异步专题

上节课难点知识总结

- 箭头函数
- 模版字符串
- 解构赋值
- 展开运算符
- babel 使用

课堂主题

异步专题

课堂目标

1. 掌握递归的执行流程
2. 明确什么是异步，什么是同步
3. 掌握 Promise 的使用方法
4. 掌握 Async 和 Await 的使用方法

正课内容

同步和异步

同步和异步是一种消息通知机制

- 同步：A调用B，B处理获得结果，才返回给A。A在这个过程中，一直等待B的处理结果，没有拿到结果之前，需要A（调用者）一直等待和确认调用结果是否返回，拿到结果,然后继续往下执行。 在 JS 中，正常的代码执行，全部走的都是同步模式,必须拿到一行的执行结果，再去走下一行

---

alert(1); fn2(); function fn2(){ console.log(2); }

- 异步：A调用B，无需等待B的结果，B通过状态，通知等来通知A或回调函数来处理。 调用结果返回时，会以消息或回调的方式通知调用者

在 JS 中，定时器，动画帧 这些操作，都是异步操作，所以是在 回调函数中 处理执行后的内容

---

setTimeout(()=>{ console.log(1); },100);
fn2(); function fn2(){ console.log(2); }

回调地狱

- 封装简易运动框架
- 最早我们处理异步流程，都是通过回调来处理的， 但是回调多了，代码的结构就必然嵌套层级特别多，造成可读性和维护性的直线下降 - 这就是回调地狱

Promise 对象

ES6的Promise对象是一个构造函数，用来生成Promise实例。 所谓Promise对象，就是代表了未来某个将要发生的事件（通常是一个异步操作）。 它的好处在于，有了Promise对象，就可以将异步操作以同步操作的流程表达出来，避免了层层嵌套的回调函数

Promise 基本语法

new Promise(function(resolve,reject){

})

Promise 内部状态

- Pending 在等待(异步流程执行中)
- Fulfilled(标准)||Resolved 执行成功 - 调用resolve之后改变
- Rejected 执行失败 - 调用 reject 之后改变

then

promise.then(onFulfilled,onRejected)

参数： onFulfilled 当Promise变成接受状态（fulfillment）时，该参数作为回调函数被调用（参考：Function）。该函数有一个参数，即接受的最终结果（the fulfillment value）。如果传入的 onFulfilled 参数类型不是函数，则会在内部被替换为(x) => x ，即原样返回 promise 最终结果的函数

```
onRejected
    当Promise变成拒绝状态（rejection ）时，该参数作为回调函数被调用（参考：Function）。该函数有一个参数,，即拒绝的原因（the rejection reason）。
```

返回值： 当一个Promise完成（fulfilled）或者失败（rejected），返回函数将被异步调用（由当前的线程循环来调度完成）。具体的返回值依据以下规则返回： - 如果then中的回调函数返回一个值，那么then返回的Promise将会成为接受状态，并且将返回的值作为接受状态的回调函数的参数值。 - 如果then中的回调函数没有返回值，那么then返回的Promise将会成为接受状态，并且该接受状态的回调函数的参数值为 undefined。 - 如果then中的回调函数抛出一个错误，那么then返回的Promise将会成为拒绝状态，并且将抛出的错误作为拒绝状态的回调函数的参数值。 - 如果then中的回调函数返回一个已经是接受状态的Promise，那么then返回的Promise也会成为接受状态，并且将那个Promise的接受状态的回调函数的参数值作为该被返回的Promise的接受状态回调函数的参数值。 - 如果then中的回调函数返回一个已经是拒绝状态的Promise，那么then返回的Promise也会成为拒绝状态，并且将那个Promise的拒绝状态的回调函数的参数值作为该被返回的Promise的拒绝状态回调函数的参数值。 - 如果then中的回调函数返回一个未定状态（pending）的Promise，那么then返回Promise的状态也是未定的，并且它的终态与那个Promise的终态相同；同时，它变为终态时调用的回调函数参数与那个Promise变为终态时的回调函数的参数是相同的。

Promise.reject

Promise.reject(reason) 返回一个状态为 Rejected 的 Promise 对象

参数： reason 失败原因

Promise.resolve

Promise.resolve(value) 返回一个状态为 resolved 的 Promise 对象

参数： value 将被Promise对象解析的参数

Promise.catch

> 捕获前一个promise抛出的错误

Promise.all

> Promise.all方法用于将多个Promise实例，包装成一个新的Promise实例，当所有Promise都成功的时候，整个Promise.all才成功

Promise.race

> 与Promise.all方法类似将多个promise包装成一个新的promise实例
> 但是其中有一项的状态发生改变新的实例的状态就会随着改变

async函数

只要函数名之前加上async关键字，就表明该函数内部有异步操作。 该异步操作应该返回一个Promise对象，前面用await关键字注明。 当函数执行的时候，一旦遇到await就会先返回，等到触发的异步操作完成，再接着执行函数体内后面的语句

总结

- 同步和异步
- Promise 写法
- Async 和 await 写法

练习

把简易版运动框架的链式操作自己实现一遍

扩展视频安排

- for of循环 和 迭代器编写
- Generator 函数 和 co 执行函数