Promise.resolve()：

笔记本：	1.有关印象笔记		
创建时间：	2020/6/3 14:42	更新时间：	2020/7/12 1:18
作者：	1639079350@qq.com		
标签：	3.promise静态方法		

Promise.resolve()：

1.相同操作的不同写法：

```
// 直接链式调用：reslove传参
    Promise.resolve('foo').then(function (v) {
            console.log(v); // foo
        })

// 分状态链式操作：resolve传参 + then
        new Promise(function (resolve,reject) {
            resolve('foo');
        }).then(function(r) {
            console.log(r); // foo
        })

// 传入对象原始状态链式操作
        Promise.resolve({
            then: function( onFilfilled,onReiected) {
                onFilfilled('foo');
            }}).then(function (v) {
                console.log(v); // foo
            })
```

2.用Promise.resolve()包装对象，得到的是原本的对象：【待验证】

```
        var promise = ajax('/api/user.json');
        var promise2 = Promise.resolve(promise);
        console.log(promise === promise2);
```

Promide.reject()：参数 = 失败的原因（和.catch连用）

```
        Promise.reject('anything').catch(function(err) {
            console.log(err); // anything
        })
```

```
Promise.reject(new Error('rejected')).catch(function(err) {
    console.log(err); // Error: rejected
})
```

promise并行执行：

1.Promise.all()：

（1）所有任务成功结束才成功，否则失败；

（2）接收一个数组，每个元素都是promise对象，返回一个全新的promise对象；

2.Promise.race()：只跟随第一个结束的任务结束；

```
// Promise.all()
var promise = Promise.all([
    ajax('url1'),
    ajax('url1'),
]);
promise.then(function(v) {
    console.log(v);
}).catch(function (err) {
    console.log(err);
})

// Promise.race(): 500ms以内任务没有结束就报错：
const request = ajax('url');
const timeout = new Promise((resolve,reject) => {
    setTimeout(() => reject(new Error('timeout')),500);
})

Promise.race([
    request,timeout
]).then( v=> {
    console.log(v);
}).catch( error => {
    console.log( error);
})
```