11 什么是 React Fiber

更新时间：2020-08-19 18:38:16

> 一个不注意小事情的人，永远不会成功大事业。——戴尔·卡耐基

前言

React Fiber 是 React 在 v16 版本中引入的架构，主要目标是解决应用程序的更新任务与（外部）其他任务（如动画渲染）在 CPU 资源分配方面的问题。

网上有很多介绍 React Fiber 相关的文章，如果是首次接触 React Fiber 的读者读完那些文章肯定对其理解的不是很到位，因为那些文章中对 React Fiber 的描述主要停留在抽象概念层面。事实上，大部分的抽象概念在实际应用中都有对应的实体。那么，React Fiber 在实际应用中到底是什么呢？

React Fiber 的本质

从概念上将 React Fiber 是一种程序架构，但是在 React 应用程序运行过程中它的实际体现是一个 JavaScript 对象，该对象主要是由两个构造函数的实例层层引用组成。这两个构造函数分别是 `FiberRootNode` 和 `FiberNode`。

> 在后面文章中将统一称 `FiberRootNode` 的实例为 `fiberRoot` 对象， `FiberNode` 的实例为 `Fiber` 对象或者 `Fiber` 结点

React 元素到 Fiber 结点的转换

在第二章中提到，React 元素是一种对象，它们描述了最终渲染到屏幕的页面结构。比如前面提到的 `UpdateCounter` 组件的元素树形结构是图 3.1.1 这样。

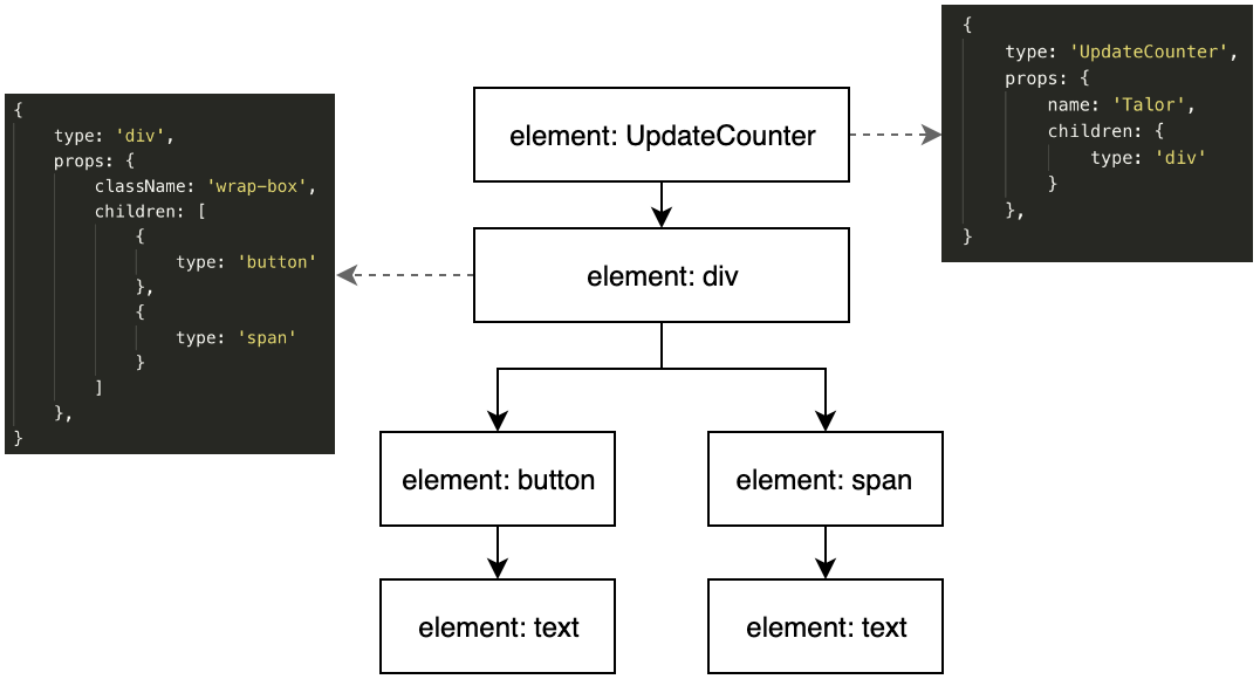

图 3.1.1 React 元素树形结构

React 对元素数据结构的定义中包含了 type 、 props 等属性，主要是用于描述应用程序的 UI 部分。因此，React 元素的数据结构并不能完成「创建更新」以及「将更新渲染到屏幕」等渲染工作，这部分工作必须由 Fiber 结点（具有更加丰富的数据结构）来完成。 UpdateCounter 组件的元素树最终会转换为 Fiber 树，见图 3.1.2。

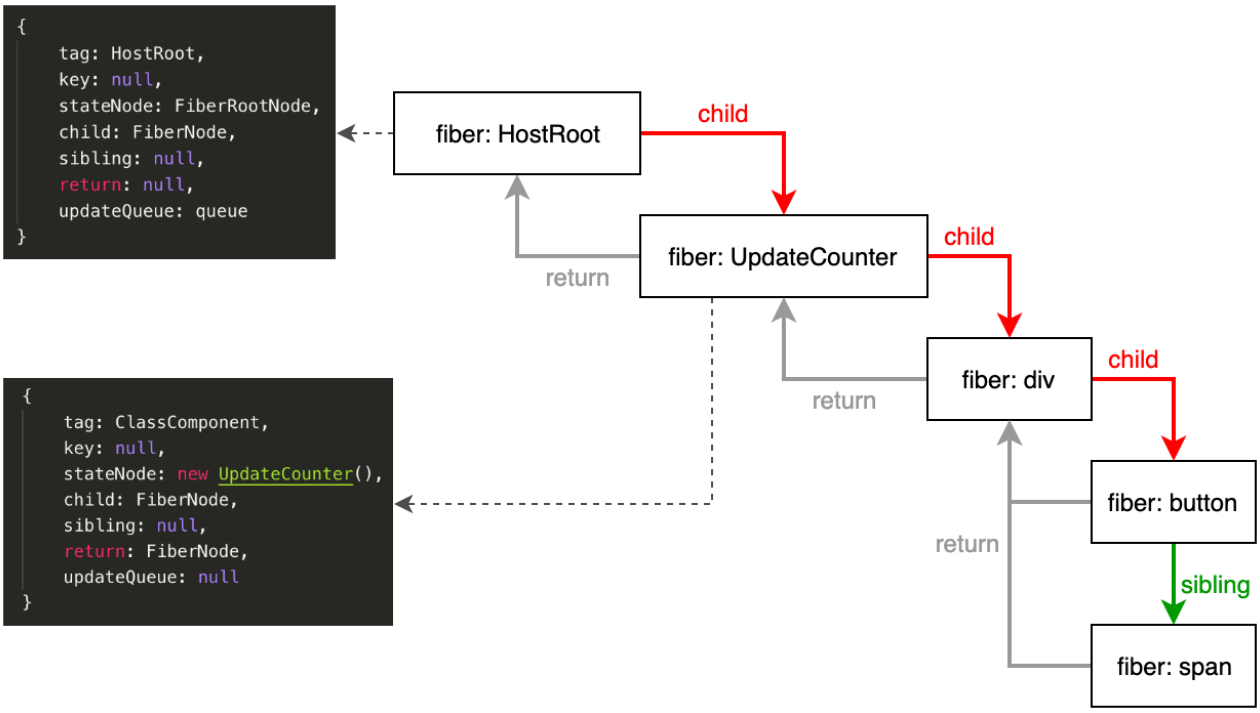

图 3.1.2 React 应用程序运行时 Fiber 树形结构

React 应用程序的 Fiber 树相对于元素树携带的信息更为丰富，它包含了 stateNode 、 child 、 sibling 、 return 以及 updateQueue 等属性，每个 Fiber 结点通过 child ， sibling ， return 分别指向了孩子结点，兄弟节点和父结点，最终形成了一个闭环（也是一种树形结构）。React 可以在这个 Fiber 树上面轻松找到任意一个需要更新的结点，然后对其进行更新处理。

FiberRootNode 和 FiberNode 的实例组合成 Fiber 架构的实体

图 3.1.2 展示的 Fiber 树上的每一个结点均是 FiberNode 构造函数的实例，前面我们说到，React Fiber 架构的实体由两个构造函数的实例构成，现在我们为图 3.1.2 加上 FiberRootNode 构造函数的实例，见图 3.1.3。

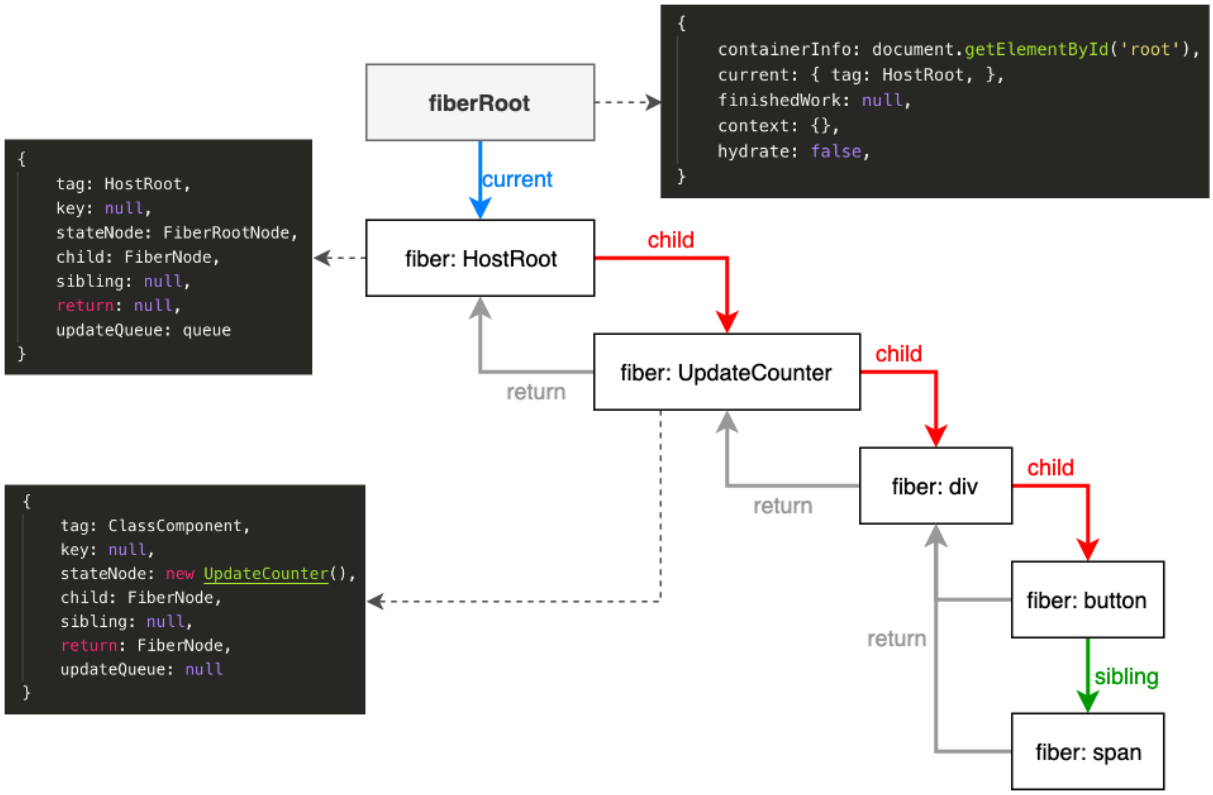

图 3.1.3 React Fiber 架构实体

fiberRoot 对象是整个 Fiber 架构的入口对象，在应用程序的更新过程中，React 都会以这个对象为根基，查找到对应的 Fiber 结点，调用生命周期函数以及标记对应的 effectTag 等。

> 注：上面图中描述的 FiberRootNode 和 FiberNode 这两个构造函数的实例对象均为简化版，这两个构造函数内部属性会在本章第四节给出详细介绍。

小结

现在我们知道了，React Fiber 架构在应用程序运行中的体现就是如图 3.1.3 一样的庞大对象，React 可以很方便的找到这个庞大对象上面的任何一个结点。那么，我们来思考一个问题就是，React 团队为什么要引入 Fiber 架构呢？请看下一节。

}

10 React 如何定义更新队列以及它们之间的相互作用关系？

12 关于 React Fiber不得不说的事