结构化程序设计和执行过程

时间循环是 js 引擎之外的事情

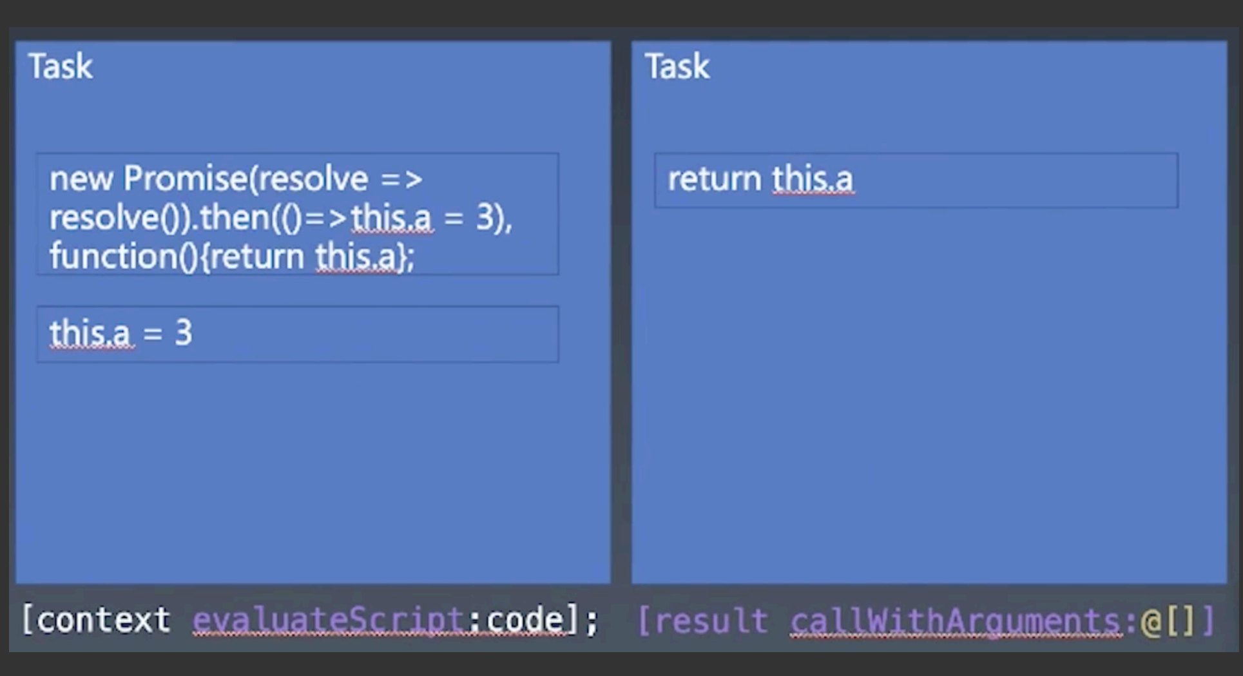
事件循环 = 宏任务队列 - 《重学前端》是 **js 调用方使用 js 的一种方式。**
在引擎 js context 之外，不是 js 引擎的一部分，也不是 js 语言的一部分

```
事件循环 <=> while(true){} // 一定条件退出的结构
```

- 代码片段
- 模块 module
- 函数 function

```
<!-- 1、代码片段 -->
<script>
var a = 1;

a++;


new Promise(resolve => resolve()).then(() => { console.log('a'); });

</script>


<!-- 2、函数 -->
<script>
setTimeout(function(){}, 1000);

</script>


<!-- 3、模块 -->
<script type="module">

// ...
</script>
```

https://developer.apple.com/documentation/javascriptcore/jsvalue

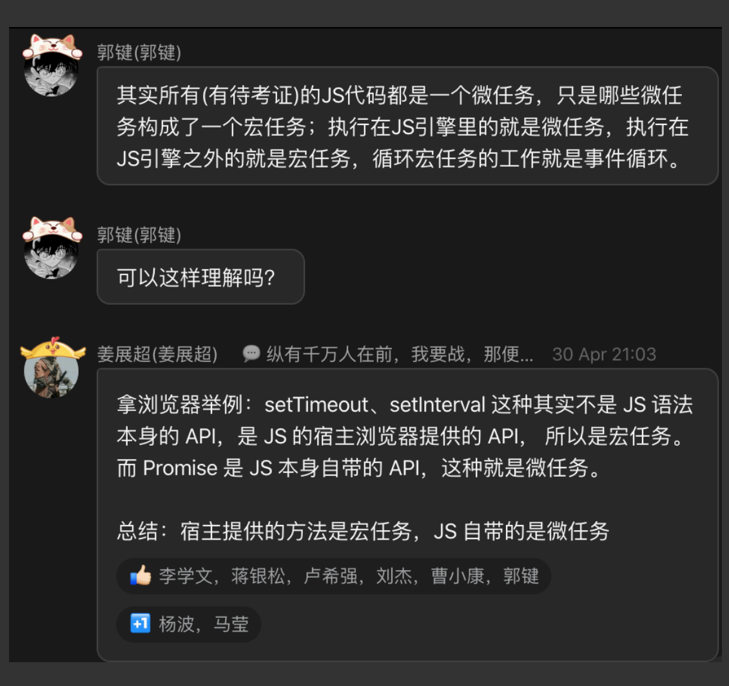


```
new Promise(resolve => resolve()).then(()=>this.a = 3), function(){return this.a};
this.a = 3
```

```
return this.a
```

```
[context evaluateScript:code];  [result callWithArguments:@[]]
```

jsc 执行 evaluateScript 或者 callWithArguments 就是一个宏任务。一个宏任务可由多个微任务（then）聚合，任何 js 代码都是在微任务中执行的。
微任务是保存在js 的执行队列



郭键(郭键)
其实所有(有待考证)的JS代码都是一个微任务，只是哪些微任务构成了一个宏任务；执行在JS引擎里的就是微任务，执行在JS引擎之外的就是宏任务，循环宏任务的工作就是事件循环。

郭键(郭键)
可以这样理解吗?

姜展超(姜展超) 纵有千万人在前，我要战，那便... 30 Apr 21:03
拿浏览器举例：setTimeout、setInterval 这种其实不是 JS 语法本身的 API，是 JS 的宿主浏览器提供的 API， 所以是宏任务。而 Promise 是 JS 本身自带的 API，这种就是微任务。

总结：宿主提供的方法是宏任务，JS 自带的是微任务
👍 李学文，蒋银松，卢希强，刘杰，曹小康，郭键
+1 杨波，马莹

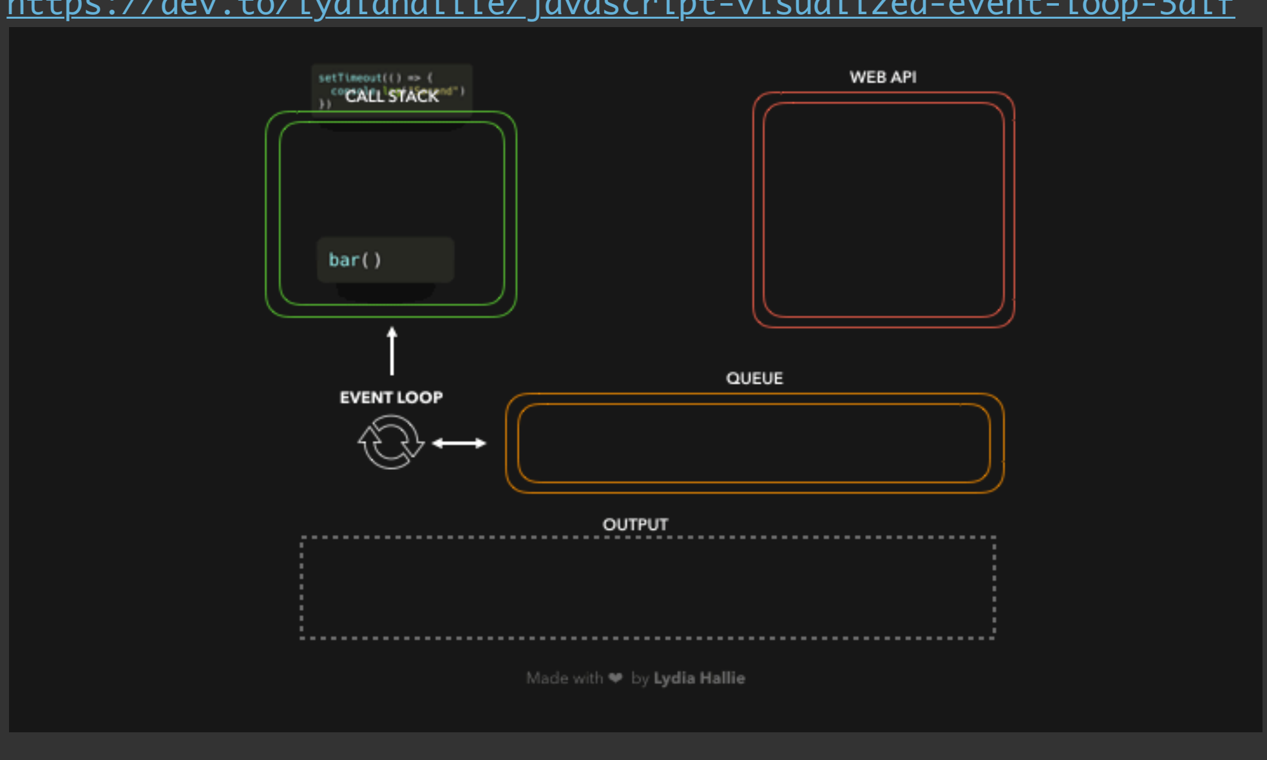
Promise，异步结构化的方式

一个 then 就对应一个微任务

一个Script标签内的就算是一个宏任务

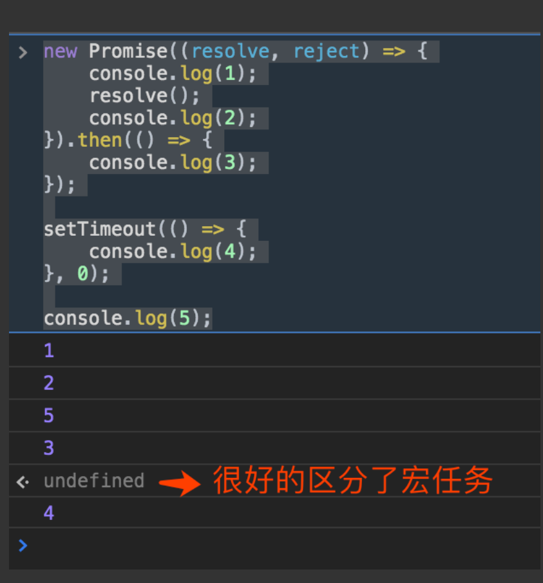
js 引擎类似于一个库，调用多少次就执行多少次。宏任务队列类似于一个服务，等待 js 代码输入，然后执行
js 不可访问任务列表。

https://dev.to/lydiahallie/javascript-visualized-event-loop-3dif



https://juejin.im/post/59e85eebf265da430d571f89

```
new Promise((resolve, reject) => {
    console.log(1);
    resolve();
    console.log(2);
}).then(() => {
    console.log(3);
});


setTimeout(() => {
    console.log(4);
}, 0);


console.log(5);
```



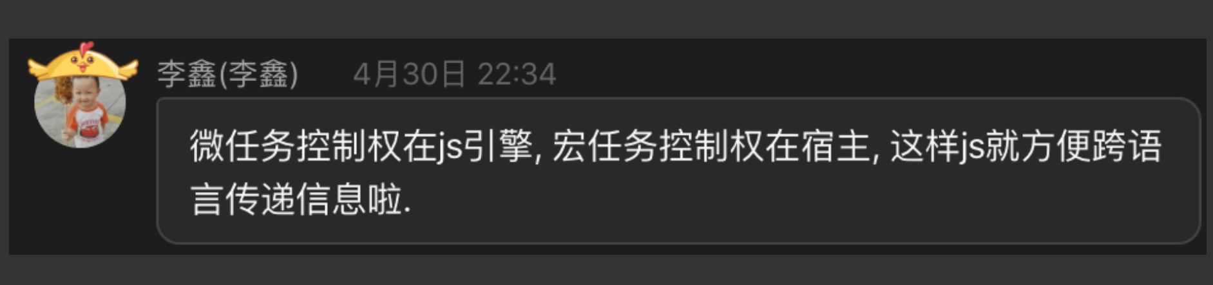
```
> new Promise((resolve, reject) => {
      console.log(1);
      resolve();
      console.log(2);
  }).then(() => {
      console.log(3);
  });

  setTimeout(() => {
      console.log(4);
  }, 0);

  console.log(5);
  1
  2
  5
  3
< undefined
  4
>
```

很好的区分了宏任务

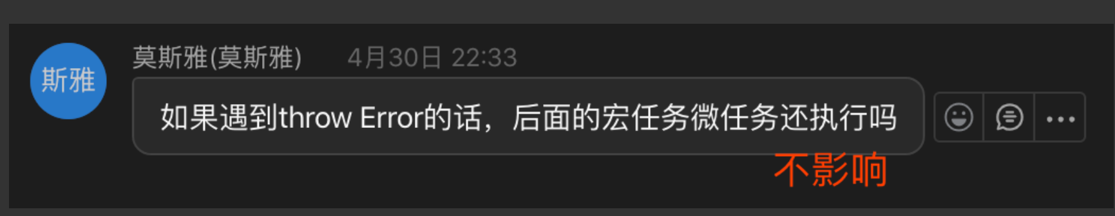
https://juejin.im/post/5e58c618e51d4526ed66b5cf

对应 js 标准中 8.4 Jobs and **Job Quenes —— 微任务**

浏览器端目前只有 MutationObserver 和 Promise 产生微任务。



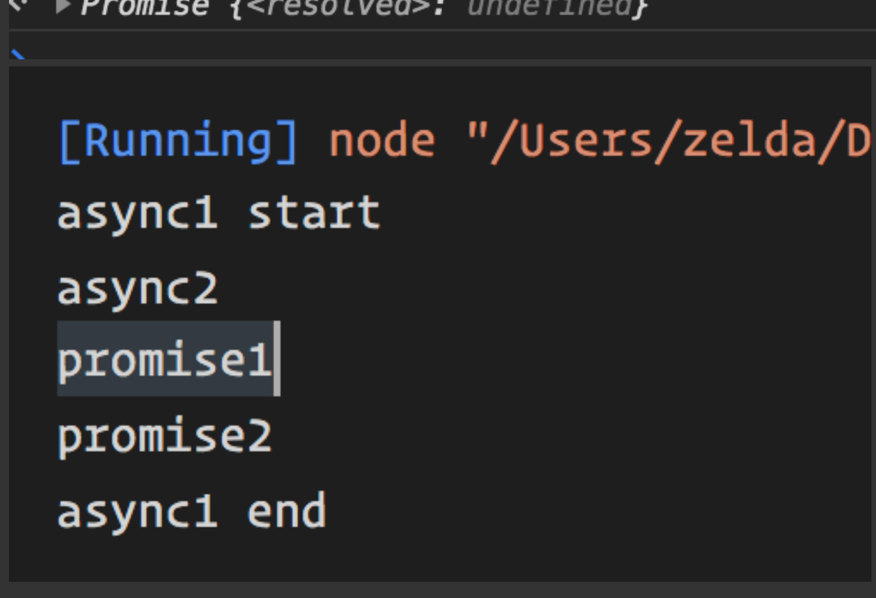
李鑫(李鑫) 4月30日 22:34
微任务控制权在js引擎, 宏任务控制权在宿主, 这样js就方便跨语言传递信息啦.



莫斯雅(莫斯雅) 4月30日 22:33
如果遇到throw Error的话，后面的宏任务微任务还执行吗
不影响

safari、node 与 Chrome 表现不同，await

```
> async function async1() {
      console.log('async1 start'); // 第一个宏任务 1
      await async2();
      console.log('async1 end'); // 第一个微任务 1
  }
  async function async2() {
      console.log('async2');  // 第一个宏任务 2
  }
  async1(); // 开始执行
  new Promise(function (resolve) {
      console.log('promise1');  // 第一个宏任务 3
      resolve();
  }).then(function () {
      console.log('promise2'); // 第一个微任务 2
  });
  async1 start
  async2
  promise1
  async1 end
  promise2
< ▸ Promise {<resolved>: undefined}
```

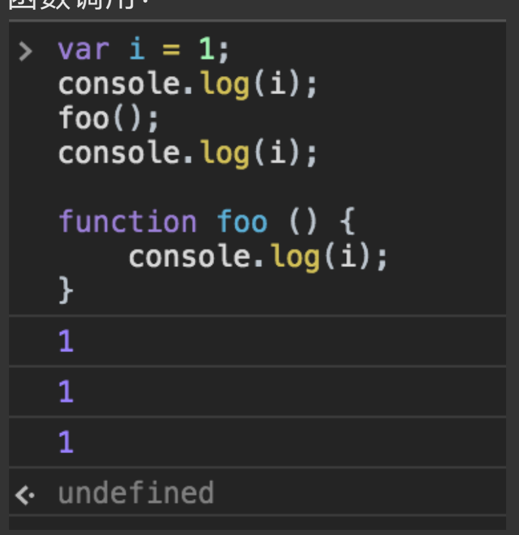


```
[Running] node "/Users/zelda/D
async1 start
async2
promise1
promise2
async1 end
```

JS 执行粒度：
- 直接量/变量/this
- 表达式，不同类型的表达式
- 语句/声明，Completion Record
- 函数调用（关键的数据结构：Execution Context）
- 微任务（Promise）
- 宏任务
- JS Context => Realm
    - Global Object
    - js 标准 18 The Global Object

作业
https://antv-g6.gitee.io/zh/docs/manual/introduction

函数调用：



```
> var i = 1;
  console.log(i);
  foo();
  console.log(i);

  function foo () {
      console.log(i);
  }
  1
  1
  1
< undefined
```

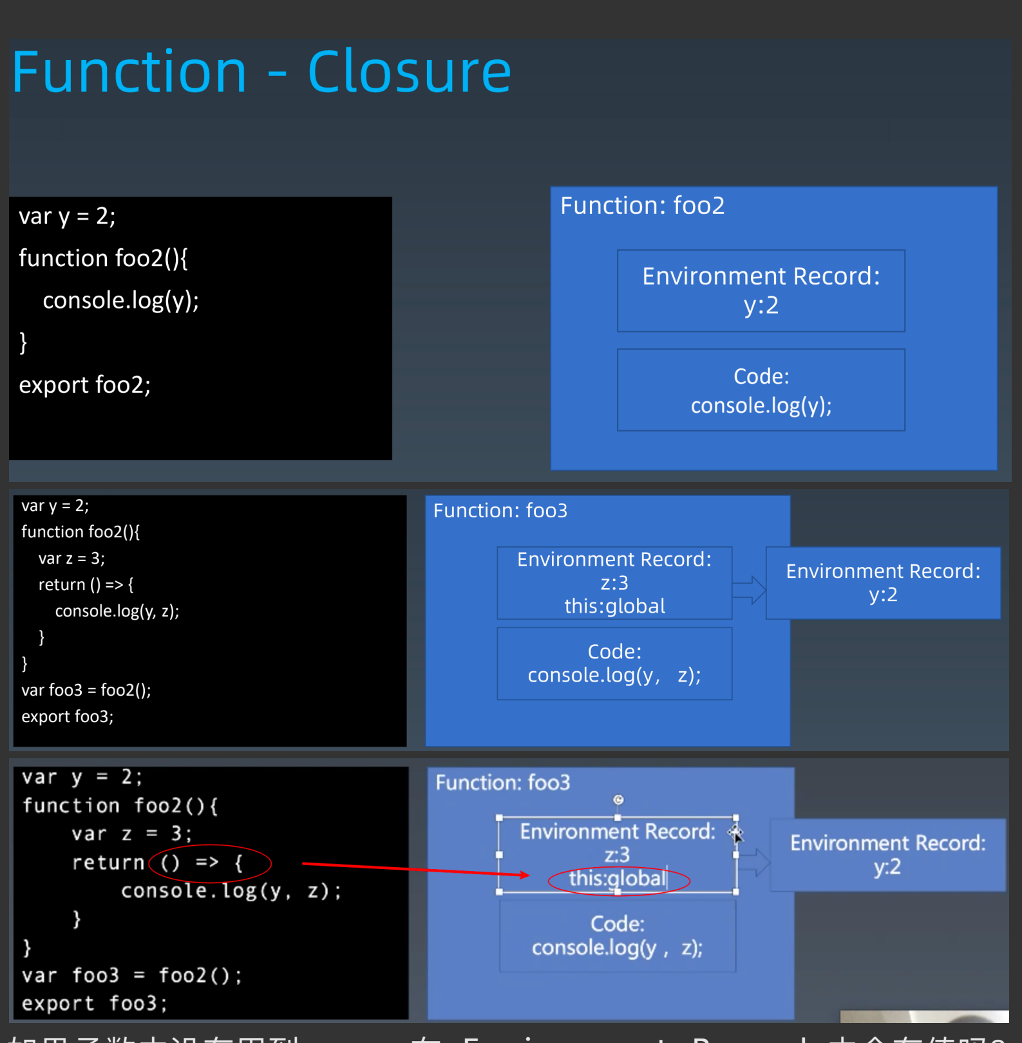
为什么老师说取不到 i? 但是代码在一起是可以获得的。模块化就取不到。

切换环境 => 栈

执行上下文栈，Execution Context Stack
[Execution Context, Execution Context, …, Running Execution Context]

Execution Context: ECMA Code Execution Context + Generator Execution Context:
- code evaluation state，代码执行位置，async/await/generator 用得到
- Function，在 function 中执行
- Script or Module，在 script/module 中初始化
- Generator，generator产生，只在 Generator Execution Context 有
- Realm
- LexicalEnvironment，词法环境，取变量值所需要的环境
    - this
    - new.target
    - super
    - 变量
- VariableEnvironment，变量环境
    - 处理 var 声明，历史遗留包袱

Environment Record
- Global Environment Record
- Object Environment Record
- Declarative Environment Record
    - Function Environment Record
    - module Environment Record



Function - Closure

```
var y = 2;
function foo2(){
   console.log(y);
}
export foo2;
```

```
var y = 2;
function foo2(){
   var z = 3;
   return () => {
      console.log(y, z);
   }
}
var foo3 = foo2();
export foo3;
```

```
var y = 2;
function foo2(){
    var z = 3;
    return () => {
        console.log(y, z);
    }
}
var foo3 = foo2();
export foo3;
```

如果函数中没有用到 y,z，在 Environment Record 中会有值吗?

Realm
什么情况需要 Realm? 字面量创建的对象的原型依赖是哪个 Realm

函数式编程：https://coolshell.cn/articles/17524.html