

Basic Methods

```
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 {propertyName} = objectToDeconstruct
dar→ const [propertyName] = arrayToDeconstruct
sti→ setInterval(() => { }, intervalTime
sto→ setTimeout(() => { }, delayTime
prom→ return new Promise((resolve, reject) => {
}
cmmb→ comment block
```

PropTypes

```
pta→ PropTypes.array
ptar→ PropTypes.array.isRequired
ptb→ PropTypes.bool
ptbr→ PropTypes.boolean.isRequired
ptf→ PropTypes.func
ptfr→ PropTypes.function.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)
```

PropTypes (cont)

```
> 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 = {}
```

Components

```
rfc
import React from 'react'
export default function $1() {
  return <div v>$ 0</ div>
}
rfcp
import React from 'react'
import PropTypes from 'prop-types'
function $1(props) {
  return <div v>$ 0</ div>
}
$1.propTypes = {}
export default $1
rfce
import React from 'react'
function $1() {
  return <div v>$ 0</ div>
}
export default $1
```

React

```
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'
```

React (cont)

```
> 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'
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'
imbr→ 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'
cdm→ componentDidMount = () => { }
scu→ shouldComponentUpdate = (nextProps, nextState) => { }
cdup→ componentDidMount = (prevProps, prevState) => { }
cwun→ componentWillUnmount = () => { }
gdsfp→ static getDerivedStateFromProps(nextProps, prevState) { }
gsbu→ getSnapshotBeforeUpdate = (prevProps, prevState) => { }
```

Redux

```
rxaction→ redux action template
rxconst→ export const $1 = '$1'
rxreducer→ redux reducer template
rxselect→ redux selector template
rxslice→ redux slice template
```

Console

```
clg→ console.log(object)
clo→ console.log( 'object', object)
ctm→ console.time( 'timeId')
cte→ console.timeEnd( 'timeId')
cas→ console.assert( exp res sio n,o bject)
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)
```

Console (cont)

```
> cwa→ console.warn
cin→ console.info
```

Components

```
rcredux
import React, { Component } from 'react'
import { connect } from 'react -redux'
export const FileName = () => {
  return <div v=$ 4</ div>
}
const mapStateToProps = (state) => ({}))
const mapDispatchToProps = {}
export default connect( mapS tat eTo Props,
mapDis pat chT oPr ops )(F ile Name)
rfreduxp
import React, { Component } from 'react'
import PropTypes from 'prop- types'
import { connect } from 'react -redux'
export const FileName = () => {
  return <div v=$ 4</ div>
}
FileNa me.p ro pTypes = {
  $2: $3,
}
const mapStateToProps = (state) => ({}))
const mapDispatchToProps = {}
export default connect( mapS tat eTo Props,
mapDis pat chT oPr ops )(F ile Name)
reduxmap
const mapStateToProps = (state) => ({}))
const mapDispatchToProps = {}
```