1. &#39;对应 '
2. [网站元数据meta标签的含义和用法](http://blog.csdn.net/freshlover/article/details/25322839)
3. 提示用户浏览器版本过旧需要更新，点击链接后跳转到的网站<http://browsehappy.com/>
4. [svg Sprite技术介绍 张鑫旭](http://www.zhangxinxu.com/wordpress/2014/07/introduce-svg-sprite-technology/)
5. [根据背景色自动改变文字的颜色](http://ilucas.me/2015/10/29/reverse-text-color-mix-blend-mode/)
6. [WebKit最新特性srcset简介](http://developer.51cto.com/art/201309/410720.htm)
7. [属性初始化器语法](https://doc.react-china.org/docs/handling-events.html#%E5%90%91%E4%BA%8B%E4%BB%B6%E5%A4%84%E7%90%86%E7%A8%8B%E5%BA%8F%E4%BC%A0%E9%80%92%E5%8F%82%E6%95%B0)
8. [不用bind的安装](https://babeljs.io/docs/plugins/transform-class-properties/)

search = () => {

alert(1);

}

1. react动态绑定类名
   1. 安装 npm install classnames
   2. import classNames from 'classnames'
   3. classNames({  
       'swiper-pagination-bullet': true,  
       'swiper-pagination-bullet-active': index === i  
      })
2. [父子组件之间的通讯](http://taobaofed.org/blog/2016/11/17/react-components-communication/)
3. JSX中行间使用 SVG
   1. npm i svg-inline-loader --save-dev
   2. ,  
      {  
       test: /\.svg$/,  
       loader: 'svg-inline-loader'  
      }
   3. xmlns:xlink和xlink:href分别写为xmlnsXlink和xlinkHref
4. [在 React中使用 Svg](http://blog.csdn.net/deeplies/article/details/78072497)
5. 组件之间的事件传递（一个组件的事件触发另一个组件的状态的改变，通过调用父级的方法，改变父级的state（重新渲染组件，重新传递props），通过props来改变组件
6. 当组件有动画并且动画会改变state时，动画可以写在componentWillReceiveProps来防止组件陷入无限渲染的死循环中，如果开始就需要渲染一次动画，可以再在componentDidMount中再调用一次。
7. 随机色'#'+(0x1000000+(Math.random())\*0xffffff).toString(16).substr(1,6)
8. 列表延迟渲染的实现
   1. react-transition-group 实现动画,延迟思路,创建一个新的组件,以this.props.children传进去, 挨个往数组中加,加完用setState渲染
   2. 使用[rc-queue-anim](https://motion.ant.design/api/queue-anim#animConfig)
9. 不需要在每个组件里面手动编写 shouldComponentUpdate。

PureRenderMixin 的mixin形式

// react官方demo

import PureRenderMixin from 'react-addons-pure-render-mixin';

class FooComponent extends React.Component {

constructor(props) {

super(props);

this.shouldComponentUpdate = PureRenderMixin.shouldComponentUpdate.bind(this);

}

1. 因为对象的引用关系，导致在子组件里面接受到的 this.props.value 与 nextProps.value 永远都是相等的。

解决方案:

* immutable.js：FaceBook官方提出的不可变数据解决方案，主要解决了复杂数据在deepClone和对比过程中性能损耗
* imutable 在shouldComponentUpdate中的使用:

import { is } from 'immutable';

shouldComponentUpdate: (nextProps = {}, nextState = {}) => {

const thisProps = this.props || {}, thisState = this.state || {};

if (Object.keys(thisProps).length !== Object.keys(nextProps).length ||

Object.keys(thisState).length !== Object.keys(nextState).length) {

return true;

}

for (const key in nextProps) {

if (!is(thisProps[key], nextProps[key])) {

return true;

}

}

for (const key in nextState) {

if (thisState[key] !== nextState[key] || !is(thisState[key], nextState[key])) {

return true;

}

}

return false;

}

1. 路由切换动画加载

解决方案：在根组件中的componentWillReceiveProps监听（location.pathname），通过触发action来显示进度或者加载图标，在componentDidMount可以触发加载完成的action然后改变store,在监听这个store变化重新渲染，如果用按需加载的方式，那么就不用在componentWillReceiveProps看当前的url是否变化了，直接在按需加载的getComponent里写，