**underscore**

正如jQuery统一了不同浏览器之间的差异，让我们可以简单的对DOM进行操作，underscore则提供了一套完善的函数式编程的接口，让我们可以更方便的在JavaScript中实现函数式编程。

jQuery在加载时，会把自身绑定到唯一的全局变量$上，underscore与其类似，会把自身绑定到唯一的全局变量\_上。

eg:

用underscore实现map()操作如下：

‘use strict’;

\_.map({a:1,b:2,c:3},(v,k)=>k+’=’+v);//[‘a=1’,’b=2’,’c=3’]

//和Array的map()与filter类似，underscore的map()和filter传入的函数为function(value,key),第一个参数接收value，第二个参数接收key。

但是underscore的这两个函数可以作用与object。

Collections

1. Object作map()操作的返回结果是Array

Object作\_.mapObject操作的返回结果是Object

1. 当集合的所有元素都满足条件时，\_.every()函数返回true，

当集合的至少一个元素满足条件时，\_.some()函数返回true

1. 判断一个字符串是否小写

function isLowerCase(str){

return str === str.toLowerCase();

}

1. max / min
   1. 这两个函数直接返回集合中最大和最小的数
   2. 空集合会返回-Infinity和Infinity，所以要先判断集合不为空：
   3. 注意，如果集合是Object，max()和min()只作用于value，忽略掉key
2. groupBy

eg:

var scores = [20, 81, 75, 40, 91, 59, 77, 66, 72, 88, 99];

var groups = \_.groupBy(scores, function (x) {

if (x < 60) {

return 'C';

} else if (x < 80) {

return 'B';

} else {

return 'A';

}

});

// 结果:

// {

// A: [81, 91, 88, 99],

// B: [75, 77, 66, 72],

// C: [20, 40, 59]

// }

1. shuffle/sample
   1. \_.shuffle[1,2,3,4,5,6];//[4,2,1,6,5,3]每次结果都不一样
   2. sample()则是随机选择一个或多个元素

Arrays

1. first / last

这两个函数分别取第一个和最后一个元素：

1. flatten()接收一个Array，无论这个Array里面嵌套了多少个Array，flatten()最后都把它们变成一个一维数组
2. zip()把两个或多个数组的所有元素按索引对齐例如，你有一个Array保存了名字，另一个Array保存了分数，现在，要把名字和分数给对上，用zip()轻松实现：
3. unzip()则是反过来
4. object()函数将数组变为对象
5. range()让你快速生成一个序列，不再需要用for循环实现了
6. \_.range(0, -10, -1);
7. bind()可以帮我们把console对象直接绑定在log()的this指针上，以后调用log()就可以直接正常调用了
8. 假设我们经常计算2y，每次都写Math.pow(2, y)就比较麻烦，如果创建一个新的函数能直接这样写pow2N(y)就好了，这个新函数pow2N(y)就是根据Math.pow(x, y)创建出来的偏函数，它固定住了原函数的第一个参数（始终为2）
9. 如果我们不想固定第一个参数，想固定第二个参数怎么办？比如，希望创建一个偏函数cube(x)，计算x3，可以用\_作占位符，固定住第二个参数

eg:

var cube = \_.partial(Math.pow, \_, 3);

cube(3); // 27

11. memoize

如果一个函数调用开销很大，我们就可能希望能把结果缓存下来，以便后续调用时直接获得结果。举个例子，计算阶乘就比较耗时

1. once()保证某个函数执行且仅执行一次
2. delay()可以让一个函数延迟执行，效果和setTimeout()是一样的，但是代码明显简单了,如果要延迟调用的函数有参数，把参数也传进去：

eg:

var log = \_.bind(console.log, console);

\_.delay(log, 2000, 'Hello,', 'world!');

// 2秒后打印'Hello, world!': var log = \_.bind(console.log, console);

\_.delay(log, 2000, 'Hello,', 'world!');

// 2秒后打印'Hello, world!'

Objects

keys()可以非常方便地返回一个object自身所有的key，但不包含从原型链继承下来的。

allKeys()除了从object自身上的key,还包含从原型链上继承下来的：

values（）返回object自身但是不包含原型链继承的所有值。

mapObject()就是针对object的map版本。

invert()把object上的每个key-value来个交换，key变成value，value变成key。

extend/extendOwn:

extend()就是把object的key-value合并到第一个object并返回。

如果有相同的key，后面的object的value将覆盖前面的object的value。

extendOwn()和extend()类似，但获取属性忽略从原型链上继承下来的属性。

clone()会把原有对象的所有属性复制到新的对象中。

注意，clone()是“软链接”。两个对象引用的value其实是同一对象。修改一个会影响到另一个。

isEqual()是对两个对象进行深度比较，如果完全相同则返回“true”;

isEqual()也可以比较Array。

### Chaining

chain()使对象能够链式调用。

eg:

\_.chain([1,4,9,16,25])

.map(Math.sqrt)

.filter(x=>x%2===1)

.value();

//[1,3,5]