**小知识代码梳理**

1. 订阅发布模式，通过once事件，过滤重复性事件响应（运行环境：Node）

**场景：**数据库查询时，避免大量并发重复请求做相同的数据查询

**方式：**通过status 锁定在一个查询周期内的连续多次请求，仅查询一次。该时间段内的所有并发请求都绑定selected事件。在完成查询时，触发selected事件所有绑定事件的对象均能获得并处理数据。（即下文所说的一对多）此处可能因为并发引起once侦听器过多出现警告，需要调用setMaxListeners(0)移除警告，或设置更大的阈值。

**功能：**

1、避免了大量请求造成数据库查询阻塞的同时。

2、once该事件仅触发一次并自动销毁，解决侦听事件过多产生的性能问题。

var emitter = new events.EventEmitter();

var status = ' ready';

var select = function(callback) {

emitter.once('seleted', callback );

if(status == ' ready') {

status = ' pending';

db.select(' SQL', function(results) {

status = ' ready';

emitter.emit(' seleted', results);

})

}

}

1. 函数的多对一调用。结合订阅发布模式，实现事件多对多绑定

**解释：**事件与侦听器一对多，即一次trigger，触发多次绑定的同一个事件。

事件与侦听器多对一，即只有在一个事件被多次trigger，并传入数据后，才能综合获得的数据执行对应方法。

**目的：**处理多个IO可以并行调用，但因为返回的数据在处理函数中有依赖关系，只能串行执行。  
**方式：**借助哨兵变量和第三方函数实现多对一。哨兵变量用来检查函数执行的次数，函数外的数组用来保存每次执行返回的数据。使用**偏函数**处理哨兵变量和第三方函数的关系，可以达到更好的接口功能。

结合订阅发布模式，实现事件与侦听器的多对多

var after = function(times, callback) {

var count = 0, result = [];

return function(key, value){

result[key] = value;

if(++count == times) {

callback (result)

}

}

}

var done = after(3, render) // emit调用3次后执行render，实现多次调用一次执行

var emitter = new events.Emitter();

emitter.on('finish', done);

emitter.on('finish', other); // 多个on('finish', function(){})，实现一次触发，多次执行

fs.readFile(template\_path, 'utf8', function(err, temple){

emitter.emit('finish', 'template', data);

});

db.query(sql, function(err, data){

emitter.emit('finish', 'data', data);

});

110n.get(function(err, resources){

emitter.emit('finish', 'resources', data);

});