同步和异步

- JS 是单线程，只能一次干一件事。
- 同步会阻塞代码执行。
- 异步不会阻塞代码执行。等待同步代码执行完，再执行异步代码。

```
console.log(1);
setTimeout(() => {
    console.log(2);
}, 1000);
console.log(3);
setTimeout(() => {
    console.log(4);
}, 0);
console.log(5);
```

1. 上面代码先打印 1。
2. 遇到 setTimeout 先记下，执行后面的同步代码，打印 3。
3. 又遇到 setTimeout 先记下，执行后面的同步代码，打印 5。
4. 然后打印 4。
5. 等待 1000 ms 后，打印 2。

异步代码不会阻塞，而是执行后面的同步代码，等待同步代码执行完了，再回过头来执行异步代码。

```
console.log(1);
alert(2);
console.log(3);
```

上面代码先打印 1，然后弹出弹框显示 2，如果不关闭弹框，3 不会打印，就好像最后一步代码被阻塞了。

连环问：promise 解决了什么问题

promise 解决了之前异步回调函数层层嵌套的问题。

连环问：手写 promise 加载一张图片

```
// 手写 promise 加载一张图片
function loadImage(src) {
    return new Promise((resolve, reject) => {
        const img = document.createElement("img");
        img.src = src;
        img.alt = "promise方式加载的图片";
        img.onload = () => {
            resolve(img);
        };
```

```
        img.onerror = (err) => {
            const err = new Error("图片加载失败！");
            reject(err);
        }
    });
}
const url1 =
"https://gimg2.baidu.com/image_search/src=http%3A%2F%2Fpic8.nipic.com%2F20100722%2F11494_104536084657_2.jpg&refer=http%3A%2F%2Fpic8.nipic.com&app=2002&size=f9999,10000&q=a80&n=0&g=0n&fmt=auto?sec=1650548796&t=7fd6430013f61e5e198cba77b0797724";
const url2 =
"https://img0.baidu.com/it/u=2381213954,401093073&fm=253&fmt=auto&app=138&f=JPEG?w=500&h=281";

loadImage(url1)
.then(img1 => {
    console.log("图片1宽度", img1.width);
    return img1;
})
.then(img1 => {
    console.log("图片1高度", img1.height);
    return loadImage(url2);
})
.then(img2 => {
    console.log("图片2宽度", img2.width);
    return img2;
})
.then(img2 => {
    console.log("图片2高度", img2.height);
})
.catch(err => {
    console.log(err);
});
```