VALENTINOGAGLIARDI

LAST UPDATED 17TH MAY 2018 BY VALENTINO GAGLIARDI

React Redux Tutorial for Beginners: The Definitive Guide (2018)

*The simplest **React Redux tutorial** I wish I had when I started learning (updated to webpack4!)*

When I first started learning **Redux** I wish I could find the simplest tutorial ever.

Despite the great resources out there I couldn't wrap my head around some of the Redux concepts.

I knew what's the **state**. But **actions**, **action creators, and reducers?** They were obscure for me.

Last but not least I didn't know how to glue **React and Redux together**.

During those days I started writing my own React Redux tutorial and since then I learned a lot.

I taught myself the Redux fundamentals by writing this guide. I hope it'll be useful for all those learning React and Redux.

Enjoy the reading!

Table of Contents

React Redux tutorial: who this guide is for

The following React Redux guide is exactly what you're looking for if:

- you have a good grasp of Javascript, ES6, and React
- you're looking forward to learn Redux in the most simple way

React Redux tutorial: what you will learn

In the following guide you will learn:

1. what is Redux
2. how to use Redux with React

React Redux tutorial: a minimal React development environment

To follow along with the guide you should have a good grasp of **Javascript**, **ES6**, and **React**.

Also, create a **minimal React development environment** before starting off.

Feel free to use **webpack** or **Parcel.**

To get started with **webpack** clone my Github repo:

```
1. git clone https://github.com/valentinogagliardi/webpack-4-quickstart.git
```

Remove the file `./src/App.js` from the repo before moving forward.

React Redux tutorial: what is the state?

To **understand what is Redux** you must first understand what is the **state**.

If you have ever worked with React the term state should be no surprise to you.

I guess you already wrote **stateful React components** like the following:

```
import React, { Component } from "react";

class ExampleComponent extends Component {
  constructor() {
    super();

    this.state = {
      articles: [
        { title: "React Redux Tutorial for Beginners", id: 1 },
        { title: "Redux e React: cos'è Redux e come usarlo con React", id: 2 }
      ]
    };
  }

  render() {
    const { articles } = this.state;
    return <ul>{articles.map(el => <li key={el.id}>{el.title}</li>)}</ul>;
  }
}
```

A **stateful React component** is basically a Javascript ES6 class.

Every stateful React component **carries its own state**. In a React component the state holds up **data**.

The component might render such data to the user.

And there's setState for updating the local state of a component.

Everybody learned React this way:

- render some data from the **local state**
- update the state with React setState

React Redux tutorial: what problem does Redux solve?

Now, it is fine to keep the state within a React component as long as the application remains **small**.

But things could become tricky in more complex scenarios.

You will end up with a bloated component filled with methods for managing and updating the state. The **frontend shouldn't know about the business logic**.

Fine.

So what are the alternatives for managing the state of a React component? **Redux** is one of them.

Redux solves a problem that might not be clear in the beginning: it helps giving **each React component** the **exact piece of state** it needs.

Redux holds up the **state** within a **single location**.

Also with Redux the **logic for fetching and managing the state** lives **outside React**.

The benefits of this approach might be not so evident. Things will look clear as soon as you'll get your feet wet with Redux.

In the next section we'll see why you should learn Redux and when to use Redux within your applications.

React Redux tutorial: should I learn Redux?

Are you trying to learn Redux but you're going nowhere?

Redux literally scares most beginners. But that shouldn't be your case.

Redux is not that hard. The key is: don't rush learning Redux just because everybody is using it.

You should **start learning Redux** if you're **motivated and passionate** about it.

Take your time.

I started to learn Redux because:

- I was 100% interested in learning how Redux works
- I was eager to improve my React skills
- the combination React/Redux is ubiquitous

- Redux is **framework agnostic**. Learn it once, use it everywhere (Vue JS, Angular)

React Redux tutorial: should I use Redux?

It is feasible to build complex React application without even touching Redux. That comes at a cost.

Redux has a cost as well: it adds another layer of abstraction. But I prefer thinking about **Redux as an investment**, not as a cost.

Another common question for Redux beginners is: how do you know **when you're ready to use Redux**?

If you think about it there is no rule of thumb for determining when you do need **Redux for managing the state**.

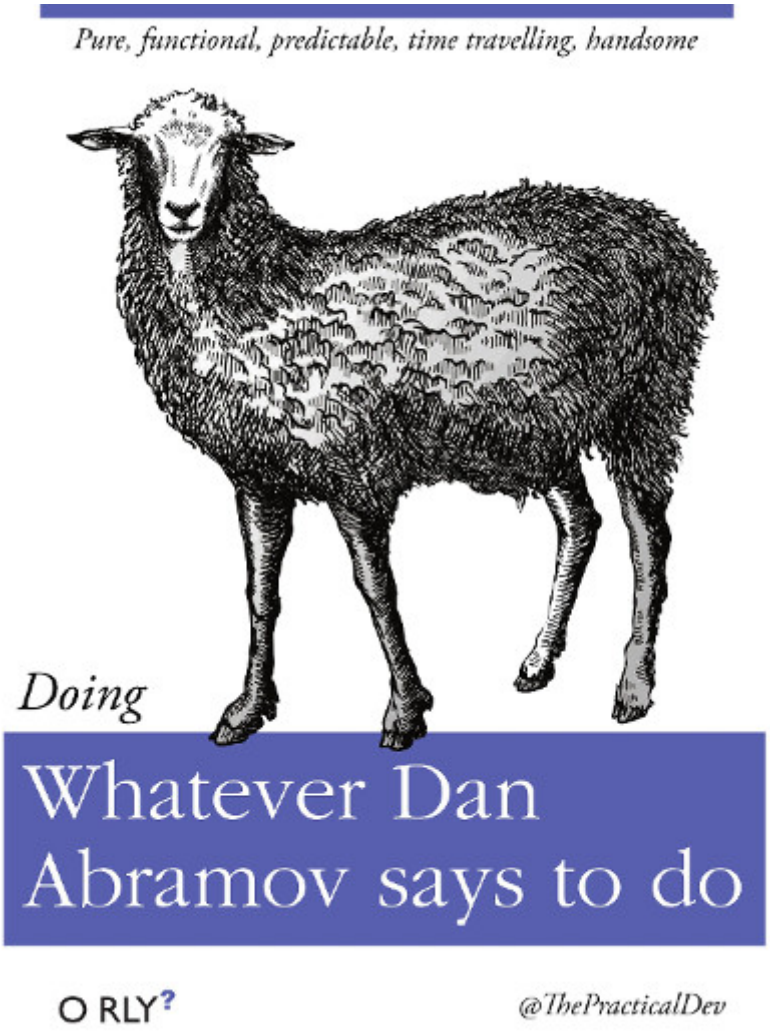
What I found is that you should **consider using Redux** when:

- multiple React components needs to access the same state but do not have any parent/child relationship
- you start to feel awkward passing down the state to multiple components with props

If that makes still no sense for you do not worry, I felt the same.

Dan Abramov says "Flux libraries are like glasses: you'll know when you need them."

And in fact it worked like that for me.



Courtesy of dev.to

Before going further take your time to understand what problem does Redux solve and whether you're motivated or not to learn it.

Be aware that Redux is not useful for smaller apps. It really shines in bigger ones. Anyway, learning Redux even if you're not involved in something big wouldn't harm anyone.

In the next section we'll start building a proof of concept to introduce:

- the **Redux fundamental principles**
- Redux alongside with React

React Redux tutorial: getting to know the Redux store

Actions. Reducers. I kind of knew about them. But one thing wasn't clear to me: **how were all the moving parts glued together**?

Were there some **minions** or what?

In Redux there are no minions (unfortunately).

The **store orchestrates all the moving parts in Redux**. Repeat with me: **the store**.

The store in Redux is like the human brain: it's kind of magic.

The **Redux store is fundamental**: the **state of the whole application** lives **inside the store**.

So to start playing with Redux we should **create a store for wrapping up the state**.

Move into your React development environment and install Redux:

```
cd webpack-4-quickstart/
```

```
npm i redux --save-dev
```

Create a directory for the store:

```
mkdir -p src/js/store
```

Create a new file named `index.js` in `src/js/store` and finally initialize the store:

```
// src/js/store/index.js

import { createStore } from "redux";
import rootReducer from "../reducers/index";

const store = createStore(rootReducer);

export default store;
```

createStore is the function for creating the Redux store.

createStore takes a reducer as the first argument, rootReducer in our case.

You may also pass an initial state to createStore. But most of the times you don't have to. Passing an initial state is useful for server side rendering. Anyway, **the state comes from reducers**.

***NOTE**: see Reducer returned undefined during initialization*

What matters now is understanding what does a reducer do.

In Redux **reducers produce the state**. The state is not something you create by hand.

Armed with that knowledge let's move on to our first Redux reducer.

React Redux tutorial: getting to know Redux reducers

While an initial state is useful for SSR, in Redux **the state must return entirely from reducers**.

Cool but what's a reducer?

A reducer is just a Javascript function. A reducer **takes two parameters**: **the current state** and an **action** (more about actions soon).

The third principle of Redux says that the state is immutable and cannot change in place.

This is why the reducer must be pure. A pure function is one that returns the exact same output for the given input.

In plain React the local state changes in place with setState. In Redux you cannot do that.

Creating a reducer is not that hard. It's a plain Javascript function with two parameters.

In our example we'll be creating a **simple reducer taking the initial state** as the first parameter. As a **second parameter** we'll provide **action**. As of now the reducer will do nothing than returning the initial state.

Create a directory for the root reducer:

```
1. mkdir -p src/js/reducers
```

Then create a new file named `index.js` in the `src/js/reducers`:

```
// src/js/reducers/index.js

const initialState = {
  articles: []
};

const rootReducer = (state = initialState, action) => state;

export default rootReducer;
```

I promised to keep this guide as simple as possibile. That's why our first reducer is a silly one: it returns the initial state without doing anything else.

Notice how the initial state is passed as a default parameter.

In the next section we'll add an action to the mix. That's where things will become interesting.

React Redux tutorial: getting to know Redux actions

Redux reducers are without doubt the most important concept in Redux. **Reducers produce the state of the application**.

But **how does a reducer know when to produce the next state**?

The second principle of Redux says the **only way to change the state is by sending a signal to the store**.This signal is an **action**. "**Dispatching an action**" is the process of sending out a signal.

Now, how do you change an immutable state? You won't. The resulting state is a copy of the current state plus the new data.

That's a lot to know.

The reassuring thing is that **Redux actions are nothing more than Javascript objects**. This is what an action looks like:

```
{
```

```
  type: 'ADD_ARTICLE',
  payload: { name: 'React Redux Tutorial', id: 1 }
}
```

Every action needs a type property for describing how the state should change.

You can specify a payload as well. In the above example the payload is a new article. A reducer will add the article to the current state later.

It is a best pratice to **wrap every action within a function**. Such function is an **action creator**.

Let's put everything together by creating a simple Redux action.

Create a directory for the actions:

```
mkdir -p src/js/actions
```

Then create a new file named `index.js` in `src/js/actions`:

```
// src/js/actions/index.js

export const addArticle = article => ({ type: "ADD_ARTICLE", payload: article
});
```

So, the **type property** is nothing more than a string.

The reducer will use that string to determine how to calculate the next state.

Since strings are prone to typos and duplicates it's **better to have action types declared as constants**.

This approach helps **avoiding errors that will be difficult to debug**.

Create a new directory:

```
mkdir -p src/js/constants
```

Then create a new file named `action-types.js` into the `src/js/constants`:

```
// src/js/constants/action-types.js

export const ADD_ARTICLE = "ADD_ARTICLE";
```

Now open up again `src/js/actions/index.js` and update the action to use action types:

```
// src/js/actions/index.js

import { ADD_ARTICLE } from "../constants/action-types";

export const addArticle = article => ({ type: ADD_ARTICLE, payload: article });
```

We're one step closer to have a working Redux application. Let's refactor our reducer!

React Redux tutorial: refactoring the reducer

Before moving forward let's recap the main Redux concepts:

- the **Redux store** is like a brain: it's in charge for **orchestrating all the moving parts** in Redux
- the **state of the application lives as a single, immutable object** within the store
- as soon as **the store receives an action it triggers a reducer**
- the **reducer returns the next state**

What's a **Redux reducer** made of?

A reducer is a Javascript function taking **two parameters**: the **state** and the **action**.

A reducer function has a **switch statement** (although unwieldy, a naive reducer could also use if/else).

The **reducer calculates the next state depending on the action type**. Moreover, **it should return at least the initial state when no action type matches**.

When the action type matches a case clause the **reducer calculates the next state** and **returns a new object**. Here's an excerpt of the code:

```
// ...
  switch (action.type) {
    case ADD_ARTICLE:
      return { ...state, articles: [...state.articles, action.payload] };
    default:
      return state;
  }
// ...
```

The reducer we created in the previous section does nothing than returning the initial state. Let's fix that.

Open up `src/js/reducers/index.js` and update the reducer as follow:

```
import { ADD_ARTICLE } from "../constants/action-types";

const initialState = {
  articles: []
};

const rootReducer = (state = initialState, action) => {
  switch (action.type) {
    case ADD_ARTICLE:
      state.articles.push(action.payload);
      return state;
    default:
      return state;
  }
};

export default rootReducer;
```

What do you see here?

Although it's valid code the **above reducer breaks** the main Redux principle: **immutability**.

Array.prototype.push is an impure function: it alters the original array.

Making our reducer compliant is easy. Using Array.prototype.concat in place of Array.prototype.push is enough to keep the initial array immutable:

```
import { ADD_ARTICLE } from "../constants/action-types";

const initialState = {
  articles: []
};

const rootReducer = (state = initialState, action) => {
  switch (action.type) {
    case ADD_ARTICLE:
      return { ...state, articles: state.articles.concat(action.payload) };
    default:
      return state;
  }
};

export default rootReducer;
```

We're not done yet! With the spread operator we can make our reducer even better:

```
import { ADD_ARTICLE } from "../constants/action-types";

const initialState = {
  articles: []
};

const rootReducer = (state = initialState, action) => {
  switch (action.type) {
    case ADD_ARTICLE:
      return { ...state, articles: [...state.articles, action.payload] };
    default:
      return state;
  }
};

export default rootReducer;
```

In the example above the initial state is left utterly untouched.

The initial articles array doesn't change in place.

The initial state object doesn't change as well. The resulting state is a copy of the initial state.

There are two key points for **avoiding mutations in Redux**:

- Using concat(), slice(), and ...spread for arrays
- Using Object.assign() and ...spread for objects

The **object spread operator** is still in stage 3. Install Object rest spread transform to **avoid a SyntaxError Unexpected token** when using the object spread operator in Babel:

```
npm i --save-dev babel-plugin-transform-object-rest-spread
```

Open up `.babelrc` and update the configuration:

```
{
    "presets": ["env", "react"],
    "plugins": ["transform-object-rest-spread"]
}
```

Redux protip: the reducer will grow as your app will become bigger. You can split a big reducer into separate functions and combine them with combineReducers

In the next section we'll play with Redux from the console. Hold tight!

React Redux tutorial: Redux store methods

This will be super quick, I promise.

I want you to play with the brower's console for gaining a quick understanding of how Redux works.

Redux itself is a small library (2KB). The Redux store exposes a simple API for managing the state. The most important methods are:

- getState for accessing the current state of the application
- dispatch for dispatching an action
- subscribe for listening on state changes

We will play in the brower's console with the above methods.

To do so we have to export as global variables the store and the action we created earlier.

Create `src/js/index.js` and update the file with the following code:

```
import store from "../js/store/index";
import { addArticle } from "../js/actions/index";

window.store = store;
window.addArticle = addArticle;
```

Open up `src/index.js` as well, clean up its content and update it as follows:

```
import index from "./js/index"
```

Now run webpack dev server (or Parcel) with:

```
npm start
```

head over http://localhost:8080/ and open up the console with F12.

Since we've exported the store as a global variable we can access its methods. Give it a try!

Start off by **accessing the current state**:

```
store.getState()
```

output:

```
{articles: Array(0)}
```

Zero articles. In fact we haven't update the initial state yet.

To make things interesting we can listen for state updates with subscribe.

The **subscribe method accepts a callback that will fire whenever an action is dispatched**. Dispatching an action means notifying the store that we want to change the state.

Register the callback with:

```
store.subscribe(() => console.log('Look ma, Redux!!'))
```

To **change the state in Redux we need to dispatch an action**. To dispatch an action you have to call the dispatch method.

We have one action at our disposal: addArticle for adding a new item to the state.

Let's dispatch the action with:

```
store.dispatch( addArticle({ name: 'React Redux Tutorial for Beginners', id: 1
}) )
```

Right after running the above code you should see:

```
Look ma, Redux!!
```

To verify that the state changed run again:

```
store.getState()
```

The output should be:

```
1. {articles: Array(1)}
```

And that's it. This is Redux in its simplest form.

Was that difficult?

Take your time to explore these three Redux methods as an exercise. Play with them from the console:

- getState for **accessing the current state** of the application
- dispatch for **dispatching an action**
- subscribe for **listening on state changes**

That's everything you need to know for getting started with Redux.

Once you feel confident head over the next section. We'll go straight to connecting React with Redux!

React Redux tutorial: connecting React with Redux

After learning Redux I realized it wasn't so complex.

I knew how to access the current state with getState.

I knew how to dispatch an action with dispatch

I knew how to listen for state changes with subscribe

Yet I didn't know how to couple **React and Redux** together.

I was asking myself: should I call getState within a React component? How do I dispatch an action from a React component? And so on.

Redux on its own is framework agnostic. You can use it with vanilla Javascript. Or with Angular. Or with React. There are bindings for joining together Redux with your favorite framework/library.

For React there is react-redux.

Before moving forward install react-redux by running:

```
npm i react-redux --save-dev
```

To demonstrate how React and Redux work together we'll build a super simple application. The application is made of the following components:

- an App component
- a List component for displaying articles
- a Form component for adding new articles

(The application is a toy and it does nothing serious other than displaying a list and a form for adding new items. Nonetheless it's still a good starting point for learning Redux)

React Redux tutorial: react-redux

react-redux is a Redux binding for React. It's a small library for connecting Redux and React in an efficient way.

The most important method you'll work with is connect

What does react-redux's **connect** do? Unsurprisingly it connects a React component with the Redux store.

You will use connect with two or three arguments depending on the use case. The fundamental things to know are:

- the mapStateToProps function
- the mapDispatchToProps function

What does mapStateToProps do in react-redux? mapStateToProps does exactly what its name suggests: it **connects a part of the Redux state** to the props of a React component. By doing so a connected React component will have access to the exact part of the store it needs.

What does mapDispatchToProps do in react-redux? mapDispatchToProps does something similar, but for actions. **mapDispatchToProps connects Redux actions to React props**. This way a connected React component will be able to dispatch actions.

Is everything clear? If not, stop and take your time to re-read the guide. I know it's a lot to learn and it requires time. Don't worry if you don't get Redux right know. It will click sooner or later.

In the next section we'll finally get our hands dirty!

React Redux tutorial: App component and Redux store

We saw that mapStateToProps connects a portion of the Redux state to the props of a React component. You may wonder: is this enough for connecting Redux with React? No, it's not.

To start off **connecting Redux with React we're going to use Provider**.

Provider is an high order component coming from react-redux.

Using layman's terms, Provider wraps up your React application and makes it aware of the entire Redux's store.

Why so? We saw that in Redux the store manages everything. React must talk to the store for accessing the state and dispatching actions.

Enough theory.

Open up `src/js/index.js`, wipe out everything and update the file with the following code:

```
1.  import React from "react";
2.  import { render } from "react-dom";
3.  import { Provider } from "react-redux";
4.  import store from "./store/index";
5.  import App from "./components/App";
6.
7.  render(
8.    <Provider store={store}>
```

```
    <App />
  </Provider>,
  document.getElementById("app")
);
```

You see? Provider wraps up your entire React application. Moreover it gets the store as a prop.

Now let's create the **App** component since we're requiring it. It's nothing special: App should import a List component and render itself.

Create a directory for holding the components:

```
mkdir -p src/js/components
```

and a new file named `App.js` inside `src/js/components`:

```
// src/js/components/App.js
import React from "react";
import List from "./List";

const App = () => (
  <div className="row mt-5">
    <div className="col-md-4 offset-md-1">
    <h2>Articles</h2>
      <List />
    </div>
  </div>
);

export default App;
```

Take moment and look at the component without the markup:

```
import React from "react";
import List from "./List";

const App = () => (
      <List />
);

export default App;
```

then move on to creating **List**.

React Redux tutorial: List component and Redux state

We have done nothing special so far.

But our new component, List, will interact with the Redux store.

A brief recap: the key for connecting a React component with Redux is connect.

Connect takes at least one argument.

Since we want List to get a list of articles it's a matter of connecting `state.articles` with the component. How? With **mapStateToProps**.

Create a new file named `List.js` inside `src/js/components`. It should look like the following:

```
1.  // src/js/components/List.js
2.
3.  import React from "react";
4.  import { connect } from "react-redux";
5.
6.  const mapStateToProps = state => {
7.    return { articles: state.articles };
8.  };
9.
10. const ConnectedList = ({ articles }) => (
11.   <ul className="list-group list-group-flush">
12.     {articles.map(el => (
13.       <li className="list-group-item" key={el.id}>
14.         {el.title}
15.       </li>
16.     ))}
17.   </ul>
18. );
19.
20. const List = connect(mapStateToProps)(ConnectedList);
21.
22. export default List;
```

The List component receives the prop `articles` which is a copy of the `articles` array. Such array lives inside the Redux state we created earlier. It comes from the reducer:

```
1.  const initialState = {
2.    articles: []
```

```
};

const rootReducer = (state = initialState, action) => {
  switch (action.type) {
    case ADD_ARTICLE:
      return { ...state, articles: [...state.articles, action.payload] };
    default:
      return state;
  }
};
```

Then it's a matter of using the prop inside JSX for generating a list of articles:

```
{articles.map(el => (
  <li className="list-group-item" key={el.id}>
    {el.title}
  </li>
))}
```

React protip: take the habit of validating props with PropTypes

Finally the component gets exported as List. List is the result of connecting the stateless component ConnectedList with the Redux store.

A stateless component does not have its own local state. Data gets passed to it as props

Still confused? I was too. Understanding how **connect** works will take some time. Fear not, the road to learn Redux is paved with "ah-ha" moments.

I suggest taking a break for exploring both **connect** and **mapStateToProps.**

Once you're confident about them head over the next section!

React Redux tutorial: Form component and Redux actions

The Form component we're going to create is a bit more complex than List. It's a form for adding new items to our application.

Plus it is a **stateful component**.

A stateful component in React is a component carrying its own local state

A stateful component? “Valentino, we’re talking about Redux for managing the state! Why on earth would you give Form its own local state??”

Even when using Redux it is totally fine to have stateful components.

Not every piece of the application’s state should go inside Redux.

In this example I don’t want any other component to be aware of the Form local state.

And that’s perfectly fine.

What does the component do?

The component contains some logic for updating the local state upon a form submission.

Plus it receives a Redux action as prop. This way it can update the global state by dispatching the addArticle action.

Create a new file named `Form.js` inside `src/js/components`. It should look like the following:

```
1.  // src/js/components/Form.js
2.  import React, { Component } from "react";
3.  import { connect } from "react-redux";
4.  import uuidv1 from "uuid";
5.  import { addArticle } from "../actions/index";
6.
7.  const mapDispatchToProps = dispatch => {
8.    return {
9.      addArticle: article => dispatch(addArticle(article))
10.   };
11. };
12.
13. class ConnectedForm extends Component {
14.   constructor() {
15.     super();
16.
17.     this.state = {
18.       title: ""
19.     };
20.
21.     this.handleChange = this.handleChange.bind(this);
22.     this.handleSubmit = this.handleSubmit.bind(this);
23.   }
24.
```

```
  handleChange(event) {
    this.setState({ [event.target.id]: event.target.value });
  }

  handleSubmit(event) {
    event.preventDefault();
    const { title } = this.state;
    const id = uuidv1();
    this.props.addArticle({ title, id });
    this.setState({ title: "" });
  }

  render() {
    const { title } = this.state;
    return (
      <form onSubmit={this.handleSubmit}>
        <div className="form-group">
          <label htmlFor="title">Title</label>
          <input
            type="text"
            className="form-control"
            id="title"
            value={title}
            onChange={this.handleChange}
          />
        </div>
        <button type="submit" className="btn btn-success btn-lg">
          SAVE
        </button>
      </form>
    );
  }
}

const Form = connect(null, mapDispatchToProps)(ConnectedForm);

export default Form;
```

What can I say about the component? Besides **mapDispatchToProps** and **connect** it's standard React stuff.

mapDispatchToProps connects Redux actions to React props. This way a connected component is able to dispatch actions.

You can see how the action gets dispatched in the handleSubmit method:

```
// ...
  handleSubmit(event) {
    event.preventDefault();
    const { title } = this.state;
    const id = uuidv1();
    this.props.addArticle({ title, id }); // Relevant Redux part!!
// ...
  }
```

```
// ...
```

Finally the component gets exported as Form. Form is the result of connecting ConnectedForm with the Redux store.

Side note: the first argument for connect must be `null` when mapStateToProps is absent like in the Form example. Otherwise you'll get `TypeError: dispatch is not a function`.

Our components are all set!

Update App to include the Form component:

```
import React from "react";
import List from "./List";
import Form from "./Form";

const App = () => (
  <div className="row mt-5">
    <div className="col-md-4 offset-md-1">
      <h2>Articles</h2>
      <List />
    </div>
    <div className="col-md-4 offset-md-1">
      <h2>Add a new article</h2>
      <Form />
    </div>
  </div>
);

export default App;
```

Install uuid with:

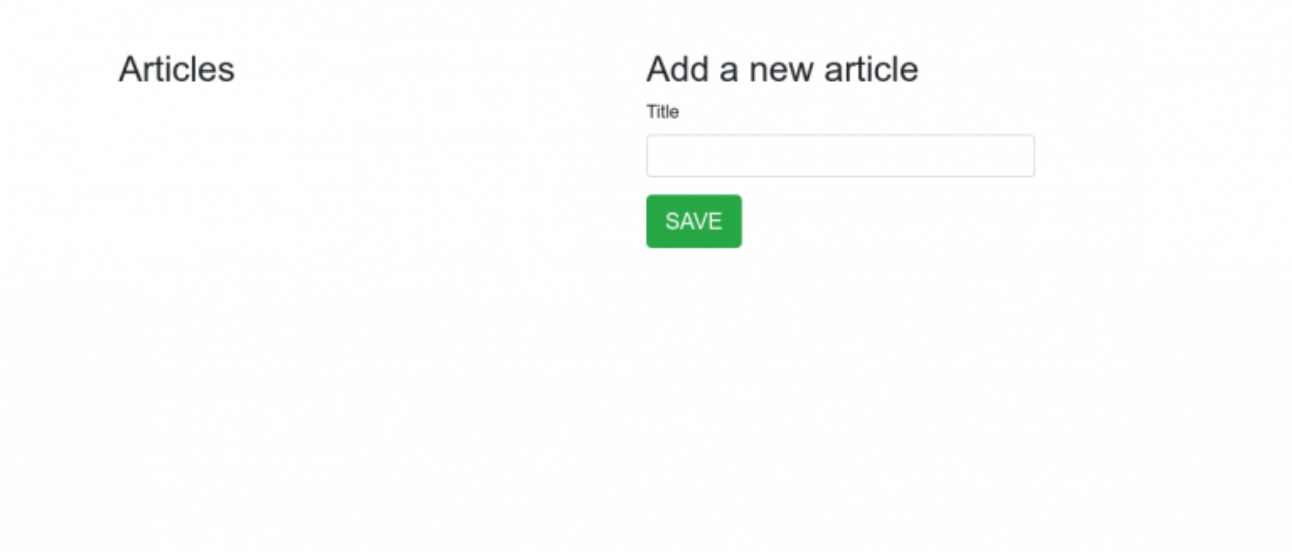
```
npm i uuid --save-dev
```

Now run webpack (or Parcel) with:

```
npm start
```

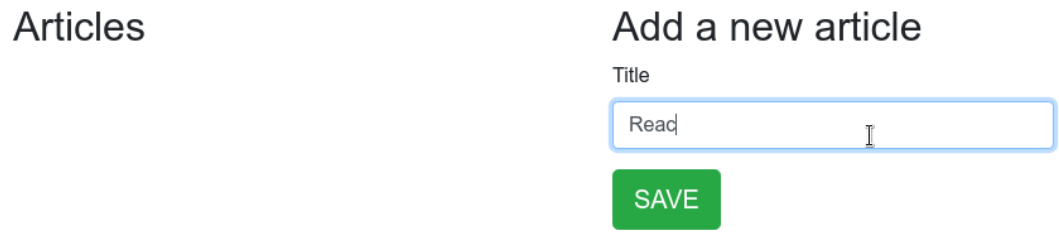
and head over to http://localhost:8080

You should see the following working POC:



Nothing fancy but still useful for showing React and Redux at work!

The **List component on the left is connected to the Redux store**. It will re-render whenever you add a new item.



Whoaaa!

You can find the code for that silly app (with prop-types and a splitted reducer) on my Github repo.

React Redux tutorial: wrapping up

I hope you'll learn something from this guide. I tried my best to keep things as simple as possibile. I would love to hear your feedback in the comments below!

Redux has a lot of boilerplate and moving parts. Don't get discouraged. Pick Redux, play with it and take your time to absorb all the concepts.

I went from zero to **understanding Redux by small steps**. You can do it too!

Also, take your time to investigate **why** and **if** you should **use Redux in your application**.

Either way think of Redux as an investment: learning it is **100% worthwile**.

React Redux tutorial: books for learning Redux

What's the best way for **levelling up your Redux skills** once you finish my tutorial?

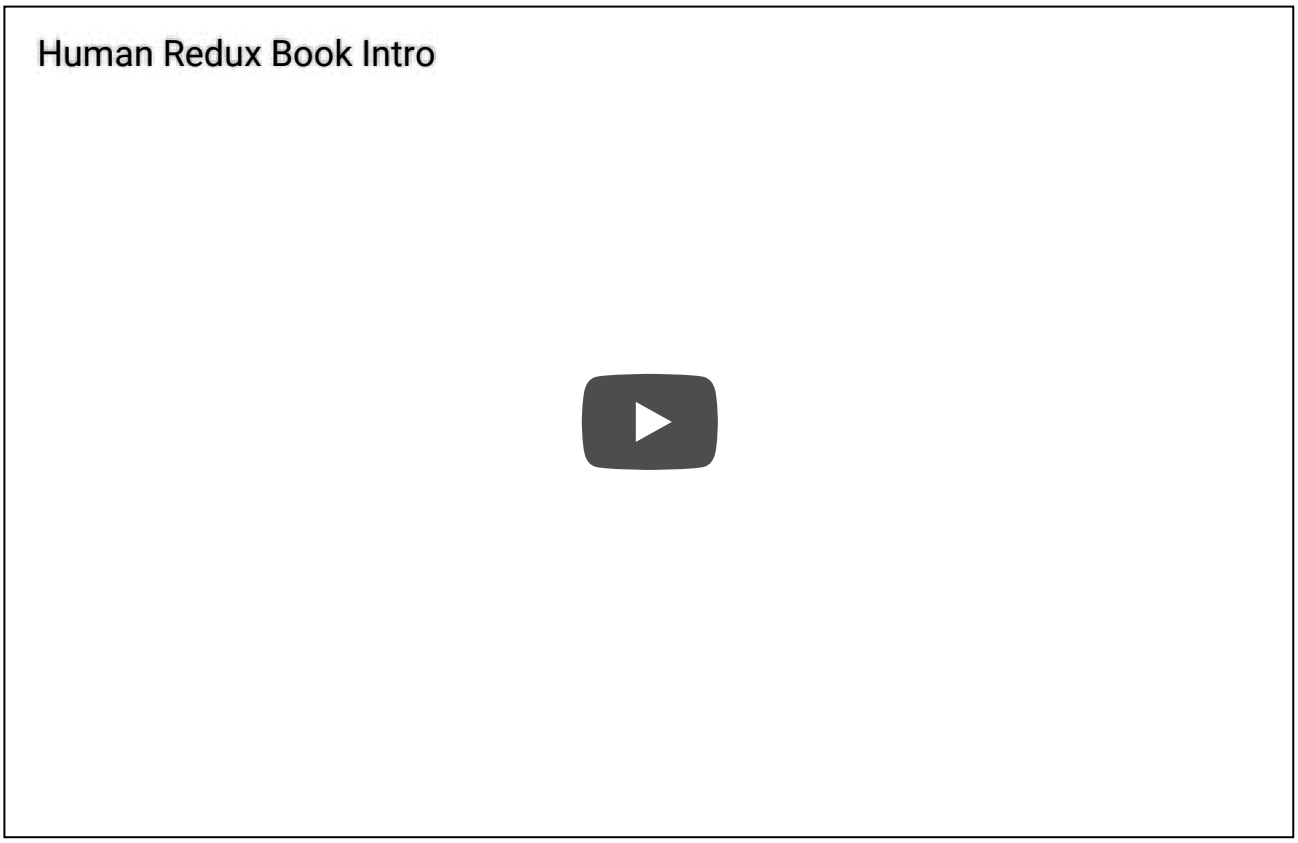
Reading a book of course. Even better, **reading a book written by an expert**.

I had the privilege to put my hands on Human Redux by Henrik Joreteg.

So far I love the book.

Henrik has a lot of experience in building real world complex web applications.

He know Progressive Web Apps and how to use Redux.



Buy Henrik's book if you want to:

- learn real world Redux patterns
- level up your Redux skills
- learn about advanced Redux topics (**middlewares**, **async actions, selectors, reselect**)

I highly recommend Human Redux even if you're just starting out with Redux.

It is a nice companion for my tutorial. Plus you'll learn from a **real expert**.

Go grab it!

Other useful resources for learning Redux:

The official documentation: Redux Docs.

Resources for learning Redux by Mark Erikson.

Dan Abramov has a nice write-up that will help you understand if your application needs Redux

There's also an interesting thread on dev.to with Dan explaining when is it time to stop managing state at the component level and switching to Redux

React Redux tutorial: resources for learning functional programming

Redux scares most beginners because it revolves around functional programming and pure functions.

Suggesting some resources is the best I can do since functional programming is beyond the scope of this guide. These are the best places for learning more about functional programming and pure functions:

Professor Frisby's Mostly Adequate Guide to Functional Programming by Brian Lonsdorf

Functional-Light Javascript by Kyle Simpson

React Redux tutorial: async actions in Redux

I wasn't sure whether talking about **async actions** would have been appropriate or not.

Most Redux beginners struggle to learn plain Redux. Handling async actions in Redux is not a straighforward task so better not overcomplicate things.

When you'll feel confident about the core Redux concepts go check A Dummy's Guide to Redux and Thunk in React by Matt Stow. It's a nice introduction to handling API calls in Redux with **redux-thunk**.

Thanks for reading and happy coding!

Valentino Gagliardi

Consultant, Developer Coach. Are you stuck on a project? Let's talk!

JAVASCRIPT

145 Replies to “React Redux Tutorial for Beginners: The Definitive Guide (2018)”

Kalai

6TH JANUARY 2018 AT 3:41 AM

Thanks for posting this tutorial. As a beginner, I found it very easy to understand the redux concepts.. kudos!

Olivier

6TH JANUARY 2018 AT 12:42 PM

Exactly what i was waiting for! Thanks a lot Valentino.

Eric

7TH JANUARY 2018 AT 10:26 PM

Wonderful Tutorial. The best I have come across so far. Your encouragement words are also helpful.. LOL

hugo

10TH JANUARY 2018 AT 3:47 PM

You lost me on this last piece of code too bad

hugo

10TH JANUARY 2018 AT 3:48 PM

The Form *

Valentino Gagliardi

10TH JANUARY 2018 AT 4:27 PM

Yes, that's really bad. How can I help you?

ajay

13TH JANUARY 2018 AT 6:28 PM

please write post for redux-saga

Valentino Gagliardi

13TH JANUARY 2018 AT 7:17 PM

Let's see what can I do! Stay tuned!

Samson Ssali

3RD APRIL 2018 AT 4:56 AM

Can't wait for this too

Julia

16TH JANUARY 2018 AT 9:18 AM

Just finished the tutorial, it was extremely helpful in breaking everything into discrete steps. + so many upvotes! As a beginner, it was great to see everything explained clearly (e.g. why push bad, spread operator good). I didn't even know that objects could have spread operators, but makes a lot of sense.

Some comments as I was running the code:

1. I was having some issues with the paths specified. I used "./js/store/index" instead of "../js/store/index".

2. Was having an issue with exporting App to src/index.js so I deleted the `export default App` statement and did `export const App` at the top and imported { App } like so.

3. Interestingly, using arrow functions to bind methods didn't work. It may have something to do with babel presents specified. After googling a bit, it didn't seem like best practice anyways for performance (oops, been writing React wrong!).

Other comments:

1. Would be great to see an article for redux-thunk and redux-saga, comparing the two.

2. Would be great to learn about how to test Redux with React.

3. Would be great to see a fullstack example with Node.

Thank you so much for sharing your knowledge!

Valentino Gagliardi

16TH JANUARY 2018 AT 10:08 AM

Julia: thanks for your comment! It's really rewarding to see other people learning from my tutorials.

This Redux tutorial took me 1 week. It was written specifically for other fellow developers like you.

I appreciate your suggestions! Stay tuned on this blog, maybe I'll put something together about redux-thunk and redux-saga.

Happy coding!

Mark Addinall

21ST FEBRUARY 2018 AT 5:53 AM

Hi. Great little tutorial! Well done.
My little App now follows your patterns. I have a few components/containers

that use Redux/connect.

The router doesn't seem to want to play with connect(). Any ideas?

Valentino Gagliardi

21ST FEBRUARY 2018 AT 10:44 AM

Thank you Mark. Do you have a working demo? I can take a look at the issue.

Yariv Gilad

20TH MAY 2018 AT 8:35 PM

try withrouter() or react-router-redux –
https://reacttraining.com/react-router/core/guides/redux-integration

S Eggleston

16TH JANUARY 2018 AT 10:04 AM

Thanks for the tutorial, it also worked well for an experienced developer new to redux

A few tiny tweaks:
* `Making our reducer complaint is easy.` <- Tiny typo, I think it should be "compliant"
* .bablerc was already configured
* I needed to run `npm install uuid`

Valentino Gagliardi

16TH JANUARY 2018 AT 10:12 AM

Thank you! I fixed the article.

Priyal

19TH JANUARY 2018 AT 1:19 PM

Thanks a lot

Valentino Gagliardi

19TH JANUARY 2018 AT 3:04 PM

Ehi Priyal thanks for reading!

Percy

19TH JANUARY 2018 AT 9:37 PM

I avoided redux in my first app because it looked too complicated.
When I did decide to learn it, I went from pole to post trying to find the right book, video, tutorial, article…you name it..anything that would help me grasp the concept (I am a seasoned developer) …but everyone trying to teach redux just seemed to get into very convoluted jargon, diagrams and explanations.
I sat down with this page, your page, open and in one afternoon I am all set. I know redux (imagine Neo saying “I know kung fu!”)
Thank you so much for a straight forward tutorial. Now I can go back and end expand my knowledge on the once-upon-a-time-complicated tutorials!

Pedro Marandi

20TH JANUARY 2018 AT 3:54 AM

Fantastic tutorial, Thank you Valentino.

I just have doubt in this sentence that you said: “A pure function is one that returns the exact same output for the given input.”

Is that the same term with side effects or it’s another principle of pure functions?

Pedro Marandi

20TH JANUARY 2018 AT 3:59 AM

Because as I noticed you are talking about immutability but the provided definition is not about side effects. That’s why I confused.

Anyway, thank you for the amazing tutorial.

Valentino Gagliardi

22ND JANUARY 2018 AT 9:51 AM

Pedro, thanks for stopping by!

A pure function is a function that:

– returns the same output for the given input

– has no side effects

Those are two distinct principles, yet they are both about functional programming.

Immutability is another functional programming concept and is strictly related to pure functions.

So, yeah, they're intertwined with each other as you can see!

Mohammed Jhaur

20TH JANUARY 2018 AT 5:18 AM

thanks, it is a very helpful for experience those are new in react

Valentino Gagliardi

22ND JANUARY 2018 AT 9:54 AM

Mohammed, I'm glad it helped 🙂

MANISH

29TH JANUARY 2018 AT 1:47 PM

GOOD ONE

Valentino Gagliardi

30TH JANUARY 2018 AT 2:59 PM

Thank you Manish 🙂

Searene

20TH JANUARY 2018 AT 1:14 PM

This is the best redux tutorial I have ever seen, thank you!

Valentino Gagliardi

22ND JANUARY 2018 AT 10:19 AM

Searene, thank you! I'm glad it helped!

Erika

20TH JANUARY 2018 AT 6:38 PM

Wow!! Thank you so much for this blogpost. You made everything so simple 🙂

Valentino Gagliardi

22ND JANUARY 2018 AT 10:19 AM

Erika, thank you for your kind words! Happy coding!

Ewa

21ST JANUARY 2018 AT 9:02 PM

Wow! Thank you so much for this tutorial. I'm a begginer and I'm not afraid (of Redux). Please continue your work.

Valentino Gagliardi

22ND JANUARY 2018 AT 10:33 AM

Thank you!

Shamaran

24TH JANUARY 2018 AT 4:49 PM

Really is a good article to understand the how redux get used with react

Steve Sharp

25TH JANUARY 2018 AT 5:21 AM

Typo in section 7 heading: “Rect Redux tutorial: should I use Redux?”
Rect -> React

Valentino Gagliardi

25TH JANUARY 2018 AT 7:48 AM

Good catch Steve! It’s fixed now

Kamil

25TH JANUARY 2018 AT 9:50 PM

Outstanding article!!! cleared my major confusion with React/Redux which i was struggling for few weeks which is mapDispatchToProps usage in the container component.Now i will play with your repo to get better understanding.

Valentino Gagliardi

26TH JANUARY 2018 AT 9:46 AM

Kamil, I’m happy to hear that!

kojo-zil

25TH JANUARY 2018 AT 10:13 PM

wow, i just love this awesome tutorial. Thanks a lot

Valentino Gagliardi

26TH JANUARY 2018 AT 9:46 AM

Thanks for stopping by!

kozycombs

30TH JANUARY 2018 AT 12:40 AM

Thank you for this tutorial and all your other tutorials on react. I am an angular developer looking to explore more of reactjs and I find your tutorials very easy to follow as a react beginner. I was unsure on how to begin learning redux and this has given me a starting point.
Please keep up the good work. cheers

Valentino Gagliardi

30TH JANUARY 2018 AT 3:03 PM

It's a pleasure helping other fellow developers. I'm glad you found my article useful! How do you like React? Happy coding!

Delboy

2ND FEBRUARY 2018 AT 3:01 PM

Great article !

Valentino Gagliardi

5TH FEBRUARY 2018 AT 11:36 AM

Thank you 🙂

Birender

6TH FEBRUARY 2018 AT 2:52 AM

Great! Can you please share some integration with API, or some real time scenarios.

Thanks for this great artical.

Valentino Gagliardi

6TH FEBRUARY 2018 AT 7:05 AM

Why not? Maybe in the future 🙂 Stay tuned

elijah

6TH FEBRUARY 2018 AT 6:18 PM

Hi, I think I am missing something, but I am getting a ‘module build failed’ error

Module build failed: SyntaxError: Unexpected token (8:15)

8 | return { …state, articles: […state.articles, action.payload] };

for some reason it doesn’t like returning this object with state.

Am i somehow missing babel or webpack, from the minimal template i downloaded from you?

so far this is a wonderful tutorial!

elijah

6TH FEBRUARY 2018 AT 6:42 PM

Hi disregard thelast comment, I missed the .babelrc

however I am trying to follow your console redux part, and I amgetting this error wheni try and type the following:

store.dispatch(addArticle({ name: “React Redux Tutorial for Beginners”, id: 1 }))

Error: Actions may not have an undefined “type” property. Have you misspelled a constant?

why is this happening?

harryzhux

8TH FEBRUARY 2018 AT 6:37 PM

A small mistake: In the example of initial reducer that breaks the immutability
return { …state, articles: state.articles.push(action.payload) };
actually returns { articles: 1 } (the number of articles, not the array of article)

So it probably should rephrase to (not really a good situation to use spread op here for the whole article, in my opinion)
state.articles.push(action.payload)
return state

Valentino Gagliardi

9TH FEBRUARY 2018 AT 8:51 AM

Good catch! I’ll fix it

masha

9TH FEBRUARY 2018 AT 5:03 AM

Great article! One of the simplest one on Redux I could find to get the concepts right.. Something I’m struggling to understand though in the reducer code, how exactly are we returning two values? (state,articles)? Is this some concept on functional programming?

return { …state, articles: [...state.articles, action.payload] };

Dustin M

9TH FEBRUARY 2018 AT 11:10 AM

This is wonderful! mapStateToProps and mapDispatchToProps finally clicked after seeing them in two different components. Best explanation of react/redux I have read.

Valentino Gagliardi

9TH FEBRUARY 2018 AT 2:28 PM

Thank you Dustin! I'm glad to help!

Fabian

10TH FEBRUARY 2018 AT 1:06 AM

Thanks a lot for putting this together and sharing it! It was very generous of you. Redux comes with a paradigm shift that can seem daunting at first but this tutorial really helps!

Valentino Gagliardi

15TH FEBRUARY 2018 AT 9:28 PM

Hi Fabian! I'm glad it helped!

ruel

14TH FEBRUARY 2018 AT 2:00 PM

nice one!

Christian H. Bohlbro

15TH FEBRUARY 2018 AT 8:48 PM

Wow, this was really helpful to get me started with react. Redux really felt like one big blur that could take hours to get into, but this made it really simple. Very great and thorough guide. Thanks a lot! 😀

Valentino Gagliardi

15TH FEBRUARY 2018 AT 9:31 PM

It's nice to read these words from you Christian! Happy coding!

dupleypunx

19TH FEBRUARY 2018 AT 7:18 PM

Thanks man for creating this awesome tutorial.
it's easy and worth to follow.
as a novice I feel enlightened with your guide!

thanks a lot !

Rodolfo

21ST FEBRUARY 2018 AT 2:39 PM

Really good tutorial.

Really helpful to me!

Thanks a lot for taking the time to share your learning.

Valentino Gagliardi

22ND FEBRUARY 2018 AT 4:56 PM

Thank you Rodolfo!

Gokul T

23RD FEBRUARY 2018 AT 10:13 AM

nice tutorial, really helpfull

Chase

25TH FEBRUARY 2018 AT 3:27 AM

Excellent tutorial, well done sir.

Sania

1ST MARCH 2018 AT 7:45 AM

Thankyou for helping us fellow developers. Stay blessed

Valentino Gagliardi

1ST MARCH 2018 AT 1:17 PM

You too 🙂

David

1ST MARCH 2018 AT 9:13 AM

Great tutorial for a Redux beginner. Thanks!

Valentino Gagliardi

1ST MARCH 2018 AT 4:40 PM

Thanks for stopping by David!

Steve

5TH MARCH 2018 AT 2:51 AM

Good tutorial, thanks for writing, I found it useful.

If I can suggest one small improvement – it would be helpful if you explained why mapDispatchToProps is useful (so the component doesn't have to know about redux / dispatch) and also what happens on the form submit. I got confused for a while thinking

this.props.addArticle was the action creator in actions/index.js when in fact it comes from mapDispatchToProps.

Valentino Gagliardi

5TH MARCH 2018 AT 2:41 PM

Thanks for your suggestions Steve!

Ashok

8TH MARCH 2018 AT 11:42 AM

this is forever the best learning material I found for react – redux from all over the wide web.
Thank you so much for such a masterpiece tutorial. would help me, if you have some reference for saga as well.

andy

10TH MARCH 2018 AT 2:24 AM

Nice article, but I can't clone your repo, it says permission deined.

Tapy

15TH MAY 2018 AT 5:13 AM

The code posted only works for those with permissions on the repo use this instead:

git clone https://github.com/valentinogagliardi/webpack-4-quickstart.git

James

10TH MARCH 2018 AT 4:55 PM

Hey, this guide really helped me. Thanks for all your hard work.

Wanted to let you know about one small typo. In the section: “React Redux tutorial: App component and Redux store”, there is a typo in the imports of the `./js/index.js`.

should be:

import store from “./store/index”;
import App from “./components/App”;

not:

import store from “../js/store/index”;
import App from “../js/components/App”;

Valentino Gagliardi

11TH MARCH 2018 AT 2:03 PM

Thank you. It should be fixed now

Mark

11TH MARCH 2018 AT 12:13 AM

Thank you for a splendid article that really helped to demystify redux for me.

Valentino Gagliardi

11TH MARCH 2018 AT 2:51 PM

I’m glad it helped!

Gal Sivan

13TH MARCH 2018 AT 2:34 PM

Hi Valentino and thanks so much for the tutorial.

This may be a really dumb mistake, but I keep getting and error when I try to dispatch addArticle. Can’t figure out where exactly I am not importing it:

store.dispatch(addArticle({ name: 'React Redux Tutorial for Beginners', id: 1 }))
VM13714:1 Uncaught TypeError: addArticle is not a function
at :1:17

Thanks again,
Gal

Gal Sivan

13TH MARCH 2018 AT 5:12 PM

this is very embarrassing i am sorry. i forgot to save src/js/actions/index.js… 🙂

please remove my comment and thanks again for the guide!!

Valentino Gagliardi

13TH MARCH 2018 AT 6:45 PM

Don't be embarassed 🙂 And thanks for reading!

Francesco Lodovici

14TH MARCH 2018 AT 10:38 PM

You truly deserve all the beers in this world.
Best React-Redux tutorial ever.
Period.

Nick

18TH MARCH 2018 AT 3:25 AM

awesome tutorial. Definitely helped me get my feet wet with redux.

Valentino Gagliardi

19TH MARCH 2018 AT 2:57 PM

Good to hear!

ujwal

18TH MARCH 2018 AT 8:41 PM

Very nice! I am an experienced developer but I started away from redux because of all the jargons. This was very simple. I can use redux now 😀

ujwal

18TH MARCH 2018 AT 8:42 PM

I meant to say, I stayed away from redux…

Kishore

19TH MARCH 2018 AT 7:16 PM

Hey man, Great post for redux novice like me. Looking forward for other articles related to react and redux.

Valentino Gagliardi

22ND MARCH 2018 AT 9:28 AM

yeah! Thanks for reading!

Johan

19TH MARCH 2018 AT 9:42 PM

Very clear and well written tutorial! Impressive! I had no problem following the concepts or the code even though I have never used redux or react before.

Valentino Gagliardi

22ND MARCH 2018 AT 9:28 AM

Good job! Happy coding!

Rutul

20TH MARCH 2018 AT 1:17 PM

You are the greatest person who has knowledge of Redux…. with excellent teaching and explaining skills

Valentino Gagliardi

22ND MARCH 2018 AT 9:30 AM

You're too kind! Thank you!

GalinaSophia

21ST MARCH 2018 AT 6:40 AM

Thanks for your simple tutorial about React/Redux for beginners.
I am a developer who is trying to learn react and redux and has problems while understanding tutorials from several websites.
After reading your tutorial, it gets better to understanding the relationships between reducer,action and state.
Also how to glue react and redux.
Thanks very much.

Valentino Gagliardi

22ND MARCH 2018 AT 9:38 AM

I'm glad it helped! Keep up the good work!

Lê Hữu Việt Anh

21ST MARCH 2018 AT 9:36 AM

Thanks for this tutorial. It is so amazing, I have been confused about Redux until reading this. Thank you very much and wish you the best.

Valentino Gagliardi

22ND MARCH 2018 AT 10:10 AM

Thank you!

Ruben

21ST MARCH 2018 AT 4:51 PM

You published an excellent guide for us. Even as React beginner I was capable to understand and reproduce all steps with the expected results.
Thank you very much from Colombia.

Valentino Gagliardi

22ND MARCH 2018 AT 10:12 AM

That's wonderful Ruben! Greetings from Italy 🙂

Silvia Rebelo

25TH MARCH 2018 AT 8:40 PM

Joining the sentiment of the rest of people here:

THANKS Valentino for taking the time to write and share this super comprehensive tutorial with a great simple (and very common) scenario.

I now have a clearer idea of Redux and how it actually works.

Grazzie mille!

Valentino Gagliardi

27TH MARCH 2018 AT 9:25 AM

Thanks for you comment Silvia! I'm flattered!

Super Sac

27TH MARCH 2018 AT 12:31 PM

Really nice guide.
My special thanks is due to you.
Thanks again.

Lucas Ponzo

28TH MARCH 2018 AT 11:02 PM

Simple, direct and effective! Thank you so much!

Dwi Setiyadi

29TH MARCH 2018 AT 8:12 AM

My bad, i use combineReducers and i forgot to add initialState. Please ignore my comment above. Thanks for nice tutorial.

David Marsland

29TH MARCH 2018 AT 4:52 PM

Excellent tutorial, thanks so much.
Everything in your Redux and React tutorial worked great and your explanations are excellent.

One minor issue I had was getting your bootstrap styles to work. I added

// src/js/index.js
import 'bootstrap/dist/css/bootstrap.min.css';

Then:
npm i bootstrap –save-dev

That worked for me. Any other suggestions?

thanks,
David

Valentino Gagliardi

29TH MARCH 2018 AT 7:46 PM

Hi David, thanks for reading. You're right, I changed the starting repo to https://github.com/valentinogagliardi/webpack-4-quickstart which does not have Bootstrap in the head tag. Good catch!

Aishwarya

29TH MARCH 2018 AT 5:35 PM

For some reason, I had this unjustified fear associated with learning redux. This tutorial is great and not a bit overwhelming. Thanks for taking the time to write this tutorial, Valentino!

Valentino Gagliardi

29TH MARCH 2018 AT 7:47 PM

You know Aishwarya I felt the same at the time! I was literally scared to learn Redux 🙂
Happy coding!

Yogesh

31ST MARCH 2018 AT 10:48 AM

This article is really awesome. This is meant for those who just started with Redux. Even I had a lot of fear with Redux. But this article really simplifies everything.
I read this article once and everything was kind of clear.But it will take some time for sure.
I really appreciate you for this !!!! Good luck

master

31ST MARCH 2018 AT 8:47 PM

Thank you Yogesh!

Rada

2ND APRIL 2018 AT 6:49 PM

I was about to rip my hair off when trying to understand Redux. This tutorial gave me back the hope. Everything makes so much more sense now. Thank you sooooo much!!!

Valentino Gagliardi

3RD APRIL 2018 AT 9:06 AM

Cool! Thanks for reading!

Mahesh

3RD APRIL 2018 AT 11:24 AM

Hi Valentino,

Thanks for the super-simple intro to Redux!

Some questions/suggestions:
1. Why is immutability of state such a big deal (any benefits)? It will be good if you can touch upon this.
2. It will be more helpful if you can begin with an example WITHOUT Redux, as a stateful React component, pointing out the issues in it and then show how Redux solves it. Can you modify this tutorial to such a format? (I am just pointing to the fact that learning is more effective when you start from the familiar and then move to the unfamiliar)

Thanks again for the great tutorial!!

Valentino Gagliardi

9TH APRIL 2018 AT 11:47 AM

Mahesh thank you for your comment. It is unlikely I'll change the tutorial's structure. However I'll take your suggestions in account for the future. Cheers!

Sergio

7TH APRIL 2018 AT 9:13 PM

Thanks for the POST!

Easy way to understand React and Redux!

Statbat

10TH APRIL 2018 AT 2:06 AM

By far the best tutorial on React Redux out there. I wish book authors can also read and get some tips.
It is simple, clear and not confusing. I can appreciate the simplicity and the way all is explained. Kudos!

Valentino Gagliardi

12TH APRIL 2018 AT 4:44 PM

Thank you! I appreciate your words!

Lavanya

11TH APRIL 2018 AT 12:02 PM

Great work

Valentino Gagliardi

12TH APRIL 2018 AT 4:39 PM

Thank you!

Bob

14TH APRIL 2018 AT 5:47 PM

Wow, that was super helpful. I'm learning redux for a new app I will build and was confused for days where to even start. This took me a big step forward on my journey to the store.

I was a little confused about the placement of Files during the tutorial. The github-repo saved me at the end.

Thanks from Berlin,
bob

Valentino Gagliardi

16TH APRIL 2018 AT 5:10 PM

Ciao from Italy! Thank you for the feedback Bob!

cil

15TH APRIL 2018 AT 2:04 AM

Thank you so much for the tutorial. I feel like I have a handle on React and Redux separately, but was unsure about React-Redux bindings. Would love to see more tutorials, maybe with thunks??

SirMcPotato

15TH APRIL 2018 AT 5:35 PM

Well, as a React/Redux beginner, i haven't yet read the entire article that I can already say it's an excellent resource! Concise,detailed, very well written with pertinent examples, all I need to learn efficiently how to use Redux. Thanks a lot! I'd give you a beer for that awesome tutorial 😛

Valentino Gagliardi

16TH APRIL 2018 AT 5:14 PM

Thank you Sir Potato!

Palash Taneja

16TH APRIL 2018 AT 5:02 PM

Good stuff! Clearest tutorial on Redux, that I've seen!

Jaydeep Maun

18TH APRIL 2018 AT 6:19 AM

Nice Article , Explained in easy to understand flow. I was able to follow each step without difficulty . I would sub title the Blog as Redux Simplified .
Thanks a lot for the blog.

Valentino Gagliardi

18TH APRIL 2018 AT 7:30 AM

Thanks for your feedback Jaydeep!

Harpreet Gill

18TH APRIL 2018 AT 1:13 PM

Hi VALENTINO,

First of all, Thank you for such a helpful tutorial.
I am following all the instructions in the tutorial and creating my first React + Redux app.
When I fired npm start. It is giving me following error in browser console.

ERROR in ./src/js/reducers/index.js
Module build failed: SyntaxError: Unexpected token (8:15)

6 | switch (action.type) {
7 | case ADD_ARTICLE:
> 8 | return { ...state, articles: [...state.articles, action.payload] };
| ^
9 | default:
10 | return state;
11 | }

Can you please guide me to make it work ?

Thanks
Harpreet Gill

Harpreet Gill

18TH APRIL 2018 AT 1:22 PM

Hi VALENTINO,

I have found the issue. I forgot to update .babelrc file.

Thanks for your tutorial.

Harpreet

Valentino Gagliardi

18TH APRIL 2018 AT 4:26 PM

You're welcome

Daria

18TH APRIL 2018 AT 5:58 PM

Thank you, your tutorial was very helpful!

Digvijay

18TH APRIL 2018 AT 9:41 PM

I am Using combineReducers and getting "Uncaught TypeError: Cannot convert undefined or null to object". Without the combineReducer it is working fine. Below is snippet of my Reducers.

article.js:
import { ADD_ARTICLE } from "../actiontypes/action-types";

const initialState = {
articles: []

```
};

const articleReducer = (state = initialState, action) => {

switch (action.type){
case ADD_ARTICLE:
return {...state, articles: [...state.articles, action.payload] };
default:
return state;
}
};

export default articleReducer;
```

reducers\index.js

```
import { combineReducers } from 'redux';
import articleReducer from './article';

export default combineReducers({
articles : articleReducer
});
```

prasannajit

22ND APRIL 2018 AT 10:13 AM

Very helpful. Clear and concise.

yakov

23RD APRIL 2018 AT 7:42 AM

Thanks a lot.

Wondeful.

Recommended.

aFrontEndDev

30TH APRIL 2018 AT 4:25 PM

Absolutely wonderful! The best free Redux beginner's guide i have found accross the web !

Thanks a lot my friend 🙂

sajith

9TH MAY 2018 AT 12:34 PM

wonderful wonderful and wonderful… Thanks

IK

10TH MAY 2018 AT 8:10 AM

Thank you for the tutorial. I was looking at many sites to grab the concept of Redux, however had difficulties to understand.
Your tutorial is really easy and flawless. I really appreciate your work.

Valentino Gagliardi

17TH MAY 2018 AT 12:48 PM

Thank you!

Stephen

11TH MAY 2018 AT 8:01 PM

Thanks Valentino!

Exactly what I needed. Much appreciated.

Valentino Gagliardi

17TH MAY 2018 AT 12:49 PM

Hey Stephen I'm glad it helped! Cheers

Matt

16TH MAY 2018 AT 6:23 PM

This looks great, and I'm going to step though it as you've laid it out. One thing, this github command doesn't work:

git clone git@github.com:valentinogagliardi/webpack-4-quickstart.git

But this one does:
git clone https://github.com/valentinogagliardi/webpack-4-quickstart.git

Just putting this here to help out others that want to use this tutorial. Thanks!

Valentino Gagliardi

17TH MAY 2018 AT 12:54 PM

Hey Matt you're right, I've fixed the instructions

TommyBB

17TH MAY 2018 AT 10:08 AM

Really good article. Helped me a lot.

Valentino Gagliardi

17TH MAY 2018 AT 12:52 PM

I'm glad it helped!

Ing Developer

17TH MAY 2018 AT 4:29 PM

Thank you so much for all your knowledge, that was so amazing, easy and fast to understand the basis for this topic that is Redux.

James

18TH MAY 2018 AT 6:40 PM

Amazing article. Very well written and organized

A.A Dotun

21ST MAY 2018 AT 11:17 AM

Thanks Valentino Gagliardi,
I found this interesting and you really save my life from those four folders: Actions , reducers, store, and constants. i read poeples code but never understand the concept because i dont know how they are link together but now ,i can chase my dream. Thanks man.

Valentino Gagliardi

21ST MAY 2018 AT 1:01 PM

Thanks for your kind words!

A.A Dotun

21ST MAY 2018 AT 1:37 PM

But in addition, i would like if you can do a write up on two entry in webpack , am currently working on personal project like that, my reason for doing this is to learn more,like react, webpack and others.
In this project i have two entry, and the reasons for those entries are because,
my home page as different dependecy from others e.g like my styling(css), but since the output are bundled and release on a single page and ,i need to load my dependency differently so it wont conflict one another.
this is my webpack.common.js

// import path from ‘path’
const path = require(‘path’)

```
module.exports = {
devServer: {
contentBase: './client/dist'
},
output: {
filename: '[name].bundle.js',
path: path.resolve(__dirname, 'dist'),
publicPath: '/'
},
watch: true,
watchOptions: {
poll: 1000,
aggregateTimeout: 1000,
ignored: /node_modules/
},
resolve: {
extensions: ['.jsx', '.js']
},
module: {
rules: [
{
test: /\.s?css$/,
use: ['style-loader', 'css-loader', 'sass-loader']
},
{
test: /\.(png|gif|jpe?g|svg)$/i,
use: ['file-loader']
},
{
test: /.jsx?$/,
enforce: 'pre',
use: [
{
loader: 'babel-loader'
}
],
include: path.join(__dirname, '/client')
},
{
test: /\.(woff|woff2|eot|ttf|otf)$/,
use: ['file-loader']
}
]
}
}
```

and my webpack.dev.js

```
const merge = require('webpack-merge')
const path = require('path')
const webpack = require('webpack')
const HtmlWebpackPlugin = require('html-webpack-plugin')
const CleanWebpackPlugin = require('clean-webpack-plugin')
const common = require('./webpack.common.js')
// to allow Dotenv files
const Dotenv = require('dotenv-webpack')

module.exports = merge(common, {
mode: 'development',
entry: {
app: './src/index.js',
print: './client/index.jsx'
},
devtool: 'cheap-eval-source-map',
devServer: {
contentBase: './dist'
},
plugins: [
new CleanWebpackPlugin(['dist']),
new webpack.HotModuleReplacementPlugin(),
new webpack.NoEmitOnErrorsPlugin(),
new Dotenv({
path: './.env',
safe: false
}),
new HtmlWebpackPlugin({
template: './client/index.html',
filename: 'index.html',
inject: 'body'
})
],
optimization: {
splitChunks: {
cacheGroups: {
commons: {
name: 'commons',
chunks: 'initial',
minChunks: 2,
minSize: 0
}
}
```

```
},
occurrenceOrder: true // To keep filename consistent between different modes (for example building only)
},
stats: {
colors: true
},
output: {
filename: '[name].bundle.js',
path: path.resolve(__dirname, 'dist'),
publicPath: '/'
}
})
```

So ,please how do i go about it, thanks in anticipation.