Sunday • 更新于 2022-10-19

上一节 03：框架实现：... 05：框架实现：... 下一节

04：框架实现：computed 的响应性：初见调度器，处理脏的状态

根据之前的代码可知，如果我们想要实现 **响应性**，那么必须具备两个条件：

1. 收集依赖：该操作我们目前已经在 `get value` 中进行。
2. 触发依赖：该操作我们目前尚未完成，而这个也是我们本小节主要需要做的事情。

那么根据第二小节的源码可知，这部分代码是写在 `ReactiveEffect` 第二个参数上的：

代码块

```
1    () => {
2                if (!this._dirty) {
3                    this._dirty = true
4                    triggerRefValue(this)
5                }
6            }
```

这个参数是一个匿名函数，被叫做 `scheduler` 调度器。

该匿名函数中，又涉及到了一个 `_dirty` 变量，该变量我们把它叫做 **脏**。

那么想要实现 `computed` 的响应性，就必须要搞明白这两个东西的概念：

调度器

调度器 `scheduler` 是一个相对比较复杂的概念，它在 `computed` 和 `watch` 中都有涉及，但是在当前的 `computed` 实现中，它的作用还算比较清晰。

所以根据我们秉承的：**没有使用就当做不存在** 的理念，我们只需要搞清楚，它在当前的作用即可。

根据我们在第二小节的源码阅读，我们可以知道，此时的 `scheduler` 就相当于一个 **回调函数。**

在 `triggerEffect` 只要 `effect` 存在 `scheduler`，则就会执行该函数。

_dirty 脏

对于 `dirty` 而言，相对比较简单了。

它只是一个变量，我们只需要知道：**它为 false 时，表示需要触发依赖。为 true 时表示需要重新执行 run 方法，获取数据。** 即可。

实现

那么明确好了以上两个概念之后，接下来我们就来进行下 `computed` 的响应性实现：

1. 在 `packages/reactivity/src/computed.ts` 中，处理脏状态和 `scheduler`：

代码块

```
1    export class ComputedRefImpl<T> {
2        ...
3
4        /**
5         * 脏：为 false 时，表示需要触发依赖。为 true 时表示需要重新执行 run 方法，获取数据。即
6         */
```

```
    constructor(getter) {
        this.effect = new ReactiveEffect(getter, () => {
            // 判断当前脏的状态，如果为 false，表示需要《触发依赖》
            if (!this._dirty) {
                // 将脏置为 true，表示
                this._dirty = true
                triggerRefValue(this)
            }
        })
        this.effect.computed = this
    }

    get value() {
        // 触发依赖
        trackRefValue(this)
        // 判断当前脏的状态，如果为 true ，则表示需要重新执行 run，获取最新数据
        if (this._dirty) {
            this._dirty = false
            // 执行 run 函数
            this._value = this.effect.run()!
        }

        // 返回计算之后的真实值
        return this._value
    }
}
```

2. 在 `packages/reactivity/src/effect.ts` 中，添加 `scheduler` 的处理：

代码块

```
export type EffectScheduler = (...args: any[]) => any


/**
 * 响应性触发依赖时的执行类
 */
export class ReactiveEffect<T = any> {
    /**
     * 存在该属性，则表示当前的 effect 为计算属性的 effect
     */
    computed?: ComputedRefImpl<T>

    constructor(
        public fn: () => T,
        public scheduler: EffectScheduler | null = null
    ) {}
    ...
}
```

3. 最后不要忘记，触发调度器函数

代码块

```
/**
 * 触发指定依赖
 */
export function triggerEffect(effect: ReactiveEffect) {
  // 存在调度器就执行调度函数
  if (effect.scheduler) {
    effect.scheduler()
  }
  // 否则直接执行 run 函数即可
  else {
    effect.run()
  }
}
```

此时，重新执行测试实例，则发现 `computed` 以具备响应性。