libuv → event loop

Promises

Nodejs → official docs

① ticks → one full trip of event loop.

Single iteration

Promises ticks timers I/O callback poll check close

process.nextTick → this func" takes a callback.

By calling process.nextTick we instruct node to invoke this cb func" at the end of curret operation before the next tick starts.

globals

process.env

process

require

module

fs

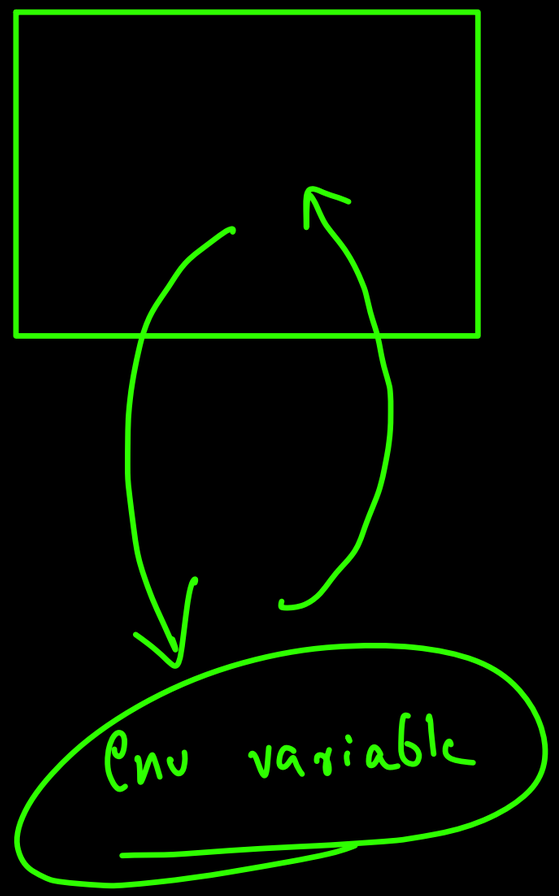
file system

env variables

diff properties assoc to your own process.

module pathway

code → process → env variable


Env variable

↳ 1) → execution of nextTick callbacks.

↳ all of the cb are added one by one in the nextTick queue.

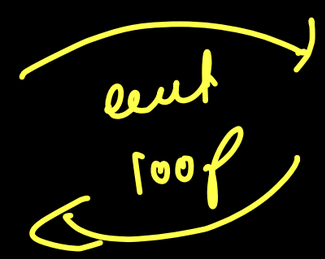
↳ whatever cb are there in the nextTick queue are one by one executed.

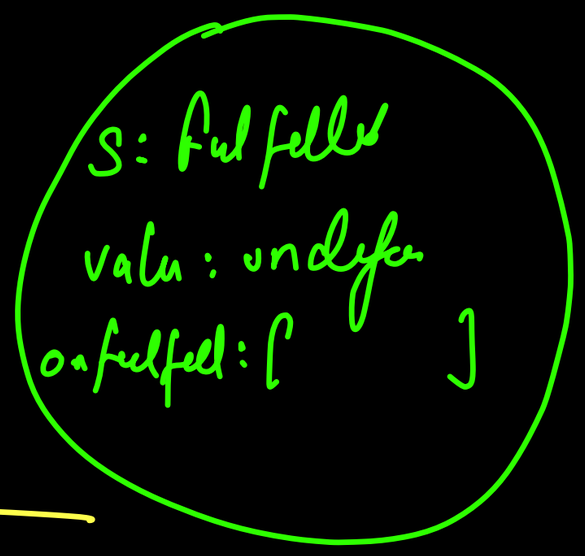
Priority nextTick queue > Priority Microtask queue.

```
a=10
Promise.resolve().then(() => ____);
  ↳ process.nextTick(() => ____);
console.log(a)
```

cb1 cb2 cb3

10





→ The first iteration of event loop starts when your current set of instructions are done (main thread is done), and nextTick queue is empty, then the iteration of event loop start.

1. nextTick Queue will be resolved
2. Event loop tick (iteration starts)
3. Microtask queue / Promise queue (the queue who promise cb are present) starts exec.

4. Timers queue (contain all the callbacks of timers)
Starts exec

Start
Next Pick 1
Promise

```js
console.log("Start");

setTimeout(() => {
    process.nextTick(() => console.log("Next Tick 2")); // cb will be waiting in nextTick queue
}, 0); // cb will be waiting in timer queue

Promise.resolve().then(() => console.log("Promise 1")); // cb will be waiting in promise queue

process.nextTick(() => console.log("Next Tick 1")); // cb will be waiting in nextTick queue

setTimeout(() => console.log("Timer 2"), 0); // cb will be waiting in timer queue
```

cb1 cb5 cb2 cb3 cb4

Start → N → T → 1

Node.js → libuv

Timer → 1ms

Timer → 1ms

Cb3 Cb5

cb2

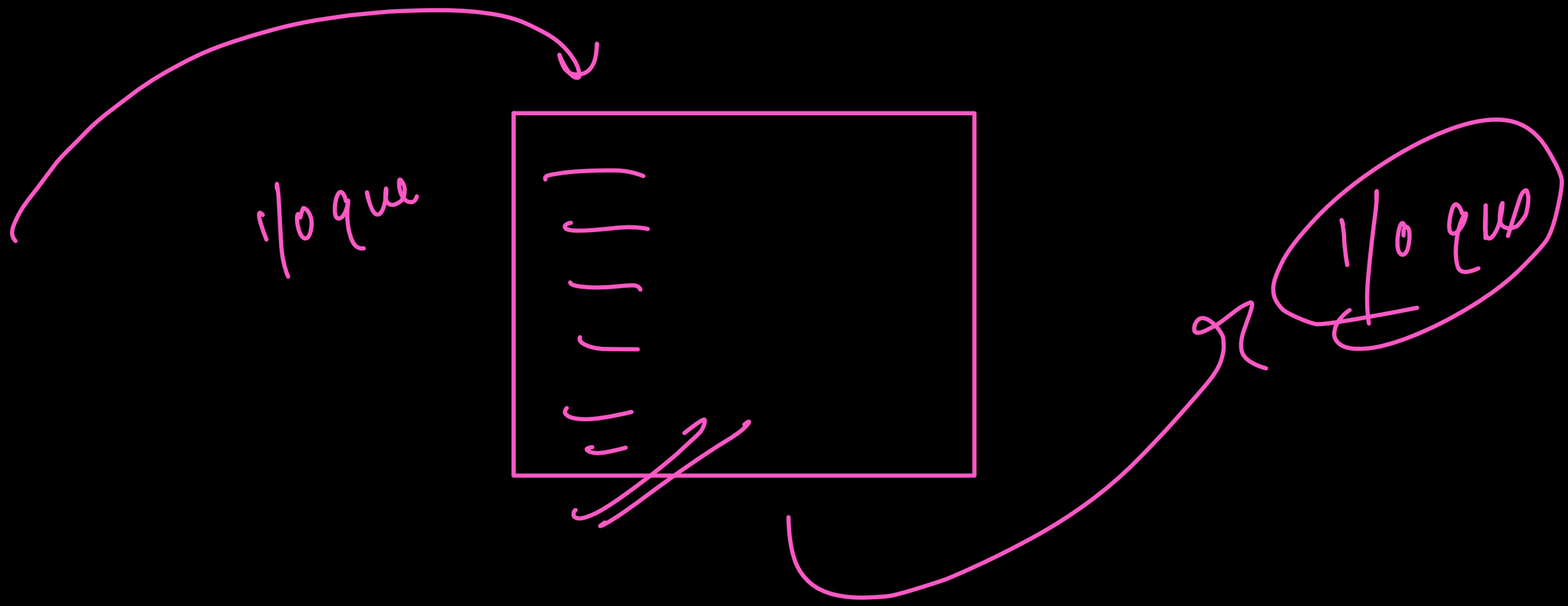
Cb1

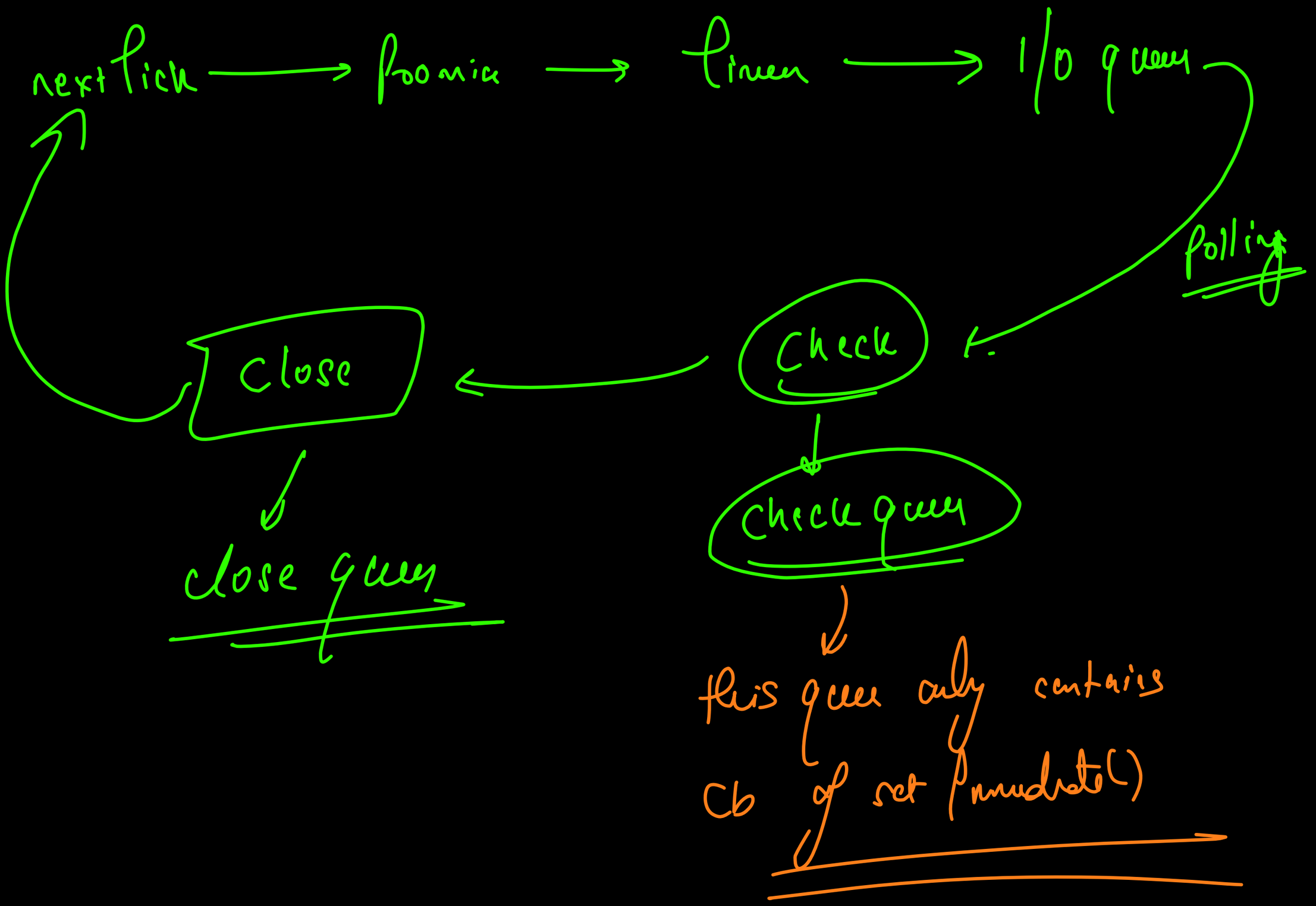
Next tick Queue

Promise queue

Timer queue

1. Any cb in next tick queue
2. Any cb in Promise queue
3. We one by one execute cb in Timer queue. But after every cb exec, we check step 1 and 2.
4. → Pending callback → I/O queue or callback queue





```js
const fs = require('fs');

fs.readFile('./readme.md', 'utf8', (err, data) => { // I/O queue - Callback queue
    if (err) {
      console.error(err);
      return;
    }
    console.log(data);
});

process.nextTick(() => console.log("Next tick"));
Promise.resolve(() => console.log("Promise 1"));
setTimeout(() => console.log("Timer 1"), 0);
setImmediate(() => console.log("Immediate callback"));

for(let i = 0 ; i < 10000000000; i++ ) {} // > 5s
```

cb5

cb1

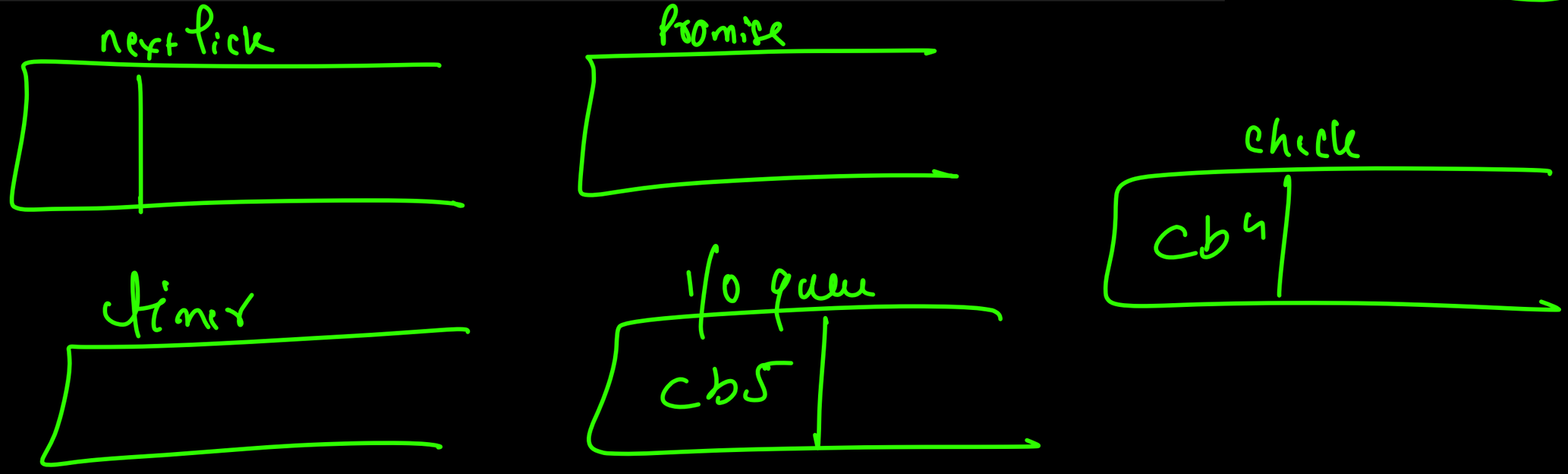
cb2

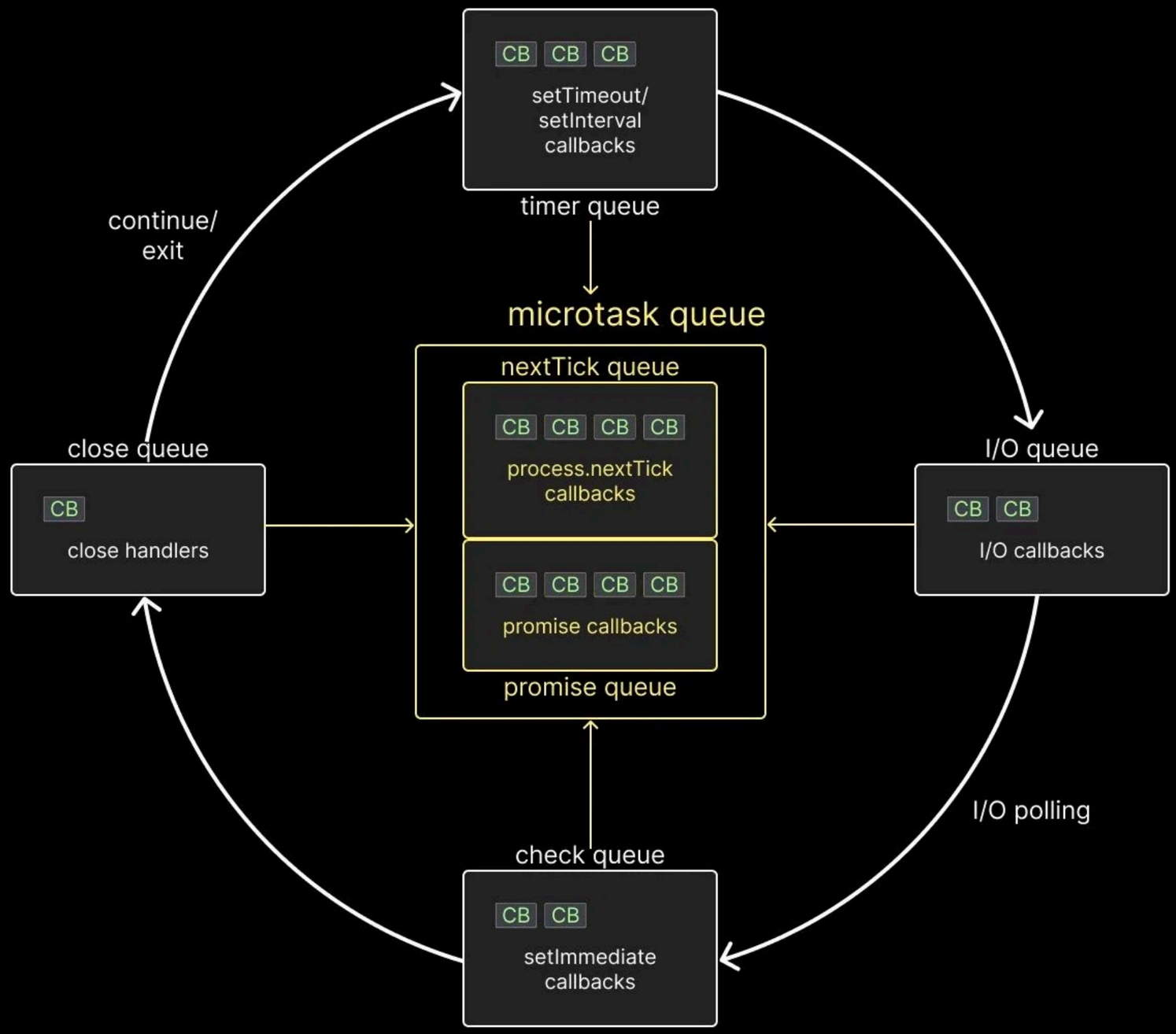
cb3

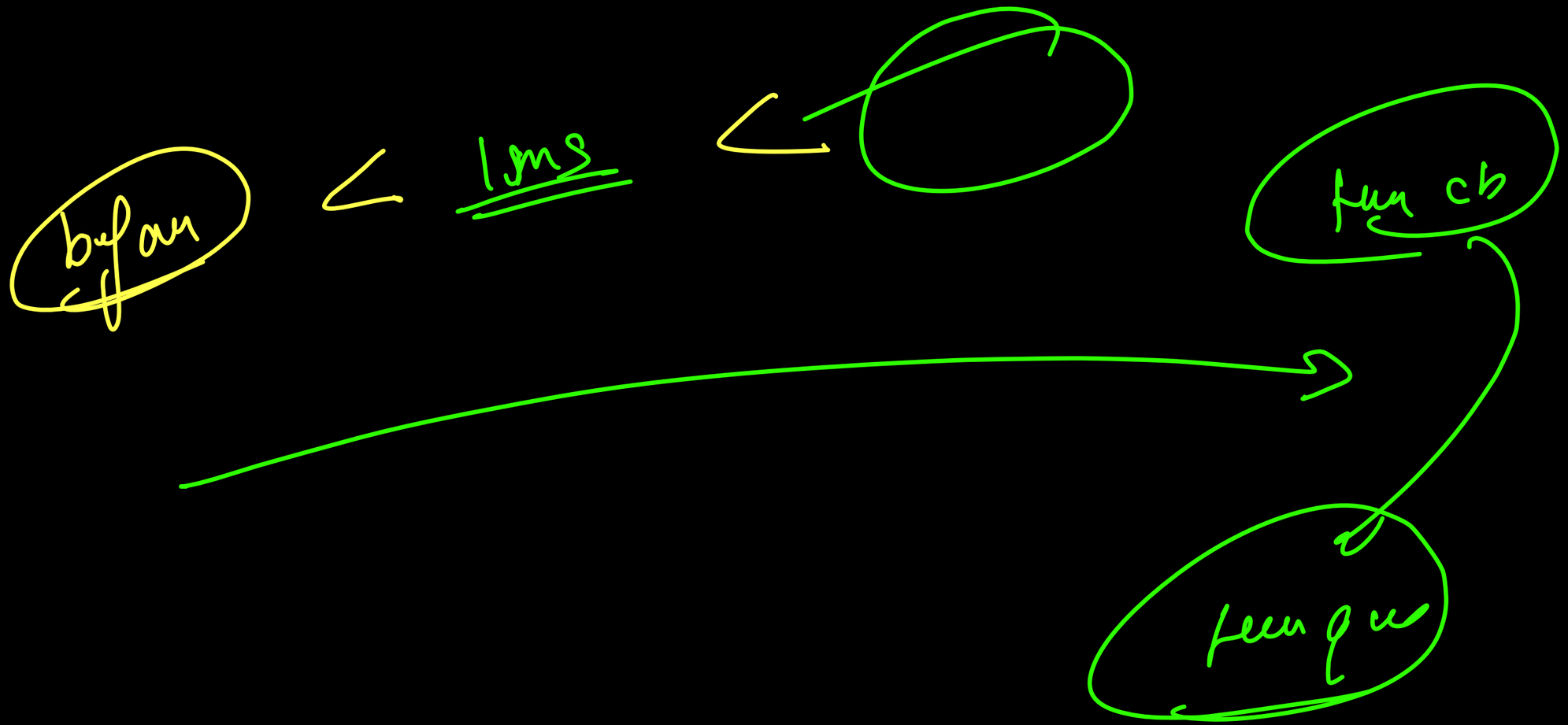
cb4

libuv
node
libev

fulfilled
undefined
[ ]







1 ms

0.003 ms

Real

0.004 ms

1 ms

$1\,S \rightarrow 10^8$

$1000\,ms \rightarrow 10^8$

$1\,ms \rightarrow 10^5$

11 am