React 性能优化浅析

许峰

2019年12月26日

- 个人理解
- 工具
- useMemo 和 useCallback
- 总结

React本来就执行很快，不要过早的担心或者优化性能。

先有质量地实现需求，然后去找到性能的瓶颈去优化代码。

优化方向：

- 减少重新 render 的次数。
- 减少计算的量。

性能优化不能靠感觉，可以借助工具和代码来提高一些组件的性能。

工具主要有两个: Chrome Performance, React Developer Tools Profiler。

查看每个组件发生了什么，能够确定哪个组件可能会出现性能问题，然后进行优化。

React Developer Tools Profiler 中勾选Highlight Updates,黄色和红色并不一定是不好的，之所以出现黄色或者红色，组件这个时候确实因为某些state或者props改变导致了频繁更新。

代码有: why-did-you-update、 useWhyDidYouUpdate

https://github.com/maicki/why-did-you-update
https://usehooks.com/useWhyDidYouUpdate/

重复渲染:

在Function组件中，每次重新渲染意味着整个功能将再次运行。

重新渲染的原因：state 发生变换、props 发生变化、父组件重新渲染。

useMemo 和 useCallback

1. 传递给 useMemo 的函数开销?

如果执行的操作开销不大，那么就不需要记住返回值。使用 useMemo 的成本可能会超过该函数的成本。

2. 记忆（memoized）值的引用

给定相同的输入值时，对记忆（memoized）值的引用是否会发生变化?

总结：

- 回调函数和 state 作为 prop 传入子组件时，这些组件可能会进行额外的重新渲染，props 尽量原子化， React.memo 在给定相同 props 的情况下渲染相同的结果，来提高组件的性能表现。
- Form 更新时会触发所有子组件的更新，所以表单里包含了一个很大的组件，例如 Table，会造成性能瓶颈。
- 对组件的 state 和 props 进行合理的划分
- 合理拆分组件