1. **new的时候发生了什么事**
2. 创建一个空对象
3. 把当前类的属性和方法拷贝给空对象
4. 抛出这个对象
5. **vue的数据类型有两种**
6. 自己组建内部的date
7. 父组件传过来的props
8. **react初次dom渲染的函数**

constructor –> componentWillMount --> render –> componentDidMount

1. constructor：阶段可做与数据相关的操作，如：获取sectionstroge、Cookie

localstroge，和 请求数据

1. componentWillMount：功能与constructor类似
2. render：他是一个纯函数，不能调用setstate进行数据改变，不做数据的产生和保存，它的作用是返回数据和dom拼接好的jsx结构
3. componentDidMount：dom编译完成，并且渲染到真实dom中。可以做dom处理，可以做具体事件监听，还可以做某些插件实例化。
4. vue中视图的更新是异步的，react中数据的更新是异步的
5. **react更新阶段的生命周期函数**

componentWillReceiveProps 🡪 shouldComponentUpdate 🡪 componentWillUpdate 🡪 render 🡪 componentDidUpdate

1. componentWillReceiveProps：相当于vue的computed计算属性，作用：当父组件出入子组件的数据进行改变是，会执行此函数，可以将props再次赋值给state。也可以说是用来监听props变化的
2. shouldComponentUpdate：这是一个优化函数，此处必须有返回值true或者false，返回值为true时才会更新视图，在提高项目渲染性能时用。
3. componentWillUpdate：在render之前最后一次可以改变数据的地方
4. componentDidUpdate：同componentDidMount一个用法
5. **卸载过程**

componentWillUnmount：与dom相关的操作，在组件销毁前，一定在这里做相应的处理，比如事件监听，插件的dom实例化等等

1. 处理react的类定义的方法时，无法自动绑定this，解决此问题一可以在constructor的函数中用bind绑定，this.clickFn=this.clickFn.bind(this)，二在函数调用时用箭头函数绑定：<button onClick={()=>this.clickFn()}>
2. **Jsx的 onClick和html的onclick的区别**

Html onclick的弊端：注册的事件处理函数是全局环境，会污染全局环境。使用onclick的dom元素，如果在dom树中删除时，必须手动注销事件处理器，否则会造成内存泄漏

Jsx的onclick：挂载的没一个函数都是组件内部的，而不是全局的，无论多少个onclick都是事件委托的方式，在dom树顶层添加一个事件监听函数，此函数会根据具体组件分配具体的函数

1. **组件通讯**
2. 父给子传递，props
3. 子给父传递，借用callback函数
4. 兄弟组件之间的通讯：状态提升（借用公共父组件传值）和 发布者订阅者模式
5. ref只可以由父组件获取子组件的数据
6. this.props.children获取组件开标签、闭标签之间的内容
7. **解决路由匹配问题的方法**
8. 越详细的路由越要往前放，权限越大的路由越往后放
9. 给权限大的路由加一个exact属性
10. **路由组件的渲染方式**
11. Component：可以渲染状态组件和无状态组件
12. Render：只能渲染无状态组件
13. **路由钩子函数**
14. Basename：添加基路由
15. getUserconfirmation:第一次进入路由时调用的钩子函数，无实际意义
16. keyLength：设置它里面路由的location.Key的长度，默认是6。Key的作用：点击同一个链接时，每次该路由下的location.Key都会改变，可以通过key变化来刷新页面
17. **route 的props的三个属性：**

match、loction、history

1. **高阶组件**
2. 高阶组件抛出的是函数而不是组件
3. 函数里面抛出组件
4. 高阶组件使用时会接收一个组件作为参数传入
5. 一般采用state的状态来管理是否到了渲染传入组件的时机
6. 如果处理数据，获取数据落其他的业务处理完毕，再渲染传入组件