链滴

JavaScript 内存应在何时及如何使用?

作者：Vanessa

原文链接：https://ld246.com/article/1584409146550

来源网站：链滴

2020-03-17

回答

内存是一种常用的技术，可以显著的提升代码速度。他使用缓存来存储结果，因此再次调用该耗时函时将不需要再执行相同的运行。基于这一定义，我们能简单的抽取出一些标准来帮助我们确定在我们代码中何时应使用内存：

- 当执行过慢，消耗过多或函数调用时间过长时就应该使用内存
- 内存可以加快后续函数调用，因此当你需要在相同情况下多次调用同一函数时最好使用他
- 我们将结果存储在内存中，因此需避免相同函数在不同的情况下被多次调用的情况

一个简单的，面向对象的内存实现示例如下：

```
class MyObject {
  constructor(data) {
    this.data = data;
    this.data[this.data.length - 2] = { value: 'Non-empty' };
  }

  firstNonEmptyItem() {
    return this.data.find(v => !!v.value);
  }

  firstNonEmptyItemMemo() {
    if (!this.firstNonEmpty)
      this.firstNonEmpty = this.data.find(v => !!v.value);
    return this.firstNonEmpty;
  }
}

const myObject = new MyObject(Array(2000).fill({ value: null }));

for (let i = 0; i < 100; i ++)
  myObject.firstNonEmptyItem();       // 耗时约 7ms
for (let i = 0; i < 100; i ++)
  myObject.firstNonEmptyItemMemo();   // 耗时约 1ms
```

加分回答

- 上面的示例展现了一种在类内部实现的内存，使用他的前提是假设他的数据结构在对象的整个生命期中都不会发生改变，这样我们只能调用特定的方法，因此他不能被重用。除此外，他还不支持给函传参，否则会改变结果
- 因此我们可以根据给定的函数和参数，使用 Map 来存储不同的结果值，具体可实现可参见该代片段
- 通过使用 JavaScript 中的 Proxy 捕获器 handler.apply() 也可以提供一种有趣的可替代方式，如，使用他也可以达到同样的目的：

```
const memoize = fn => new Proxy(fn, {
  cache: new Map(),
  apply (target, thisArg, argsList) {
```

```
    let cacheKey = argsList.toString();
    if(!this.cache.has(cacheKey))
      this.cache.set(cacheKey, target.apply(thisArg, argsList));
    return this.cache.get(cacheKey);
  }
});

const fibonacci = n => (n <= 1 ? 1 : fibonacci(n - 1) + fibonacci(n - 2));
const memoizedFibonacci = memoize(fibonacci);

for (let i = 0; i < 100; i ++)
  fibonacci(30);                    // 耗时约 1212ms
for (let i = 0; i < 100; i ++)
  memoizedFibonacci(30);            // 耗时约 13ms
```

返回总目录

每天 30 秒系列之前端面试