**react 中的ref是什么**

ref是React提供的用来操纵React组件实例或者DOM元素的接口

**1.字符串**通过 this.refs.a 来引用真实dom的节点 （dom节点上使用的）

**2.回调函数**回调函数就是在dom节点或组件上挂载函数，函数的入参是dom节点或组件实例，达到的效果与字符串形式是一样的，都是获取其引用。

**3.React.createRef()**在React 16.3版本后，使用此方法来创建ref。将其赋值给一个变量，通过ref挂载在dom节点或组件上，该ref的current属性将能拿到dom节点或组件的实例

**虚拟DOM和真实DOM的区别**



虚拟DOM

虚拟 DOM 是一个轻量级的 JavaScript 对象，它最初只是真实 DOM 的副本。它是一个节点树，将元素、它们的属性和内容列为对象及其属性。React 的 render 函数从 React 组件中创建一个节点树。然后它更新这棵树以响应数据模型中由用户或系统执行的各种操作引起的变化。

虚拟 DOM 分三个简单的步骤：

1.每当任何底层数据发生变化时，整个 UI 都会在虚拟 DOM 表示中重新呈现。

2.然后计算之前的 DOM 和新的 DOM 之间的差异。

3.计算完成后，真正的 DOM 将只更新实际更改的内容。

**react:props和state有什么区别**

1.state是组件自身数据的管理对象而props是别的组件传递来的数据的管理对象

2.state是可以修改的，而props是不能修改的

3.state只能在类组件中使用，而props既可以在类组件中使用也可以在函数组件中使用

**react diff 原理（常考，大厂必考）**

把树形结构按照层级分解，只比较同级元素。

给列表结构的每个单元添加唯一的 key 属性，方便比较。

React 只会匹配相同 class 的 component（这里面的 class 指的是组件的名字）

合并操作，调用 component 的 setState 方法的时候, React 将其标记为 dirty.到每一个事件循环结束, React 检查所有标记 dirty 的component 重新绘制.

选择性子树渲染。开发人员可以重写 shouldComponentUpdate 提高diff 的性能。

**reacte生命周期**

初始化阶段：

getDefaultProps:获取实例的默认属性

getInitialState:获取每个实例的初始化状态

componentWillMount：组件即将被装载、渲染到页面上

render:组件在这里生成虚拟的 DOM 节点

componentDidMount:组件真正在被装载之后

运行中状态：

componentWillReceiveProps:组件将要接收到属性的时候

调用

shouldComponentUpdate:组件接受到新属性或者新状态的

时候（可以返回 false，接收数据后不更新，阻止 render 调

用，后面的函数不会被继续执行了）

componentWillUpdate:组件即将更新不能修改属性和状态

render:组件重新描绘

componentDidUpdate:组件已经更新

销毁阶段：

componentWillUnmount:组件即将销毁

常用的生命周期函数

1.constructor：构造函数是react的第一个生命周期，这里一般放一些初始化的操作代码（如变量赋值、state初始化、bind绑定等）

2.render：是生命周期函数中唯一一个必须要写的函数

3.componentDidMount：组件挂在完毕周期，该周期一般用于网络请求

4.getSnap (13.280, -0.820, -5.82%)shotBeforeUpdate:该周期不能单独使用，一定要配合componentDidUpdate一起使用，否则报错，改周期函数需要有返回值，返回我们需要在更新前获取的数据

5.componentDidUpdate:组件更新完毕的周期函数，在组件渲染完毕后触发，该周期可能触发多次，第四个周期必须和这个周期一块用。第四个周期更新 这个周期接收

6.componentWillUnmount：组件解除挂载之前触发，一般在该周期中做组件的销毁善后处理（副作用代码的清理，例如：定时器、事件监听等）

7.getDerivedStateFromProps：较复杂的生命周期，写在子组件中 因为是静态的 所以前面要写static

8.shouldComponentUpdate

**路由钩子及应用场景**

beforeRouteEnter 使用场景：登录验证

beforeRouteUpdate 使用场景：路由更新的时候

beforeRouteLeave 使用场景：路由离开的时候

**$route和$router的区别**

$route用来获取路由信息的，它是路由信息的一个对象，里面包含路由的一些基本信息，包括name、meta、path、hash、query、params、fullPath、matched、redirectedFrom等。而$router主要是用来操作路由的，它是VueRouter的实例，包含了一些路由的跳转方法，钩子函数等

**高阶组件**

高阶组件是一个以组件为参数并返回一个新组件的函数。HOC运行重用代码、逻辑和引导抽象。最常见的可能是Redux的connect函数。除了简单分享工具库和简单的组合，HOC最好的方式是共享React组件之间的行为。如果你发现你在不同的地方写了大量代码来做同一件事时，就应该考虑将代码重构为可重用的HOC

**defer与async的区别是：**

defer要等到整个页面在内存中正常渲染结束（DOM结构完全生成，以及其他脚本执行完成），才会执行；

async一旦下载完，渲染引擎就会中断

**react中怎么实现类似vue中v-if的这种操作**

1.条件表达式渲染 （适用于两个组件二选一的渲染）

2.&& 操作符渲染 (适用于一个组件有无的渲染)

3.利用变量输出组件渲染 （适用于有多个组件多种条件下的渲染）

4.利用函数方法输出组件或者利用函数式组件进行渲染 （适用于多个子组件需要根据复杂的条件输出的情况）

**render在什么时候执行**

在组件实例化和存在期时执行 挂载完毕后

**怎么把一个一般组件定义成路由组件**

withRouter 可以加工一般组件，让一般组件具备路由组件所特有的API 它的返回值是一个新组建

**编程式路由导航有哪些**

通过JS 代码来实现页面跳转

history 是 React 路由提供的，用于获取浏览器历史记录的相关信息

push(path)：跳转到某个页面，参数path 表示要跳转的路径

go(n)： 前进或后退到某个页面，参数n 表示前进或后退页面数量（比如：-1 表示后退到上一页）

**路由传参**

**1.动态路由传参**

优点：

1、‘传参和接收都比较简单’

2、刷新页面参数不会丢失

缺点：

1、 ‘当复杂数据对象或数组需要传参时，这样做比较麻烦，需要通过json字符串的方式进行处理’

2、多个参数的传递不方便

3、参数会出现在url上，不够安全

动态路由一般都是用来传递某个唯一的值，比如详情或编辑的id

**2.search传参**

通过设置link的path属性，进行路由的传参，当点击link标签的时候，会在上方的url地址中显示传递的整个url

优点：1、‘传参和接收都比较简单’

2、刷新页面参数不会丢失

3、可以传递多个参数

缺点：

1、‘当复杂数据对象或数组需要传参时，这样做比较麻烦，需要通过json字符串的方式进行处理’

2、参数会出现在url上，不够安全

**3.隐式路由state传参**

所谓隐式路由传参，就是传参的信息不回暴露在url中，当点击该link标签，想要获取到传递的参数，就在对应的路由组件中，通过获取即可

优点：

1、‘传参和接收都比较简单’

2、可以传递多个参数

3、传递对象数组等复杂参数方便

4、不会暴露给用户，比较安全

缺点：

1、如果手动刷新当前路由时，数据参数有可能会丢失！！！’

在react中，最外层包裹了BrowserRouter时，不会丢失,但如果使用的时HashRouter，刷新当前页面时，会丢失state中的数据

**受控组件和非受控组件**

看到ref绑定是非受控组件

看到onChange就是受控组件，是要配合value来用

**react组件通信**

父传子：state定义，props接收

子传父：ref

跨组件通信：context 先定义

**什么是redux**

redux是一个专门用于做数据状态管理的JS库，众所周知，由于React是单向数据流，即父组件向子组件可以传递数据，但是反过来就不行。如果我们想要获得兄弟组件的数据，就需要先访问它们共同的父组件。给我们带来了很大的麻烦，于是redux应运而生。

ReactComponent（action），Store（存放数据的仓库），Reducers（处理各种业务逻辑的仓库管理员)

**redux累加器**

reduce() 方法接收一个函数作为累加器 ，数组中的每个值（从左到右）开始缩减，最终计算为一个值。

reduce() 可以作为一个高阶函数，用于函数的compose

**vue和react的区别和优缺点**

区别：vue是双向绑定的，采用template；react是单向的，采用jsx。Vue的优缺点：简单、快速、强大、对模块友好，但不支持IE8。React的优缺点：速度快、跨浏览器兼容、模块化；但学习曲线陡峭，需要深入的知识来构建应用程序。

**1.监听数据变化的实现原理不同**

**2.数据绑定**

vue：

vue是双向绑定， Vue.js 最核心的功能有两个，一是响应式的数据绑定系统，二是组件系统。所谓双向绑定，指的是vue实例中的data与其渲染的DOM元素的内容保持一致，无论谁被改变，另一方会相应的更新为相同的数据。这是通过设置属性访问器实现的。

react：

react是单向数据流；react中通过将state（Model层）与View (2.350, -0.210, -8.20%)层数据进行双向绑定达数据的实时更新变化，具体来说就是在View层直接写JS代码Model层中的数据拿过来渲染，一旦像表单操作、触发事件、ajax请求等触发数据变化，则进行双同步。

**3.组件通信的不同**

**4.框架本质不同**

**Vue.js的优缺点**

**优点：**

1. 简单：官方文档很清晰，比 Angular 简单易学。

2. 快速：异步批处理方式更新 DOM。

3. 组合：用解耦的、可复用的组件组合你的应用程序。

4. 紧凑：~18kb min+gzip，且无依赖。

5. 强大：表达式 & 无需声明依赖的可推导属性 (computed properties)。

6. 对模块友好：可以通过 NPM、Bower 或 Duo 安装，不强迫你所有的代码都遵循 Angular 的各种规定，使用场景更加灵活。

**缺点：**

1. 新生儿：Vue.js是一个新的项目，没有angular那么成熟。

2. 影响度不是很大：google了一下，有关于Vue.js多样性或者说丰富性少于其他一些有名的库。

3. 不支持IE8：

**React的优缺点**

**优点：**

1. 速度快：在UI渲染过程中，React通过在虚拟DOM中的微操作来实现对实际DOM的局部更新。

2. 跨浏览器兼容：虚拟DOM帮助我们解决了跨浏览器问题，它为我们提供了标准化的API，甚至在IE8中都是没问题的。

3. 模块化：为你程序编写独立的模块化UI组件，这样当某个或某些组件出现问题是，可以方便地进行隔离。

4. 单向数据流：Flux是一个用于在JavaScript应用中创建单向数据层的架构，它随着React视图库的开发而被Facebook概念化。

5. 同构、纯粹的javascript：因为搜索引擎的爬虫程序依赖的是服务端响应而不是JavaScript的执行，预渲染你的应用有助于搜索引擎优化。

**6.**兼容性好：比如使用RequireJS来加载和打包，而Browserify和Webpack适用于构建大型应用。它们使得那些艰难的任务不再让人望而生畏。

**缺点：**

React本身只是一个V而已，并不是一个完整的框架，所以如果是大型项目想要一套完整的框架的话，基本都需要加上ReactRouter和Flux才能写大型应用。

**react中使用axios发送请求的常用方法**

axios中使用GET请求时有两中方式：

一种是使用axios.get的方式进行

一种是使用axios(config { ... })的方式进行

**react的hook**

useState 设置和改变state，代替原来的state和setState

useEffect 代替原来的生命周期，componentDidMount，componentDidUpdate 和 componentWillUnmount 的合并版

useEffect()有两个参数，第一个参数是要执行的函数，第二个参数是一个依赖项数组数组(根据需求第二个参数可选是否填写)，

根据数组里的变量是否变化决定是否执行函数

useLayoutEffect 与 useEffect 作用相同，但它会同步调用 effect

useMemo 控制组件更新条件，可根据状态变化控制方法执行,优化传值

useCallback useMemo优化传值，usecallback优化传的方法，是否更新

useRef 跟以前的ref，一样，只是更简洁了

useContext 上下文爷孙及更深组件传值

useReducer 代替原来redux里的reducer,配合useContext一起使用

useDebugValue 在 React 开发者工具中显示自定义 hook 的标签，调试使用。

useImperativeHandle 可以让你在使用 ref 时自定义暴露给父组件的实例值。