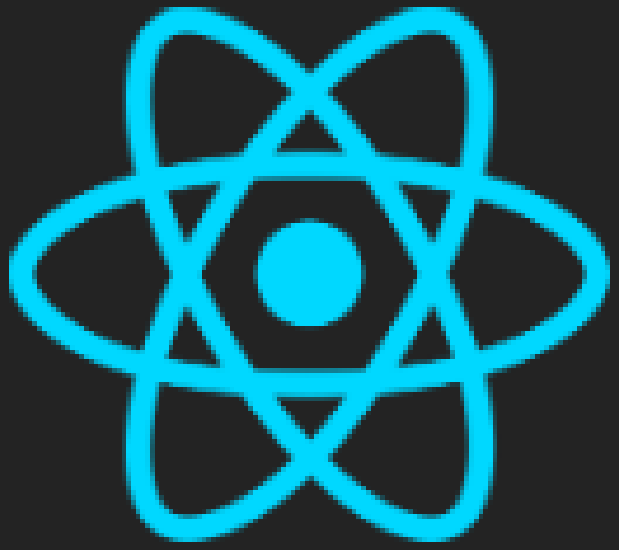
react-dom-diff



speaker:stanny

Aim

- 理解 virtual dom 工作原理
- 理解 key 属性的作用

Question

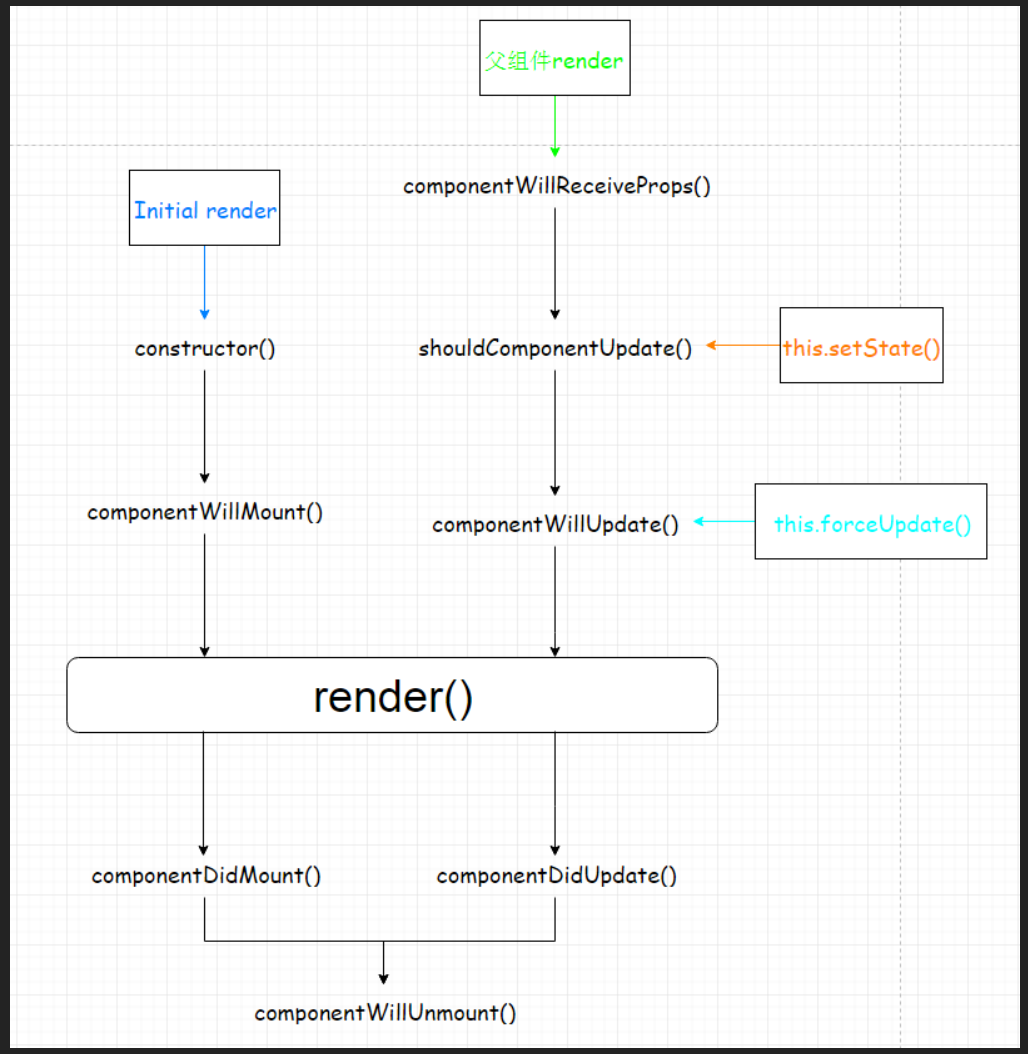
- code1 VS code2

```
<div className="wrapper">
    <div className="A"
    style={{ display: `${shouldShow === true ? 'block' : 'none'}` }}>
        <div className="C">
            <div className="D"></div>
        </div>
    </div>
    <div className="B"></div>
</div>
```

```
11  <div className="wrapper">
12      {shouldShow ?
13          <div className="A">
14              <div className="C">
15                  <div className="D"></div>
16              </div>
17          </div>
18          : ''}
19      <div className="B"></div>
20  </div>
```

- 这两段代码对于 react 来言有何不同
- 😄

回顾 · 组件生命周期 & 何时重新渲染



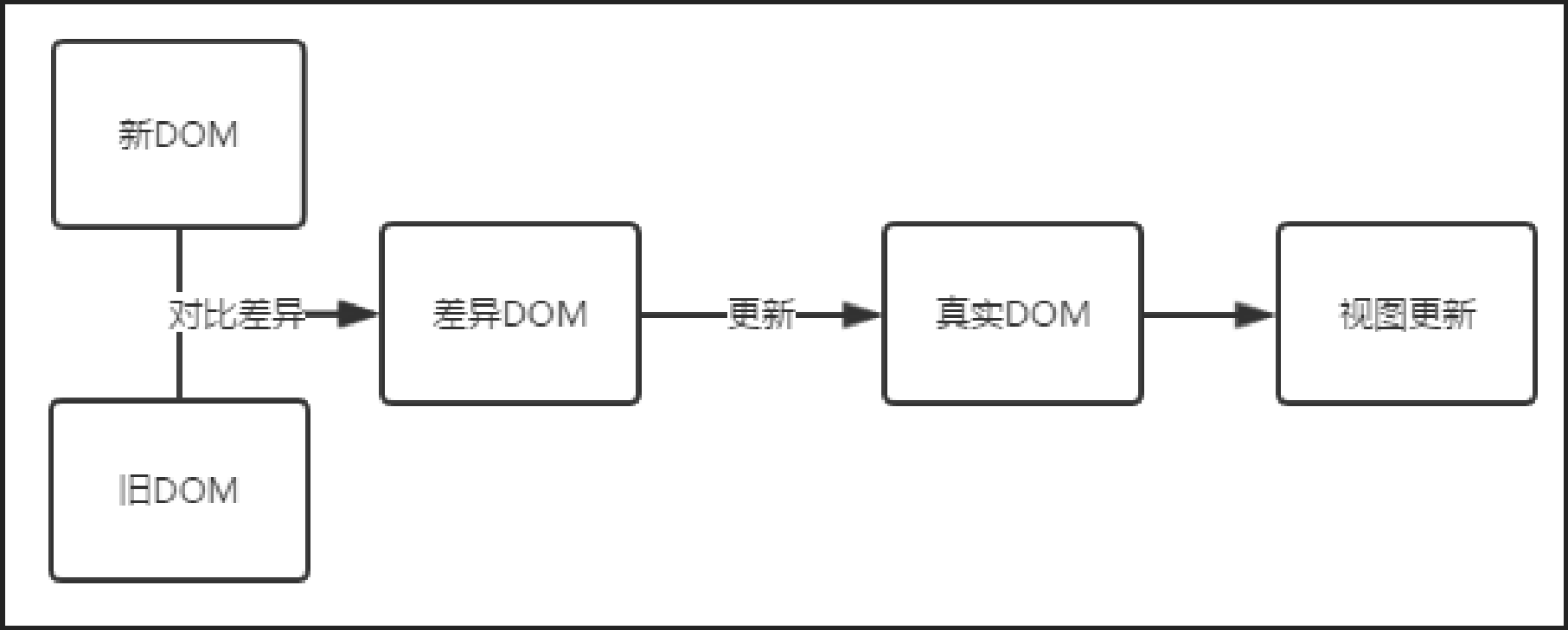
Virtual DOM

- 我的理解就是没有插入真实的 DOM 的节点

```
var div = document.createElement('div')
...
// document.body.appendChild(div)
```

- JSX 的运行基础
- react 优秀性能的关键

Virtual DOM 工作机制



- react 组件内部维护了一套虚拟 dom 的状态，状态变化的时候，产生新的虚拟 dom ，对比新旧两套虚拟，计算 diff，然后 只是把 diff 的部分用一种高效的方式更新到了真实 dom

diff

找两棵树的不同，找到最小的转换步骤

传统 diff 算法

- 循环递归对节点进行依次对比，效率低下
- 时间复杂度 O(n^3)

react-diff

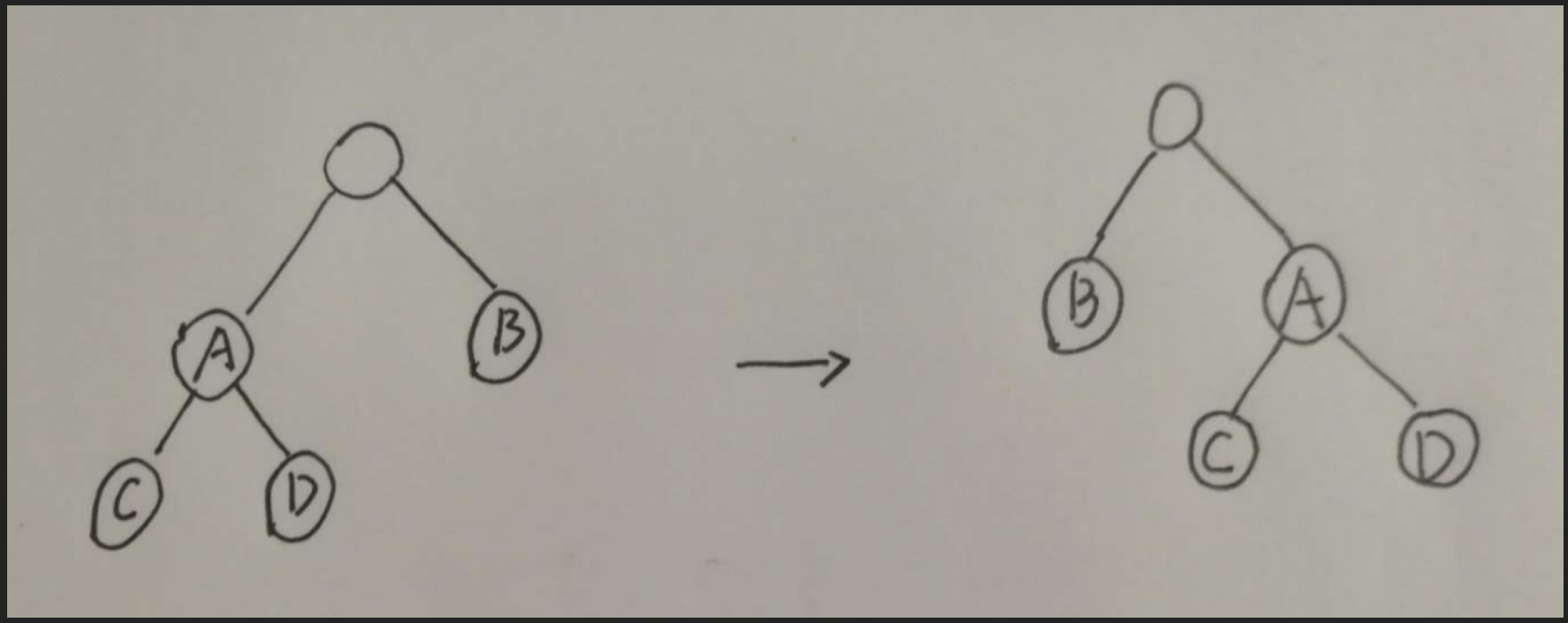
- 时间复杂度优化至 O(n)
- shouldComponentUpdate() 返回 true 后准备调用 diff

react-diff 工作原理

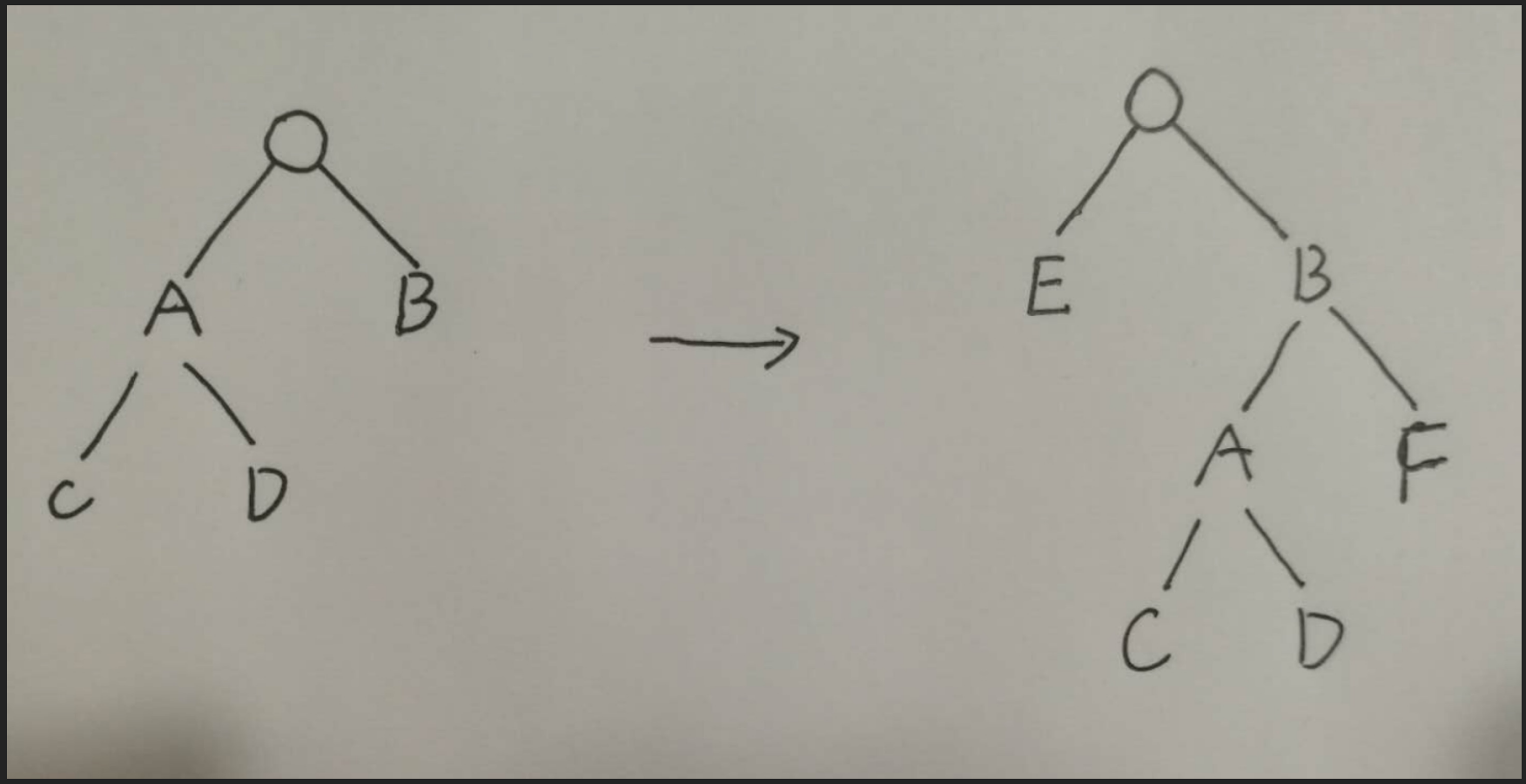
- 广度优先遍历
- 当发现节点已经不存在，直接删除，不会再递归对比
- 兄弟节点调换顺序依赖唯一 key 属性

画个图解释一下过程

- 类型相同的兄弟节点交换顺序



- 节点跨层移动



- 节点跨层移动效率低?

react-diff 的两个假设

1. 组件 DOM 结构相对稳定，很少出现跨层移动的现象（UI 的特点）
2. 同一层级的一组子节点有唯一 key 属性

空说无凭

[跑一段代码 看实例]

小结

1. 算法复杂度为 O(n)
2. 虚拟 DOM 如何计算 diff
3. key 属性的作用
4. 保持稳定的 DOM 有利于性能提升