1, React中有哪些生命周期函数?

componentWillMount() 渲染前 (不建议使用)

componentDidMount() 渲染后

componentWillReceiveProps() 接收props传值时

shouldComponentUpdate()控制组件是否更新

componentWillUpdate() 组件将要更新

componentDidUpdate() 组件已经更新

componentWillUnmount() 组件将要移除

2, React组件中的state 和 props 有何区别?

· State 是一种数据结构，在组件构造器中定义, 是可读可写的, 用于组件内部数据的初始化和更新。 state中一般只放纯数据

· Props 则是组件的配置。props 由父组件传递给子组件，就子组件而言，props 是只读的。组件不能改变自身的 props，但可以修改传递给子组件的props。props也可以传递回调函数

3, React中keys是什么,有什么作用?

Keys是列表通过map循环时给循环标签添加的属性,用于标记每一个循环的元素

在循环中要保证每一个循环标签keys属性值都不相同

在列表数据更新时, 通过keys可以快速高效的区分哪些元素是新的,然后确保视图更新的正确和高效

4, React组件之间通讯有哪些方式?

父组件向子组件传值: 父组件通过自定义属性向子组件传值,子组件props参数接收并处理

子组件向父组件传值: 父组件通过自定义属性向子组件传函数,子组件props接收函数并调用

非父子组件传值: 在全局作用域下定义变量, 通过在不同组件中对全局变量的赋值与取值来实现组件传值

5, react的优缺点

优点：

可以通过构造函数或类结构描述视图组件,

集成虚拟DOM（渲染性能好）

单向数据流（好处是更容易追踪数据变化排查问题)

一切都是component：代码更加模块化，重用代码更容易，可维护性高

大量使用 es6 新特性

缺点：

jsx的一个问题是，渲染函数render()常常包含大量逻辑，最终看着更像是程序片段，而不是视觉呈现。后期如果发生需求更改，维护起来比较麻烦

功能强大而全面，比vue更难上手

6, react使用单向数据流有什么好处?

单向数据流是对数据传递的一种约束, 他保证了组件的数据传递结构稳定且不易耦合.

数据之能从父组件向下流动到子组件中，反过来则不行。这样会防止从子组件意外改变父级组件的状态 , 极大的降低了我们组件间通信的代码耦合

数据流动单一, 便于追踪, 追查问题比较便捷

7,使用jsx时有哪些注意事项?

Jsx渲染的时候 for 和class不能使用，必须通过htmlfor和classname替代

组件模板外部使用js不能用“{}“包裹，但模板内部必须使用“{}“包裹起来

jsx中不支持单标签写法, html中的单标签必须写成双标签或末尾加/

如果标签有多层嵌套结构, 建议在根标签外层加() 而且()必须跟return在一行

8, 类组件和函数式组件有何不同?

函数式组件通过ES5的构造函数结构创建, 一般以数据的展示为主, 功能简单, 组件中的逻辑代码较少

类组件通过ES6的类结构创建, 允许使用更多功能,如组件状态数据,生命周期钩子, 访问redux仓库等

9, refs的作用是什么?

Refs是React提供给我们安全的访问DOM元素或者某个组件实例的句柄, 它是组件对象this的一个属性, 可以在组件模板中的标签添加ref属性, 然后组件中使用this.refs即可访问对应的DOM元素

10,shouldComponentUpdate函数有什么作用?

shouldComponentUpdate这个函数是用来判断是否需要调用render函数重新描绘DOM, 因为DOM的绘制非常消耗性能, 如果我能能在这个函数中写一些优化算法逻辑,控制DOM绘制的频率和次数, 则能极大的提高网页渲染效率, 优化性能

11, react路由跳转时如何传递数据?

路由传值一共有4中方式:

(1)使用url添加?拼接字符串形式传值, 目标组件使用this.props.location.search 接收

(2)使用友好url动态传值, 目标组件使用this.props.match.params接收

(3)使用query对象传值, 目标组件使用this.props.location.query接收

(4)使用state对象传值, 目标组件使用this.props.location.state接收

此外还可以使用编程式导航跳转路由并传值this.props.history.push()

注意：使用对象传值以及编程式导航传值时如果页面刷新，那么传递的值就会消失；

12,什么是Redux?

Redux是一款热门的前端开发库, 它是javascript程序的可预测状态容器,可用于react项目的状态管理, 使用Redux开发的应用易于测试,可以在不同环境中运行,并显示相同行为.

13,Redux有哪三大原则?

1）唯一数据源(整个应用的 state 状态数据被储存在一个状态树(对象)中，单一状态树更容易跟踪数据的变化, 方便调试检查应用程序)

2）State 状态是只读的, 想要更改数据必须经过派发action事件,通过接收action参数修改

3）reducer必须是纯函数（一个输入必须对应着唯一的输出, 返回值取决于参数）

14,什么情况下建议使用redux?

某个组件的状态，需要共享

某个状态需要在任何地方都可以拿到

一个组件需要改变全局状态

一个组件需要改变另一个组件的状态

15,React组件之间传递数据有哪些方式?

组件传值: 包括父传子, 子传父, 兄弟组件

路由传值: 包括url拼接传值, 友好url传值和对象传值

Redux传值: 把需要传递的数据放入状态管理中,各组件共享

16, 说一下你对高阶组件的理解

高阶组件: 是 React 中用于重用组件逻辑的高级技术, 它本身不是react中的组件, 而是一个函数, 这个函数接受一个react组件作为参数,并返回一个新组件, 实现了对原有组件的增强和优化, 可以对原有组件中的state, props和逻辑执行增删改操作, 一般用于代码重用和组件增强优化

17, 高阶组件有哪些实现方式?

属性代理。高阶组件通过被包裹的React组件来操作props, 可以增强组件模板和props

反向继承。高阶组件继承于被包裹的React组件 可以更新state

18, 说说你对Hooks组件的理解

hooks组件即使用了hooks语法构建的函数式组件

hooks是react中的一项新功能, 它可以在不使用class类的情况下实现state组件状态和生命周期等功能, 通过useState函数实现组件状态,通过useEffect函数实现生命周期

hooks语法是向下兼容的, 在旧版本的react项目中可用.

hooks 可以很好的替代高阶组件实现组件的抽象和复用

hooks只能在函数式组件中使用, 不能在普通函数或class类组件中使用, 也不建议在循环或判断逻辑中使用

19, Hooks 与 React 生命周期的关系

Hooks可以模拟实现react组件的生命周期,通过API函数useEffect并控制其第二个参数的传入可以模拟组件不同时期的生命周期钩子

20, Hooks 与 高阶组件有何区别?

它们之间最大的不同在于，高阶组件仅仅是一种开发模式，它本身是js函数结构, 而hooks是react提供的API模式，它既能更加自然的融入到react的渲染过程也更加符合react的函数编程理念。

21,什么是渲染劫持?

渲染劫持指对一个组件渲染内容的装饰或修改, 一般通过高阶组件来实现, 把一个组件传入高阶组件, 可以对这个组件的模板进行修改后执行渲染, 也可以阻止组件渲染,并修改组件中的数据和逻辑

22,react严格模式有什么作用?

StrictMode严格模式是一个用来突出显示应用程序中潜在问题的工具, 它可以识别不安全的生命周期调用, 警告过时或已弃用的API, 并检测意外的副作用

23,react有状态组件和无状态组件有什么区别?

有状态组件通过class类结构定义,也叫类组件,主要用于处理业务逻辑和做数据交互

无状态组件通过函数结构定义, 也叫函数式组件, 主要用于定义模板,用于数据的展示

24,函数式组件有没有生命周期?

函数时组件默认是没有生命周期的,因为函数式组件的主要功能是展示数据, 如果需要做很多业务逻辑的情况下可以选用类组件,使用类组件的生命周期, 也可以使用hooks提供的useEffect函数模拟实现组件的生命周期

25,什么是受控组件和非受控组件?

受控组件和非受控组件是针对表单组件处理数据时的不同概念

受控组件指组件的状态数据根据用户输入,实时更新,显示在视图中, 例如input标签使用onChange绑定输入事件并通过setSatate函数更新state数据, 此时组件中的数据是可控的

非受控组件指组件状态数据与表单标签没有直接关联, 用户输入与视图更新不同步, 例如input标签没有绑定onChange事件或者value属性, 而使用refs 的DOM查找操作表单数据, 并用作逻辑处理, 此时组件中的数据是不可控的

总之一句话: 受控组件就是内部表单通过了value和onChange绑定的组件

26, 说一说你对react虚拟DOM的理解

虚拟DOM是对DOM的抽象，本质上是JavaScript对象, 即通过js对象模拟DOM中的节点,由于react使用jsx语法构建页面,浏览器无法直接运行jsx, 所以会把jsx结构解析成js对象结构, 然后通过js对象渲染DOM树, 当数据更新时虚拟DOM会通过内部的diff算法,得到节点的变化,从而局部更新变化的DOM节点, 避免了不必要的DOM操作, 提高性能

27, 说一下react虚拟DOM优化性能的实现步骤

用JavaScript对象结构表示DOM树的结构(虚拟DOM)；然后用这个js对象构建一个真实DOM树，插到文档当中

当状态变更的时候，重新构造一棵新的虚拟DOM树。然后通过diff算法比较新的树和旧的树，记录两棵树差异

把diff算法比较的差异更新到步骤1所构建的真正的DOM树上，视图就更新了

28, 说一说虚拟DOM中diff的原理

diff算法就是进行虚拟节点对比，并返回一个patch对象，用来存储两个节点不同的地方，然后用patch记录的不同点去局部更新DOM, 相当于给真实DOM打补丁更新