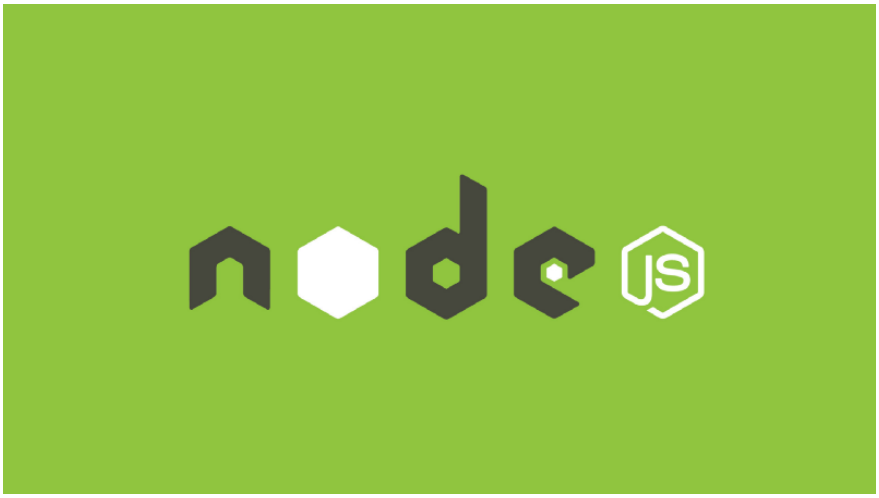
Questions and Answers

Vigo Webs Follow
Dec 10, 2017 · 10 min read



Q1. What is Node.js? What is it used for?

Node.js is a run-time JavaScript environment built on top of Chrome's V8 engine. It uses an event-driven, non-blocking I/O model. It is lightweight and so efficient. Node.js has a package ecosystem called **npm**.

Node.js can be used to build different types of applications such as web application, real-time chat application, REST API server etc. However, it is mainly used to build network programs like web servers, similar to PHP, Java, or ASP.NET. Node.js was developed by Ryan Dahl in 2009.

Q2. What is Event-driven programming?

Event-driven programming is building our application based on and respond to events. When an event occurs, like click or keypress, we are running a callback function which is registered to the element for that event.

Event driven programming follows mainly a publish-subscribe pattern.

```
function addToCart(productId){
  event.send("cart.add", {id: productId});
}

event.on("cart.add", function(event){
  show("Adding product " + event.id);
});
```

Q3. What is *Event loop* in Node.js work? And How does it work?

The *Event loop* handles all async callbacks. Node.js (or JavaScript) is a single-threaded, event-driven language. This means that we can attach listeners to events, and when a said event fires, the listener executes the callback we provided.

Whenever we are call `setTimeout`, `http.get` and `fs.readFile`, Node.js runs this operations and further continue to run other code without waiting for the output. When the operation is finished, it receives the output and runs our callback function.

So all the callback functions are queued in an loop, and will run one-by-one when the response has been received.

Q4. What is REPL in Node.js?

REPL means Read-Eval-Print-Loop. It is a virtual environment that comes with Node.js. We can quickly test our JavaScript code in the Node.js REPL environment.

To launch the REPL in Node.js, just open the command prompt and type `node`. It will change the prompt to `>` in Windows and MAC.

Now we can type and run our JavaScript easily. For example, if we type `10 + 20`, it will print `30` in the next line.

Q5. What is the purpose of `module.exports` in Node.js?

A module encapsulates related code into a single unit of code. This can be interpreted as moving all related functions into a file. Imagine that we created a file called `greetings.js` and it contains the following two functions:

```
module.exports = {
  sayHelloInEnglish: function() {
    return "HELLO";
  },

  sayHelloInSpanish: function() {
    return "Hola";
  }
};
```

In the above code, `module.exports` exposes two functions to the outer world. We can import them in another file as follow:

```
var greetings = require("./greetings.js");

greetings.sayHelloInEnglish(); // Hello

greetings.sayHelloInSpanish(); //Hola
```

Q6. What is the difference between Asynchronous and Non-blocking?

Asynchronous literally means not synchronous. We are making HTTP requests which are asynchronous, means we are not waiting for the server response. We continue with other block and respond to the server response when we received.

The term Non-Blocking is widely used with IO. For example non-blocking read/write calls return with whatever they can do and expect caller to execute the call again. Read will wait until it has some data and put calling thread to sleep.

Q7. What is Tracing in Node.js?

Tracing provides a mechanism to collect tracing information generated by V8, Node core and userspace code in a log file. Tracing can be enabled by passing the `--trace-events-enabled` flag when starting a Node.js application.

```
node --trace-events-enabled --trace-event-categories v8,node server.js
```

The set of categories for which traces are recorded can be specified using the `--trace-event-categories` flag followed by a list of comma separated category names. By default the `node` and `v8` categories are enabled.

Running Node.js with tracing enabled will produce log files that can be opened in the `chrome://tracing` tab of Chrome.

Q8. How will you debug an application in Node.js?

Node.js includes a debugging utility called `debugger` . To enable it start the Node.js with the `debug` argument followed by the path to the script to debug.

Inserting the statement `debugger;` into the source code of a script will enable a breakpoint at that position in the code:

Q9. Difference between `setImmediate()` vs `setTimeout()`

`setImmediate()` and `setTimeout()` are similar, but behave in different ways depending on when they are called.

- `setImmediate()` is designed to execute a script once the current poll (event loop) phase completes.
- `setTimeout()` schedules a script to be run after a minimum threshold in ms has elapsed.

The order in which the timers are executed will vary depending on the context in which they are called. If both are called from within the main module, then timing will be bound by the performance of the process.

Q10. What is `process.nextTick()`

`setImmediate()` and `setTimeout()` are based on the event loop. But `process.nextTick()` technically not part of the event loop. Instead, the `nextTickQueue` will be processed after the current operation completes, regardless of the current phase of the event loop.

Thus, any time you call process.nextTick() in a given phase, all callbacks passed to process.nextTick() will be resolved before the event loop continues.

Q11. What is `package.json` ? What is it used for?

This file holds various metadata information about the project. This file is used to give information to `npm` that allows it to identify the project as well as handle the project's dependencies.

Some of the fields are: `name` , `name` , `description` , `author` and `dependencies` .

When someone installs our project through `npm` , all the dependencies listed will be installed as well. Additionally, if someone runs `npm install` in the root directory of our project, it will install all the dependencies to `./node_modules` directory.

Q12. What is `libuv` ?

`libuv` is a multi-platform support library with a focus on asynchronous I/O. It was primarily developed for use by Node.js, but it’s also used by Luvit, Julia, pyuv, and others.

When the node.js project began in 2009 as a JavaScript environment decoupled from the browser, it is using Google’s V8 and Marc Lehmann’s `libev` , node.js combined a model of I/O – evented – with a language that was well suited to the style of programming; due to the way it had been shaped by browsers. As node.js grew in popularity, it was important to make it work on Windows, but libev ran only on Unix. `libuv` was an abstraction around libev or IOCP depending on the platform, providing users an API based on libev. In the node-v0.9.0 version of libuv libev was removed.

Some of the features of `libuv` are:

- Full-featured event loop backed by epoll, kqueue, IOCP, event ports.
- Asynchronous TCP and UDP sockets
- Asynchronous file and file system operations
- Child processes
- File system events

Q13. What are some of the most popular modules of Node.js?

There are many most popular, most starred or most downloaded modules in Node.js. Some of them are:

- express
- async
- browserify
- socket.io
- bower
- gulp
- grunt

Q14. What is `EventEmitter` in Node.js?

All objects that emit events are instances of the `EventEmitter` class. These objects expose an `eventEmitter.on()` function that allows one or more functions to be attached to named events emitted by the object.

When the `EventEmitter` object emits an event, all of the functions attached to that specific event are called *synchronously*.

```
const events = require('events');
const eventEmitter = new events.EventEmitter();

let myEvent = function ringBell() {
  console.log('Event is emitted');
}

eventEmitter.on('emitEvent', myEvent);

eventEmitter.emit('emitEvent');
```

Q15. What is `Streams` in Node.js?

Streams are pipes that let you easily read data from a source and pipe it to a destination. Simply put, a stream is nothing but an `EventEmitter` and implements some specials methods. Depending on the methods implemented, a stream becomes Readable, Writable, or Duplex (both readable and writable).

For example, if we want to read data from a file, the best way to do it from a stream is to listen to data event and attach a callback. When a chunk of data is available, the readable stream emits a data event and your callback executes. Take a look at the following snippet:

```
var fs = require('fs');
var readableStream = fs.createReadStream('textFile.txt');
var fileData = '';

readableStream.on('data', function(chunk) {
  data += chunk;
});

readableStream.on('end', function() {
  console.log(data);
});
```

Types of streams are: Readable, Writable, Duplex and Transform.

Q16. What is the difference between `readFile` vs `createReadStream` in Node.js?

readFile—is for asynchronously reads the entire contents of a file. It will read the file completely into memory before making it available to

the User. `readFileSync` is synchronous version of `readFile` .

createReadStream—It will read the file in chunks of the default size 64 kb which is specified before hand.

Q17. What is `crypto` in Node.js? How do you cipher the secured information in Node.js?

The `crypto` module in Node.js provides cryptographic functionality that includes a set of wrappers for OpenSSL's hash, HMAC, cipher, decipher, sign and verify functions.

```
var crypto = require('crypto');

var secret = 'abcdefg';
var hash = crypto.createHmac('sha256', secret)
  .update('I love cupcakes')
  .digest('hex');
console.log(hash);

  //   c0fa1bc00531bd78ef38c628449c5102aeabd49b5dc3a2a516ea6ea959d6658e
```

Q18. What is the use of Timers is Node.js?

The Timers module in Node.js contains functions that execute code after a set period of time. Timers do not need to be imported via `require()` , since all the methods are available globally to emulate the browser JavaScript API.

The Node.js API provides several ways of scheduling code to execute at some point after the present moment. The functions below may seem familiar, since they are available in most browsers, but Node.js actually provides its own implementation of these methods.

Node.js Timer provides `setTimeout()` , `setImmediate()` and `setInterval` .

Q19. What is the use of DNS module in Node.js?

`dns` module which provide underlying system's name resolution and DNS look up facilities. DNS module consists of an asynchronous network wrapper.

The most commonly used functions in DNS module are:

- `dns.lookup(adress, options, callback)` - The dns lookup method takes any website address as its first parameter and returns the corresponding first IPV4 or IPV6 record. The options parameter can be an integer or object. If no options are provided both IPV4 and IPV6 are valid inputs. The third parameter is the callback functions.
- `dns.lookupservice(address, port, callback)` - This function converts any physical address such as “www.knowledgehills.com” to array of record types. The record types are specified by the second parameter “rrbyte”. Finally the third method is the callback function.
- `dns.getServers()` - This function returns an array of IP address strings, formatted according to rfc5952, that are currently configured for DNS resolution. A string will include a port section if a custom port is used.
- `dns.setServers()` - This function sets the IP address and port of servers to be used when performing DNS resolution. The `dns.setServers()` method must not be called while a DNS query is in progress.

Q20. What is a Callback function in Node.js?

Node.js, being an asynchronous platform, doesn’t wait around for things like file I/O to finish—Node.js uses callbacks. A callback is a function called at the completion of a given task; this prevents any blocking, and allows other code to be run in the meantime.

```
function processData (callback) {
  fetchData(function (err, data) {
    if (err) {
      console.log("An error has occured. Abort everything!");
      callback(err);
    }
    data += 1;
    callback(data);
  });
}
```

Callbacks are the foundation of Node.js. Callbacks give us an interface with which to say, “and when you’re done doing that, do all this.” This allows us to have as many IO operations as our OS can handle happening at the same time. For example, in a web server with hundreds or thousands of pending requests with multiple blocking queries, performing the blocking queries asynchronously gives you the ability to be able to continue working and not just sit still and wait until the blocking operations come back.

Q21. What are the security mechanisms available in Node.js?

We can secure our Node.js application in the following ways:

Authentication—Authentication is one of the primary security stages at which user is identified as permitted to access the application at all. Authentication verifies the user’s identity through one or several checks. In Node.js, authentication can be either session-based or token-based. In session-based authentication, the user’s credentials are compared to the user account stored on the server and, in the event of successful validation, a session is started for the user. Whenever the session expires, the user needs to log in again. In token-based authentication, the user’s credentials are applied to generate a string called a token which is then associated with the user’s requests to the server.

Error Handling—Usually, the error message contains the explanation of what’s actually gone wrong for the user to understand the reason. At the same time, when the error is related to the application code syntax, it can be set to display the entire log content on the frontend. For an experienced hacker, the log content can reveal a lot of sensitive internal information about the application code structure and tools used within the software.

Request Validation—Another aspect which has to be considered, while building a secure Node.js application, is a validation of requests or, in other words, a check of the incoming data for possible inconsistencies. It may seem that invalid requests do not directly affect the security of a Node.js application, however, they may influence its performance and robustness. Validating the incoming data types and formats and rejecting requests not conforming to the set rules can be an additional measure of securing your Node.js application.

Node.js Security Tools and Best Practices—We can use tools like **helmet** (protects our application by setting HTTP headers), **csurf** (validates tokens in incoming requests and rejects the invalid ones), **node rate limiter** (controls the rate of repeated requests. This function can protect you from brute force attacks) and **cors** (enables cross-origin resource sharing).

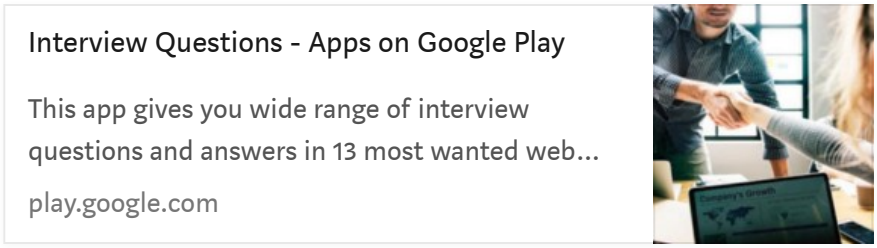
Q22. What is the passport in Node.js?

Passport.js is a simple, unobtrusive Node.js authentication middleware for Node.js. Passport.js can be dropped into any Express.js-based web application.

Passport recognizes that each application has unique authentication requirements. Authentication mechanisms, known as strategies, are packaged as individual modules. Applications can choose which strategies to employ, without creating unnecessary dependencies.

By default, if authentication fails, Passport will respond with a `401 Unauthorized` status, and any additional route handlers will not be invoked. If authentication succeeds, the `next` handler will be invoked and the req.user property will be set to the authenticated user.

For more Node JS Interview Questions and answer install our Android App:



or visit our blog:

http://blog.vigowebs.com/post/2017/nodejs-interview-questions/