



```
48     return cRef as any
    }
```

2. 在 `packages/shared/src/index.ts` 中, 创建工具方法:

```
<> 代码块
1  /**
2   * 是否为一个 function
3   */
4   export const isFunction = (val: unknown): val is Function =>
5     typeof val === 'function'
```

3. 在 `packages/reactivity/src/effect.ts` 中, 为 `ReactiveEffect` 增加 `computed` 属性:

```
<> 代码块
1  export class ReactiveEffect<T = any> {
2    /**
3     * 存在该属性, 则表示当前的 effect 为计算属性的 effect
4     */
5     computed?: ComputedRefImpl<T>
6     ....
```

4. 最后不要忘记在 `packages/reactivity/src/index.ts` 和 `packages/vue/src/index.ts` 导出

5. 创建测试实例: `packages/vue/examples/reactivity/computed.html` :

```
<> 代码块
1  <body>
2    <div id="app"></div>
3  </body>
4  <script>
5    const { reactive, computed, effect } = Vue
6
7    const obj = reactive({
8      name: '张三'
9    })
10
11    const computedObj = computed(() => {
12      return '姓名: ' + obj.name
13    })
14
15    effect(() => {
16      document.querySelector('#app').innerHTML = computedObj.value
17    })
18
19    setTimeout(() => {
20      obj.name = '李四'
21    }, 2000);
22  </script>
```

那么此时, 我们可以发现, 计算属性, 可以正常展示。

**但是:** 当 `obj.name` 发生变化时, 我们可以发现浏览器 **并不会** 跟随变化, 即: **计算属性并非是响应性的**。那么想要完成这一点, 我们还需要进行更多的工作才可以。

02: 源码阅读: `computed` 的响应 ◀ 上一节      下一节 ▶ 04: 框架实现: `computed` 的响应性: 初见...

✎ 我要提出意见反馈

