**React-Native之TextInput组件**

0.3042017.11.22 17:16:08字数 1555阅读 5085

* 文本输入框是iOS开发中最常用的控件之一。Object-C中有UITextView和UITextfield，这两个虽然都具有文本输入功能，但是部分属性还是有区别，所以这两个空间的应用场景也可能不同。而React-Native中的TextInput似乎结合了OC中两个文本输入控件的所有功能，而且还有扩展，用起来也更得心应手。

**常用属性**

* value  
  value = {'我是默认文字'} // text, 默认文字
* clearButtonMode  
  'never' ： 从不显示  
  'while-editing': 获得焦点时显示，失焦后不显示  
  'unless-editing' ：获得焦点时不显示，失焦后显示  
  'always' : 一直显示  
  clearButtonMode = {'always'} // enum, 清除按钮显示时机
* clearTextOnFocus（iOS）  
  clearTextOnFocus = {true} // bool,如果为true，每次开始输入的时候都会清除文本框的内容。
* enablesReturnKeyAutomatically（iOS）  
  enablesReturnKeyAutomatically = {true} // bool,如果为true，键盘会在文本框内没有文字的时候禁用确认按钮。默认值为false。
* dataDetectorTypes（iOS）  
  'phoneNumber : 电话号码'  
  'link : 链接'  
  'address : 邮箱'  
  'calendarEvent : 日历事件'  
  'none : 无意义'  
  'all : 以上全部支持'  
  dataDetectorTypes = {'phoneNumber'} // enum,设置 text input 内能被转化为可点击URL的数据的类型。当且仅当 multiline={true} 和 editable={false} 时起作用。默认情况下不检测任何数据类型。可接受一个类型值或类型值数组。
* placeholder  
  placeholder = {'我是占位文字'} // text, 占位文字
* placeholderTextColor  
  placeholderTextColor = {'#0094ff'} // color, 占位文字字体颜色
* keyboardType  
  所有平台可用：'default'、'numeric'、'email-address'、'phone-pad'  
  其他类型：'ascii-capable'、'numbers-and-punctuation'、'url'、'number-pad'、'name-phone-pad'、'decimal-pad'、'twitter'、'web-search'  
  keyboardType = {'number-pad'} // enum, 键盘类型
* keyboardAppearance  
  'default' ： 默认颜色  
  'light': 亮色  
  'dark' ：暗黑色  
  keyboardAppearance = {'default'} // enum, 键盘颜色
* password  
  password = {true} // bool, 密文输入，多行显示时失效
* multiline  
  multiline = {true} // bool, 超过一行时换行显示，此时密文属性失效
* maxLength  
  maxLength = {40} // number, 允许输入的最大字符长度
* editable  
  editable = {true} // bool, 允许输入
* autoCapitalize  
  'characters' : 所有字符  
  'words' : 每个单词的第一个字符  
  'sentences' : 每句话的第一个字符（默认）  
  'none' : 不自动切换任何字符为大小写  
  autoCapitalize = {'words'} // enum, 控制是否要自动将特定字 符切换为大小写
* returnKeyType  
  跨平台可用: 'done'、'go'、'next'、'search'、'send'  
  限iOS：'default'、'energency-call'、'google'、'join'、'route'、'yehoo'  
  限Android：'none'、'previous'  
  returnKeyType = {true} // enum, 键盘回车键显示内容
* blurOnSubmit  
  blurOnSubmit = {true} // bool, 如果为true，文本框会在提交的时候失焦(隐藏)。对于单行输入框默认值为true，多行则为false。注意：对于多行输入框来说，如果将blurOnSubmit设为true，则在按下回车键时就会失去焦点同时触发onSubmitEditing事件，而不会换行。
* autoCorrect  
  autoCorrect = {false} // bool, 如果为false，会关闭键盘上的拼音自动修正，默认为true
* autoFocus  
  autoFocus = {true} // bool, 如果为ture，在componenDidMount后会获得焦点。默认false
* caretHidden  
  caretHidden = {true} // bool, 如果为ture，则隐藏光标。默认false

**常用方法**

* onBlur  
  function, 当文本框失去焦点的时候调用。
* onChange  
  function, 当文本框内容变化时调用。
* onChangeText  
  function, 当文本内容变化是调用函数。改变后的文字内容会作为参数传递。
* onEndEditing  
  function, 当文本输入结束后调用此回调函数。
* onFocus  
  function, 当文本框获得焦点的时候调用此回调函数。
* onLayout  
  function, 当组件挂载或者布局变化的时候调用，参数为{x, y, width, height}。
* onScroll  
  function, 在内容滚动时持续调用，传回参数的格式形如{ nativeEvent: { contentOffset: { x, y } } }。 也可能包含其他和滚动事件相关的参数，但是在Android上，出于性能考虑，不会提供contentSize参数。
* onSubmitEditing  
  function, 此回调函数当软键盘的'确定'/'提交'按钮被按下的时候调用此函数。如果multiline={true}，此属性不可用。
* onSelectionChange  
  function, 长按选择文本时，选择范围变化时调用此函数，传回参数的格式形如 { nativeEvent: { selection: { start, end } } }。
* onKeyPress(iOS)  
  function, 当一个键被按下的时候调用此回调。传递给回调函数的参数为{ nativeEvent: { key: keyValue } }，其中keyValue即为被按下的键。会在onChange之前调用。

**创建TextInput**

* 引入TextInput组件库
* 创建
* import React, { Component } from 'react';
* import {
* AppRegistry,
* StyleSheet,
* Text,
* View,
* TextInput, // 引入组件库
* } from 'react-native';
* // 设备宽度
* var Dimensions = require('Dimensions'); // 引入内库
* var {width} = Dimensions.get('window');
* export default class TextInputAssembly extends Component {
* constructor(props) {
* super(props);
* this.state={
* text: '我是默认字符',
* };
* }
* render() {
* return (
* <View style={styles.container}>
* <TextInput
* // 样式
* style = {styles.textInputStyle}
* // 默认文字
* // value = {'12345'}
* // 占位文字
* placeholder = '我是占位文字'
* // 占位文字字体颜色
* placeholderTextColor = {'#0094ff'}
* // 是否密文输入
* // password={true}
* // 键盘类型
* keyboardType = {'default'}
* // 键盘颜色
* keyboardAppearance = {'default'}
* // 多行显示(设置多行显示时，密文属性失效)
* multiline={false}
* // 最大字符长度
* // maxLength = {40}
* // 是否允许编辑
* // editable = {true}
* // 清除按钮出现的时机
* clearButtonMode = {'unless-editing'}
* // 如果为true,文本框会在提交的时候失焦。对于单行输入框默认值为ture,多行则为false。
* // 注意：对于多行输入框来说，如果为ture情况下，则在按下回车键时就会失去焦点同事触发
* // onSubmitEditing事件，而不会换行
* blurOnSubmit = {true}
* // 如果为false，会关闭键盘上拼写自动修正。默认值是true
* autoCorrect = {false}
* // 如果为true,在componentDidMount后会获得焦点。默认值false
* autoFocus = {true}
* // 订阅用户输入
* onChangeText = {(text) => this.setState({text})}
* value = {this.state.text}
* />
* <Text style={styles.textStyle}>
* {this.state.text}
* </Text>
* </View>
* );
* }
* }
* const styles = StyleSheet.create({
* container: {
* flex: 1,
* backgroundColor: '#F5FCFF',
* },
* textInputStyle: {
* marginLeft: 20,
* marginRight: 20,
* marginTop: 50,
* height: 30,
* borderWidth: 1,
* borderColor: '#ff00b8',
* color: '#00baff',
* fontSize: 15,
* },
* textStyle: {
* margin: 20,
* alignItems: 'center',
* fontSize : 20,
* backgroundColor: '#00f7ff',
* }
* });

AppRegistry.registerComponent('TextInputAssembly', () => TextInputAssembly);