学习 Promise

Promise

promise.then(callback)

在callback中使用return返回值时，return的值会由Promise.resolve()进行包装处理，返回一个新的Promise对象。无论返回的是什么值，promise.then()会将回调函数的返回值进行转换返回一个新的Promise对象，而不是简单的只是注册了回调函数。

promise.then()和promise.catch()的区别

promise.catch()其实本质上只是promise.then(undefined,callback(err))的语法糖。但是两者最大的区别是在实际使用时错误的捕获范围，考虑下面两种情况：

1. promise.then(onResolve,onReject)
2. promise.then(onResolve).catch(onReject)
   两种调用方式当`promise chain`（每一次promise调用会增加promise chain）上第一个promise 状态变为reject时Error都会被捕获到，1中Error会在.then中被捕获，2中Error会在.catch中被捕获。
   但是后一次的`promise chain`即.then()新返回的Promise对象的onResolve回调中发生错误时，1是无法捕获的，2却能捕获到。

总结：
promise.catch()和promise.then(onResolve,onReject)的onReject都能捕获到其之前的`promise chain`中抛出的Error，但是promise.then(onResolve,onReject)的onReject无法捕获onResolve中的错误。

Promise测试

Mocha支持异步测试，因此同样也支持Promise的支持，并且针对Promise测试有一些特别的地方。

Mocha中普通异步测试

在mocha中使用回调函数的风格对异步处理进行测试。

```
describe('this is a asynchronous test',function(){
    it('should use done',function(done){
        setTimeout(function(){
            // assert
            done()  // tell mocha the test finish
        },100)
    })
})
```

和普通的测试几乎一样，唯一的区别是需要传入done回调函数，并在测试完成（通常是断言后）调用done()通知测试完成。

Mocha中简单使用done的Promise测试

因此测试Promise时和测试其他的异步测试类似：

```
describe('this is a asynchronous test',function(){
    it('should use done',function(done){
        Promise.resolve.then(function(value){
            // assert value
            done();
        })
    })
})
```

只需要在promise 的onResolve回调中进行断言，但是这种方式会有问题：当断言失败的时候，throw 出来的error会被Promise所捕获而不会被测试框架捕获，因此测试不会结束，直到超时，并且在测试报告中显示的错误不是断言错误而是done()未调用导致超时的错误。

改进：
针对这个问题我们可以简单的改一下done调用的时机来解决。

```
describe('this is a asynchronous test',function(){
    it('should use done',function(done){
        Promise.resolve.then(function(value){
            // assert value
        }).then(done,done)
    })
})
```

当断言通过时调用`done()`,断言失败时调用`done(err)`，因此都能被mocha捕获到。
但是这种方式需要添加额外的`.then(done,done)`，忘记添加的话则会超时。

Mocha支持的Promise的测试风格

Mocha官网关于Promise测试：

> Alternately, instead of using the done() callback, you can return a promise. This is useful if the APIs you are testing return promises instead of taking callbacks:

即可以简单的return Promise而不用使用done回调的形式进行测试。

良好的Promise测试

- 明确Promise状态应该为Fulfilled和Reject其中之一
- 在`.then(onFulfilled,onReject)`调用非期望回调抛出错误，调用期望回调则在期望回调中进行传入参数的断言。
  例如，期望promise 状态为Fulfilled，并且传入的值为期望的值

```
return promise.then(function(value){
    expect(value).to.equal('expect value')
},function(){
    throw new Error('期望Promise状态为Fulfilled，但为Rejec
t')
})
```