博学谷
www.boxuegu.com

第二章 React原理剖析

一样的在线教育，不一样的教学品质

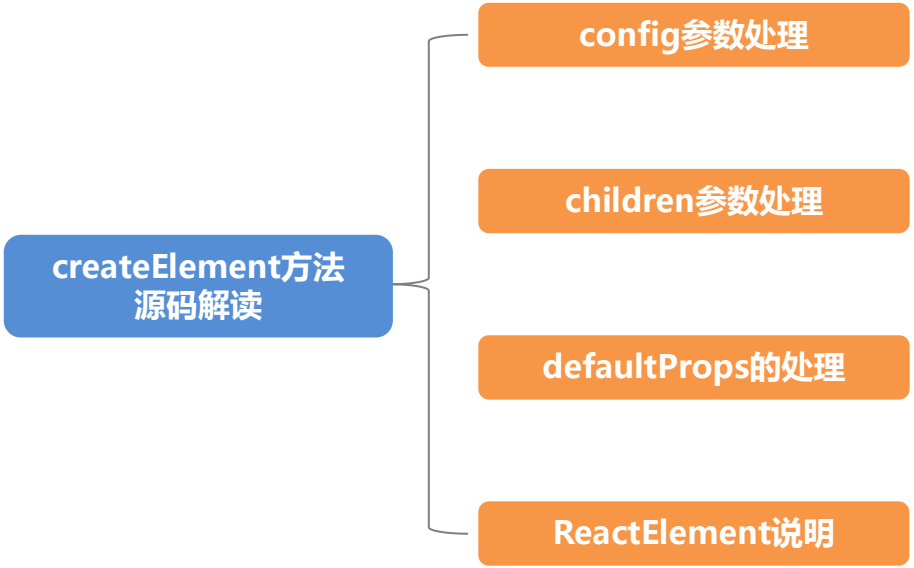
目录 Contents

1. createElement方法原理

模拟实现createElement方法

createElement( )方法的**作用**：创建对应的元素，而所创建的元素其实就是虚拟DOM.

2. createElement方法源码解读



目录 Contents

- createElement方法原理
- Component源码分析
- ReactDom.render源码分析
- 任务调度介绍
- 虚拟DOM和Diff算法

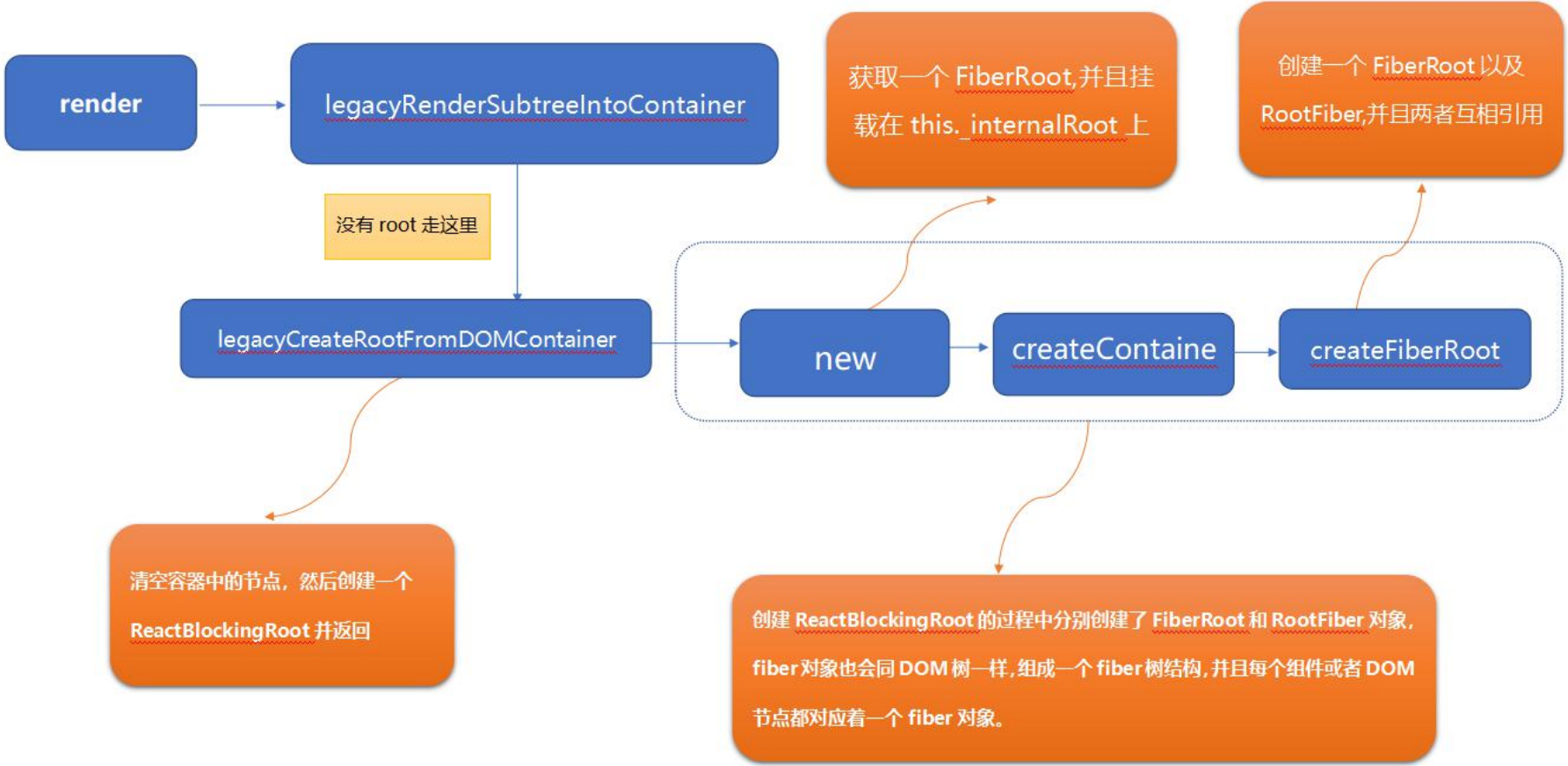
1. Component源码分析

Component是组件的意思，在创建类组件的时候，都需要继承React.Component.

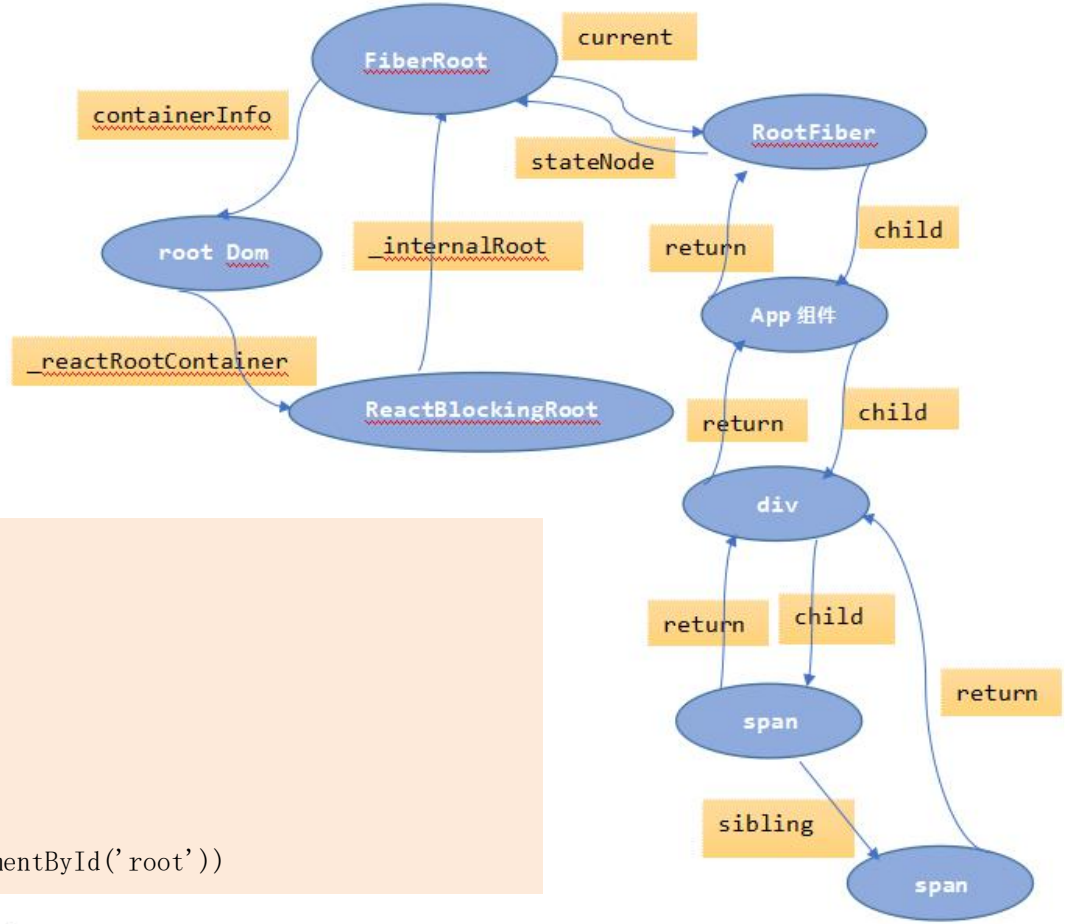
```
function Component(props, context, updater) {
  this.props = props;
  this.context = context;
  // If a component has string refs, we will assign a different object later.
  this.refs = emptyObject;
  // We initialize the default updater but the real one gets injected by the
  // renderer.
  this.updater = updater || ReactNoopUpdateQueue;
}
```

目录 Contents

1. ReactDom.render源码分析



2. fiber树结构的说明




```
const APP = () => (
            <div>
                  <span></span>
                  <span></span>
            </div>
    )
ReactDom.render(<APP/>, document.getElementById('root'))
```

目录 Contents

1.任务调度介绍

微软最开始的操作系统是DOS,这个系统我们称之为‘单任务操作系统’，也就是这个操作系统同一个时间只允许运行一个程序。在这种系统中，如果你想执行多个任务，只能等待前一个进程（程序）退出，然后再去执行另一个进程。

现在的操作系统都是多任务的操作系统，操作系统会按照一定的调度策略，将CPU的执行权分配给多个进程， 多个进程都有被执行的机会，让它们交替执行，形成一种“同时在运行”假象, 因为CPU速度太快，人类根本感觉不到 。

常见的任务调度策略

(1) 先到先得

这是最简单的调度策略, 简单说就是没有调度。谁先来谁就先执行，执行完毕后就执行下一个 .

(2)轮转

这是一种基于时钟的抢占策略，这也是抢占策略中最简单的一种: 公平地给每一个进程一定的执行时间，当时间消耗完毕或阻塞，操作系统就会调度其他进程，将执行权抢占过来。

1.任务调度介绍

React 16版本之前的不足

调和阶段(Reconciler)：React 会自顶向下通过递归，遍历新数据生成新的 Virtual DOM，然后通过 Diff 算法，找到需要变更的元素(Patch)，放到更新队列里面去。

渲染阶段(Renderer)： 遍历更新队列，通过调用DOM的API,实际更新渲染对应元素。

在React16版本之前的问题：

在调和阶段，由于是采用递归的遍历方式， 这种也被成为 Stack Reconciler 。这种方式有一个特点：

一旦任务开始进行，就无法中断，那么 js 将一直占用主线程， 一直要等到整棵 Virtual DOM 树计算完成之后，才能把执行权交给渲染引擎，那么这就会导致一些用户交互、动画等任务无法立即得到处理，出现卡顿，非常影响用户体验。

1.任务调度介绍

什么是React Fiber?

React Fiber 可以让自己的调和阶段的这个过程变成可被中断， ‘适时’ 地让出cpu执行权，这样除了可以让浏览器及时地响应用户的交互，这种分批延时对DOM进行操作，可以得到更好的用户体验。所以说，React Fiber是一种调度算法，这个调度算法又称之为 Fiber reconciler 。

React Fiber的思想就是让React渲染的过程可以被中断，可以将控制权交回浏览器，让位给高优先级的任务，浏览器空闲后再恢复渲染。

目录 Contents

1.虚拟DOM原理

虚拟DOM本质上就是一个JS对象，用来描述你希望在屏幕上看到的内容。

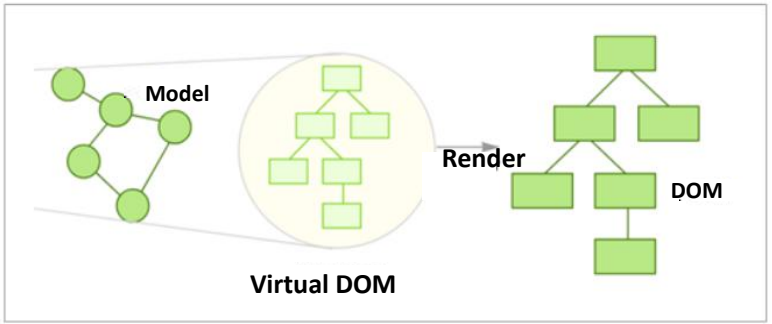
操作虚拟DOM的性能比操作真实DOM的性能高

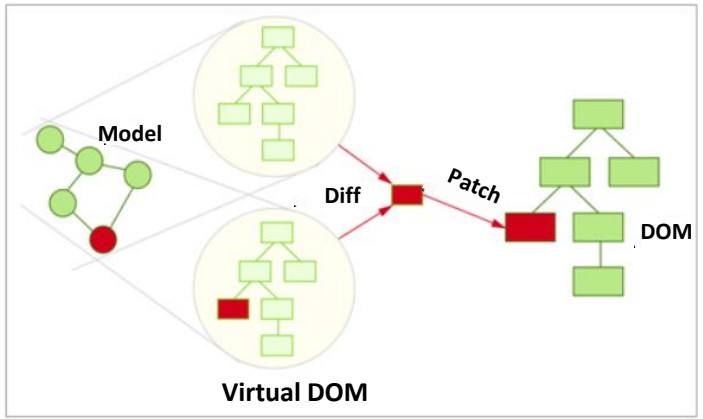
```
var A=document.getElementById('test');

A.style.backgroundColor = "black";

A.style.backgroundColor = "red";

A.style.backgroundColor = "black";
```

2. Diff算法原理

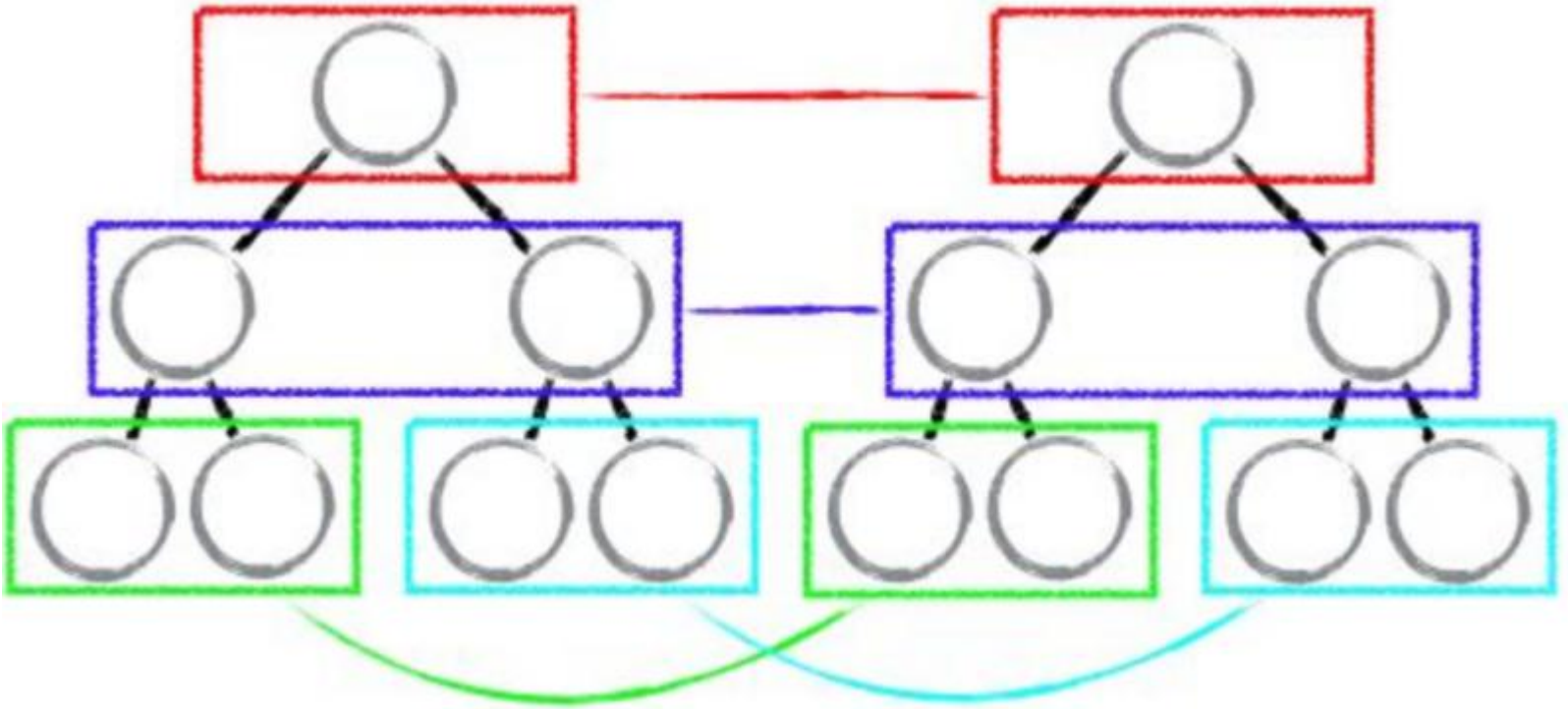
diff算法的执行过程

- 初次渲染时，React会根据初始化的state（model），创建一个虚拟DOM对象（树）
- 根据虚拟DOM生成真正的DOM，渲染到页面
- 当数据变化后(setState())，会重新根据新的数据，创建新的虚拟DOM对象（树）
- 与上一次得到的虚拟DOM对象，使用Diff算法比对（找不同），得到需要更新的内容
- 最终，React只将变化的内容更新（patch）到DOM中，重新渲染到页面





2. Diff算法原理



2. Diff算法原理

一样的在线教育，不一样的教学品质