Express.js Interview Questions and Answers

What is 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.

Express is a part of MEAN stack, a full stack JavaScript solution used in building fast, robust, and maintainable production web applications.

2. Why use Express.Js?

Express.js is a lightweight Node.js framework that gives us ability to create server-side web applications faster and smarter. The main reason for choosing Express is its simplicity, minimalism, flexibility, and scalability characteristics. It provides easy setup for middlewares and routing.

3. Write a 'Hello World' Express.js application?

To create a simple Express.Js application first we need to install Express in our NodeJs application.

npm install express

After that in the app.js file write the code

```
const express = require('express');
const app = express();
const PORT = 8000;

app.get('/', (req, res) => {
  res.send('Hello World!');
});

app.listen(PORT, () => {
  console.log(`Server is listening at port :${PORT}`);
});
```

4. Differentiate between Node.js and Express.js?

Node.js is the runtime environment that allows you to execute JavaScript on the server side, on the other hand Express.js is a framework built on top of Node.js that provides a set of tools for building web applications and APIs.

Express.js is not the only framework available for Node.js, but it is widely used because to its simplicity and flexibility.

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

Express.js is a JavaScript backend framework. It is mainly designed to develop complete web applications and APIs. Express is the backend

component of the MERN stack which stands for MongoDB, Express.js, React.js, Node.js.

6. Mentions few features of Express.js.

Few features of the Express.js includes

Routing: Express provides a simple way to define routes for handling HTTP requests. Routes are used to map different URLs to specific pieces of code, making it easy to organize your application's logic.

Middleware: Express uses middleware functions to perform tasks during the request-response cycle. Middleware functions have access to the request, response, and the next middleware function.

HTTP Utility Methods: Express mainly used for handling HTTP methods like GET, POST, PUT, and DELETE. This makes it easy to define how the application should respond to different types of HTTP requests.

Static File Serving: It can also serve static files, such as images, CSS, and JavaScript, with the help of built-in express.static middleware.

Security: It includes features and middleware to strengthen the security of your web applications, such as the helmet middleware to secure your app.

7. Explain the structure of an Express JS application?

The structure of an Express JS application can vary greatly depending on its complexity and the specific needs of the project. However, here is a basic approach that is commonly used:

Entry point: This is the starting point of the application where you set up your server, connect to your database, add middleware, and define the main routes.

Routes directory: This directory contains files for the app's routes.

Controllers directory: This directory contains files that define the logic to handle requests for a specific route.

Models directory: This directory is used for creating the schema models for the different data.

Middleware directory: This directory contains custom middleware functions that you can use in your routes.

Views directory: If you're using a templating engine, this directory contains your view templates.

Public directory: This directory contains static files that are served directly by the server such as images, CSS files, and JavaScript files.

This structure separates concerns in a logical way, making the application easier to understand and maintain.

8. What are some popular alternatives to Express JS?

There are several popular alternatives to Express.js which includes: Koa.js, Hapi.js, Sails.js, Fastify etc.

9. Which major tools can be integrated with Express JS?

There are many tools and libraries that can be integrated with Express.js such as:

Database tools: MongoDB, MySQL, PostgreSQL.

Template Engines: EJS, Pug, Mustache.

Authentication libraries: Passport.js.

Logging libraries: Morgan, Winston.

Validation libraries: Joi, express-validator.

ORM libraries: Sequelize, Mongoose.

10. What is .env file used for?

The .env file is used for storing sensitive information in a web application which we don't want to expose to others like password, database connection string etc. It is a simple text file where each line represents a key-value pair, and these pairs are used to configure various aspects of the application.

11. What are JWT?

Json Web Tokens are mainly a token which is used for authentication and information exchange. When a user signs in to an application, the application then assigns JWT to that user. Subsequent requests by the user will include the assigned JWT. This token tells the server what routes, services, and resources the user is allowed to access. Json Web Token includes 3 part namely- Header, Payload and Signature.

12. Create a simple middleware for validating user.

```
// Simple user validation middleware
const validateUser = (req, res, next) => {
  const user = req.user;

  // Check if the user object is present
  if (!user) {
```

```
    return res.status(401).json({ error: 'Unauthorized - User not found' });
  }

  // If the user is valid, move to the next middleware or route handler
  next();
};

// Example of using the middleware in an Express route
app.get('/profile', validateUser, (req, res) => {
  const user = req.user;
  res.json({ message: 'Profile page', username: user.username });
});
```

13. What is Bcrypt used for?

Bcrypt is a password hashing function which is used to securely hash and store user passwords. It is designed to be slow and computationally intensive, making it resistant to brute-force attacks and rainbow table attacks. Bcrypt is a key component in enhancing the security of user authentication systems.

14. Why should you separate the Express app and server?

In Express.js, it is recommended to separate the Express App and the server setup. This provides the modularity and flexibility and makes the codebase more easier to maintain and test. Here are some reasons why you should separate the Express app and server:

Modularity: You can define routes, middleware, and other components in the Express app independently of the server configuration.

Ease of Testing: Separation makes it easier to write unit tests for the Express app without starting an actual server. You can test routes, middleware, and other components in isolation.

Reusability: You can reuse the same Express app in different server configurations.

Configuration Management: Separating the app and server allows for cleaner configuration management.

Scalability: It provides a foundation for a scalable code structure. As your application grows, it will easier to maintain the code.

15. What do you understand about ESLint?

EsLint is a JavaScript linting tool which is used for automatically detecting incorrect patterns found in ECMAScript/JavaScript code. It is used with the purpose of improving code quality, making code more consistent, and avoiding bugs. ESLint is written using Node.js to provide a fast runtime environment and easy installation via npm.

16. Define the concept of the test pyramid.

The Test Pyramid is a concept in software testing that represents the distribution of different types of tests. It was introduced by Mike Cohn, and it suggests that a testing strategy should be shaped like a pyramid, with the majority of tests at the base and fewer tests as you move up. The Test Pyramid consists of three levels: Unit Tests, Integration Tests, and End-to-End (E2E) Tests.

17. Differentiate between res.send() and res.json().

Both res.send() and res.json() serves similar purposes with some difference. So it depends on the data type which we are working with. Choose res.json() when you are specifically working with JSON data. Use res.send() when you need versatility and control over the content type or when dealing with various data types in your responses.

18. What is meant by Scaffolding in Express JS?

Scaffolding in Express.js refers to the process of generating a basic project structure automatically. This can speed up the initial setup and help maintain consistency in the way projects are structured, especially in large teams.

19. How would you install an Express application generator for scaffolding?

Express application generator are used for quickly setting up a new Express application with some basic structure. You can install it using Node Package Manager (npm), which comes with Node.js.

To install it globally:

npm install -g express-generator

20. What is Yeoman and how to install Yeoman for scaffolding?

Yeoman is a scaffolding tool for web applications that helps developers to create new projects by providing a generator-based workflow.

To install Yeoman run the following command:

```
npm install -g yo
```

Yeoman works with generators, which are packages that define the structure and configuration of a project. You can install a generator like this:

```
npm install -g generator-express
```

Once installed, you can use Yeoman to create a new application:

```
yo appname
```

21. Explain what CORS is in Express JS?

CORS (Cross-Origin Resource Sharing) is a security feature implemented by web browsers to control how web pages in one domain can request and interact with resources hosted on another domain.

In the context of Express.js, CORS refers to a middleware that enables Cross-Origin Resource Sharing for your application. This allows the application to control which domains can access your resources by setting HTTP headers.

22. What are Built-in Middlewares?

Express.js, includes a set of built-in middlewares that provide common functionality. These built-in middlewares are included by default when you create an Express application and can be used to handle various tasks. Here are some of the built-in middlewares in Express:

express.json(): This middleware is used to parse incoming JSON requests. It automatically parses the request body if the Content-Type header is set to application/json.

express.Router(): The express.Router() function is often used to create modular route handlers. It allows you to group route handlers together and then use them as a middleware.

express.static(): This middleware is used to serve static files, such as images, CSS, and JavaScript files, from a specified directory.

23. How would you configure properties in Express JS?

In Express JS, you can configure properties using the app.set() method. This method allows you to set various properties and options which affects the behavior of the Express application.

app.set(name, value);

Here, name represents the name of the property you want to configure, and value is the value you want to assign to that property. Express provides a wide range of properties that you can configure based on your application's requirements.

24. Which template engines do Express support?

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

25. Elaborate on the various methods of debugging on both Linux and Windows systems?

The debugging is the vital need at the time of software development to identifying issues in the application's logic, handling of HTTP requests, middleware execution, and other aspects specific to web development. Here are some methods commonly used for debugging an Express.js application on both Linux and Windows:

Console.log: The simplest way to debug an Express JS application is by using console.log(). You can output messages to the console which can be viewed in the terminal.

Node Inspector: This is a powerful tool that allows you to debug your applications using Chrome Developer Tools. It supports features like setting breakpoints, stepping over functions, and inspecting variables.

Visual Studio Code Debugger: VS Code provides a built-in debugger that works on both Linux and Windows. It supports advanced features like conditional breakpoints, function breakpoints, and logpoints.

Utilizing debug module: The debug module is a small Node.js debugging utility that allows you to create debugging scopes.

26. Name some databases that integrate with Express JS?

Express.js can support a variety of the databases which includes:

MySQL

MongoDB

PostgreSQL

SQLite

Oracle

27. How would you render plain HTML using Express JS?

In Express.js, you can render plain HTML using the res.send() method or res.sendFile() method.

Sample code:

```
//using res.send

const express = require('express');
const app = express();
const port = 8000;

app.get('/', (req, res) => {
  const htmlContent = '<html><body><h1>Hello, World!</h1></body></html>';
  res.send(htmlContent);
});

app.listen(port, () => {
  console.log(`Server is listening on port ${port}`);
});
```

28. What is the use of 'Response.cookie()' function?

The response.cookie() function in Express.js is used to set cookies in the HTTP response. Cookies are small pieces of data sent from a server and stored on the client's browser. They are commonly used to store information about the user or to maintain session data.

Basic syntax of response.cookie():

res.cookie(name, value, [options]);

29. Under what circumstances does a Cross-Origin resource fail in Express JS?

When a Cross-Origin Resource Sharing request is made, the browser enforces certain security checks, and the request may fail under various circumstances:

No CORS Headers: The server doesn't include the necessary CORS headers in its response.

Mismatched Origin: The requesting origin does not match the origin specified in the Access-Control-Allow-Origin header.

Restricted HTTP Methods: The browser enforces restrictions on which HTTP methods are allowed in cross-origin requests.

No Credentials: The browser makes restrictions on requests that include credentials (such as cookies or HTTP authentication).

30. What is Pug template engine in Express JS?

Pug is a popular template engine for Express.js and other Node.js frameworks. You can use Pug to render dynamic HTML pages on the

server side. It allows you to write templates using a syntax that relies on indent.

31. What is meant by the sanitizing input process in Express JS?

Sanitizing input in Express.js application is an important security practice to prevent various types of attacks, such as Cross-Site Scripting (XSS) and SQL injection. It involves cleaning and validating user input before using it in your application so that it does not contain malicious code or can be a security risk.

32. How to generating a skeleton Express JS app using terminal command?

To generate a skeleton for an Express.js application using the terminal, you can use the Express application generator which is a command-line tool provided by the Express.js framework. This generator will setup a basic directory structure which includes necessary files, and installs essential dependencies.

Steps to generate:

Step 1: Open your terminal and install the Express application generator globally using the following command:

```
npm install -g express-generator
```

Step 2: After that you can use the express command to generate your Express.js app.

```
express my-express-app
```

Step 3:Now go to the app directory and install the dependencies and start the app by running-

npm install

npm start

33. What are middlewares in Express.Js?

Middleware functions are those functions that have the access to request and response object and the next middleware or function. They can add functionality to an application, such as logging, authentication, and error handling.

34. What are the types of middlewares?

There are mainly five types of Middleware in Express.js:

Application-level middleware

Router-level middleware

Error-handling middleware

Built-in middleware

Third-party middleware

35. List the built-in middleware functions provided by Express.

Express.js comes with several built-in middleware functions. Few of them are:

express.json: This is used for parsing incoming requests with JSON payloads.

express.static: This is used to serve static files like images, CSS files, and JavaScript files.

express.urlencoded: This is used for parsing incoming requests with URL-encoded payloads.

express.raw: This is used for parse incoming requests with a raw body.

express.text: This is used for parse incoming requests with a text body.

36. Mention some third-party middleware provided by Express JS.

Express.js allows you to use third-party middleware to extend and enhance the functionality of your web application. Here are some commonly used third-party middleware in Express.js:

body-parser: This middleware is used to parse incoming request bodies, allowing you to access form data or JSON payloads on req.body.

cors: This module provides middleware to enable Cross-Origin Resource Sharing (CORS) in your Express application.

morgan: Morgan is a middleware module that provides request logging functionality.

helmet: Helmet helps to secure Express apps by setting various HTTP headers.

express-session: This middleware is used for managing user sessions in your Express application.

passport: This middleware is used for implementing authentication and authorization in Express applications.

37. When application-level Middleware is used?

Application-level middlewares are bound to an instance of the Express application and are executed for every incoming request. These middlewares are defined using the app.use() method, and they can perform tasks such as logging, authentication, setting global variables, and more.

38. Explain Router-level Middleware.

Router-level middlewares are specific to a particular router instance. This type of middleware is bound to an instance of express.Router(). Router-level middleware works similarly to application-level middleware, but it's only invoked for the routes that are handled by that router instance. This allows you to apply middleware to specific subsets of your routes, keeping your application organized and manageable.

39. How to secure Express.Js application?

It is very important to secure your application to protect it against various security threats. We can follow few best practices in our Express.js app to enhance the security of our application.

Keep Dependencies Updated: Regularly update your project dependencies, including Express.js and other npm packages.

Use Helmet Middleware: The helmet middleware helps secure your application by setting various HTTP headers. It helps prevent common web vulnerabilities.

Set Secure HTTP Headers: Configure your application to include secure HTTP headers, such as Content Security Policy (CSP), Strict-Transport-Security (HSTS), and others.

Use HTTPS: Always use HTTPS to encrypt data in transit. Obtain an SSL certificate for your domain and configure your server to use HTTPS.

Secure Database Access: Use parameterized queries or prepared statements to prevent SQL injection attacks. Ensure that your database credentials are secure and not exposed in configuration files.

40. What is Express router() function?

The express.Router() function is used to create a new router object. This function is used when you want to create a new router object in your program to handle requests.

Syntax:

express.Router([options])

41. What are the different types of HTTP requests?

The primary HTTP methods are commonly referred to as CRUD operations, representing Create, Read, Update, and Delete. Here are the main HTTP methods:

GET: The GET method is used to request data from a specified resource.

POST: The POST method is used to submit data to be processed to a specified resource.

PUT: The PUT method is used to update a resource or create a new resource if it does not exist.

PATCH: The PATCH method is used to apply partial modifications to a resource.

DELETE: The DELETE method is used to request that a specified resource be removed.

42. Do Other MVC frameworks also support scaffolding?

The Scaffolding technique is supported by other MVC frameworks also which includes- 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.

43. Which are the arguments available to an Express JS route handler function?

In Express JS route handler function, there are mainly3 arguments available that provide useful information and functionality.

req: This represents the HTTP request object which holds information about the incoming request. It allows you to access and manipulate the request data.

res: This represents the HTTP response object which is used to send the response back to the client. It provides methods and properties to set response headers, status codes, and send the response body.

next: This is a callback function that is used to pass control to the next middleware function in the request-response cycle.

44. How can you deal with error handling in Express.js?

Express.js provides built-in error-handling mechanism with the help of the next() function. When an error occurs, you can pass it to the

next middleware or route handler using the next() function. You can also add an error-handling middleware to your application that will be executed whenever an error occurs.

45. What is the difference between a traditional server and an Express.js server?

A traditional server is a server that is built and managed independently. Traditional server may provide a basic foundation for handling HTTP requests and responses. While an Express.js server is built using the Express.js framework. It runs on top of Node.js. Express.js provides a simple and efficient way to create and manage web applications. It offers a wide range of features and tools for handling routing, middleware, and request or response objects.

46. What is the purpose of the next() function in Express.js?

The next() function is used to pass control from one middleware function to the next function. It is used to execute the next middleware function in the chain. If there are no next middleware function in the chain then it will give control to router or other functions in the app. If you don't call next() in a middleware function, the request-response cycle can be terminated, and subsequent middleware functions won't be executed.

47. What is the difference between app.route() and app.use() in Express.js?

app.route() is more specific to route handling and allows you to define a sequence of handlers for a particular route, on the other hand app.use() is a more general-purpose method for applying middleware globally or to specific routes.

48. Explain what dynamic routing is in Express.js.

Dynamic routing in Express.js include parameters, which allows you to create flexible and dynamic routes in your web application. This parameters are used in your route handlers to customize the behaviour based on the data provided.

In Express, dynamic routing is achieved by using route parameters, denoted by a colon (:) followed by the parameter name.

Here's a simple example:

```
const express = require('express');
const app = express();

// Dynamic route with a parameter
app.get('/users/:userId', (req, res) => {
  const userId = req.params.userId;
  res.send(`User ID: ${userId}`);
});

// Start the server
const port = 8000;
app.listen(port, () => {
  console.log(`Server is listening on port ${port}`);
```

```
});
```

49. How to serve static files in Express.Js?

In Express.js, you can serve static files using the built-in express.static middleware. This middleware function takes the root directory of your static files as an argument and serves them automatically.

50. What is the use of app.use() in Express.js?

app.use() is used to add middleware functions in an Express application. It can be used to add global middleware functions or to add middleware functions to specific routes.