1. **用过Hooks吗，介绍一下？（\*\*\*）---尽量介绍每个hooks，最起码介绍常用的hook**

答：Hoosk是react 16.8引入的特性，它允许你在不写class的情况下操作state和react的其他特性；Hooks只是多了一种写组件的方法，使编写一个组件更简单更方便，同时可以自定义hook吧公共的逻辑提取出来，让逻辑在多个组件之间共享

1. **高阶组件和hooks的区别**

答：两者都能用来进行逻辑复用。区别在于高阶组件为接收组件，对其进行包装，而 hooks 是以函数调用的形式共享数据。

1. **setState,是同步还是异步？**

答：setState 只在合成事件和钩子函数中是“异步”的，在原生事件和 setTimeout 中都是同步的。

1. **hooks生命周期怎么去模拟?**

答：使用useEffect来模拟，当第二个参数为空数组时，相当于componentDidMount，当第二个参数没有或者为变量时，相当于componentDidUpdate,当第一个参数返回一个函数时，这个函数相当于componentUnMount;

1. **调用 setState 之后发生了什么？**

答：在代码中调用 setState 函数之后，React 会将传入的参数对象与组件当前的状态合并，然后触发所谓的调和过程（Reconciliation）。经过调和过程，React 会以相对高效的方式根据新的状态构建 React 元素树并且着手重新渲染整个 UI 界面。在 React 得到元素树之后，React 会自动计算出新的树与老树的节点差异，然后根据差异对界面进行最小化重渲染。在差异计算算法中，React 能够相对精确地知道哪些位置发生了改变以及应该如何改变，这就保证了按需更新，而不是全部重新渲染。

1. **react：列表页筛选完，跳转到详情页后，点返回，想看到之前的筛选结果，怎么做。（相当于实现vue的keep-alive）?**

答：（1）通过插件或封装高阶组件来实现（2）还可以通过react 16.3新推出的createPortal API实现，createPortal API吧组件都挂载到app外面，然后吧dom移动到显示的位置来实现缓存，这样做可以很好的与react router结合，也可以在切换页面是使用动画

1. **调用setState()时触发了哪些react的生命周期？**

答：shouldComponentUpdate()当shouldComponentUpdate函数被调用的时候，this.state没有得到更新

componentWillUpdate()当componentWillUpdate函数被调用的时候，this.state依然没有得到更新。

render()直到render函数被调用的时候，this.state才得到更新,或者，当shouldComponentUpdate函数返回false，这时候更新过程就被中断了，render函数也不会被调用了，这时候React不会放弃掉对this.state的更新的，所以虽然不调用render，依然会更新this.state。

componentDidUpdate()当setState执行成功后在该生命周期中可以拿到最新的数据

1. **setState两次，render执行几次？**

答：连续调用了两次this.setState，但是只会引发一次更新生命周期，不是两次，因为React会将多个this.setState产生的修改放在一个队列里，缓一缓，攒在一起，觉得差不多了再引发一次更新过程。这样做的好处是，不用每次setState就去触发一次render()，这样太消耗性能。

1. **redux数据流向是怎样的？**

答：

1. view组件通过click触发事件
2. Dispatch一个action
3. 通过store把当前状态state和action传给reducer
4. Reducer返回一个新的状态存储在store中
5. Store又把新的state传递给view组件触发组件更新
6. **redux有哪些弊端？怎么优化？**

答：简单的异步交互使用thunk，复杂的异步交互建议下载处理异步的中间件，过于复杂的异步交互thunk处理起来比较复杂，这就是弊端 优化就是下载react专门的处理异步的中间件

1. **介绍redux（\*\*\*）**

答：redux 就相当于 vuex ，就是数据状态管理模式，当我们要做的项目中有很多组件需要共享数据时，这时候就可以用 redux 搭建。工作流程：调用dispatch发起action=>进入reducer，返回新的state=>触发subscribe监听函数=>更新视图

1. **Redux异步处理中间件？**

答：redux的异步处理中间件是redux-thunk。redux如果需要使用异步操作，那么就需要使用中间件，而redux自带着一些中间件的用法，在store中可以在createStore中传入我们的中间件，但在之前需要先引入对应插件applyMiddleware，以及下载执行异步操作的redux-thunk。

1. **介绍一下redux-saga**

答：redux-saga 是一个用于管理应用程序 Side Effect（副作用，例如异步获取数据，访问浏览器缓存等）的 library，它的目标是让副作用管理更容易，执行更高效，测试更简单，在处理故障时更容易。

可以想像为，一个 saga 就像是应用程序中一个单独的线程，它独自负责处理副作用。 redux-saga 是一个 redux 中间件，意味着这个线程可以通过正常的 redux action 从主应用程序启动，暂停和取消，它能访问完整的 redux state，也可以 dispatch redux action。

redux-saga 使用了 ES6 的 Generator 功能，让异步的流程更易于读取，写入和测试。

1. **redux-thunk与redux-sage的原理**

答redux-thunk的原理：我们的action都是通过返回一个对象，接着执行对应的操作，但是我们添加了redux-thunk中间件后，action也可以返回一个函数，我们的异步操作就在这里面来执行 ,然后返回一个函数，在函数中发起请求异步操作，之后的store调用发起这个action时就会执行里面的函数，发起请求得到对应数据，得到数据后我们就还需要再次发起action来将数据添加到state中，这样我们的这个异步操作就完成了。

Redux-sage的原理：redux-saga相当于在Redux原有数据流中多了一层，对Action进行监听，捕获到监听的Action后可以派生一个新的任务对state进行维护。

1. **Redux-thunk 为什么可以解决异步**

答：如果使用了thunk中间件，当我们dispatch的是一个函数，thunk会拦截将dispatch作为参数传进去，如果dispatch的是一个对象，则不进行任何操作

1. **React 中 key 的作用是什么 ----顺便介绍一下前后虚拟dom是怎么对比的**

答：Key用于识别唯一的 Virtual DOM 元素及其驱动 UI 的相应数。 是 React 用于追踪哪些列表中元素被修改、被添加或者被移除的辅助标识。

1. **react key有什么作用 , update这个生命周期调用setState会怎么样**

答：在render函数执行的时候，新旧两个虚拟DOM会进行对比，如果两个元素有不同的key，那么在前后两次渲染中就会被认为是不同的元素，这时候旧的那个元素会被unmount，新的元素会被mount，以此来提高性能；

不能在update这个生命周期调用setState,会发生死循环（无限渲染）

1. **有哪些动作可以触发render？**

答：通过setState；this.forceUpdate()；通过状态管理，如mobx，redux等

1. **写过高阶组件吗？----- 最好列举出来应用场景**

答：高阶组件只是一个包装了另外一个 React 组件的 React 组件；形式通常为一个纯函数；一般用作与代码复用，逻辑抽象，渲染劫持，state抽象和更改，props更改，它的应用场景如下：1. 操作props 2. 抽离state 3. 通过 ref 访问到组件实例 4. 反向继承 5. 渲染劫持 6. 条件渲染

1. **react diff 原理?**

答：1.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 合并操作，调用 component 的 setState 方法的时候, React 将其标记, 到每一个事件循环结束, React 检查所有标记的 component 重新绘制. 选择性子树渲染。

1. **路由监听**

答:在react中，所有的路由组件props身上都会有三个属性，而这三个属性记录了路径的变化情况，路径变化，属性就会变化，属性变化，就会触发componentWillReciveProps这个钩子，我们可以利用这个钩子做一些处理，而要监听路由的变化，就要监听整个项目的路由，所以我们要找到最大的组件 最大的组件是app，但是他是函数式组件，我们不能动他，所以我们要找到除了app以下的我们自己定义的最大的组件,一般的我们回去监听layout组件,使用reaact-router-dom库中withRoute高阶组件,他是一个函数，接受一个组件作为参数,使用他可以在layout中获取到peops属性,可以看到history属性等,

1. **生命周期是用来干嘛的，不同生命周期分别能做哪些事？**

答:生命周期的作用：React实例从创建开始到最终销毁的过程,在每个周期中我们加入需要的处理,通过生命周期来覆盖修改state与props的值

不同生命周期的作用：

* 1. constructor()
     1. (1)constructor()中完成了React数据的初始化，它接受两个参数：props和context，当想在函数内部使用这两个参数时，需使用super()传入这两个参数。
  2. componentWillMount()
     1. 一般用的比较少，它更多的是在服务端渲染时使用。它代表的过程是组件已经经历了constructor()初始化数据后，但是还未渲染DOM时。
  3. componentDidMount()
     1. (1)组件第一次渲染完成，此时dom节点已经生成，可以在这里调用ajax请求，返回数据setState后组件会重新渲染
  4. componentWillUnmount ()
     1. 在此处完成组件的卸载和数据的销毁。
     2. clear你在组建中所有的setTimeout,setInterval
     3. 移除所有组建中的监听 removeEventListener
  5. 更新过程：componentWillReceiveProps (nextProps)
     1. 在接受父组件改变后的props需要重新渲染组件时用到的比较多
     2. 接受一个参数nextProps
     3. 通过对比nextProps和this.props，将nextProps的state为当前组件的state，从而重新渲染组件
  6. shouldComponentUpdate(nextProps,nextState)
     1. 主要用于性能优化(部分更新)
     2. 唯一用于控制组件重新渲染的生命周期，由于在react中，setState以后，state发生变化，组件会进入重新渲染的流程，在这里return false可以阻止组件的更新
  7. componentWillUpdate (nextProps,nextState)
     1. shouldComponentUpdate返回true以后，组件进入重新渲染的流程，进入componentWillUpdate,这里同样可以拿到nextProps和nextState。
  8. componentDidUpdate(prevProps,prevState)
     1. 组件更新完毕后，react只会在第一次初始化成功会进入componentDidmount,之后每次重新渲染后都会进入这个生命周期，这里可以拿到prevProps和prevState，即更新前的props和state
  9. render()
     1. render函数会插入jsx生成的dom结构，react会生成一份虚拟dom树，在每一次组件更新时，在此react会通过其diff算法比较更新前后的新旧DOM树，比较以后，找到最小的有差异的DOM节点，并重新渲染。
  10. getDerivedStateFromProps(nextProps, prevState)
      1. 代替componentWillReceiveProps()。
      2. 老版本中的componentWillReceiveProps()方法判断前后两个 props 是否相同，如果不同再将新的 props 更新到相应的 state 上去。这样做一来会破坏 state 数据的单一数据源，导致组件状态变得不可预测，另一方面也会增加组件的重绘次数。
  11. getSnapshotBeforeUpdate(prevProps, prevState)
      1. 代替componentWillUpdate。
      2. 常见的 componentWillUpdate 的用例是在组件更新前，读取当前某个 DOM 元素的状态，并在 componentDidUpdate 中进行相应的处理。

1. **状态管理除了redux还了解哪些？**

答：Mobx: 它的数据流简明清晰，并且也是单向数据流；Mobx组成部分: Actions、State、Computed Values、Reactionsa

1. **React组件间的通讯？**

答：react组件通讯主要有父子，子父，跨组件通讯，

父子：父组件通过向子组件传递 props，子组件通过props进行接收。

子父：父组件将一个函数作为 props 传递给子组件，子组件调用该回调函数，便可以向父组件通信。

跨组件通讯：使用context对象进行传参。

1. **说说react或者vue的组件通信对比（\*\*\*）**

答：React：

* + 1. 父组件向子组件通信：父组件通过props向子组件传递信息
    2. 子组件向父组件通信
       1. 利用回调函数
       2. 利用自定义事件机制
    3. 跨级组件通信
       1. 层层组件传递props
       2. 使用contex
    4. 没有嵌套关系组件之间的通信: redux

Vue:

1. 父组件与子组件传值
   1. 父组件传给子组件：子组件通过props方法接受数据;
   2. 子组件传给父组件：$emit 方法传递参数
2. 非父子组件间的数据传递: (1)兄弟组件传值eventBus，就是创建一个事件中心，相当于中转站，可以用它来传递事件和接收事件。$emit 传值，$on() 接收值。项目比较小时，用这个比较合适
3. 整个状态存储：vuex
4. **react合成事件带来的优势**

答：实现了一套高效的事件注册，存储，分发和重用逻辑，在DOM事件体系基础上做了很大改进，减少了内存消耗，简化了事件逻辑，并最大化的解决了IE等浏览器的不兼容问题。

1. **React与vue的差异（\*\*\*）**

答：1. react整体是函数式的思想，为单向数据流，vue为思想是响应式的，也就是基于是数据可变的

2.react使用jsx将html，css，js放在一起写，而vue是组合在一起，用各自的处理方式。

3．react是类式的写法，api很少，而vue是声明式的写法，通过传入各种options，api和参数都很多

4. react可以通过高阶组件（Higher Order Components--HOC）来扩展，而vue需要通过mixins来扩展

5.react大部分代码需要自己封装，自由度跟高，vue各项都已封装完成，自由度低。

a.组件的创建不一样，b.react使用class和函数创建组件（只有局部）；c. 而vue则使用vue实例创建components方法和components属性（有局部和全局）；d. 在react中没有指令，计算属性，watch监听，computed，这些内容；e. 在react组件中，只有几个概念，生命周期，state，props，合成事件，f. 在vue中可以试用v-model完成控件和属性值得同步（双向数据的绑定），而在react中并没与指令的概念，所以使用受控组件代替； g. 在vue中使用get，set拦截器，处理数据的绑定，而react则使用的是setState手动触发

1. **Redux-Sage的使用**

答：saga需要一个全局监听器（watcher saga），用于监听组件发出的action，将监听到的action转发给对应的接收器（worker saga），再由接收器执行具体任务，任务执行完后，再发出另一个action交由reducer修改state，这里必须注意：watcher saga监听的action和对应worker saga中发出的action不能是同一个，否则造成死循环  
在saga中，全局监听器和接收器都使用Generator函数和saga自身的一些辅助函数实现对整个流程的管控

1. **Redux的中间件**

答：react中的常用中间件有redux-thunk,redux-sage,react- persist。

1. **React在开发h5与pc存在哪些区别？**

答：（1） PC考虑的是浏览器的兼容性，而移动端开发考虑的更多的是手机兼容性，因为目前不管是android手机还是ios手机，一般浏览器使用的都是webkit内核，所以说做移动端开发，更多考虑的应该是手机分辨率的适配，和不同操作系统的略微差异化。（2） 在部分事件的处理上，移动端多出来的事件是触屏事件，而缺少的是hover事件。 另外包括移动端弹出的手机键盘的处理，这样的问题在PC端都是遇不到的。 （3）在布局上，移动端开发一般是要做到布局自适应的，我使用的一直是rem布局，感觉很好。（4） 在动画处理上，PC端由于要考虑IE的兼容性，所以通常使用JS做动画的通用性会更好一些，但是CSS3做了很大的牺牲， 而在手机端，如果要做一些动画、特效等，第一选择肯定是CSS3， 既简单、效率又高。（5）  微信的一些接口组好能去实现一遍，熟悉一下肯定是有好处的，比如通过微信分享文章，title、description、icon等图标的配置，这些还是要会的。（6）百度地图的一些API接口，也得去实现一下，这些对于移动端来说，LBS是一个非常重要的特性，所以地图这块肯定是要了解的，在加上百度地图这块已经是一个比较成熟的平台了，所以学起来也比较容易。（7） CSS3的动画一定要比较熟练，这在移动端用的还是比较多的。（8）一般pc端用jquery，移动端用zepto，因为移动端的流量还是比较重要的， 所以引入的资源或者插件，能小则小，一个30k的资源和一个80k的资源，在移动端的差别还是挺大的。

1. **介绍一下React的高阶组件**

答：高阶组件 不属于 React 的 API，它是一种实现模式，本质上是一个函数，接受一个或多个 React 组件作为参数，返回一个全新的 React 组件，而不是改造现有的组件，这样的组件被称为高阶组件。开发过程中，有的功能需要在多个组件类复用时，这时可以创建一个 高阶组件。

1. **React常用构造函数**

答：constructor 函数时组件最先执行的函数，constructor()：子类继承父类时的构造方法，主要时用以定义this.属性 在react中一些默认的数据可以直接在此处定义

1. **哪些方法会触发react重新渲染？**

答：（1）state的改变（只要使用了this.setState({})不管有没有改变值都会重新渲染）（2）props 的更新，（3） this.forceUpdate() 强制更新

1. **React渲染流程**

答：1.babel会将jsx转化为React.createElement的表达式

2.React.createElement render(){ return element } 在render函数调用时，生成一个element(虚拟节点)

3.众多element组成虚拟dom

4.ReactDOMComponent将众多element转化为真实节点

5.当数据发生改变时，render()函数会被再次调用，构造出新的虚拟dom，跟原来的虚拟dom用Diff算法进行比较，找到需要更新的地方批量改动，再补到真实的DOM上

1. **讲一下跨域,请求后台的跨域 -----把常见的跨域解决方式给说一下**

答：因为浏览器收到同源策略的限制，当前域名的js只能读取同域下的窗口属性。什么叫做同源策略？就是不同的域名, 不同端口, 不同的协议不允许共享资源的，保障浏览器安全。同源策略是针对浏览器设置的门槛。如果绕过浏览就能实现跨域，所以说早期的跨域都是打着安全路数的擦边球，都可以认为是 hack 处理

1. **图片的预加载和懒加载？----最好说一下大致怎么实现预加载和懒加载的**

答：**图片预加载：**就是在网页全部加载之前，提前加载图片，当用户需要查看时可直接从本地缓存中渲染，以提供给用户更好的体验，减少等待的时间。

**图片懒加载（缓载）：**延迟加载图片或符合某些条件时才加载某些图片。

1. **讲一下 react的虚拟dom，生命周期里面什么时候应该调用接口，清除定时器用什么，不清除的话有什么后果**

答：虚拟DOM是为了提高DOM渲染性能的, 把一个真实的DOM树转换成js对象树(createElment),当我们改变页面的时候,就会产生一个目标的对象树,两棵对象树进行比较,在内存中进行比较,比较出的不同点进行渲染。

在componentdidmount中调用接口，使用clearInterval清除定时器，

不清除定时器的后果：1.影响性能，

2.不确定在离开页面后定时器不不会调用，防止不必要的报错

1. **react里面的 ref的作用是什么，react路由用过哪个讲一下，redux用过哪些中间件？ yield作用是什么？**

答：ref的作用是获取真实dom和组件实例；history和hash都用过。History路由的原理是根据js的history对象下pushState方法改变浏览器地址不会刷新页面做一个视图切换的效果，hash路由根据浏览器地址hash值的变化，触发onhashchange事件实现的视图切换效果。History更美观，hash路由兼容性更好；用过redux-thunk和redux-saga，yield作用是代理yield表达式，将需要函数本身产生的值委托出去，yiled表达式的返回值就是其后面函数或可迭代对象的返回值。

1. **说一下v-if和v-for有什么注意事项**

答：v-for比v-if的优先级要高，同时使用每一次都需要遍历一次数据，极大影响性能。解决办法：必要情况下可以将运算更改为计算属性；还可以将v-if放到父级运行

1. **假如有一个异步的方法API 你在useEffect怎么去调它？**

答：第二个参数设定为空数组，初始渲染执行一次

1. **什么是跨域，跨域的解决办法？**

答：服务器之间的协议不同、域名不同、端口不同就是跨域。

解决办法：

1. Jsonp跨域
2. 反向代理
3. CORS跨域资源共享，即后端设置请求头允许跨域请求
4. **Saga 是怎么解决异步的？**

答：使用saga模式和es6 generators进行实现。Saga模式将业务逻辑分为对个独立的事务，如果业务过称遇到错误并无法进行下去，就可以执行修正事务来修正完成的步骤，保证最终的统一性。而每一个saga就是一个generators，可以再系统内无限期运行。当特定action执行时，就会被唤醒。

1. **Jsx是什么？**

答：为了解决react代码的可读性及便利性，有webpack打包解析的一种可以在react组件内部构建标签的xml语法。

1. **一般使用那个生命周期请求API？**

答：componentDidMount 初始渲染后执行的生命周期函数

1. **setState() 同步？还是异步**

答：异步执行，特殊情况下是同步的。比如，在原生事件和定时器里运行

1. **纯函数**

答：首先这个函数本身的执行结果之和参数相关，不依赖外部变量。其次运行过程没有副作用

1. **React和vue优缺点**

答：

react优点：

1. react性能更强大，通过在虚拟dom中的微操作实现局部更新
2. react兼容性更好,标准化API，解决跨浏览器问题
3. 单向数据流，数据流向更加清晰
4. 纯粹的javascript有助于搜索引擎的优化

缺点 ：react本身其实并不是一个完整的框架，如果是大型项目的话，都需要react-router、redux等库的支持

Vue优点：

1. 简单快速，双向数据绑定和指令使的项目搭建的非常快
2. 易于使用：使用html标准模板，更轻松使用和修改现有应用
3. 顺畅的集成系统：无论是单页应用程序还是复杂的Web界面，Vue.js都可以更平滑地集成更小的部件，而不会对整个 系统产生任何影响。

缺点：因为实现机制使用了Object.defineProperty，不支持ie8.由于vue响应机制的问题，当state较多时，监听的属性多了，会导致性能下降，因此比较大的项目不太适合使用

1. **如果数据量较大，有什么好的展示方法**
2. 数据分页显示
3. 数据懒加载（上拉加载更多）
4. 虚拟列表
5. **宏任务和微任务**

每次执行栈执行的代码就是一个宏任务，而当前task执行结束后立即执行的任务就是微任务。微任务因为无需等待渲染，响应速度相比宏任务更快，。

运行机制：先运行宏任务=>遇到微任务，加入到微任务队列=>执行微任务=>开始新的宏任务...

宏任务：

1. script（整体代码）
2. setTimeout
3. setInterval
4. I/O
5. UI交互事件

微任务：

1. Promise
2. Object.observe
3. Process.nextTick