全部开发者教程

08：总结

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 03：框架实现：... 05：框架实现：... 下一节

04：框架实现：什么是 WeakMap？它和 Map 有什么区别？

对比 WeakMap 和 Map 的文档可知，他们两个具备一个核心共同点，那就是：**都是 {key, value} 的结构对象。**

但是对于 `WeakMap` 而言，他却存在两个不同的地方：

1. `key` 必须是对象
2. `key` 是弱引用的

其中第一个不同点比较好理解，但是第二个不同点是什么意思呢？那么我们本小节就来看一下这个 **弱引用** 指的是什么？

> 概念
> 弱引用：不会影响垃圾回收机制。即：WeakMap 的 key **不再存在任何引用时**，会被直接回收。
> 强引用：会影响垃圾回收机制。存在强应用的对象永远 **不会** 被回收。

我们来看下面两个例子：

代码块

```
// map
<script>
  // target 对象
  let obj = {
    name: '张三'
  }
  // 声明 Map 对象
  const map = new Map()
  // 保存键值对
  map.set(obj, 'value')
  // 把 obj 置空
  obj = null
</script>
```

在当前这段代码中，如果我们在浏览器控制台中，打印 `map` 那么打印结果如下：

即：**虽然 `obj` 已经不存在任何引用了，但是它并没有被回收，依然存在于 `Map` 实例中。**这就证明 `Map` 是强应用的，哪怕 `obj` 手动为 `null`，但是它依然存在于 `Map` 实例中。

接下来同样的代码，我们来看 `WeakMap`：

索引目录

04：框架实现：什么是 WeakMap？它和 Map 有什么区别？

总结

代码块

```
// target 对象
let obj = {
  name: '张三'
}
// 声明 Map 对象
const wm = new WeakMap()
// 保存键值对
wm.set(obj, 'value')
// 把 obj 置空
obj = null
```

在当前这段代码中，如果我们在浏览器控制台中，打印 `wm` 那么打印结果如下：

此时 `WeakMap` 中不存在任何值，即：`obj` **不存在其他引用时，** `WeakMap` **不会阻止垃圾回收，基于** `obj` **的引用将会被清除。**这就证明了 `WeakMap` 的 **弱引用特性。**

总结

那么由以上可知，对于 `WeakMap` 而言，它存在两个比较重要的特性：

1. `key` 必须是对象
2. `key` 是弱引用的

03：框架实现：构建 reactive 函数，获取 pr... ◂ 上一节　　下一节 ▸ 05：框架实现：createGetter && createSe...

我要提出意见反馈