

全部开发者教程

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

02：源码阅读：ref 复杂数据类型的响应性

03：框架实现：ref 函数 - 构建复杂数据类型的响应性

04：总结：ref 复杂数据类型的响应性

05：源码阅读：ref 简单数据类型的响应性



Sunday • 更新于 2022-10-19

◀ 上一节 02：源码阅读：... 04：总结：ref ... 下一节 ▶

03：框架实现：ref 函数 - 构建复杂数据类型的响应性

在上一小节中，我们已经查看了 vue 3 中 ref 函数针对复杂数据类型的响应性处理代码逻辑，那么这一小节，我们就可以实现一下对应的代码。

1. 创建 packages/reactivity/src/ref.ts 模块：

<> 代码块

```
1  import { createDep, Dep } from './dep'
2  import { activeEffect, trackEffects } from './effect'
3  import { toReactive } from './reactive'
4
5  export interface Ref<T = any> {
6    value: T
7  }
8
9  /**
10   * ref 函数
11   * @param value unknown
12   */
13  export function ref(value?: unknown) {
14    return createRef(value, false)
15  }
16
17  /**
18   * 创建 RefImpl 实例
19   * @param rawValue 原始数据
20   * @param shallow boolean 形数据，表示《浅层的响应性（即：只有 .value 是响应性的）》
21   * @returns
22   */
23  function createRef(rawValue: unknown, shallow: boolean) {
24    if (isRef(rawValue)) {
25      return rawValue
26    }
27    return new RefImpl(rawValue, shallow)
28  }
29
30  class RefImpl<T> {
31    private _value: T
32
33    public dep?: Dep = undefined
34
35    // 是否为 ref 类型数据的标记
36    public readonly __v_isRef = true
37
38    constructor(value: T, public readonly __v_isShallow: boolean) {
39      // 如果 __v_isShallow 为 true，则 value 不会被转化为 reactive 数据，即如果当前 v
40      this._value = __v_isShallow ? value : toReactive(value)
41    }
42
43    /**
44     * get语法将对象属性绑定到查询该属性时将被调用的函数。
45     * 即：xxx.value 时触发该函数
46     */
47    get value() {
```

索引目录

03：框架实现：ref



```

50     }
51
52     set value(newVal) {}
53 }
54
55 /**
56  * 为 ref 的 value 进行依赖收集工作
57  */
58 export function trackRefValue(ref) {
59     if (activeEffect) {
60         trackEffects(ref.dep || (ref.dep = createDep()))
61     }
62 }
63
64 /**
65  * 指定数据是否为 RefImpl 类型
66  */
67 export function isRef(r: any): r is Ref {
68     return !!r && r.__v_isRef === true
69 }

```

2. 在 `packages/reactivity/src/reactive.ts` 中, 新增 `toReactive` 方法:

```

<> 代码块
1  /**
2   * 将指定数据变为 reactive 数据
3   */
4   export const toReactive = <T extends unknown>(value: T): T =>
5       isObject(value) ? reactive(value as object) : value

```

3. 在 `packages/shared/src/index.ts` 中, 新增 `isObject` 方法:

```

<> 代码块
1  /**
2   * 判断是否为一个对象
3   */
4   export const isObject = (val: unknown) =>
5       val !== null && typeof val === 'object'

```

4. 在 `packages/reactivity/src/index.ts` 中, 导出 `ref` 函数:

```

<> 代码块
1  ...
2   export { ref } from './ref'

```

5. 在 `packages/vue/src/index.ts` 中, 导出 `ref` 函数: :

```

<> 代码块
1   export { reactive, effect, ref } from '@vue/reactivity'

```

至此, `ref` 函数构建完成。

我们可以增加测试案例 `packages/vue/examples/reactivity/ref.html` 中:

```

<> 代码块
1  <script>
2    const { ref, effect } = Vue
3
4    const obj = ref({
5      name: '张三'
6    })
7
8    // 调用 effect 方法
9    effect(() => {
10      document.querySelector('#app').innerText = obj.value.name

```

[意见反馈](#)

[收藏教程](#)

[标记书签](#)



```
12
13     setTimeout(() => {
14         obj.value.name = '李四'
15     }, 2000);
16 </script>
```

可以发现代码测试成功。

02: 源码阅读: ref 复杂数据类型的响应性 ◀ 上一节 下一节 ▶ 04: 总结: ref 复杂数据类型的响应性

 我要提出意见反馈

[企业服务](#) [网站地图](#) [网站首页](#) [关于我们](#) [联系我们](#) [讲师招募](#) [帮助中心](#) [意见反馈](#) [代码托管](#)



Copyright © 2022 imooc.com All Rights Reserved | 京ICP备 12003892号-11 京公网安备11010802030151号



 意见反馈

 收藏教程

 标记书签