# **Testing**

## Other JavaScript environments - node & NPM

Node.js comes with a package manager called npm, which stands for Node Package Manager. The package manager allows you to use a large number of libraries and frameworks as Node.js modules. An npm module is a standalone piece of code that has been published on the npm website. Sometimes an npm module is also referred to as an npm package.

*Which of the following statements are true for Node.js?*

* *Node.js is a separate, standalone environment without ties to the JavaScript in the browser.*
* *Node.js can run in multiple settings, for example, on the command line, in a desktop application, or on the back-end of a web app (on a server).*

When you want to start a new project, first, open a folder on your machine where you want to place your project files, then run the npm command. These projects can be different shapes and sizes, but they all have at least one thing in common, the package.json file that gets created after you run the npm command. The package.json file holds all the instructions on all the node modules that are pulled from the npm repository of open-source modules.

## What is testing?

You need to make sure that your JavaScript code works before you launch it.

There are many custom testing frameworks available in JavaScript. One of the strengths of such frameworks is that I don't have to use comments to describe my expectations. The test syntax itself becomes expectation documenting. When I write tests, those tests are a better alternative to comments in my source code because they specify what expectations my source code is trying to satisfy.

Chart

Description automatically generated with low confidence

Testing Javascript code allows for conciseness, clarity, and repeatability.

*Which of the following is a correct example of a Jest test?*

* *expect(concatStrings("123", "456")).toBe("123456");*

When tests fail, you say that they are red, and when they pass, you say that they are green. If a test fails, then it's a sign that I need to write the code in such a way that it passes its test. Once my test passes, I need to improve both the app's code and the test code, but without changing the behaviour of either. This is known as refactoring.

Refactoring is updating code, without affecting the results it produces.

This cycle is the basis of the test-driven development or TDD approach to programming. Let me explain the TDD approach. First, you write a failing test, then you write your source code so that the previously failing test now passes. Finally, you optimize your source code without changing its results.

## Types of testing

|  |  |
| --- | --- |
| **End 2 end** | More specific to web development into. E2e testing tries to imitate how a user might interact with your app. This means that in e2e testing you need to open your web application in a browser and then test it by interacting with the page the same way a user might interact with it. For example, clicking on the log in button are going through the process of adding an item to the shopping cart. In other words, you're testing the entire finished software product from the perspective of the end user.  **E2E testing is slow and time consuming.** |
| **Integration** | Integration testing is testing how parts of your system interact with other parts of your system. In other words, it's testing how separate parts of your apps work together.  **Integration tests are faster and cheaper than E two E tests but not as fast or as cheap as unit testing.** |
| **Unit** | Unit testing is the process of testing the smallest units of your source code in isolation. A good example of this is functions. A unit is the smallest piece of code that you can test separately from the rest of the app. Practically the smallest unit of testable code in Js is usually a function or a method. Unit tests are self-contained. They're meant to test code in isolation, preferably separate from the rest of your app. This makes unit tests fast to run and easy to write. |

*Which type of testing involves the process of testing the smallest units of your source code in isolation?*

* *Unit testing*

### Introduction to Jest

Jest is a JavaScript testing framework. It's often used for testing code like React, a JavaScript library maintained by Meta and a community of individual developers and companies. Besides plain JavaScript and React code just allows you to test Babel, TypeScript, Node, Angular, Vue, and various other frameworks.

|  |  |
| --- | --- |
| **Mocking** | Mocking allows you to separate the code that you are testing from its related dependencies. In other words, you can use the mocking features to make sure that your unit testing is stand-alone. For example, you can test the front-end functionality of your web app by mocking the data as if it came back from a server when in fact it came from the client. |
| **Snapshot** | Snapshot testing is used by developers to verify that there are no regressions in the DOM of our apps after some changes to the code base are made. |
| **Code coverage** | Code coverage is a measure of what percentage of my code is covered by tests. |

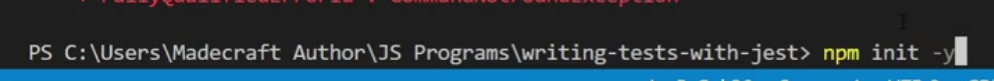
*Which of the following features does Jest provide? Check all that apply.*

* *A range of frameworks can be tested with Jest.*
* *Jest can be used to write unit tests.*
* *Jest supports code coverage.*

## Writing tests with Jest

A screenshot of a computer screen

Description automatically generated with medium confidence





Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

*What will happen when you run this piece of code on the command line at the root of your project's folder: npm install --save-dev jest?*

* *It will add Jest as a devDependency inside package.json.*

## TDD (Test-Driven Development)

Every piece of software is built according to formal or informal requirements. The purpose of the requirements is to explain, in human language the intricacies of what the piece of software does.

test driven development is a streamlined process of writing code that will satisfy some requirements.

On a high level. A software development teams work consists of the following receiving requirements which will become a feature of the app that's being developed. Writing a failing test for that to build feature before it gets built. Making this failing test pass by coding that given feature in comparison with the traditional development process, that TDD approach might seem somewhat upside down.

Diagram

Description automatically generated

it's important to note that one of the rules of TDD is that you should write as little code as possible to make the test pass.

In short you write the test before creating the actual code for your program.

Graphical user interface, application, Teams

Description automatically generatedA picture containing text

Description automatically generated

This process is usually explained in three words. Red, Green. Re factor, red represents the failing test. Green on the other hand, represents the passing test after you make updates to the source code.

The re factor represents the final tweaks to the code that don't change implementation details, which can always be confirmed by running another subsequent test when implemented correctly.

TDD brings huge benefits to an organization because it allows for automated testing in any platform projects grow bigger over time and become complex. Making sure that all the tests are passing is a strong signal that the current requirement and all the previous requirements for this piece of your app have been delivered successfully and that nothing is breaking.

*Following the TDD approach has many benefits. Select all the benefits following the TDD approach has for you as a developer. Check all that apply.*

* *Implementations can be tested using various inputs.*
* *Minimize regressions.*