全部开发者教程

07：框架实现：构建 effect 函数，生成 ReactiveEffect 实例

08：框架实现：track && trigger

09：框架实现：构建 track 依赖收集函数

10：框架实现：构建 trigger 触发依赖

11：总结：单一依赖的 reactive

12：功能升级：响应数据对应多个 effect

13：框架实现：构建 Dep 模块，处理一对多的依赖关系

14：reactive 函数的局限性

15：总结

第六章：响应系统 - ref 的响应性

01：前言

Sunday • 更新于 2022-10-19

上一节 12：功能升级：... 14：reactive 函... 下一节

13：框架实现：构建 Dep 模块，处理一对多的依赖关系

想要处理 `dep` 模块，那么我们需要对 `track` 和 `trigger` 进行改造：

track

1. 创建 `packages/reactivity/src/dep.ts` 模块：

代码块

```ts
import { ReactiveEffect } from './effect'

export type Dep = Set<ReactiveEffect>

 /**
  * 依据 effects 生成 dep 实例
  */
export const createDep = (effects?: ReactiveEffect[]): Dep => {
    const dep = new Set<ReactiveEffect>(effects) as Dep
    return dep
}
```

2. 在 `packages/reactivity/src/effect.ts` ，修改 `KeyToDepMap` 的泛型：

代码块

```ts
import { Dep } from './dep'

type KeyToDepMap = Map<any, Dep>
```

3.

4. 修改 `track` 方法，处理 `Dep` 类型数据：

代码块

```ts
export function track(target: object, key: unknown) {
    ...
    // 获取指定 key 的 dep
    let dep = depsMap.get(key)
    // 如果 dep 不存在，则生成一个新的 dep，并放入到 depsMap 中
    if (!dep) {
        depsMap.set(key, (dep = createDep()))
    }

    trackEffects(dep)
}

/**
 * 利用 dep 依次跟踪指定 key 的所有 effect
 * @param dep
 */
export function trackEffects(dep: Dep) {
    dep.add(activeEffect!)
}
```

索引目录

13：框架实现：构

track

trigger

测试

此时，我们已经把指定 `key` 的所有依赖全部保存到了 `dep` 函数中，那么接下来我们就可以在 `trigger` 函数中，依次读取 `dep` 中保存的依赖。

trigger

1. 在 `packages/reactivity/src/effect.ts` 中：

代码块

```
export function trigger(
    target: object,
    key?: unknown,
) {
    // 依据 target 获取存储的 map 实例
    const depsMap = targetMap.get(target)
    // 如果 map 不存在，则直接 return
    if (!depsMap) {
        return
    }
    // 依据指定的 key，获取 dep 实例
    let dep: Dep | undefined = depsMap.get(key)
    // dep 不存在则直接 return
    if (!dep) {
        return
    }
    // 触发 dep
    triggerEffects(dep)
}

/**
 * 依次触发 dep 中保存的依赖
 */
export function triggerEffects(dep: Dep) {
    // 把 dep 构建为一个数组
    const effects = isArray(dep) ? dep : [...dep]
    // 依次触发
    for (const effect of effects) {
        triggerEffect(effect)
    }
}

/**
 * 触发指定的依赖
 */
export function triggerEffect(effect: ReactiveEffect) {
    effect.run()
}
```

至此，我们即可在 `trigger` 中依次触发 `dep` 中保存的依赖

测试

此时，再次运行测试实例 `packages/vue/examples/reactivity/reactive-dep.html` ，发现即可实现一对多的依赖关系。

12：功能升级：响应数据对应多个 effect ◂ 上一节　　下一节 ▸ 14：reactive 函数的局限性

我要提出意见反馈