Getting Started

Terms

Automated testing
AAA pattern
Code coverage
End-to-end tests
Integration tests
Test framework
Test-driven Development (TDD)
Unit tests

Summary

- Automated testing involves writing code to verify that our software behaves as expected. When done properly, it can significantly reduce the need for and cost of manual testing. However, it cannot entirely replace manual testing, as certain aspects of an application (e.g., aesthetics) are best assessed by human testers.

- In automated testing, there are three primary types of tests: *unit tests*, *integration tests*, and *end-to-end tests*.

- Unit tests validate the correctness of small, isolated units within an application.

- Integration tests examine interactions between different units of an application.

- End-to-end tests assess the entire application as a whole, simulating user interactions from the user interface to the back-end services.

- A *testing framework* is a collection of tools used for writing and executing tests. Popular testing frameworks include Jest, Vitest, and Mocha, among others.

- Vitest offers native support for EcmaScript Modules (ESM) and TypeScript, making it a suitable choice for modern JavaScript projects.

- Test structures often follow the *AAA (Arrange-Act-Assert) pattern*. During the Arrange phase, we set up the test environment. In the Act phase, we perform the action to be tested, and in the Assert phase, we validate the outcome against our expectations.

- *Test-driven Development (TDD)* is a development approach where tests are written before the corresponding code is implemented, promoting a clear development process and well-tested code.

- *Code coverage tools* measure the proportion of code that is tested. While striving for high code coverage is valuable, aiming for 100% coverage is often impractical in many projects, and it does not guarantee a bug-free application, as 100% coverage does not imply the absence of defects.

Writing Tests

```
import { describe, expect, it } from ‘vitest’;

describe(‘test suite’, () => {
   it(‘test case’, () => {
      expect(max(1, 2)).toBe(2);
   });
});
```