Promise：CommonJS社区提出Promise的规范。更优的异步编程解决方案，解决传统异步编程中回调函数嵌套过深的问题。

笔记本：	1.有关印象笔记		
创建时间：	2020/5/30 15:38	更新时间：	2020/7/12 1:17
作者：	1639079350@qq.com		
标签：	2.promise		
URL：	about:blank		

Promise：CommonJS社区提出Promise的规范。更优的异步编程解决方案，解决传统异步编程中回调函数嵌套过深的问题。

状态和回调：
1.状态（Pending - Fulfilled - Rejected）确定就不可以更改；两种状态（Fulfilled - Rejected）只能选择其一；
2.状态回调对应的函数：Fulfilled - onFulfilled、Rejected - onRejected
3.常见误区：多层回调不是promise嵌套，而是promise链式操作。

```
// 创建promise实例(两种状态) -- 调用实例方法 [分别调用两种状态方法]
const promise = new Promise(function (reject, resolve) {
    resolve(100);
})

const promise = new Promise(function (reject, resolve) {
    reject(new Error('promise is reject'));
})

promise.then(function (value) {
    console.log('resolve', value); // resolve 100
}, function (error) {
    console.log('resolve', error); // resolve Error: promise is reject
})
```

回调过程：
1.Promise对象的then方法会返回新的Promise对象作为后面then方法回调的参数，后面的then方法就是在为上一个then返回的Promise注册回调；
2.如果回调中返回的是Promise，那后面then方法的回调会等待它的结束；

```
function ajax (url) {
  return new Promise(function (resolve, reject) {
    var xhr = new XMLHttpRequest()
    xhr.open('GET', url)
    xhr.responseType = 'json'
    xhr.onload = function () {
      if (this.status === 200) {
        resolve(this.response)
      } else {
        reject(new Error(this.statusText))
      }
    }
    xhr.send()
  })
}

ajax('/api/users.json').then(function (res) {
  console.log(res)
}, function (error) {
  console.log(error)
})
```

异常处理：

1.onReject函数：作为then的参数直接调用。若果then中返回了第二个promise,则捕获不到第二个promise的异常;

2.实例的catch函数注册onRejected方法：更常见，适合链式调用；

```
// onReject函数:
ajax(url)
    .then(function onFulfilled(value) {
        log('onFilfilled', value);
    }, function onReject(error) {
        log('onRejected', value);
    })

// catch链式调用;
ajax(url)
    .then(function onFulfilled(value) {
        log('onFilfilled', value);
    })
    .catch(function onReject(error) {
        log('onRejected', value);
    })
```