ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

SEO Interview

Digital Marketing

ADVERTISEMENT

Python Interview

Django Interview

Pascal Interview

Ruby Interview

ADVERTISEMENT

Ruby on Rails Interview

Memcached Interview

Go Interview

OpenStack Interview

Scala Interview

Control Systems

Electrical Machines

Power System

Digital Electronics

ADVERTISEMENT

Robotics Interview

ADVERTISEMENT

TypeScript Interview

Swift Interview

Blockchain Interview

Bitcoin Interview

AWS Interview

Informatica Interview

QA Interview

React Interview

ADVERTISEMENT

Most Asked Express.js Interview Questions and Answers

Following is a list of most frequently asked Express.js interview questions and their best possible answers.

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 ADVERTISEMENT	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 ADVERTISEMENT	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 ADVERTISEMENT ADVERTISEMENT	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:

×

```
var express = require('express');
var app = express();
app.get('/', function (req, res) {
  res.send('Welcome to JavaTpoint!');
});
var server = app.listen(8000, function () {
  var host = server.address().address;
  var port = server.address().port;
  console.log('Example app listening at http://%s:%s', host, port);
});
```

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

Output:

```
Welcome to JavaTpoint!
```

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

```
var bodyParser = require('body-parser')
app.use( bodyParser.json() );       // to support JSON-encoded
app.use(bodyParser.urlencoded({     // to support URL-encoded
  extended: true
}));
```

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.

```
npm install express-generator -g
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:

```
cd myApp
npm install
```

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

```
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.

> **Note: The third argument is optional and should be omitted; however, in some cases, it is helpful.**

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
ADVERTISEMENT Architecture ADVERTISEMENT	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 ADVERTISEMENT	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.
ADVERTISEMENT 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 ADVERTISEMENT ADVERTISEMENT	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

ADVERTISEMENT

Use the following command on Linux:

ADVERTISEMENT

```
DEBUG=express:*
node app.js
```

Use the following command on Windows:

```
set DEBUG=express:*
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:

```
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();
});
```

ADVERTISEMENT

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

ADVERTISEMENT

ADVERTISEMENT ADVERTISEMENT

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

ADVERTISEMENT

ADVERTISEMENT

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:

```
// error handler
app.use(function(err, req, res, next) {
// set locals, only providing error in development
res.locals.message = err.message;
res.locals.error = req.app.get('env') === 'development' ? err : {};
```

Free Learning Platform For Better Future

```
npm install errorhandler --save
```

Create a variable:

ADVERTISEMENT

ADVERTISEMENT

```
var errorhandler = require('errorhandler')
```

ADVERTISEMENT

Use the middleware as following:

```
if (process.env.NODE_ENV === 'development') {
 // only use in development
  app.use(errorhandler({log: errorNotification}))
}
function errorNotification(err, str, req) {
 var title = 'Error in ' + req.method + ' ' + req.url
 notifier.notify({
   title: title,
   message: str
 })
}
```

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

See the Example:

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

> **Note: 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.**

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.

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

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:

```
npm install MODULE_NAME
```

You can load it by using requires and used later:

See the Example:

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

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

See the Example:

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

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.

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT
ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

ADVERTISEMENT

Latest Courses