



Redux cheatsheet

Creating a store

```
import { createStore } from 'redux'
function counter (state = { value: 0 }, action) {
  switch (action.type) {
 case 'INCREMENT'
   return { value: state.value + 1 }
 case 'DECREMENT':
   return { value: state.value - 1 }
 default:
   return state
let store = createStore(counter)
// Optional - you can pass `initialState` as a second arg
let store = createStore(counter, { value: 0 })
```

A store is made from a reducer function, which takes the current state, and returns a new state depending on the action it was given.

Using a store

```
let store = createStore(counter)
// Dispatches an action; this changes the state
store.dispatch({ type: 'INCREMENT'
store.dispatch({ type: 'DECREMENT' })
// Gets the current state
store.getState()
// Listens for changes
store.subscribe(() => { ... })
Dispatch actions to change the store's state.
```

React Redux

Provider

```
import { Provider } from 'react-redux'
React.render(
 <Provider store={store}>
  </Provider>, mountNode)
```

The <Provider> component makes the store available in your React components. You need this so you can use connect().

Shorthand

```
export default connect(
  (state) => ({
   message: state.message
  (dispatch) => ({
   onMessageClick: (message) => {
     dispatch({ type: 'click', message })
)(App)
```

Same as above, but shorter.

Combining reducers

```
const reducer = combineReducers({
 counter, user, store
Combines multiple reducers into one reducer function. See: combineReducers
```

Mapping state

```
import { connect } from 'react-redux'
// A functional React component
function App ({ message, onMessageClick }) {
   <div onClick={() => onMessageClick('hello')}>
   </div>
// Maps `state` to `props`:
// These will be added as props to the component.
function mapState (state) {
 return { message: state.message }
// Maps `dispatch` to `props`:
function mapDispatch (dispatch) {
 return {
   onMessageClick (message) {
     dispatch({ type: 'click', message })
// Connect them:
export default connect(mapState, mapDispatch)(App)
```

Middleware

Signature

Applying middleware

```
const logger = store => dispatch => action { dispatch(action) }

const logger = store => {
    // This function runs on createStore().
    // It returns a decorator for dispatch().

return dispatch => {
    // Runs on createStore(), too.
    // It returns a new dispatch() function

return action => {
    // Runs on every dispatch()
    }
}

Middlewares are simply decorators for dispatch() to allow you to take different kinds of actions, and to perform different tasks when receiving actions.
```

const store = createStore(reducer, {}, enhancer)

References

 Redux
 React-redux
 Usage with React

 (npmjs.com)
 (redux.js.org)

4

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

devhints.io / Search 357+ cheatsheets

Q

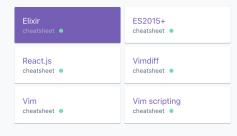






React.js cheatsheet •	Enzyme cheatsheet •
Enzyme v2 cheatsheet •	Awesome Redux cheatsheet •
Flux architecture cheatsheet	React-router cheatsheet

Top cheatsheets



cb