REACT基础知识复习（一）：  
 1. REACT是FACE-BOOK公司开发的一款MVC版JS框架  
 MVC：Model（数据层）、VIEW（视图层）、CONTROLLER（控制层）  
 核心思想：通过数据的改变来影响视图的渲染（数据驱动）  
  
 2. 基于脚手架 CREATE-REACT-APP 快速构建一个REACT工程项目结构  
 自动安装REACT的核心组件：REACT/REACT-DOM  
 自动安装WEBPACK，并且完成相关的配置：  
 ->区分了开发环境和生产环境  
 ->安装BABEL以及对应的语言解析包，可以把REACT和ES6进行编译处理  
 ->安装CSS/STYLE/FILE等加载器，处理CSS等合并压缩的问题  
 ->安装了ES-LINT，可以进行代码检测  
 ->安装了很多的插件，可以实现JS和CSS以及HTML的分离，打包，压缩等  
 ->安装了WEBPACK-DEV-SERVER，可以在开发环境下，编译后自动创建服务，打开浏览器，当代码修改后，自动保存编译，页面自动刷新渲染等  
 ...  
  
 【使用脚手架】  
 ```  
 把脚手架安装到全局环境下，以后应用命令操作，完成项目结构的搭建  
 $ npm install create-react-app -g  
  
 创建项目结构目录  
 项目名遵循NPM发包命名规范：名字只能是/^[a-z0-9\_-]$/  
 $ create-react-app 项目名  
  
 特点：如果当前电脑安装了YARN，创建工程目录的时候，走的是YARN安装，YARN和NPM主体相同，但是处理起来还有一定的区别，所以我们以后继续向工程中安装模块以及执行配置脚本打包编译的时候，尽可能使用YARN，不建议和NPM混用  
 ```  
  
 【工程化目录】  
 |- node\_modules  
 | |- .bin 所有在本地可执行的命令脚本(react-scripts.cmd)  
 | |- ...  
 |  
 |- package.json 当前项目的配置清单  
 |  
 |- public 存放的是当前项目的HTML页面（有可能放一部分静态资源）  
 | |- index.html  
 | |- ...  
 |  
 |- src 存放的是项目需要的所有JS或者静态资源等（包括组件、STORE、路由、数据模型、AJAX请求等等内容 “我们开发的内容基本上所有东西都在SRC中写”）  
 | |- index.js 当前项目的入口文件  
 | |- ...  
  
 【暴露WEBPACK配置项】  
 脚手架构建项目的时候，为了结构的美化，把所有的WEBPACK配置等都隐藏到了NODE\_MODULES中（REACT-SCRIPTS中），真实项目中，我们经常会基于脚手架构建的结构自己在安装配置一些信息（例如：LESS处理的配置），此时我们需要把配置项暴露出来  
 `$ yarn eject` 此操作是不可逆转的（而且操作之前需要把所有修改的文件提交到GIT仓库中）  
  
 |- config  
 | |- webpack.config.dev.js 开发环境下的WP配置  
 | |- webpack.config.prod.js 生产环境下的WP配置  
 | |- paths.js 基本配置项（包含项目的入口信息）  
 | |- ...  
 |  
 |- scripts  
 | |- start.js / build.js / test.js 当我们执行yarn start/build/test的时候，执行的就是这三个JS文件  
  
 【可执行的本地脚本命令】  
 $ yarn start  
 ->创建一个端口号为3000，协议为HTTP的WEB服务  
 ->按照webpack.config.dev.js把项目编译  
 ->打开浏览器，预览我们正在开发的项目  
 ->当项目文件修改的时候，自动重新编译，浏览器页面自动刷新，展示最新的效果  
 ```  
 [WINDOWS]  
 $ set HTTPS=true&&yarn start  
 $ set PORT=1234&&yarn start  
  
 [MAC/LINUX]  
 $ HTTPS=true yarn start  
 $ PORT=1234 yarn start  
 ```  
  
 $ yarn build  
 ->生成一个build文件夹，存放最后打包的文件  
 ->基于webpack.config.prod.js，把项目进行编译打包  
 ->部署上线的时候，只需要把buid中的内容发布即可  
  
  
 【基于脚手架配置LESS】  
 安装LESS和对应的加载器  
 $ yarn add less less-loader  
  
 修改开发和生产环境下的WEBPACK配置项  
 ```  
 [DEV:159~193行]  
 {  
 test: /\.(css|less)$/,  
 use: [  
 require.resolve('style-loader'),  
 {  
 loader: require.resolve('css-loader'),  
 ...  
 },  
 ...  
 {  
 loader: require.resolve('less-loader')  
 },  
 ],  
 },  
  
 [PROD:169~212行]  
 {  
 test: /\.(css|less)$/,  
 loader: ExtractTextPlugin.extract(  
 Object.assign(  
 {  
 fallback: {  
 loader: require.resolve('style-loader'),  
 options: {  
 hmr: false,  
 },  
 },  
 use: [  
 {  
 loader: require.resolve('css-loader'),  
 ...  
 },  
 ...  
 {  
 loader: require.resolve('less-loader'),  
 }  
 ],  
 },  
 extractTextPluginOptions  
 )  
 ),  
 },  
 ```  
  
=================================  
REACT基础知识复习（二）：  
 1. REACT是基于独有的JSX语法实现视图（数据和HTML）渲染的  
  
 2. JSX语法  
 A：JSX语法的渲染使用的是ReactDOM.render  
 ```  
 ReactDOM.render([JSX元素]，[指定的容器]，[回调函数：当我们把JSX放到指定容器内，触发执行的函数]);  
 ```  
  
 B：JSX = JAVASCRIPT + XML（HTML）  
 ```  
 1. 不推荐存放JSX的容器是BODY（写BODY会报错），一般都是使用我么自己创建的一个元素（例如：创建#ROOT的DIV容器等）  
 ReactDOM.render(<h2>珠峰培训</h2>, root);  
  
 2. 不允许出现两个“根”元素，如果需要绑定复杂的结构，最外层嵌套一个容器做为根元素即可  
 ReactDOM.render(<h2>珠峰培训</h2><h3>哈哈</h3>, root); =>错误的  
 ReactDOM.render(<div>  
 <h2>珠峰培训</h2>  
 <h3>哈哈</h3>  
 </div>, root); =>正确的  
  
 3. 把数据嵌入到JSX中（不是嵌入到元素的属性中，而是正常的内容中）  
 => 可以嵌入变量或者直接的数据值  
 let name='xxx';  
 ReactDOM.render(<div>  
 <h2>{name}</h2>  
 <h3>{'哈哈'}</h3>  
 </div>, root);  
  
 => 不能嵌入对象（代指：{}、/^$/、日期对象、函数、或者数组中的某一项是前面的也不行 [一维简单的数据是可以的]）  
 ReactDOM.render(<div>  
 <h2>{{name:'xxx'}}</h2> NO  
 <h3>{new Date()}</h3> NO  
 <h3>{[12,23,34]}</h3> OK  
 <h4>{(() => {  
 return '呵呵'; OK:把自执行函数的结果嵌入进来  
 })()}</h4>  
 </div>, root);  
  
 => 可以嵌入基本类型值（null/undefined/布尔值都是空元素，也就是不显示任何的内容）  
  
 => 大括号中可以嵌入JS表达式（执行JS代码需要有返回结果的）  
 循环创建的JSX元素需要设置标识KEY，并且在当前循环的时候，这个KEY需要唯一；而使用MAP是因为它有返回值，返回的是替换后的数组；  
 ReactDOM.render(<ul>  
 {  
 data.map((item, index) => {  
 return <li key={index}>  
 {item.id}&nbsp;&nbsp;{item.title}  
 </li>;  
 })  
 }  
 </ul>, root);  
  
 使用三元运算符解决判断操作，（IF和SWITCH都不可以）  
 ReactDOM.render(<ul>  
 {name ? '哈哈' : '呵呵'}  
 </ul>, root);  
  
 4. 可以给JSX元素设置属性  
 =>属性值对应大括号中 对象、函数 都可以放（也可以放JS表达式）  
 =>STYLE属性值必须是对象（不能是字符串）  
 =>CLASS 用 CLASS-NAME 代替  
 =>...  
 ```  
  
=================================  
REACT基础知识复习（三）：  
 1. 生命周期函数  
  
 【调取组件】  
 constructor  
 componentWillMount 第一次渲染之前  
 render 渲染  
 componentDidMount 第一次渲染之后  
  
 【组件重新渲染：内部状态改变、传递给组件的属性改变】  
 状态改变：  
 shouldComponentUpdate  
 =>是否允许组件更新：返回TRUE是允许，返回FALSE则不再继续向下走  
 componentWillUpdate  
 =>更新之前：和SHOULD一样，方法中通过this.state.xxx获取的还是更新前的状态信息，方法有两个参数：nextProps/nextState存储的是最新的属性和状态信息  
 render 更新  
 componentDidUpdate 更新之后  
  
 属性改变：  
 componentWillReceiveProps(nextProps/nextState)  
 =>接收最新属性之前，基于this.props.xxx获取的是原有的属性信息，nextProps存储的是最新传递的属性信息  
 shouldComponentUpdate 是否允许组件更新  
 componentWillUpdate 更新之前  
 render 更新  
 componentDidUpdate 更新之后  
  
 【组件销毁】  
 componentWillUnmount 组件销毁之前  
  
 =====  
  
 组件的属性是只读的：只能调取组件时候传递进来，不能自己在组件内部修改（但是可以设置默认值和规则）  
  
 组件的状态是可读写的：状态改变会引发组件的重新更新（状态是基于setState改变）  
  
 组件实例上可以放一些信息：这些信息只是为了方便在组件内任意方法中获取和使用的  
  
 实例上挂载的REFS：就是用来操作DOM的  
  
 实例上挂载的context：是用来实现组件之间信息传递的