Node Js Interview Questions And Answers

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

Node.js is server-side scripting based on Google's V8 JavaScript engine. It is used to build scalable programs especially web applications that are computationally simple but are frequently accessed.
You can use Node.js in developing 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- Which types of application developed using Node js?

You can develop Real-time web applications, Network applications, Distributed Systems, and General-purpose applications using the Node js.

3- What is the difference between Node.js vs Ajax?

The difference between Node.js and Ajax is that Ajax (short for Asynchronous Javascript and XML) is a client-side technology, often used for updating the contents of the page without refreshing it. While,Node.js is Server Side Javascript, used for developing server software. Node.js does not execute in the browser but by the server.

4- What are the functionalities of NPM in Node.js?

NPM (Node Package Manager) provides two functionalities:
An online repository for Node.js packages
Command-line utility for installing packages, version management and dependency management of Node.js packages

Gaurav Pandey

Swipe>>

5- In which Language Node Js is written?

Node js is written in C, C++,JavaScript.It uses Google's open source V8 Javascript Engine to convert Javascript code to C++. (node js interview questions and answers pdf)

6- Why should you use Node.js?

Scalable network programs can be developed easily by Node.js and if you like to know why we should use Node, js then due to below-listed advantages it is usually used by the organizations:
- Great concurrency is yielded by Node.js
- Every feature of Node.js is asynchronous
- It is never blocked
- It is quite faster
- A unified programming language and data type is offered by this

7- Explain various stream of Node.js?

In Node.js stream allow users to read data from source and o write data to a destination in a continuous process.
They are just an object and following four types of streams are there in **Node.js they are:**
To provide read operation
To provide a write operation
To provide both read and write operation
A duplex stream form that can perform computations as per available data.

Gaurav Pandey

Swipe>>

8- What is npm? What is the main functionality of npm?

npm stands for Node Package Manager. Following are the two main functionalities of npm:
Online repositories for node.js packages/modules which are searchable on search.nodejs.org
Command-line utility to install packages, do version management and dependency management of Node.js packages.

9- Which Is The First Argument Usually Passed To A Node.Js Callback Handler?

Node.js core modules follow a standard signature for its callback handlers and usually the first argument is an optional error object. And if there is no error, then the argument defaults to null or undefined.

10- What is the difference between AngularJS and Node.js?

Angular.JS is a web application development framework while Node.js is a runtime system.

11- Explain callback in Node.js?

A callback function is called at the completion of a given task. This 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.

12- What is error-first callback in Node JS?

In order to check for the proper working of the code, we need to verify error-free execution. In this regard, error-first callbacks are used, that will send error first, followed by related data to the error.

13- What are some of the most popular packages of Node.js?

- **Async**: Async is a utility module which provides straight-forward, powerful functions for working with asynchronous JavaScript.
- Browserify: Browserify will recursively analyze all the require() calls in your app in order to build a bundle you can serve up to the browser in a single <script> tag.
- **Bower**: Bower is a package manager for the web. It works by fetching and installing packages from all over, taking care of hunting, finding, downloading, and saving the stuff you're looking for.
- **Csv**: csv module has four sub modules which provides CSV generation, parsing, transformation and serialization for Node.js.
- **Debug**: Debug is a tiny node.js debugging utility modelled after node core's debugging technique.
- **Express**: Express is a fast, un-opinionated, minimalist web framework. It provides small, robust tooling for HTTP servers, making it a great solution for single page applications, web sites, hybrids, or public HTTP APIs.
- **Grunt**: is a JavaScript Task Runner that facilitates creating new projects and makes performing repetitive but necessary tasks such as linting, unit testing, concatenating and minifying files (among other things) trivial.
- **Gulp**: is a streaming build system that helps you automate painful or time-consuming tasks in your development workflow.
- **Hapi**: is a streaming build system that helps you automate painful or time-consuming tasks in your development workflow.

- **Http-server**: is a simple, zero-configuration command-line http server. It is powerful enough for production usage, but it's simple and hackable enough to be used for testing, local development, and learning.
- **Inquirer**: A collection of common interactive command line user interfaces.
- **Jquery**: jQuery is a fast, small, and feature-rich JavaScript library.
- **Jshint**: Static analysis tool to detect errors and potential problems in JavaScript code and to enforce your team's coding conventions.
- **Koa**: Koa is web app framework. It is an expressive HTTP middleware for node.js to make web applications and APIs more enjoyable to write.
- **Lodash**: The lodash library exported as a node module. Lodash is a modern JavaScript utility library delivering modularity, performance, & extras.
- **Less**: The less library exported as a node module.
- **Moment**: A lightweight JavaScript date library for parsing, validating, manipulating, and formatting dates.
- **Mongoose**: It is a MongoDB object modeling tool designed to work in an asynchronous environment.
- **MongoDB**: The official MongoDB driver for Node.js. It provides a high-level API on top of mongodb-core that is meant for end users.
- **Npm**: is package manager for javascript.
- **Nodemon**: It is a simple monitor script for use during development of a node.js app, It will watch the files in the directory in which nodemon was started, and if any files change, nodemon will automatically restart your node application.
- **Nodemailer**: This module enables e-mail sending from a Node.js applications.
- **Optimist**: is a node.js library for option parsing with an argv hash.
- **Phantomjs**: An NPM installer for PhantomJS, headless webkit with JS API. It has fast and native support for various web standards: DOM handling, CSS selector, JSON, Canvas, and SVG.
- **Passport**: A simple, unobtrusive authentication middleware for Node.js. Passport uses the strategies to authenticate requests. Strategies can range from

verifying username and password credentials or authentication using OAuth or OpenID.

- **Q**: Q is a library for promises. A promise is an object that represents the return value or the thrown exception that the function may eventually provide.
- **Request**: Request is Simplified HTTP request client make it possible to make http calls. It supports HTTPS and follows redirects by default.
- **Socket.io**: Its a node.js realtime framework server.
- **Sails**: Sails : API-driven framework for building realtime apps, using MVC conventions (based on Express and Socket.io)
- **Through**: It enables simplified stream construction. It is easy way to create a stream that is both readable and writable.
- **Underscore**: Underscore.js is a utility-belt library for JavaScript that provides support for the usual functional suspects (each, map, reduce, filter...) without extending any core JavaScript objects.
- **Validator**: A nodejs module for a library of string validators and sanitizers.
- **Winston**: A multi-transport async logging library for Node.js
- **Ws**: A simple to use, blazing fast and thoroughly tested websocket client, server and console for node.js
- **Xml2js**: A Simple XML to JavaScript object converter.
- **Yo:** A CLI tool for running Yeoman generators
- **Zmq**: Bindings for node.js and io.js to ZeroMQ .It is a high-performance asynchronous messaging library, aimed at use in distributed or concurrent applications.
-

14- How does Node.js support multi-processor platforms, and does it fully utilize all processor resources?

Since Node.js is by default a single thread application, it will run on a single processor core and will not take full advantage of multiple core resources. However, Node.js provides support for deployment on multiple-core systems, to

take greater advantage of the hardware. The Cluster module is one of the core Node.js modules and it allows running multiple Node.js worker processes that will share the same port.

The cluster module helps to spawn new processes on the operating system. Each process works independently,

so you cannot use shared state between child processes. Each process communicates with the main process by IPC and pass server handles back and forth.

Cluster supports two types of load distribution:

- The main process listens on a port, accepts new connection and assigns it to a child process in a round robin fashion.
- The main process assigns the port to a child process and child process itself listen the port.

15- What is typically the first argument passed to a Node.js callback handler?

The first argument to any callback handler is an optional error object

```
function callback(err, results) {
    // usually we'll check for the error before handling
results
    if(err) {
        // handle error somehow and return
    }
    // no error, perform standard callback handling
}
```

16- What is JIT and how is it related to Node.js?

Node.js has depended on the V8 JavaScript engine to provide code execution in the language. The V8 is a JavaScript engine built at the google development center, in Germany. It is open source and written in C++. It is used for both client side (Google Chrome) and server side (node.js) JavaScript applications. A central piece of the V8 engine that allows it to execute JavaScript at high speed is the JIT (Just In Time) compiler. This is a dynamic compiler that can optimize code during runtime. When V8 was first built the JIT Compiler was dubbed FullCodegen. Then, the V8 team implemented Crankshaft, which included many performance optimizations that FullCodegen did not implement.

The V8 was first designed to increase the performance of the JavaScript execution inside web browsers. In order to obtain speed, V8 translates JavaScript code into more efficient machine code instead of using an interpreter. It compiles JavaScript code into machine code at execution by implementing a JIT (Just-In-Time) compiler like a lot of modern JavaScript engines such as SpiderMonkey or Rhino (Mozilla) are doing. The main difference with V8 is that it doesn’t produce bytecode or any intermediate code.

17- What is chrome v8 engine?

V8 is the name of the JavaScript engine that powers Google Chrome. It's the thing that takes our JavaScript and executes it while browsing with Chrome. V8 provides the runtime environment in which JavaScript executes. The DOM, and the other Web Platform APIs are provided by the browser.

V8 is Google’s open source high-performance JavaScript and WebAssembly engine, written in C++. It is used in Chrome and in Node.js, among others. It implements ECMAScript and WebAssembly, and runs on Windows 7 or later, macOS 10.12+, and Linux systems that use x64, IA-32, ARM, or MIPS

processors. V8 can run standalone, or can be embedded into any C++ application.

18. What is difference between put and patch?

PUT and PATCH are HTTP verbs and they both relate to updating a resource. The main difference between PUT and PATCH requests are in the way the server processes the enclosed entity to modify the resource identified by the Request-URI.

In a PUT request, the enclosed entity is considered to be a modified version of the resource stored on the origin server, and the client is requesting that the stored version be replaced.

With PATCH, however, the enclosed entity contains a set of instructions describing how a resource currently residing on the origin server should be modified to produce a new version.

Also, another difference is that when you want to update a resource with PUT request, you have to send the full payload as the request whereas with PATCH, you only send the parameters which you want to update.

The most commonly used HTTP verbs POST, GET, PUT, DELETE are similar to CRUD (Create, Read, Update and Delete) operations in database. We specify these HTTP verbs in the capital case. So, the below is the comparison between them.

- POST - create

- GET - read
- PUT - update
- DELETE - delete

PATCH: Submits a partial modification to a resource. If you only need to update one field for the resource, you may want to use the PATCH method.

19- List types of Http requests supported by Node.js?

The HTTP core module is a key module to Node.js networking.

```
const http = require('http')

http.METHODS

require('http').METHODS
[ 'ACL',
  'BIND',
  'CHECKOUT',
  'CONNECT',
  'COPY',
  'DELETE',
  'GET',
  'HEAD',
  'LINK',
  'LOCK',
  'M-SEARCH',
  'MERGE',
  'MKACTIVITY',
  'MKCALENDAR',
  'MKCOL',
```

'MOVE',
'NOTIFY',
'OPTIONS',
'PATCH',
'POST',
'PROPFIND',
'PROPPATCH',
'PURGE',
'PUT',
'REBIND',
'REPORT',
'SEARCH',
'SUBSCRIBE',
'TRACE',
'UNBIND',
'UNLINK',
'UNLOCK',
'UNSUBSCRIBE']

http.STATUS_CODES

require('http').STATUS_CODES
{ '100': 'Continue',
'101': 'Switching Protocols',
'102': 'Processing',
'200': 'OK',
'201': 'Created',
'202': 'Accepted',
'203': 'Non-Authoritative Information',
'204': 'No Content',
'205': 'Reset Content',
'206': 'Partial Content',

```
'207': 'Multi-Status',
'208': 'Already Reported',
'226': 'IM Used',
'300': 'Multiple Choices',
'301': 'Moved Permanently',
'302': 'Found',
'303': 'See Other',
'304': 'Not Modified',
'305': 'Use Proxy',
'307': 'Temporary Redirect',
'308': 'Permanent Redirect',
'400': 'Bad Request',
'401': 'Unauthorized',
'402': 'Payment Required',
'403': 'Forbidden',
'404': 'Not Found',
'405': 'Method Not Allowed',
'406': 'Not Acceptable',
'407': 'Proxy Authentication Required',
'408': 'Request Timeout',
'409': 'Conflict',
'410': 'Gone',
'411': 'Length Required',
'412': 'Precondition Failed',
'413': 'Payload Too Large',
'414': 'URI Too Long',
'415': 'Unsupported Media Type',
'416': 'Range Not Satisfiable',
'417': 'Expectation Failed',
'418': 'I\'m a teapot',
'421': 'Misdirected Request',
```

'422': 'Unprocessable Entity',
'423': 'Locked',
'424': 'Failed Dependency',
'425': 'Unordered Collection',
'426': 'Upgrade Required',
'428': 'Precondition Required',
'429': 'Too Many Requests',
'431': 'Request Header Fields Too Large',
'451': 'Unavailable For Legal Reasons',
'500': 'Internal Server Error',
'501': 'Not Implemented',
'502': 'Bad Gateway',
'503': 'Service Unavailable',
'504': 'Gateway Timeout',
'505': 'HTTP Version Not Supported',
'506': 'Variant Also Negotiates',
'507': 'Insufficient Storage',
'508': 'Loop Detected',
'509': 'Bandwidth Limit Exceeded',
'510': 'Not Extended',
'511': 'Network Authentication Required' }

20- What are the key features of Node.js?

- Asynchronous event driven IO helps concurrent request handling – All APIs of Node.js are asynchronous. This feature means that if a Node receives a request for some Input/Output operation, it will execute that operation in the background and continue with the processing of other requests. Thus it will not wait for the response from the previous requests.
- Fast in Code execution – Node.js uses the V8 JavaScript Runtime engine, the one which is used by Google Chrome. Node has a wrapper over the

JavaScript engine which makes the runtime engine much faster and hence processing of requests within Node.js also become faster.

- Single Threaded but Highly Scalable – Node.js uses a single thread model for event looping. The response from these events may or may not reach the server immediately. However, this does not block other operations. Thus making Node.js highly scalable. Traditional servers create limited threads to handle requests while Node.js creates a single thread that provides service to much larger numbers of such requests.
- Node.js library uses JavaScript – This is another important aspect of Node.js from the developer's point of view. The majority of developers are already well-versed in JavaScript. Hence, development in Node.js becomes easier for a developer who knows JavaScript.
- There is an Active and vibrant community for the Node.js framework – The active community always keeps the framework updated with the latest trends in the web development.
- No Buffering – Node.js applications never buffer any data. They simply output the data in chunks.

21- What is NPM? What is the need of NPM in Node.js?

NPM stands for Node.js Package Management. It comes with Node.js platform and allows us to install various packages for Node.js. This package manages and supports various commands to install and remove the modules. Here one important note is we require either package.json file or the node_modules folder to install modules locally.

One of the best things about npm is that it locally stores all dependencies. For example, if module X uses module A version 1.0, and module Y uses module A version 1.5, then both X and Y will have their own local copies of module A.

The need for npm package

While I was working on a real node.js application I encountered many instances where I needed to write a few libraries. Recently, I thought these libraries could be useful for others. Afterward, I discovered the npm package.

22- Explain Karma And Jasmine in Node.js?

Karma is a tool made on top of NodeJS to run JavaScript test cases. This is not a testing framework like Jasmine or Mocha or Chai etc. It only allows us to run JavaScript test cases written using testing frameworks like Jasmine.

Karma runs JavaScript test cases against real browsers through Command Line Interface (CLI) rather than a virtual browser and DOM. Since DOM rendering across browsers varies, for the correctness of the test report it uses real browsers. It can configure what browsers need to be used while running Karma.

23- List some of the big advantages of using Node.js?

Ability to build scalable programs
Increased concurrency
Asynchronous capabilities

24- What is the global installation of dependencies?

Globally installed packages/dependencies are stored in /npm directory. Such dependencies can be used in CLI (Command Line Interface) function of any node.js but can not be imported using require() in Node application directly. To install a Node project globally use -g flag.

Reference:
https://github.com/learning-zone/nodejs-interview-questions
https://svrtechnologies.com/

Gaurav Pandey

Swipe>>