

search



7 Hardest Node.js Interview Questions & Answers



Alex FullStack.Cafe Jul 18 '18 Updated on Sep 05, 2018 · 4 min read

#node #interview #questions #fullstack



Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code server-side. Node.js lets developers use JavaScript for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser.



80



18



139





search

separation for dev and prod environments

Topic: **Node.js**

Difficulty: ★ ★ ★ ★

A perfect and flawless configuration setup should ensure:

- keys can be read from file AND from environment variable
- secrets are kept outside committed code
- config is hierarchical for easier findability

Consider the following config file:

```
var config = {
  production: {
    mongo : {
      billing: '****'
    }
  },
  default: {
    mongo : {
      billing: '****'
    }
  }
}

exports.get = function get(env) {
  return config[env] || config.default;
}
```



80



18



139



search



```
const config = require('./config/config.js').get(process.env.NODE_ENV);  
const dbconn = mongoose.createConnection(config.mongo.billing);
```

Source: github.com/i0natan/nodebestpractices

Q2: What are the timing features of Node.js?

Topic: **Node.js**

Difficulty: ★ ★ ★ ★

The Timers module in Node.js contains functions that execute code after a set period of time.

- **setTimeout/clearTimeout** - can be used to schedule code execution after a designated amount of milliseconds
- **setInterval/clearInterval** - can be used to execute a block of code multiple times
- **setImmediate/clearImmediate** - will execute code at the end of the current event loop cycle
- **process.nextTick** - used to schedule a callback function to be invoked in the next iteration of the Event Loop

```
function cb(){  
  console.log('Processed in next iteration');
```



80



18



139



search



Output:

Processed in the first iteration

Processed in next iteration

Source: github.com/jimuyouyou

Q3: Explain what is Reactor Pattern in Node.js?

Topic: **Node.js**

Difficulty: ★ ★ ★ ★ ★

Reactor Pattern is an idea of non-blocking I/O operations in Node.js. This pattern provides a handler(in case of Node.js, a *callback function*) that is associated with each I/O operation. When an I/O request is generated, it is submitted to a *demultiplexer*.

This *demultiplexer* is a notification interface that is used to handle concurrency in non-blocking I/O mode and collects every request in form of an event and queues each event in a queue. Thus, the demultiplexer provides the *Event Queue*.



80



18



139



search



items in the Event Queue. Every event has a callback function associated with it, and that callback function is invoked when the Event Loop iterates.

Source: hackernoon.com

Q4: What is LTS releases of Node.js why should you care?

Topic: **Node.js**

Difficulty: ★ ★ ★ ★ ★

An **LTS(Long Term Support)** version of Node.js receives all the critical bug fixes, security updates and performance improvements.

LTS versions of Node.js are supported for at least 18 months and are indicated by even version numbers (e.g. 4, 6, 8). They're best for production since the LTS release line is focussed on stability and security, whereas the *Current* release line has a shorter lifespan and more frequent updates to the code. Changes to LTS versions are limited to bug fixes for stability, security updates, possible npm updates, documentation updates and certain performance improvements that can be demonstrated to not break existing applications.



80



18



139





search

server :

Topic: **Node.js**

Difficulty: ★ ★ ★ ★ ★

Keeping the API declaration separated from the network related configuration (port, protocol, etc) allows testing the API in-process, without performing network calls, with all the benefits that it brings to the table: fast testing execution and getting coverage metrics of the code. It also allows deploying the same API under flexible and different network conditions. Bonus: better separation of concerns and cleaner code.

API declaration, should reside in app.js:

```
var app = express();
app.use(bodyParser.json());
app.use("/api/events", events.API);
app.use("/api/forms", forms);
```

Server network declaration, should reside in /bin/www:

```
var app = require('../app');
var http = require('http');
```

/**



80



18



139



search



```
app.set('port', port);

/**
 * Create HTTP server.
 */

var server = http.createServer(app);
```

Source: github.com/i0natan/nodebestpractices

Q6: What is the difference between process.nextTick() and setImmediate() ?

Topic: **Node.js**

Difficulty: ★ ★ ★ ★ ★

The difference between `process.nextTick()` and `setImmediate()` is that `process.nextTick()` defers the execution of an action till the next pass around the event loop or it simply calls the callback function once the ongoing execution of the event loop is finished whereas `setImmediate()` executes a callback on the next cycle of the event loop and it gives back to the event loop for executing any I/O operations.

Source: codingdefined.com

Q7: Rewrite the code sample without try/catch



80



18



139



search



Consider the code:

```
async function check(req, res) {  
  try {  
    const a = await someOtherFunction();  
    const b = await somethingElseFunction();  
    res.send("result")  
  } catch (error) {  
    res.send(error.stack);  
  }  
}
```

Rewrite the code sample without try/catch block.

Answer:

```
async function getData(){  
  const a = await someFunction().catch((error)=>console.log(error));  
  const b = await someOtherFunction().catch((error)=>console.log(error));  
  if (a && b) console.log("some result")  
}
```

or if you wish to know which specific function caused error:



80



18



139



search



```
    throw new CustomErrorHandler({
      code: 101,
      message: "a failed",
      error: error
    })
  });
const b = await someUtil().
catch((error) => {
  throw new CustomErrorHandler({
    code: 102,
    message: "b failed",
    error: error
  })
});
//someoeoe
if (a && b) console.log("no one failed")
} catch (error) {
  if (!(error instanceof CustomErrorHandler)) {
    console.log("gen error", error)
  }
}
}
```

Source: medium.com

Thanks 🙌 for reading and good luck on your interview!
Check more FullStack Interview Questions & Answers on www.fullstack.cafe



Alex  FullStack.Cafe + FOLLOW

👋 Product enthusiast. FullStack Dev. Au Currently working on: www.FullStack.Cafe



80



18



139



search



PREVIEW

SUBMIT



Jesse Ditson



Jul 19 '18

I'm glad folks are posting interview questions on here, we can all do better when it comes to crafting them. In light of that, i've noticed a pattern of asking questions like "what is the reactor pattern" that I personally think are pretty counter productive. Asking about industry terminology rather than the application of said concepts locks out non-industry (e.g. self-taught) folks, who are otherwise likely going to be some of your best candidates!

Thank you for the post, some other questions on here are quite good. Hopefully discussions like this can help us all hire the very best people.



8

REPLY



Nahuel



Jul 19 '18

Thank you, I can totaly relate to this, I've working with NodeJS self-taught for almost 3 years now, and I've never read anything about "reactor pattern"... Had no idea that that, had a name.



2

REPLY



Janne "Lietu" Enberg



Jul 20 '18

in your Q7 the "solution" does not perform the same actions as the first one, you're ignoring the arguments, and using console.log instead of res.send .. I'd fail you for that answer

also LTS is not a Node.js term, it's a generic term .. you will find it in use in linux distros and many other places

oh and I massively disagree on these difficulty ratings



80



18



139



search



Indeed Q7 answer's is totally wrong and don't do the same as the first version.
if "someOtherFunction" throw it will print and continue with execution of
"somethingElseFunction".

The second answer is even more wrong as it includes a try/catch



REPLY




Leandro Rodrigues Jul 19 '18 

Hi Guys,

Quick question - At Q1 what do you mean with "secrets are kept outside committed code". Could you provide an example?



REPLY


Janne "Lietu" Enberg  Jul 20 '18 

it means you don't put your mysql password, paypal login information, etc. in your version control so it's basically safe to show your version control to someone without them getting access to your accounts or servers

how you do it, is by e.g. creating a `config.local.*` -file that you never store in the version control and generate for each environment, or by reading some variables from the environment, or some such .. often you will want to use a system like Kubernetes, SaltStack, Chef, or some such to manage these secrets and their deployments securely



REPLY

André Costa Lima   Jul 20 '18 

I agree, but the example posted is not consistent with what you said above or it is somewhat misleading. The example appears to demonstrate storage of sensitive configuration data in a JS module which is being required in the code. The best practice is to read configuration from environment variables directly, no?



80



18



139



search



I have no idea where you came up with this "best practice" from, it's one way to deal with it. The file is just fine as well as long as you don't store it in version control, i.e. add it to your `.gitignore` or similar.

If someone gets on the server they can read the process environment just as well as the file.



THREAD



Memoria

Aug 8 '18

Being able to pass secret by environment variable is a MUST and part of the 12 factors. 12factor.net/config

You also need to commit your config file otherwise new people needs to understand how to works.

You could have a `config.default` but then it means you will have to maintain it and deal with optional configuration.



THREAD



Janne "Lietu" Enberg

Aug 11 '18

You quote the "12 factors" as if it is your holy book, that alone is something to worry about.

Also the only real point that page has is:

A litmus test for whether an app has all config correctly factored out of the code is whether the codebase could be made open source at any moment, without compromising any credentials.

That doesn't mean or even hint at the need for environment variables, simply that your secrets and similar configuration isn't committed to your source in a human readable format. You CAN even commit your secrets and still pass that check, if you encrypt them so they can only be decrypted in the environment they're deployed to (often employed strategy e.g. when working with Salt Stack).

There is no reason secrets "must" be passed by environment variables. That might be a



80



18



139



search



What I do often is commit a config that works for dev environment with a filename like `config.example.*`, and then simply deliver another config file for other deployments.

Additional concern is making sure your devs will have to spend minimal effort on fixing their environment, so it's even regularly worth symlinking or otherwise automating the use of your dev config in dev envs in a way that it doesn't accidentally get used on other environments if the config isn't properly created there.



1

THREAD



Janne "Lietu" Enberg



Aug 11 '18

One fun thing is when you specifically work with languages such as Python. If you have a file called `settings.py` which contains your settings coding against it is super easy from `settings` import `PAYPAL_USER`, and reconfiguring your app during unit testing is super convenient as you can just monkey patch the module.

This file can additionally of course have logic that reads environment variables into it, something like:

```
# settings.py
import sys
import os

PAYPAL_USER = "foo"
# ...

_l = dir(sys.modules[__name__])
for _key in os.environ:
    if _key in _l:
        setattr(sys.modules[__name__], _key, os.environ[_key])
```



1

THREAD



Memoria

Aug 11 '18

Regarding the `config.example` issue.

It is recognized as a bad practice because you're creating a file that is not requested anywhere and waiting to rot.



80



18



139



search



because everything will still work the way it is.

If you've ever work on projects with a big team either enterprise or open source. Many of projects using config.example simply doesn't work out of the box because of the config.example being just a piece of lies.

As for 12factors, well the 3rd point, says explicitly:

III. Config

Store config in the environment

I've mentioned the 12factors since the OP already mentioned it in another of his post.

Now without it being the holy grail or anything, it's a set of "best" practices that most if not nearly all recent IT books are derived from when they're talking about production ready code. Not having heard of it in 2018 is something to worry about.

@acostalima was asking if env was a best practice and you cannot deny him that.

You surely could do another way, a best practice doesn't mean the "one and only".

As you say is depends on your way of deployment. For example, if you're running on cloud providers like AWS, then use of secret manager makes a lot of sense for any secrets config.



1

THREAD



Janne "Lietu" Enberg



Aug 11 '18

Your devs will use their config.dev which is not committed

Um no, don't do that. Use a config that IS committed as the developer environment config, so when you add something to it you automatically set the correct value for all other devs.

Anyway, there is little point in continuing this.



80



18



139



search



Thanks you both for your insights. □



REPLY



Tomcy John

Jul 21 '18 ■■■

You could and should use a secret management external product line hashicorp vault and manage secrets there. From node use vault to retrieve the secrets as and when needed. This is one of the best approach that I know off.



REPLY



AsyncAwait

Jul 20 '18 ■■■

In the last example since there is now dependency in async functions, why not just:

```
const check = (req, res) =>
  Promise.all([someOtherFunction(), somethingElseFunction()])
    .then(() => res.send('result'))
    .catch((error) => res.send(error.stack))
```



REPLY



Memoria

Aug 8 '18 ■■■

only if "somethingElseFunction" does not depends on the result of "someOtherFunction"



REPLY



Jeremy Short

Nov 4 '18 ■■■

This is not correct, because in your code, both functions will run concurrently, whereas in the original code, they run sequentially.



80



18



139



search



Classic DEV Post from Jul 2 '18

A meaningful README.md



Pratik Ambani

Let's discuss something nobody cares about.



143



5

Another Post You Might Like

How to debug Node.js in a Docker container



Alex Barashkov

More and more teams are moving their development environments to Docker contain...



278



8

Another Post You Might Like

Creating a text editor in Electron: part 2 - writing files



aurel kurtula

creating a text editor with electron



50



5



80



18



139



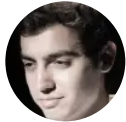
search

**DEV Weekly Episode 37 - javascript news podcast**

Tim Ermilov - Apr 7

**JavaScript: Armadilhas do Async/Await em loops**

Eduardo Rabelo - Apr 8

**Building My First Alexa Skill with Node.js**

Jonathan Brizio - Apr 7

**Web server basics: HTTP and sockets**

Phil Eaton - Apr 8

[Home](#) [About](#) [Privacy Policy](#) [Terms of Use](#) [Contact](#) [Code of Conduct](#)DEV Community copyright 2016 - 2019 

80



18



139

