Link: [Unit Testing For Beginners | Quick Guide | by Aditya Krishnan | Medium](https://medium.com/%40adityakrshnn/unit-testing-for-beginners-d196fc911873)

**Unit Testing for Beginners**

Advantages of Unit Tests:

* Points out regressions in code.
* Makes refactoring code easier as it points to any failures in code before the code is pushed.
* Detects code quality; Code that’s easier to read and test is usually better code.

Jest can do the following:

* Check the return value of a function.
* Check the return type of function.
* Check if a function called another function.
* Check how many times a function was called.
* Check if the returned array/object contains or doesn’t contain something and many more.

**Note: Functions should only be tested for important functionality.**

**test** and **describe**

* **test** represents either whole or part of a specification that should be tested.
* Each test case for a function should be written inside a separate **test** function. All the test cases of a function should be put inside the same **describe** function. All the functions of a file / module should have their own **describe** functions.
* **expect** compares / checks one value with another.
* Eg – **expect(A).equalTo(B);**
* **Jest.SpyOn** - You can spy on a function to check if it was called or even make it do something else just for a particular test. For example, you can mock a function’s return value or implementation.

**Some Ways to Test A Function**

* If you think that for a set of inputs, the output should be a particular value. Then put a test for that.
* If you think that your function should call this other function for correct functioning, then put a test for that.
* If a function should call another function with a particular set of arguments, then check for that.

**Best Practices**

* Make sure to mock implementations of every other function other than your function. Otherwise, you are at risk of breaking your unit test because someone else changed their function.
* Write functional code as much as possible. They are much easier to test as they have the same outputs for the same inputs.
* Make sure your functions do only one thing. If it does multiple important steps, separate them out into more functions, so that you can test them separately. Otherwise, your test specification will get complex (which increases cyclomatic complexity and is bad).
* If using **Jest.SpyOn**, do not forget to **mockRestore** after you do the **expect**, otherwise, the spy will remain for all remaining tests as well.