第十四课 享元模式

学习目录

- **使用场景**
- **使用方式**
- **优缺点**

一. 使用场景

一开始看到这个享元模式名称的时候大家可能很好奇这是一种什么样的设计模式，因为从名称中我们很难知道这个设计模式的作用，其实享元模式的含义特别简单，它其实是一种性能优化的结构性设计模式，在处理代码中大量重复对象带来的内存增加问题上特别好用。简单来说，我们如果按照正常开发流程需要在程序中 new 出来 3000 个差不多结构的实例对象，由于浏览器内存限制，并且假如这 3000 个差不多结构的实例对象比较复杂，那么肯定会对性能带来比较大的影响。享元模式可以让我们原本需要 new 出来 3000 个实例对象变成只需要 new3 个实例对象，这是不是就是对性能的增强了。

这么一解释大家基本就明白了享元模式的作用，它是把那些需要大量重复构建的对象进行解耦处理，把构建对象的构造函数中的状态进行划分，把不容易变化的状态定义成实例对象的内部状态，其实是对象的内部属性，把容易变化的状态定义成实例对象的外部状态，其实也是对象的外部属性，这样我们也就可以使用工厂模式或者单例模式来构建符合业务要求的对象了。

享元模式的核心思想是找出对象或者函数的内部状态和外部状态，让实例对象可以共享内部状态，根据业务需求构减少共享对象的数量。

二．使用方式

我们还是以构建数字的函数来看下享元模式的基本结构。

普通数字构建

```
function SetTextFn(type,num){
    this.div = null;
    this.type = type;
    this.num = num;
}
SetTextFn.prototype.init = function(){
    this.div = document.createElement('div');
    this.div.innerHTML = this.type + this.num;
    document.body.appendChild(this.div);
}
for(var i=0;i<30;i++){
    var s1 = new SetTextFn('星星课堂',100 + i);
    s1.init();
}
for(var i=0;i<50;i++){
    var s1 = new SetTextFn('xingxingclassroom',300 + i);
    s1.init();
}
```

从这个例子中，我们可以发现一个大问题，虽然我构建出来了这么多数字，但是我也在内存中快速增加了几十个实例对象，这显示是不合理的，因为这些实例对象会增加内存开销，假如实例对象足够复杂，需要的实例对象数量又继续增加，那这个代码的性能绝对很差。

其实我们主要是想构建出 30 个星星课堂和 50 个 xingxingclassroom，实际上来说我们只需要实例化两次得到两个实例对象即可，因此我们可以用享元模式来帮助我们实现这个目标。

首先这里的 type 其实可以作为构造函数的内部状态保存在构造函数内部，num 与下标的组合完全可以独立成外部状态，区别内部状态和外部状态的标志是内部状态可以被实例对象所共享，很明显这里 type 是被其他实例对象共享的，而传入的 num 和下标的组合不能被共享，因此可以让 num 与下标的组合作为外部状态。

享元模式构建数字

```
function SetTextFn(type){
    this.div = null;
    this.type = type;
    this.num = 0;
}
SetTextFn.prototype.init = function(){
    this.div = document.createElement('div');
    this.div.innerHTML = this.type + this.num;
    document.body.appendChild(this.div);
}
```

```
SetTextFn.prototype.setAttrFn = function(num){
    this.num = num;
}

var SetTextFactory = (function(){
    var cache = {};
    return {
        created:function(type){
            var result = cache[type];
            if(!result){
                result = new SetTextFn(type);
                cache[type] = result;
            }
            return result;
        }
    }
})();

var f1 = SetTextFactory.created('星星课堂');
var f2 = SetTextFactory.created('xingxingclassroom');
for(var i=0;i<30;i++){
    f1.setAttrFn(100 + i);
```

```
        f1.init();
    }
    for(var i=0;i<50;i++){
        f2.setAttrFn(300 + i);
        f2.init();
    }
```

这里我们把实例对象通过一个工厂来构建出来，同时把内部状态放在实例对象实例化的过程中，把外部状态通过一个挂钩函数在程序运行期间把数字动态添加进去，因此程序中实例对象的数量减少了，我们可以用两个实例对象来构建出 30 与 50 个实例对象构建的数字。

三．优缺点

1.享元模式可以减少程序中大量的重复对象，对于程序性能的增强有很大帮助，特别适合用在需要构建大量重复对象的业务场景中使用。

2.享元模式先区分出内部状态和外部状态，通过工厂单例来获取共享的对象，调用对象的方法传入外部状态，从而保证了对象的可复用性与多样性。

3.享元模式由于需要构建享元模式工厂和享元模式管理器，所以适合大量重复对象的场景，因此不适合普通业务场景，在普通业务场景使用享元模式可能会增加代码复杂度。

4.享元模式使用关键是区分好内部状态和外部状态，因此构建一个共享对象要求业务逻辑需求相对稳定保持不变，错误的内部状态和外部状态会让共享对象的使用相对复杂。

谢谢观看！

我是星星课堂老师：周小周