

## 6 代理模式

代理模式是为一个对象提供一个代用品或者占位符，以便控制对它的访问

这种模式非常有意义，例如现实生活中的明星经纪人作为代理，客户实际上访问的是替身对象，由替身对象做出一些处理之后再转交给本体对象

### 第一个例子-小明追MM的故事

小明有一个女神A，然后打听到A和他有一个共同的朋友B，决定让B代替自己完成送花这件事情

```
1  /**
2   * 不用代理B
3   */
4  var Flower = function(){};
5
6  var xiaoming = {
7      sendFlower: function(target){
8          var flower = new Flower();
9          target.receiveFlower(flower);
10     }
11 }
12
13 var A = {
14     receiveFlower: function(flower){
15         console.log('收到花' + flower)
16     }
17 }
18
19 xiaoming.sendFlower(A);
20
21 /**
22 * 加入代理B
23 */
24
25 // ...
26 var B = {
27     receiveFlower: function(flower){
28         A.receiveFlower(flower);
29     }
30 }
```

```
31
32 xiaoming.sendFlower(B);
```

上述代理B看上去毫无用处，只是简单转发了一下，但是实际情况下代理B可能更了解女神A，B会监听A的心情变化并在适当的时候转交花，提高小明的成功率：

```
1  /**
2   * 代理B发挥用处了
3   */
4  var A = {
5      receiveFlower: function(flower){
6          console.log('收到花' + flower)
7      },
8      listenGoodMood: function(fn){
9          // 假设10s条件后心情变化
10         setTimeout(() => {fn()}, 10000)
11     }
12 }
13
14 var B = {
15     receiveFlower: function(flower){
16         A.listenGoodMood(() => {
17             A.receiveFlower(flower);
18         })
19     }
20 }
21
22 xiaoming.sendFlower(B);
```

这里是不是能体会到一点proxy的价值了呢？他更懂目标，也将小明和女神解耦开来了

## 保护代理和虚拟代理

- 保护代理：如果上述代理B的工作是过滤一些不合理或者无用的对A的请求，那么就叫做保护代理
  - 例如年龄太大或者太穷的表白对象
  - JS中我们无法判断谁访问了某个对象，出发加上业务标记
- 虚拟代理：假设之前new Flower的代价比较大，我们可以将new的职责交给B，在A心情好的时候再去花费开销new Flower，延迟到真正需要它的时候再进行
  - 本章主要讨论的是虚拟代理

## 真实示例-虚拟代理实现图片预加载

Web开发中，图片预加载是常用的技术，因为直接给img的src设置图片链接，有时候会因为网络或者图片过大等原因造成空白。通常做法先设置loading图片展位，然后异步加载图片，之后再回填到img节点，这个场合就很适合虚拟代理：

```
1  /**
2   * 图片预加载
3   */
4
5  var myImage = (function(){
6      var imgNode = document.createElement('img');
7      document.body.appendChild(imgNode);
8
9      return {
10         setSrc: function(src){
11             imgNode.src = src;
12         }
13     }
14 })()
15
16 // 引入代理对象proxy，避免页面空白 - 图片被加载好之前会出现菊花图loading.gif占位
17
18 var proxyImage = (function(){
19     var img = new Image;
20     img.onload = function(){
21         myImage.setSrc(img.src);
22     }
23     return {
24         setSrc: function(src){
25             // 加载本地占位
26             myImage.setSrc('file:///C:/Users/loading.gif');
27             // Image对象去异步加载真正的图片-加载完成后通过onload顶替myImage
28             img.src = src;
29         }
30     }
31 })()
32
33 proxyImage.setSrc('http://xxx/x.png');
```