
WEICHUANG
React
Weiss Field Watch (White Dial)
$950.00
Canvas & Suede Backpack
$165.00
2 ITEMS IN YOUR CART
Marschall Pack
QTY: 1
$359.00
EDC Kit 2
QTY: 1
$229.00
Handmade Picking Baskets
$129.00
$45.00
SHOP NOW
CONDITION

第一节 脚手架

React搭建脚手架

1、安装node　　node –v (Node >= 6)　npm –v(5.2.0+)

2、安装react 脚手架

(sudo) npm install -g create-react-app

3、create-react-app命令来创建react项目

create-react-app react-app

npm cache clean –force (清空缓存)

cd react-app 进入项目目录

npm start 启动项目

第二节 React Router

唯创网讯
WEICHUANGWANGXUN

安装 react router

```
npm install react-router-dom
```

引入 所需对象

```
import React from "react";
import { BrowserRouter as Router, Route, Link } from "react-router-dom";
```

import { BrowserRouter as Router, Route, Link } from "react-router-dom";

把**BrowserRouter** 重命名为 **Router** 也可不修改

路由基本组件

1、 <BrowserRouter>：使用 HTML5 提供的 history API 来保持 UI 和 URL 的同步；

<HashRouter>：使用 URL 的 hash (例如：window.location.hash) 来保持 UI 和 URL 的同步；

2、Link是react路由中的点击切换到哪一个组件的链接

3、Route代表了你的路由界面，path代表路径，component代表路径所对应的界面。

4、switch 仅渲染与当前位置匹配的第一个子元素。

5、exact 路由严格匹配模式 例： /link’与’ /’ 正常情况下它们是匹配的。严格模式下他们是不匹配的

简单路由跳转

```
<li>
  <Link to="/about">About</Link>
</li>
```

```
<Route path="/about" component={About} />
```

```
const About = () => (
  <div>
    <h2>About</h2>
  </div>
);
```

唯创网讯
WEICHUANGWANGXUN

URL参数

```
<Link to="/a/5">A页面</Link>
```

```
<Route path="/a/:id"  component={PageA}></Route>
```

```
return (
    <div>
        {console.log(this.props.match.params.id)}
        A页面---
    </div>
)
```

match 可接收
路由的默认对象

<Route>的渲染方式

component： 一个React组件。当带有component参数的route匹配成功后，route会返回一个新的元素，其为component参数所对应的React组件

render： 一个返回React element的函数[注5]。当匹配成功后调用该函数。该过程与传入component参数类似。

```
<Route path="/render"  render={()=><h1>render出的页面</h1>}></Route>
```

children： 一个返回React element的函数。与上述两个参数不同，无论route是否匹配当前location，其都会被渲染。

唯创网讯
WEICHUANGWANGXUN

重定向 Redirect

Redirect重定向是路由的重定向，应该写在组件Route中，一般使用render来实现它

```
<p><Link to='/class1/redirect'>redirect</Link></p>
```

```
<Route path="/class1/redirect" render={()=>(
  <Redirect to="/class1/protected"/>
)}/>
```

```
import {
  BrowserRouter as Router,
  Route,
  Link,
  Redirect,
  withRouter
} from "react-router-dom";
```

唯创网讯
WEICHUANGWANGXUN

自定义链接

```
<OldSchoolMenuLink to="/about" label="About" />
```

```
const Jump = (props) => {
    return <Link to={props.to}>jump</Link>
}
```

```
const OldSchoolMenuLink = ({ label, to, activeOnlyWhenExact }) => (
  <Route
    path={to}

    children={({ match }) => (
      <div className={match ? "active" : ""}>
        {match ? "> " : ""}
        <Link to={to}>{label}</Link>
      </div>
    )}
  />
);
```

官方案例：https://reacttraining.com/react-router/web/example/custom-link

编程式导航

```
<RouterChange/>
```

```
const RouterChange = withRouter(({history})=>(
  <div>
    <button onClick={()=>{history.push("/class1/protected")}}>跳转</button>
  </div>
))
```

react-router 提供了一个withRouter组件 withRouter可以包装任何自定义组件，将react-router 的 history,location,match 三个对象传入。 无需一级级传递react-router 的属性，当需要用的router 属性的时候，将组件包一层withRouter，就可以拿到需要的路由信息

唯创网讯
WEICHUANGWANGXUN

阻止跳转 Prompt

```
import { BrowserRouter as Router, Route, Link, Prompt } from "react-router-dom";
```

```
<Prompt
  when={this.state.isBlocking}
  message={location =>
    `你确定要离开当前页面跳转至 ${location.pathname}`
  }
/>
```

使用Prompt 组件： when 为布尔值是否开启验证
message: string 当用户离开当前页面时，设置的提示信息。
<Prompt message="确定要离开？" />
message: func 当用户离开当前页面时，设置的回掉函数

唯创网讯
WEICHUANGWANGXUN

NavLink

<NavLink>是<Link>的一个特定版本，会在匹配上当前的url的时候给已经渲染的元素添加参数

- **activeClassName(string)：** 设置选中样式，默认值为**active**

- **activeStyle(object)：** 当元素被选中时，为此元素添加样式

- **isActive(func)**判断链接是否激活的额外逻辑的功能

```
// activeClassName选中时样式为selected
<NavLink
  to="/faq"
  activeClassName="selected"
>FAQs</NavLink>

// 选中时样式为activeStyle的样式设置
<NavLink
  to="/faq"
  activeStyle={{
    fontWeight: 'bold',
    color: 'red'
  }}
>FAQs</NavLink>
```

模糊匹配

```
<Link to="/class2/about">About Us (static)</Link>
<Link to="/class2/company">Company (static)</Link>
<Link to="/class2/kim">Kim (dynamic)</Link>
<Link to="/class2/chris">Chris (dynamic)</Link>
<Switch>
  <Route path="/class2/about" component={About} />
  <Route path="/class2/company" component={Company} />
  <Route path="/class2/:user" component={User} />
</Switch>
```

如果想进行模糊匹配例如about 和 company 匹配到相应的路由而Kim 和 Chris 匹配到user则必须将带有参数的route放置到最后

第二节 Redux

唯创网讯
WEICHUANGWANGXUN

Redux

Redux 是 JavaScript 状态容器，提供可预测化的状态管理

首先明确一点，**Redux** 是一个有用的架构，但不是非用不可。事实上，大多数情况，你可以不用它，只用 **React** 就够了

有人说："如果你不知道是否需要 Redux，那就是不需要它。"

Redux 的创造者 **Dan Abramov**又补充说：

"只有遇到 React 实在解决不了的问题，你才需要 Redux 。"

- 用户的使用方式复杂
- 不同身份的用户有不同的使用方式（比如普通用户和管理员）
- 多个用户之间可以协作
- 与服务器大量交互，或者使用了WebSocket
- View要从多个来源获取数据

Redux

- 官网地址： **https://redux.js.org/**

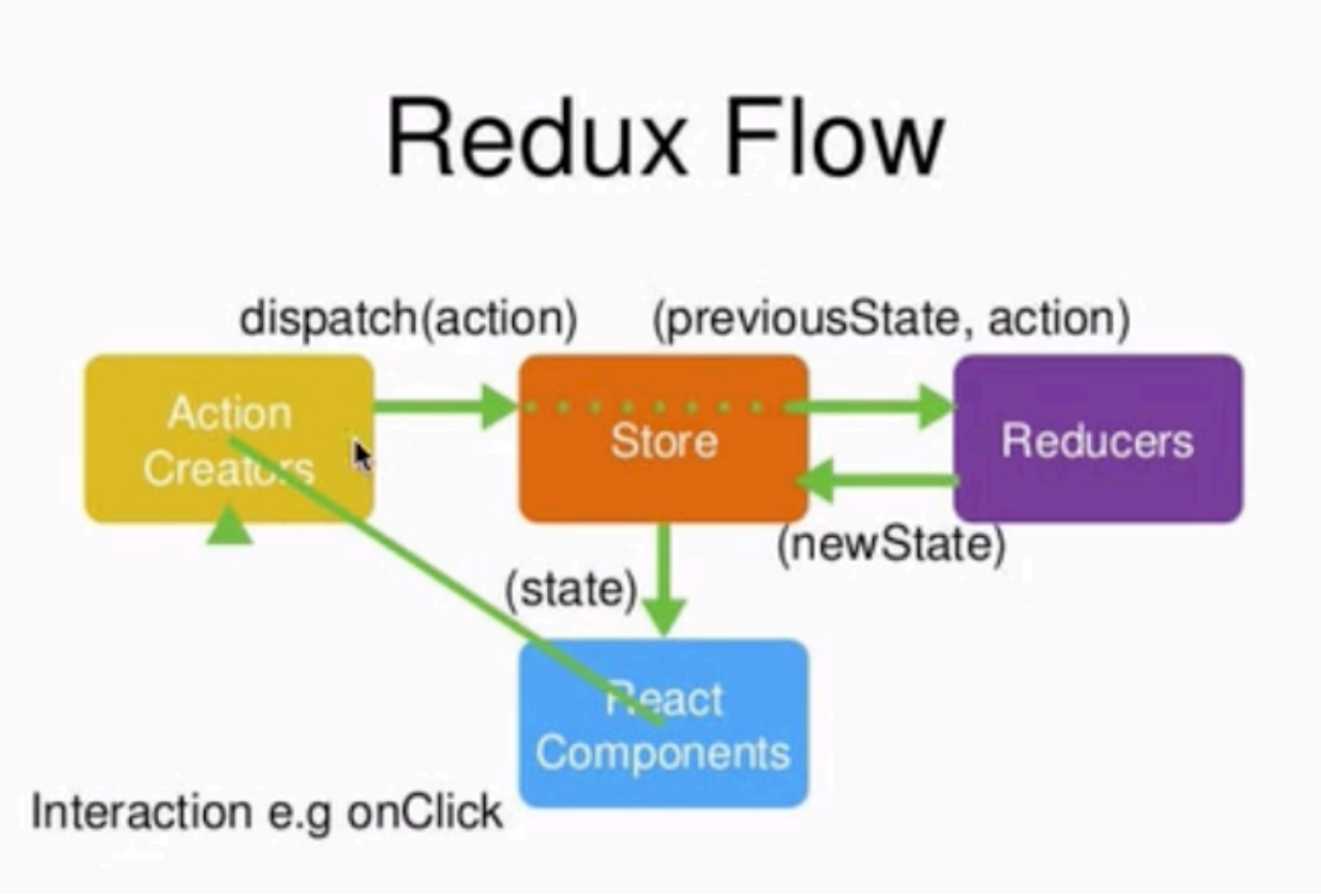
- 中文网址： **http://www.redux.org.cn/**

- 安装：

  (sudo) npm install redux react-redux

Redux

应用中所有的 **state** 都以一个对象树的形式储存在一个单一的 ***store*** 中。惟一改变 **state** 的办法是触发 ***action***，一个描述发生什么的对象。为了描述 **action** 如何改变 **state** 树，你需要编写 ***reducers***。



store

- 引入react-redux 提供的provider组件,可以让容器组件拿到state（state来自store）

<Provider store= {store}> </Provider>

- store保存数据的地方

```
import { createStore } from 'redux';

const initialState = 0;
//创建store  存储数据的
const store = createStore(()=>{

},initialState);

export default store;
```

Action

- Action本质上是 JavaScript 普通对象。我们约定，action 内必须使用一个字符串类型的 type 字段来表示将要执行的动作。多数情况下，type 会被定义成字符串常量。当应用规模越来越大时，建议使用单独的模块或文件来存放 action。

```
const addTodo = (text) => {
    return {
        type:'ADD_TODO',
        text
    }
}
export default addTodo;
```

Reducer

- Store 收到 Action 以后，必须给出一个新的 State，这样 View 才会发生变化。这种 State 的计算过程就叫做 Reducer。
- reducer 就是一个纯函数，接收旧的 state 和 action，返回新的 state。
  - combineReducers() 所做的只是生成一个函数，这个函数来调用你的一系列 reducer，每个 reducer 根据它们的 key 来筛选出 state 中的一部分数据并处理，然后这个生成的函数再将所有 reducer 的结果合并成一个大的对象

```
import { combineReducers } from 'redux';
const count = (state = 0,action)=> {
    switch (action.type){
        case 'ADD':
            return state+1;
        default:
            return state;
    }
}
const reducer = combineReducers({
    count
})
export default reducer;
```

获取state

- Connect(react-redux提供的)

  store 里能直接通过 store.dispatch() 调用 dispatch() 方法调用action，但是多数情况下你会使用 react-redux提供的 connect() 帮助器来调用。

- mapStateToProps 将state转化成props属性

```
const mapStateToProps = state => ({
    todos:state.todos
})
```

- mapStateToProps 将state转化成props属性

```
const  mapDispatchToProps = (dispatch) => {
    return {
      add: () => dispatch(add())
    };
}
```


Thank you
谢 谢 观 看