二、PromiseKit的原理

再来看看PMKResolve函数，它是Promise的真正核心。它执行所有的handler操作，并且设置result值。因为之前有把block存在handler中，所以block就可以顺利执行。

```
static void PMKResolve(PMKPromise *this, id result) {
    void (^set)(id) = ^(id r){
        NSArray *handlers = PMKSetResult(this, r);
        for (void (^handler)(id) in handlers)
            handler(r);
    };

    if (IsPromise(result)) {
        PMKPromise *next = result;
        dispatch_barrier_sync(next->_promiseQueue, ^{
            id nextResult = next->_result;

            if (nextResult == nil) {  // ie. pending
                [next->_handlers addObject:^(id o){
                    PMKResolve(this, o);
                }];
            } else
                set(nextResult);
        });
    } else
        set(result);
}
```

这个一个递归函数，能形成递归的条件就是那句PMKResolve(this,o);当nextResult = nil时候，此时这个promise还是pending状态，还没有被执行，这个时候就要递归调用，直到nextResult不为nil,就会调用set方法，set方法是一个匿名函数，里面for循环会依次循环，执行handler数组里面的每一个block。里面的if语句先判断result是否是一个promise，如果不是promise，就会执行set方法，依次调用各个block。

一、PromiseKit的简介

二、PromiseKit的原理

三、PromiseKit的实战

四、PromiseKit的总结

四、 Promise.resolve()

三、 Promise.reject()