



Wear
Weiss Field
Watch
(White Dial)
\$950.00

Accessories
Canvas
& Suede
Backpack
\$165.00

ing Camping
Hatchet
\$66.00
\$42.00
-30%

Soft Leather
Weekender P

Home Decor
Handmade Picking Baskets
\$129.00

W
WEICHUANG

React

2 ITEMS IN YOUR CART

Marshall Pack
Qty: 1
\$359.00

EDC Kit 2
Qty: 1
\$229.00

Mue
Shaving
\$45.00

SHOP NOW

CONDITION
New
Manufacturer Refurbished
Working

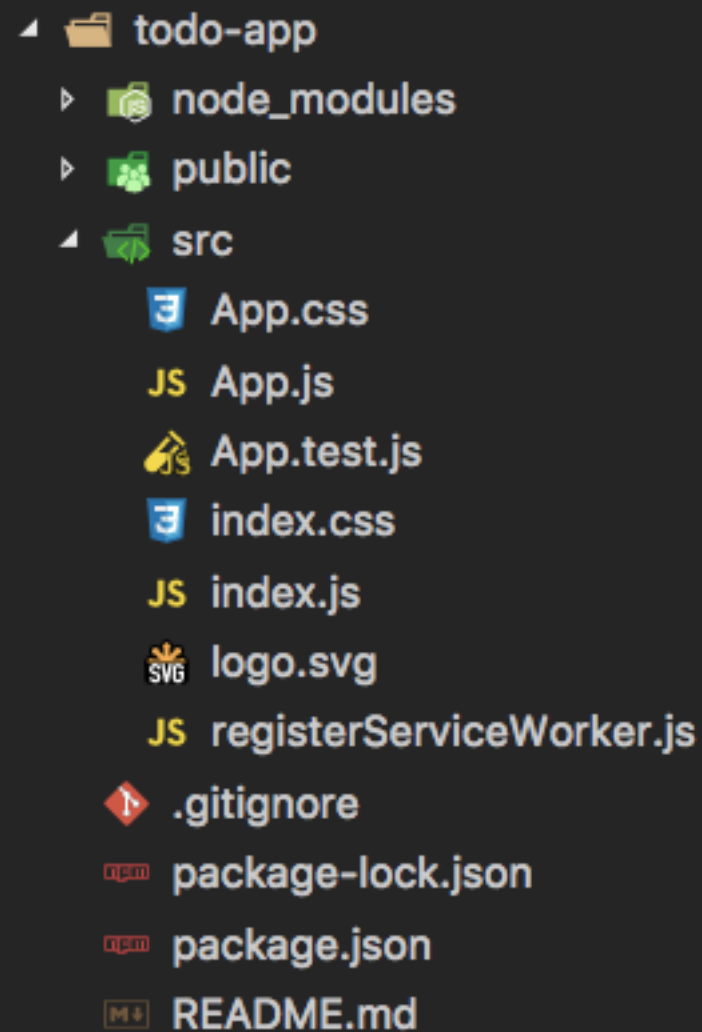
DISCOUNT
TOTAL

第一节 React APP

通过命令安装 react APP

```
npm install -g create-react-app  
create-react-app my-app
```

```
cd my-app  
npm start
```



The image shows a file explorer view of a React application directory structure. The root directory is 'todo-app'. It contains several subdirectories and files:

- node_modules**: A directory containing installed packages.
- public**: A directory for static assets.
- src**: A directory for the application source code, containing:
 - App.css**: A CSS file.
 - App.js**: A JavaScript file.
 - App.test.js**: A JavaScript file for testing.
 - index.css**: A CSS file.
 - index.js**: A JavaScript file.
 - logo.svg**: An SVG image file.
 - registerServiceWorker.js**: A JavaScript file for service workers.
- .gitignore**: A file used to specify files to be ignored by Git.
- package-lock.json**: A file that locks the version of the packages installed.
- package.json**: A file that describes the application's dependencies and scripts.
- README.md**: A file containing information about the project.

第二节 React Router

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

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

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

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

URL参数

```
<li>
  <Link to="/yahoo">Yahoo</Link>
</li>
```

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

```
const Child = ({ match }) => (
  <div>
    <h3>ID: {match.params.id}</h3>
  </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。
- 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。

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";
```

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

```
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>

```
<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

```
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

<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

```
1 // activeClassName选中时样式为selected
2 <NavLink
3   to="/faq"
4   activeClassName="selected"
5 >FAQs</NavLink>
6
7 // 选中时样式为activeStyle的样式设置
8 <NavLink
9   to="/faq"
10  activeStyle={{
11    fontWeight: 'bold',
12    color: 'red'
13  }}
14 >FAQs</NavLink>
15
```

```
<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



Thank you

谢

谢

观

看