Snippets info

Every space inside { }; and ( ) means that this is pushed into next line :) $ represent each step after tab.

Basic Methods

| **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 |
| fregion→ | full region block |

React Native JavaScript

| **Prefix** | **Method** |
| --- | --- |
| imr→ | import React from 'react'; |
| imrc→ | import React, { Component } from 'react'; |
| imrn→ | import { $1 } from 'react-native'; |
| imrpc→ | import React, { PureComponent } from 'react'; |

StyleSheet

| **Prefix** | **Method** |
| --- | --- |
| rnss→ | const styles = StyleSheet.create({}); |
| just→ | justifyContent: '', |
| align→ | alignItems: '${1}', |
| as→ | aspectRatio: '', |
| bor→ | borderWidth: , |
| flex→ | flexDirection: '', |
| h→ | height: , |
| w→ | width: , |
| l→ | left: '', |
| mar→ | marginHorizontal: '', |
| max→ | maxWidth: , |
| min→ | minWidth: , |
| over→ | overflow: , |
| padding→ | paddingHorizontal: , |
| pos→ | position: , |
| ri→ | right: , |
| z→ | zIndex: , |
| di→ | direction: , |
| back→ | backgroundColor: , |
| sha→ | shadowColor: , |
| op→ | opacity: , |
| e→ | elevation: , |

React

| **Prefix** | **Method** |
| --- | --- |
| imr→ | import React from 'react'; |
| 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'; |
| redux→ | import { connect } from 'react-redux'; |
| rconst→ | constructor(props) with this.state |
| rconc→ | constructor(props, context) with this.state |
| est→ | this.state = { }; |
| cwm→ | componentWillMount = () => { }; |
| cdm→ | componentDidMount = () => { }; |
| cwr→ | componentWillReceiveProps = (nextProps) => { }; |
| scu→ | shouldComponentUpdate = (nextProps, nextState) => { }; |
| cwup→ | componentWillUpdate = (nextProps, nextState) => { }; |
| cdup→ | componentDidUpdate = (prevProps, prevState) => { }; |
| cwun→ | componentWillUnmount = () => { }; |
| ren→ | render() { return( ) }; |
| sst→ | this.setState({ }) |
| ssf→ | this.setState((state, props) => return { }) |
| props→ | this.props.propName |
| state→ | this.state.stateName |

Redux

| **Prefix** | **Method** |
| --- | --- |
| rx→ | import { connect } from 'react-redux'; |
| rxaction→ | redux action template |
| rxconst→ | export const $1 = '$1'; |
| rxmap→ | mapping to props template |
| rxreducer→ | redux reducer template |
| rxselect→ | redux selector template |

PropTypes

| **Prefix** | **Method** |
| --- | --- |
| ipt→ | import PropTypes from 'prop-types'; |
| pt→ | Component.PropTypes = {}; |
| dfp→ | Component.defaultProps = {}; |
| 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 |

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 Native Components

rnc

import React, { Component } from 'react';  
import { Text, View } from 'react-native';  
  
export default class $1 extends Component {  
 constructor(props) {  
 super(props);  
 this.state = {  
 };  
 }  
  
 render() {  
 return (  
 <View>  
 <Text> $2 </Text>  
 </View>  
 )  
 }  
}

rnce

import React, { Component } from 'react';  
import { Text, View } from 'react-native';  
  
export class $1 extends Component {  
 constructor(props) {  
 super(props);  
 this.state = {  
 };  
 }  
  
 render() {  
 return (  
 <View>  
 <Text> $2 </Text>  
 </View>  
 )  
 }  
}  
  
export default $1

rnpc

import React, { PureComponent } from 'react';  
import { Text, View } from 'react-native';  
  
export default class $1 extends PureComponent {  
 constructor(props) {  
 super(props);  
 this.state = {  
 };  
 }  
  
 render() {  
 return (  
 <View>  
 <Text> $2 </Text>  
 </View>  
 )  
 }  
}

rnpce

import React, { PureComponent } from 'react';  
import { Text, View } from 'react-native';  
  
class $1 extends PureComponent {  
 constructor(props) {  
 super(props);  
 this.state = {  
 };  
 }  
  
 render() {  
 return (  
 <View>  
 <Text> $2 </Text>  
 </View>  
 )  
 }  
}  
  
export default $1

rncsl

import React from 'react';  
import { Text, View } from 'react-native';  
  
const $1 = ({  
 params1,  
}) => (  
 <View>  
 <Text>$2</Text>  
 </View>  
);  
  
export default $1;

Typescript React Native

tsrnc

import \* as React from 'react';  
import { View, StyleSheet, Text, } from 'react-native';  
  
export interface $1Props {  
}  
  
export default class $1 extends React.Component<$1Props, any> {  
 constructor(props: $1Props) {  
 super(props);  
 }  
  
 render() {  
 return (  
 <View>  
 <Text>  
 $2  
 </Text>  
 </View>  
 );  
 }  
}

tsrnc

import \* as React from 'react';  
import { View, StyleSheet, Text, } from 'react-native';  
  
export interface $1Props {  
}  
  
export interface $1State {  
}  
  
export default class $1 extends React.Component<$1Props, $1State> {  
 constructor(props: $1Props) {  
 super(props);  
 this.state = {  
 };  
 }  
  
 render() {  
 return (  
 <View>  
 <Text>$1</Text>  
 </View>  
 );  
 }  
}

tsrnpc

import \* as React from 'react';  
import { View, StyleSheet, Text } from 'react-native';  
  
export interface AppProps {  
}  
  
export default class App extends React.PureComponent<AppProps, any> {  
 constructor(props: AppProps) {  
 super(props);  
 }  
  
 public render() {  
 return (  
 <View>  
 <Text>  
 App  
 </Text>  
 </View>  
 );  
 }  
}

tsrncsl

import \* as React from 'react';  
import { View, Text } from 'react-native';  
  
export interface AppProps {  
}  
  
export function App (props: AppProps) {  
 return (  
 <View>  
 <Text>App</Text>  
 </View>  
 );  
}

Others

cmmb

/\*\*  
|--------------------------------------------------  
| $1  
|--------------------------------------------------  
\*/

desc

describe('$1', () => {  
 $2  
})

test

test('should $1', () => {  
 $2  
})

tit

it('should $1', () => {  
 $2  
})

stest

import { ${1: ComponentName }, mapStateToProps, mapDispatchToProps } from '${2:path}/${1:ComponentName}'  
  
describe('<${1:ComponentName} />', () => {  
 const defaultProps = {  
  
 }  
  
 const setup = buildSetup(${1: ComponentName }, defaultProps)  
  
 test('render', () => {  
 expect(setup().wrapper).toMatchSnapshot()  
 })  
})

sjtest

import toJson from 'enzyme-to-json'  
import { ${1:ComponentName} }, mapStateToProps, mapDispatchToProps } from '${2:path}/${1:ComponentName}'  
  
describe('<${1:ComponentName} />', () => {  
 const defaultProps = {  
  
 }  
  
 const setup = buildSetup(${1: ComponentName }, defaultProps)  
  
 test('render', () => {  
 expect(toJson(setup().wrapper)).toMatchSnapshot()  
 })  
})

sntest

import 'react-native'  
import React from 'react'  
import renderer from 'react-test-renderer'  
  
import ${1:ComponentName} from '../${1:ComponentName}'  
  
it('renders correctly', () => {  
 const tree = renderer.create(<${1:ComponentName} />).toJSON()  
  
 expect(tree).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  
}

Lorem Ipsum

| **Prefix** | **Method** |
| --- | --- |

li1→

Lorem Ipsum is simply dummy text of the printing and typesetting industry.

li2→

Lorem Ipsum is simply dummy text of the printing and typesetting industry.  
Lorem Ipsum has been the industry standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.

li4→

Lorem Ipsum is simply dummy text of the printing and typesetting industry.  
Lorem Ipsum has been the industry standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.  
It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged.  
It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

li5→

Lorem Ipsum is simply dummy text of the printing and typesetting industry.  
Lorem Ipsum has been the industry standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book.  
It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged.  
It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.  
The standard chunk of Lorem Ipsum used since the 1500s is reproduced below for those interested. Sectio