Node.js 押题

Node.js 的 EventLoop 是什么?

浏览器里的微任务和任务是什么?

express.js 和 koa.js 的区别是什么?

Node.js 的 EventLoop 是什么?

背景知识

Event Loop、计时器、nextTick - 掘金

JavaScript 是单线程的，有了 event loop 的加持，Node.js 才可以非阻塞地执行 I/O 操作，把这些操作尽量转移给操作系统来执行。 我们知道大部分现代操作系统都是多线程的，这些操作系统可以在后台执行多个操

https://juejin.cn/post/6844903582538399752

Node.js 将各种函数（也叫任务或回调）分成至少 6 类，按先后顺序调用，因此将时间分为六个阶段：

1. timers 阶段（setTimeout）
2. I/O callbacks 该阶段不用管
3. idle, prepare 该阶段不用管
4. poll 轮询阶段，停留时间最长，可以随时离开。
    a. 主要用来处理 I/O 事件，该阶段中 Node 会不停询问操作系统有没有文件数据、网络数据等
    b. 如果 Node 发现有 timer 快到时间了或者有 setImmediate 任务，就会主动离开 poll 阶段
5. check 阶段，主要处理 setImmediate 任务
6. close callback 该阶段不用管

Node.js 会不停的从 1 到 6 循环处理各种事件，这个过程叫做事件循环（Event Loop）。

nextTick

process.nextTick(fn) 的 fn 会在什么时候执行呢?

在 Node.js 11 之前，会在每个阶段的末尾集中执行（俗称队尾执行）。

在 Node.js 11 之后，会在每个阶段的任务间隙执行（俗称插队执行）。

浏览器跟 Node.js 11 之后的情况类似。可以用 window.queueMicrotask 模拟 nextTick。

Promise

Promise.resolve(1).then(fn) 的 fn 会在什么时候执行?

这要看 Promise 源码是如何实现的，一般都是用 process.nextTick(fn) 实现的，所以直接参考 nextTick。

async / await

这是 Promise 的语法糖，所以直接转为 Promise 写法即可。

面试题1：

```
setTimeout(() => {
  console.log('setTimeout')
})

setImmediate(() => {
  console.log('setImmediate')
})
// 在 Node.js 运行会输出什么?
// A setT setIm
// B setIm setT
// C 出错
// D A 或 B
// 在浏览器执行会怎样?
```

面试题2：

```
async function async1(){
    console.log('1')                    // 2
    async2().then(()=>{
      console.log('2')
    })
}
async function async2(){
    console.log('3')                    // 3
}
console.log('4')                         // 1
setTimeout(function(){
    console.log('5')
},0)
async1();
new Promise(function(resolve){
    console.log('6')                     // 4
    resolve();
}).then(function(){
    console.log('7')
})
console.log('8')                            // 5
//4 1 3 6 8 2 7 5
```

浏览器里的微任务和任务是什么?

浏览器中并不存在宏任务，宏任务（Macrotask）是 Node.js 发明的术语。

浏览器中只有任务（Task）和微任务（Microtask）。

1. 使用 script 标签、setTimeout 可以创建任务。
2. 使用 Promise#then、window.queueMicrotask、MutationObserver、Proxy 可以创建微任务。

执行顺序是怎样的呢?

微任务会在任务间隙执行（俗称插队执行）。

> ⚠️ 注意，微任务不能插微任务的队，微任务只能插任务的队。

面试题:

为什么以下代码的返回结果是 0 1 2 3 4 5 6 而不是 0 1...

Promise.resolve() // then0 .then(() => { console.log(0); return Promise.resolve(4); }) ...

知 https://www.zhihu.com/question/495934384

```
Promise.resolve()
  .then(() => {
      console.log(0);
      return Promise.resolve('4x');
  })
  .then((res) => {console.log(res)})

Promise.resolve().then(() => {console.log(1);})
                 .then(() => {console.log(2);}, ()=>{console.log(2.1)})
                 .then(() => {console.log(3);})
                 .then(() => {console.log(5);})
                 .then(() => {console.log(6);})
```

注意，多个 then 里面的回调并不会一次性插入等待队列中，而是执行完一个再插入下一个。

express.js 和 koa.js 的区别是什么?

1. 中间件模型不同：express 的中间件模型为线型，而 koa 的为U型（洋葱模型）。

2. 对异步的处理不同：express 通过回调函数处理异步，而 koa 通过generator 和 async/await 使用同步的写法来处理异步，后者更易维护，但彼时 Node.js 对 async 的兼容性和优化并不够好，所以没有流行起来。
3. 功能不同：express 包含路由、渲染等特性，而 koa 只有 http 模块。

总得来说，express 功能多一点，写法烂一点，兼容性好一点，所以当时更流行。虽然现在 Node.js 已经对 await 支持得很好了，但是 koa 已经错过了风口。

不过 express 和 koa 的作者都是 TJ 大神。