Snippets

Snippets info

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

TypeScript has own components and own snippets. Use search or just type `ts` before every component snippet.

I.E. `tsrcc`

React Hooks

- Hooks from official docs are added with hook name as prefix.

Basic Methods

Prefix	Method

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) => { }`
`ednf→`	`export default function functionName(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`

React

Prefix	Method
`imr→`	`import React from 'react'`
`imrd→`	`import ReactDOM from 'react-dom'`
`imrc→`	`import React, { Component } from 'react'`
`imrpc→`	`import React, { PureComponent } from 'react'`
`imrm→`	`import React, { memo } from 'react'`
`imrr→`	`import { BrowserRouter as Router, Route, NavLink} from 'react-router-dom'`
`imbr→`	`import { BrowserRouter as Router} from 'react-router-dom'`
`imbrc→`	`import { Route, Switch, NavLink, Link } from react-router-dom'`
`imbrr→`	`import { Route } from 'react-router-dom'`
`imbrs→`	`import { Switch } from 'react-router-dom'`
`imbrl→`	`import { Link } from 'react-router-dom'`
`imbrnl→`	`import { NavLink } from 'react-router-dom'`
`imrs→`	`import React, { useState } from 'react'`
`imrse→`	`import React, { useState, useEffect } from 'react'`
`redux→`	`import { connect } from 'react-redux'`
`est→`	`this.state = { }`
`cdm→`	`componentDidMount = () => { }`
`scu→`	`shouldComponentUpdate = (nextProps, nextState) => { }`
`cdup→`	`componentDidUpdate = (prevProps, prevState) => { }`
`cwun→`	`componentWillUnmount = () => { }`
`gdsfp→`	`static getDerivedStateFromProps(nextProps, prevState) { }`
`gsbu→`	`getSnapshotBeforeUpdate = (prevProps, prevState) => { }`
`sst→`	`this.setState({ })`

Prefix	Method
`ssf→`	`this.setState((state, props) => return { })`
`props→`	`this.props.propName`
`state→`	`this.state.stateName`
`rcontext→`	`const $1 = React.createContext()`
`cref→`	`this.$1Ref = 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`
`rxslice→`	`redux slice template`

PropTypes

Prefix	Method
`pta→`	`PropTypes.array`
`ptar→`	`PropTypes.array.isRequired`

Prefix	Method
`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({ })`

Prefix	Method
`ptshr→`	`PropTypes.shape({ }).isRequired`
`ptany→`	`PropTypes.any`
`ptypes→`	`static propTypes = {}`

Console

Prefix	Method
`clg→`	`console.log(object)`
`clo→`	``console.log(`object`, object)``
`clj→`	``console.log(`object`, JSON.stringify(object, null, 2))``
`ctm→`	``console.time(`timeId`)``
`cte→`	``console.timeEnd(`timeId`)``
`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 Components

`rcc`

```
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'

export const $1 = () => {
  return <div>$0</div>
}
```

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 { connect } from 'react-redux'

export class FileName extends Component {
  render() {
    return <div>$4</div>
  }
}

const mapStateToProps = (state) => ({})

const mapDispatchToProps = {}

export default connect(mapStateToProps, mapDispatchToProps)(FileName)
```

rcreduxp

```
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)
```

rfcredux

```
import React, { Component } from 'react'
import { connect } from 'react-redux'

export const FileName = () => {
  return <div>$4</div>
}

const mapStateToProps = (state) => ({})

const mapDispatchToProps = {}

export default connect(mapStateToProps, mapDispatchToProps)(FileName)
```

rfreduxp

```
import React, { Component } from 'react'
import PropTypes from 'prop-types'
import { connect } from 'react-redux'

export const FileName = () => {
  return <div>$4</div>
}

FileName.propTypes = {
  $2: $3,
}

const mapStateToProps = (state) => ({})

const mapDispatchToProps = {}

export default connect(mapStateToProps, mapDispatchToProps)(FileName)
```

reduxmap

```
const mapStateToProps = (state) => ({})

const mapDispatchToProps = {}
```

React Native Components

rnc

```
import React, { Component } from 'react'
import { Text, View } from 'react-native'

export default class FileName extends Component {
  render() {
    return (
      <View>
        <Text> $2 </Text>
      </View>
    )
  }
}
```

rnf

```
import React from 'react'
import { View, Text } from 'react-native'

export default function $1() {
  return (
    <View>
      <Text> $2 </Text>
    </View>
  )
}
```

rnfs

```
import React from 'react'
import { StyleSheet, View, Text } from 'react-native'

export default function $1() {
  return (
    <View>
      <Text> $2 </Text>
    </View>
  )
}

const styles = StyleSheet.create({})
```

rnfe

```
import React from 'react'
import { View, Text } from 'react-native'

const $1 = () => {
  return (
    <View>
      <Text> $2 </Text>
    </View>
  )
}

export default $1
```

rnfes

```
import React from 'react'
import { StyleSheet, View, Text } from 'react-native'
```

```
const $1 = () => {
  return (
    <View>
      <Text> $2 </Text>
    </View>
  )
}

export default $1

const styles = StyleSheet.create({})
```

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
```

Others

`cmmb`

```
/**
|--------------------------------------------------
| $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 } from '../$1'
```

```
describe('<$1 />', () => {
  const defaultProps = {}
  const wrapper = renderer.create(<$1 {...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 } from '../$1'

describe('<$1 />', () => {
  const defaultProps = {}
  const wrapper = renderer.create(
    <Provider store={store}>
      <$1 {...defaultProps} />)
    </Provider>,
  )

  test('render', () => {
    expect(wrapper).toMatchSnapshot()
  })
})
```

sntest

```
import 'react-native'
import React from 'react'
import renderer from 'react-test-renderer'

import $1 from '../$1'

describe('<$1 />', () => {
  const defaultProps = {}

  const wrapper = renderer.create(<$1 {...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 from '../$1'

describe('<$1 />', () => {
  const defaultProps = {}
  const wrapper = renderer.create(
    <Provider store={store}>
      <$1 {...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 = (WrappedComponent) => {
  const hocComponent = ({ ...props }) => <WrappedComponent {...props} />

  hocComponent.propTypes = {}

  return hocComponent
}

export default (WrapperComponent) =>
  connect(mapStateToProps, mapDispatchToProps)($1(WrapperComponent))
```

hoc

```
import React from 'react'
import PropTypes from 'prop-types'

export default (WrappedComponent) => {
  const hocComponent = ({ ...props }) => <WrappedComponent {...props} />

  hocComponent.propTypes = {}

  return hocComponent
}
```