# JS Unit Testing

## Learning Objectives

- [ ] Understand what unit testing is and how it relates to other testing methods
- [ ] Know how to write a unit test that checks the output of a function
- [ ] Understand what Test Driven Development is and how it is used
- [ ] Know how to run unit tests locally (via the command line)

---

## What are unit tests

When developing apps, errors inevitably emerge in your code. That's why apps must be tested regularly to ensure that they work as expected. Since manual testing by clicking in the UI is very time consuming and cumbersome, tests are automated.

Test automation can be performed in several ways. A good classification is provided by the Testing Trophy.

Static Testing refers to linting and can be understood as a spell checker for your code.

Unit tests check whether a single function works as intended. The goal is to test each individual unit independently and isolated from other data and external influences. The test can be run automatically after each modification to ensure latest changes won't break the app.

---

## Test Driven Development (TDD)

With Test driven development (TTD) the test is written first. Afterwards you write the function, which should create a desired result.

Following this approach you get feedback as early as possible whether your work is going into the right direction. In the beginning all tests will fail. Gradually more and more test will be successful. The implementation of your function is finished as soon as all test runs are successful.

---

## How to test with jest

Tests are placed in a file next to the code you like to test, but with `.test.js` as filename ending.

```
calculator.js           <--- Code used in your app
calculator.test.js      <--- Tests for this code
```

When writing tests, you build various different scenarios that check a certain desired result of a function. Each of such test cases is wrapped into a function called test(). The first argument of the `test()` function is a description of the test case in plain english.

Unit tests usually have a similar structure. First, the function to be tested is called and any required arguments are passed. Afterwards the result of this function call is passed to the expect() function. Thus different matcher functions like `toBe()` or `toEqual()` can be used.

This way the actually generated result of the function is checked against an expected result defined in the test case.

```
import { add } from "calculator";

test("adds the numbers 1, 2 and 3 correctly", () => {
  const result = add(1, 2, 3);
  expect(result).toBe(6);
});

test("adds the numbers 13, 28 and 42 correctly", () => {
  const result = add(13, 28, 42);
  expect(result).toBe(83);
});
```

---

## Run Tests locally

Run all tests:

```
npm run test
```

When executing this commands you will see the result of the test run: a list of all included tests and whether they were successful or failed.

---

## Resources

- Jest
- Testing Trophy and Testing Classifications