异步和同步

- 基于 JS 是单线程语言
- 异步不会阻塞代码执行
- 同步会阻塞代码执行

单线程和异步

- 遇到等待（网络请求，定时任务）不能卡住
- 需要异步
- 回调 callback 函数形式

应用场景

- 网络请求，如 ajax 图片加载
- 定时任务，如 setTimeout

知识点

- 单线程和异步
- 应用场景
- callback hell 和 Promise

应用场景

```
// 图片加载
console.log('start')
let img = document.createElement('img')
img.onload = function () {
    console.log('loaded')
}
img.src = '/xxx.png'
console.log('end')
```


题目

```
// setTimeout 笔试题
console.log(1)
setTimeout(function () {
    console.log(2)
}, 1000)
console.log(3)
setTimeout(function () {
    console.log(4)
}, 0)
console.log(5)
```


题目

- 同步和异步的区别是什么？
- 手写用 Promise 加载一张图片
- 前端使用异步的场景有哪些？

Promise

```
function getData(url) {
    return new Promise((resolve, reject) => {
        $.ajax({
            url,
            success(data) {
                resolve(data)
            },
            error(err) {
                reject(err)
            }
        })
    })
}
```

Promise

```
const url1 = '/data1.json'
const url2 = '/data2.json'
const url3 = '/data3.json'
getData(url1).then(data1 => {
    console.log(data1)
    return getData(url2)
}).then(data2 => {
    console.log(data2)
    return getData(url3)
}).then(data3 => {
    console.log(data3)
}).catch(err => console.error(err))
```


异步和同步

```
// 异步
console.log(100)
setTimeout(function () {
    console.log(200)
}, 1000)
console.log(300)
```

```
// 同步
console.log(100)
alert(200)
console.log(300)
```


应用场景

```
// setTimeout
console.log(100)
setTimeout(function () {
    console.log(200)
}, 1000)
console.log(300)
```

```
// setInterval
console.log(100)
setInterval(function () {
    console.log(200)
}, 1000)
console.log(300)
```

单线程和异步

- JS 是单线程语言，只能同时做一件事儿
- 浏览器和 nodejs 已支持 JS 启动进程，如 Web Worker
- JS 和 DOM 渲染共用同一个线程，因为 JS 可修改 DOM 结构

应用场景


```
// ajax
console.log('start')
$.get('./data1.json', function (data1) {
    console.log(data1)
})
console.log('end')
```


callback hell

```
// 获取第一份数据
$.get(url1, (data1) => {
    console.log(data1)

    // 获取第二份数据
    $.get(url2, (data2) => {
        console.log(data2)

        // 获取第三份数据
        $.get(url3, (data3) => {
            console.log(data3)

            // 还可能获取更多的数据
        })
    })
})
```