 'res respond'; perform {Picker, Text, StyleSheet, View, TextInput, Button} from 'react-inactive'; Const App = ()= > {const [currency, setCurrency] = useState('US Dollar'); return <View> <Text>(Demo Form</Text> <View> placeholder=Email></TextInput> <TextInput securetextentry={true} placeholder=Password></TextInput> <Picker ></Picker > </View> </View> </View> onValueChange={currentCurrency => setCurrency(currentCurrency)}> <Picker.Item label=USD value=US Dollars></Picker.Item > <Picker.Item label=EUR value=Euro></Picker.Item > <Picker.Item label=NGN value=Naira></Picker.Item> <Text>Selected: {currency});</Text> }; constants = StyleSheet.create({//Check project repo for styles }); Export Default App; To perform this component, we need to update our App.js file as follows: import react from 'react'; Import Picker of './src/Components/Picker' Const App = ()= > {return <Picker></Picker> ();}; Export Default App; What we've done is simply delivering the Picker component we created in the src/components/Picker.js file here. When the program loads, the voter file is delivered. If we run the program at this point, we need to get the following output on the simulator: You can take your knowledge of this component further by learning which props the components take to control how the voter options are displayed. A good resource for this will be the official documentation for the voter component. Reak Native Shapes - Slider The React Native Slider component is mostly used to select a single value from a variety of values. This feature is especially in shapes when you need to offer users with a variety of values from a defined minimumValue to a maximumValue. A practical application example of this component will be in product or performance ratings. To demonstrate this, we'll create a new component, build our shape and implement the slider. In the src we created earlier, create a new src/components/Slider.js file and update it with the code below: import React, {useState} from 'respond'; perform {Slider, Text, StyleSheet, View, TextInput} from 'reak native'; Const App = () => {const[value, setValue] = useState(0); yield <View> <Text>(Demo Form</Text> <View > <TextInput placeholder=Email></TextInput > <TextInput securetextentry={true} placeholder=Password></TextInput > <Text>Rate your teams performance this quarter</Text> <Slider step={1} minimumvalue={0} maximumvalue={100} value={value} unvaluechange={slideValue===>setValue (slideValue)} minimum TrackTintColor=#1fb28a maximumTrackTintColor==#d3d3d3 thumbTintColor=#b9e4c9/> <Text&gtSlide Value: {%%</Text> </Slider> </View> </View>); export default app; Here we perform the slider component of React Native Core. It will be worthy to note that the slider component is extracted from React Native core and will be removed in future releases. When this happens, the slider component will be installed directly from: npm i @react-native community/slider - except Then imported from: import Slider from '@react-indigenous community/slider'; This procedure works but still requires some rather manual processes that have not yet been well documented. Therefore, we will continue with the conventional import of React Native core in this demonstration. To deliver this component when the program and see what it looks like, we'll update the App.js file again as follows: import responds from 'respond'; Import Picker of './src/Components/Picker' into Import Slider of './src/Components/Slider' Const App = ()= > { return <Slider></Slider > (); }; Export Default App; Slider demo Start the app again, and we need to get the following output: More application examples This component has a wide range of applications that go beyond forms. Here's a screenshot of the Gmail iOS app synchronizing settings: Respond native forms – modal The modal UI component allows you to present content directly on top of a parent (fencing) view. This functionality is usually very useful when you have the need to perform a number of activities while avoiding navigation in different pages. Just like the slider component, the React Native modal component is also extracted from React Native core in the community package that is now available via npm. The big difference is the additional features that came with the react-native-modal package. Examples are animations, inline style prop, more customizing options, etc. As a result, the earlier modal component of React Native core will be decimated in future releases. Modal demo To release it further, we'll build a demo program to show how you can implement the modal component on your own. In my case, I want to show a login form in my modals when a user clicks it, but first, let's install the package of npm: npm i react-inative-modal #OR or wire add reaak native-modal then we create a Login.js file in the/components folder. In this file, we will define the form we want to deliver on the modals: import React, {useState' from 'responding'; enter {Text, View, TextInput} from 'reack native' ; const LoginForm = ()=> {const [value, setValue] = useState(0); return <View> &gtt;Text > (Login form </Text > <View > <TextInput placeholder=Enter Email></TextInput > <TextInput securetextentry={true} placeholder=Enter Password></TextInput > </View> </View >); }; export default LogonForm; It's a sign-up form that does virtually nothing. I only define the email and password fields to give you a visual understanding of the supposed usage matter. Next we will create the modal component src/components/ Modal.js and update it this way: Import React, {useState} of 'react'; import {Button, View, StyleSheet} from 'react native'; import Modale from 'react-native-modals'; import LoginForm from './Login'; const ModalDemo = ()= > { const [isModalVisible, setIsModalVisible] = useState (false); const toggleModal = ()= > { setIsModalVisible <View styles={styles.container}> <Button title=Click here to login onpress={toggleModal}></Button> <Modal <View> <LoginForm></LoginForm> <View> <Button title=Hide modal onpress={toggleModal}></Button> </View> </View> </Modal> </View> (!isModalVisible); }; terugkeer (); konststyle = StyleSheet.create({ container: flex: 1, agtergrondKleur: { { alignments: 'center', fair Feeling: 'center', };; export defaults ModalDemo; Here we first enter the React native modal component we installed earlier. We also have the same for logging in form we created to deliver on the modals. Next, we deliver the modals with buttons to show and hide the modals. Initially, the modals will be hidden. We will do this by setting up the isVisible prop of the modal component to fake. When clicking the sign-in button, we switch the toggleModal() function that changes the value of the isvisible prop of false to where. When this happens, the modals will then be visible. In the modal component, we delivered the login form and also a button to hide the modals by switching the value of the isVisible prop. There are a number of other props available for the modal component for other adjustments such as styling and animations. For example, we may decide to change the standard behavior of the modals by modifying the animation styles. E.g., let's slow down the modal exit speed when we're on the hidden modal button. We can do this with the animationOutTiming prop by setting up a higher value to it. The same applies to the animation of the modals from the top of the screen rather than below as we have seen in the last survey above. More animated props return (<View style={styles.container}> <Button title=Click here to login onpress={toggleModal}></Button> <Modal animationouttiming={1000} animationout={{'slideOutUp'} isvisible={isModalVisible}> <View> <LoginForm></LoginForm> <View style={{marginTop: 150}}=> <Button title=Hide modal onpress={toggleModal}></Button> </View> </View> </Modal > </View >); And this update will yield a different modal behavior as you would expect. You can find more available props for the modal component here. Conclusion Here we explained and demonstrated how to build better React Native forms using only native UI components. We covered the picker component, the slider, and also the modals. We have some very nice examples to give you a practical experience of how the components work and how to build yours. You can find the source code for this project here. Here.