如何使用react脚手架搭建一个项目

- npx create-react-app app-name

react中的组件传参

- 父传子

```
#父传子将父组件中的属性传入子组件，子组件可以用props直接接收并使用。
```

- 子传父

```
#子传父是子组件调用父组件中传来的方法来改变父组件中的值。
```

- 非相关组件传参

```
#使用context上下文，通过使用context.Provider数据提供者来实现全局的数据共享。
```

在react中class定义的组件生命周期钩子函数?

可以大致分为初始化阶段、运行中阶段和销毁阶段

初始化阶段

- constructor( )
- //设置组件的初始化状态
- componentWillMount( )
- //组件在挂载之前触发，这时可以开启定时器或者向服务器发送请求
- render( )
- //组件的渲染
- componentDidMount( )
- //组件已经渲染完毕，此时可以执行DOM的相关操作

运行中阶段

- componentWillReceiveProps( )
- //组件接收到新的属性时触发
- shouldComponentUpdate( )
- //当组件接收到新属性，或者组件的状态发生改变时触发。组件首次渲染时并不会触发

```
#此处可以对react进行优化处理，决定后续生命周期函数是否需要继续执行
shouldComponentUpdate(newProps, newState) {
    if (newProps.number < 5) return true;
    return false
}
//该钩子函数可以接收到两个参数，新的属性和状态，返回true/false来控制组件是否需要更新。
```

- componentWillUpdate( )
- //组件更新之前触发

- componentDidUpdate( )
- //组件已经更新完毕，此时页面中产生了新的DOM元素，可以进行DOM操作

销毁阶段

- componentWillUnMuont( )
- //组件销毁之前触发，此时可以做一些清理操作，例如清除定时器，或者一些事件。

react中class定义的组件和function定义的组件的区别?

- function定义的组件没有this指向的问题
- class定义的组件有自己的局部状态（this.state）和自己的生命周期函数，function定义的组件是无状态组件，但是在16.8之后可以用hooks（useEffect）来模拟组件的局部状态和生命周期。
- 官方建议使用function来定义组件，写法简单，并且便于理解。
- 我目前一直在使用function来定义组件

react中hooks是如何模拟组件的生命周期的?

componentDidMount

```
function Example() {
  useEffect(() => console.log('mounted'), []);
  return null;
}
```

useEffect 拥有两个参数，第一个参数作为回调函数会在浏览器布局和绘制完成后调用，因此它不会阻碍浏览器的渲染进程。
第二个参数是一个数组

- 当数组存在并有值时，如果数组中的任何值发生更改，则每次渲染后都会触发回调。
- 当它不存在时，每次渲染后都会触发回调。
- 当它是一个空列表时，回调只会被触发一次，类似于 componentDidMount。

componentDidUpdate

```
useEffect(() => console.log('count updated'),[count]);
```

componentWillUnmount

```
useEffect(() => {
  return () => {
    console.log('will unmount');
  }
}, []);
```

- 当在 useEffect 的回调函数中返回一个函数时，这个函数会在组件卸载前被调用。我们可以在这里面清除定时器或事件监听器。

react和vue的区别和共同点?

共同点:

- 都使用虚拟dom。
- 都提供了响应式和组件化的视图组件。

- 都有全局状态管理的库。(vue-router、vuex、react-router、redux等等)

不同点：

- 在性能方面，当组件的状态发生变化时，React的机制会触发整个组件树的重新渲染（shouldComponentUpdate()),Vue提供了优化的重新渲染，其中系统在渲染过程中跟踪依赖关系并相应地工作。重新渲染Vue是最显着的特征，使其成为全世界开发人员广泛接受的框架。
- React
  react非常的灵活，并且它的库是非常丰富的，后盾是facebook，在react中数据是单向流动的，react在setState之后会重新走渲染的流程，如果shouldComponentUpdate返回的是true，就继续渲染，如果返回了false，就不会重新渲染，开发大型应用程序更好。
- Vue
  vue性能更棒，占用空间更少，适合开发单页面应用程序，在vue中而Vue的思想是响应式的，也就是基于是数据可变的，通过对每一个属性建立Watcher来监听，当属性变化的时候，响应式的更新对应的虚拟dom，并且易于学习和上手，文档也比较友好。

react中的高阶组件有没有了解过?

- 高阶组件就是一个没有副作用的纯函数。
- 高阶组件是把一个组件当做参数，返回值为新组件的函数。
- 高阶组件实际上就是高阶函数，如map、forEach
- 常用的高阶组件：withRouter（路由中的），connect（reac-redux中的）

redux是什么?

- redux是一个全局状态管理插件，用来操作数据的，它可以结合任何一个js库来使用（vue，安哥拉）等，react是一个针对视图层的library，一般结合react来使用。
- redux是单向数据流，state用来存储数据，所有的数据改变都在reducer中进行，redux中还有一个action，用来组织数据。
- 单向数据流：数据是单向流动的，view通过dispatch来派发一个action改变数据，数据改变之后，页面重新渲染。

redux分为三个部分：

- state用来存储数据
- reducer用来处理和改变数据，每一个reducer都是一个function，返回一个新的对象作为数据，目的是不进行原始数据的对比，使性能更高。
- action用来组织数据，传递给reducer来进行数据的改变操作，每一个action都有一个属性叫type。表示数据以什么方式进行改变。

react中的错误边界?

- React16.X中引入了错误边界（Error Boundaries）概念。
- 它可以捕获它的子组件中产生的错误，类似于try-catch，只不过是以react组件的形式来实现的。
- 有了错误边界，即使某个组件的结果有错误，整个React程序挂载也不会被挂掉。只有出错的那个组件会显示一个后备界面，而整个程序仍然完全正常运行。
- 这里的componentDidCatch()函数使用方法和JavaScript中的catch {}代码块差不多，但只能用于组件。只有类组件才可以成为错误边界。
- 在componentDidCatch()函数内部我们把hasError状态设置为true。然后在渲染方法中检查那个状态。如果出错状态是真，就渲染后备界面；如果是false就把想渲染的React组件界面当作子组件界面渲染出来。

尽管如此，以下错误Error Boundaries依旧无法捕获：

- 事件错误

- Error Boundaries本身的错误
- 异步代码

react中react-router和react-router-dom的有什么区别?

api方面

- React-router：
  提供了路由的核心api。如Router、Route、Switch等，但没有提供有关dom操作进行路由跳转的api；
- React-router-dom：
  提供了BrowserRouter、Route、Link等api，可以通过dom操作触发事件控制路由。
- Link组件，会渲染一个a标签；BrowserRouter和HashRouter组件，前者使用pushState和popState事件构建路由，后者使用 hash 和 hashchange 事件构建路由。

使用区别

- react-router-dom在react-router的基础上扩展了可操作dom的api。
  Swtich 和 Route 都是从react-router中导入了相应的组件并重新导出，没做什么特殊处理。
- react-router-dom中package.json依赖中存在对react-router的依赖，故此，不需要额外安装react-router。

如何使用redux（使用流程）越详细越好?

- 在脚手架中安装react-redux
- 使用createStore去创建一个全局的store，用来保存所有的state，createStore接收一个reducer作为参数，你可以使用combineReducers传入多个reducer。
- 在reducer中，接收两个参数，第一个参数表示数据的初始状态，第二个参数表示action，并且reducer会返回一个新的对象作为数据，这样的话可以不进行原始对象的比较，性能会提高。
- 想要改变数据的话，就是view通过dispatch去派发一个action去执行相应的reducer，并且在store中进行更新，store改变的话，view就会重新渲染。
- 我们可能要使用react-redux中的connect和Provider方法 去关联我们的数据。Provider通过context上下文向子组件提供store,connect把redux中的数据和组件中的props做关联，这里用到的方法是mapStateToProps把store中的数据去映射到组件的props中，这样在组件中就可以通过props去访问到redux中的数据。
- 如果需要发送异步请求的话，还需要react-thunk插件，需要在creaceStore中做一个配置。

react中的key值有什么作用?

- key属性的使用，则涉及到diff算法中同级节点的对比策略，当我们指定key值时，key值会作为当前组件的id，diff算法会根据这个id来进行匹配。如果遍历新的dom结构时，发现组件的id在旧的dom结构中存在，那么react会认为当前组件只是位置发生了变化，因此不会将旧的组件销毁重新创建，只会改变当前组件的位置，然后再检查组件的属性有没有发生变化，然后选择保留或修改当前组件的属性。
- 用key是为了区分在前后两次渲染中元素的对应关系，防止发生不必要的更新操作。如果我们用index来标识key，数组在执行插入、排序等操作之后，原先的index并不再对应到原先的值，那么这个key就失去了本身的意义，并且会带来一些其他的问题。

react hooks解决什么问题?

- react hooks 解决了函数定义的组件没有生命周期和局部状态的问题，使用hooks可以模拟类定义组件的生命周期。

两种路由模式的区别？hash和history?

- hash路由
  hash模式是通过改变锚点(#)来更新页面URL，并不会触发页面重新加载，我们可以通过window.onhashchange监听到hash的改变，从而处理路由hash 虽然出现在 URL 中，但不会被包括在 HTTP 请求中，对后端完全没有影响，因此改变 hash 不会重新加载页面。
- history路由
  history模式是通过调用window.history对象上go、back、forward去操作浏览器的历史记录栈来实现页面的无刷新跳转。

解释useEffect的第二个参数传不同值的区别；

useEffect 拥有两个参数，第一个参数作为回调函数会在浏览器布局和绘制完成后调用，因此它不会阻碍浏览器的渲染进程。第二个参数是一个数组：

- 当数组存在并有值时，如果数组中的任何值发生更改，则每次渲染后都会触发回调。
- 当它不存在时，每次渲染后都会触发回调。
- 当它是一个空列表时，回调只会被触发一次，类似于componentDidMount。

什么是纯函数?

- 返回一个新值，并且没有任何副作用。

reducer是纯函数吗？为什么?

- reducer必须是一个纯函数，Redux只通过比较新旧两个对象的存储位置来比较新旧两个对象是否相同。如果你在reducer内部直接修改旧的state对象的属性值，那么新的state和旧的state将都指向同一个对象。因此Redux认为没有任何改变，返回的state将为旧的state。

你认为vue和react有什么区别?

- vue是单文件组件，组件内部是使用高度封装的模板（template）语法，数据是双向绑定。
- react是jsx语法，并且数据是单向流动的。
- vue是一个完整的框架，react主要是针对视图层的一个library
- vue和react都是用了虚拟的dom。

可控组件和非可控组件是什么?

- 主要是表单的value是否受state控制
- 需要自行监听onChange，更新state

介绍一下hooks?

- useState 定义局部状态
- useEffect 模拟生命周期
- useRefs 使用ref获取dom
- useContext context上下文
- useParams 获取url中的params
- useLocation 获取url中的location对象
- useMemo 做性能优化
- useCallback

redux性能优化的最佳实践

- pureComponent中自带shouldComponentUpdate这个生命周期。
- 如果是普通component，当组件接收到新的props时，render都会重新渲染，如果是pureComponent，那么当传入新的props时会跟旧的props进行一个浅比较，如果没有变化，那

么render函数不会重新渲染。
- 一般会和immutable.js这个库结合使用，会将性能优化做到极致。

虚拟dom是什么？为什么虚拟dom会提升react的性能?

- 虚拟dom实际上是一个js对象。
- 真实dom进行比对的时候，非常复杂，真实dom树上会有方法，有属性，有事件，会非常消耗性能。通过将真实dom对象转化成js中的对象，就不具有真实dom中的特性，单纯比较js对象性能就会比较快。

diff算法是什么?

- diff算法中如果key值相同就直接复用，不用重复地创建dom了。

webpack中,是借助loader完成JSX的转化？还是babel?

- 因为react的代码是没办法直接在浏览器中运行的，是借助脚手架将代码转化为ES5，才得以在浏览器中运行，（在vue中是借助webpack中的vue-loader转化）react是借助babel中的preset-react来转化的。

setState是异步的，你在什么时候遇到过坑?

- setState是异步的，导致在setState之后获取refs会是上一次的值。
- 放在setState中的第二个回调函数中就可以解决这个问题。

ref的作用是什么，你在什么业务场景下使用过refs?

- ref用来获取dom元素
- 比如当一个图片加载完毕的时候，我想获取这个图片的宽和高，就要用到refs，比如放大镜。

redux中间件的原理是什么?

- 中间件实际是对dispatch的改装
- 中间件指的是action和store之间对dispatch的改装
- 派发action的时候，先走中间件，再去store中
- 从action (中间件)-> store -> reducer -> store

你会把数据统一放到redux中管理，还是把共享数据放在redux中管理?

- 所有的数据都应该放在redux中去管理。
- 当一个组件中既有state又有props和redux存储来存储数据。一个组件二三百行，查错不知道去哪儿查，这样不便于维护。
- 当业务逻辑需要拓展的时候，state中存储数据，当别的组件需要你这个数据的时候，会很麻烦，redux更方便。
- 不必担心redux消耗内存，redux内存5个G（chrome浏览器）

componentWillReceiveProps的调用时机?

- componentWillReceiveProps这个生命周期在16.3之后已经被废弃，在组件接受到一个新的props时被调用，当组件初始化render的时候不会被调用。

setState是同步的还是异步的?

- 看具体情况：

- 首先如果直接在setState后面获取state的值是获取不到的，在原生环境中是异步的。
- 在异步请求（ajax中）或者setInterval,setTimeout，setState就是同步更新的。

执行两次setState的时候会render几次？会不会立即触发?

- 只执行一次，不会立即触发，因为react中有批处理机制，React会把setState的调用合并为一个来执行，也就是说，当执行setState的时候，state中的数据并不会马上更新，会按照先进先出，按顺序进行执行，但是在 Ajax、setTimeout 等异步方法中，每 setState 一次，就会 re-render 一次。

为什么直接修改this.state无效

setState本质是通过一个队列机制实现state更新的。 执行setState时，会将需要更新的state放入状态队列，而不会立刻更新state，队列机制可以批量更新state。
如果不通过setState而直接修改this.state，那么这个state不会放入状态队列中，下次调用setState时对状态队列进行合并时，会忽略之前直接被修改的state，这样我们就无法合并了，而且实际也没有把你想要的state更新上去。

react事件和dom事件区别

- 所有事件挂载到 document上；event不是原生对象，是syntheticEvent合成的事件对象；dispatchEvent

state 和 props 区别是啥?

- state 是组件自己管理数据，控制自己的状态，可变；
- props 是外部传入的数据参数，不可变；
- 没有state的叫做无状态组件，有state的叫做有状态组件；
- 多用 props，少用 state，也就是多写无状态组件。

请简述TCP\UDP的区别

- TCP面向连接，UDP面向非连接即发送数据前不需要建立链接
- TCP提供可靠的服务（数据传输），UDP无法保证
- TCP面向字节流，UDP面向报文
- TCP数据传输慢，UDP数据传输快

IP地址分为哪几类?

- 5

React.memo()和React.PureComponent组件异同：

- 异：React.memo()是函数组件，React.PureComponent是类组件。
- 同：都是对接收的props参数进行浅比较，解决组件在运行时的效率问题，优化组件的重渲染行为。
- useMemo()是定义一段函数逻辑是否重复执行。
- 若第二个参数为空数组，则只会在渲染组件时执行一次，传入的属性值的更新也不会有作用。所以useMemo()的第二个参数，数组中需要传入依赖的参数。

如何避免Ajax数据的重新获取

- 使用react-redux来进行全局状态的管理
- 判断redux中有没有之前的数据，如果有就直接复用，不需要再次发送请求。

组件是什么，类是什么，类被编译成什么?

- 组件是页面的一部分，是一个功能的集合。
- 类实际上是构造函数。
- 类就是被编译成了构造函数。

reselect是什么

- 类似于vue中的computed计算属性
- 派生数据
- 存在值的缓存，为了做性能优化

React-router的基本原理，hashHistory，和BrowserHistory。

- 使用HashHistory不需要后端做配合。
- BrowserHistory需要后端的配置，假如后端不太行，建议不用这个。

什么情况下使用异步组件

- Reloadable库
- 当主文件体积过大的时候，使用异步组件
- 就是一种路由懒加载（按需加载）

在react中如何防范XSS攻击

- 慎用：dangerousSetInnerHTML={{__html:alert(1)}}

getDerivedStateFromProps?

- 是为了替代16.3中被废弃的ComponentShouldUpdate