Vue3 如何实现响应式

Vue2 通过 `Object.defineProperty` 实现响应式。Vue3 通过使用 Proxy 和 Reflect 来实现响应式系统。

`Object.defineProperty` 的缺点：

- Vue2 深度监听需要一次性递归；
- 无法监听新增和删除属性（需要 Vue.set 和 Vue.delete）；
- 无法监听原生数组，需要特殊处理。

1. Proxy 使用方法

JavaScript 中的 Proxy 对象是用于定义基本操作的自定义行为。它可以让你在某些操作上添加自定义的行为，比如读写属性、设置 getter 和 setter、apply 操作等等。

```
let target = {
  name: 'target',
};

let handler = {
  get: function(target, prop) {
    console.log(`Reading ${prop}`);
    return target[prop];
  },
  set: function(target, prop, value) {
    console.log(`Writing ${prop}`);
    target[prop] = value;
  },
};

let proxy = new Proxy(target, handler);

console.log(proxy.name); // Reading name
console.log(proxy.age); // undefined

proxy.age = 25; // Writing age
```

在这个例子中，读取 `proxy.name` 时，它实际上会去读取 `target.name`，但是在这个过程中，它还会打印出 `Reading name`。同样，当你写入 `proxy.age` 时，它也会打印出 `Writing age`。

2. Proxy 优缺点

优点：能规避 Object.defineProperty 的问题

- Proxy 深度监听时是监听到下一层，什么时候 get 到这一层级什么时候监听，不会一次性全部监听所有层级。
- Proxy 第二个参数中 deleteProperty 能实现删除属性
- Proxy 第二个属性中 set 能实现新增属性。

- 可原生监听数组变化。

缺点：无法兼容所有浏览器，无法 ployfill