一、todoList案例相关知识点

```
1.拆分组件、实现静态组件，注意：className、style的写法
2.动态初始化列表，如何确定将数据放在哪个组件的state中?
            --某个组件使用：放在其自身的state中
            --某些组件使用：放在他们共同的父组件state中（官方称此操作为：状态提升）
3.关于父子之间通信：
        1.【父组件】给【子组件】传递数据：通过props传递
        2.【子组件】给【父组件】传递数据：通过props传递，要求父提前给子传递一个函数
4.注意defaultChecked 和 checked的区别，类似的还有：defaultValue 和 value
5.状态在哪里，操作状态的方法就在哪里
```

二、github搜索案例相关知识点

```
1.设计状态时要考虑全面，例如带有网络请求的组件，要考虑请求失败怎么办。
2.ES6小知识点：解构赋值+重命名
            let obj = {a:{b:1}}
            const {a} = obj; //传统解构赋值
            const {a:{b}} = obj; //连续解构赋值
            const {a:{b:value}} = obj; //连续解构赋值+重命名
3.消息订阅与发布机制
            1.先订阅，再发布（理解：有一种隔空对话的感觉）
            2.适用于任意组件间通信
            3.要在组件的componentWillUnmount中取消订阅
4.fetch发送请求（关注分离的设计思想）
            try {
                const response= await fetch(`/api1/search/users2?
q=${keyWord}`)
                const data = await response.json()
                console.log(data);
            } catch (error) {
                console.log('请求出错',error);
            }
```

三、路由的基本使用

```
    1.明确好界面中的导航区、展示区
    2.导航区的a标签改为Link标签
            <Link to="/xxxxx">Demo</Link>
    3.展示区写Route标签进行路径的匹配
            <Route path='/xxxx' component={Demo}/>
    4.<App>的最外侧包裹了一个<BrowserRouter>或<HashRouter>
```

四、路由组件与一般组件

```
    1.写法不同：
            一般组件：<Demo/>
            路由组件：<Route path="/demo" component={Demo}/>
    2.存放位置不同：
            一般组件：components
            路由组件：pages
```

```
        3.接收到的props不同：
                    一般组件：写组件标签时传递了什么，就能收到什么
                    路由组件：接收到三个固定的属性
                                        history:
                                                    go: ƒ go(n)
                                                    goBack: ƒ goBack()
                                                    goForward: ƒ goForward()
                                                    push: ƒ push(path, state)
                                                    replace: ƒ replace(path,
state)
                                        location:
                                                    pathname: "/about"
                                                    search: ""
                                                    state: undefined
                                        match:
                                                    params: {}
                                                    path: "/about"
                                                    url: "/about"
```

五、NavLink与封装NavLink

```
            1.NavLink可以实现路由链接的高亮，通过activeClassName指定样式名，以改变高亮的样
式
            2.封装NavLink：<MyNavLink/> 一般组件，

            应用：
              <MyNavLink to="/about" a={1} b={2}>About<MyNavLink>
              <MyNavLink to="/home" a={1} b={2}>Home<MyNavLink>
              //About和Home也会作为children属性传入自定义组件里的<NavLink/>中

            <MyNavLink>:
              export default class MyNavLink extends Component {
                render () {
                  return (
                     <NavLink activeClassName="atguigu" className="list-group-
item" {...this.props}/>
                  )
                }
              }
```

六、Switch的使用

```
            1.通常情况下，path和component是一一对应的关系。
            2.Switch可以提高路由匹配效率(单一匹配)。
```

七、解决多级路径刷新页面样式丢失的问题

```
            1.public/index.html 中 引入样式时不写 ./ 写 / （常用）
            2.public/index.html 中 引入样式时不写 ./ 写 %PUBLIC_URL% （常用）
            3.使用HashRouter
```

八、路由的严格匹配与模糊匹配

```
1.默认使用的是模糊匹配（简单记：【输入的路径】必须包含要【匹配的路径】，且顺序要一致）
2.开启严格匹配：<Route exact={true} path="/about" component={About}/>
3.严格匹配不要随便开启，需要再开，有些时候开启会导致无法继续匹配二级路由
```

九、Redirect的使用

```
1.一般写在所有路由注册的最下方，当所有路由都无法匹配时，跳转到Redirect指定的路由
2.具体编码：
        <Switch>
            <Route path="/about" component={About}/>
            <Route path="/home" component={Home}/>
            <Redirect to="/about"/>
        </Switch>
```

十、嵌套路由

```
1.注册子路由时要写上父路由的path值
2.路由的匹配是按照注册路由的顺序进行的
```

十一、向路由组件传递参数

```
1.params参数
            路由链接(携带参数)：<Link to='/demo/test/tom/18'}>详情</Link>
            注册路由(声明接收)：<Route path="/demo/test/:name/:age" component={Test}/>
            接收参数：this.props.match.params
2.search参数
            路由链接(携带参数)：<Link to='/demo/test?name=tom&age=18'}>详情</Link>
            注册路由(无需声明，正常注册即可)：<Route path="/demo/test" component={Test}/>
            接收参数：this.props.location.search
            备注：获取到的search是urlencoded编码字符串，需要借助querystring解析：
                import qs from "querystring"
                const a={id:1,name:'12'}
                qs.stringify(a)-----id=1&&name='12'
                let str="car=benci&&count=1"
                qs.parse(str)------{car:'benci',count:1}
3.state参数
            路由链接(携带参数)：<Link to={{pathname:'/demo/test',state:{name:'tom',age:18}}}>详情</Link>
            注册路由(无需声明，正常注册即可)：<Route path="/demo/test" component={Test}/>
            接收参数：this.props.location.state
            备注：刷新也可以保留住参数
```

十二、编程式路由导航

借助this.prosp.history对象上的API对操作路由跳转、前进、后退
-this.prosp.history.push()
-this.prosp.history.replace()
-this.prosp.history.goBack()
-this.prosp.history.goForward()
-this.prosp.history.go()
withRouter(一般组件)----使一般组件具备路由组件特有的api：history location search,这样，在一般组件上也可以使用history的跳转功能等。

十三、BrowserRouter与HashRouter的区别

1.底层原理不一样：
BrowserRouter使用的是H5的history API，不兼容IE9及以下版本。
HashRouter使用的是URL的哈希值。
2.path表现形式不一样
BrowserRouter的路径中没有#,例如：localhost:3000/demo/test
HashRouter的路径包含#,例如：localhost:3000/#/demo/test
3.刷新后对路由state参数的影响
(1).BrowserRouter没有任何影响，因为state保存在history对象中。
(2).HashRouter刷新后会导致路由state参数的丢失！！！
4.备注：HashRouter可以用于解决一些路径错误相关的问题。

十四、antd的按需引入+自定主题

1.安装依赖：yarn add react-app-rewired customize-cra babel-plugin-import less less-loader
2.修改package.json

```
....
    "scripts": {
        "start": "react-app-rewired start",
        "build": "react-app-rewired build",
        "test": "react-app-rewired test",
        "eject": "react-scripts eject"
    },
....
```

3.根目录下创建config-overrides.js

```
//配置具体的修改规则
const { override, fixBabelImports,addLessLoader} = require('customize-cra');
module.exports = override(
    fixBabelImports('import', {
        libraryName: 'antd',
        libraryDirectory: 'es',
        style: true,
    }),
    addLessLoader({
        lessOptions:{
            javascriptEnabled: true,
            modifyVars: { '@primary-color': 'green' },
        }
    }),
);
```

4.备注：不用在组件里亲自引入样式了，即：import 'antd/dist/antd.css'应该删掉