JS柯里化（currying）

Alex Sun
2015-01-18

关于柯里化，可以参看wiki的介绍，但是有些绕，举一个例子，假如有如下函数：

```
function add(x, y) {
    return x + y;
}
```

现在希望有另外一个函数，可以通过如下方式调用：

```
function currying(fn) {
    // ...
}

currying(add, 2)(3); //add(2, 3)
```

简而言之，可以理解为目前有一个函数f，接收若干个参数a1-an，然后有一个柯里化函数，接收的第一个参数为f，后面接收参数为a1-am(m<n),该柯里化函数返回一个函数g，g接收a(m+1)-an。当然，该说法并不准确，只是一个例子而已。那么根据这个思路，可以实现currying如下：

```
function currying(fn) {
    var args = Array.prototype.slice.call(arguments, 1);
    return function(y) {
        var newArgs =
args.concat(Array.prototype.slice.call(arguments));
        return fn.apply(null, newArgs);
    }
}
```

可以发现，柯里化主要依赖的就是闭包。当然，这个例子比较简单，下面看一个稍微复杂的例子，现在把add函数加强，让其接收5个参数，然后又一个柯里化函数currying，可以实现如下方式调用：

```
currying(add, 1, 2, 3, 4, 5);
currying(add, 1, 2, 3)(4, 5);
currying(add, 1)(2, 3)(4, 5);
```

```
currying(add)(1)(2)(3)(4)(5);
```

第一个参数是一个函数，后面可以链式调用参数。这里需要知道原始的函数的参数个数，可以通过fn.length来得到。

```
function currying(fn) {
    var len = fn.length;
    var args = Array.prototype.slice.call(arguments, 1);

    if (args.length >= len) {
        return fn.apply(null, args.slice(0, len));
    } else {
        return function() {
            var newArgs =
args.concat(Array.prototype.slice.call(arguments));
            newArgs.unshift(fn);
            return currying.apply(null, newArgs);
        };
    }
}
```