19.异步 Promise

学习要点：

1.Promise 介绍

2.实例测试

本节课我们来开始学习 ES6 新增的 Promise 异步通信方案的功能。

一. Promise 介绍

1. Promise：即异步通信编程的一种解决方案，它比传统回调式更加的强大；
2. ES6 之前非常多层次嵌套的同步、异步，执行顺序混乱且不好维护；
3. Promise 就很好的解决了这些问题，我们先了解一下它的语法：

```
//创建一个 Promise 实例
let p = new Promise((resolve, reject) => {
    //一顿异步通信操作后，返回成功或失败
    //然后判断成功或失败去执行 resolve 或 reject
    if (false) {
        //console.log('异步通信执行成功！');
        resolve('执行成功！');
    } else {
        //console.log('异步通信执行失败！');
        reject('执行失败！');
    }
});

//then 方法可执行 resolve 的回调函数
//catch 方法可执行 reject 的回调函数
p.then((value) => {
    console.log(value);
}).catch((reason) => {
    console.log(reason);
});
```

PS：如果你有过很多层异步通信实战基础，上面提供的方法会突然感觉清晰很多；

PS：因为它把多层嵌套的回调函数给分离出来，通过 then 和 catch 来实现；

4. 通过上面例子的语法，我们发现 p 作为 Promise 实例，可以进行连缀链式操作；
5. 当执行了 then 方法后，本身依旧返回了当前的 Promise 实例，方便链式；
6. 注释中也说明了，通过构造方法的两个参数去执行回调函数，并传递参数；
7. 事实上，catch()方法还可以作为 then 第二参数进行存在，方便多层回调；

```
//一体化操作
p.then((value) => {
    console.log(value);
```

```
}, (reason) => {
    console.log(reason);
});
```

二．实例测试

1. 我们做个模拟多层异步通信的实例测试，要异步多个内容，并按指定顺序执行；
2. 先给出不进行 Promise 异步，看它执行的顺序：

```
//模拟异步 1
setTimeout(() => {
    console.log('1.返回异步通信');
}, 3500);

//模拟异步 2
setTimeout(() => {
    console.log('2.返回异步通信');
}, 800);

//模拟异步 3
setTimeout(() => {
    console.log('3.返回异步通信');
}, 1500);
```

PS：这里不管你怎么调节，最终输出结果总是：2，3，1。需求顺序要：1，2，3；

3. 将上面模拟异步通信，通过 Promise 进行改装，再看看执行结果；

```
let p1 = new Promise((resolve, reject) => {
    //模拟异步 1
    setTimeout(() => {
        //console.log('1.异步通信');
        resolve('1.返回异步通信');
    }, 3500);
});

let p2 = new Promise((resolve, reject) => {
    //模拟异步 2
    setTimeout(() => {
        //console.log('2.异步通信');
        resolve('2.返回异步通信');
    }, 800);
});


let p3 = new Promise((resolve, reject) => {
    //模拟异步 3
    setTimeout(() => {
```

```
        //console.log('3.异步通信');
        resolve('3.返回异步通信');
    }, 1500);
});


//执行回调
p1.then((value) => {
    console.log(value);
    return p2;
}).then((value) => {
    console.log(value);
    return p3;
}).then((value) => {
    console.log(value);
});
```