15 理解 React Fiber 架构中中的 effectTag 与位运算

更新时间：2020-08-28 10:25:48

> 老骥伏枥，志在千里； 烈士暮年，壮心不已。 ——曹操

前言

上一节提到，Fiber 结点中 `effectTag` 属性想当重要，它用于标识结点的更新类型（如插入、修改或者删除）。本节将详细介绍 `effectTag` 属性的取值以及其位运算。

effectTag 可以取哪些值

`effectTag` 是构造函数 `FiberNode` 的一个属性，它用于标识当前 Fiber 结点的「更新」类型，该属性的取值见代码示例 3.5.1。

```
// 源码位置：packages/shared/ReactSideEffectTags.js
export const NoEffect = /*              */ 0b000000000000;
export const PerformedWork = /*         */ 0b000000000001;

export const Placement = /*             */ 0b000000000010;
export const Update = /*                */ 0b000000000100;
export const PlacementAndUpdate = /*    */ 0b000000000110;
export const Deletion = /*              */ 0b000000001000;
export const ContentReset = /*          */ 0b000000010000;
export const Callback = /*              */ 0b000000100000;
export const DidCapture = /*            */ 0b000001000000;
export const Ref = /*                   */ 0b000010000000;
export const Snapshot = /*              */ 0b000100000000;
export const Passive = /*               */ 0b001000000000;

export const LifecycleEffectMask = /*   */ 0b001110100100;

export const HostEffectMask = /*        */ 0b001111111111;
```

代码示例 3.5.1 effectTag 的取值范围

NoEffect 一般作为 EffectTag 的初始值，或者用于 EffectTag 的比较判断，其值为 0 表示没有副作用，也就是不涉及更新。

PerformedWork 由 React devtools 读取， NoEffect 和 PerformedWork 都不会被 commit，当创建 Effect List（后面会介绍）时，会跳过 NoEffect 和 PerformedWork 。

Placement 表示向树中插入新的子节点，对应的状态为 MOUNTING，当执行 commitPlacement 函数完成插入后，清除该标志位。

Update 表示当 props 、 state 、 context 发生变化或者 forceUpdate 时，会标记为 Update ，检查到标记后，执行 commitUpdate 函数进行属性更新，与其相关的生命周期函数为 componentDidMount 和 componentDidUpdate 。

Deletion 标记将要卸载的结点，检查到标记后，执行 commitDeletion 函数对组件进行卸载，在节点树中删除对应对节点，与其相关的生命周期函数为 componentWillUnmount 。

为什么使用数字二进制表示 EffectTag 的值

我们知道数字的二进制表示主要由 0 和 1 组成，在逻辑判断中数字 1 和数字 0 一般表示相反的描述，比如 1 表示「正确」那么 0 就表示「错误」。此外，我们也可以用 1 和 0 表示「有」和「没有」。

仔细观察一下 EffectTag 的取值，大部分的变量里面只有一个 1 ，其他的都是 0 。这说明了什么呢？

Placement 的值为 0b000000000010 ，其中的 1 在右边第二位。 Update 的值为 0b000000000100 ，其中的 1 在右边第三位。而 PlacementAndUpdate 的值为 0b000000000110 ，它有两个 1 分别在右边第二位和第三位。这时我们就发现了一个规律，如果数值中只有一个 1 ，说明该数值只代表「单一操作」，比如 Placement 和 Update 分别代表的是「插入」和「删除」。如果数值中有多个 1 ，说明该数值代表「复合操作」，比如 PlacementAndUpdate 代表的是既要「插入」也要「更新」。

因此，我们可以将 **EffectTag 的数值二进制表示中不同位置的 1 就理解为不同的「操作」**。

如何理解 React 中的位运算

React 在很多地方用数字二进制来表示一些变量的值（如上面的 EffectTag），这样做有什么好处呢？答案是方便做位运算，见代码示例 3.5.2。

```
workInProgress.effectTag |= PerformedWork
// 等同于
workInProgress.effectTag = workInProgress.effectTag | PerformedWork
```

代码示例 3.5.2

在 JavaScript 中 `|` 和 `&` 两个运算符分别代表的是按位或和按位与，这两个运算符都是按照数值的二进制来进行运算的。`|` 的运算法则是两位其中一个为 `1`，结果为 `1`，否则为 `1`。`&` 的运算法则是两位同时为 `1`，结果才为 `1`，否则为 `0`。

那么，React 中的位运算（如代码示例 3.5.2）该怎么理解呢？

```
var effectTag = Update

effectTag |= Ref // effectTag此时为0b0000010000100

console.log((effectTag & Update) === Update) // true
console.log((effectTag & Ref) === Ref) // true
console.log((effectTag & Placement) === Placement) // false
effectTag &= ~Ref
console.log((effectTag & Ref) === Ref) // false
```

代码示例 3.5.3

结合代码示例 3.5.1 和代码示例 3.5.3 可以发现 `(effectTag & Update) === Update` 和 `(effectTag & Ref) === Ref` 的返回值为 `true`，`(effectTag & Placement) === Placement` 的返回值为 `false`，这是因为 `effectTag` 数值的二进制串中 `Update` 和 `Ref` 的对应位值均为 `1`，而 `Placement` 对应位的值为 `0`。因此，我们可以对代码示例 3.5.3 中 `&` 运算符的实际意义理解成：`&` 操作可以用来判断某个变量中是否含有某个属性。

> 注：这里可能需要大家稍微思考一些哦！

小结

本章主要介绍的是 React Fiber 架构的基础理论。React 在应用程序运行时在这个架构上面做的一些工作，如更新计算以及将更新映射到屏幕将会在第五章中详细介绍。下一章，我们会介绍 React 世界中的任务调度基础。

}

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

16 如何理解 React 中的更新与任务