

Basic Methods	Basic Methods (cont)	React (cont)	React (cont)
imp → import moduleName from 'module'	sto → setTimeout(() => {}, delayTime)	imrpc → import React, { PureComponent } from 'react'	gsbu → getSnapshotBeforeUpdate = (prevProps, prevState) => {}
imn → import 'module'	prom → return new Promise((-resolve, reject) => {})	imrpcp → import React, { PureComponent } from 'react' & import PropTypes from 'prop-types'	ren → render() { return() }
imd → import { destructured-Module } from 'module'	cmmb → comment block	redux → import { connect } from 'react-redux'	sst → this.setState({})
ime → import * as alias from 'module'	Console	rconst → constructor(props) with this.state	ssf → this.setState((state, props) => return { })
ima → import { originalName as aliasName } from 'module'	clg → console.log(object)	rconc → constructor(props, context) with this.state	props → propName
exp → export default moduleName	cas → console.assert(expression, object)	est → this.state = {}	state → this.state.stateName
exd → export { destructured-Module } from 'module'	ccl → console.clear()	cwm → componentWillMount = () => {} DEPRECATED!!!	rcontext → const \${1:contextName} = React.createContext()
exa → export { originalName as aliasName } from 'module'	cco → console.count(label)	cdm → componentDidMount = () => {}	cref → this.\${1:refName}Ref = React.createRef()
enf → export const functionName = (params) => {}	cdi → console.dir	cwr → componentWillReceiveProps = (nextProps) => {} DEPRECATED!!!	fred → const ref = React.createRef()
edf → export default (params) => {}	cer → console.error(object)	scu → shouldComponentUpdate = (nextProps, nextState) => {}	bnd → this.methodName = this.methodName.bind(this)
met → methodName = (params) => {}	cgr → console.group(label)	cwup → componentWillUpdate = (nextProps, nextState) => {} DEPRECATED!!!	Redux
fre → arrayName.forEach(-element => {})	cge → console.groupEnd()	cdup → componentDidUpdate = (prevProps, prevState) => {}	rxaction → redux action template
fof → for(let itemName of objectName { })	ctr → console.trace(object)	cwun → componentWillUnmount = () => {}	rxconst → export const \$1 = '\$1'
fin → for(let itemName in objectName { })	ctr → console.warn	gdsfp → static getDerivedStateFromProps(nextProps, prevState) { }	rxreducer → redux reducer template
anfn → (params) => {}	cin → console.info		rxselect → redux selector template
nfn → const functionName = (params) => {}	React		React Native
dob → const {propName} = objectToDescruct	imr → import React from 'react'		imm → import { \$1 } from 'react-native'
dar → const [propName] = arrayToDescruct	imr → import React, { Component } from 'react'		mstyle → const styles = StyleSheet.create({})
**sti → setInterval(() => {}, intervalTime)	imr → import React, { Component } from 'react' & import PropTypes from 'prop-types'		



GraphQL

```

graphql→ import { compose,
            graphql } from
            'react-apollo'

expgql→ export default
            compose(grap-
            hql($1, { name: $2
            })))($3)
  
```

PropTypes

```

pta→ PropTypes.array
ptar→ PropTypes.array.is-
        Required
ptb→ PropTypes.bool
ptbr→ PropTypes.bool.isR-
        equired
ptf→ PropTypes.func
ptptr→ PropTypes.func.isR-
        equired
ptn→ PropTypes.number
ptnr→ PropTypes.number.i-
        sRequired
pto→ PropTypes.object
ptor→ PropTypes.object.i-
        sRequired
pts→ PropTypes.string
ptsr→ PropTypes.string.isRe-
        quired
ptnd→ PropTypes.node
ptndr→ PropTypes.node.isR-
        equired
ptel→ PropTypes.element
ptelr→ PropTypes.element.i-
        sRequired
  
```

PropTypes (cont)

```

pti→ PropTypes.instance-
        Of(name)
ptir→ PropTypes.instance-
        Of(name).isRequired
pte→ PropTypes.oneOf([In-
        ame])
pter→ PropTypes.oneOf([In-
        ame]).isRequired
ptet→ PropTypes.oneOf-
        Type([name])
ptetr→ PropTypes.oneOf-
        Type([name]).isReq-
        uired
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 = {}
  
```



By **Pedrecal**
cheatography.com/pedrecal/

Not published yet.
 Last updated 31st May, 2018.
 Page 2 of 2.

Sponsored by **Readable.com**
 Measure your website readability!
<https://readable.com>