Express.js is a free and open source software and web application framework for Node JS which was designed for the development of web applications and API’s. It is also the standard server framework for Node JS. It was developed by TJ Holowaychuk on 16th November 2010. Along with the MongoDB database and Angular JS frontend framework, express JS is the backend part of the MEAN stack. With the help of this, you will be able to manage everything from routes to handling requests and views. It even provides a robust set of features for various web and mobile applications.

**Q1.List some features of Express JS.**

Some of the main features of Express JS are listed below: –

It is used for setting up middlewares so as to provide a response to the HTTP or RESTful requests.

With the help of express JS, the routing table can be defined for performing various HTTP operations.

It is also used for dynamically rendering HTML pages which are based on passing arguments to the templates.

It provides each and every feature which is provided by core Node JS.

The performance of Express JS is adequate due to the presence of a thin layer prepared by the Express JS.

It is used for organizing the web applications into the MVC architecture.

Everything from routes to rendering view and performing HTTP requests can be managed by Express JS.

**Q2**.Write the steps for setting up an Express JS application.

Following are the steps used to set up an express JS application: –

A folder with the same name as the project name is created.

A file named package.json is created inside the folder created.

“npm install” command is run on the command prompt. It installs all the libraries present in package.json.

A file named server.js is created.

“Router” file is created inside the package which consists of a folder named index.js.

“App” is created inside the package which has the index.html file.

This way, an express JS application is set up.

**Q3.**What do you mean by Express JS?

Express JS is an application framework which is light-weighted node JS. A number of flexible, useful and important features are provided by this JavaScript framework for the development of mobile as well as web applications with the help of node JS.

==>

Express JS is a framework which helps to develop web and mobile applications. Its works on nodejs plateform. Its sub part of node.js.

**Q4**.Name the type of web applications which can be built using Express JS.

Single-page, multi-page, and hybrid web applications can be built using Express JS.

**Q5**. What is the use of Express JS?

Express.js is a lightweight web application which helps in organizing the web application into MVC architecture on the server side.

**Q6**.What function are arguments available to Express JS route handlers?

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 which is used to pass control to one of the subsequent route handlers.

The third argument is optional and may be omitted, but in some cases, it is useful where there is a chain of handlers and control can be passed to one of the subsequent route handlers skipping the current one.

**Q7**.How to config properties in Express JS?

In Express JS, there are two ways for configuring the properties:

With process.ENV:

A file with the name “.env” is to be created inside the project folder.

All the properties are to be added in the “.env” file.

Any of the properties can be used in server.js.

With require JS:

A file with the name “config.json” is to be created in the config folder inside the project folder.

The config properties are to be added in the config.json file.

Now, require should be used to access the config.json file.

**Q8**.How can models be defined in Express JS?

There is no notion of any database in Express JS. So, the concept of models is left up to third-party node modules, allowing the users to interface with nearly any type of database.

**Q9**.Which template engine is supported by express JS?

Express JS supports any template engine that conforms to the (path, locals, callback) signature.

**Q10**.Why to use Express.js?

Below are the few reasons why to use Express with Node.js

Express js is built on top of Node.js. It is the perfect framework for ultra-fast Input / Output.

Cross Platform

Support MVC Design pattern

Support of NoSQL databases out of the box.

Multiple templating engine support i.e. Jade or EJS which reduces the amount of HTML code you have to write for a page.

Support Middleware, basic web-server creation, and easy routing tools.

**Q11**. How To Output Pretty Html In Express.js?

Answer : app.set('view options', { pretty: true });

**Q 12**. How To Remove Debugging From An Express App?

Answer :var io = require('socket.io').listen(app, { log: false });

io.set('log level', 1);

**Q 13**. How To Do 404 Errors?

Answer : app.get('\*', function(req, res){

res.send('what???', 404);

});

**Q14**. How To Download A File?

Answer :

app.get('/download', function(req, res){

var file = \_\_dirname + '/download-folder/file.txt';

res.download(file);

});

**Q 15**. What Is The Parameter “next” Used For In Express?

Answer :app.get('/userdetails/:id?', function(req, res, next){

});

req and res which represent the request and response objects

next passes control to the next matching route.

**Q16**. What Function Arguments Are Available To Express.js Route Handlers?

Answer :The arguments available to an Express.js route handler function are:

req - the request object

res - the response object

next (optional) - a function to pass control to one of the subsequent route handlers

The third argument may be omitted, but is useful in cases where you have a chain of handlers and you would like to pass control to one of the subsequent route handlers, and skip the current one.

**Q 177**. How To Allow Cors In Expressjs? Explain With An Example?

Answer : In order to allow CORS in Express.js, add the following code in server.js:

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();

});

**Q18**. How To Redirect 404 Errors To A Page In Expressjs?

Answer :In server.js add the following code to redirect 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 19**. How To Enable Debugging In Express App?

Answer : In different Operating Systems, we have following commands:

On Linux the command would be as follows:

$ DEBUG=express:\* node index.js

On Windows the command would be:

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

**Q.20)** What is Scaffolding in Express JS?

Scaffolding is creating the skeleton structure of application

There are 2 way to do this:

Express application generator

Yeoman

1. Express application generator:

Use express-generator to quickly create an application skeleton.

npm install express-generator -g

express myTestExpressApp

**Q.21)** Serving static files in Express?

#Example:

app.use(express.static('public'))

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

**Q.22)** What is routing and how routing works in Express.js?

Routing refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method (GET, POST, and so on).

Each route can have one or more handler functions, which are executed when the route is matched.

Route Syntax:

app.METHOD(PATH, HANDLER)

Where:

app is an instance of express.

METHOD is an HTTP request method, in lowercase.

PATH is a path on the server.

HANDLER is the function executed when the route is matched.

#Example:

app.get('/', function (req, res) {

res.send('Express Js Interview Questions')

})

**Q23**. Dynamic routing and how it 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 calls.

**Q.24)** What is middleware in Express Js?

"A function that is invoked by the Express routing layer before the final request handler."

Middleware functions can perform the following tasks:

Execute any code.

Make changes to the request and the response objects.

End the request-response cycle.

Call the next middleware function in the stack.

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.

An Express application can use the following types of middleware:

Application-level middleware

Router-level middleware

Error-handling middleware

Built-in middleware

Third-party middleware

1. Application-level middleware:

This kind of middleware method is bind to the app Object using app.use() method.

***//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 works in the same way as application-level middleware, except it is bound to an instance of express.Router()

3. Built-in middleware:

Starting with version 4.x, Express no longer depends on Connect.

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. NOTE: Available with Express 4.16.0+

express.urlencoded parses incoming requests with URL-encoded payloads. NOTE: Available with Express 4.16.0+

4. Third-party middleware:

There are a number of third party middleware, such as body-parser cookie-parser, mongoose and so on.

To handle HTTP POST request in Express.js version 4 and above, you need to install middleware module called body-parser, body-parser extract 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.

**const bodyParser = require('body-parser');**

**app.use(bodyParser.json());**

**app.use(bodyParser.urlencoded({ extended: false }))**

**Q.25)** Database integration in Express Js?

Adding the capability to connect databases to Express apps is just a matter of loading an appropriate Node.js driver for the database in your app

Express Js supports many RDBMS & NoSQL Ddatabase like

MongoDB

MySQL

Oracle

PostgreSQL

SQL Server

SQLite

#Example: Install MongoDB

>>npm install mongodb

**const 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.26)** Error handling in Express Js and How to redirect 404 errors to a page in Express Js?

**const express = require('express'),**

**app = express();**

**app.use(function (err, req, res, next) {**

**console.error(err.stack)**

**res.status(500).send('Something went wrong, Express Js Interview Questions')**

**})**

***//To redirect 404 errors***

**const express = require('express'),**

**app = express();**

**app.use(function(req, res, next) {**

**res.status(404).json({**

**errorCode: 404,**

**errorMsg: "route not found"**

**});**

**});**