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 我的课程



Q

从所有教程的词条中查询…

◆ 上一节 08: 框架实现: ...10: 问题分析: ... 下一节 →

首页 > 慕课教程 > Vue3源码分析与构建方案 > 09: 框架实现: 初步实现 watch 数据监听器

全部开发者教程 Ξ

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

08: 框架实现: 深入 scheduler 调度系统实现机制

09: 框架实现: 初步实现 watch 数据监听器

10: 问题分析: watch 下的依 频收集

11: 框架实现: 完成 watch 数 据监听器的依赖收集

12: 总结: watch 数据侦听器

13: 总结

索引目录

09: 框架实现: 初

?

 \odot

Sunday • 更新于 2022-10-19

09: 框架实现: 初步实现 watch 数据监听器

那么这一小节, 我们来看一下 watch 函数应该如何进行实现。

1. 创建 packages/runtime-core/src/apiWatch.ts 模块, 创建 watch 与 doWatch 函数:

```
<> 代码块
 1
      * watch 配置项属性
     export interface WatchOptions<Immediate = boolean> {
        immediate?: Immediate
         deep?: boolean
 7
     }
 8
9
10
      * 指定的 watch 函数
11
      * @param source 监听的响应性数据
       * @param cb 回调函数
13
      * @param options 配置对象
      * @returns
14
15
     export function watch(source, cb: Function, options?: WatchOptions) {
16
17
         return doWatch(source as any, cb, options)
18
19
20
     function doWatch(
21
         source,
22
         cb: Function.
         { immediate, deep }: WatchOptions = EMPTY_OBJ
23
24
25
         // 触发 getter 的指定函数
26
         let getter: () => any
27
         // 判断 source 的数据类型
2.8
         if (isReactive(source)) {
29
30
             // 指定 getter
             getter = () => source
31
32
             // 深度
33
             deep = true
34
         } else {
             getter = () => {}
35
36
         }
37
38
         // 存在回调函数和deep
39
         if (cb && deep) {
40
             // TODO
             const baseGetter = getter
41
             getter = () => baseGetter()
42
         }
43
45
         // 旧值
46
         let oldValue = {}
         // job 执行方法
47
48
         const job = () \Rightarrow {
```

```
51
                const newValue = effect.run()
                if (deep || hasChanged(newValue, oldValue)) {
53
                   cb(newValue, oldValue)
54
                    oldValue = newValue
55
                }
            }
56
57
        }
58
59
        // 调度器
60
        let scheduler = () => queuePreFlushCb(job)
61
62
        const effect = new ReactiveEffect(getter, scheduler)
63
        if (cb) {
64
65
            if (immediate) {
66
                job()
67
            } else {
               oldValue = effect.run()
68
           }
69
70
       } else {
71
          effect.run()
72
73
74
       return () => {
75
          effect.stop()
76
77
    }
```

2. 在 packages/reactivity/src/reactive.ts 为 reactive 类型的数据, 创建 标记:

```
<> 代码块
1 export const enum ReactiveFlags {
    IS_REACTIVE = '__v_isReactive'
3
4
5
    function createReactiveObject(
6
    ) {
8
      // 未被代理则生成 proxy 实例
9
      const proxy = new Proxy(target, baseHandlers)
1.0
      // 为 Reactive 增加标记
11
      proxy[ReactiveFlags.IS_REACTIVE] = true
12
13 ...
14
   }
```

?

0

3. 在 packages/shared/src/index.ts 中创建 EMPTY_OBJ:

```
1    /**
2  * 只读的空对象
3  */
4    export const EMPTY_OBJ: { readonly [key: string]: any } = {}
```

- 4. 在 packages/runtime-core/src/index.ts 和 packages/vue/src/index.ts 中导出 watch 函数
- 5. 创建测试实例 packages/vue/examples/reactivity/watch.html:

```
<> 代码块
1
    <script>
      const { reactive, watch } = Vue
2
3
     const obj = reactive({
4
      name: '张三'
8
    watch(obj, (value, oldValue) => {
9
     console.log('watch 监听被触发');
console.log('value', value);
10
11
12
13
      setTimeout(() => {
14
       obj.name = '李四'
15
    }, 2000);
16
17 </script>
```

此时运行项目,却发现,当前存在一个问题,那就是 watch 监听不到 reactive 的变化

那么这是因为什么呢?

大家可以先思考下这个问题,我们下一小节继续来说~~~

✔ 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

* * 6

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

