

全部开发者教程

09: 引入代码格式化工具：prettier 让你的代码结构更加规范

10: 模块打包器：rollup

11: 初见框架雏形：配置路径映射

12: 总结

第四章：响应系统 - 响应系统的核心设计原则

01: 前言

02: JS 的程序性

03: 如何让你的程序变得更加“聪明”？

04: vue 2 的响应性核心 API: Object.defineProperty

05: Object.defineProperty 在设计层的缺陷

06: vue3的响应性核心 API: proxy

Sunday • 更新于 2022-10-19

03: 如何让你的... 05: Object.defi... 下一节 >

04: vue 2 的响应性核心 API: Object.defineProperty

vue2 以 Object.defineProperty 作为响应性的核心 API，该 API 可以监听：**指定对象的指定属性的 getter 和 setter**

那么接下来我们就可以借助该 API，让我们之前的程序进行**自动计算**，该 API 接收三个参数：**指定对象、指定属性、属性描述符对象**

<> 代码块 预览 复制

```
1  <script>
2    // 定义一个商品对象，包含价格和数量
3    let quantity = 2
4    let product = {
5      price: 10,
6      quantity: quantity
7    }
8    // 总价格
9    let total = 0;
10   // 计算总价格的匿名函数
11   let effect = () => {
12     total = product.price * product.quantity;
13   };
14
15   // 第一次打印
16   effect();
17   console.log(`总价格: ${total}`); // 总价格: 20
18
19   // 监听 product 的 quantity 的 setter
20   Object.defineProperty(product, 'quantity', {
21     // 监听 product.quantity = xx 的行为，在触发该行为时重新执行 effect
22     set(newVal) {
23       // 注意：这里不可以是 product.quantity = newVal，因为这样会重复触发 set 行为
24       quantity = newVal
25       // 重新触发 effect
26       effect()
27     },
28     // 监听 product.quantity，在触发该行为时，以 quantity 变量的值作为 product.quantity 的属
29     get(val) {
30       return quantity
31     }
32   });
33 </script>
```

那么此时我们就通过 Object.defineProperty 方法成功监听了 quantity 属性的 getter 和 setter 行为，现在当 quantity 发生变化时，effect 函数将重新计算，以此得到最新的 total

03: 如何让你的程序变得更加“聪明”？ < 上一节 下一节 > 05: Object.defineProperty 在设计层的缺陷

索引目录

04: vue 2 的响应

