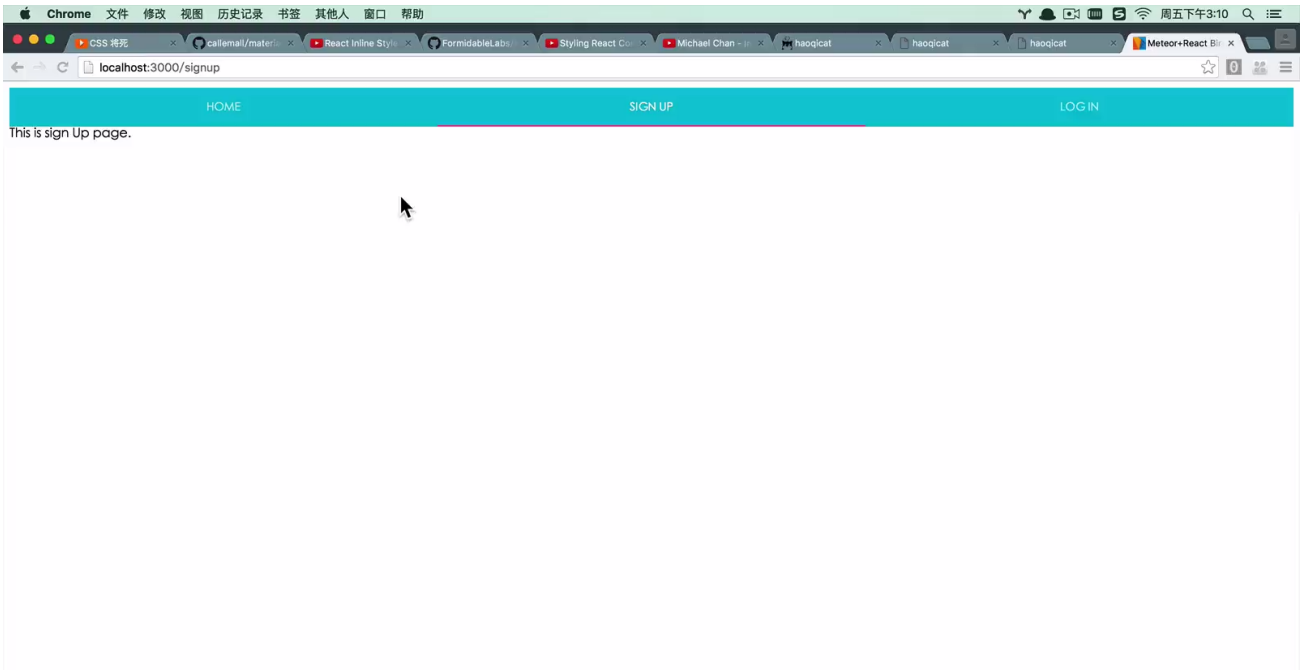

导航栏状态控制

本节其实是个非常好的例子来揭示组件其实是状态机的概念，通过操作组件的 state 状态值，改变组件的显示状态，另外还会涉及到 React 组件生命周期方法的使用。

React state 状态值的妙用

先来研究下可控模式的 Tabs 组件，到底怎样控制导航栏的下划线停留位置呢？很简单，Tabs 组件有一个 value 属性可以用来决定哪一个 Tab 组件处于选中状态，也就是红色下划线会提留的位置。我们先测试一下，修改 `NavBar.jsx` 文件，给 Tabs 组件添加一个 value 属性，其值与第三个 Tab 组件的 value 值保持一致。

```
<Tabs value='/login'>
```

查看页面会发现导航栏中的 `LOG IN` 标签出现了一条红色下划线。但问题是，Tabs 组件的 value 值是固定的，也就是后续我们无论怎样操作，红色下划线都不会改变位置。

显然这个 value 值要时刻改变，所以我们用一个状态值来控制它。

```
<Tabs value={this.state.tabIndex}>
```

这样 value 值就会随着 state 变量 `tabIndex` 值的变化而变化。此时访问页面，会发现导航栏不见了，浏览器控制台报告错误信息：

```
Uncaught TypeError: Cannot read property 'tabIndex' of null
```

所以我们要先初始化 `tabIndex` 变量为 `/`，默认让 `HOME` 标签是活跃的。

```
constructor(props) {
  super(props);
  this.state = {tabIndex: '/'};
}
```

变量初始化完成之后，点击各个 tab 标签，红色下划线位置不变，所以我们还需要改变 `tabIndex` 的值，在 handleChange 方法中添加一行代码：

```
handChange(value) {
  this.setState({tabIndex: value});
}
```

让 `tabIndex` 的值与被选中的 Tab 组件的 value 值保持一致。

查看更改：React state 状态值的妙用

React 生命周期方法的使用

不过，这会儿刷新页面，红色下划线依然停留在 `HOME` 标签的位置，因为 NavBar 组件会重新挂载，state 变量 `tabIndex` 又初始化为 `/`。所以我们需要再改变一下 `tabIndex` 的值，那这次如何获取所选中的 Tab 组件的 value 值呢？下面，我们要借助 React Router 的 `context.router` 变量提供的 isActive 接口获取当前活跃的路径。

注意：当判断 `/` 路径的时候，要添加第二个参数 `true`，确保活跃路径是 `/`，要不然会把其它的路径都判定为 `/`，红色下划线又会始终提留在第一个标签下。

```
this.context.router.isActive('/', ture)
```

那我们什么时候再重新设置一下 `tabIndex` 的值呢？这次 React 的生命周期方法就派上用场了。在组件将要挂载的时候，会执行 componentWillMount 生命周期方法，我们在这个生命周期方法中改变 state 变量 `tabIndex`：

```
componentWillMount() {
  this.setState({
    tabIndex: this.getSelectedIndex()
  });
}
```

查看更改：刷新页面保持 Tab 组件的选中状态

这样，重新刷新页面的时候，红色下划线的位置就不会改变了。

不过，现在的导航栏依然不完美，还存在一个问题，当点击浏览器中的前进和后退按钮的时候，红色下划线的位置不会根据路径的改变而改变，所以我们还得重新设置一下 `tabIndex` 变量，这回是在 React 的 componentWillReceiveProps 生命周期方法中设置。

查看更改：点击浏览器中的前进后退按钮改变导航栏状态

到目前为止，我们导航栏的功能终于让人满意了，但是样式还有待完善，导航栏两边和顶部都有留白，下节将讲述如何定义组件样式。