Redux

1. 下载 redux

```
npm i redux
```

2. 配置仓库

数据初始化：

1. 创建一个存储数据的仓库

```
import { createStore } from "redux";
const store = createStore();
```

2. 设置初始数据

   Redux 中的 state 统一交给 reducer 处理。

```
export const count = (state = 0, action) => {
    return state;
}
```

3. 将数据添加到仓库

```
import { createStore, combineReducers } from "redux";
import { count } from "./counter/reducers.js";
const store = createStore(
    combineReducers({ count })
);
```

3. 连接组件与仓库

1. 下载 react-redux

```
npm i react-redux
```

2. 连接 store 与组件

```
import { Provider } from "react-redux";
import store from "./store/store.js"

export default function App() {
    return (
        <Provider store={store}>
            <RootRouter></RootRouter>
        </Provider>
    )
}
```

3. 组件访问仓库

```
import React, { Component } from 'react'
import { connect } from "react-redux";

class Counter extends Component {
    render() {
        return (
            <>
                <h1>计数器：{this.props.count}</h1>
            </>
        )
    }
}

// state 获取到的是仓库中所有数据
const mapStateToProps = state => {
    return { count: state.count };
}
export default connect(mapStateToProps)(Counter);
```

4. redux-actions

```
npm i redux-actions
```

1. state 在 store 内
2. 通过 reducer 修改 state（reducer 函数的返回值会覆盖掉 store 中的旧 state）
3. 通过 dispatch 触发 reducer 的执行
    1. 调用 dispatch 时传递一个 action 对象：{ type: '任务一'}

拆分：

1. reducers.js：所有任务对应的操作
2. actions.js：{ type: '任务一'}、{ type: '任务二'}、{ type: '任务三'} .....
3. actions-type.js：'任务一'、'任务二' ....