**REACT面试题汇总**

**1.如何理解React中的组件间数据传递？**

**a)父--子 通过prop传递**

**b)子--父 在父中创建一个可以修改state的方法，之后把这个方法通过prop传递给子，在子中调用这个方法从而达到修改父state的目的。**

**c)非父子组件如何数据共享or数据通信：创建全局对象，监视对象on，数据发送方触发自定义事件同时传递数据。**

**2.使用create-react-app后创建的工程下有registerServiceWorker.js是做什么用的？**

**答：registerServiceWorker就是为react项目注册了一个service worker，用来做资源的缓存，这样你下次访问时，就可以更快的获取资源。而且因为资源被缓存，所以即使在离线的情况下也可以访问应用（此时使用的资源是之前缓存的资源）。注意，registerServiceWorker注册的service worker 只在生产环境中生效（process.env.NODE\_ENV === ‘production’）**

**3.setState后发生了什么？**

**答：点击事件开始时，框架调用ReactEventListener.dispatchEvent之后执行batchedUpdates开启事务。setState中通过this.updater.enqueueSetState方法把state加入队列(react-dom中实现了enqueueSetState方法)事件绑定的方法执行完成以后，框架会继续执行batchedUpdates进行批量更新。**

**4.React性能优化是哪个周期函数？**

**答：shouldComponentUpdate这个方法用来判断是否需要调用render方法重新描绘dom。因为dom的描绘非常消耗性能，如果我们能在shouldComponentUpdate方法中能够写出更优化的dom diff算法，可以极大的提高性能**

**5.React性能优化方案**

**a) 重写shouldComponentUpdate来避免不必要的dom操作**

**b) 使用 production 版本的react.js**

**c) 使用key来帮助React识别列表中所有子组件的最小变化。**

**6.在哪个生命周期事件中你会发出AJAX请求，为什么**

**答：AJAX 请求应该在 componentDidMount 生命周期事件中。 有几个原因:Fiber，是下一次实施React的和解算法，将有能力根据需要启动和停止渲染，以获得性能优势。其中一个取舍之一是 componentWillMount ，而在其他的生命周期事件中出发 AJAX 请求，将是具有 “非确定性的”。 这意味着 React 可以在需要时感觉到不同的时间开始调用 componentWillMount。**

**在 componentDidMount 中执行 AJAX 将保证至少有一个要更新的组件。**

**7.什么是React的refs，为什么他们很重要**

**答：refs就像是一个逃生舱口，允许您直接访问DOM元素或组件实例。为了使用它们，您可以向组件添加一个 ref 属性，该属性的值是一个回调函数，它将接收底层的 DOM 元素或组件的已挂接实例，作为其第一个参数。**

**8.双向数据绑定和单项数据的区别？**

**a)单向数据流中，父组件给子组件传递数据，但反过来不可以传递，也就是说单向数据流是从最外层节点传递到子节点，他们只需从最外层节点获取props渲染即可，如果顶层组件的 某个prop改变了，React会递归的向下便利整棵组件树，重新渲染所有使用这个属性的组件， React组件内部还具有自己的状态，这些状态只能在组件内修改；双向数据绑定是数据与视图 双向绑定，数据发生改变时，视图也改变，视图发生改变时，数据也会发生改变。**

**b)双向数据绑定的各种数据相互依赖相互绑定，导致数据问题的源头难以被跟踪到；单向 数据流的数据流动方向可以跟踪，流动单一，追查问题的时候可以更快捷，缺点是写起来不太方便，要使视图发生改变就得创建各种action来维护state。**

**9. React生命周期函数**

**a) 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,最常用**

**b) 运行中的状态：**

**componentWillReceiveProps:组件将要接收到属性的时候调用**

**shouldComponentUpdate(prestate, props):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**

**c) 销毁阶段：**

**componentWillUnmount:组件即将销毁,清理工作，事件解除绑定，清理定时器，释放资源**

**10. 怎么干预组件是否应该被渲染？怎么强制组件渲染**

**a) 在组件基本的生命周期中,shouldComponentUpdate 如果实现了这个函数,那么组件在决定重新渲染之前会先调用该函数,如果该函数返回值为false那么组件会放弃重新渲染.**

**b) react框架使用forceUpdate函数向开发者提供强制启动渲染能力,在使用该函数时,组件重新渲染之前不会调用shouldComponentUpdate.**

**11、你怎样理解“在React中，一切都是组件”这句话。**

**组件是 React 应用 UI 的构建块。这些组件将整个 UI 分成小的独立并可重用的部分。**

**每个组件彼此独立，而不会影响 UI 的其余部分。**

**12、React组件生命周期的阶段是什么？**

**React 组件的生命周期有三个不同的阶段：**

**初始渲染阶段：这是组件即将开始其生命之旅并进入 DOM 的阶段。**

**更新阶段：一旦组件被添加到 DOM，它只有在 prop 或状态发生变化时才可能更新和重新渲染。这些只发生在这个阶段。**

**卸载阶段：这是组件生命周期的最后阶段，组件被销毁并从 DOM 中删除。**

**13、React 中 key 的重要性是什么？**

**key 用于识别唯一的 Virtual DOM 元素及其驱动 UI 的相应数据。它们通过回收 DOM 中当前所有的元素来帮助 React 优化渲染。这些 key 必须是唯一的数字或字符串，React 只是重新排序元素而不是重新渲染它们。这可以提高应用程序的性能。**

**14、React diff 原理？**

**把树形结构按照层级分解，只比较同级元素。**

**给列表结构的每个单元添加唯一的 key 属性，方便比较。**

**React 只会匹配相同 class 的 component（这里面的 class 指的是组件的名字）**

**合并操作，调用 component 的 setState 方法的时候, React 将其标记为 dirty.到每一个事件循环结束, React 检查**

**所有标记 dirty 的 component 重新绘制.**

**选择性子树渲染。开发人员可以重写 shouldComponentUpdate 提高 diff 的性能。**

**15、应该在 React 组件的何处发起 Ajax 请求**

**在 React 组件中，应该在 componentDidMount 中发起网络请求。这个方法会在组件第一次“挂载”(被添加到 DOM)时执行，在组件的生命周期中仅会执行一次。更重要的是，你不能保证在组件挂载之前 Ajax 请求已经完成，如果是这样，**

**也就意味着你将尝试在一个未挂载的组件上调用 setState，这将不起作用。在 componentDidMount 中发起网络请求将**

**保证这有一个组件可以更新了。**

**16、Redux 有什么缺点？**

**一个组件所需要的数据，必须由父组件传过来，而不能像 flux 中直接从 store 取。**

**当一个组件相关数据更新时，即使父组件不需要用到这个组件，父组件还是会重新 render，可能会有效率影响，或者需**

**要写复杂的 shouldComponentUpdate 进行判断。**

**17、React 中 Element 与 Component 的区别是？**

**简单而言，React Element 是描述屏幕上所见内容的数据结构，是对于 UI 的对象表述。典型的 React Element 就是利**

**用 JSX 构建的声明式代码片然后被转化为createElement的调用组合。而 React Component 则是可以接收参数输入并且**

**返回某个 React Element 的函数或者类。**

**18、React性能优化方案**

**（1）重写shouldComponentUpdate来避免不必要的dom操作。**

**（2）使用 production 版本的react.js。**

**（3）使用key来帮助React识别列表中所有子组件的最小变化。**

**19、(组件的)状态(state)和属性(props)之间有何不同**

**State 是一种数据结构，用于组件挂载时所需数据的默认值。State 可能会随着时间的推移而发生突变，但多数时候是作**

**为用户事件行为的结果。**

**Props(properties 的简写)则是组件的配置。props 由父组件传递给子组件，并且就子组件而言，props 是不可变的(immutable)。组件不能改变自身的 props，但是可以把其子组件的 props 放在一起(统一管理)。Props 也不仅仅是数据--回调函数也可以通过props 传递。**

**20、调用 super(props) 的目的是什么？**

**在 super() 被调用之前，子类是不能使用 this 的，在 ES2015 中，子类必须在 constructor 中调用 super()。**

**传递 props 给 super() 的原因则是便于(在子类中)能在 constructor 访问 this.props。**

**21. 区分Real DOM和Virtual DOM**

**Real DOM**

**1. 更新缓慢。**

**2. 可以直接更新 HTML。**

**3. 如果元素更新，则创建新DOM。**

**4. DOM操作代价很高。**

**5. 消耗的内存较多。**

**Virtual DOM**

**1. 更新更快。**

**2. 无法直接更新 HTML。**

**3. 如果元素更新，则更新 JSX 。**

**4. DOM 操作非常简单。**

**5. 很少的内存消耗。**

**22. 什么是React？**

**React 是 Facebook 在 2011 年开发的前端 JavaScript 库。**

**它遵循基于组件的方法，有助于构建可重用的UI组件。**

**它用于开发复杂和交互式的 Web 和移动 UI。**

**尽管它仅在 2015 年开源，但有一个很大的支持社区。**

**23. React有什么特点？**

**React的主要功能如下：**

**它使用虚拟DOM 而不是真正的DOM。**

**它可以进行服务器端渲染。**

**它遵循单向数据流或数据绑定。**

**24. 列出React的一些主要优点。**

**React的一些主要优点是：**

**它提高了应用的性能**

**可以方便地在客户端和服务器端使用**

**由于 JSX，代码的可读性很好**

**React 很容易与 Meteor，Angular 等其他框架集成**

**使用React，编写UI测试用例变得非常容易**

**25. React有哪些限制？**

**React的限制如下：**

**React 只是一个库，而不是一个完整的框架**

**它的库非常庞大，需要时间来理解**

**新手程序员可能很难理解**

**编码变得复杂，因为它使用内联模板和 JSX**

**26. 什么是JSX？**

**JSX 是J avaScript XML 的简写。是 React 使用的一种文件，它利用 JavaScript 的表现力和类似 HTML 的模板语法。 这使得 HTML 文件非常容易理解。此文件能使应用非常可靠，并能够提高其性能。下面是JSX的一个例子：**

**render(){**

**return(**

**<div>**

**<h1> Hello World from Edureka!!</h1>**

**</div>**

**);}**

**27. 你了解 Virtual DOM 吗？解释一下它的工作原理。**

**Virtual DOM 是一个轻量级的 JavaScript 对象，它最初只是 real DOM 的副本。它是一个节点树，它将元素、它们的属性和内容作为对象及其属性。 React 的渲染函数从 React 组件中创建一个节点树。然后它响应数据模型中的变化来更新该树，该变化是由用户或系统完成的各种动作引起的。**

**Virtual DOM 工作过程有三个简单的步骤。**

**Virtual DOM 是一个轻量级的 JavaScript 对象，它最初只是 real DOM 的副本。它是一个节点树，它将元素、它们的属性和内容作为对象及其属性。 React 的渲染函数从 React 组件中创建一个节点树。然后它响应数据模型中的变化来更新该树，该变化是由用户或系统完成的各种动作引起的。**

**Virtual DOM 工作过程有三个简单的步骤。**

**每当底层数据发生改变时，整个 UI 都将在 Virtual DOM 描述中重新渲染。**

**然后计算之前 DOM 表示与新表示的之间的差异。**

**完成计算后，将只用实际更改的内容更新 real DOM。**

**28. 为什么浏览器无法读取JSX？**

**浏览器只能处理 JavaScript 对象，而不能读取常规 JavaScript 对象中的 JSX。所以为了使浏览器能够读取 JSX，首先，需要用像 Babel 这样的 JSX 转换器将 JSX 文件转换为 JavaScript 对象，然后再将其传给浏览器。**

**29. React与Angular有何不同？**

**1. react虚拟DOM实现原理，以及为什么虚拟 dom 会提高性能**

**1. 用 js对象结构表示 DOM 树的结构；然后用这个树构建一个真正的 DOM 树，插到文档当中。**

**2. 当状态变更的时候，重新构造一棵新的对象树。然后对比新旧虚拟DOM树，记录两棵树差异。**

**3. 把 2 所记录的差异应用到步骤 1 所构建的真正的 DOM 树上，视图就更新了。**

**原因：虚拟 dom 相当于在 js 和真实 dom 中间加了一个缓存，利用 dom diff 算法减少了对真实DOM的操作次数，从而提高性能。**

**30.React的优势以及特点**

**优势：**

**1. 实现对虚拟DOM的操作，使得它速度快，提高了Web性能。**

**2. 组件化，模块化。react里每一个模块都是一个组件，组件化开发，可维护性高。**

**3. 单向数据流，比较有序，有便于管理，它随着React视图库的开发而被Facebook概念化**

**4. 跨浏览器兼容：虚拟DOM帮助我们解决了跨浏览器问题，它为我们提供了标准化的API，甚至在IE8中都是没问题的。**

**不足：**

**1. react中只是MVC模式的View部分，要依赖引入很多其他模块开发。、**

**2. 当父组件进行重新渲染操作时，即使子组件的props或state没有做出任何改变，也会同样进行重新渲染。**

**特点：**

**1. 声明式设计：React采用声明范式，可以轻松描述应用。**

**2. 高效：React通过对DOM的模拟，最大限度地减少与DOM的交互。**

**3. 灵活：React可以与已知的库或框架很好地配合。**

**31.React-Router的实现原理**

**原理：实现URL与UI界面的同步。其中在react-router中，URL对应Location对象，**

**而UI是由react components来决定的，这样就转变成location与components之间的同步问题。**

**优点：**

**1. 风格: 与React融为一体,专为react量身打造，编码风格与react保持一致，例如路由的配置可以通过component来实现**

**2. 简单: 不需要手工维护路由state，使代码变得简单**

**3. 强大: 强大的路由管理机制，体现在如下方面**

**4. 路由配置: 可以通过组件、配置对象来进行路由的配置**

**5. 路由切换: 可以通过<Link> Redirect进行路由的切换**

**6.路由加载: 可以同步记载，也可以异步加载，这样就可以实现按需加载7.使用方式: 不仅可以在浏览器端的使用，而且可以在服务器端的使用**

**缺点：API不太稳定，在升级版本的时候需要进行代码变动。**

**32.React 中有三种构建组件的方式**

**React.createClass()、ES6 class 和无状态函数**

**33.指出(组件)生命周期方法的不同**

**componentWillMount -- 多用于根组件中的应用程序配置**

**componentDidMount -- 在这可以完成所有没有 DOM 就不能做的所有配置，并开始获取所有你需要的数据；如果需要设置事件监听，也可以在这完成**

**componentWillReceiveProps -- 这个周期函数作用于特定的 prop 改变导致的 state 转换**

**shouldComponentUpdate -- 如果你担心组件过度渲染，shouldComponentUpdate 是一个改善性能的地方，因为如果组件接收了新的 prop， 它可以阻止(组件)重新渲染。shouldComponentUpdate 应该返回一个布尔值来决定组件是否要重新渲染**

**componentWillUpdate -- 很少使用。它可以用于代替组件的 componentWillReceiveProps 和 shouldComponentUpdate(但不能访问之前的 props)**

**componentDidUpdate -- 常用于更新 DOM，响应 prop 或 state 的改变**

**componentWillUnmount -- 在这你可以取消网络请求，或者移除所有与组件相关的事件监听器**

**34.当渲染一个列表时，何为 key？设置 key 的目的是什么**

**Keys 会有助于 React 识别哪些 items 改变了，被添加了或者被移除了。Keys 应该被赋予数组内的元素以赋予(DOM)元素一个稳定的标识，选择一个 key 的最佳方法是使用一个字符串，该字符串能惟一地标识一个列表项。很多时候你会使用数据中的 IDs 作为 keys，当你没有稳定的 IDs 用于被渲染的 items 时，可以使用项目索引作为渲染项的 key，但这种方式并不推荐，如果 items 可以重新排序，就会导致 re-render 变慢。**

**35.类组件(Class component)和函数式组件(Functional component)之间有何不同**

**类组件不仅允许你使用更多额外的功能，如组件自身的状态和生命周期钩子，也能使组件直接访问 store 并维持状态**

**当组件仅是接收 props，并将组件自身渲染到页面时，该组件就是一个 '无状态组件(stateless component)'，可以使用一个纯函数来创建这样的组件。这种组件也被称为哑组件(dumb components)或展示组件**

**36.应该在 React 组件的何处发起 Ajax 请求**

**在 React 组件中，应该在 componentDidMount 中发起网络请求。这个方法会在组件第一次“挂载”(被添加到 DOM)时执行，在组件的生命周期中仅会执行一次。更重要的是，你不能保证在组件挂载之前 Ajax 请求已经完成，如果是这样，也就意味着你将尝试在一个未挂载的组件上调用 setState，这将不起作用。在 componentDidMount 中发起网络请求将保证这有一个组件可以更新了。**

**37.React diff 原理**

**把树形结构按照层级分解，只比较同级元素。**

**给列表结构的每个单元添加唯一的 key 属性，方便比较。**

**React 只会匹配相同 class 的 component（这里面的 class 指的是组件的名字）**

**合并操作，调用 component 的 setState 方法的时候, React 将其标记为 dirty.到每一个事件循环结束, React 检查所有标记 dirty 的 component 重新绘制.**

**选择性子树渲染。开发人员可以重写 shouldComponentUpdate 提高 diff 的性能。**