大地基础2

- https://zh-hans.reactjs.org/docs
- **--save 将保存配置信息到package.json。默认为dependencies节点中。**
- **--dev将保存配置信息DevDependencies节点中。**
- **因此：**
- **--save: 将保存配置信息到package.json的dependencies节点中。**
- **--save--dev:将保存配置信息到package.json的 DevDependencies节点中。**
- https://www.limitcode.com/detail/59a15b1a69e95702e0780249.html
-
- 域名、端口、协议三者有一个不同就会造成跨域问题。
- defaultProps和propTypes https://www.jianshu.com/p/959a064a2290
  - defaultProps：父子组件传值中，如果父组件调用子组件的时候不给子组件传值，则可以在子组件中使用defaultProps定义的默认值。
    - Header.**defaultProps**={
      - title:'标题'
    - }
  - propTypes：验证父组件传值的类型合法性。
    - import PropTypes from 'prop-types';
    - Header.**propTypes** = { //注意这里首字母小写
      - title ： **PropTypes.string** //注意这里首字母大写 定义属性类型为string
    - }
    - PropTypes类型列表：func\any(任何类型)\bool\array\number\object\string\symbol(ES6新增的symbol类型)\node\element(react element)
    - symbol：https://www.cnblogs.com/diligenceday/p/5462733.html
  - 一般来说，这两个都是用在子组件里。
- **获取服务器的数据 axios fetch jsonp**
  - axios https://www.npmjs.com/package/axios
  - fetch jsonp
    - 安装npm install fetch-jsonp --save
    - 引入 import fetchJsonp from 'fetch-jsonp'
    - 看文档使用https://www.npmjs.com/package/fetch-jsonp

- fetchJsonp('/users.jsonp')
  - .then(function(response){
    - return response.json();
  - }).then(function(json){
    - console.log(' parsed json ' , json)
  - }).catch(function(ex){
    - console.log(' parsed failed ' , ex)
  - })

- React生命周期函数 https://www.jianshu.com/p/514fe21b9914
  - 组件加载之前，组件加载完成，以及组件更新数据，组件销毁，触发的一系列的方法。这就是组件的 生命周期函数。
  - 组件加载的时候触发的函数：
    - constructor（不是生命周期 ）、**componentWillMount**、render（不是生命周期） 、**componentDidMount** （处理数据请求和DOM操作）
  - 组件数据更新的时候触发的生命周期函数：
    - shouldComponentUpdate、componentWillUpdate、render、componentDidUpdate
  - 在父组件里面改变props传值的时候触发的：
    - componentWillReceiveProps （然后执行组件数据更新的一系列函数）
  - 组件销毁的时候触发：
    - componentWillUnmount
  - 必须记住的……
    - 加载的时候：componentWillMount render componentDidMount（dom操作，后台数据获取）
    - 更新的时候：componentWillUpdate render componentDidUpdate
    - 销毁的时候：componentWillUnmount

- 路由
- **图片：**
  - 网络图片加载
    - 网络图片的加载，和一般的dom元素没有区别：
      -
    - 但是，图片一般是后台传过来的变量，我们一般在JSX语法的{ }中来表示

        - ****
    - 本地图片加载
        - require
            -
        - import
            - **import img from '../assets/logo.png'**
            - ****
- **react二维数组遍历解析：嵌套即可**
- **react获取动态路由数据：** this.props.match.params.id
- **react中JS实现跳转路由 （以用户登录为例）**
    - 1引入Redirect
    - 2创建flag，标识是否登录 (登录成功后修改flag，就会重新render，进而判断……)
    - 3执行判断flag决定是否跳转（render里面）
        - **<Redirect to={{pathname:'/'}} /> //跳转到跟目录**
- react路由的嵌套
- **react中将带标签的字符串转义为html解析** https://zh-hans.reactjs.org/docs/dom-elements.html
    - <p **dangerouslySetInnerHTML= { {__html : item.content } }** />
    - **dangerouslySetInnerHTML**
        - dangerouslySetInnerHTML是react为浏览器DOM提供 innerHTML的替换方案。通常来讲，使用代码直接设置HTML存在风险，因为很容易无意中使用用户暴露于跨站脚本（XSS）的攻击。因此，你可以直接在react中设置HTML，但当你想设置dangerouslySetInnerHTML时，需要向其传递包含key为__html的对象，以此来警你。例如：
            - function createMarkup() {
                - return **{__html: 'First · Second'};**
            - }
            - function MyComponent() {
                - return <div **dangerouslySetInnerHTML**={createMarkup()} />;
            - }
    -
- https://www.jianshu.com/p/b02f6b15cd09 Javascript语法规范错误提示代码

幕布 - 思维概要整理工具