### 1) What is Express.js?

Express.js, or simply Express, is a free, open-source, lightweight, and fast backend web application framework for Node.js. It is released as open-source software under the MIT License.

It is designed for building single-page, multi-page, and hybrid web applications and APIs. It is called the de facto standard server framework for Node.js. It was founded and developed by ****TJ Holowaychuk**** in 2010 and written in JavaScript.

### 2) What are some distinctive features of Express?

As Express is a lightweight, minimal and flexible Node.js web application framework, it provides a robust set of features for web and mobile applications. Following is the list of some distinctive features of this framework:

* js can be used to design single-page, multi-page, and hybrid web applications and APIs.
* It allows to set up middleware to respond to HTTP/RESTful Requests.
* It defines a routing table to perform different HTTP operations (method and URL).
* It allows to dynamically rendering HTML Pages based on passing arguments to templates.
* It provides high performance because of its ultra-fast I/O. It prepares a thin layer; therefore, the performance is adequate.
* Its MVC-like structure makes it organize the web application into MVC architecture.
* It provides good database support. It supports RDBMS as well as NoSQL databases.
* It is asynchronous and single-threaded.
* Its robust API makes routing easy.

### 3) Is Express.js front-end or backend framework?

Express.js or Express is a JavaScript backend framework. It is mainly designed to develop complete web applications (single-page, multi-page, and hybrid web applications) and APIs. Express is the backend component of the ****MEAN stack**** where ****M stands for MongoDB****, which handles database; ****E stands for Express,**** which handles backend; ****A stands for AngularJS****, which is for the front-end, and ****N stands for Node****.

### 4) Why do we use Express.js?

Express.js is an automatically prebuilt Node.js framework that facilitates us to create server-side web applications faster and smarter. The main reason for choosing Express is its simplicity, minimalism, flexibility, and scalability characteristics.

### 5) What is the difference between Express.js and Node.js?

Node.js is an open-source, cross-platform run-time environment used for executing JavaScript code outside of a browser. Node.js is not a framework or a programming language; it is a platform that acts as a web server. Many big companies such as Paypal, Uber, Netflix, Wallmart, etc., are using this. On the other hand, Express is a small framework based on the functionality of Node.js.

****Some key differences between Express.js and Node.js:****

|  |  |  |
| --- | --- | --- |
| **Feature** | **Express.js** | **Node.js** |
| Definition | Express.js is a lightweight and fast backend web application framework for Node.js. | Node.js is an open-source and cross-platform that is used to execute JavaScript code outside of a browser. |
| Usage | Express.js is used to develop complete web applications such as single-page, multi-page, and hybrid web applications and APIs. It uses approaches and principles of Node.js. | Node.js is used to build server-side, input-output, event-driven apps. |
| Features | Express has more features than Node.js. | Node.js has fewer features as compared to Express.js. |
| Building Block | Express.js is built on Node.js. | Node.js is built on Google's V8 engine. |
| Written in | Express.js is written in JavaScript only. | Node.js is written in C, C++, and JavaScript language. |
| Framework/Platform | Express.js is a framework of Node.js based on its functionalities. | Node.js is a run-time platform or environment designed for server-side execution of JavaScript. |
| Controllers | Express.js is assigned with controllers. | Node.js is assigned with controllers. |
| Routing | Routing is provided in Express.js. | Routing is not provided in Node.js. |
| Middleware | Express.js uses middleware to arrange the functions systematically on the server-side. | Node.js doesn't use any such provision of middleware. |
| Coding | Express is easy to code and requires less coding time. | Node.js requires more coding time as compare to Express.js. |

### 6) How does an Express code look like?

The express.js program is saved with ".js" extension.

****See the example:****

1. var express = require('express');
2. var app = express();
3. app.get('/', function (req, res) {
4. res.send('Welcome to JavaTpoint!');
5. });
6. var server = app.listen(8000, function () {
7. var host = server.address().address;
8. var port = server.address().port;
9. console.log('Example app listening at http://%s:%s', host, port);
10. });

When you run the Node.js command prompt, the app will listen at the specified server address and give the following output.

****Output:****

### 7) Write a code to get post a query in Express.js.

1. var bodyParser = require('body-parser')
2. app.use( bodyParser.json() );       // to support JSON-encoded
3. app.use(bodyParser.urlencoded({     // to support URL-encoded
4. extended: **true**
5. }));

### 8) What do you understand by Scaffolding in Express.js?

Scaffolding is a technique used for creating the skeleton structure of an application. It facilitates users to easily create their public directories, routes, views, etc., or a web application skeleton. Generally, users manually create their public directory, add middleware, create separate route files, etc. Using a scaffolding tool, they can set up all these things to directly get started with building their application.

There are two ways to install Scaffolding and use it in your application.

1. Express application generator
2. Yeoman

****Express application generator:**** This is used to create an application skeleton quickly. Use the following command to install the Express application generator.

1. npm install express-generator -g
2. express myApp

By using the above command, a project named "myApp" will be created along with following the files/folders in the project.

* ****Bin:**** The bin folder contains one file called www is the main configuration file of the app.
* ****Public:**** The public folder contains JavaScript, CSS, and images, etc.
* ****Routes:**** This folder contains the routing files.
* ****Views:**** The view folder contains the view files of the application.
* ****js:**** The app.js file is the main file of the application.
* ****json:**** The package.json file is the manifest file. It contains all metadata of the project, such as the packages used in the app (called dependencies) etc.

Now, we have to install all the dependencies mentioned in the package.json file by using the following command:

1. cd myApp
2. npm install

****Yeoman:**** Use the following command in your terminal to install Yeoman:

1. npm install -g yeoman

Yeoman uses generators to scaffold out applications.

### 9) Do Other MVC frameworks also support scaffolding?

The Scaffolding technique is also supported by other MVC frameworks other than Express. The following frameworks mainly support it: Ruby on Rails, OutSystems Platform, Play framework, Django, MonoRail, Brail, Symfony, Laravel, CodeIgniter, YII, CakePHP, Phalcon PHP, Model-Glue, PRADO, Grails, Catalyst, Seam Framework, Spring Roo, ASP.NET, etc.

Click here for more information on Scaffolding: <https://www.javatpoint.com/expressjs-scaffolding>

### 10) Which are the arguments available to an Express JS route handler function?

Following are the arguments that are available to an Express.js route handler-function:

* ****Req:**** the request object
* ****Res:**** the response object
* ****Next (optional):**** It is a function employed to pass management to one of the above route handlers.

### 11) What is the difference between Express and Django?

Django is a standalone and lightweight web server for testing and development. On the other hand, Express.js is a Node.js framework that can set the middleware to reply to any HTTP request.

Following is a list of some key differences between Express.js and Django:

|  |  |  |
| --- | --- | --- |
| **Aspects** | **Express.js** | **Django** |
| Architecture | Express follows the MVC architecture. | Django supports the MTV (Model Template View) design. It uses managing data for interacting and validating. |
| Framework | Express is a free, open-source, lightweight, and fast backend web application framework for Node.js to build single-page, multi-page, and hybrid web applications and APIs. | This is a Python-based framework used to develop computer apps in a specified time frame. |
| Efficiency | It is best for developing web applications rapidly on Node.js. | It is more efficient and delivers at a fast speed so, it is cost-effective. |
| Programming language | The Express framework is programmed in Node.js. | Django is programmed in Python programming language. |
| Complexity | Express.js is less complex than Django. | Django is more complex than Express.js |
| Scalability | It provides better scalability. | It is less scalable. |
| Flexibility | Express is a flexible, minimal API-developing Node.js tool. | It provides limited flexibility. |
| Full-stack development | It provides a full-stack development that reduces the cost as you don't need to hire several developers to administer a web application's backend and frontend. | It does not deliver full-stack development. |
| Companies using this technology | Companies such as PayPal, IBM, Fox Sports, etc., are using this technology. | Companies such as Instagram, Mozilla, Bitbucket, etc., are using this technology. |

### 12) How can you enable debugging in Express.js app?

There are different ways to enable debugging in Express.js app in different Operating Systems

****Use the following command on Linux:****

1. DEBUG=express:\*
2. node app.js

****Use the following command on Windows:****

1. set DEBUG=express:\*
2. node app.js

### 13) How can you allow CORS in Express.js?

We can allow CORS in Express.js, by adding the following code in server.js:

1. app.all('\*', function(req, res, next) {
2. res.set('Access-Control-Allow-Origin', '\*');
3. res.set('Access-Control-Allow-Methods', 'GET, POST, DELETE, PUT');
4. res.set('Access-Control-Allow-Headers', 'X-Requested-With, Content-Type');
5. **if** ('OPTIONS' == req.method) **return** res.send(200);
6. next();
7. });

### 14) How can you deal with error handling in Express.js? Explain with an example.

Error handling is much easier in the Express versions over Express 4.0. Use the following steps to do the error handling:

Create an Express.js application. There is no built-in middleware like error handler in express 4.0, so you have to either install a middleware or create a custom one.

****Create a Middleware:****

Create a middleware as following:

1. // error handler
2. app.use(function(err, req, res, next) {
3. // set locals, only providing error in development
4. res.locals.message = err.message;
5. res.locals.error = req.app.get('env') === 'development' ? err : {};
6. // render the error page
7. res.status(err.status || 500);
8. res.render('error');
9. });

****Install Error Handler Middleware:****

****Install the errorhandler as following:****

1. npm install errorhandler --save

****Create a variable:****

1. var errorhandler = require('errorhandler')

****Use the middleware as following:****

1. **if** (process.env.NODE\_ENV === 'development') {
2. // only use in development
3. app.use(errorhandler({log: errorNotification}))
4. }
5. function errorNotification(err, str, req) {
6. var title = 'Error in ' + req.method + ' ' + req.url
7. notifier.notify({
8. title: title,
9. message: str
10. })
11. }

### 15) Write the code to start serving static files in Express.js.

See the Example:

1. app.use(express.**static**('public'))
2. app.use('/static', express.**static**(path.join(\_\_dirname, 'public')))

### 16) What is Middleware in Express.js? What are the different types of Middleware?

Middleware is a function invoked by the Express routing layer before the final request handler.

Middleware functions are used to perform the following tasks:

* It is used to execute any code.
* It is also used to make changes to the request and the response objects.
* It is responsible for ending the request-response cycle.
* It can call the next middleware function in the stack.

****Type of Middleware****

Following are the main types of Middleware:

* Application-level Middleware
* Router-level Middleware
* Error-handling Middleware
* Built-in Middleware
* Third-party Middleware

****Application-level middleware:**** The application-level middleware method is used to bind to the app object using app.use() method. It applies on all routes.

1. //This middleware will execute for each route.
2. app.use(function (req, res, next) {
3. console.log('Current Time:', Date.now())
4. next()
5. })

****Router-level Middleware:**** The router-level Middleware is used to bind to a specific instance of express.Router().Built-in Middleware: The built-in Middleware was introduced with version 4.x. It ends the dependency on Connect.

There are the following built-in middleware functions in Express.js:

* ****static:**** It is used to serve static assets such as HTML files, images, etc.
* ****json:**** It is used to parse the incoming requests with JSON payloads. It is available with Express 4.16.0+
* ****urlencoded:**** It is used to parse the incoming requests with URL-encoded payloads. It is available with Express 4.16.0+

****Third-party Middleware:**** There are many third-party middleware available such as:

* Body-parser
* Cookie-parser
* Mongoose
* Sequelize
* Cors
* Express-validator

To handle HTTP POST requests in Express.js version 4 and above, we have to install a middleware module called body-parser. Body-parser extracts the entire body portion of an incoming request stream and exposes it on req.body, The Middleware was a part of Express.js earlier, but now you have to install it separately. You can install it by using the following command:

1. npm install MODULE\_NAME

You can load it by using requires and used later:

****See the Example:****

1. var bodyParser = require('body-parser');
2. app.use(bodyParser.json());
3. app.use(bodyParser.urlencoded({ extended: **false** }))

#### Note: You can use multiple middleware as an array on a single route.

****See the Example:****

1. var middlewareArray = [middleware1, middleware2]
2. app.get('/home', middlewareArray, function (req, res, next) {
3. //Code snippets
4. })

### 17) Which template engines do Express support?

Express.js supports any template engine that follows the (path, locals, callback) signature.

### 18) How can we render a pain HTML?

There is no need to "render" HTML with the res.render() function. If you have a specific file, you can use the res.sendFile() function, but you should use the express if you serve many assets from a directory.static() middleware function.

### **1. How do you install an express application generator for scaffolding?**

Express application generator is used for quickly creating an application skeleton. The given command is used for installing the express application generator.

npm install expr

ess-generator –g

express myApp

It will create the project "myApp" with some files. Then we install all the dependencies stated in package.json using the given command:

cd myApp

npm install

### **2. How do you install a yeoman for scaffolding?**

Generators are used by yeoman for scaffolding the applications. And we can use the following command to install yeoman.

npm install –g yeoman

### **3. Mention the arguments that are available in an Express JS route handler function.**

The arguments that are available in the route handler function of Express JS are given below:

* ****Res**** - It is the response object.
* ****Req**** - It is the request object
* ****Next (optional)**** - This argument is used for passing the management to any of the above-given route handlers.

****[ Learn Complete**[Express JS Tutorial](https://mindmajix.com/express-js-tutorial" \o "Express JS Tutorial" \t "https://mindmajix.com/_blank)**]****

### **4. Mention the ways of debugging on Linux as well as Windows.**

Debugging on Windows can be done as follows:

set DEBUG = express:\*

node app.js

And debugging on [Linux](https://mindmajix.com/linux-tutorial" \o "Linux Tutorial" \t "https://mindmajix.com/_blank) can be done as follows:

DEBUG = express:\*

node app.js

### **5. List the built-in middleware functions provided by Express.**

Express JS provides the following built-in middleware functions:

1. ****Static****: We use it for serving static assets like images, HTML files, etc.
2. ****JSON****: This is available in Express 4.16.0+. And we use it for passing the incoming requests with JSON payloads.
3. ****URL encoded****: This is also available in Express 4.16.0+. And we use it for passing the incoming requests with URL-encoded payloads.

### **6. Mention some third-party middleware provided by Express JS.**

Some of the many third-party middlewares that the Express JS provides are:

* Cookie-parser
* Body-parser
* Cors
* Mongoose
* Express-validator
* Sequelize.

[](https://bit.ly/3if9dmk)

### **7. When is application-level Middleware used?**

We use the application-level Middleware for binding the app object with the help of the app.use() method. It can be applied on all routes. The syntax is given below:

// This Middleware executes for each route.

App.use(function (req, res, next) {

console.log(‘Current Time:’, Date.now())

next()

})

### **8. Tell us about Router-level Middleware and Built-in Middleware.**

****Router-level Middleware****- We use the router-level Middleware for binding with a particular instance of Express.Router().

****Built-in Middleware****- The version 4.x of Express introduced the built-in Middleware. The dependency on connecting gets removed by use of this Middleware.

### **9. Mention some of the databases with which Express JS is integrated.**

A myriad of NoSQL and RDBMS databases are supported by Express JS, such as:

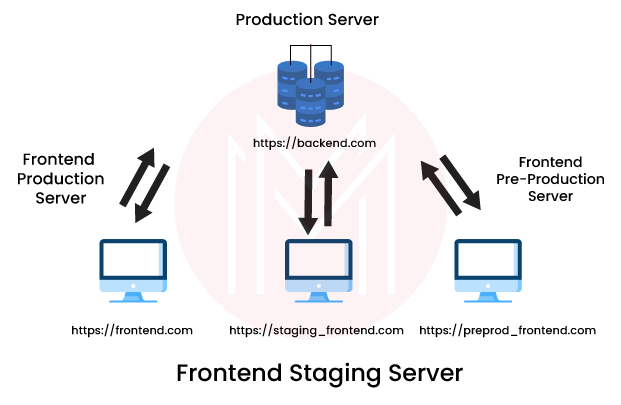
* MySQL
* [MongoDB](https://mindmajix.com/what-is-mongodb" \o "What is MongoDB?" \t "https://mindmajix.com/_blank)
* PostgreSQL
* SQLite
* Oracle
* SQLite.

|  |
| --- |
| ****Related Article**[MongoDB vs PostgreSQL](https://mindmajix.com/mongodb-vs-postgresql" \o "MongoDB vs PostgreSQL" \t "https://mindmajix.com/_blank)** |

### **10. What is meant by CORS in Express JS? And what are the ways by which it can be achieved?**

CORS is the acronym for Cross-origin resource sharing. We can request the restricted resources from another server or domain using this mechanism. And we can do this mainly in the following three ways:

1. Express cors module
2. Res.header() (or res.set()): Multiple headers can be set using this way.
3. Res.setHeader(): Only a single header can be set in this way.



### **11. What ways are provided by Express JS to configure the properties?**

Express JS provides us with two ways for configuring the properties, which are given below:

* With process.ENV
* With require.JS.

### **12. How are the properties configured with process.ENV?**

The properties are configured by the given steps:

* We create a file within the project folder and name it ".env."
* All other properties are let to be separate within the ".env" file.
* We can employ any of the properties in server.js.

### **13. How are the properties configured with require.JS?**

The properties are configured by the given steps:

* We create a file within the config folder of the project folder and name it "config.json."
* All the config properties are present there within the config.json file.

### **14. How can the Express JS application be structured?**

There is no specific answer to this question. The dimensions of our application and hence the concerned team define the solution in different situations. The express logic in the Routes and alternative applications can board as many files as we want in any directory structure. The given examples can be read for further inspiration:

* Route map
* Route listings
* MVC vogue controllers.

### **15. How is the plain HTML rendered?**

We don’t need to render HTML with the function- res.render(). Instead, we can use the res.sendFile() function if we have a specific file. And we can use the Express.static() middleware function if we serve several assets from a directory.

## **Express JS Interview Questions For Experienced**

### **16. Write the code for "Hello world" using Express.**

Create a new file by the name- index.js and type the following commands:

var express = require (‘express’);

var app = express ();

app.get (‘/’, function (req, res){

res.send (“Hello world”);

});

app.listen (3000);

Now go to the terminal after saving it and type:

nodemon index.js

### **17. What are the most used HTTP methods in Express JS?**

The following HTTP methods are the most used ones:

* ****GET****- A specified resource's representation is requested by the GET method. These requests can only retrieve data.
* ****POST****- Posting of the data enclosed in the request as a new entity is done using the POST method. The entity is identified by the URI.
* ****PUT****- Modification in the existing entity is done with the data enclosed in the request identified by the URI.
* ****DELETE****- The request for deleting the specified source is made by the DELETE method.

### **18. How can the cookies be manipulated using ‘Response.cookie()’?**

We use the “res.cookie(‘username’, ‘Flavio’)” command is used for manipulating. But it accepts a third parameter conaining various options as specified below:

res.cookie (‘username’, ‘Flavio’, { domain: ‘flaviocopes.com’, path: ‘/administrator’, secure: true })

res.cookie (‘username’, ‘Flavio’ , { expires: new Date(Date.now() + 90000), httpOnly: true})

### **19. When does a Cross-Origin resource get failed in Express JS?**

A cross-Origin can fail in the following scenarios-

* If it's to a different domain
* If it's to a different port
* If it's to a different subdomain
* If it's to a different protocol.

### **20. How can you use a Pug template engine inside Express?**

We will first install it using the given command:

npm install pug

Then we will set it as following when initializing the Express app:

const express = require ('express')

const app = express()

app.set ('view engine', 'pug')

### **21. What do you mean by the sanitizing input process?**

People can always enter weird things via the client-side code. They use the tools to POST things directly to our endpoints. For this, the Express provides various sanitizing methods to prevent these happenings.

### **22. Mention some methods for sanitizing.**

Consider the following sanitizing methods:

* Trim() will trim the characters at the beginning as well as the ending of a string.
* Escape() will replace ‘, “, <, >, &, / with the corresponding HTML entities.
* NormalizeEmail() will canonicalize an email address.
* Blacklist() will remove the characters appearing on the blacklist.

### **23. Give an example of HTML form code allowing user to upload a file.**

Consider the given example of HTML form code that allows a user to upload file.

<form method = “POST” action = “/submit-form”>

<input type = “file” name = “document” />

<input type = “submit” />

</form>

### **24. What are the methods that you can call when the Formidable.File objects arise giving the information about the uploaded file?**

We can call the following methods in such cases-

* ****File.name****- the name of the file
* ****File.path****- the path to which the file is written
* ****File.size****- the size of the file in bytes
* ****File. type****- the file's MIME-type.

### **25. What steps will you follow to set up HTTP for Express with the help of Let's Encrypt and Certbot?**

We will follow the given steps to set up HTTP:

1. Installing certbot
2. Generating the SSL certificate by Certbot
3. Allowing Express to serve the static files
4. Confirming the domain
5. Obtaining the certificate
6. Setting up the renewal.

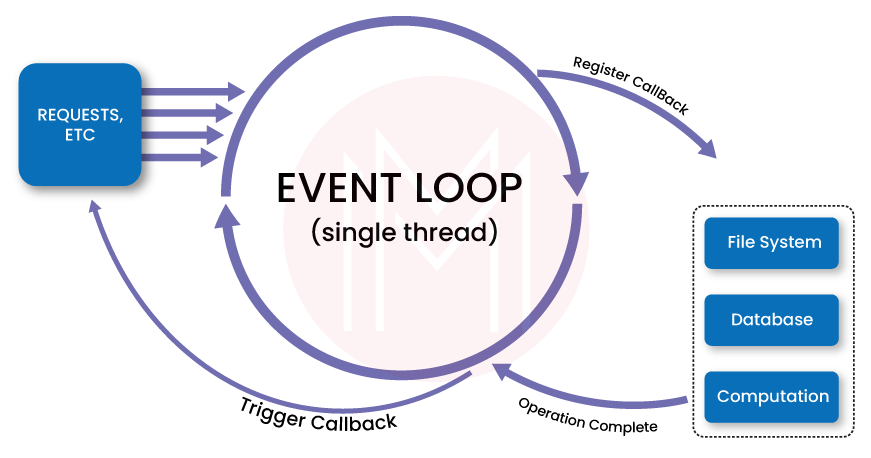
### **26. What options are available in the terminal command for generating a skeleton Express JS app?**

The options available are given below:

* --sessions or –s for adding session report
* --hogan or –H for adding Hogan.js engine support
* --ejs or –e for adding EJS engine support
* --css <engine> or –c <engine> for adding style sheet support
* -jshtml or –J for adding JSHTML engine support
* --force or –f for forcing app generation on the directory which is non-empty.

### **27. What is meant by an event-loop in Node JS?**

The event-loop manages the async content using a listener and queue. The main thread sends the async function to a different thread whenever it requires to be executed. Alongside, v8 is allowed to execute the main code. The event loop has different stages including pending callbacks, timers, check, poll, close callbacks, etc. with different FIFO queues.



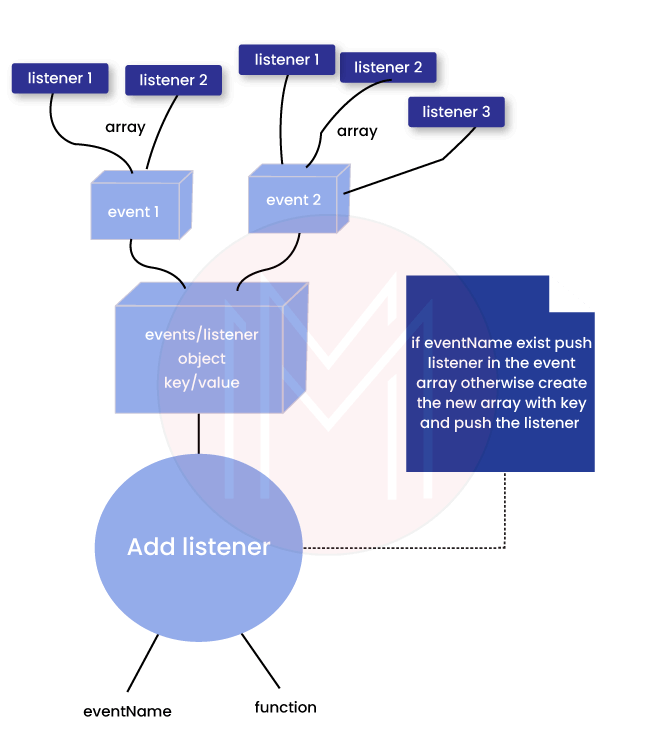
### **28. Why should the Express server and app be separated?**

The server initializes the Middleware, routes, and other application logic. On the other hand, the app contains all the business logic that the server-initiated routes will serve. This enables the encapsulation of the business logic from the application logic for smooth functioning.

|  |
| --- |
|  |

### **29. What is meant by an Event emitter in Node JS?**

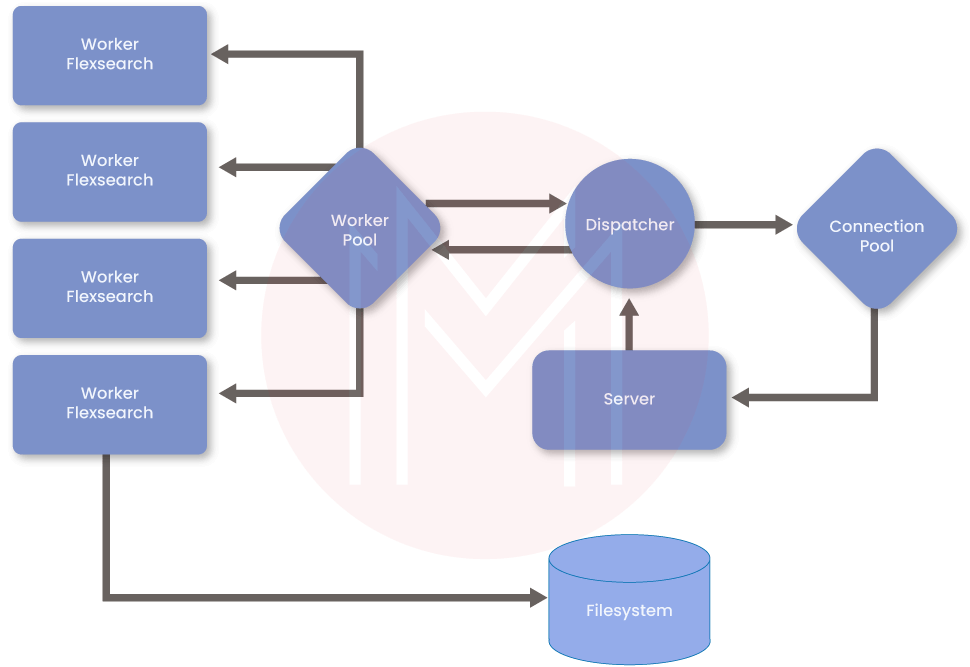
It's a class of Node JS capable of emitting events. We do this by attaching the named events emitted by the object by using the function- eventEmitter.on().



### **30. Differentiate between worker threads and clusters in Node JS.**

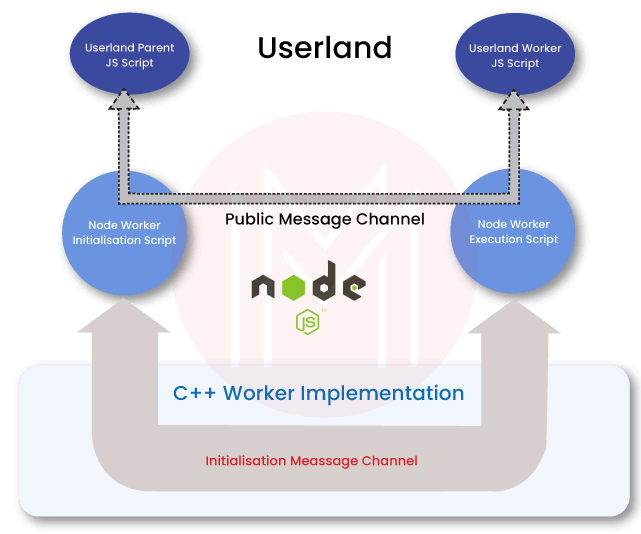
****Cluster****:

* Each CPU has one process with an IPC to communicate.
* Clusters help when multiple servers are required to accept HTTP requests through a single port.
* The processes have separate memory because of being spanned in different CPUs leading to memory issues.



****Worker Threads****:

* Only a single process is there with multiple threads.
* Each Node contains one Node having most APIs accessible.
* The memory is shared with other threads.
* We can use this for CPU-intensive tasks.



## **Most Common Express JS FAQs**

### **1. List down major companies that use Express JS.**

The following major companies use Express JS-

* Twitter
* Stack
* Accenture
* Intuit
* Bepro Company
* Trustpilot
* BlaBlaCar.

### **2. What are the popular alternatives to Express JS?**

[React JS](https://mindmajix.com/react-js-tutorial" \o "React JS Tutorial" \t "https://mindmajix.com/_blank), Meteor, Mean, Flask, Catalyst, Django, Apache Flex, and Laravel are some of the popular alternatives to Express JS.

### **3. Enlist some distinct features of Express JS.**

Some distinct features of Express JS are given below-

* We can design single-page, multi-page, and hybrid web apps as well as APIs with Express JS.
* A routing table is defined for performing HTTP operations.
* Middleware can be set up for responding to RESTful/HTTP requests.
* The MVC-like structure enables organizing the web apps into MVC architecture.
* HTML pages can be dynamically rendered on the basis of passing arguments to templates.
* NoSQL, as well as RDBMS databases, are supported by it.
* High performance is delivered due to its super-fast I/O. the performance is adequate because of the thin layer prepared by it.
* Routing gets easy by its robust API.
* It is single-threaded as well as asynchronous.

### **4. Which major tools integrate with Express JS?**

The following popular tools integrate with Express JS:

* Sentry
* Node JS
* Datadog
* Mean
* Nodemon
* Bugsnag
* LoopBack
* Sails JS.

### **5. Is Express JS a back-end framework or a front-end framework?**

Express JS is a back-end framework built on JavaScript. It is the [MEAN stack](https://mindmajix.com/what-is-mean-stack" \o "What is MEAN stack?" \t "https://mindmajix.com/_blank)'s back-end component. Here, 'M' refers to MongoDB, and it manages the database. 'E' refers to Express, and it manages the back-end. 'A' refers to AngularJS, and it handles the back-end. And 'N' refers to Node.

### **6. Differentiate between Node JS and Express JS.**

[Node JS](https://mindmajix.com/node-js/what-is-node-js" \o "What is Node JS?" \t "https://mindmajix.com/_blank) is an open-source platform on which the JavaScript code is executed outside of a browser. It is used by several companies, including Uber, Walmart, Netflix, etc. It is a platform acting as a web server and not a programming language or framework. On the other hand, Express JS is a framework built on Node JS.

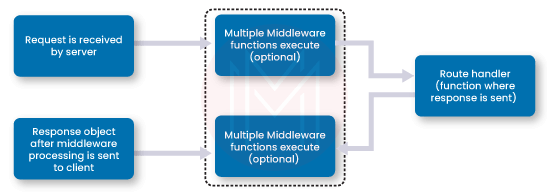
### **7. Which template engines are supported by Express?**

All template engines following the locals, path, and callback signatures are supported by Express JS.

### **8. What is Middleware, and what are its functions?**

Middleware is the function that we invoke before the final request process through the express routing layer. Its functions are given below:

1. Any code like setting headers, validation, etc., can be executed.
2. Changes can be made to the response (res) and request (req) objects.
3. The request-response cycle can also be ended by Middleware.
4. The next middleware function can be called in the stack for proceeding and processing the final request.



### **9. List the main types of Middleware.**

The main types of Middleware are given below:

* Router-level Middleware
* Application-level Middleware
* Built-in Middleware
* Error-handling Middleware
* Third-party Middleware.

### **10. What is meant by Scaffolding in Express JS? What are the ways to achieve this?**

The process of creating the structure of the application is referred to as scaffolding. The two ways of achieving it are given below:

* Express application generator
* Yeoman.

### **Conclusion**

Node JS along with Express JS, allows building a complex and high-level web application from a single line of code easily. That's why it is one of the most sought-after skills in the industry today. And this was a comprehensive list of Express JS Interview Questions that are often asked in interviews. We hope that it will be beneficial for your scheduled interview in the near future.

### 1. What is Node.js? Where can you use it?

[Node.js is an open-source](https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs" \o "Node.js is an open-source" \t "https://www.simplilearn.com/tutorials/nodejs-tutorial/_blank), cross-platform [JavaScript](https://www.simplilearn.com/tutorials/javascript-tutorial/introduction-to-javascript" \o "JavaScript" \t "https://www.simplilearn.com/tutorials/nodejs-tutorial/_blank) runtime environment and library to run web applications outside the client’s browser**.**It is used to create server-side web applications.

Node.js is perfect for data-intensive applications as it uses an asynchronous, event-driven model. You can use  I/O intensive web applications like video streaming sites. You can also use it for developing: Real-time web applications, Network applications, General-purpose applications, and Distributed systems.

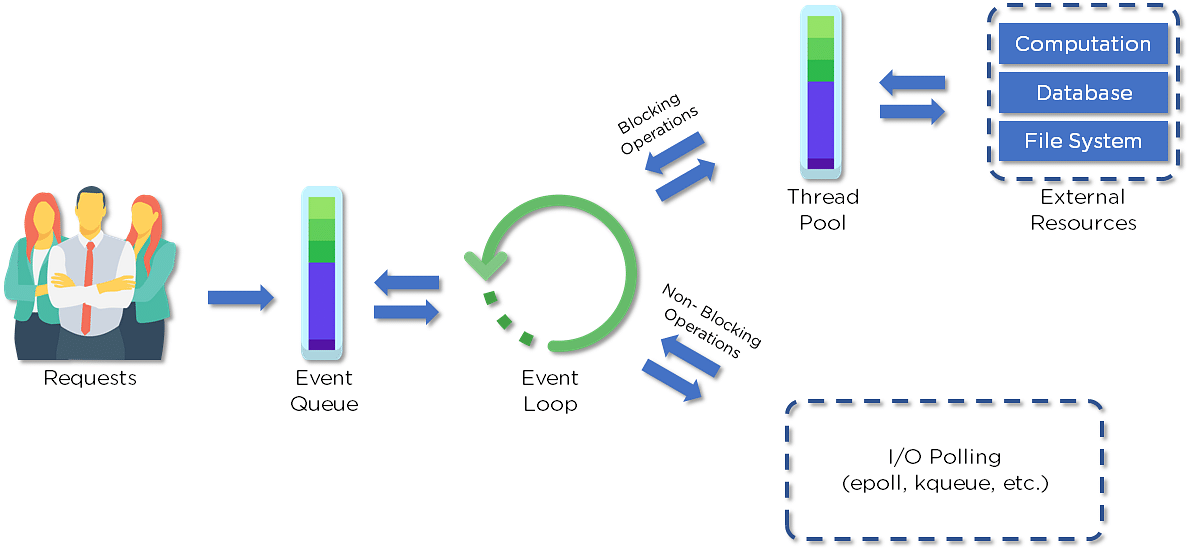
### 2. Why use Node.js?

Node.js makes building scalable network programs easy. Some of its advantages include:

* It is generally fast
* It rarely blocks
* It offers a unified programming language and data type
* Everything is asynchronous
* It yields great concurrency

### 3. How does Node.js work?

A web server using Node.js typically has a workflow that is quite similar to the diagram illustrated below. Let’s explore this flow of operations in detail.



* Clients send requests to the webserver to interact with the web application. Requests can be non-blocking or blocking:
* Querying for data
* Deleting data
* Updating the data
* Node.js retrieves the incoming requests and adds those to the Event Queue
* The requests are then passed one-by-one through the Event Loop. It checks if the requests are simple enough not to require any external resources
* The Event Loop processes simple requests (non-blocking operations), such as I/O Polling, and returns the responses to the corresponding clients

A single thread from the Thread Pool is assigned to a single complex request. This thread is responsible for completing a particular blocking request by accessing external resources, such as computation, database, file system, etc.

Once the task is carried out completely, the response is sent to the Event Loop that sends that response back to the client.

### 4. Why is Node.js Single-threaded?

Node.js is single-threaded for async processing. By doing async processing on a single-thread under typical web loads, more performance and scalability can be achieved instead of the typical thread-based implementation.

### 5. If Node.js is single-threaded, then how does it handle concurrency?

The Multi-Threaded Request/Response Stateless Model is not followed by the Node JS Platform, and it adheres to the Single-Threaded Event Loop Model. The Node JS Processing paradigm is heavily influenced by the JavaScript Event-based model and the JavaScript callback system. As a result, Node.js can easily manage more concurrent client requests. The event loop is the processing model's beating heart in Node.js.

### 6. Explain callback in Node.js.

A callback function is called after a given task. It allows other code to be run in the meantime and prevents any blocking.  Being an asynchronous platform, Node.js heavily relies on callback. All APIs of Node are written to support callbacks.

### 7. What are the advantages of using promises instead of callbacks?

* The control flow of asynchronous logic is more specified and structured.
* The coupling is low.
* We've built-in error handling.
* Improved readability.

### 8. How would you define the term I/O?

* The term I/O is used to describe any program, operation, or device that transfers data to or from a medium and to or from another medium
* Every transfer is an output from one medium and an input into another. The medium can be a physical device, network, or files within a system



### 9. How is Node.js most frequently used?

Node.js is widely used in the following applications:

1. Real-time chats
2. Internet of Things
3. Complex SPAs (Single-Page Applications)
4. Real-time collaboration tools
5. Streaming applications
6. Microservices architecture

### 10. Explain the difference between frontend and backend development?

|  |  |
| --- | --- |
| **Front-end** | **Back-end** |
| Frontend refers to the client-side of an application | Backend refers to the server-side of an application |
| It is the part of a web application that users can see and interact with | It constitutes everything that happens behind the scenes |
| It typically includes everything that attributes to the visual aspects of a web application | It generally includes a web server that communicates with a database to serve requests |
| HTML, CSS, JavaScript, AngularJS, and ReactJS are some of the essentials of frontend development | Java, PHP, Python, and Node.js are some of the backend development technologies |

If you are curious to explore interview questions related to frontend development, you can check out our article on [ReactJS Interview Questions and Answers](https://www.simplilearn.com/tutorials/reactjs-tutorial/reactjs-interview-questions" \o "ReactJS Interview Questions and Answers" \t "https://www.simplilearn.com/tutorials/nodejs-tutorial/_blank).

### 11. What is NPM?

NPM stands for Node Package Manager, responsible for managing all the packages and modules for Node.js.

Node Package Manager provides two main functionalities:

* Provides online repositories for node.js packages/modules, which are searchable on search.nodejs.org
* Provides command-line utility to install Node.js packages and also manages Node.js versions and dependencies

### 12. What are the modules in Node.js?

Modules are like JavaScript libraries that can be used in a Node.js application to include a set of functions. To include a module in a Node.js application, use the **require()** function with the parentheses containing the module's name.

Node.js has many modules to provide the basic functionality needed for a web application. Some of them include:

|  |  |
| --- | --- |
| **Core Modules** | **Description** |
| HTTP | Includes classes, methods, and events to create a Node.js HTTP server |
| util | Includes utility functions useful for developers |
| fs | Includes events, classes, and methods to deal with file I/O operations |
| url | Includes methods for URL parsing |
| query string | Includes methods to work with query string |
| stream | Includes methods to handle streaming data |
| zlib | Includes methods to compress or decompress files |

### 13. What is the purpose of the module .Exports?

In Node.js, a module encapsulates all related codes into a single unit of code that can be parsed by moving all relevant functions into a single file. You may export a module with the module and export the function, which lets it be imported into another file with a needed keyword.

### 14. Why is Node.js preferred over other backend technologies like Java and PHP?

Some of the reasons why Node.js is preferred include:

* Node.js is very fast
* Node Package Manager has over 50,000 bundles available at the developer’s disposal
* Perfect for data-intensive, real-time web applications, as Node.js never waits for an API to return data
* Better synchronization of code between server and client due to same code base
* Easy for web developers to start using Node.js in their projects as it is a JavaScript library

### 15. What is the difference between Angular and Node.js?

|  |  |
| --- | --- |
| **Angular** | **Node.js** |
| It is a[frontend development](https://www.simplilearn.com/how-to-become-a-front-end-developer-article" \o "frontend development" \t "https://www.simplilearn.com/tutorials/nodejs-tutorial/_blank) framework | It is a server-side environment |
| It is written in[TypeScript](https://www.simplilearn.com/tutorials/programming-tutorial/advanced-typescript" \o "TypeScript" \t "https://www.simplilearn.com/tutorials/nodejs-tutorial/_blank) | It is written in [C, C++ languages](https://www.simplilearn.com/tutorials/cpp-tutorial/difference-between-c-and-cpp" \o "C, C++ languages" \t "https://www.simplilearn.com/tutorials/nodejs-tutorial/_blank) |
| Used for building single-page, client-side web applications | Used for building fast and scalable server-side networking applications |
| Splits a web application into MVC components | Generates database queries |

**Also Read:**[What is Angular?](https://www.simplilearn.com/tutorials/angular-tutorial/what-is-angular" \o "What is Angular?" \t "https://www.simplilearn.com/tutorials/nodejs-tutorial/_blank)

### 16. Which database is more popularly used with Node.js?

[MongoDB](https://www.simplilearn.com/tutorials/mongodb-tutorial" \o "MongoDB" \t "https://www.simplilearn.com/tutorials/nodejs-tutorial/_blank) is the most common database used with Node.js. [It is a NoSQL](https://www.simplilearn.com/rise-of-nosql-and-why-it-should-matter-to-you-article" \o "It is a NoSQL" \t "https://www.simplilearn.com/tutorials/nodejs-tutorial/_blank), cross-platform, document-oriented database that provides high performance, high availability, and easy scalability.

### 17. What are some of the most commonly used libraries in Node.js?

There are two commonly used libraries in Node.js:

* [ExpressJS](https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-express-js" \o "ExpressJS" \t "https://www.simplilearn.com/tutorials/nodejs-tutorial/_blank)- Express is a flexible Node.js web application framework that provides a wide set of features to develop web and mobile applications.
* **Mongoose** - [Mongoose](https://www.simplilearn.com/tutorials/nodejs-tutorial/nodejs-mongodb" \o "Mongoose" \t "https://www.simplilearn.com/tutorials/nodejs-tutorial/_blank) is also a Node.js web application framework that makes it easy to connect an application to a database.

### 18. What are the pros and cons of Node.js?

|  |  |
| --- | --- |
| **Node.js Pros** | **Node.js Cons** |
| Fast processing and an event-based model | Not suitable for heavy computational tasks |
| Uses JavaScript, which is well-known amongst developers | Using callback is complex since you end up with several nested callbacks |
| Node Package Manager has over 50,000 packages that provide the functionality to an application | Dealing with relational databases is not a good option for Node.js |
| Best suited for streaming huge amounts of data and I/O intensive operations | Since Node.js is single-threaded, CPU intensive tasks are not its strong suit |

### 19. What is the command used to import external libraries?

The “require” command is used for importing external libraries. For example - **“var http=require (“HTTP”).”**  This will load the HTTP library and the single exported object through the HTTP variable.

Now that we have covered some of the important beginner-level Node.js interview questions let us look at some of the intermediate-level Node.js interview questions.

IMG_262

## Node.js Interview Questions and Answers For Intermediate Level

### 20. What does event-driven programming mean?

An event-driven programming approach uses events to trigger various functions. An event can be anything, such as typing a key or clicking a mouse button. A call-back function is already registered with the element executes whenever an event is triggered.

### 21. What is an Event Loop in Node.js?

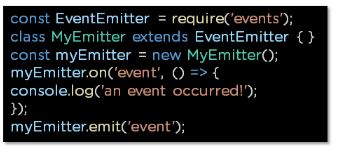
Event loops handle asynchronous callbacks in Node.js. It is the foundation of the non-blocking input/output in Node.js, making it one of the most important environmental features.

### 22. Differentiate between process.nextTick() and setImmediate()?

The distinction between method and product. This is accomplished through the use of nextTick() and setImmediate(). next Tick() postpones the execution of action until the next pass around the event loop, or it simply calls the callback function once the event loop's current execution is complete, whereas setImmediate() executes a callback on the next cycle of the event loop and returns control to the event loop for any I/O operations.

### 23. What is an EventEmitter in Node.js?

* EventEmitter is a class that holds all the objects that can emit events
* Whenever an object from the EventEmitter class throws an event, all attached functions are called upon synchronously



### 24. What are the two types of API functions in Node.js?

The two types of API functions in Node.js are:

* Asynchronous, non-blocking functions
* Synchronous, blocking functions

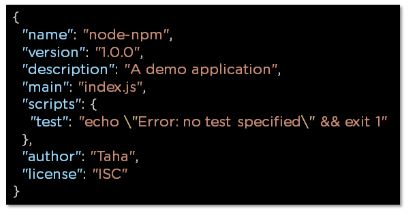
Learn to build network applications quickly and efficiently using JavaScript with the [Node.js Training](https://www.simplilearn.com/node-js-certification-training?source=GhPreviewCTAText" \o "Node.js Training" \t "https://www.simplilearn.com/tutorials/nodejs-tutorial/_blank). Click to enroll now!

### 25. What is the package.json file?

The package.json file is the heart of a Node.js system. This file holds the metadata for a particular project. The package.json file is found in the root directory of any Node application or module

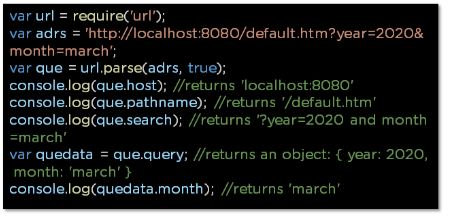
This is what a package.json file looks like immediately after creating a Node.js project using the command: **npm init**

You can edit the parameters when you create a Node.js project.



### 26. How would you use a URL module in Node.js?

The URL module in Node.js provides various utilities for URL resolution and parsing. It is a built-in module that helps split up the web address into a readable format.



### 27. What is the Express.js package?

Express is a flexible Node.js web application framework that provides a wide set of features to develop both web and mobile applications

### 28. How do you create a simple Express.js application?

* The request object represents the HTTP request and has properties for the request query string, parameters, body, HTTP headers, and so on
* The response object represents the HTTP response that an Express app sends when it receives an HTTP request



### 29. What are streams in Node.js?

Streams are objects that enable you to read data or write data continuously.

There are four types of streams:

**Readable –** Used for reading operations

**Writable −** Used for write operations

**Duplex −** Can be used for both reading and write operations

**Transform −** A type of duplex stream where the output is computed based on input

### 30. How do you install, update, and delete a dependency?



### 31. How do you create a simple server in Node.js that returns Hello World?



* Import the HTTP module
* Use createServer function with a callback function using request and response as parameters.
* Type “hello world."
* Set the server to listen to port 8080 and assign an IP address

### 32. Explain asynchronous and non-blocking APIs in Node.js.

* All Node.js library APIs are asynchronous, which means they are also non-blocking
* A Node.js-based server never waits for an API to return data. Instead, it moves to the next API after calling it, and a notification mechanism from a Node.js event responds to the server for the previous API call

### 33. How do we implement async in Node.js?

As shown below, the async code asks the JavaScript engine running the code to wait for the request.get() function to complete before moving on to the next line for execution.



### 34. What is a callback function in Node.js?

A callback is a function called after a given task. This prevents any blocking and enables other code to run in the meantime.

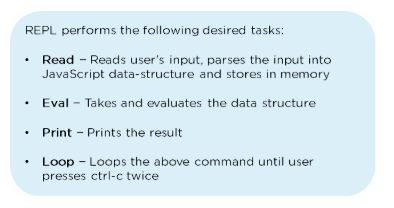
In the last section, we will now cover some of the advanced-level Node.js interview questions.

## Node.js Interview Questions and Answers For Experienced Professionals

This section will provide you with the Advanced Node.js interview questions which will primarily help experienced professionals.

### 35. What is REPL in Node.js?

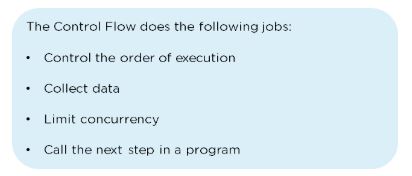
REPL stands for Read Eval Print Loop, and it represents a computer environment. It’s similar to a Windows console or Unix/Linux shell in which a command is entered. Then, the system responds with an output



### 36. What is the control flow function?

The control flow function is a piece of code that runs in between several asynchronous function calls.

### 37. How does control flow manage the function calls?



### 38. What is the difference between fork() and spawn() methods in Node.js?

|  |  |
| --- | --- |
| **fork()** | **spawn()** |
| IMG_274 | IMG_275 |
| fork() is a particular case of spawn() that generates a new instance of a V8 engine. | Spawn() launches a new process with the available set of commands. |
| Multiple workers run on a single node code base for multiple tasks. | This method doesn’t generate a new V8 instance, and only a single copy of the node module is active on the processor. |

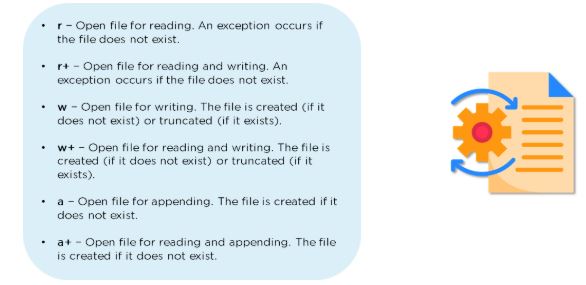
### 39. What is the buffer class in Node.js?

Buffer class stores raw data similar to an array of integers but corresponds to a raw memory allocation outside the V8 heap. Buffer class is used because pure JavaScript is not compatible with binary data

### 40. What is piping in Node.js?

Piping is a mechanism used to connect the output of one stream to another stream. It is normally used to retrieve data from one stream and pass output to another stream

### 41. What are some of the flags used in the read/write operations in files?



### 42. How do you open a file in Node.js?

IMG_277

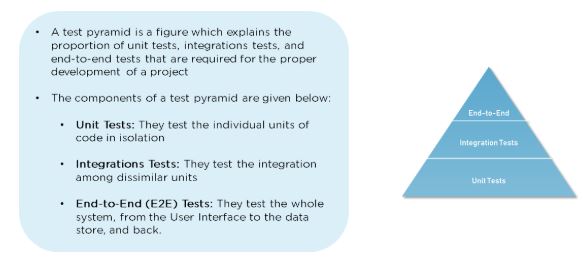
### 43. What is callback hell?

* Callback hell, also known as the pyramid of doom, is the result of intensively nested, unreadable, and unmanageable callbacks, which in turn makes the code harder to read and debug
* improper implementation of the asynchronous logic causes callback hell

### 44. What is a reactor pattern in Node.js?

A reactor pattern is a concept of non-blocking I/O operations. This pattern provides a handler that is associated with each I/O operation. As soon as an I/O request is generated, it is then submitted to a demultiplexer

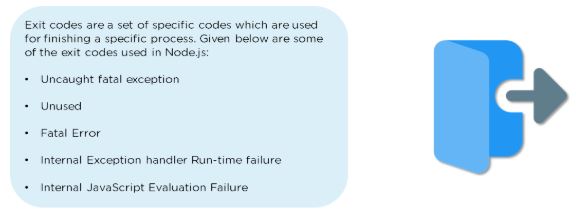
### 45. What is a test pyramid in Node.js?



### 46. For Node.js, why does Google use the V8 engine?

The V8 engine, developed by Google, is open-source and written in [C++](https://www.simplilearn.com/tutorials/cpp-tutorial/learn-cpp-basics" \o "C++" \t "https://www.simplilearn.com/tutorials/nodejs-tutorial/_blank). Google Chrome makes use of this engine. V8, unlike the other engines, is also utilized for the popular Node.js runtime. V8 was initially intended to improve the speed of JavaScript execution within web browsers. Instead of employing an interpreter, V8 converts JavaScript code into more efficient machine code to increase performance. It turns JavaScript code into machine code during execution by utilizing a JIT (Just-In-Time) compiler, as do many current JavaScript engines such as SpiderMonkey or Rhino (Mozilla).

### 47. Describe Node.js exit codes.



### 48. Explain the concept of middleware in Node.js.

Middleware is a function that receives the request and response objects. Most tasks that the middleware functions perform are:

* Execute any code
* Update or modify the request and the response objects
* Finish the request-response cycle
* Invoke the next middleware in the stack

### 49. What are the different types of HTTP requests?

HTTP defines a set of request methods used to perform desired actions. The request methods include:

**GET:**Used to retrieve the data

**POST:**Generally used to make a change in state or reactions on the server

**HEAD:**Similar to the GET method, but asks for the response without the response body

**DELETE:** Used to delete the predetermined resource

### 50. How would you connect a MongoDB database to Node.js?

To create a database in MongoDB:

* Start by creating a MongoClient object
* Specify a connection URL with the correct IP address and the name of the database you want to create

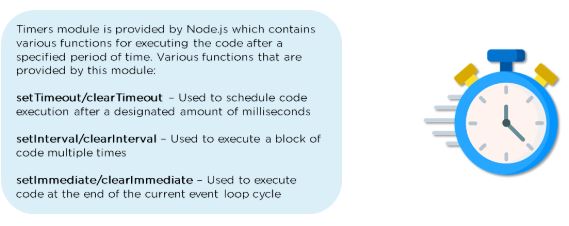


### 51. What is the purpose of NODE\_ENV?

### IMG_281

### 52. List the various Node.js timing features.

As you prepare for your upcoming job interview, we hope that this comprehensive guide has provided more insight into what types of questions you’ll be asked.



### 53. What is WASI, and why is it being introduced?

The WASI class implements the WASI system called API and extra convenience methods for interacting with WASI-based applications. Every WASI instance represents a unique sandbox environment. Each WASI instance must specify its command-line parameters, environment variables, and sandbox directory structure for security reasons.

#### **1. What do you mean by Express JS and what is its use?**

****Answer:****  
Express JS is an application framework that is light-weighted node JS. Variety of versatile, helpful and vital options are provided by this [JavaScript](https://www.educba.com/careers-in-javascript/) framework for the event of mobile additionally as internet applications with the assistance of node JS.  
Express JS Use – Express.js could be a light-weight internet application that helps in organizing the net application into MVC design on the server aspect.

#### **2. Write the steps for setting up an Express JS application?**

****Answer:****  
Following are the steps accustomed for An Express JS application:

* A folder with a constant name because the project name is made.
* A file named package.json is made within the folder created.
* “npm install” command is run on the electronic communication. It installs all the libraries gift in package.json.
* A file named server.js is made.
* “Router” file is made within the package that consists of a folder named index.js.
* “App” is made within the package that has the index.html file.

Let us move on to the next Express JS Interview Questions

#### **3. What function are arguments available to Express JS route handlers?**

****Answer:****  
The arguments which are available to an Express JS route handler-function are-

* ****Req:**** the request object
* ****Res:**** the response object
* ****Next (optional):**** a function that is employed to pass management to 1 of the following route handlers.

The third argument is optional and should be omitted, however, in some cases, it’s helpful wherever there’s a series of handlers and management will be passed to 1 of the following route handlers skipping this one.

#### **4. How to Config properties in Express JS?**

****Answer:****  
In Express JS, there are 2 ways that for configuring the properties:

With process.ENV:

* A file with the name “.env” is to be created within the project folder.
* All the properties are to be other within the “.env” file.
* Any of the properties will be employed in server.js.

With require.JS:

* A file with the name “config.json” is to be created within the config folder within the project folder.
* The config properties are to be present within the config.json file.
* Now, ought to be accustomed access the config.json file.

#### **5. How Should I Structure my Express JS Application?**

****Answer:****  
This is the basic Express JS Interview Questions asked in an interview. There is no definitive answer to the current question. The solution depends on the dimensions of your application and therefore the team that’s concerned.  
Routes and alternative application-Express logic will board as several files as you would like, in any directory structure you favor. Read the subsequent examples for inspiration:

* Route listings
* Route map
* MVC vogue controllers

Also, there are third-party extensions for Express JS applications that modify a number of these patterns:

* Resourceful routing

### **Part 2 – Express JS Interview Questions (Advanced)**

Let us now have a look at the advanced Express JS Interview Questions.

#### **6. How to allow CORS in Express JS? Explain with an example?**

****Answer:****  
In order to permit CORS in Express.js, add the subsequent code in server.js:

For Example:

app.all('\*', function(req, res, next) {  
res.set('Access-Control-Allow-Origin', '\*');  
res.set('Access-Control-Allow-Methods', 'GET, POST, DELETE, PUT');  
res.set('Access-Control-Allow-Headers', 'X-Requested-With, Content-Type');  
if ('OPTIONS' == req.method) return res.send(200);  
next();  
});

#### **7. How to enable debugging in express app?**

****Answer:****  
In different operative Systems, we’ve got following commands:

On UNIX operating system the command would be as follows:

$ DEBUG=express:\* node index.js

On Windows the command would be:

set DEBUG=express:\* & node index.js

From Webstrome IDE

C:\Program Files (x86)\JetBrains\WebStorm 2016.2.4\bin\runnerw.exe" "C:\Program Files\nodejs\node.exe" --debug-brk=61081 --expose\_debug\_as=v8debug E:\Development\nodejd\librarey\bin\www

Let us move on to the next advanced Express JS Interview Questions.

#### **8. Explain Error Handling In Express.js Using An Example?**

****Answer:****  
From Express 4.0 Error handling is easier. The steps are as following:

Create a middleware as following:

// error handler  
app.use(function(err, req, res, next) solely providing error in development  
res.locals.message = err.message;  
res.locals.error = req.app.get('env') === 'development' ? err : ;  
// render the error page  
res.status(err.status || 500);  
res.render('error');  
});

Install Error Handler Middleware:

* Install errorhandler

npm install errorhandler --save

* Create a variable

var e errorhandler = require('errorhandler')

* Use the middleware as following:

if (process.env.NODE\_ENV === 'development') solely use in development  
app.use(errorhandler())  
}  
function errorNotification(err, str, req) power unit title = 'Error in ' + req.method + ' ' + req.url  
notifier.notify()  
}

#### **9. What is the use of next in Express JS?**

****Answer:****  
Next -It passes management to a consecutive matching route. OR a operate to pass management to 1 of the following route handlers.  
The argument could also be omitted, however, is beneficial in cases wherever you have got a series of handlers and you’d wish to pass management to 1 of the following route handlers, and skip this one.

app.get('/user details/:id?', function(req, res, next));

* ****Req and Res:**** It represents the request and response objects
* ****Next:**** It passes management to a consecutive matching route.

#### **10. How to Redirect 404 Errors to A Page In ExpressJS?**

****Answer:****  
In server.js add the subsequent code to send 404 errors back to a page in our ExpressJS App:

/\* Define fallback route \*/  
app.use(function(req, res, next) {  
res.status(404).json({errorCode: 404, errorMsg: "route not found"});  
});

**Q:- What is Express JS?**

1. Express.js is a free open-source, flexible node.js web application framework.
2. It is designed for building backend web applications and APIs, etc.
3. It has been developed by TJ Holowaychuk in 2010 and written in JavaScript.

**Express.js key features:**

Following are the key features of express framework:

1. **Middlewares:**Express middleare are functions that access of - request (req), response (res) and next().
2. **Routing:**It is used to define routes in order to perform and handle HTTP requests/operations.
3. **Templates Engine:**It has SSR renders html template used to render the HTML on the browser.
4. **High Performance:**Express prepare a thin layer, therefore, the performance is adequate.
5. **Database Support:**Express supports RDBMS as well as NoSQL databases - MySQL, MongoDB, etc.
6. **MVC Support:**It is a web application which supports MVC architecture.
7. **ORM Support:**It support various ORM/ODM - Mongoose, Sequelize, etc.

**Q:- What is Scaffolding in Express.js?**

Scaffolding is creating the skeleton structure of application

There are 2 way to do this, by using:

1. Express Application Generator
2. Yeoman

**1. Express Application Generator:**

Use express-generator tool to quickly create an skeleton of your application.

npm install -g express-generator express myTestExpressApp

Th above command will create your project named - myTestExpressApp with followings.

1. **bin:**It is also called **www** is the main configuration file of the app.
2. **public:**It contains static assets - javascript, css, images, etc.
3. **routes:** It contains route files of the app.
4. **views:** It contains the view files (HTML) of the application.
5. **app.js:** It is the main file of the application which execute first.
6. **package.json:**It is the manifest file. It contains all metadata info of the project.

Install all the dependencies mentioned in the package.json file:

cd myTestExpressApp npm install

**Q:- How to enable debugging in express app?**

In different Operating Systems, we have following commands:

**On Linux:**

DEBUG=express:\* node app.js

**On Windows:**

set DEBUG=express:\* node app.js

**Q:- Serving static files in Express.js?**

app.use(express.static('public')) app.use('/static', express.static(path.join(\_\_dirname, 'public')));

**Q:- What is routing and how routing works in Express.js?**

Routing refers to determining how an application responds to a request.

**Route Syntax:**

app.METHOD(PATH, HANDLER);

**Where:**

* **app** is an instance of **express**.
* **METHOD** is an HTTP request method (get, post, put, etc.).
* **PATH** is a path/endpoint on the server.
* **HANDLER** is a function executed when the route is matched.

**#Example: A route with path / and get method.**

app.get('/', function (req, res) { res.send('Express.js Interview Questions') })

**Q:- How dynamic routing works in express.js?**

When someone pass parameters in URL (i.e. Parametrized URL), this routing phenomenon is called dynamic routing.

var express = require('express'), app = express(); app.get('/article/:id', function(req , res){ res.render('article' + req.params.id); })

In above example: id is a parameters, which can be different for different request.

**Q:- What is Middleware in express.js?**

Middleware is a function that is invoked by the express routing layer before the final request processed.

**Middleware functions can perform the following tasks:**

1. Execute any code - validation, setting headers, etc.
2. You can make changes to the request (req) and response (res) objects.
3. You can also end the request-response cycle, if rquired.
4. You can call the next middleware function in the stack to proceed and process the final request.

*If the current middleware function does not end the request-response cycle, it must call next() to pass control to the next middleware function. Otherwise, the request will be left hanging.*

**Types of Middleware:**

1. Application-level middleware
2. Router-level middleware
3. Error-handling middleware
4. Built-in middleware
5. Third-party middleware

**1. Application-level middleware:**

This kind of middleware method is bind to the app Object using app.use() method. It applies on all routes.

//This middleware will execute for each route. app.use(function (req, res, next) { console.log('Current Time:', Date.now()) next() })

**2. Router-level middleware:**

Router-level middleware binds to an specific instance of **express.Router()**

**3. Built-in middleware:**

Express has the following built-in middleware functions:

* **express.static** serves static assets such as HTML files, images, and so on.
* **express.json** parses incoming requests with JSON payloads.
* **express.urlencoded** parses incoming requests with URL-encoded payloads.

**4. Third-party middleware:**

There are a lots of third party middleware, such as

* Body-parser
* Cookie-parser
* Mongoose
* Sequelize
* Cors
* Express-validator

To handle HTTP POST request in express.js version 4 and above, you need to install a middleware module called body-parser, body-parser extract the entire body portion of an incoming request stream and exposes it on req.body, this middleware was a part of express.js earlier but now you have to install it separately.

These can be installed by using command:

>> npm install MODULE\_NAME

And they can be loaded using requires and used later.

#Example:

var bodyParser = require('body-parser'); app.use(bodyParser.json()); app.use(bodyParser.urlencoded({ extended: false }))

**Note:** Multiple Middleware can be used as an array on a single route.

var middlewareArray = [middleware1, middleware2] app.get('/home', middlewareArray, function (req, res, next) { //Code snippets })

**Q:- Database integration in express.js?**

Express.js supports of many RDBMS & NoSQL databases like

* MongoDB
* MySQL
* Oracle
* PostgreSQL
* SQL Server
* SQLite

#Example: Install MongoDB

>>npm install mongodb var MongoClient = require('mongodb').MongoClient MongoClient.connect('mongodb://localhost:27017/test\_db', function (err, db) { if (err) throw err db.collection('mammals').find().toArray(function (err, result) { if (err) throw err console.log(result) }) })

**Q:- Error handling in Express.js?**

var express = require('express'), app = express(); app.use(function (err, req, res, next) { console.error(err.stack) // error first callback res.status(500).send('Something went wrong!') })

var express = require('express'), app = express(); app.use(function(req, res, next) { res.status(404).json({ errorCode: 404, errorMsg: "Not found!" }); });

**Q:- How to allow CORS in Express.js (Node.js)? Explain with an Example?**

*Cross-origin resource sharing (CORS) is a mechanism that allows restricted resources to be requested from another domain/server.*

**There are mainly three ways you can do this using**

1. res.setHeader() - It allow to set only single header
2. res.header() OR res.set() - It allow to set multiple headers.
3. express cors module

**1. Enable CORS for all resources using res.setHeader().**

app.use(function(req, res, next) { // Website you wish to allow to connect res.setHeader("Access-Control-Allow-Origin", "\*"); // Request methods you wish to allow res.setHeader( "Access-Control-Allow-Methods", "GET, POST, OPTIONS, PUT, PATCH, DELETE" ); // Request headers you wish to allow res.setHeader( "Access-Control-Allow-Headers", "X-Requested-With,content-type" ); // Set to true if you need the website to include cookies in the requests sent // to the API (e.g. in case you use sessions) res.setHeader("Access-Control-Allow-Credentials", true); // Pass to next layer of middleware next(); });

**2. Enable CORS for all resources using res.header()**

app.use(function(req, res, next) { res.header("Access-Control-Allow-Origin", "\*"); res.header( "Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept" ); next(); });

**3. Enable CORS for all resources using express CORS module.**

If you are using **express.js** you can use **[cors module](https://expressjs.com/en/resources/middleware/cors.html" \t "https://www.fullstacktutorials.com/interviews/_blank)**

**3.1) Enable CORS Requests for All Servers.**

var express = require("express"); var cors = require("cors"); var app = express(); app.use(cors()); app.get("/", function(req, res, next) { res.json({ msg: "This is CORS-enabled for all origins!" }); }); app.listen(80, function() { console.log("CORS-enabled web server listening on port 80"); });

**3.2) Enable CORS Requests for Some Specific Servers**

var express = require("express"); var cors = require("cors"); var app = express(); var whitelist = ["http://server1.com", "http://server2.com"]; var corsOptions = { origin: function(origin, callback) { if (whitelist.indexOf(origin) !== -1) { callback(null, true); } else { callback(new Error("Not allowed by CORS")); } } }; app.get("/", cors(corsOptions), function(req, res, next) { res.json({ msg: "This is CORS-enabled for a whitelisted domain." }); }); app.listen(80, function() { console.log("CORS-enabled web server listening on port 80"); });

**Q:- How to implement JWT authentication in Express app ? Explain with an Example?**

* Create a folder called 'keys' inside project folder.
* Install some dependencies as following: npm install jsonwebtoken –-save
* Add the login router routes/index.js

router.post('/login, function(req, res) { // find the user User.findOne({ name: req.body.username }, function(err, res) { if (err) throw err; if (!res) { res.json({ success: false, message: Login failed.' }); } else if (res) { // check if password matches if (res.password != req.body.password) { res.json({ success: false, message: Login failed. Wrong password.' }); } else { var token = jwt.sign(res, app.get('superSecret'), { expiresInMinutes: 1600 }); // return the information including token as JSON res.json({ success: true, message: 'Valid token!', token: token }); } } }); });

* Use the token in application

jwt = require("express-jwt"); app.use(function(req, res, next) { var token = req.body.token || req.query.token || req.headers['x-access-token']; if (token) { jwt.verify(token, app.get('superSecret'), function(err, decoded) { if (err) { return res.json({ success: false, message: 'Invalid token.' }); } else { req.decoded = decoded; next(); } }); } else { return res.status(403).send({ success: false, message: 'No token given.' }); } });

### ****1. Differentiate between JavaScript and Node.js.****

|  |  |  |
| --- | --- | --- |
| ****JavaScript vs Node.js**** | | |
| **Features** | ****JavaScript**** | ****Node.js**** |
| Type | Programming Language | Interpreter and environment for JavaScript |
| Utility | Used for any client-side activity for a web application | Used for accessing or performing any non-blocking operation of any operating system |
| Running Engine | Spider monkey (FireFox), JavaScript Core (Safari), V8 (Google Chrome), etc. | V8 (Google Chrome) |

### ****2. What Is Node.js?****

Node.js is an extremely powerful framework developed on ****Chrome’s V8 JavaScript engine**** that compiles the JavaScript directly into the native machine code. It is a lightweight framework used for creating server-side web applications and extends JavaScript API to offer usual server-side functionalities. It is generally used for large-scale application development, especially for video streaming sites, single page application, and other web applications.

### ****3. List down the major benefits of using Node.js?****

|  |  |
| --- | --- |
| ****Features**** | ****Description**** |
| ****Fast**** | Node.js is built on Google Chrome’s V8 JavaScript Engine which makes its library very fast in code execution |
| ****Asynchronous**** | Node.js based server never waits for an API to return data thus making it asynchronous |
| ****Scalable**** | It is highly scalable because of its event mechanism which helps the server to respond in a non-blocking way |
| ****Open Source**** | Node.js has an extensive open source community which has contributed in producing some excellent modules to add additional capabilities to Node.js applications |
| ****No Buffering**** | Node.js applications simply output the data in chunks and never buffer any data |

### ****4. What is the difference between Angular and Node.js?****

|  |  |
| --- | --- |
| ****Angular**** | ****Node.js**** |
| 1. It is an ****open source**** web application development framework | 1. It is a cross-platform ****run-time environment**** for applications |
| 2. It is written in TypeScript | 2. It is written in ****C, C++ and JavaScript**** languages |
| 3. Used for building ****single-page**** client-side web applications | 3. Used for building f****ast and scalable****server-side networking applications |
| 4. Angular itself is a ****web application framework**** | 4. Node.js has many ****different frameworks****like Sails.js, Partial.js, and Express.js, etc. |
| 5. Ideal for creating ****highly active and interactive**** web apps | 5. Ideal for developing ****small size****projects |
| 6. Helpful in ****splitting an app into MVC components**** | 6. Helpful in ****generating database queries**** |
| 7. Suitable for developing ****real-time applications**** | 7. Suitable in situations where something ****faster and more scalable**** is required |

### ****5. Why Node.js is single threaded?****

Node.js uses a single threaded model in order to support async processing. With async processing, an application can perform better and is more scalable under web loads. Thus, Node.js makes use of a [single-threaded model](https://www.edureka.co/blog/nodejs-tutorial/" \l "nodearchitecture" \t "https://www.edureka.co/blog/interview-questions/top-node-js-interview-questions-2016/_blank) approach rather than typical thread-based implementation.

[](https://www.edureka.co/nodejs-certification-training?utm_source=blogbanner&utm_campaign=curriculum" \t "https://www.edureka.co/blog/interview-questions/top-node-js-interview-questions-2016/_blank)

### **[Node.js Certification Training Course](https://www.edureka.co/nodejs-certification-training?utm_source=blogbanner&utm_campaign=curriculum" \t "https://www.edureka.co/blog/interview-questions/top-node-js-interview-questions-2016/_blank)**

[Explore Curriculum](https://www.edureka.co/nodejs-certification-training?utm_source=blogbanner&utm_campaign=curriculum" \t "https://www.edureka.co/blog/interview-questions/top-node-js-interview-questions-2016/_blank)

### ****6. How do Node.js works?****

Node.js is a virtual machine that uses JavaScript as its scripting language and runs on a v8 environment. It works on a single-threaded event loop and a non-blocking I/O which provides high rate as it can handle a higher number of concurrent requests. Also, by making use of the [‘HTTP’ module](https://www.edureka.co/blog/nodejs-tutorial/" \l "http" \t "https://www.edureka.co/blog/interview-questions/top-node-js-interview-questions-2016/_blank), Node.js can run on any stand-alone web server.

### ****7. Where Node.js can be used?****

Node.js can be used to develop:

* Real-Time Web Applications
* Network Applications
* Distributed Systems
* General Purpose Applications

### ****8. How many types of API functions are there in Node.js?****

There are two types of API functions in Node.js:

* Asynchronous, non-blocking functions
* Synchronous, blocking functions

### 9. **What is the difference between Asynchronous and Non-blocking?**

|  |  |
| --- | --- |
| ****Asynchronous**** | ****Non-blocking**** |
| Asynchronous means not synchronous. Using these we can make asynchronous HTTP requests that do not wait for the server to respond. These functions continue to respond to the request for which it has already received the server response. | Non-blocking functions are used in regards with I/O operations. They immediately respond with whatever data is available and keeps on running as per the requests. In case, any answer couldn’t be retrieved then the API returns immediately with an error. |

In case you are facing any challenges with these Node.js Interview Questions, please mention your problems in the section comment section below.

### ****10. What is package.json?****

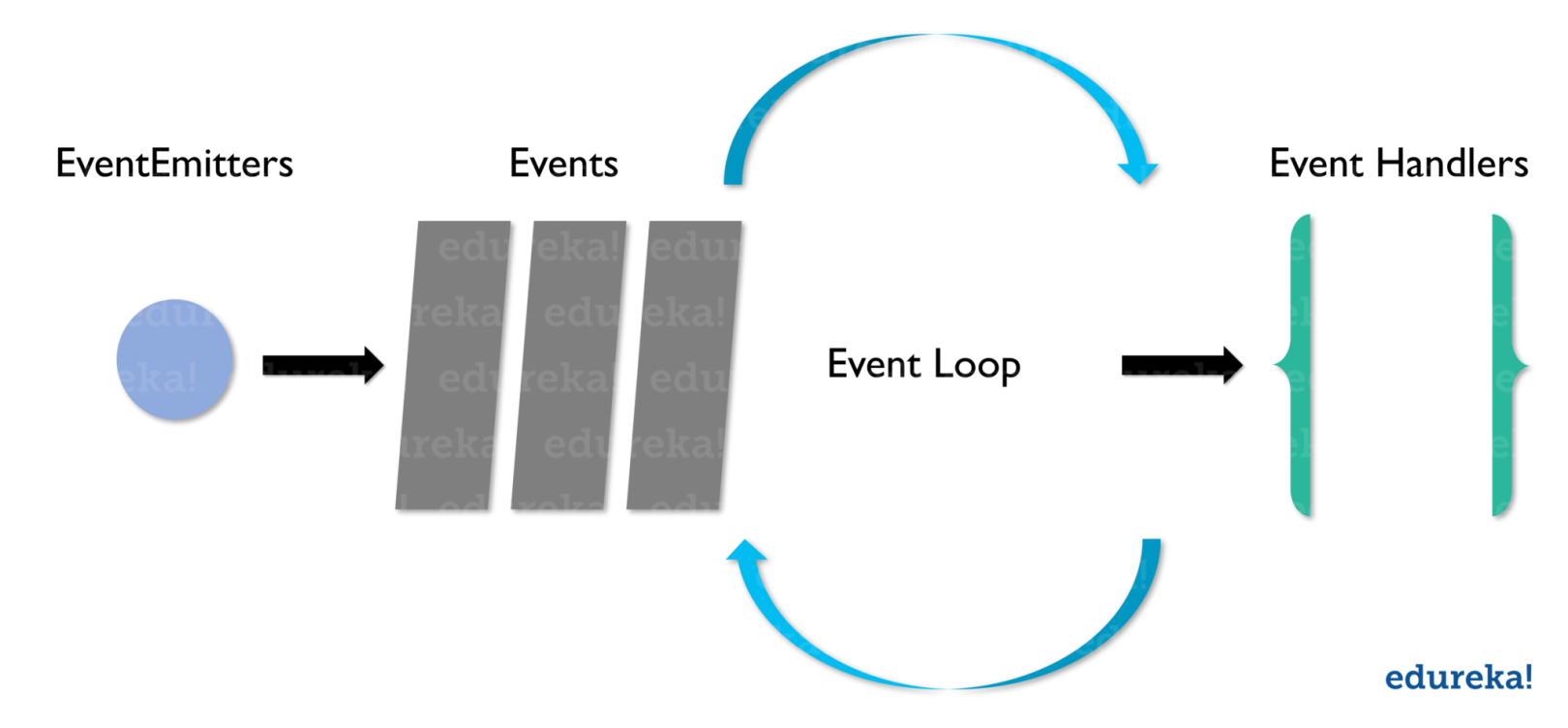
The **[package.json file](https://www.edureka.co/blog/nodejs-tutorial/" \l "json" \t "https://www.edureka.co/blog/interview-questions/top-node-js-interview-questions-2016/_blank)** in Node.js is the heart of the entire application. It is basically the manifest file that contains the metadata of the project where we define the properties of a package.

### **IMG_257**11. What do you understand by Event-driven programming?****

Event-driven programming is an approach that heavily makes use of events for triggering various functions. An event can be anything like a mouse click, key press, etc. When an event occurs, a call back function is executed that is already registered with the element. This approach mainly follows the publish-subscribe pattern. Because of [event-driven programming](https://www.edureka.co/blog/nodejs-tutorial/" \l "events" \t "https://www.edureka.co/blog/interview-questions/top-node-js-interview-questions-2016/_blank), Node.js is faster when compared to other technologies.

### ****12. What is an *Event loop* in Node.js and how does it work?****

An event loop in Node.js handles all the asynchronous callbacks in an application. It is one of the most important aspects of Node.js and the reason behind Node.js have non-blocking I/O. Since Node.js is an event-driven language, you can easily attach a listener to an event and then when the event occurs the callback will be executed by the specific listener. Whenever functions like setTimeout, http.get, and fs.readFile are called, Node.js executed the event loop and then proceeds with the further code without waiting for the output. Once the entire operation is finished, Node.js receives the output and then executes the callback function. This is why all the callback functions are placed in a queue in a loop. Once the response is received, they are executed one by one.



### ****13. Explain  REPL in the context of Node.js.****

REPL in Node.js stands for ****R****ead, ****E****val, ****P****rint, and ****L****oop. It represents a computer environment such as a window console or Unix/Linux shell where any command can be entered and then the system can respond with an output. Node.js comes bundled with a REPL environment by default. REPL can perform the below-listed tasks:

* **Read:** Reads the user’s input, parses it into JavaScript data-structure and then stores it in the memory.
* **Eval:** Receives and evaluates the data structure.
* **Print:**Prints the final result.
* **Loop:** Loops the provided command until CTRL+C is pressed twice.

### ****14. List down the tasks which should be done asynchronously using the event loop?****

Below is the list of the tasks which must be done asynchronously using the event loop:

* I/O operations
* Heavy computation
* Anything requiring blocking

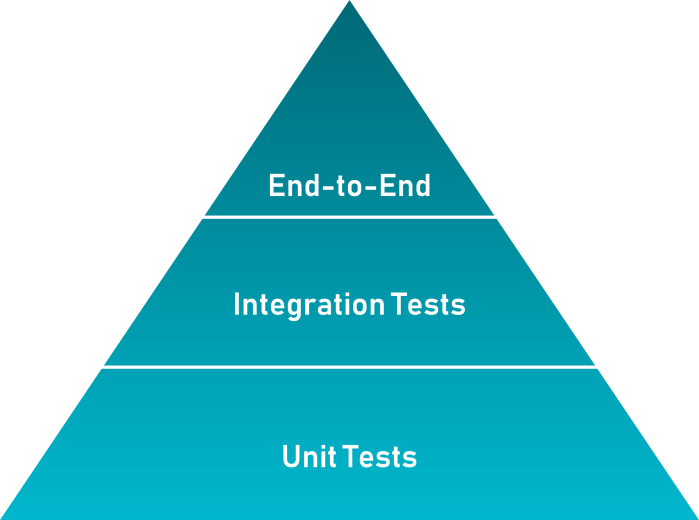
### ****15. List down the steps using which “Control Flow” controls the function calls in Node.js?****

1. Control the order of execution
2. Collect data
3. Limit concurrency
4. Call the next step in the program

Want to upskill yourself to get ahead in your career? Check out this video

## 

### ****16. What do you understand by a test pyramid?****

A test pyramid basically is a diagram that describes the ratio of how many unit tests, integration tests, and end-to-end test are required to be written for the successful development of the project.

### ****17. What is an error-first callback in Node.js?****

Error-first callbacks in Node.js are used to pass errors and data. The very first parameter you need to pass to these functions has to be an error object while the other parameters represent the associated data. Thus you can pass the error object for checking if anything is wrong and handle it. In case there is no issue, you can just go ahead and with the subsequent arguments.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var myPost = **new** Post({title: 'edureka'});  myPost.save(function(err,myInstance){  **if**(err){  //handle error and return  }  //go ahead with `myInstance`  }); |

### ****18. Explain the purpose of module.exports?****

A module in Node.js is used to encapsulate all the related codes into a single unit of code which can be interpreted by shifting all related functions into a single file. For example, suppose you have a file called greet.js that contains the two functions as shown below:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | module.exports = {  greetInHindi: function(){  **return** "NAMASTE";  },  greetInKorean: function(){  **return** "ANNYEONGHASEYO";  }}; |

As you can see module.exports provide two functions which can be imported in another file using below code:

|  |  |
| --- | --- |
| 1  2  3 | var eduGreets = require ("./greet.js");  eduGreets.greetInHindi() //NAMASTE  eduGreets.greetInKorean() //ANNYEONGHASEYO |

### ****19. What do you understand by Reactor Pattern in Node.js?****

****Reactor Pattern**** in Node.js is basically a concept of non-blocking I/O operations. This pattern provides a handler that is associated with each I/O operation and as soon as an I/O request is generated, it is then submitted to a demultiplexer. This demultiplexer is a notification interface which is capable of handling concurrency in non-blocking I/O mode. It also helps in collecting each and every request in the form of an event and then place each event in a queue. Thus resulting in the generation of the Event Queue. Simultaneously, we have our event loop which iterates the events present in the Event Queue.

### ****20. What’s the difference between ‘front-end’ and ‘back-end’ development?****

|  |  |
| --- | --- |
| ****Front-end Development**** | ****Back-end Development**** |
| 1. Uses mark up and web languages like HTML, CSS, [JavaScript](https://www.edureka.co/blog/what-is-javascript/" \t "https://www.edureka.co/blog/interview-questions/top-node-js-interview-questions-2016/_blank) | 1. Uses programming and scripting languages like [Python](https://www.edureka.co/blog/python-tutorial/" \t "https://www.edureka.co/blog/interview-questions/top-node-js-interview-questions-2016/_blank), [Ruby](https://www.edureka.co/blog/ruby-on-rails-tutorial/" \t "https://www.edureka.co/blog/interview-questions/top-node-js-interview-questions-2016/_blank), Perl, etc. |
| 2. Based on asynchronous requests and AJAX | 2. Based on Server Architecture |
| 3. Better Accessibility | 3. Enhanced Security |
| 4. Used for SEO | 4. Used for Backup |

### ****21. What are LTS releases of Node.js?****

****LTS****stands ****L****ong ****T****erm ****S****upport version of Node.js that receives all the critical bug fixes along with security updates and performance improvements. These versions are supported for at least 18 months and mainly focus on stability and security. The modifications done to the LTS versions are restricted to the bug fixes, security upgrade, npm, and documentation updates, performance improvement, etc.

### ****22. List down the major security implementations within Node.js?****

Major security implementations in Node.js are:

1. Authentications
2. Error Handling

### ****23. What do you understand by callback hell?****

Callback Hell is also known as the Pyramid of Doom. It is a pattern caused by intensively nested callbacks which are unreadable and unwieldy. It typically contains multiple nested callback functions which in turn make the code hard to read and debug. It is caused by improper implementation of the asynchronous logic.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | async\_A(function(){  async\_B(function(){  async\_C(function(){  async\_D(function(){  ....  });  });  });  }); |

In case you are facing any challenges with these Node.js Interview Questions, please mention your problems in the section comment section below.

### ****24. Explain libuv.****

Libuv is a multi-platform support [library of Node.js](https://www.edureka.co/blog/node-js-npm-tutorial/" \t "https://www.edureka.co/blog/interview-questions/top-node-js-interview-questions-2016/_blank) which majorly is used for asynchronous I/O. It was primarily developed for Node.js,  with time it is popularly practiced with other systems like as Luvit, pyuv, Julia, etc. Libuv is basically an abstraction around libev/ IOCP depending on the platform, providing users an API based on libev. A few of the important features of libuv are:

* Full-featured event loop backed
* File system events
* Asynchronous file & file system operations
* Asynchronous TCP & UDP sockets
* Child processes

### ****25. Explain the concept of middleware in Node.js?****

In general, middleware is a function receives the Request and Response objects. In other words, in an application’s request-response cycle these functions have access to various request &  response objects along with the next function of the cycle. The next function of middleware is represented with the help of a variable, usually named next. Most commonly performed tasks by the middleware functions are:

* Execute any type of code
* Update or modify the request and the response objects
* Finish the request-response cycle
* Invoke the next middleware in the stack

### ****26. Explain the concept of URL module.****

The **URL module** of Node.js provides various utilities for **URL** resolution and parsing. It is a built-in module that helps in splitting up the web address into a readable format:

|  |  |
| --- | --- |
| 1 | var url = require('url'); |

**For example:**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | var url = require('url');  var adrs = '<http://localhost:8082/default.htm?year=2019&amp;amp;month=april>';  var q = url.parse(adr, **true**);  console.log(q.host); //returns 'localhost:8082'  console.log(q.pathname); //returns '/default.htm'  console.log(q.search); //returns '?year=2019 and month=april'  var qdata = q.query; //returns an object: { year: 2019, month: 'april' }  console.log(qdata.month); //returns 'april' |

### ****27. What do you understand by ESLint?****

ESLint is an open source project initially developed by Nicholas C. Zakas in 2013 which aims to provide a linting utility for JavaScript through a plug. Linters in Node.js are good tools for searching certain bug classes, especially those which are related to the variable scope.

### ****28. For Node.js, why Google uses V8 engine?****

Google uses V8 as it is a Chrome runtime engine that converts [JavaScript code](https://www.edureka.co/blog/data-types-in-javascript/" \t "https://www.edureka.co/blog/interview-questions/top-node-js-interview-questions-2016/_blank) into native machine code. This, in turn, speeds up the application execution and response process and give you a fast running application.

### ****29. Explain the working of the control flow function.****

In Node.js, the control flow function is basically the code that is executed between the asynchronous function calls. Below are the steps that must be followed for executing it:

1. Firstly, the order of execution must be controlled.
2. Then, the required data need to be collected.
3. Next, the concurrency must be limited.
4. Once done, the next step of the program has to be invoked.

### ****30. List down the two arguments that async.queue takes as input?****

Below are the two arguments that async.queue takes as input:

1. Task Function
2. Concurrency Value

### ****31. Differentiate between spawn() and fork() methods in Node.js?****

In Node.js, the spawn() is used to launch a new process with the provided set of commands. This method doesn’t create a new V8 instance and just one copy of the node module is active on the processor. When your child process returns a large amount of data to the Node you can invoke this method.

Syntax:

|  |  |
| --- | --- |
| 1 | child\_process.spawn(command[, args][, options]) |

Whereas, the fork() in Node.js is a special instance of spawn() that executes a new instance of the V8 engine. This method simply means that multiple workers are running on a single Node code base for various task.

Syntax:

|  |  |
| --- | --- |
| 1 | child\_process.fork(modulePath[, args][, options]) |

In case you are facing any challenges with these Node.js Interview Questions, please mention your problems in the section comment section below.

### ****32. What do you understand by global objects in Node.js?****

In Node.js, Globals are the objects which are global in nature and are available in all the modules of the application. You can use these objects directly in your application, rather than having to include them explicitly. The global objects can be modules, functions, strings, object, etc. Moreover, some of these objects can be in the module scope instead of global scope.

### ****33. Explain the concept of stub in Node.js.****

In Node.js, stubs are basically the programs or functions that are used for stimulating the module or component behavior. During any test cases, stubs provide the canned answers of the functions.

### ****34. How assert works in Node.js?****

In Node.js, assert is used to write tests. It only provides feedback only when any of the running test cases fails. This module gives you a set of assertion tests which are then used for testing invariants. It is basically used internally by Node.js but using require(‘assert’) code, it can be used in other applications as well.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | var **assert** = require('assert');  function mul(a, b) {  **return** a \* b;  }  var result = mul(1,2);  **assert**( result === 2, 'one multiplied by two is two'); |

### ****35. Define the concept of the test pyramid. Explain the process to implement them in terms of HTTP APIs.****

The test**pyramid** is basically a concept that is developed by Mike Cohn. According to this, you should have a higher number of low-level unit **tests** as compared to high-level end-to-end **tests that**running through a GUI.

In terms of HTTP APIs it may be defined as:

* A higher number of low-level unit tests for each model
* Lesser integration tests to test model interactions
* Lesser acceptance tests for testing actual HTTP endpoints

### ****36. Explain the purpose of ExpressJS package?****

Express.js is a framework built on top of Node.js that facilitates the management of the flow of data between server and routes in the server-side applications.  It is a lightweight and flexible framework that provides a wide range of features required for the web as well as mobile application development. Express.js is developed on the middleware [module of Node.js](https://www.edureka.co/blog/what-is-node-js/" \t "https://www.edureka.co/blog/interview-questions/top-node-js-interview-questions-2016/_blank) called **connect**. The connect module further makes use of ****http**** module to communicate with Node.js. Thus, if you are working with any of the connect based middleware modules, then you can easily integrate with Express.js.

### ****37. Differentiate between process.nextTick() and setImmediate()?****

In Node.js, process.nextTick() and setImmediate(), both are functions of the Timers module which help in executing the code after a predefined period of time. But these functions differ in their execution. The process.nextTick function waits for the execution of action till the next pass around in the event loop or once the event loop is completed only then it will invoke the callback function. On the other hand, setImmediate() is used to execute a callback method on the next cycle of the event loop which eventually returns it to the event loop in order to execute the I/O operations.

### ****38. Explain the usage of a buffer class in Node.js?****

Buffer class in Node.js is used for storing the raw data in a similar manner of an array of integers. But it corresponds to a raw memory allocation that is located outside the V8 heap. It is a global class that is easily accessible can be accessed in an application without importing a buffer module. Buffer class is used because pure JavaScript is not compatible with binary data. So, when dealing with TCP streams or the file system, it’s necessary to handle octet streams.

### ****39. How does Node.js handle the child threads?****

In general, Node.js is a single threaded process and doesn’t expose the child threads or thread management methods. But you can still make use of the child threads using spawn() for some specific asynchronous I/O tasks which execute in the background and don’t usually execute any JS code or hinder with the main event loop in the application. If you still want to use the threading concept in your application you have to include a module called ChildProcess explicitly.

### ****40. Explain stream in Node.js along with its various types.****

Streams in Node.js are the collection of data similar to arrays and strings. They are objects using which you can read data from a source or write data to a destination in a continuous manner. It might not be available at once and need not to have fit in the memory. These streams are especially useful for reading and processing a large set of data. In Node.js, there are four fundamental types of streams:

1. Readable: Used for reading large chunks of data from the source.
2. Writeable: Use for writing large chunks of data to the destination.
3. Duplex: Used for both the functions; read and write.
4. Transform: It is a duplex stream that is used for modifying the data.

## ****Advanced Node.js Interview Questions for Experienced Professional****

### ****41. What is the use of NODE\_ENV?****

If the project is in the production stage, Node.js promotes the convention of making use of NODE\_ENV variable to flag it. This helps in taking better judgment during the development of the projects. Also, when you set your NODE\_ENV to production, your application tends to perform 3 times faster.

### ****42. Differentiate between readFile vs createReadStream in Node.js?****

Node.js provides two ways to read and execute files which are using readFile and CreateStream. readFile() is a fully buffered process which returns the response only when the complete file is pushed into the buffer and is read. It is a memory intensive process and in case of large files, the processing can be very slow. Whereas createReadStream is a partially buffered which treats the entire process as an event series. The entire file is split into chunks which are then processed and sent back as a response one by one. Once done, they are finally removed from the buffer. Unlike readFile, createReadStream is really effective for the processing of the large files.

### ****43. List down the various timing features of Node.js.****

Node.js provides a Timers module which contains various functions for executing the code after a specified period of time. Below I have listed down the various functions provided by this module:

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

### ****44. Explain the concept of Punycode in Node.js?****

In Node.js, Punycode is an encoding syntax that is used for converting Unicode (UTF-8) string of characters into a basic ASCII string of characters. It is important as the hostnames can only understand the ASCII characters. Thus, Node.js version 0.6.2 onwards, it was bundled up with the default Node package. If you want to use it with any previous versions, you can easily do that by using the following code:

#### Syntax:

|  |  |
| --- | --- |
| 1 | punycode = require('punycode'); |

### ****45. Differentiate between Node.js vs Ajax?****

The most basic difference between Node.js and Ajax that, Node.js is a server-side JavaScript whereas Ajax is a client-side technology. In simpler terms, Ajax is mostly used for updating or modifying the webpage contents without having to refresh it. On the other hand, Node.js is required to develop the server software that are typically executed by the servers instead of the web browsers.

### ****46. Does Node.js provide any Debugger?****

Node.js do provide a simple TCP based protocol and debugging client that comes built-in. In order to debug your JavaScript file, you can use the below debug argument followed by ****js**** file name that you want to debug.

**Syntax:**

|  |  |
| --- | --- |
| 1 | node debug [script.js | -e "script" | &amp;lt;host&amp;gt; : &amp;lt;port&amp;gt; ] |

In case you are facing any challenges with these Node.js Interview Questions, please mention your problems in the section comment section below.

### ****47. Describe the exit codes of Node.js.****

In Node.js, exit codes are a set of specific codes which are used for finishing a specific process. These processes can include the global object as well. Below are some of the exit codes used in Node.js:

* Uncaught fatal exception
* Unused
* Fatal Error
* Internal Exception handler Run-time failure
* Internal JavaScript Evaluation Failure

### ****48. What do you understand by an Event Emitter in Node.js?****

EventEmitter is a Node.js class that includes all the objects that are capable of emitting events. These objects contain an eventEmitter.on() function through which more than one function can be attached to the named events that are emitted by the object. Whenever an EventEmitter object throws an event, all the attached functions to that specific event are invoked synchronously. Below code shows how to us the EventEmitter in your application:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | **const** EventEmitter = require('events');  **class** MyEmitter **extends** EventEmitter { }  **const** myEmitter = **new** MyEmitter();  myEmitter.on('event', () =&amp;gt; {  console.log('an event occurred!');  });  myEmitter.emit('event'); |

### ****49. Is cryptography supported in Node.js?****

Yes, Node.js does support cryptography through a module called Crypto. This module provides various cryptographic functionalities like cipher, decipher, sign and verify functions, a set of wrappers for open SSL’s hash HMAC etc. For example:

Syntax:

|  |  |
| --- | --- |
| 1  2  3  4 | **const** crypto = require'crypto');  **const** secret = 'akerude';  **const** hash = crypto.createHmac('swaEdu', secret).update('Welcome to Edureka').digest('hex');  console.log(hash); |

### ****50. Explain the reason as to why Express ‘app’ and ‘server’ must be kept separate.****

Express ‘app’ and ‘server’ must be kept separate as by doing this, you will be separating the API declaration from the network related configuration which benefits in the below listed ways:

* It allows testing the API in-process without having to perform the network calls
* Faster testing execution
* Getting wider coverage metrics of the code
* Allows deploying the same API under flexible and different network conditions
* Better separation of concerns and cleaner code

API declaration should reside in app.js:

|  |  |
| --- | --- |
| 1  2  3  4 | 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:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | var app = require('../app');  var http = require('http');  //Get port from environment and store in Express  var port = normalizePort(process.env.PORT || '8000');  app.set('port', port);  //Create HTTP server.  var server = http.createServer(app); |

### **1. What is a first class function in Javascript?**

When functions can be treated like any other variable then those functions are first-class functions. There are many other programming languages, for example, scala, Haskell, etc which follow this including JS. Now because of this function can be passed as a param to another function(callback) or a function can return another function(higher-order function). map() and filter() are higher-order functions that are popularly used.

### **2. What is Node.js and how it works?**

Node.js is a virtual machine that uses JavaScript as its scripting language and runs Chrome’s V8 JavaScript engine. Basically, Node.js is based on an event-driven architecture where I/O runs asynchronously making it lightweight and efficient. It is being used in developing desktop applications as well with a popular framework called electron as it provides API to access OS-level features such as file system, network, etc.

### **3. How do you manage packages in your node.js project?**

It can be managed by a number of package installers and their configuration file accordingly. Out of them mostly use npm or yarn. Both provide almost all libraries of javascript with extended features of controlling environment-specific configurations. To maintain versions of libs being installed in a project we use package.json and package-lock.json so that there is no issue in porting that app to a different environment.

### **4. How is Node.js better than other frameworks most popularly used?**

* Node.js provides simplicity in development because of its non-blocking I/O and even-based model results in short response time and concurrent processing, unlike other frameworks where developers have to use thread management.
* It runs on a chrome v8 engine which is written in c++ and is highly performant with constant improvement.
* Also since we will use Javascript in both the frontend and backend the development will be much faster.
* And at last, there are ample libraries so that we don’t need to reinvent the wheel.

### **5. Explain the steps how “Control Flow” controls the functions calls?**

* Control the order of execution
* Collect data
* Limit concurrency
* Call the following step in the program.

### **6. What are some commonly used timing features of Node.js?**

* ****setTimeout/clearTimeout****– This is used to implement delays in code execution.
* ****setInterval/clearInterval**** – This is used to run a code block multiple times.
* ****setImmediate/clearImmediate**** – Any function passed as the setImmediate() argument is a callback that's executed in the next iterationof the event loop.
* ****process.nextTick**** – Both setImmediate and process.nextTick appear to be doing the same thing; however, you may prefer one over the other depending on your callback’s urgency.

### **7. What are the advantages of using promises instead of callbacks?**

The main advantage of using promise is you get an object to decide the action that needs to be taken after the async task completes. This gives more manageable code and avoids callback hell.

### **8. What is fork in node JS?**

A fork in general is used to spawn child processes. In node it is used to create a new instance of v8 engine to run multiple workers to execute the code.

### **9. Why is Node.js single-threaded?**

Node.js was created explicitly as an experiment in async processing. This was to try a new theory of doing async processing on a single thread over the existing thread-based implementation of scaling via different frameworks.

### **10. How do you create a simple server in Node.js that returns Hello World?**

**var** http = require("http");

http.createServer(**function** (request, response) {

response.writeHead(200, {'Content-Type': 'text/plain'});

response.end('Hello World\n');

}).listen(3000);

### **11. How many types of API functions are there in Node.js?**

There are two types of API functions:

* ****Asynchronous, non-blocking functions**** - mostly I/O operations which can be fork out of the main loop.
* ****Synchronous, blocking functions**** - mostly operations that influence the process running in the main loop.

### **12. What is REPL?**

PL in Node.js stands for ****R****ead, ****E****val, ****P****rint, and ****L****oop, which further means evaluating code on the go.

### **13. List down the two arguments that async.queue takes as input?**

* Task Function
* Concurrency Value

### **14. What is the purpose of module.exports?**

This is used to expose functions of a particular module or file to be used elsewhere in the project. This can be used to encapsulate all similar functions in a file which further improves the project structure.  
  
For example, you have a file for all utils functions with util to get solutions in a different programming language of a problem statement.

const getSolutionInJavaScript = async ({

problem\_id

}) => {

...

};

const getSolutionInPython = async ({

problem\_id

}) => {

...

};

module.exports = { getSolutionInJavaScript, getSolutionInPython }

Thus using module.exports we can use these functions in some other file:

const { getSolutionInJavaScript, getSolutionInPython} = require("./utils")

### **15. What tools can be used to assure consistent code style?**

ESLint can be used with any IDE to ensure a consistent coding style which further helps in maintaining the codebase.

## **Intermediate Node.js Interview Questions**

### **16. What do you understand by callback hell?**

async\_A(function(){

async\_B(function(){

async\_C(function(){

async\_D(function(){

....

});

});

});

});

For the above example, we are passing callback functions and it makes the code unreadable and not maintainable, thus we should change the async logic to avoid this.

### **17. What is an event-loop in Node JS?**

Whatever that is async is managed by event-loop using a queue and listener.  We can get the idea using the following diagram:

Node.js Event Loop

So when an async function needs to be executed(or I/O) the main thread sends it to a different thread allowing v8 to keep executing the main code. Event loop involves different phases with specific tasks such as timers, pending callbacks, idle or prepare, poll, check, close callbacks with different FIFO queues. Also in between iterations it checks for async I/O or timers and shuts down cleanly if there aren't any.

### **18. If Node.js is single threaded then how does it handle concurrency?**

The main loop is single-threaded and all async calls are managed by libuv library.

For example:

**const** crypto = require("crypto");**const** start = Date.now();**function** **logHashTime**() {

crypto.pbkdf2("a", "b", 100000, 512, "sha512", () => {console.log("Hash: ", Date.now() - start);

});

}

logHashTime();

logHashTime();

logHashTime();

logHashTime();

This gives the output:

Hash: 1213Hash: 1225Hash: 1212Hash: 1222

This is because libuv sets up a thread pool to handle such concurrency. How many threads will be there in the thread pool depends upon the number of cores but you can override this.

### **19. Differentiate between process.nextTick() and setImmediate()?**

Both can be used to switch to an asynchronous mode of operation by listener functions.   
  
process.nextTick() sets the callback to execute but setImmediate pushes the callback in the queue to be executed. So the event loop runs in the following manner  
****timers–>pending callbacks–>idle,prepare–>connections(poll,data,etc)–>check–>close callbacks****

In this process.nextTick() method adds the callback function to the start of the next event queue and setImmediate() method to place the function in the check phase of the next event queue.

### **20. How does Node.js overcome the problem of blocking of I/O operations?**

Since the node has an event loop that can be used to handle all the I/O operations in an asynchronous manner without blocking the main function.   
  
So for example, if some network call needs to happen it will be scheduled in the event loop instead of the main thread(single thread). And if there are multiple such I/O calls each one will be queued accordingly to be executed separately(other than the main thread).

Thus even though we have single-threaded JS, I/O ops are handled in a nonblocking way.

### **21. How can we use async await in node.js?**

Here is an example of using async-await pattern:

// this code is to retry with exponential backoff**function** **wait** (timeout) {

**return** **new** Promise((resolve) => {setTimeout(() => {

resolve()

}, timeout);

});

}**async** **function** **requestWithRetry** (url) {

**const** MAX\_RETRIES = 10;

**for** (**let** i = 0; i <= MAX\_RETRIES; i++) {**try** {

**return** **await** request(url);

} **catch** (err) {

**const** timeout = Math.pow(2, i);

console.log('Waiting', timeout, 'ms');

**await** wait(timeout);

console.log('Retrying', err.message, i);

}

}

}

### **22. What is node.js streams?**

Streams are instances of EventEmitter which can be used to work with streaming data in Node.js. They can be used for handling and manipulating streaming large files(videos, mp3, etc) over the network. They use buffers as their temporary storage.  
  
There are mainly four types of the stream:

* ****Writable:**** streams to which data can be written (for example, fs.createWriteStream()).
* ****Readable:**** streams from which data can be read (for example, fs.createReadStream()).
* ****Duplex:****streams that are both Readable and Writable (for example, net.Socket).
* ****Transform:**** Duplex streams that can modify or transform the data as it is written and read (for example, zlib.createDeflate()).

### **23. What are node.js buffers?**

In general, buffers is a temporary memory that is mainly used by stream to hold on to some data until consumed. Buffers are introduced with additional use cases than JavaScript’s Unit8Array and are mainly used to represent a fixed-length sequence of bytes. This also supports legacy encodings like ASCII, utf-8, etc. It is a fixed(non-resizable) allocated memory outside the v8.

### **24. What is middleware?**

Middleware comes in between your request and business logic. It is mainly used to capture logs and enable rate limit, routing, authentication, basically whatever that is not a part of business logic. There are third-party middleware also such as body-parser and you can write your own middleware for a specific use case.

### **25. Explain what a Reactor Pattern in Node.js?**

Reactor pattern again a pattern for nonblocking I/O operations. But in general, this is used in any event-driven architecture.   
  
There are two components in this: 1. Reactor 2. Handler.  
  
****Reactor****: Its job is to dispatch the I/O event to appropriate handlers  
****Handler****: Its job is to actually work on those events

### **26. Why should you separate Express app and server?**

The server is responsible for initializing the routes, middleware, and other application logic whereas the app has all the business logic which will be served by the routes initiated by the server. This ensures that the business logic is encapsulated and decoupled from the application logic which makes the project more readable and maintainable.

### **27. For Node.js, why Google uses V8 engine?**

Well, are there any other options available? Yes, of course, we have [Spidermonkey](https://developer.mozilla.org/en-US/docs/Mozilla/Projects/SpiderMonkey" \t "https://www.interviewbit.com/node-js-interview-questions/_blank) from Firefox, Chakra from Edge but Google’s v8 is the most evolved(since it’s open-source so there’s a huge community helping in developing features and fixing bugs) and fastest(since it’s written in c++) we got till now as a JavaScript and WebAssembly engine. And it is portable to almost every machine known.

### **28. Describe the exit codes of Node.js?**

Exit codes give us an idea of how a process got terminated/the reason behind termination.   
  
A few of them are:

* Uncaught fatal exception - (code - 1) - There has been an exception that is not handled
* Unused - (code - 2) - This is reserved by bash
* Fatal Error - (code - 5) - There has been an error in V8 with stderr output of the description
* Internal Exception handler Run-time failure - (code - 7) - There has been an exception when bootstrapping function was called
* Internal JavaScript Evaluation Failure - (code - 4) - There has been an exception when the bootstrapping process failed to return function value when evaluated.

### **29. Explain the concept of stub in Node.js?**

Stubs are used in writing tests which are an important part of development. It replaces the whole function which is getting tested.    
  
This helps in scenarios where we need to test:

* External calls which make tests slow and difficult to write (e.g HTTP calls/ DB calls)
* Triggering different outcomes for a piece of code (e.g. what happens if an error is thrown/ if it passes)

For example, this is the function:

**const** request = require('request');**const** getPhotosByAlbumId = (id) => {**const** requestUrl = `https://jsonplaceholder.typicode.com/albums/${id}/photos?\_limit=3`;**return** **new** Promise((resolve, reject) => {

request.get(requestUrl, (err, res, body) => {

**if** (err) {

**return** reject(err);

}

resolve(JSON.parse(body));

});

});

};module.exports = getPhotosByAlbumId;

To test this **function** **this** **is** **the** **stubconst** **expect** = **require**('chai').**expect**;**const** request = require('request');**const** sinon = require('sinon');**const** getPhotosByAlbumId = require('./index');

describe('with Stub: getPhotosByAlbumId', () => {

before(() => {

sinon.stub(request, 'get')

.yields(null, null, JSON.stringify([

{

"albumId": 1,

"id": 1,

"title": "A real photo 1",

"url": "https://via.placeholder.com/600/92c952",

"thumbnailUrl": "https://via.placeholder.com/150/92c952"

},

{

"albumId": 1,

"id": 2,

"title": "A real photo 2",

"url": "https://via.placeholder.com/600/771796",

"thumbnailUrl": "https://via.placeholder.com/150/771796"

},

{

"albumId": 1,

"id": 3,

"title": "A real photo 3",

"url": "https://via.placeholder.com/600/24f355",

"thumbnailUrl": "https://via.placeholder.com/150/24f355"

}

]));

});

after(() => {

request.get.restore();

});

it('should getPhotosByAlbumId', (done) => {

getPhotosByAlbumId(1).then((photos) => {

expect(photos.length).to.equal(3);

photos.forEach(photo => {

expect(photo).to.have.property('id');

expect(photo).to.have.property('title');

expect(photo).to.have.property('url');

});

done();

});

});

});

## **Advanced Node.js Interview Questions**

### **30. What is an Event Emitter in Node.js?**

EventEmitter is a Node.js class that includes all the objects that are basically capable of emitting events. This can be done by attaching named events that are emitted by the object using an eventEmitter.on() function. Thus whenever this object throws an even the attached functions are invoked synchronously.

**const** EventEmitter = require('events');**class** **MyEmitter** **extends** **EventEmitter** {}**const** myEmitter = **new** MyEmitter();

myEmitter.on('event', () => {

console.log('an event occurred!');

});

myEmitter.emit('event');

### **31. Enhancing Node.js performance through clustering.**

Node.js applications run on a single processor, which means that by default they don’t take advantage of a multiple-core system. Cluster mode is used to start up multiple node.js processes thereby having multiple instances of the event loop. When we start using cluster in a nodejs app behind the scene multiple node.js processes are created but there is also a parent process called the ****cluster manager**** which is responsible for monitoring the health of the individual instances of our application.

Clustering in Node.js

### **32. What is a thread pool and which library handles it in Node.js**

The Thread pool is handled by the libuv library. libuv is a multi-platform C library that provides support for asynchronous I/O-based operations such as file systems, networking, and concurrency.

### **33. What is WASI and why is it being introduced?**

Web assembly provides an implementation of [WebAssembly System Interface](https://wasi.dev/" \t "https://www.interviewbit.com/node-js-interview-questions/_blank) specification through WASI API in node.js implemented using WASI class. The introduction of WASI was done by keeping in mind its possible to use the underlying operating system via a collection of POSIX-like functions thus further enabling the application to use resources more efficiently and features that require system-level access.

### **34. How are worker threads different from clusters?**

****Cluster:****

* There is one process on each CPU with an IPC to communicate.
* In case we want to have multiple servers accepting HTTP requests via a single port, clusters can be helpful.
* The processes are spawned in each CPU thus will have separate memory and node instance which further will lead to memory issues.

****Worker threads:****

* There is only one process in total with multiple threads.
* Each thread has one Node instance (one event loop, one JS engine) with most of the APIs accessible.
* Shares memory with other threads (e.g. SharedArrayBuffer)
* This can be used for CPU-intensive tasks like processing data or accessing the file system since NodeJS is single-threaded, synchronous tasks can be made more efficient leveraging the worker's threads.

### **35. How to measure the duration of async operations?**

Performance API provides us with tools to figure out the necessary performance metrics. A simple example would be using async\_hooks and perf\_hooks

'use strict';

const async\_hooks = require('async\_hooks');

const {

performance,

PerformanceObserver

} = require('perf\_hooks');

const set = new Set();

const hook = async\_hooks.createHook({

init(id, type) {

if (type === 'Timeout') {

performance.mark(`Timeout-${id}-Init`);

set.add(id);

}

},

destroy(id) {

if (set.has(id)) {

set.delete(id);

performance.mark(`Timeout-${id}-Destroy`);

performance.measure(`Timeout-${id}`,

`Timeout-${id}-Init`,

`Timeout-${id}-Destroy`);

}

}

});

hook.enable();

const obs = new PerformanceObserver((list, observer) => {

console.log(list.getEntries()[0]);

performance.clearMarks();

observer.disconnect();

});

obs.observe({ entryTypes: ['measure'], buffered: true });

setTimeout(() => {}, 1000);

This would give us the exact time it took to execute the callback.

### **36. How to measure the performance of async operations?**

Performance API provides us with tools to figure out the necessary performance metrics.   
A simple example would be:

const { PerformanceObserver, performance } = require('perf\_hooks');

const obs = new PerformanceObserver((items) => {

console.log(items.getEntries()[0].duration);

performance.clearMarks();

});

obs.observe({ entryTypes: ['measure'] });

performance.measure('Start to Now');

performance.mark('A');

doSomeLongRunningProcess(() => {

performance.measure('A to Now', 'A');

performance.mark('B');

performance.measure('A to B', 'A', 'B');});