至此，我们有关React 的开发技术要点，基本上阐述完毕了，这一节 我们把学过的所有知识点运用起来，构建一个单页APP实例，通过本节实践，我们可以体会到，现代单页APP与传统的多页应用有怎样的不同！

单页APP的一个重大改进就是，页面切换时，不需要重新从服务器加载新的网页，这使得单页APP能够用户一种流畅而又良好的使用体验。

具体的代码胜过千言万语，没有什么比自己亲手把它做出来后所得到的体会更深刻了。（调出应用实例），我们这个单页应用实例，需要依赖一个React 组件，他叫React Router. 我们点击头部的选项卡之后，下面的显示内容会根据所点击的选项卡自动切换。接下来，我们将一步一步的把这个应用实例做出来，完成后，你不但学会了如何使用React Router控件，还能够把过往学过的知识点复习一遍，加深理解和掌握。

第一步还是老样子，先完成基本框架代码。

```
<!DOCTYPE html>
<html charset="utf-8">
    <head>
        <title>React Singe Page App</title>
        <script src="react.js"></script>
        <script src="react-dom.js"></script>
        <script src="browser.min.js"></script>
        <style type="text/css"></style>
    </head>

    <body>
        <div id="container"></div>
        <script type="text/babel">
            var destination =
            document.querySelector("#container");
            ReactDOM.render(
                <div>
                    Hello!
                </div>,
                destination
                );
```

```
        </script>
    </body>
</html>
```

先加载上面代码，没出错的话，我们再继续。接下来我们要把React Router 组件引入到我们的项目中，所以增加下面一行代码：

```
<script src="ReactRouter.min.js"></script>
```

单页APP应用，他有一个不变的基础部分，我们称之为APP frame,它其实是一个容器，把应用中，那些经常变动的界面部分包容起来。还有一些视觉元素是保持不变的，例如页眉，页脚，导航栏等等。我们的App frame将包含一个导航栏，一个空白显示区用来显示可变的界面部分。我们将用一个React组件来实现这种基础功能。代码如下：

```
var App = React.createClass({
                render: function() {
                        return (
                                <div>
                                    <h1>Simple SPA</h1>
                                    <ul className="header">
                                        <li>李白</li>
                                        <li>杜甫</li>
                                        <li>李商隐</li>
                                    </ul>

                                    <div className="content">
                                    </div>
                                </div>
                                )
                    }
            })


            ReactDOM.render(
                <div>
                    <App/>
```

```
                </div>,
                destination
                );
```

完成代码后加载，大家可以看到一个列表显示在页面的旁边。现在的页面看起来太简单，我们后面慢慢改进，让页面的展示效果丰满起来。现在，大家注意看，我们将把React Router组件引入到项目中，我们把代码做一个修改：

```
ReactDOM.render(
                <ReactRouter.Router
                history={ReactRouter.hashHistory}>
                    <ReactRouter.Route path="/"
                    component={App}>

                    </ReactRouter.Route>
                </ReactRouter.Router>
                ,
                destination
                );
```

大家可能对这段代码还不理解，没关系，完成后，先把代码加载到浏览器看看。

我们先通过ReactRouter.Router 把ReactRouter组件里面的一个子组件引入到代码中，Router这个组件的作用，是处理有关页面路由的逻辑，在这个组件里面，我们做的叫做“路由配置”，大家看里面有关属性叫path,path对应的是一个斜杠，这个斜杠表示的是网页应用的根路径。 component属性跟path对应起来，那么当前代码的意思是，如果用户在浏览器中打开的是网页APP的根路径，那么就把APP组件绘制到界面上。

由于刚才我们加载页面时，url对应的就是应用的根路径，所以Router把App组件加载到页面里，调用它的render函数，这就是为何我们看到的情形跟我们原来直接把App控件显示到页面上的效果一样。

显示主页 React router 将根据用户输入的URL的情况，选择适当的组件在

界面上绘制。当前当用户访问根目录时，我们在界面上显示的是App组件，在App组件中，特意留下了一个空白区域，也就是类名是content的div标签，现在我们构造一个控件叫Home,然后把它显示到这个空白区域。

代码如下：

```
var Home = React.createClass({
                render: function() {
                return (
                        <div>
                            <h2>望天门山</h2>
                            <p>天门中断楚江开，碧水东流至此回</p>
                            <p>两岸青山相对出，孤帆一片日边来 </p>
                        </div>
                        );
                }
            });

            var App = React.createClass({
                render: function() {
                        return (
                                <div>
                                    <h1>Simple SPA</h1>
                                    <ul className="header">
                                        <li>李白</li>
                                        <li>杜甫</li>
                                        <li>李商隐</li>
                                    </ul>

                                    <div className="content">
                                        <Home/>
                                    </div>
                                </div>
                                )
                }
            })
```

上面代码完成后，可以看到Home组件中的内容被绘制到界面上。当前我们

设计上有个问题是，Home控件的显示，我们是写死在代码中的，我们期望的是，把上方列表中的三个选项当成选项卡，用户点一下，就显示相应的内容，所以接下来我们还需要做相应修改。

在继续修改业务逻辑前，我们当前APP的显示太朴素，太难看了，我们现在界面设计上处理一下，让我们的界面变得好看点。所以我们先添加一些css代码妆点界面：

```
<style type="text/css">
            body {
                background-color: #FFCC00;
                padding: 20px;
                margin: 0;
            }

            h1, h2, p, ul, li {
                font-family: Helvetica, Arial, sans-serif;
            }

            ul.header li {
                display: inline;
                list-style-type: none;
                margin: 0;
            }

            ul.header {
                background-color: #111;
                padding: 0;
            }

            ul.header li a {
                color: #FFF;
                font-weight: bold;
                text-decoration: none;
                padding: 20px;
                display: inline-block;
            }

            .content {
```

```
            background-color: #FFF;
            padding: 20px;
        }

        .content h2 {
            padding: 0;
            margin: 0;
        }

        .content li {
            margin-bottom: 10px;
        }
    </style>
```

添加上面代码后，我们的界面好看多了，只不过三个列表选项不见了，中间多了一条黑色的粗线，我们一会就处理这个问题。

避免ReactRouter 前缀 大家看到，我们在使用ReactRouter 控件时，前面总有一个前缀就是ReactRouter. ,这个前缀作用不大，平白无故增添我们敲键盘的无用功，现在我们想办法把这个前缀给去掉。

这里我们用到最新版的js语法规范，我们添加一下代码：

```
var {
            Router,
            Route,
            IndexRoute,
            IndexLink,
            Link,
            hashHistory
        } = ReactRouter;
```

它的作用类似于c++ 的include 或是java 的import, 有了这段代码，我们就可以把ReactRouter. 这个前缀给去掉了：

```
ReactDOM.render(
            <Router history={hashHistory}>
```

```
            <Route path="/"
            component={App}>

            </Route>
        </Router>
        ,
        destination
        );
```

修改后，再次加载页面，我们看到页面还是像以前一样没有变化。

回到我们的业务逻辑，当前我们的Home组件的显示方式是写死的，页面一加载，Home组件就绘制出来，我们想要的是，用户点击哪个选项卡，我们就显示对应的页面，要实现这个功能，我们就不能写死Home组件的显示方式，必须依赖于Router组件来控制哪个组件可以在页面上显示。

我们做以下修改：

```
var App = React.createClass({
        render: function() {
                return (
                        <div>
                            <h1>Simple SPA</h1>
                            <ul className="header">
                                <li>李白</li>
                                <li>杜甫</li>
                                <li>李商隐</li>
                            </ul>

                            <div className="content">
                                {this.props.children}
                            </div>
                        </div>
                        )
        }
});


ReactDOM.render(
```

```
<Router history={hashHistory}>
    <Route path="/"
    component={App}>
        <IndexRoute component={Home}/>
    </Route>
</Router>
,
destination
);
```

添加上面代码，然后加载页面，你会看到页面显示情况不变，但是我们显示Homek控件的方式变了，它的显示不再是写死的，一个重大变动是，在content区域，我们通过{this.props.children}属性来对应要显示的组件。同时在ReactDOM.render函数中，我们使用了一个控件叫IndexRoute, 这个控件来决定哪个组件能够在页面上展示。或许现在你对Router的机制还是很困惑，我们接着往下走，慢慢的你就能掌握它的原理。

原来我们有三个列表项，但他们被一条黑色粗线给覆盖住了。这条粗线本来是最为背景的。我们接下来着手处理这个问题。同时，我们还希望这三个列表项是可以被点击的，一旦用户点击某个列表项之后，我们的程序能得到相应的点击事件，并且知道哪个列表项被用户点击了，然后通过Router控件，显示相应的组件内容。

我们先给每个列表项添加一个Link控件，这样每个列表项就变成可以点击的形式了，修改代码如下：

```
var App = React.createClass({
                render: function() {
                        return (
                                <div>
                                    <h1>Simple SPA</h1>
                                    <ul className="header">
                                        <li>
                                          <Link to="/">李白</Link>
                                        </li>
                                        <li>
                                          <Link to="/df">杜甫</Lin
```

```
                    </li>
                    <li>
                      <Link to="/lsy">李商隐</l
                    </li>
                </ul>

                <div className="content">
                    {this.props.children}
                </div>
            </div>
            )
        }
    });
```

上面的代码修改后，大家可以看到，原来被黑线覆盖的三个选项出现了，同时每个选项是可点击的，大家可以尝试点击一下，然后看看在浏览器中的url显示，例如我们点击了“杜甫”选项后，在url的后面添加了"/df", 这个内容恰好就是我们设置在Link 控件中，to属性所对应的内容，点击“李商隐”选项，同理可以看到url后面添加了"/lsy"。

现在，我们点击选项，主界面显示的还是Home控件的内容，接下来我们先添加另外两个控件，然后在把他们的显示跟对应选项链接起来。代码如下：

```
var Dufu = React.createClass({
        render: function() {
            return (
                <div>
                    <h2>登高</h2>
                    <p>风急天高猿啸哀,渚清沙白鸟飞回
                    <p>无边落木萧萧下,不尽长江滚滚来
                    <p>万里悲秋常作客,百年多病独登台
                    <p>艰难苦恨繁霜鬓,潦倒新停浊酒杯
                </div>
                );
        }
    });
```

```
var Lsy = React.createClass({
    render: function() {
            return (
                    <div>
                        <h2>锦瑟</h2>
                        <ol>
                        <li>锦瑟无端五十弦,一段一柱思年华
                        <li>庄生晓梦迷蝴蝶,望帝春心托杜鹃
                        <li>沧海月明珠有泪,蓝田日暖玉生烟
                        <li>此情可待成追忆,只是当时已惘然
                        </ol>
                    </div>
                    )
    }
});
```

然后在ReactDOM.renderh函数中做如下修改：

```
ReactDOM.render(
                <Router history={hashHistory}>
                    <Route path="/"
                    component={App}>
                        <IndexRoute component={Home}/>
                        <Route path="df" component={Dufu}/>
                        <Route path="lsy" component={Lsy}/>
                    </Route>
                </Router>
                ,
                destination
                );
```

添加上面代码后，我们重新加载页面，然后点击某个选项，例如我点击“杜甫”选项，那么对应的组件Dufu及其相关内容就可以显示出来了。

到此，我们应用的开发接近尾声了，我们需要完成最后一点功能，让我们整个应用看起来更精致一些，我们现在点击某个选项时，对应的内容是能

正确显示了，但是我们无法分辨当前哪个选项被选中了，所以，我们添加一点小功能，让用户知道当前他选择的是哪个选项，实现该功能的办法是，在ReactRouter中添加一个属性叫activeClassName，然后在css中添加相关的显示属性. 由此我们的代码修改如下：

```
var App = React.createClass({
                render: function() {
                        return (
                                <div>
                                    <h1>Simple SPA</h1>
                                    <ul className="header">
                                        <li>
                                          <IndexLink to="/"
                                         activeClassName="active">
                                              李白
                                          </IndexLink>
                                        </li>
                                        <li>
                                          <Link to="/df"
                                           activeClassName="active
                                          >杜甫</Link>
                                        </li>
                                        <li>
                                          <Link to="/lsy"
                                          activeClassName="active"
                                          >李商隐</Link>
                                        </li>
                                    </ul>

                                    <div className="content">
                                        {this.props.children}
                                    </div>
                                </div>
                                )
                }
            });
```

然后添加css相关代码：

```
.active {
                background-color: #0099FF;
            }
```

完成上面代码后，重新加载页面，点击某个选项卡，我们可以看到，某个选择的选项卡会被高亮选中。

总结： 本节，我们详细介绍了ReactRouter控件的用法，同时结合以前我们所学的知识点，开发除了一个较为简单的单页APP,从这个例子我们可以感受到单页APP的本质，他在进行界面切换时，根本不需向服务器发出请求，也不需要反复加载新页面，由此他能够给用户带来近似于桌面应用的体验。