

总结 event loop 过程 1

- ♦ 同步代码，一行一行放在 Call Stack 执行
- ♦ 遇到异步，会先“记录”下，等待时机（定时、网络请求等）
- ♦ 时机到了，就移动到 Callback Queue

总结 event loop 过程 2

- ♦ 如 Call Stack 为空（即同步代码执行完）Event Loop 开始工作
- ♦ 轮询查找 Callback Queue，如有则移动到 Call Stack 执行
- ♦ 然后继续轮询查找（永动机一样）

DOM 事件和 event loop

```
console.log('Hi')

setTimeout(function cb1() {
  console.log('cb1') // cb 即 callback
}, 5000)

console.log('Bye')
```

```
<button id="btn1">提交</button>

<script>
console.log('Hi')

$('#btn1').click(function (e) {
  console.log('button clicked')
})

console.log('Bye')
</script>
```

DOM 事件和 event loop

- ◆ JS 是单线程的
- ◆ 异步 (setTimeout , ajax 等) 使用回调 , 基于 event loop
- ◆ DOM 事件也使用回调 , 基于 event loop

Promise

- ◆ 三种状态
- ◆ 状态的表现和变化
- ◆ then 和 catch 对状态的影响

@3234109

状态的表现

- ◆ pending 状态 , 不会触发 then 和 catch
- ◆ resolved 状态 , 会触发后续的 then 回调函数
- ◆ rejected 状态 , 会触发后续的 catch 回调函数

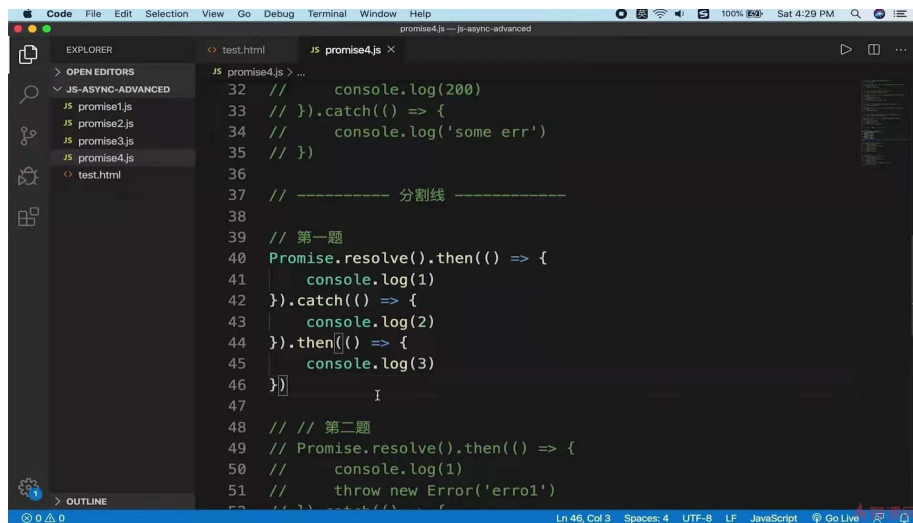
三种状态

- ◆ pending resolved rejected
- ◆ pending —> resolved 或 pending —> rejected
- ◆ 变化不可逆

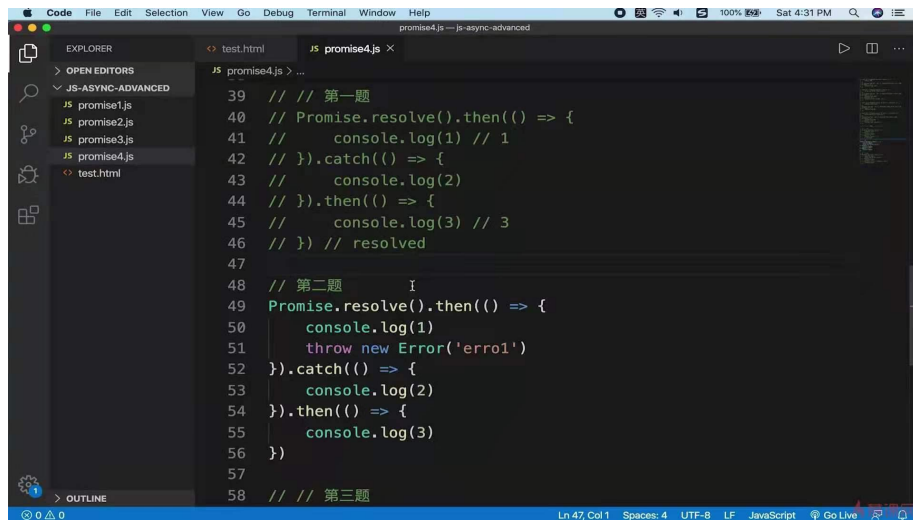
then 和 catch 改变状态

- ◆ then 正常返回 resolved，里面有报错则返回 rejected
- ◆ catch 正常返回 resolved，里面有报错则返回 rejected

慕课网



```
32 // console.log(200)
33 // }).catch(() => {
34 // console.log('some err')
35 // })
36
37 // ----- 分割线 -----
38
39 // 第一题
40 Promise.resolve().then(() => {
41   console.log(1)
42 }).catch(() => {
43   console.log(2)
44 }).then(() => {
45   console.log(3)
46 })
47
48 // // 第二题
49 // Promise.resolve().then(() => {
50 //   console.log(1)
51 //   throw new Error('erro1')
52 // })
```



```
39 // // 第一题
40 // Promise.resolve().then(() => {
41 //   console.log(1) // 1
42 // }).catch(() => {
43 //   console.log(2)
44 // }).then(() => {
45 //   console.log(3) // 3
46 // }) // resolved
47
48 // 第二题
49 Promise.resolve().then(() => {
50   console.log(1)
51   throw new Error('erro1')
52 }).catch(() => {
53   console.log(2)
54 }).then(() => {
55   console.log(3)
56 })
57
58 // // 第三题
```

```
52 // }).catch(() => {
53 // console.log(2) // 2
54 // }).then(() => {
55 // console.log(3) // 3
56 // }) // resolved
57
58 // 第三题
59 Promise.resolve().then(() => {
60 console.log(1)
61 throw new Error('erro1')
62 }).catch(() => {
63 console.log(2)
64 }).catch(() => { // 注意这里是 catch
65 console.log(3)
66 })
67
```

Promise 总结

- ◆ 三种状态，状态的表现和变化
- ◆ then 和 catch 对状态的影响（重要）
- ◆ then 和 catch 的链式调用（常考）