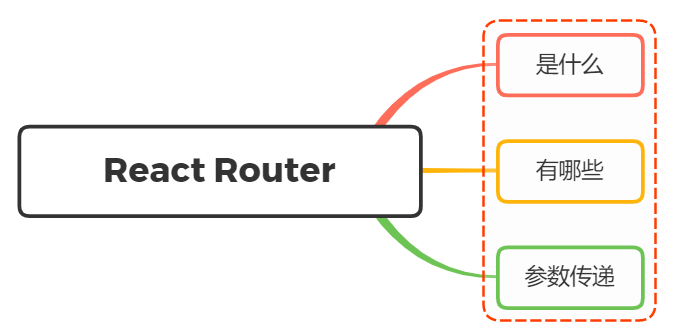
# 面试官：说说你对React Router的理解？常用的Router组件有哪些？



## 一、是什么

react-router等前端路由的原理大致相同，可以实现无刷新的条件下切换显示不同的页面

路由的本质就是页面的URL发生改变时，页面的显示结果可以根据URL的变化而变化，但是页面不会刷新

因此，可以通过前端路由可以实现单页(SPA)应用

react-router主要分成了几个不同的包：

* react-router: 实现了路由的核心功能
* react-router-dom： 基于 react-router，加入了在浏览器运行环境下的一些功能
* react-router-native：基于 react-router，加入了 react-native 运行环境下的一些功能
* react-router-config: 用于配置静态路由的工具库

## 二、有哪些

这里主要讲述的是react-router-dom的常用API，主要是提供了一些组件：

* BrowserRouter、HashRouter
* Route
* Link、NavLink
* switch
* redirect

### BrowserRouter、HashRouter

Router中包含了对路径改变的监听，并且会将相应的路径传递给子组件

BrowserRouter是history模式，HashRouter模式

使用两者作为最顶层组件包裹其他组件

import { BrowserRouter as Router } from "react-router-dom";  
  
export default function App() {  
 return (  
 <Router>  
 <main>  
 <nav>  
 <ul>  
 <li>  
 < a href=" ">Home</ a>  
 </li>  
 <li>  
 < a href="/about">About</ a>  
 </li>  
 <li>  
 < a href="/contact">Contact</ a>  
 </li>  
 </ul>  
 </nav>  
 </main>  
 </Router>  
 );  
}

### Route

Route用于路径的匹配，然后进行组件的渲染，对应的属性如下：

* path 属性：用于设置匹配到的路径
* component 属性：设置匹配到路径后，渲染的组件
* render 属性：设置匹配到路径后，渲染的内容
* exact 属性：开启精准匹配，只有精准匹配到完全一致的路径，才会渲染对应的组件

import { BrowserRouter as Router, Route } from "react-router-dom";  
  
export default function App() {  
 return (  
 <Router>  
 <main>  
 <nav>  
 <ul>  
 <li>  
 < a href="/">Home</ a>  
 </li>  
 <li>  
 < a href="/about">About</ a>  
 </li>  
 <li>  
 < a href="/contact">Contact</ a>  
 </li>  
 </ul>  
 </nav>  
 <Route path="/" render={() => <h1>Welcome!</h1>} />  
 </main>  
 </Router>  
 );  
}

### Link、NavLink

通常路径的跳转是使用Link组件，最终会被渲染成a元素，其中属性to代替a标题的href属性

NavLink是在Link基础之上增加了一些样式属性，例如组件被选中时，发生样式变化，则可以设置NavLink的一下属性：

* activeStyle：活跃时（匹配时）的样式
* activeClassName：活跃时添加的class

如下：

<NavLink to="/" exact activeStyle={{color: "red"}}>首页</NavLink>  
<NavLink to="/about" activeStyle={{color: "red"}}>关于</NavLink>  
<NavLink to="/profile" activeStyle={{color: "red"}}>我的</NavLink>

如果需要实现js实现页面的跳转，那么可以通过下面的形式：

通过Route作为顶层组件包裹其他组件后,页面组件就可以接收到一些路由相关的东西，比如props.history

const Contact = ({ history }) => (  
 <Fragment>  
 <h1>Contact</h1>  
 <button onClick={() => history.push("/")}>Go to home</button>  
 <FakeText />  
 </Fragment>  
);

props中接收到的history对象具有一些方便的方法，如goBack，goForward,push

### redirect

用于路由的重定向，当这个组件出现时，就会执行跳转到对应的to路径中，如下例子：

const About = ({  
 match: {  
 params: { name },  
 },  
}) => (  
 // props.match.params.name  
 <Fragment>  
 {name !== "tom" ? <Redirect to="/" /> : null}  
 <h1>About {name}</h1>  
 <FakeText />  
 </Fragment>  
)

上述组件当接收到的路由参数name 不等于 tom 的时候，将会自动重定向到首页

### switch

swich组件的作用适用于当匹配到第一个组件的时候，后面的组件就不应该继续匹配

如下例子：

<Switch>  
 <Route exact path="/" component={Home} />  
 <Route path="/about" component={About} />  
 <Route path="/profile" component={Profile} />  
 <Route path="/:userid" component={User} />  
 <Route component={NoMatch} />  
</Switch>

如果不使用switch组件进行包裹

除了一些路由相关的组件之外，react-router还提供一些hooks，如下：

* useHistory
* useParams
* useLocation

### useHistory

useHistory可以让组件内部直接访问history，无须通过props获取

import { useHistory } from "react-router-dom";  
  
const Contact = () => {  
 const history = useHistory();  
 return (  
 <Fragment>  
 <h1>Contact</h1>  
 <button onClick={() => history.push("/")}>Go to home</button>  
 </Fragment>  
 );  
};

### useParams

const About = () => {  
 const { name } = useParams();  
 return (  
 // props.match.params.name  
 <Fragment>  
 {name !== "John Doe" ? <Redirect to="/" /> : null}  
 <h1>About {name}</h1>  
 <Route component={Contact} />  
 </Fragment>  
 );  
};

### useLocation

useLocation 会返回当前 URL的 location对象

import { useLocation } from "react-router-dom";  
  
const Contact = () => {  
 const { pathname } = useLocation();  
  
 return (  
 <Fragment>  
 <h1>Contact</h1>  
 <p>Current URL: {pathname}</p >  
 </Fragment>  
 );  
};

## 三、参数传递

这些路由传递参数主要分成了三种形式：

* 动态路由的方式
* search传递参数
* to传入对象

### 动态路由

动态路由的概念指的是路由中的路径并不会固定

例如将path在Route匹配时写成/detail/:id，那么 /detail/abc、/detail/123都可以匹配到该Route

<NavLink to="/detail/abc123">详情</NavLink>  
  
<Switch>  
 ... 其他Route  
 <Route path="/detail/:id" component={Detail}/>  
 <Route component={NoMatch} />  
</Switch>

获取参数方式如下：

console.log(props.match.params.xxx)

### search传递参数

在跳转的路径中添加了一些query参数；

<NavLink to="/detail2?name=why&age=18">详情2</NavLink>  
  
<Switch>  
 <Route path="/detail2" component={Detail2}/>  
</Switch>

获取形式如下：

console.log(props.location.search)

### to传入对象

传递方式如下：

<NavLink to={{  
 pathname: "/detail2",   
 query: {name: "kobe", age: 30},  
 state: {height: 1.98, address: "洛杉矶"},  
 search: "?apikey=123"  
 }}>  
 详情2  
</NavLink>

获取参数的形式如下：

console.log(props.location)

## 参考文献

* http://react-guide.github.io/react-router-cn/docs/API.html#route