1. react-router：

**被分成3个包**：react-router，react-router-dom，react-router-native

React-router提供核心的组件和函数，react-router-dom和react-router-native 提供运行环境所需的特定组件；

**React-router** ：提供了router的核心api 包括 router，route，switch等，但他没有提供dom操作进行跳转的api；

**React-router-dom**：提供了BrowserRouter，route，link等api。我们可以通过dom的事件控制路由；

**React-router 提供两种路由器类型**：

1. BrowserRouter 服务区动态请求数据时用BrowserRouer

使用的是HTML5 history api（pushState, replaceState, popState)来使url动态改变的

1. HashRouter 被用于静态网站 通过hash值对路由进行控制 url后会有#号；

**Route：**

是路由的原材料，控制路径对应显示的组件；经常用的是exact、path以及component属性。

1.该标签有三种渲染方式component、render、children（绝大多数情况使用component组件就好了）

2.三种渲染方式都会得到三个属性match、history、location；

3.渲染组件时，route props跟着一起渲染

4.children方式渲染会不管地址栏是否匹配都渲染一些内容，在这里加动画一时很常见的做法

exact控制到匹配的路径就不在匹配

path标识路由的路径

component显示的是路由对应的组件；

**Link和NavLink:**

两者都可以控制路由跳转 不同都是NavLink的API更多

Link： 主要API是to to可以接受一个string或者object 来控制url

NavLink：可以为当前选中的路由设置类名，样式以及回调函数等

<NavLink>是<Link>的一个特定版本, 会在匹配上当前URL的时候会给已经渲染的元素添加样式参数；

activeClassName，当地址匹配时添加相应class；

activeStyle，当地址匹配时添加相应style；

exact，当地址完全匹配时，才生效；

isActive，添加额外逻辑判断是否生效。

**Match：**

是在使用router之后被放入props的一个属性，在class创建的组件中我们需要this.props.match 来获取match之中的信息；

params: 通过解析URL中动态的部分获得的键值对；

isExact: 当为true时，整个URL都需要匹配；

path: 在需要嵌套<Route/>的时候用到；

url: 在需要嵌套<Link/>的时候会用到；

获取方式: 以this.props.match方式

**Switch :**

常常会用来包裹route，他里面不能放其他元素，用来只显示一个路由

**Link与a标签的区别是 link可以知道链接路径的变化，可以通过这点来对link进行设计**

如：

<li><Link to ="/about" activeStyle={{color：'red'}}>关于</link></li>

或者：

<li><Link to="/about" activeClassName="active">About</Link></li>

**Redirect 组件 (重定向)：**

有些时候，我们匹配一个路径，但是可能这个路径，我们更希望它指向一个新的展示界面，而不是它原本的路径匹配界面。

Redirect组件的必须属性是to属性，表示重定向的新地址。

Redirect组件的基本使用方式：

<Route path="/home" render={()=><Redirect to="/other"/>}/>

<Redirect/>可以写在<Route/>的render属性里面，也可以跟<Route/>平级；

to: 依旧是可以跟字符串或对象

push: 添加该属性时，地址不会被覆盖，而是添加一条新纪录；

from: 重定向，与<Route/>平级时。

**withRouter组件：**

withRouter高阶组件的方式去进行动态路由的跳转