React.js cheatsheet

React is a JavaScript library for building user interfaces. This guide targets React v15 to v16.

Components

```
import React from 'react' import ReactDOM from 'react-dom'
```

```
class Hello extends React.Component { render () { return <div className='message-box'> Hello {this.props.name} </div> } }
```

```
const el = document.body ReactDOM.render(<Hello name='John' />, el)
```

Use the React.js jsfiddle to start hacking. (or the unofficial jsbin)

Import multiple exports

```
import React, {Component} from 'react' import ReactDOM from 'react-dom'
```

```
class Hello extends Component { ... }
```

States

```
constructor(props) { super(props) this.state = { username: undefined } }
```

```
this.setState({ username: 'rstacruz' })
```

```
render () { this.state.username const { username } = this.state ··· }
```

Use states (`this.state`) to manage dynamic data.

With Babel you can use proposal-class-fields and get rid of constructor

```
class Hello extends Component { state = { username: undefined }; ... }
```

See: States

Children

```
<AlertBox> <h1>You have pending notifications</h1> </AlertBox>
```

```
class AlertBox extends Component { render () { return <div className='alert-box'> {this.props.children} </div> } }
```

Children are passed as the `children` property.

Defaults

Setting default props

```
Hello.defaultProps = { color: 'blue' }
```

See: defaultProps

Setting default state

```
class Hello extends Com
{ super(props) this.sta
```

Set the default state in the

And without constructor u
fields.

```
class Hello extends Com
true } } }
```

See: Setting the default sta

Other components

Functional components

```
function MyComponent ({ name }) { return <div
className='message-box'> Hello {name} </div> }
```

Functional components have no state. Also, their props are passed as the first parameter to a function.

See: Function and Class Components

Pure components

```
import React, {PureComponent} from 'react' class
MessageBox extends PureComponent { ··· }
```

Performance-optimized version of `React.Component`. Doesn't rerender if props/state hasn't changed.

See: Pure components

Lifecycle

Mounting

`constructor` (props)	Before rendering #
`componentWillMount()`	Don't use this #
`render()`	Render #
`componentDidMount()`	After rendering (DOM available) #
`componentWillUnmount()`	Before DOM removal #
`componentDidCatch()`	Catch errors (16+) #

Set initial the state on `constructor()`. Add DOM event handlers, timers (etc) on `componentDidMount()`, then remove them on `componentWillUnmount()`.

Updating

`componentDidUpdate`	
`shouldComponentUpd`	
`render()`	
`componentDidUpdate`	

Called when parents chan
These are not called for in

See: Component specs

DOM nodes

References

```
class MyComponent extends Component { render () {
return <div> <input ref={el => this.input = el} />
</div> } componentDidMount () { this.input.focus()
} }
```

DOM Events

```
class MyComponent exten
<input type="text" valu
onChange={event => this
onChange (event) { this
```

Allows access to DOM nodes.

See: Refs and the DOM

```
event.target.value }) }
```

Pass functions to attribute

See: Events

Other features

Transferring props

```
<VideoPlayer src="video.mp4" />
```

```
class VideoPlayer extends Component { render () {
return <VideoEmbed {...this.props} /> } }
```

Propagates src="..." down to the sub-component.

See Transferring props

Top-level API

```
React.createClass({ ...
```

```
ReactDOM.render(<Compon
ReactDOM.unmountCompone
```

```
ReactDOMServer.renderTo
ReactDOMServer.renderTo
```

There are more, but these

See: React top-level API

JSX patterns

Style shorthand

```
const style = { height: 10 } return <div style=
{style}></div>
```

```
return <div style={{ margin: 0, padding: 0 }}>
</div>
```

See: Inline styles

Inner HTML

```
function markdownify()
<div dangerouslySetInne
markdownify()}} />
```

See: Dangerously set inne

Lists

Conditionals

```
<Fragment> {showMyComponent ? <MyComponent /> :
<OtherComponent />} </Fragment>
```

```
class TodoList extends
const { items } = this.
{items.map(item => <Tod
{item.key} />)} </ul> }
```

Always supply a key prope

Short-circuit evaluation

```
<Fragment> {showPopup && <Popup />} ... </Fragment>
```

New features

Returning multiple elements

You can return multiple elements as arrays or fragments.

Arrays

```
render () { // Don't forget the keys! return [ <li
key="A">First item</li>, <li key="B">Second
item</li> ] }
```

Fragments

```
render () { // Fragments don't require keys! return
( <Fragment> <li>First item</li> <li>Second
item</li> </Fragment> ) }
```

See: Fragments and strings

Returning strings

```
render() { return 'Look ma, no spans!'; }
```

You can return just a string.

See: Fragments and strings

Portals

```
render () { return React.createPortal(
this.props.children,
document.getElementById('menu') ) }
```

This renders `this.props.children` into any location in the DOM.

See: Portals

Property validation

PropTypes

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

See: Typechecking with PropTypes

`any`	Anything
Basic	
`string`	
`number`	
`func`	Function
`bool`	True or false
Enum	
`oneOf`(any)	Enum types
`oneOfType`(type array)	Union
Array	
`array`	
`arrayOf`(...)	
Object	
`object`	
`objectOf`(...)	Object with values of a certain type
`instanceOf`(...)	Instance of a class
`shape`(...)	
Elements	
`element`	React element
`node`	DOM node
Required	
`(···).isRequired`	Required

Basic types

```
MyComponent.propTypes = { email: PropTypes.string,
seats: PropTypes.number, callback: PropTypes.func,
isClosed: PropTypes.bool, any: PropTypes.any }
```

Enumerables (oneOf)

```
MyCo.propTypes = { direction: PropTypes.oneOf([
'left', 'right' ]) }
```

Custom validation

```
MyCo.propTypes = { customProp: (props, key,
componentName) => { if
(!/matchme/.test(props[key])) { return new
Error('Validation failed!') } } }
```

Also see

- React website (reactjs.org)
- React cheatsheet (reactcheatsheet.com)
- Awesome React (github.com)
- React v0.14 cheatsheet Legacy version

19 Comments for this cheatsheet. Write yours!

devhints.io / **Search 380+ cheatsheets**

Over 380 curated cheatsheets, by developers for developers.

Devhints home

Other React cheatsheets

- Redux cheatsheet
- Enzyme cheatsheet
- Enzyme v2 cheatsheet
- Awesome Redux cheatsheet
- Flux architecture cheatsheet
- React-router cheatsheet

Top cheatsheets

- Elixir cheatsheet
- ES2015+ cheatsheet
- Vimdiff cheatsheet
- Vim cheatsheet
- Vim scripting cheatsheet
- Capybara cheatsheet