1. How many callbacks queues in Node.js and what are they? Give examples to explain how different callbacks are enqueued in different queues.

There are 6 different queues in every loop:

* + Timers: this phase executes callbacks scheduled by setTimeout() and setInterval().

setTimeout(() => {

console.log(‘Hello World with 1 second delay’);

}, 1000);

* + I/O: this phase contains callbacks from I/O operations, such as reading, writing from a file

const fs = require('fs');

fs.readFile(‘helloworld.txt', 'utf8', (err, data) => {

console.log('This callback is from the I/O queue.');

});

* + Check: this phase contains callbacks that are scheduled to execute immediately after the current event loop cycle like setImmediate()

setImmediate(() => console.log('Immediate'));

* + close: this is a special queue that ensures a callback is executed at the beginning of the next event loop cycle, after the current operation completes.
  + Microtask: includes nextTick queue and promise queue
    - nextTick: this is a special queue that ensures a callback is executed at the beginning of the next event loop cycle, after the current operation completes.

process.nextTick(() => {

console.log('This callback is from the next tick queue.');

});

* + - Promise: when a promise is settled (either resolved or rejected), its associated callbacks are scheduled as microtasks.

Promise.resolve('Resolved Promise')

.then(value => console.log(value));

1. Answer

start

end

nextTick 1

nextTick 2

Promise...1

Promise...2

timeout 1

timeout 2

nextTick 3

timeout 3

console.log is synchronous so it will executed and printed out first followed by

"start" "end".

setTimeout() was added to Timers queue and wait til call back

new Promise was add to Promise queue (MicroTask queue) and wail til call back

process.nextTick() was add to nextTick queue (MicroTask queue) and wail til call back

process.nextTick() has highest priority (higher than promise in the same Microtask Queue).

So console continue to call back and execute: nextTick first then Promise.

Then, it will continue to call back and execute setTimeout() => timeout 1

With the second setTimeout(), it will continue to apply the same rule with synchronous console.log

and print nextTick, then finish second setTimeout()

Finally, console print => timeout 3

1. Answer

timeout

Immediate

readFile...

The order following queue position in event loop, start with Timers (setTimeout()), then readFile() should be add into I/O, but because of readFile() must go through Polling to check complete or not, so I/O queue at this time is empty and setImmediate() was printed while readFile() was checking.

And after another loop, readFile() completed and added into I/O queue then executed later.

1. Answer

=> The output will be "undefined" because of "this" belongs to global and "message" when using node, it has

a wrapper, so that "message" can not expose to global.

If the same code running in browser, the output will be "Hello World" because we don't have a wrapper,

message will add into the window object, and it can successfully call.

1. Answer

It will throw error: getName is not a function.

Because getName in app.js is an objecct, so it can not call by function

1. Answer

It will throw error: Cannot find module './pattern2'

Because that file path is not exist (or couldn't find)

But if we pass: const getName = require('./pattern1');

It will work because module.export is a function so we can use getName() to call it.

1. Answer

It will print: Josh Edward

Because in const {getFullname} = require('./pattern1');

Object returned by require is destructured using { getFullname }

that only the getFullname property from the exported object is

assigned to the local variable getFullname.

1. Answer

{ firstname: 'John', lastname: 'Smith' }

because instead of coding above, wrapper already implicit have wrapper function

and use apply to call.