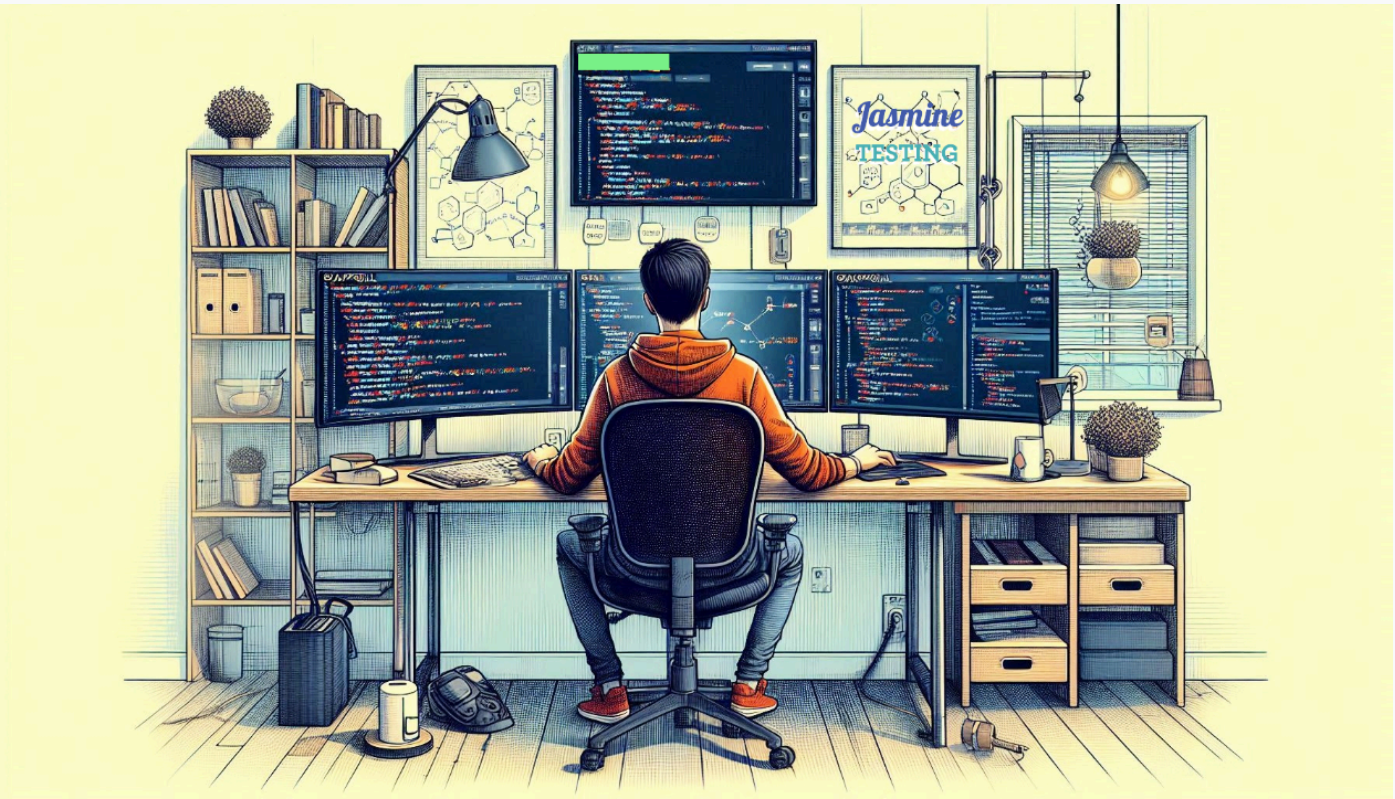
INTERVIEW QUESTIONS

Jasmine Testing Interview Questions: 30 Key Questions and Answers for JavaScript in 2025

Nishi Khan 11 Feb 2025 0 451 0 Share



Basic Level Questions

1. **What is Jasmine in JavaScript?**

Answer: Jasmine is a behavior-driven development (BDD) testing framework for JavaScript, primarily used for unit testing. It is often used in web development for testing JavaScript code, particularly in Angular applications. It allows developers to write tests in a clear, human-readable syntax that describes how a system should behave. Jasmine provides a clean syntax, making it easier to write tests for functions and components.

2. **What are the key features of Jasmine?**

Answer:

- **No dependencies:** Jasmine does not require any external libraries, making it lightweight and easy to use.
- **Asynchronous testing support:** Jasmine provides built-in methods like done() and async/await for testing asynchronous code.

behavior without invoking the original function.

3. **How do you install Jasmine?**
Answer: You can install Jasmine using npm (Node Package Manager) by running the following command:

npm install --save-dev jasmine
After installation, you can initialize Jasmine in your project with npx jasmine init to set up configuration files.

4. **What are the specs in Jasmine?**
Answer: A **spec** is a single test case in Jasmine. It is defined using the it() function. Inside the it() block, you write the code that describes the behavior being tested. A spec always contains an expectation (assertion) that compares actual output with expected results.

Example:

```
it("should add two numbers", function() {
  expect(add(2, 3)).toBe(5);
});
```

5. **What is a suite in Jasmine?**
Answer: A **suite** is a collection of related tests (specs). It is defined using the describe() function, which groups multiple it() blocks together. Suites help organize tests, making it easier to manage and read.

Example:

```
describe("Math Operations", function() {
  it("should add two numbers", function() {
    expect(add(2, 3)).toBe(5);
  });
  it("should subtract two numbers", function() {
    expect(subtract(5, 3)).toBe(2);
  });
});
```

6. **What are matchers in Jasmine?**
Answer: Matchers are functions that allow you to compare expected and actual values in your tests. Jasmine provides several built-in matchers, such as:

- toBe(): Checks for strict equality (===).
- toEqual(): Checks for deep equality (used for objects and arrays).
- toContain(): Checks if an array or string contains a specific element. Example:

```
expect(3).toBe(3); // Passes
expect([1, 2, 3]).toContain(2); // Passes
```

values. Spies can replace real functions with mock versions that simulate behavior but don't execute the code.

Example:

```
let spy = jasmine.createSpy('myFunction');
spy('Hello');
expect(spy).toHaveBeenCalledWith('Hello');
```

8. **How do you disable a test case in Jasmine?**

Answer:

You can disable a test case by using xdescribe() or xit() instead of describe() and it(). This is helpful when you want to temporarily disable certain tests without deleting the code.

Example:

```
xit("should test if function adds two numbers", function() {

  expect(add(2, 3)).toBe(5);
});
```

9. **What is the purpose of the beforeEach() and afterEach() functions?**

Answer: beforeEach() runs before each test case in the suite, and afterEach() runs after each test case. These functions are typically used for setup and teardown tasks such as initializing variables or cleaning up resources.

Example:

```
beforeEach(function() {
  spy = jasmine.createSpy('mySpy');
});
afterEach(function() {
  // Clean up after tests if needed
});
```

10. **How do you test asynchronous code in Jasmine?**

Answer: Jasmine provides two main methods for testing asynchronous code:

- **Using the done callback**: Pass the done callback to the test function to let Jasmine know when the test is complete.
- **Using async/await**: If using async/await, you can mark your test function as async to handle promises easily.
- Example with done callback:

```
it("should load data asynchronously", function(done) {
  asyncFunction().then(function() {
    expect(true).toBe(true);
    done();
  });
});
```

11. **What is the difference between toBe() and toEqual() in Jasmine?**

Answer:

- toBe() compares two values using strict equality (===), meaning both the value and the type must be identical.
- toEqual() checks for deep equality, meaning it checks whether two objects have the same properties and values, even if they are not the same object in memory.
- Example:

```
expect(1).toBe(1); // Passes (strict equality)
expect({ a: 1 }).toEqual({ a: 1 }); // Passes (deep equality)
```

12. **How do you mock dependencies in Jasmine?**

Answer: You can use **spies** to mock dependencies and track calls to functions or methods. By replacing real implementations with spies, you can simulate specific behaviors without triggering the actual code.

Example:

```
spyOn(myObject, 'methodName').and.returnValue('Mocked Value');
```

13. **How do you test promises in Jasmine?**

Answer: You can test promises in Jasmine using async/await or the done callback. If you're using async/await, ensure that the test function is marked as async and use await to wait for the promise to resolve.

Example:

```
it("should resolve a promise", async function() {
  const result = await someAsyncFunction();
  expect(result).toBe('expected value');
});
```

14. **What is spyOn() in Jasmine, and how is it used?**

Answer: spyOn() is a function used to create a spy for a specific method on an object. It allows you to track calls to that method and modify its behavior if needed (e.g., return a mock value or execute a callback).

Example:

```
spyOn(myObject, 'methodName').and.returnValue('Mocked Value');
```

15. **How can you check if a function was called using Jasmine spies?**

Answer: You can check if a function was called using the toHaveBeenCalled() matcher. It verifies that the spy was called at least once.

Example:

```
spyOn(myObject, 'methodName');
myObject.methodName();
expect(myObject.methodName).toHaveBeenCalled();
```

16. How do you set up a Jasmine configuration file?

Example:
npx jasmine init

This will generate the default configuration file, which can be modified as needed.

17. What is the difference between beforeAll() and afterAll() in Jasmine?
Answer:

- beforeAll() runs once before all the specs in a suite, useful for setup that is shared across tests (e.g., initializing objects).
- afterAll() runs once after all the specs in a suite, ideal for cleanup or teardown operations.

Example:

```
beforeAll(function() {
// Setup code, runs once before all tests
});
afterAll(function() {
// Teardown code, runs once after all tests
});
```

18. Can Jasmine be used for testing non-browser JavaScript?
Answer: Yes, Jasmine can be used for testing non-browser JavaScript. While it's often associated with testing client-side code in the browser, it works equally well for server-side code running in Node.js environments. It doesn't rely on the browser, so it can be used to test any JavaScript logic in server-side applications.

19. How do you test error handling in Jasmine?
Answer: To test error handling, you can use toThrow() or toThrowError() matchers to verify that a function throws an error. You can also test specific error types and messages using toThrowError().

Example:

```
it("should throw an error when called with invalid input", function() {
 expect(function() { myFunction(null); }).toThrowError("Invalid input");
});
```

20. How do you run Jasmine tests from the command line?
Answer: You can run Jasmine tests from the command line by using the jasmine command after installing it globally or via npm. The basic command is:
npx jasmine

This will look for spec files and execute all the test cases. You can also specify specific files to run:
npx jasmine spec/mySpec.js

21. What is the use of toHaveBeenCalledWith() in Jasmine?
Answer: toHaveBeenCalledWith() is a matcher used to assert that a spy was called with specific arguments. This helps ensure that the function was called with the expected parameters.

```
myFunction('arg1', 'arg2');
expect(spy).toHaveBeenCalledWith('arg1', 'arg2');
```

22. How do you handle promises and async/await in Jasmine tests?

Answer: Jasmine supports testing promises using async/await by marking the test as async and using await to wait for promise resolution. Alternatively, you can use the done callback for promise handling.

Example with async/await:

```
it("should return the correct value from a promise", async function() {
 const result = await myAsyncFunction();
 expect(result).toBe("expected value");
});
```

23. How do you mock HTTP requests in Jasmine?

Answer: To mock HTTP requests in Jasmine, you can use a library like **Jasmine-Ajax** or manually spy on XMLHttpRequest or fetch. This allows you to mock and intercept HTTP requests to simulate different server responses.

Example with Jasmine-Ajax:

```
beforeEach(function() {
 jasmine.Ajax.install();
});
afterEach(function() {
 jasmine.Ajax.uninstall();
});
it("should mock an HTTP request", function() {
 const request = jasmine.Ajax.requests.mostRecent();
 request.respondWith({
  status: 200,
  responseText: '{"key": "value"}'
 });
 // Test assertions
});
```

24. What is the difference between toBeDefined() and toBeTruthy() in Jasmine?

Answer:

- toBeDefined() checks if a value is not undefined.
- toBeTruthy() checks if a value evaluates to true in a Boolean context (i.e., it's not falsy, such as false, null, 0, "", undefined, or NaN).
- Example:
-

```
expect(variable).toBeDefined();  // Passes if variable is not undefined
expect(variable).toBeTruthy();   // Passes if variable is truthy (not false, null, 0, etc.)
```

25. How can you handle multiple asynchronous operations in Jasmine?

Answer: Jasmine provides multiple ways to handle multiple asynchronous operations:

- Using multiple done() callbacks.

```
 asyncFunction1(function() {
  asyncFunction2(function() {
   expect(true).toBe(true);
   done();
  });
 });
});
```

26. Can you test UI components with Jasmine?

Answer: While Jasmine is primarily designed for unit and functional testing of JavaScript, you can use it in combination with libraries like **Jasmine-UI**, **Karma**, or **Protractor** for testing UI components. These tools allow you to simulate browser interactions and test front-end behavior. However, for more advanced UI testing, frameworks like **Cypress** or **Selenium** are typically preferred.

27. How do you test a function that interacts with the DOM in Jasmine?

Answer: You can test DOM interactions in Jasmine by manipulating the DOM in your test case and then asserting the changes. You may use document.createElement to create new DOM elements or manipulate existing ones.

Example:

```
it("should change the text content of an element", function() {
 const element = document.createElement("div");
 element.textContent = "Old Text";
 document.body.appendChild(element);

 updateTextContent(element);

 expect(element.textContent).toBe("New Text");
});
```

28. What is toThrowError() used for in Jasmine?

Answer: toThrowError() is used to test if a function throws a specific error, and it can also be used to check the error message.

Example:

```
it("should throw an error with a specific message", function() {
 expect(function() { throw new Error("Oops"); }).toThrowError("Oops");
});
```

29. What is the and.callFake() function in Jasmine?

Answer: and.callFake() is used to replace a spy's original behavior with a custom function, allowing you to define what happens when the spy is called. This is useful for testing specific logic without invoking the actual function.

Example:

```
const spy = jasmine.createSpy('myFunction').and.callFake(function() {
 return "Fake return value";
});
```

Answer:

You can test the length of an array in Jasmine using the toHaveLength() matcher (introduced in Jasmine 3.0). It checks the length of an array or a string.

Example:

```
it("should have the correct array length", function() {
 const arr = [1, 2, 3];
 expect(arr).toHaveLength(3);
});
```

qa | software testing | sqa | javascript | jasmine

Share your thoughts Or Start discussion

Related Blogs

INTERVIEW QUESTIONS 0 0 217

Smoke Testing Interview Questions: Most Asked QA Topics

1. What is Smoke Testing?Answer: Smoke Testing is a type of software testing per

ridwan
13 Mar 2025

INTERVIEW QUESTIONS 0 0 366

Mastering Katalon Studio: Common Interview Questions Explained

1. What is Katalon Studio?Answer:Katalon Studio is an automated testing tool f

Sebastian Leon
17 Feb 2025

sqa testing qa software testing qabrains testing tool

automationtesting softwaretesting mobiletesting selenium

View All

Popular Post

Can a Software Tester Become a Game Tester? Here's What You Need t...

As the gaming industry continues to grow, fueled by innovations in virtual reali

Understanding Java Object-Oriented Programming (OOP) Concepts

Java is a powerful and widely used programming language known for its versatilit

Essential Bugs to Check for in Game Testing: A Guide for Beginners

Game testing is crucial to ensure a smooth, engaging, and bug-free experience fo

JMeter: Short technique for Generating an HTML load test report using...

Pre-requisites:Install Java:Java Version: "1.8.0_291" or higher (minimum require

View All

Popular Discussion

01 Top Software Testing Interview Questions and Expert Tips from QA Leaders

02 AI tools for QA engineer

03 What is SQL?

04 Appium, WebDriver

05 What are the most effective strategies you've found for balancing speed and...

View All

QA Brains is the ultimate QA community to exchange knowledge, seek advice, and engage in discussions that enhance Quality Assurance testers' skills and expertise in software testing.

QA Topics

Web Testing

Interview Questions

Game Testing

See more →

Quick Links

Discussion

About Us

Terms & Conditions

Privacy Policy

Follow Us

For Support

support@qabrains.com