1. React渲染DOM的整体流程。从初始加载和后续更新的两个角度进行阐述。
2. 什么是JSX。JSX的特点。
3. 函数组件的特点及使用场景。
4. 类组件的特点。
5. props和state的区别。
6. 类组件中的数据的声明方式及各自的使用场景。
7. state中的数据能够不使用this.setState修改？为什么。
8. setState是同步还是异步的。
9. setState中如何获取最新修改的值。
10. forceUpdate的作用。
11. react条件渲染有几种方式。
12. 渲染列表时key的作用。能否使用数组索引作为key值。
13. 受控组件和非受控组件的区别。使用场景。
14. react中的传值方式有哪些。
15. 发布订阅模式的原理。在react中如何运用。
16. 虚拟DOM的优点是什么？diff算法的优势是什么？什么是diff算法。
17. shouldComponentUpdate的作用。在项目中如何使用。
18. react对于引用类型的数据如何进行渲染优化。
19. Immutable是什么。在react中的作用。
20. 组件树之间如何传值。具体代码实现思路。
21. Fragments的作用。
22. PureComponent类的作用。
23. 常用的生命周期钩子有哪些，以及各自的使用场景是什么。
24. React-router路由传值的方法有哪些。组件如何获取这些传递的参数。
25. 组件懒加载如何使用？优势是什么。
26. React中如何配置环境变量。
27. React中如何对路由进行登录权限控制。
28. React高阶组件的概念。使用场景是什么。
29. 路由组件和普通子组件的使用场景区别。
30. HashRouter和BrowserRouter的区别。
31. Redux的工作流程。
32. Redux如何处理异步Action。
33. 组件中如何获取Redux中的数据以及如何修改数据。
34. React中如何解决类名作用域冲突的问题。
35. 列举一些Less和Sass语法的不同之处。
36. React如何解决跨域问题。
37. Git分支的使用。Git冲突有哪几种情况？如何解决。

**Hooks题目**

1. useState相关知识点
2. useState返回的更新数据函数被调用后，不会立刻拿到最新的结果。

**const** DemoState **=** **(**props**)** **=>** **{**

/\* number为此时state读取值 ，setNumber为派发更新的函数 \*/

**let** **[**number**,** setNumber**]** **=** useState**(**0**)** /\* 0为初始值 \*/

**return** **(<**div**>**

**<**span**>{** number **}</**span**>**

**<button** onClick**={** **()=>** **{**

setNumber**(**number**+**1**)**

console**.**log**(**number**)** /\* 这里的number是不能够即使改变的 \*/

**}** **}** **></button>**

**</**div**>)**

**}**

1. 在函数组件中一次执行上下文中，state的值是固定不变的。

**function** Index**(){**

**const** **[** number**,** setNumber **]** **=** React**.**useState**(**0**)**

**const** handleClick **=** **()** **=>** **setInterval(()=>{**

// 此时 number 一直都是 0

setNumber**(**number **+** 1 **)**

**},**1000**)**

**return** **<button** onClick**={** handleClick **}** **>** 点击 **{** number **}</button>**

**}**

**function** Index**(){**

**const** **[** number**,** setNumber **]** **=** React**.**useState**(**0**)**

**const** handleClick **=** **()** **=>** **setInterval(()=>{**

// 此时 number 一直都是 0

setNumber**(**number **+** 1 **)**

**},**1000**)**

**return** **<button** onClick**={** handleClick **}** **>** 点击 **{** number **}</button>**

**}**

1. 当使用useState返回的函数修改引用数据类型时，如何保证页面渲染？
2. 当使用useState返回的函数修改基本数据类型时，如果值相同，页面是否会重新渲染？
3. 如何获取最新修改后的数据。
4. useEffect如何保证首次加载组件时执行，后续加载组件不执行回调函数。
5. useEffect如何实现组件数据更新时的调用。
6. useEffect如何实现组件卸载时的清理工作。
7. 如果useEffect内部使用了外部的函数或者数据state，但是没有在依赖项里做出设置，会出现什么情况。如何解决。
8. 尽可能将函数声明放在useEffect的内部，这样就不会构成闭包，从而解决控制台的警告，此时依赖项是无需操作的。
9. 如果一些公共的函数确实需要在其他的函数中使用，比如公共的请求，公共的工具性函数，此时可以将函数声明继续放在useEffect的内部，同时使用useRef的current属性保存这个函数，那么其它函数中就可以使用current来获取这个公共的内容进行调用。
10. 如果公共函数确实需要放在useEffect外部，那此时在useEffect内部就会形成闭包环境，控制台会发出警告，让去设置依赖项，但是不能设置依赖项，可能会造成外部函数死循环调用，可以使用useCallback对公共函数进行缓存，将缓存后的函数放在依赖项中，从而解决警告，又可以避免外部函数形成死循环调用。
11. 说一下useContext的作用，和工作方式。
12. Hooks中的组件渲染优化相关
    1. React.memo的特性
       1. 它类似于类组件中的PureComponent，可以对父组件传递过来的prop数据是否发生改变进行浅层比较，如果prop发生了改变就渲染子组件，prop没有发生改变，子组件就不会渲染。
       2. 它只对比prop，不会对当前组件的state进行对比。
    2. useCallback的特性
       1. 它会对函数进行缓存，返回缓存后的函数。
       2. 当父组件的函数需要传递给子组件进行子组件向父组件传值时，由于传递的是普通函数，当父组件渲染时，会重新生成一个新函数，此时子组件的React.memo就不起作用了，需要使用useCallback将缓存后的函数传递过去，React.memo才能发挥作用。
       3. 当组件内的某个函数必须要作为依赖项时，需使用useCallback进行缓存。
13. useMemo的作用。
14. useRef的作用。
    1. 用于操作DOM。
    2. 充当全局的公共变量，保存任意数据至current中。且useRef每次返回的都是上一次修改的最新数据，不管组件如何渲染，current里面的数据都不会重新初始化。
15. useReducer的作用。
16. 是否使用过useReducer搭配useContext来自己封装一个类似于Redux的公共状态管理的工具。
17. useImperative的作用。
18. react路由中都提供了哪些hooks的用法。各自的作用是什么。如何使用。
19. redux中提供了哪些hooks的用法。各自的作用是什么。如何使用。

扩展题：

1. [面试官：为什么说Vue 的响应式更新比 React 快？（原理深度解析） | 码农家园 (codenong.com)](https://www.codenong.com/j5e854a5e51882573954/)
2. [前端页面渲染100000条数据如何进行优化？ - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/505873741)

分页—每页多少条数据，分多少页。

1. [LocalStorage 扩容\_东方的csdn的博客-CSDN博客\_localstorage扩容](https://blog.csdn.net/qq_22930381/article/details/104830957)
2. [localStorage 存满了怎么办？\_weixin\_30905133的博客-CSDN博客](https://blog.csdn.net/weixin_30905133/article/details/99648550?spm=1001.2101.3001.6661.1&utm_medium=distribute.pc_relevant_t0.none-task-blog-2~default~CTRLIST~default-1-99648550-blog-120114688.pc_relevant_default&depth_1-utm_source=distribute.pc_relevant_t0.none-task-blog-2~default~CTRLIST~default-1-99648550-blog-120114688.pc_relevant_default&utm_relevant_index=1)

如果 localStorage 存满了，再往里存东西，或者要存的东西超过了剩余容量，结果存不进去会报错。

一、如何得到localStorage已使用容量

(function(){

if(!window.localStorage) {

console.log('浏览器不支持localStorage');

}

var size = 0;

for(item in window.localStorage) {

if(window.localStorage.hasOwnProperty(item)) {

size += window.localStorage.getItem(item).length;

}

}

console.log('当前localStorage已使用容量为' + (size / 1024).toFixed(2) + 'KB');

})()

1. [前端大容量缓存方案-IndexedDB - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/104536473)
2. [前端性能优化小结（面试干货） - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/115047733)
3. [双token校验机制\_Markov Zheng的博客-CSDN博客\_双token机制](https://blog.csdn.net/marko_zheng/article/details/121339402)
4. [react中使用pubsub-js实现消息的发布订阅 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/89015701)
5. [使用 customize-cra 修改 webpack 配置\_杏子\_1024的博客-CSDN博客](https://blog.csdn.net/weixin_44135121/article/details/109850564)
6. (Redux Toolkit)[入门 Redux | Redux 中文官网](http://cn.redux.js.org/introduction/getting-started#redux-toolkit-%E7%A4%BA%E4%BE%8B)