React源码中几个重要的对象

- **ReactElement:**

```
|- ReactElement
        |- type > - 取值为字符串(div、span等)：DOM元素
                > - 取值为组件构造函数：React组件
        |- props：组件或者DOM元素上的属性
            |- children 子ReactElement数组
        |- key：元素标识，用于优化更新
```

- **ReactClass:**

```
|- ReactClass
        |- 生命周期方法：componentWillMount
                        componentDidMount
                        componentWillUnmount
                        componentWillReceiveProps
                        componentWillUpdate
                        componentDidUpdate
                        shouldComponentUpdate
        |- render方法：生成ReactElement
        |- _reactInternalInstance: 对应的ReactComponent实例
```

- **ReactComponent:**

```
|- ReactComponent > ReactDOMTextComponent
                    ReactDOMEmptyComponent
                    **ReactDOMComponent**
                            | - _renderedChildren: 子元素的ReactComponent数组
                    **ReactCompositeComponent**
                            |- _instance: 对应的ReactClass实例
                            |- _currentElement: 当前元素对应的ReactElement
                            |- _renderedComponent：render出的元素的ReactComponent实例
        |-react-id：在DOM树中的唯一标识
        |-_mountIndex: 用来标识是父组件的第几个子组件
        |-mountComponent(id): 挂载时调用 > ReactCompositeComponent: 调用_instance挂载阶段的生命周期方法
                                                                    调用_renderedComponent的mountComponent方法
                                        > ReactDOMComponent: 递归调用子元素的mountComponent方法，生成标识为react-id
        |-receiveComponent(nextReactElement): 更新时调用 > ReactCompositeComponent：调用_instance更新阶段的生命周期
                                                                                    看更新前后render出的ReactElemen
                                                                                        - 如果相同：调用render出的
                                                                                        - 如果不相同：直接卸载原节
                                                                > ReactDOMComponent: 更新元素上的属性
                                                                                    diff和patch
```

react-id和_mountIndex

react-id是一个**html元素**在DOM树中的唯一标识，可以明确标明该html元素在dom树中的位置
example:

```
<div data-react-id='1'>
        <div data-react-id='1-1'>
                <span data-react-id='1-1-1'>sample1<span>
                <span data-react-id='1-1-2'>sample2</span>
        </div>
        <div data-react-id='1-2'>
                <img data-react-id='1-2-1' src='/sample.png'/>
        </div>
</div>
```

每个ReactComponent都有一个react-id属性，用来标记该ReactComponent生成的HTML Markup, 在执行patch更新DOM时，React就是通过这个react-id属性来选中DOM节点
子元素的react-id可以通过父元素的react-id拼接_mountIndex来构造

diff和patch

- diff的核心就是比较相同的key的Component的type和_mountIndex在更新前后的差异，然后把更新操作所需要的信息记录在diffQueue中
- patch的核心就是从diffQueue中一个一个取出更新信息对象，执行实际的更新DOM的动作
- React高效的原因在于diff这个过程是纯js操作，一次性组装出所有的更新信息对象，patch的时候再一次性执行所有DOM更新操作

组装出的更新信息对象的结构：

```
const UPDATE_TYPES = {
        MOVE_EXISTING: 1,
        REMOVE_NODE: 2,
        INSERT_MARKUP: 3
}

type DiffQueueItem = {
        parentId: string // 父节点的react-id, 例如 '1-2'
        parentNode: DOMNode // 父DOM元素，例如document.querySelector(`[data-react-id=${parentId}]
        type: number, // 更新类型：取值是UPDATE_TYPES下的枚举值
        fromIndex: number, // 节点更新前位置信息（旧_mountIndex）
        toIndex: numer, // 节点更新后的位置信息（新_mountIndex）
        markUp?: HTMLElement // 对于更新类型是INSERT_MARKUP的，该字段用来标识要添加的HTML Markup

}
```

diff的过程：

1. **flattenChildren**：将_renderdChildren数组（上一次渲染的子元素ReactComponent数组）转为键为key，值为ReactComponent的一个Map（prevChildrenComponent）
2. **generateComponentChildren**：根据prevChildrenComponent和nextChildrenElements，生成新的key-ReactComponent的Map（nextChildrenComponent）
   - 如果更新前后的childrenReactElements中，同key的ReactElement的type也相同，则nextChildrenComponent中的ReactComponent复用prevChildrenComponent中的，并调用该ReactComponent的receiveComponent方法执行更新
   - 否则，生成一个新的ReactComponent
3. 组装更新对象，推入**diffQueue**：比较prevChildrenCompoennt和nextChildrenComponent：
   - **prev和next两个map中，key相同的，值也相同**：组装MOVE_EXISTING（移动现有节点）变更对象推入diffQueue
   - **prev和next两个map中，key相同的，值不相同**：组装REMOVE_NODE（移除旧节点）和INSERT_MARKUP(挂载新节点) 两个变更对象推入diffQueue
   - **prev和next两个map中，next中有key，prev中没有**：组装INSERT_MARKUP变更对象推入diffQueue
   - **prev和next两个map中，prev中有key，next中没有**：
     组装REMOVE_NODE变更对象推入diffQueue

patch过程：
从diffQueue中取出所有变更对象，根据type不同，执行不同的DOM操作

思考

- **Q:** 所谓挂载是指用DOM操作把组件一个一个挂到DOM树上吗？
  A: 所谓挂载这个动作，并不是把组件一个一个地挂载到DOM上，组件的挂载阶段只是负责生成HTML标记而已，用DOM操作挂载组件其实只在调用ReactDOM.render时触发了一次，即把根组件的HTML标记挂载到DOM上
- **Q:** 为什么在render出的元素中有数组时，要加一个key?
  A: key用来在 diff 操作的第二步 generateComponentChildren 时，判断要不要复用上一次的ReactComponent，如果复用了的话，在实际更新DOM时是移动现有DOM节点，否则，会卸载当前DOM节点，挂载新的DOM节点，造成冗余的DOM操作
- **Q:** React的diff算法复杂度为什么是O(n)？
  A: 因为diff只对比同一层的节点（同一个父节点下的子节点）的差异
- **Q:** key和react-id分别用来干嘛？
  A: key标识同一个父节点下的唯一的一个ReactElement，react-id标识整个DOM树中的唯一一个DOM元素