慕课网首页 免费课 实战课 体系课 慕课教程 专栏 手记 企业服务



Q

03: 框架实现: 核

?

··

从所有教程的词条中查询…

首页 > 慕课教程 > Vue3源码分析与构建方案 > 03:框架实现:构建 ComputedRefImpl ,读...

全部开发者教程 ः≡

07. 心知. 151 同于双油大土 响应性

08: 总结

第七章:响应系统 computed && watch

01: 开篇

02: 源码阅读: computed 的响应

03:框架实现:构建 ComputedRefImpl,读取计 算属性的值

04: 框架实现: computed 的响应性: 初见调度器,处理脏的状态

05: 框架实现: computed 的

06: 总结: computed 计算属 性

07:源码阅读:响应性的数据 监听器 watch, 跟踪源码实现 逻辑 Sunday • 更新于 2022-10-19

◆ 上一节 02: 源码阅读: ...

04: 框架实现: ... 下一节 ▶

索引目录

03: 框架实现: 构建 ComputedRefImpl, 读取计算属性的值

对于 computed 而言,整体比较复杂,所以我们将分步进行实现。

那么对于本小节而言,我们的首先的目标是:构建 ComputedRefImpl 类,创建出 computed 方法,并且能够读取值

1. 创建 packages/reactivity/src/computed.ts :

```
<>代码块
     import { isFunction } from '@vue/shared'
     import { Dep } from './dep'
     import { ReactiveEffect } from './effect'
 4
     import { trackRefValue } from './ref'
 5
 6
 7
      * 计算属性类
8
 9
     export class ComputedRefImpl<T> {
10
         public dep?: Dep = undefined
11
         private _value!: T
12
13
         public readonly effect: ReactiveEffect<T>
14
         public readonly __v_isRef = true
15
17
         constructor(getter) {
             this.effect = new ReactiveEffect(getter)
18
19
             this.effect.computed = this
20
         }
21
22
         get value() {
23
             // 触发依赖
24
             trackRefValue(this)
             // 执行 run 函数
2.5
             this. value = this.effect.run()!
26
             // 返回计算之后的真实值
27
28
             return this._value
29
         }
30
     }
31
32
      * 计算属性
33
34
35
     export function computed(getterOrOptions) {
36
         let getter
37
         // 判断传入的参数是否为一个函数
38
         const onlyGetter = isFunction(getterOrOptions)
39
         if (onlyGetter) {
40
             // 如果是函数,则赋值给 getter
42
             getter = getterOrOptions
43
44
         const cRef = new ComputedRefImpl(getter)
```

♪ 意见反馈

♡ 收藏教程

□ 标记书签

```
48 return cRef as any
}
```

2. 在 packages/shared/src/index.ts 中, 创建工具方法:

```
// 代码块

/**
2 * 是否为一个 function
3 */
4 export const isFunction = (val: unknown): val is Function =>
5 typeof val === 'function'
```

3. 在 packages/reactivity/src/effect.ts 中, 为 ReactiveEffect 增加 computed 属性:

```
c>代码块

1  export class ReactiveEffect<T = any> {
2    //**
3    * 存在该属性,则表示当前的 effect 为计算属性的 effect
4    */
5    computed?: ComputedRefImpl<T>
6    ....
```

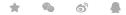
- 4. 最后不要忘记在 packages/reactivity/src/index.ts 和 packages/vue/src/index.ts 导出
- 5. 创建测试实例: packages/vue/examples/reactivity/computed.html:

```
<> 代码块
1
   <body>
2
    <div id="app"></div>
3 </body>
   <script>
4
5
     const { reactive, computed, effect } = Vue
    const obj = reactive({
8
    name: '张三'
9
    })
10
     const computedObj = computed(() => {
11
      return '姓名: ' + obj.name
12
13
14
15
     effect(() => {
      document.querySelector('#app').innerHTML = computedObj.value
16
17
18
    setTimeout(() => {
19
      obj.name = '李四'
21
   }, 2000);
22 </script>
```

那么此时,我们可以发现,计算属性,可以正常展示。

但是: 当 obj.name 发生变化时,我们可以发现浏览器 并不会 跟随变化,即: 计算属性并非是响应性的。那么想要完成这一点,我们还需要进行更多的工作才可以。

▶ 我要提出意见反馈



?

 \odot