**Forms**

Forms in React Native are just like the forms in React that you already know: the state of input form elements is controlled by the React component that renders that form. That is, form values are held in local component state, making state the "source of truth" for that form.

React Native provides a few basic components to use in your application's forms. We'll take a look at each of these more closely in the following video:

* TextInput
* KeyboardAvoidingView
* Slider
* Switch

<https://youtu.be/WxdnpxrWZkI>

## ⚠️ Oops! (onChange vs. onChangeText) ⚠️

In the above video, App renders a TextInput component with an onChange prop. With the way that the event handler, handleTextChange(), is implemented, the prop should be onChangeText.

While both methods are invoked on value change, onChangeText passes the actual value (text) as the argument. On the other hand, onChange passes the entire event object as an argument. Both are perfectly valid props, but the logic of your event handler will need to be tailored to the prop chosen. For more info, check out [**this post**](https://stackoverflow.com/questions/44416541/react-native-difference-between-onchange-vs-onchangetext-of-textinput) on Stack Overflow.

<https://youtu.be/uxbqKJchzKQ>

## Other Components

We've just seen some of the most important components built into React Native. These components will get you started with the essential functionalities in the apps that you build -- but the list of available components goes on! Feel free to review the React Native documentation for a [**complete list**](https://facebook.github.io/react-native/docs/components-and-apis.html#components-and-apis). For starters, we recommend checking out:

* [**ActivityIndicator**](https://facebook.github.io/react-native/docs/activityindicator.html)
* [**Picker**](https://facebook.github.io/react-native/docs/picker.html)
* [**WebView**](https://facebook.github.io/react-native/docs/webview.html)
* [**Modal**](https://facebook.github.io/react-native/docs/modal.html)

Note that certain components are also platform-specific! Though you want to build cross-platform components with composition, reusing as much code as possible, it may make sense for certain elements to be different depending on your audience (i.e., iOS vs. Android).

## Summary

React Native provides a variety of built-in components for developing mobile applications. While some support basic functionality in an application (e.g., text, images, lists), others offer more specialized functionality (e.g., pulling to refresh, displaying a loading indicator). Feel free to check out [**Components and APIs**](https://facebook.github.io/react-native/docs/components-and-apis.html) in the React Native documentation for an exhaustive list.