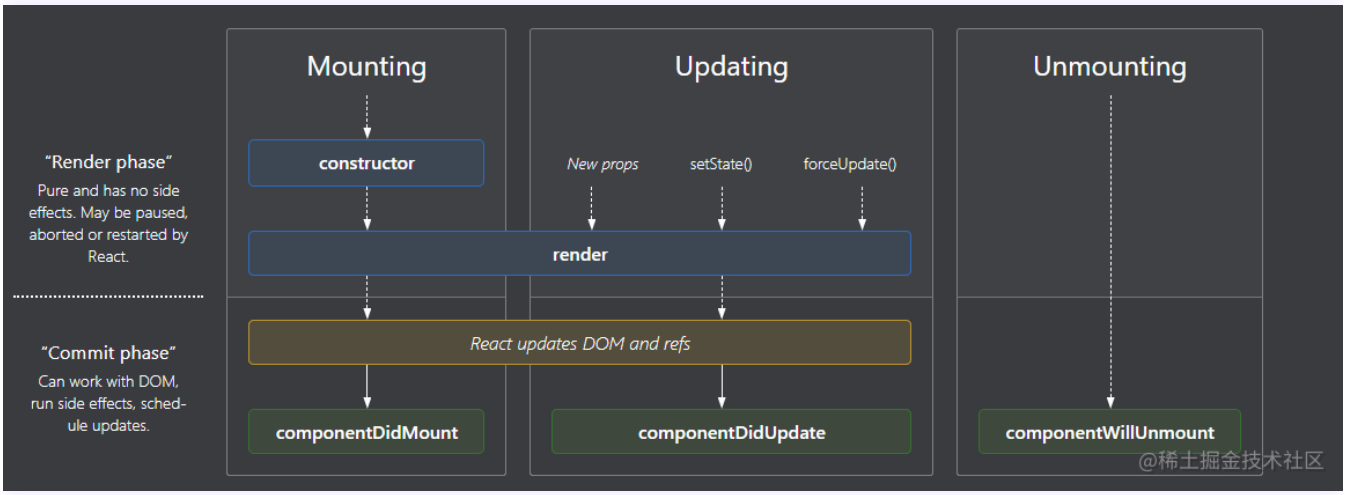
React面试题

最近在准备秋招，记录一些关于React的面试题~

React生命周期

类式组件



挂载阶段

如图，如果是挂载阶段，按如下步骤

- constructor
- 调用render函数
- componentDidMount

更新阶段

- shouldComponentUpdate
- render
- componentDidUpdate

卸载阶段

- componentWillUnMount

函数式组件

其实没有必要将函数式组件要类比于类式组件的生命周期，因为生命周期只是不同时期执行不同的回调而已，这还引入了时间这一概念，React的核心思想是声明式渲染，单向数据流，我们只需要关注状态，状态改变，视图更新就行了。

为什么要有hook，hook的出现解决了什么问题

- 使函数式组件可以有自己的状态，可以处理各种逻辑，而不是纯展示。
- 代码复用，以前类式组件用HOC，这样用着用着会导致嵌套很深问题，而自定义hook更简洁。
- 不用关注this问题，类式组件的this可不好理解
- 使代码逻辑更加合理，以前监听事件和卸载事件要分开生命周期些，而现在可以都写在同一个函数里。
- hook能解决大部分业务场景问题。

React中的render()发生了什么

- 经过babel转换，会调用React.createElement()，生成ReactElement。
- 根据ReactElement生成Fiber，然后构建Fiber树
- 如果是初次挂载的话，则将指针指向currentFiber，在commit阶段渲染到页面
- 如果是更新阶段的话，则会有diff这个过程，会给对应的fiber节点打上effectTag，然后放到一条effectList上，在commit阶段根据这个链表去更新dom，最终渲染到页面上

函数式组件的return相当于执行render()，其实这个过程也是Reconciler的过程

React的更新机制（父子组件）

React是单向数据流，如果当前组件状态改变，需要更新，则会以当前组件为目录，他的子组件也要全部重新渲染，那么怎么优化呢？

答案是用React.memo()去包裹，只有当prop改变时，才会更新

React的更新机制（setState）

众所周知，React是声明式渲染，setState调用后，组件会重新渲染，那么事实真的是如此吗，setState有什么策略优化吗？

那么如果原来的状态为1，setState(1)，组件还会重新渲染吗，这个大家可以思考一下，我实操过，是不会的，估计是setState后，React会进行一层浅比较，如果没有变，则不会重新渲染。

其实这个大家都懂，如果state是一个对象，那么我们setState的时候一般都会去set一个新对象，而不是用原来的对象，因为不这样做的话，引用没变，不会重新渲染，这其实是函数式思想的一种体现，我们一般不会修改源对象，而是用一个新对象去替换它。

还有一个策略，如果你一个回调函数，执行了多个setState，那么不会触发多次渲染，而是只执行一次渲染。

useEffect和useLayoutEffect

其实这两个函数都是处理副作用的，大部分场景下其实使用useEffect就足够了 具体区别是执行的时机不同

React的更新流程如下：

- 状态更新
- 调用render()，函数式组件是return，其实这个过程也是Reconciler的过程，会去diff，进行一系列操作
- 然后生成新的DOM
- 将修改绘制到页面上

那么useEffectLayout会在绘制到页面前执行，会阻塞绘制，此时获取到的是更新后的DOM，如果你想在绘制前做一些操作，那么可以用这个。 而useEffct是绘制完后执行的，不会阻塞绘制，是异步的。

React的事件机制

React的事件并不是直接绑定到DOM上的，而是将事件统一绑定到Continer容器上的，这样的话就可以在组件挂载销毁时**统一订阅和移除事件**

另外，他的事件也不是浏览器原生事件，而是React自己实现的合成事件

实现合成事件的目的主要是，**抹平浏览器之间的兼容问题，而且便于跨端开发**，

React中有一个合成事件层，事件都会冒泡到容器上，然后触发，至于事件怎么对应到组件，那就是会有一个映射去保存了。

结尾

以上是本人的一些学习思考总结，如果有写的不对地方，麻烦大家指出一下~