1. **What is Karma?**
   * Karma is a test runner developed by the Angular team. It allows you to execute JavaScript code in multiple real browsers.
2. **What is Jasmine?**
   * Jasmine is a behavior-driven development (BDD) testing framework for JavaScript. It provides functions and methods to structure and write test cases.
3. **Why do we use testing in Angular applications?**
   * Testing helps ensure that the application behaves as expected and catches bugs early in the development process. It also promotes maintainability, scalability, and confidence in the codebase.
4. **What are unit tests in Angular?**
   * Unit tests in Angular are focused on testing individual components, services, pipes, or directives in isolation from the rest of the application.
5. **How do you write a simple unit test in Angular using Jasmine?**
   * A simple unit test in Angular using Jasmine typically involves creating a test suite (**describe**) and one or more test specs (**it**).
6. **What is a TestBed in Angular testing?**
   * TestBed is an Angular testing utility that provides methods for configuring and creating instances of Angular components, services, and modules for testing.
7. **Explain the role of beforeEach and afterEach in Jasmine tests.**
   * **beforeEach** and **afterEach** are Jasmine functions that are used to set up and tear down the test environment before and after each test case, respectively.
8. **What is the purpose of compileComponents in Angular unit tests?**
   * **compileComponents** is used to compile the components declared in the TestBed configuration before running the tests. It ensures that the templates are compiled and the component factories are available for testing.
9. **How do you mock dependencies in Angular tests?**
   * You can mock dependencies in Angular tests by providing fake implementations or stubs for services, components, or other dependencies using TestBed or Jasmine's spyOn function.
10. **What is the difference between compileComponents and compileComponentsAsync?**
    * **compileComponents** compiles the components synchronously, while **compileComponentsAsync** compiles the components asynchronously using the TestBed.compileComponents method.
11. **How do you test asynchronous code in Angular with Jasmine?**
    * You can use Jasmine's **done** function, **async**/**await** syntax, or Angular's **fakeAsync** and **tick** functions to test asynchronous code in Angular.
12. **What is code coverage, and how is it useful in Angular testing?**
    * Code coverage is a metric that measures the percentage of code covered by tests. It helps identify areas of the codebase that are not adequately tested and ensures comprehensive test coverage.
13. **How do you configure code coverage reports in Angular tests with Karma?**
    * You can configure code coverage reports in Angular tests with Karma by adding the appropriate settings to the **karma.conf.js** file, such as specifying the reporters and coverage thresholds.
14. **What is a spy in Jasmine, and how is it used in Angular testing?**
    * A spy in Jasmine is a function that tracks calls to other functions and provides information about those calls, such as the number of times they were called and with what arguments. Spies are commonly used in Angular tests to mock dependencies and verify function calls.
15. **What are some best practices for writing effective unit tests in Angular?**
    * Some best practices for writing effective unit tests in Angular include testing one thing per test, using descriptive test names, using beforeEach and afterEach hooks appropriately, keeping tests focused and isolated, and using mock objects and spies to mock dependencies.
16. What are the methods for ‘expect’ in Jasmine?

Matchers:

Matchers are functions that compare the actual value (the value being tested) with the expected value. They determine whether the expectation is fulfilled or not.

Common matchers include:

toEqual(expected): Compares the actual and expected values using deep equality.

toBe(expected): Compares the actual and expected values using strict equality (===).

toBeTruthy(): Checks if the actual value is truthy (not falsy).

toBeFalsy(): Checks if the actual value is falsy (not truthy).

toContain(expected): Checks if an array or string contains the expected value.

toBeCloseTo(expected, precision?): Checks if the actual value is close to the expected value within a certain precision (useful for comparing floating-point numbers).