|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Snippets info Every space inside { } and ( ) means that this is pushed into next line :) $ represent each step after tab.   | **Prefix** | **Method** | | --- | --- | | imp→ | import moduleName from 'module' | | imn→ | import 'module' | | imd→ | import { destructuredModule } from 'module' | | ime→ | import \* as alias from 'module' | | ima→ | import { originalName as aliasName} from 'module' | | exp→ | export default moduleName | | exd→ | export { destructuredModule } from 'module' | | exa→ | export { originalName as aliasName} from 'module' | | enf→ | export const functionName = (params) => { } | | edf→ | export default (params) => { } | | met→ | methodName = (params) => { } | | fre→ | arrayName.forEach(element => { } | | fof→ | for(let itemName of objectName { } | | fin→ | for(let itemName in objectName { } | | anfn→ | (params) => { } | | nfn→ | const functionName = (params) => { } | | dob→ | const {propName} = objectToDescruct | | dar→ | const [propName] = arrayToDescruct | | sti→ | setInterval(() => { }, intervalTime | | sto→ | setTimeout(() => { }, delayTime | | prom→ | return new Promise((resolve, reject) => { } | | cmmb→ | comment block | | cp→ | const { } = this.props | | cs→ | const { } = this.state |  Basic MethodsReact  | **Prefix** | **Method** | | --- | --- | | imr→ | import React from 'react' | | imrd→ | import ReactDOM from 'react-dom' | | imrc→ | import React, { Component } from 'react' | | imrcp→ | import React, { Component } from 'react' & import PropTypes from 'prop-types' | | imrpc→ | import React, { PureComponent } from 'react' | | imrpcp→ | import React, { PureComponent } from 'react' & import PropTypes from 'prop-types' | | imrm→ | import React, { memo } from 'react' | | imrmp→ | import React, { memo } from 'react' & import PropTypes from 'prop-types' | | impt→ | import PropTypes from 'prop-types' | | redux→ | import { connect } from 'react-redux' | | rconst→ | constructor(props) with this.state | | rconc→ | constructor(props, context) with this.state | | est→ | this.state = { } | | cwm→ | componentWillMount = () => { } DEPRECATED!!! | | cdm→ | componentDidMount = () => { } | | cwr→ | componentWillReceiveProps = (nextProps) => { } DEPRECATED!!! | | scu→ | shouldComponentUpdate = (nextProps, nextState) => { } | | cwup→ | componentWillUpdate = (nextProps, nextState) => { } DEPRECATED!!! | | cdup→ | componentDidUpdate = (prevProps, prevState) => { } | | cwun→ | componentWillUnmount = () => { } | | gdsfp→ | static getDerivedStateFromProps(nextProps, prevState) { } | | gsbu→ | getSnapshotBeforeUpdate = (prevProps, prevState) => { } | | ren→ | render() { return( ) } | | sst→ | this.setState({ }) | | ssf→ | this.setState((state, props) => return { }) | | props→ | this.props.propName | | state→ | this.state.stateName | | rcontext→ | const ${1:contextName} = React.createContext() | | cref→ | this.${1:refName}Ref = React.createRef() | | fref→ | const ref = React.createRef() | | bnd→ | this.methodName = this.methodName.bind(this) |  React Native  | **Prefix** | **Method** | | --- | --- | | imrn→ | import { $1 } from 'react-native' | | rnstyle→ | const styles = StyleSheet.create({}) |  Redux  | **Prefix** | **Method** | | --- | --- | | rxaction→ | redux action template | | rxconst→ | export const $1 = '$1' | | rxreducer→ | redux reducer template | | rxselect→ | redux selector template |  PropTypes  | **Prefix** | **Method** | | --- | --- | | pta→ | PropTypes.array | | ptar→ | PropTypes.array.isRequired | | ptb→ | PropTypes.bool | | ptbr→ | PropTypes.bool.isRequired | | ptf→ | PropTypes.func | | ptfr→ | PropTypes.func.isRequired | | ptn→ | PropTypes.number | | ptnr→ | PropTypes.number.isRequired | | pto→ | PropTypes.object | | ptor→ | PropTypes.object.isRequired | | pts→ | PropTypes.string | | ptsr→ | PropTypes.string.isRequired | | ptnd→ | PropTypes.node | | ptndr→ | PropTypes.node.isRequired | | ptel→ | PropTypes.element | | ptelr→ | PropTypes.element.isRequired | | pti→ | PropTypes.instanceOf(name) | | ptir→ | PropTypes.instanceOf(name).isRequired | | pte→ | PropTypes.oneOf([name]) | | pter→ | PropTypes.oneOf([name]).isRequired | | ptet→ | PropTypes.oneOfType([name]) | | ptetr→ | PropTypes.oneOfType([name]).isRequired | | ptao→ | PropTypes.arrayOf(name) | | ptaor→ | PropTypes.arrayOf(name).isRequired | | ptoo→ | PropTypes.objectOf(name) | | ptoor→ | PropTypes.objectOf(name).isRequired | | ptsh→ | PropTypes.shape({ }) | | ptshr→ | PropTypes.shape({ }).isRequired | | ptany→ | PropTypes.any | | ptypes→ | static propTypes = {} |  GraphQL |graphql→|import { compose, graphql } from 'react-apollo'| expgql export default compose(graphql($1, { name: $2 }))($3) Console  | **Prefix** | **Method** | | --- | --- | | clg→ | console.log(object) | | cas→ | console.assert(expression,object) | | ccl→ | console.clear() | | cco→ | console.count(label) | | cdi→ | console.dir | | cer→ | console.error(object) | | cgr→ | console.group(label) | | cge→ | console.groupEnd() | | ctr→ | console.trace(object) | | cwa→ | console.warn | | cin→ | console.info |  React Componentsrcc import React, { Component } from 'react'  export default class FileName extends Component {  render() {  return <div>$2</div>  }  } rce import React, { Component } from 'react'  export class FileName extends Component {  render() {  return <div>$2</div>  }  }  export default $1 rcep import React, { Component } from 'react'  import PropTypes from 'prop-types'  export class FileName extends Component {  static propTypes = {}  render() {  return <div>$2</div>  }  }  export default $1 rpc import React, { PureComponent } from 'react'  export default class FileName extends PureComponent {  render() {  return <div>$2</div>  }  } rpcp import React, { PureComponent } from 'react'  import PropTypes from 'prop-types'  export default class FileName extends PureComponent {  static propTypes = {}  render() {  return <div>$2</div>  }  } rpce import React, { PureComponent } from 'react'  import PropTypes from 'prop-types'  export class FileName extends PureComponent {  static propTypes = {}  render() {  return <div>$2</div>  }  }  export default FileName rccp import React, { Component } from 'react'  import PropTypes from 'prop-types'  export default class FileName extends Component {  static propTypes = {  $2: $3,  }  render() {  return <div>$4</div>  }  } rfcp import React from 'react'  import PropTypes from 'prop-types'  function $1(props) {  return <div>$0</div>  }  $1.propTypes = {}  export default $1 rfc import React from 'react'  export default function $1() {  return <div>$0</div>  } rfce import React from 'react'  function $1() {  return <div>$0</div>  }  export default $1 rafcp import React from 'react'  import PropTypes from 'prop-types'  const $1 = props => {  return <div>$0</div>  }  $1.propTypes = {}  export default $1 rafc import React from 'react'  const $1 = () => {  return <div>$0</div>  }  export default $1 rafce import React from 'react'  const $1 = () => {  return <div>$0</div>  }  export default $1 rmc import React, { memo } from 'react'  export default memo(function $1() {  return <div>$0</div>  }) rmcp import React, { memo } from 'react'  import PropTypes from 'prop-types'  const $1 = memo(function $1(props) {  return <div>$0</div>  })  $1.propTypes = {}  export default $1 rcredux import React, { Component } from 'react'  import PropTypes from 'prop-types'  import { connect } from 'react-redux'  export class FileName extends Component {  static propTypes = {  $2: $3,  }  render() {  return <div>$4</div>  }  }  const mapStateToProps = state => ({})  const mapDispatchToProps = {}  export default connect(  mapStateToProps,  mapDispatchToProps,  )(FileName) reduxmap const mapStateToProps = state => ({})  const mapDispatchToProps = {} React Native Componentsrnc import React, { Component } from 'react'  import { Text, View } from 'react-native'  export default class FileName extends Component {  render() {  return (  <View>  <Text> $2 </Text>  </View>  )  }  } rncs import React, { Component } from 'react'  import { Text, StyleSheet, View } from 'react-native'  export default class FileName extends Component {  render() {  return (  <View>  <Text> $2 </Text>  </View>  )  }  }  const styles = StyleSheet.create({}) rnce import React, { Component } from 'react'  import { Text, View } from 'react-native'  export class FileName extends Component {  render() {  return (  <View>  <Text> $2 </Text>  </View>  )  }  }  export default $1 rncredux import React, { Component } from 'react'  import { View, Text } from 'react-native'  import PropTypes from 'prop-types'  import { connect } from 'react-redux'  export class FileName extends Component {  static propTypes = {  $2: $3,  }  render() {  return (  <View>  <Text> $2 </Text>  </View>  )  }  }  const mapStateToProps = state => ({})  const mapDispatchToProps = {}  export default connect(  mapStateToProps,  mapDispatchToProps,  )(FileName) Otherscmmb /\*\*  |--------------------------------------------------  | $1  |--------------------------------------------------  \*/ desc describe('$1', () => {  $2  }) test test('should $1', () => {  $2  }) tit it('should $1', () => {  $2  }) stest import React from 'react'  import renderer from 'react-test-renderer'  import { ${1:ComponentName} } from '../${1:ComponentName}'  describe('<${1:ComponentName} />', () => {  const defaultProps = {}  const wrapper = renderer.create(<${1:ComponentName} {...defaultProps} />)  test('render', () => {  expect(wrapper).toMatchSnapshot()  })  }) srtest import React from 'react'  import renderer from 'react-test-renderer'  import { Provider } from 'react-redux'  import store from 'src/store'  import { ${1:ComponentName} } from '../${1:ComponentName}'  describe('<${1:ComponentName} />', () => {  const defaultProps = {}  const wrapper = renderer.create(  <Provider store={store}>  <${1:${TM\_FILENAME\_BASE}} {...defaultProps} />)  </Provider>,  )  test('render', () => {  expect(wrapper).toMatchSnapshot()  })  }) sntest import 'react-native'  import React from 'react'  import renderer from 'react-test-renderer'  import ${1:ComponentName} from '../${1:ComponentName}'  describe('<${1:ComponentName} />', () => {  const defaultProps = {  }  const wrapper = renderer.create(<${1:ComponentName} {...defaultProps} />)  test('render', () => {  expect(wrapper).toMatchSnapshot()  })  }) snrtest import 'react-native'  import React from 'react'  import renderer from 'react-test-renderer'  import { Provider } from 'react-redux'  import store from 'src/store/configureStore'  import ${1:ComponentName} from '../${1:ComponentName}'  describe('<${1:ComponentName} />', () => {  const defaultProps = {}  const wrapper = renderer.create(  <Provider store={store}>  <${1:ComponentName} {...defaultProps} />  </Provider>,  )  test('render', () => {  expect(wrapper).toMatchSnapshot()  })  }) hocredux import React from 'react'  import PropTypes from 'prop-types'  import { connect } from 'react-redux'  export const mapStateToProps = state => ({  })  export const mapDispatchToProps = {  }  export const ${1:hocComponentName} = (WrappedComponent) => {  const hocComponent = ({ ...props }) => <WrappedComponent {...props} />  hocComponent.propTypes = {  }  return hocComponent  }  export default WrapperComponent => connect(mapStateToProps, mapDispatchToProps)(${1:hocComponentName}(WrapperComponent)) hoc import React from 'react'  import PropTypes from 'prop-types'  export default WrappedComponent => {  const hocComponent = ({ ...props }) => <WrappedComponent {...props} />  hocComponent.propTypes = {}  return hocComponent  } |