**Node INTERVIEW Q&A**

**1.What is NPM?**

1. NPM stands for the node package manager, npm is used for node dependency management.
2. Most of the time, we use npm as a server-side node dependency tool
3. NPM gets installed with NodeJs installation. NPM uses nested dependencies, so we can use different versions of any module in our code.
4. Nested dependencies mean that any dependency is again dependent on another dependency npm: It is a package manager for the JavaScript programming language.
5. It is the default package manager for the JavaScript runtime environment Node.js. Help to manage a project’s dependencies. npm is installed with Node automatically.
6. npm: NPM generates a ‘package-lock.json’ file. the package-lock will generate the same node\_modules folder for different npm versions

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**2.What are the modules in Node.js?**

* In Node.js, Modules are the blocks of encapsulated code that communicate with an external application on the basis of their related functionality.
* Modules can be a single file or a collection of multiples files/folders.
* The reason programmers are heavily reliant on modules is because of their re-usability as well as the ability to break down a complex piece of code into manageable chunks. Modules are of three types:

1. Core Modules/Internal modules
2. Local Modules/Custom modules
3. Third-party Modules

Core Modules**:** Node.js has many built-in modules that are part of the platform and comes with Node.js installation. These modules can be loaded into the program by using the require function. The require() function will return a JavaScript type depending on what the particular module returns. ex:- http creates an HTTP server in Node.js.

Local Modules:Unlike built-in and external modules, local modules are created locally in your Node.js application. Let’s create a simple calculating module that calculates various operations. Another file can use its exported functionality using the require() function.

Third-party modules**:** Third-party modules are modules that are available online using the Node Package Manager(NPM). These modules can be installed in the project folder or globally. Some of the popular third-party modules are mongoose, express, angular, and react.

Example**:** npm install express

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**3.What is the purpose of the module.exports?**

* Module.exports is the instruction that tells Node. js which bits of code (functions, objects, strings, etc.) to “export” from a given file so other files are allowed to access the exported code.
* The module.exports is actually a property of the module object in node.js. module. Exports is the object that is returned to the require() call.
* By module.exports, we can export functions, objects, and their references from one file and can use them in other files by importing them by require() method. easy to maintain and manage the code base in different modules.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**4.Difference between default export and named export.**

Export objects, functions, variables from the module so they can be used by other programs with the help of the import statements.

There are two types of exports.

1. Named Exports,

2. Default Exports.

Named Exports: Named exports are useful to export several values.

During the import, it is mandatory to use the same name of the corresponding object.

Named exports allow us to share multiple objects, functions or variables from a single file and were introduced with the release of ES2015.

Named exports are imported with curly braces in various files and must be imported using the name of the object, function or variable that was exported.

//Exporting individual features:

export var name1 = …, name2 = …, …, nameN; // also let, const

// Export list :

export { name1, name2, …, nameN };

//Exporting everything at once :

export { object, number, x, y, boolean, string }

// Renaming exports:

export { variable1 as name1, variable2 as name2, …, nameN };

// export features declared earlier:

export { myFunction, myVariable };

Default Exports: Default exports are useful to export only a single object, function, variable. During the import, we can use any name to import.Exports without a default tag are Named exports. Exports with the default tag are Default exports.

default export:- Using Named and Default Exports at the same time: It is possible to use Named and Default exports in the same file. It means both will be imported in the same file. Example: javascript.

function defau(req,res){

res.send("default exports")

}

function named(req,res) {

res.send('named exports')

}

module.exports = default;//default export

module.exports = {named};//named export

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**5. How do you import any module in Node.js?**

Importing functions or modules enhances the reusability of code.

Importing a Module: We need to import the module to use the functions defined in the imported module in another file. The result returned by require() is stored in a variable which is used to invoke the functions using the dot notation.

const f = require('./func')

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**6. What are the different types of HTTP requests?**

HTTP (Hypertext Transfer Protocol) specifies a collection of request methods to specify what action is to be performed on a particular resource. The most commonly used HTTP request methods are GET, POST, PUT, PATCH, and DELETE. These are equivalent to the CRUD operations (create, read, update, and delete).

GET: GET request is used to read/retrieve data from a web server. GET returns an HTTP status code of 200 (OK) if the data is successfully retrieved from the server.

POST: POST request is used to send data (file, form data, etc.) to the server. On successful creation, it returns an HTTP status code of 201.

PUT: A PUT request is used to modify the data on the server. It replaces the entire content at a particular location with data that is passed in the body payload. If there are no resources that match the request, it will generate one.

PATCH: PATCH is similar to PUT request, but the only difference is, it modifies a part of the data. It will only replace the content that you want to update.

DELETE: A DELETE request is used to delete the data on the server at a specified location.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**7.Explain the concept of middleware in Node.js.**

Express.js is a routing and Middleware framework for handling the different routing of the webpage and it works between the request and response cycle.

Middleware gets executed after the server receives the request and before the controller actions send the response.

Middleware has the access to the request object, responses object, and next, it can process the request before the server send a response. An Express-based application is a series of middleware function calls.

Middleware Chaining: Middleware can be chained from one to another, Hence creating a chain of functions that are executed in order. The last function sends the response back to the browser. So, before sending the response back to the browser the different middleware processes the request.

The next() function in the express is responsible for calling the next middleware function if there is one.

app.get(path, (req, res, next) => {}, (req, res) => {})

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**8.Explain CORS.**

CORS :(Cross-Origin Resource Sharing) is a mechanism by which data or any other resource of a site could be shared intentionally to a third party website when there is a need. Generally, access to resources that are residing in a third party site is restricted by the browser clients for security purposes.

// cors - npm i cors

const cors = require("cors")

app.use(cors())

app.get('/default',defau,(req,res)=>{

console.log("default")

})

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**9. What is Express? How does it help you to create a backend application?**

Express.js: Express is a small framework that sits on top of Node.js’s web server functionality to simplify its APIs and add helpful new features. It makes it easier to organize your application’s functionality with middleware and routing. It adds helpful utilities to Node.js’s HTTP objects. It facilitates the rendering of dynamic HTTP objects.

\*More features than Node.js

\*It is used to build web-apps using approaches and principles of Node.js.

\*Routing is provided.

\*It requires less coding time.

\*Uses middleware for the arrangement of functions systematically server-side.

\*import by const express = require ("express")

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**10. Explain min 5 status codes gets used in Backend development.**

An HTTP status code is a message a website's server sends to the browser to indicate whether or not that request can be fulfilled. Status codes specs are set by the W3C. Status codes are embedded in the HTTP header of a page to tell the browser the result of its request.

\*102 PROCESSING Processing 200 OK OK

\*400 BAD\_REQUEST Bad Request

\*401 UNAUTHORIZED Unauthorized

\*402 PAYMENT\_REQUIRED Payment Required

\*403 FORBIDDEN Forbidden

\*404 NOT\_FOUND Not Found

\*500 INTERNAL\_SERVER\_ERROR Internal Server Error

\*503 - Service Unavailable

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

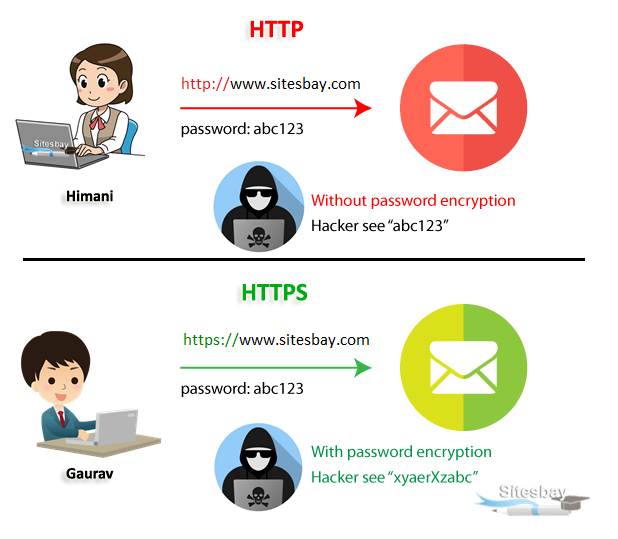
**11. What is node.js ?**

Node.js:

* Node.js is an open source and cross-platform runtime environment for executing JavaScript code outside of a browser.
* You need to remember that NodeJS is not a framework and it’s not a programming language. Most of the people are confused and understand it’s a framework or a programming language.
* We often use Node.js for building back-end services like APIs like Web App or Mobile App. It’s used in production by large companies such as Paypal, Uber, Netflix, Walmart and so on.
* Node.js is a platform for building the i/o applications which are server-side event-driven and made using JavaScript. It is used to build server-side, input-output, event-driven apps. It is built on Google’s V8 engine.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**12. Difference between HTTP and HTTPS.**

****

HTTPS is HTTP with encryption. The only difference between the two protocols is that HTTPS uses TLS (SSL) to encrypt normal HTTP requests and responses. As a result, HTTPS is far more secure than HTTP. A website that uses HTTP has http:// in its URL, while a website that uses HTTPS has https://. Some difference of http and https;

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**13. What are JWT tokens?**

JSON web token (JWT), pronounced "jot", is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. Again, JWT is a standard, meaning that all JWTs are tokens, but not all tokens are JWTs. when we logged-in somewhere , they provide us a unique token.which helps us to access through it token is provided only when the registered data and login data are matching. Ex:-facebook

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Implementation**

**14. How do you create a simple Express.js application?**

const express = require('express')

const server=express();

server.get(“/” , (req, res) =>{

res.send(“Hello!”);

})

server.listen(3000, () =>{

console.log(“server is running at port:3000”);

})

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**15. Implementation of all types of exports in backend applications.**

function defau(req,res){

res.send("default exports")

}

function named(req,res) {

res.send('named exports')

}

module.exports = default;//default export

module.exports = {named};//named export

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**16. Create a middleware which will be applicable to all the routes.**

const Checking = require("./Routes/Checking")//importing

const app = express();

// middleware

const middleware1 = (req,res,next) =>{

console.log('middleware1')

next()

}

const middleware2 = (req,res,next) =>{

console.log('middleware2')

next()

}

app.use(middleware1)

app.get('/page1',(req,res)=>{

res.send('page1 - both middleware 1 and 2 applied')

})//here we will get both middleware 1 and 2

app.get('/page2',middleware2, (req,res)=> {

res.send('page2-middleware 2 only applied')

})

app.get('/page3',middleware2, (req,res)=> {

res.send('page3 - - both middleware 1 and 2 applied')

})

app.use(express.json())

app.use("/auth",Checking)// linking child with parent

app.listen(9090,()=>{

console.log('running')

})

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**17. Implement CORS.**

// cors - npm i cors

const express = require("express");

const cors = require("cors")

const app = express();

app.use(cors())

app.get("/addUser",(req,res)=>{

console.log(req.query);

res.send("success");

})

app.put("/editUser/:Salary",(req, res) =>{

console.log(req.params);

res.send("success");

})

app.listen(8001);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**18. Implement API for signup and login. Implement an api that accepts a token , checks the token . If correct sends data else error "token is expired”.**

Db.js:

const users = [{

email:"viswa@gmail.com",

password:"viswananna"

}]

module.exports = {users}

Checking.js:

const Route = require('express').Router()

const bcrypt = require('bcrypt')

const { Router } = require('express')

const JWT = require('jsonwebtoken')

const {check,validationResult} = require('express-validator') // to validate the email or password like that && it was always array format.

const {users}= require("./db")

// Route.get("/",(req,res)=>{

// res.send("checking page is working")

// })

Route.post("/signup",[

check("email","please provide a valid email")

.isEmail(), //express validator

check("password","please provide password greater than 6 character")

.isLength({

min:6 // express validator

})

],

async(req,res)=>{

const {email,password}=req.body

console.log(email,password);

const err = validationResult(req); //if there is any in req, it will sort it

if(!err.isEmpty()){

return res.status(400).json({

err:err.array()

})

//checking our postman email and password with the database(db)

}

let user = users.find((user)=>{

return user.email === email

})

if(user){

return res.status(400).json({

err:[{

"msg":"this user is already existed"

}]

})

}

let hashpassword = await bcrypt.hash(password,10)

users.push({email,

password:hashpassword})

const token = await JWT.sign({

email

},"asdfghjklttyy",{

expiresIn:36000

})

res.json({token})

console.log(hashpassword)

res.send("checking page is working")

})

Route.get("/all",(req,res)=>{

res.json(users);})

module.exports = Route

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*END\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*