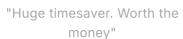


**G** Sign in with Google







"It's an excellent tool"

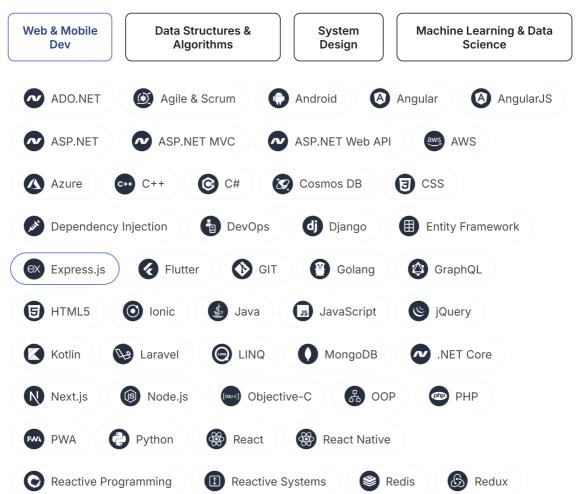


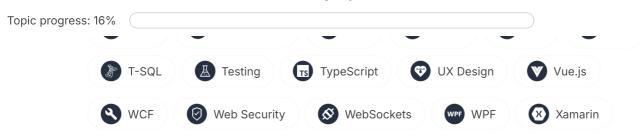
"Fantastic catalogue of questions"

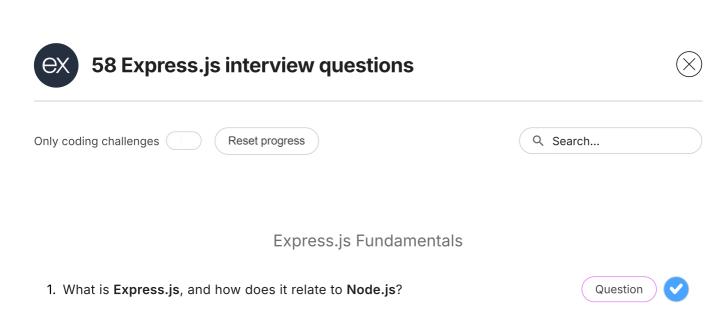
### your next tech interview with

Explore our carefully curated catalog of interview essentials covering full-stack, data structures and algorithms, system design, data science, and machine learning interview questions

Start preparing now







#### **Answer:**

**Express.js** is a web application framework that runs on **Node.js**. It simplifies the process of building web applications and APIs by providing a range of powerful features, including robust routing, middleware support, and HTTP utility methods. Thanks to its modular design, you can expand its functionality through additional libraries and Node.js modules.

#### **Key Features**

- Middleware: Express.js makes use of middleware functions that have access to the request-response cycle. This allows for a variety of operations such as logging, authentication, and data parsing.
- **Routing**: The framework offers a flexible and intuitive routing system, making it easy to handle different HTTP request methods on various URLs.
- Templates: Integrated support for template engines enables the dynamic rendering of HTML content.
- HTTP Methods: It provides built-in methods for all HTTP requests, such as get, post, put, delete, simplifying request handling.
- **Error Handling**: Express streamlines error management, and its middleware functions can specifically handle errors.
- **RESTful APIs**: Its features such as request and response object chaining, along with HTTP method support, make it ideal for creating RESTful APIs.



#### **Relationship with Node.js**

JavaScript runtime environment that allows developers to build server-side and networking applications.

Express.js accomplishes this through a layer of abstractions and a more structured approach, which Node.js, by itself, doesn't provide out of the box.

#### **Code Example: Basic Express Server**

Here is the Node.js code:

```
// Import required modules
const express = require('express');

// Create an Express application
const app = express();
const port = 3000;

// Define a route and its callback function
app.get('/', (req, res) => {
```

Click to expand

2. Explain the concept of middleware in Express.js. Question 3. How would you set up a basic **Express.js** application? Question 4. What is the purpose of the [app.use()] function? Question 5. How do you serve static files using Express.js? Question Discuss the difference between <code>app.get()</code> and <code>app.post()</code> in Question Express.js. 7. How do you retrieve the URL parameters from a GET request in Express.js? Question 8. What are route handlers, and how would you implement them? Question 9. How do you enable CORS in an Express.js application? Question 10. Explain the use of [next()] in Express.js middleware. Question Routing and Requests 11. What is the role of the express. Router class? Question 12. How do you handle 404 errors in Express.js? Question 13. What are the differences between requery and requparams? Question

Go up

How do you create a <b>middleware</b> that logs the <b>request method</b> and <b>URL</b> for every request?	Question
16. Explain how you would implement <b>nested routes</b> in <b>Express.js</b> .	Question Question
17. How can you capture and respond to URL parameters in a route?	Question Question
How do you serve different <b>content types</b> (e.g., JSON, HTML) with <b>Express.js responses</b> ?	Question Question
What are best practices for structuring a large <b>Express.js</b> application with multiple routes?	Question Question
Middleware and Error Handling	
20. Explain the concept and use of built-in middleware in Express.js.	Question Question
21. How do you write <b>custom middleware functions</b> in <b>Express.js</b> ?	Question Question
22. How do you handle file uploads in Express.js?	Question Question
23. What is express-session, and how would you use it?	Question Question
Discuss <b>error handling</b> in an <b>Express.js</b> application. How do you define an <b>error-handling middleware</b> ?	Question Question
25. Provide an example of using third-party middleware, such as body-parser or morgan.	Question Question
How do you protect against <b>SQL injection</b> or other <b>security threats</b> in <b>Express.js</b> ?	Question Question
Response and Performance	
27. How would you implement caching in an Express.js application?	Question Question
28. How do you set cookies and get cookies in an Express.js application?	Question Question
29. What are ways to improve the performance of Express.js applications?	Question Question
30. How do you <b>configure</b> an Express.js app for a <b>reverse proxy</b> , like <b>Nginx</b> ?	Question Question
31. Explain the purpose of <b>template engines</b> . How would you integrate one with <b>Express.js</b> ?	Question Question
Testing and Debugging	
32. How do you test an <b>Express.js</b> application?	Question Question
33. Discuss common debugging techniques for Express.js applications.	Question Co.

Topic	progress:	16%
-------	-----------	-----

35. How do you use a <b>debugger</b> with an <b>Express.js</b> app running in <b>Node.js</b> ?	Question Question
Express.js with Databases	
How would you connect a <b>MongoDB database</b> with an <b>Express.js</b> application?	Question Question
37. Explain how to integrate an <b>ORM</b> like <b>Sequelize</b> with <b>Express.js</b> .	© Question
38. How do you handle database errors in Express.js?	Question Question
What are the advantages of using a <b>database pooling mechanism</b> in an <b>Express.js</b> app?	Question Question
Authentication and Authorization	
40. Describe how you would implement user authentication in Express.js.	© Question
41. Explain how sessions are managed in Express.js.	(B) Question
42. Discuss how you would handle <b>user roles and permissions</b> in an <b>Express.js</b> application.	© Question
43. How can you secure <b>passwords</b> and <b>sensitive information</b> in your <b>Express.js</b> app?	Question Question
Integration and Configuration	
44. How do you integrate a third-party API in an Express.js application?	Question Question
45. Explain the steps to <b>deploy</b> an <b>Express.js</b> application to a cloud provider like <b>AWS</b> or <b>Heroku</b> .	Question Question
46. How can you ensure that your Express.js application is scalable_?	Question Question
What is the <b>Twelve-Factor App methodology</b> , and how does it apply to <b>Express.js</b> ?	Question Question
Coding Challenges	
Write an <b>Express.js middleware</b> function that limits requests to 100 per hour per <b>IP address</b> .	Coding Challenge
49. Create an <b>Express.js route</b> that accepts a <b>JSON payload</b> and responds with the same payload in reverse order.	Coding Challenge
Develop a simple <b>REST API</b> with <b>Express.js</b> that includes <b>CRUD</b> 50. <b>operations</b> for managing books. Include <b>route definitions</b> and <b>handler functions</b> that interact with a placeholder data store.	Coding Challenge Challenge

10pic progress, 1070	Topic	progress:	16%
----------------------	-------	-----------	-----

51.	Discuss strategies for building a real-time application with Express.js.	8	Question
52.	What are some common <b>optimization techniques</b> for <b>Express.js</b> applications?	8	Question
53.	Explain the concept and benefits of <b>server-side rendering</b> with <b>Express.js</b> and a <b>templating engine</b> .	8	Question
54.	How do you ensure that your <b>Express.js code is maintainable</b> and follows <b>best practices</b> ?	8	Question
	Case Studies and Scenario-Based Questions		
55.	How would you structure an <b>Express.js application</b> for a large-scale <b>e-commerce platform</b> ?	8	Question
56.	Describe the key considerations for building a secure API with Express.js.	8	Question
57.	Discuss how to handle <b>session management</b> in a distributed <b>Express.js application</b> .	8	Question
58.	What are the implications of microservices architecture for an Express.js application?	8	Question

# Unlock interview insights

Get the inside track on what to expect in your next interview. Access a collection of high quality technical interview questions with detailed answers to help you prepare for your next coding interview.

### **Track progress**

Simple interface helps to track your learning progress. Easily navigate through the wide range of questions and focus on key topics you need for your interview success.

#### Save time

Save countless hours searching for information on hundreds of low-quality sites designed to drive traffic and make money from advertising.



Go up









Ready to nail your next interview?

# and get

Kickstart your prep



Terms of

Privacy policy

Pricing

Contacts

© 2024 Devinterview.io

