DEVHINTS.IO

Edit

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 () {
```

Children

```
<AlertBox>

</AlertBox>
```

```
class AlertBox extends Component {
  render () {
    return <div className='alert-box'>
```

```
    </div>
  }
}
```

Children are passed as the `children` property.

≠ Defaults

Setting default props

```
  color: 'blue'
}
```

See: defaultProps

Setting default

```
class Hello ext
  constructor (
    super(props

  }
}
```

Set the default sta

And without const

```
class Hello ext
}
```

See: Setting the de

Other components

Functional components

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

Functional components have no state. Also, their `props` are pa

See: Function and Class Components

Pure components

```
import React, {PureComponent} from 'react

  ...
}
```

Performance-optimized version of `React.Compone`

See: Pure components

Lifecycle

Mounting

`constructor` (props)	Bef
`componentWillMount()`	
`render()`	
`componentDidMount()`	After rendering (D

Updating

`componentDidUpc`
`shouldComponent`
`render()`
`componentDidUpc`

`componentWillUnmount()`	Before
`componentDidCatch()`	Catc

Called when parer

See: Component s

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

≠ Hooks (New)

State Hook

```
import React, { useState } from 'react';

function Example() {
  // Declare a new state variable, which we'll call "count"


  return (
    <div>
      <p>You clicked {count} times</p>

        Click me
      </button>
    </div>
  );
}
```

Hooks are a new addition in React 16.8.

See: Hooks at a Glance

Declaring multi

```
function Exampl
  // Declare mu
  const [age, s
  const [fruit,
  const [todos,
  // ...
}
```

Effect hook

```
import React, {

function Exampl
  const [count,

  return (
    <div>
      <p>You cl
```

Building your own hooks

Define FriendStatus

```
import React, { useState, useEffect } from 'react';

function FriendStatus(props) {
  const [isOnline, setIsOnline] = useState(null);

  useEffect(() => {
```

```
    function handleStatusChange(status) {
      setIsOnline(status.isOnline);
    }



    };
  }, [props.friend.id]);

  if (isOnline === null) {
    return 'Loading...';
  }
  return isOnline ? 'Online' : 'Offline';
}
```

Effects may also optionally specify how to “clean up” after them by returning a function.

Use FriendStatus

```
function FriendStatus(props) {


  if (isOnline === null) {
    return 'Loading...';
  }
  return isOnline ? 'Online' : 'Offline';
}
```

See: Building Your Own Hooks

Hooks API Refe

Also see: Hooks FA

Basic Hooks

useState(initialStat

useEffect(() => { .

useContext(MyCo

Full details: Basic H

Additional Hooks

useReducer(reduc

useCallback(() =>

useMemo(() => { ... })

useRef(initialValue)

useImperativeHa

useLayoutEffect

useDebugValue(va

Full details: Additi

DOM nodes

References

```
class MyComponent extends Component {
  render () {
    return <div>

    </div>
  }

  componentDidMount () {

  }
}
```

Allows access to DOM nodes.

See: Refs and the DOM

DOM Events

```
class MyCompone
  render () {
    <input type
        value={

  }

  onChange (eve

  }
}
```

Pass functions to a

See: Events

Other features

Transferring props

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

```
class VideoPlayer extends Component {
  render () {

  }
}
```

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

See Transferring props

Top-level API

```
React.createCla
React.isValidEl
```

```
ReactDOM.render
ReactDOM.unmour
```

```
ReactDOMServer.
ReactDOMServer.
```

There are more, bu

See: React top-lev

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 markdc
<div dangerousl
```

See: Dangerously

Lists

```
class TodoList
  render () {
ite
ul>
}
```

Conditionals

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

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!
```

Returning strings

```
render() {

}
```

You can return just a string.

See: Fragments and strings

```
  ]
}
```

Fragments

```
render () {
  // Fragments don't require keys!




}
```

See: Fragments and strings

Portals

```
render () {




}
```

This renders `this.props.children` into any locat

See: Portals

≠ Property validation

PropTypes

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

See: Typechecking with PropTypes

any

Basic

string

Basic types

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

Enumerables (oneOf)

`number`	
`func`	
`bool`	
Enum	
`oneOf`(any)	Enum types
`oneOfType`(type array)	
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

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

Custom validation

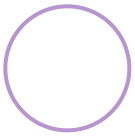
```
MyCo.propTypes = {
  customProp: (props, key, componentName)
    if (!/matchme/.test(props[key])) {
      return new Error('Validation failed
    }
  }
}
```

≠ Also see

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

▶ **31 Comments** for this cheatsheet. Write yours!

Search 352+ cheatsheets



Over 352 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
- Vue.js cheatsheet