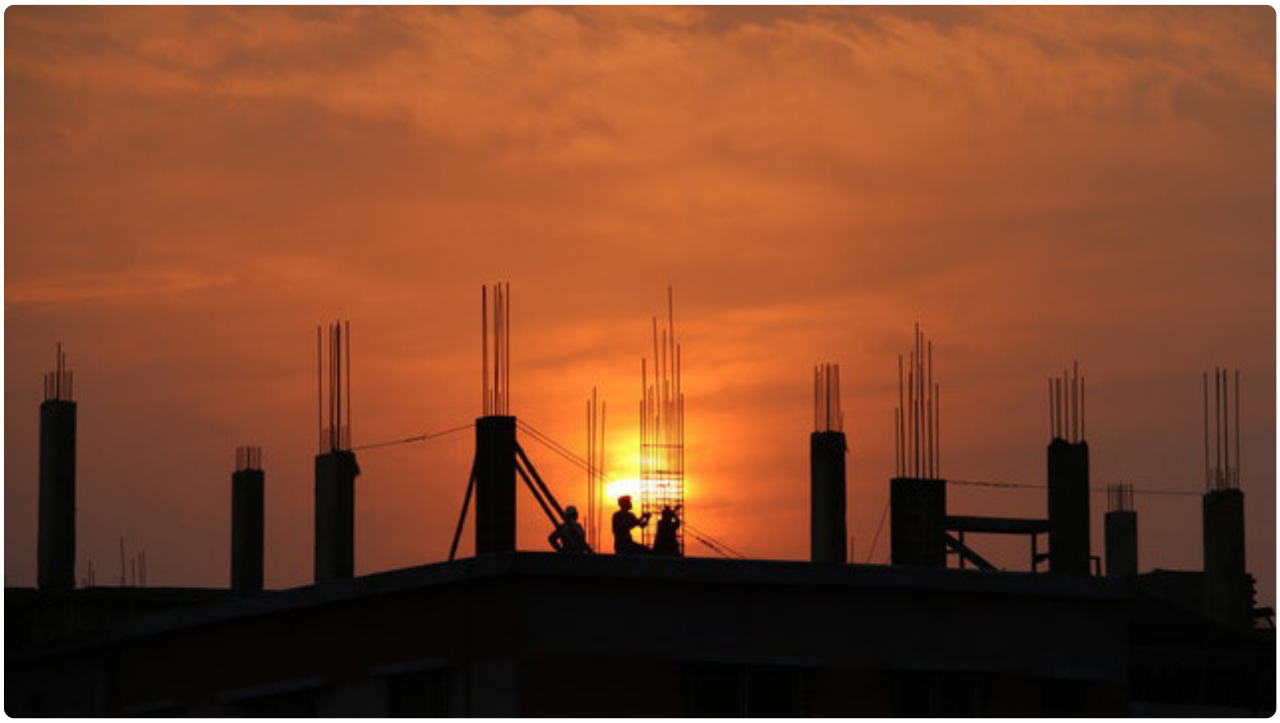
13 了解基于 React Fiber 架构的应用程序整体渲染流程

更新时间：2020-08-24 11:26:27



> 学习要注意到细处，不是粗枝大叶的，这样可以逐步学习、摸索，找到客观规律。 —— 徐特立

前言

基于 React Fiber 架构的应用程序运行过程可以分为 **prerender**，**render** 和 **commit** 三个阶段，其中只有应用程序首次渲染时才经历 **prerender** 阶段。**事实上，React 应用程序内部运行阶段的划分并没有特别明确的界定，运行阶段划分只是为了方便我们对应用程序的运行机制有个整体的认知。** 应用程序的首次渲染和更新渲染整体流程如图 3.3.1。

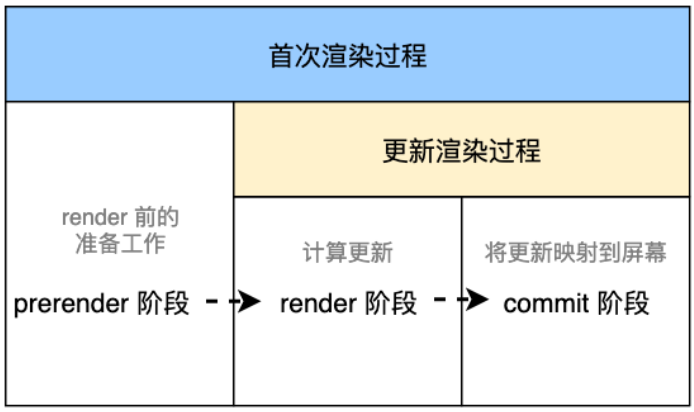


图 3.3.1 React应用程序首次渲染和更新渲染整体流程

prerender 阶段需要做好准备工作

该阶段的目标： 构建 fiberRoot 对象，做好 Fiber 架构的基建工作。

应用程序首次渲染时，也就是程序执行 ReactDOM.render(...) 时会经历 prerender 阶段。在该阶段 React 主要做一些（Fiber 架构的）**基建工作**。这些基建工作为后面 render 阶段的工作打好基础，该阶段的工作内容有：

- 检查容器（container）是否有效；
- 实例化 fiberRoot 对象，该对象是整个 Fiber 树的入口；
- 创建（整个应用程序的）更新队列（updateQueue）并添加到 fiberRoot 对象上。

prerender 阶段执行完成后，fiberRoot 对象的在内存中的结构如图 3.3.2。

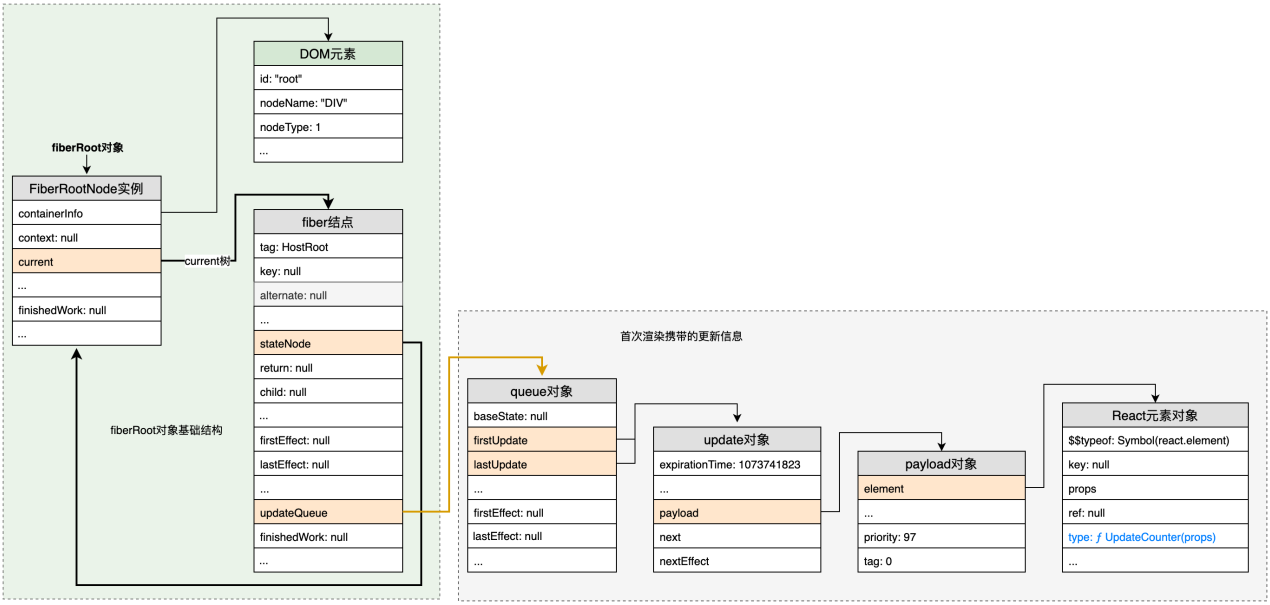


图 3.3.2 fiberRoot 对象在 prerender 结束时的内部结构

render 阶段主要工作是更新计算

该阶段目标： 确定需要在屏幕中更新的UI内容。

核心逻辑： 构建 workInProgress 对象树，收集副作用。

执行结果： 得到标记了副作用的 Fiber 结点树（一个链表，需要在 commit 阶段重点处理的信息）。

在 render 阶段，React 通过时间分片的方式来处理一个或多个 Fiber 结点的更新任务，每次更新 Fiber 结点时会先向调度器请求任务执行权，如果有更高优先级的任务（如动画）则等它们执行完成之后再执行自己的更新任务。这个工作方式如图 3.3.3 所示，中间每一个波谷代表深入某个分片的执行过程，每个波峰就是一个分片执行结束交还控制权的时机。

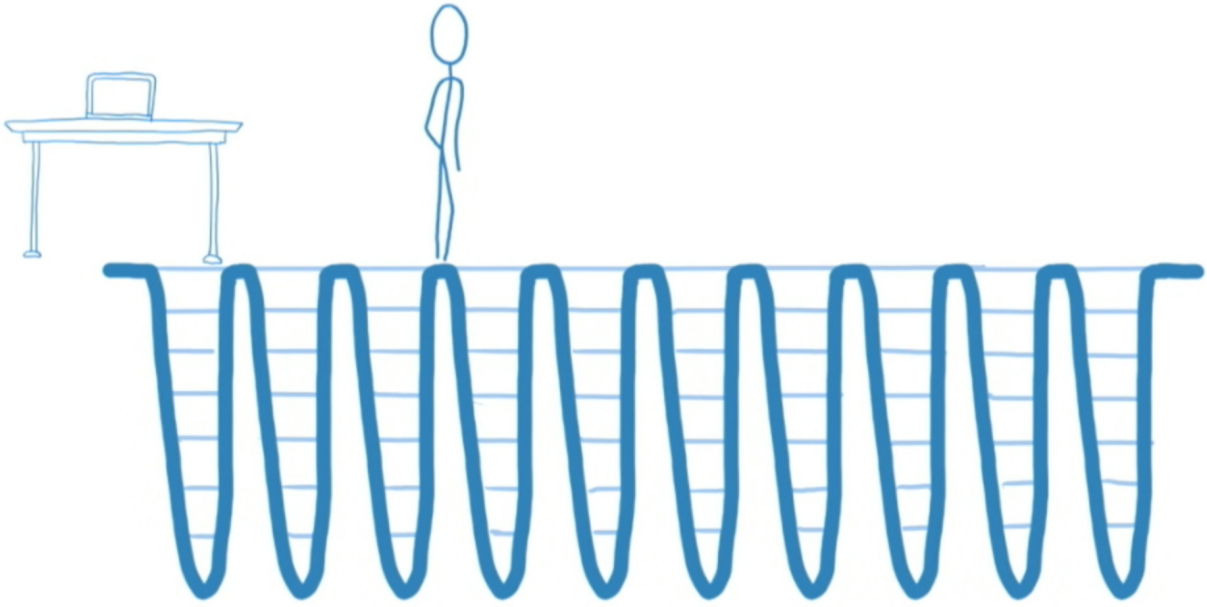


图 3.3.3 React Fiber 更新任务执行方式

得到任务执行权后，React 将每个 Fiber 结点作为最小工作单位，通过自顶向下逐个遍历 Fiber 结点，构建 workInProgress 树（一颗新的Fiber 树，更新的计算、调用部分生命周期函数等会在这个过程中完成）。这一过程总是从顶层的 HostRoot 结点开始遍历，直到找到未完成工作或者需要处理的结点。

render 阶段执行完成后，FiberRoot 对象上面的 current 属性指向了一颗「Fiber 树」，我们称它为 current 树，current 树上面的 alternate 属性指向了另一颗「Fiber 树」也就是后面要讲的 workInProgress 树。这两颗 Fiber 树通过 alternate 属性形成了一个闭环。

render 阶段执行完成后也会得到副作用列表。此时 fiberRoot 对象在内存中的结构如图 3.3.4。

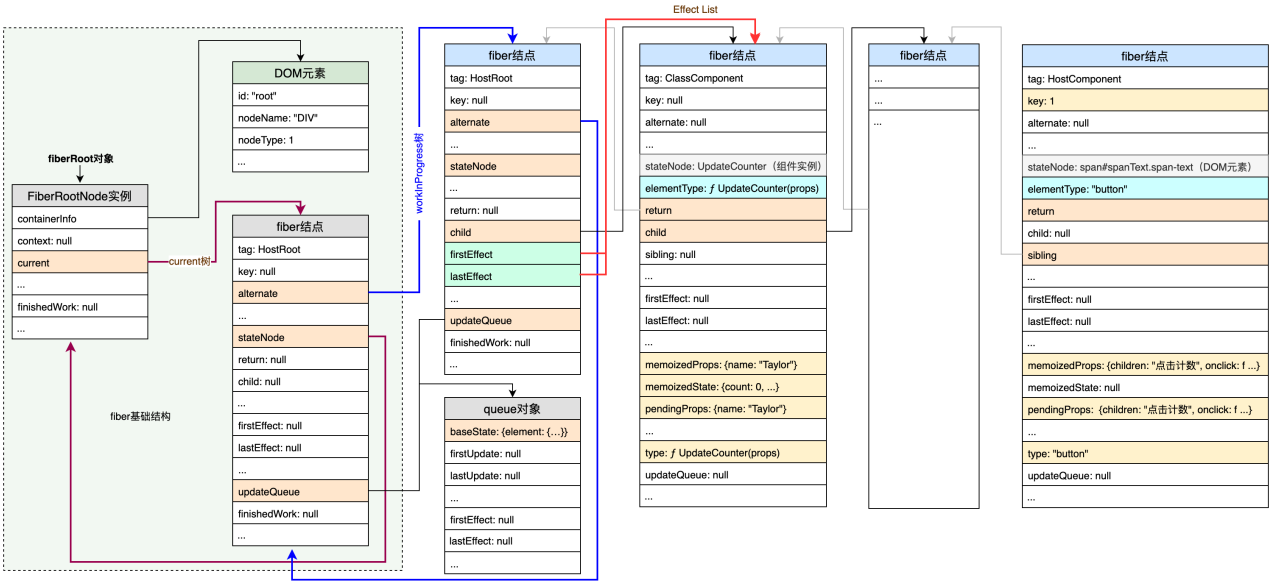


图 3.3.4 fiberRoot 对象在 render 结束时的内部结构

图 3.3.4 展示了应用程序首次渲染时在 render 阶段结束时 fiberRoot 对象的内部结构。此时的副作用列表（Effect List）就是整个 workInProgress 树。如果是更新渲染，那么副作用列表（Effect List）就会是 workInProgress 树中包括 HostRoot 结点在内的那些需要更新的 Fiber 结点集合（一般是 workInProgress 树的子集）。

commit 阶段负责将更新内容映射到屏幕

该阶段目标： 将 render 阶段得到的副作用列表中的更新信息渲染到屏幕。

执行逻辑： 通过遍历副作用列表根据副作用类型提交具体的副作用，包括 DOM 更新、调用生命周期函数、ref 更新等一系列用户可见的 UI 变化。

进入 commit 阶段时，fiberRoot 对象上面的 current 树反应 **当前** 屏幕上 UI 的状态，workInProgress 树反映 **未来** 需要映射到屏幕上 UI 的状态。副作用列表来描述需要实际做的操作，比如 DOM 的更新与增删，调用生命周期函数等等。

> 事实上，副作用列表是 workInProgress 树的子集，在后面文章中会详细介绍相关内容。

commit 阶段的工作会导致用户可见的变化，比如 DOM 更新。因此该过程不可中断，必须一直执行直到更新完成。

小结

为了方便大家对 React 应用程序运行时内部渲染流程，文章中将整个渲染流程分了三个阶段，分别是 prerender，render 和 commit 阶段。React 在每个阶段有着不同的工作内容，工作内容最开始准备基建工作到计算更新，最后将更新内容映射到屏幕。

看到这里，我们对 React Fiber 架构的工作流程已经有了整体认知。但是，我们还需要详细的了解 React Fiber 架构的两个重要构造函数 FiberRootNode 和 FiberNode，详细内容见下一节。

> **本文以及后续文章的所有对象结构图或者程序流程图均非 React 官方提供，这些图表仅供参考。图中的信息难免会有错误之处，请以程序执行结果为准。**

}

12 关于 React Fiber不得不说的事

14 深入理解React Fiber 架构的两个核心构造函数FiberRootNode和FiberNode