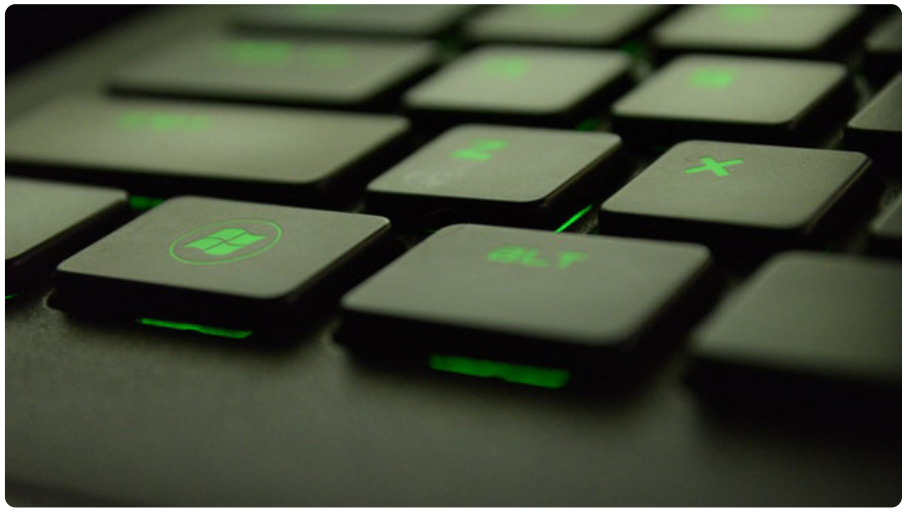
目录

21 Event Loop 力挽狂澜

更新时间：2019-09-03 09:35:03



> 加紧学习，抓住中心，宁精勿杂，宁专勿多。
>
> —— 周恩来

关于 Event Loop 的认知，很多人都是始于一些面试题，的确关于 Event Loop 的高频面试题非常多，那么这里我们就通过一道面试题来引出我们今天的主角Event Loop，如下：

```
console.log(1);
setTimeout(function() {
        console.log(2);
});
console.log(3)
const p = new Promise(function(resolve,reject) {
  console.log(4);
  resolve();
});
console.log(5);
p.then(function(){
  console.log(6)
});
console.log(7);
```

那么它的打印结果是什么，这里先不说答案，大家可以想一想，这道题在面试当中属于高频考题了，相信很多同学都遇到过，其实它背后考察的知识点就是 Event Loop。

Event Loop的背后

我们都知道 JavaScript 是单线程的语言，它在一个时间点只能处理一件事情，完成一件事情之后才可以继续另外一件事情。JavaScript 为了解决这个问题，于是产生了使用异步这种方式来模拟多线程，而支撑异步的就是 Event Loop。下面我就来介绍 Event Loop，关于 Event Loop 整个流程我们来看下面这张图：

如上就是 Event Loop 整个流程图，我们可以看到 Event Loop 当中包含 heap stack Microtask Macrotask。下面我们就来一一介绍这些概念。

- heap

  heap 是堆这种数据结构，堆其实就是我们平时所说的二叉树，这里存放的主要是 JavaScript 当中的对象，也是 Event Loop 当中一个重要的环节。

- stack

  首先 stack 翻译过来的意思就是栈，我们知道栈是一种数据结构，它的特点就是后进先出，所以我们的代码执行的时候，会将代码压入栈中进行执行，当任务完成之后，根据栈后进先出的特点，再将各个任务进行出栈。

  上面是假定没有异步任务的情况下 JavaScript 的执行顺序，当我们遇到异步任务，比如：setTimeout、addEventListener 这类异步任务的时候，这个时候异步任务不会直接被压入到栈中，而是会交给浏览器的Web API 进行维护，当这些异步任务执行完成之后，会在任务队列当中放置对应事件。当执行栈当中任务为空的时候，然后浏览器读取任务队列，再把对应的异步任务压入到执行栈执行，这就是我们经常说的事件循环。整个流程如下图：

那么这张图很好的对应我上面讲述的整个流程，而且一些疑问看了图之后，相信也可以得到一个比较清晰的解答，下面我们来介绍 Macrotask 和 Microtask。

- Macrotask/Microtask

  上面我们提到了任务队列，那么任务队列有两种，一种是 Macrotask，另外一种是 Microtask。从最开始我给出的 Event Loop 总的流程图，大家可以看出 Microtask 优先级是高于 Macrotask 的。Microtask 当中的任务也是在执行栈当中的任务执行完成后再进行执行，执行的时候和 Macrotask 有一些区别，Microtask 当中任务不会一个一个压入执行栈，而是所有任务直接压入栈中，当 Microtask 当中的任务执行完毕后，然后我们再从 Macrotask 中取栈顶的第一个任务进行执行。

Macrotask： `setTimeout` 、 `setInterval` 、 `I/O` 、 `UI Rendering` 、 `script当中的所有代码` 、 `setImmediate(Node)`

Microtask： `process.nextTick(node)` 、 `Promise` 、 `MutationObserver`

> Tips:有一些考察较详细的题目，还会考察 Microtask 当中各种任务的优先级，具体的优先级如下：
>
> process.nextTick > Promise > MutationOberser

其实到这里上面给出的面试题也就有了答案，只要我们熟练掌握 Event Loop 背后的流程，那么再次遇到这类面试题的时候就可以做到胸有成竹了。

扩展

- 在我总结的 Microtask 和 Macrotask 列表当中，我有2个标注了 Node，其实这2个任务在浏览器环境当中并不是支持，只有在 Node 环境当中才可以直接执行。process.nextTick 可以在执行栈的尾部直接触发其对应的回调函数，因此在微任务当中 process.nextTick 的优先级是最高的。

- 我们在上面介绍了 Event Loop 的理论知识，那么我们在平时实际的开发当中，可以用到 Event Loop原理背后的哪些性能优化方法呢，那么这里就要说 Event Loop 与更新渲染时机的关系了。在最开始的时候，Macrotask 当中放置了所有的 script 标签当中的代码，而 Microtask 当中为空。

  当开始执行的时候，首先 script 标签当中的代码出 Macrotask，被压入到执行栈执行，这个时候就会有对应的任务被推入 Macrotask 和 Microtask 当中，上面我们已经说过 Microtask 是所有任务一起执行，而Macrotask 则是任务一个一个执行，那么页面渲染是在 Microtask 之后才进行的，如果我们在异步操作当中进行 DOM 操作，我们尽量将这个操作用 Microtask 当中的任务包一下，这样我们就可以在页面渲染之前就执行这个 DOM 操作了。

  如果你有读过Vue的源码，其实可以发现 Vue 的当中的异步操作都是用 Promise 这个属于 Microtask 的任务来包装的，背后的原因正是我们上面讲到的原理。

小结

这一节我们讲了 Event Loop 整个流程，对于平时一些经常听到的一些 Macrotask、Microtask、执行栈等等这些概念有了更深的了解，相信学完了这一节之后，对于我最上面的面试题大家是否已经有了答案，可在评论区与我进行交流。

← 20 重绘与回流的相爱相杀 22 防抖节流背后那些事儿 →

精选留言 2

目录

火星上的慕老爷

这句话有歧义：setTimeout、addEventListener 这类异步任务的时候，这个时候异步任务不会直接被压入到栈中，而是会交给浏览器的Web API 进行维护，当这些异步任务执行完成之后，会在任务队列当中放置对应事件。 都说异步任务执行完成之后了，还放到任务队列中去干嘛呢?

0 回复 2020-06-03

红遍天下

event loop是先执行同步代码(js代码 IO操作 click)，遇到异步任务 settimeout, promise等就会将异步任务添加到任务队列，等这个事件循环中的同步任务执行完成，再去执行异步。

0 回复 2020-04-10

千学不如一看，千看不如一练