

全部开发者教程

00. 总纲

第五章：响应系统 - 初见 reactivity 模块

01：前言

02：源码阅读：reactive 的响应性，跟踪 Vue 3 源码实现逻辑

03：框架实现：构建 reactive 函数，获取 proxy 实例

04：框架实现：什么是 WeakMap？它和 Map 有什么区别？

05：框架实现：createGetter && createSetter

06：热更新的开发时：提升开发体验

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



Sunday • 更新于 2022-10-19

◀ 上一节 04：框架实现：... 06：热更新的开... 下一节 ▶

05：框架实现：createGetter && createSetter

那么当我们搞明白了 WeakMap 的特性之后，那么接下来我们就来看看 mutableHandlers 如何处理。

我们知道对于 Proxy 而言，它的 handler 可以监听 代理对象 的 getter 和 setter，那么此时的 mutableHandlers 就是监听 代理对象 getter 和 setter 的核心部分。

所以接下来我们需要创建对应的 get 和 set 监听：

<> 代码块

```
1  /**
2   * 响应性的 handler
3   */
4  export const mutableHandlers: ProxyHandler<object> = {
5    get,
6    set
7  }
```

getter

<> 代码块

```
1  /**
2   * getter 回调方法
3   */
4  const get = createGetter()
5
6  /**
7   * 创建 getter 回调方法
8   */
9  function createGetter() {
10     return function get(target: object, key: string | symbol, receiver: object) {
11         // 利用 Reflect 得到返回值
12         const res = Reflect.get(target, key, receiver)
13         // 收集依赖
14         track(target, key)
15         return res
16     }
17 }
```

setter

<> 代码块

```
1  /**
2   * setter 回调方法
3   */
4  const set = createSetter()
5
6  /**
7   * 创建 setter 回调方法
8   */
9  function createSetter() {
10     return function set(
11         target: object,
12         key: string | symbol,
13         value: any
```

索引目录

- 05：框架实现：ci
- getter
- setter
- track && trigg
- 测试

```

14     receiver: object
15   ) {
16     // 利用 Reflect.set 设置新值
17     const result = Reflect.set(target, key, value, receiver)
18     // 触发依赖
19     trigger(target, key, value)
20     return result
21   }
22 }

```

track && trigger

在 `getter` 和 `setter` 中分别调用了 `track && trigger` 方法，所以我们需要分别创建对应方法：

1. 创建 `packages/reactivity/src/effect.ts`：

<> 代码块

```

1  /**
2   * 用于收集依赖的方法
3   * @param target WeakMap 的 key
4   * @param key 代理对象的 key，当依赖被触发时，需要根据该 key 获取
5   */
6   export function track(target: object, key: unknown) {
7     console.log('track: 收集依赖')
8   }
9
10  /**
11   * 触发依赖的方法
12   * @param target WeakMap 的 key
13   * @param key 代理对象的 key，当依赖被触发时，需要根据该 key 获取
14   * @param newValue 指定 key 的最新值
15   * @param oldValue 指定 key 的旧值
16   */
17  export function trigger(
18    target: object,
19    key?: unknown,
20    newValue?: unknown
21  ) {
22    console.log('trigger: 触发依赖')
23  }

```

那么至此我们就可以：

1. 在 `getter` 时，调用 `track` 收集依赖
2. 在 `setter` 时，调用 `trigger` 触发依赖

我们可以在两个方法中分别进行一下打印，看看是否可以成功回调。

测试

在 `packages/vue/examples/reactivity/reactive.html` 中：

<> 代码块

```

1   const { reactive } = Vue
2
3   const obj = reactive({
4     name: '张三'
5   })
6   console.log(obj.name); // 此时应该触发 track
7   obj.name = '李四' // 此时应该触发 trigger

```

重新打包项目，运行以上测试。

