React

1. **虚拟DOM**：可以看作是一个使用javascript模拟了DOM结构的树形结构，相当于是真实dom和js之间的缓存
2. **hooks优点：**代码可读性更强，原本同一块功能的代码逻辑被拆分在了不同的生命周期函数中，容易使开发者维护和迭代，通过 React Hooks 可以将功能代码聚合，方便阅读维护
3. **Hooks：**1、useState 保存组件状态

2、useEffect 处理副作用

3、useContext 减少组件层级

4、useReducer

在使用上几乎跟 Redux/React-Redux 一模一样，唯一缺少的就 是无法使用 redux 提供的中间件。用法也很简单

5、useCallback 记忆函数

6、useMemo 记忆组件：只有在第二个参数数组的值发生变化时，才 会触发子组件的更新。

useCallback 的功能完全可以由 useMemo 所取代，

唯一的区别是：useCallback 不会执行第一个参数函数，而是将 它返回给你，而 useMemo 会执行第一个函数并且将函数执行结 果返回给你。所以在前面的例子中，可以返回 handleClick 来 达到存储函数的目的。所以 useCallback 常用记忆事件函数， 生成记忆后的事件函数并传递给子组件使用。而 useMemo 更适

合经过函数计算得到一个确定的值，比如记忆组件。

7、useRef 保存引用值：useRef 创建一个引用，不是拷贝

8、useImperativeHandle 透传 Ref

9、通过 useImperativeHandle 用于让父组件获取子组件内的DOM

10、useLayoutEffect 同步执行副作用

会在 DOM 更新之后同步执行；useLayoutEffect 会在 render，DOM 更新之后同步触发函数，会优于useEffect 异步触发函数

4.**redux**如何和组件链接数据  
 1.store通过reducer创建了初始状态；

2.view通过store.getState()将store中保存的state挂载在了自己的状态 上；

3.用户产生了操作，调用了actions 的方法；

4.actions的方法被调用，创建了带有标示性信息的action；

5.actions将action通过调用store.dispatch方法发送到了reducer中；

6.reducer接收到action并根据标识信息判断之后返回了新的state；

7.store的state被reducer更改为新state的时候，store.subscribe方法里的回调函数会执行，此时就可以通知view去重新获取state；

5.**函数组件**和**类组件**的区别

函数组件没有自己的状态，没有this，没有生命周期

​ 类组件有自己的状态和生命周期钩子函数，也能使组件直接访问 store 并维持状态

6.react**性能优化**

1. **.react.createElement**

react.createElement是用来创建react元素的，JSX 语法就是用 React.createElement()来构建 React 元素的。它接受三个参数，第一个参数可以是一个标签名，第二个参数为传入的属性，第三个参数作为组件的子组件

1. **Diff**
2. **Key**
3. **SetState是同步还是异步？**
4. **useRef**
5. **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 // 发送ajax请求

- 运行中状态(setState更新数据

时执行)：

- 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

1. **useEffect** 的第二个参数传不同值的区别

useEffect的第二个参数规则：可选，如果用上的话，这个参数必须是一个数组

第二个参数不传值：默认的行为，会每次 render 后都执行。

第二个参数是空数组：只运行一次，等同于类组件中的componentDidMount

第二个参数有一个值：值有变化就执行。

第二个参数有2个或2个值以上的数组，会比较每一个值，有一个不相等就执行。

1. 组件的执行与销毁先后
2. **纯函数：**

1.相同的输入永远返回相同的输出，2.不修改函数的输入值，3.不依赖外部环境状态。 4.无任何副作用

**15.reducer为什么**必须是**纯函数**.

**16.**react-router的实现原理

实现URL与UI界面的同步。

17.为什么要使用**pureComponent**?

● 当使用component时，父组件的state或prop更新时，无论子组件的state、prop 是否更新，都会触发子组件的更新，这会形成很多没必要的render，浪费很多性能

**PureComponent**与**Component**的区别？

● PureComponent自带通过props和state的浅对比来实现 shouldComponentUpate()， 而Component没有。

浅比较（shallowEqual）：是react源码中的一个函数，它代替了shouldComponentUpdate 的工作, 只比较外层数据结构，只要外层相同，则认为没有变化，不会深层次比较数据。

**pureComponent**的优缺点？

● 优点：不需要开发者使用shouldComponentUpdate就可使用简单的判断来提升性能；

● 缺点：由于进行的是浅比较，可能由于深层的数据不一致导致而产生错误的否定判 断，从而导致页面得不到更新。

为什么使用**pureComponent**可以提升性能？

● 主要在于pureComponent可以减少不必要的render，从而提高了性能，另外就是，不需要再手写shouldComponentUpdate里面的代码，从而节省了代码量；

1. 模块化开发

模块化开发：指文件的组织、管理、使用的方式。即把一个大的文件拆分成几个小的文件，他们之间相互引用、依赖

优点：1.可以加速渲染页面，所有资源加载的时间不会因为模块化而加速，但是模块化能加速渲染

2.避免命名冲突

3.代码重用高

4.思路更为清晰，降低代码耦合

19.执行两次**setState**，**render**几次，会不会立即触发。