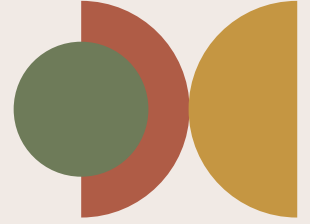
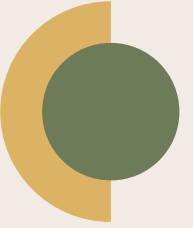


Unit Testing in Javascript

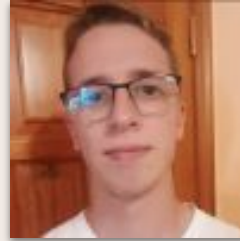


Speakers



Alexander G. Covic

- alu0101397627@ull.edu.es
- alexander.covic.16@ull.edu.es



José Lozano Armas

- alu0101392561@ull.edu.es
- jose.lozano.armas.10@ull.edu.es



Table of contents

01

Introduction

Testing, Unit Testing, Jest...

02

Installation

Quick guide on how to install and configure Jest

03

Topics

Basic, Matchers, setup and teardown...



01

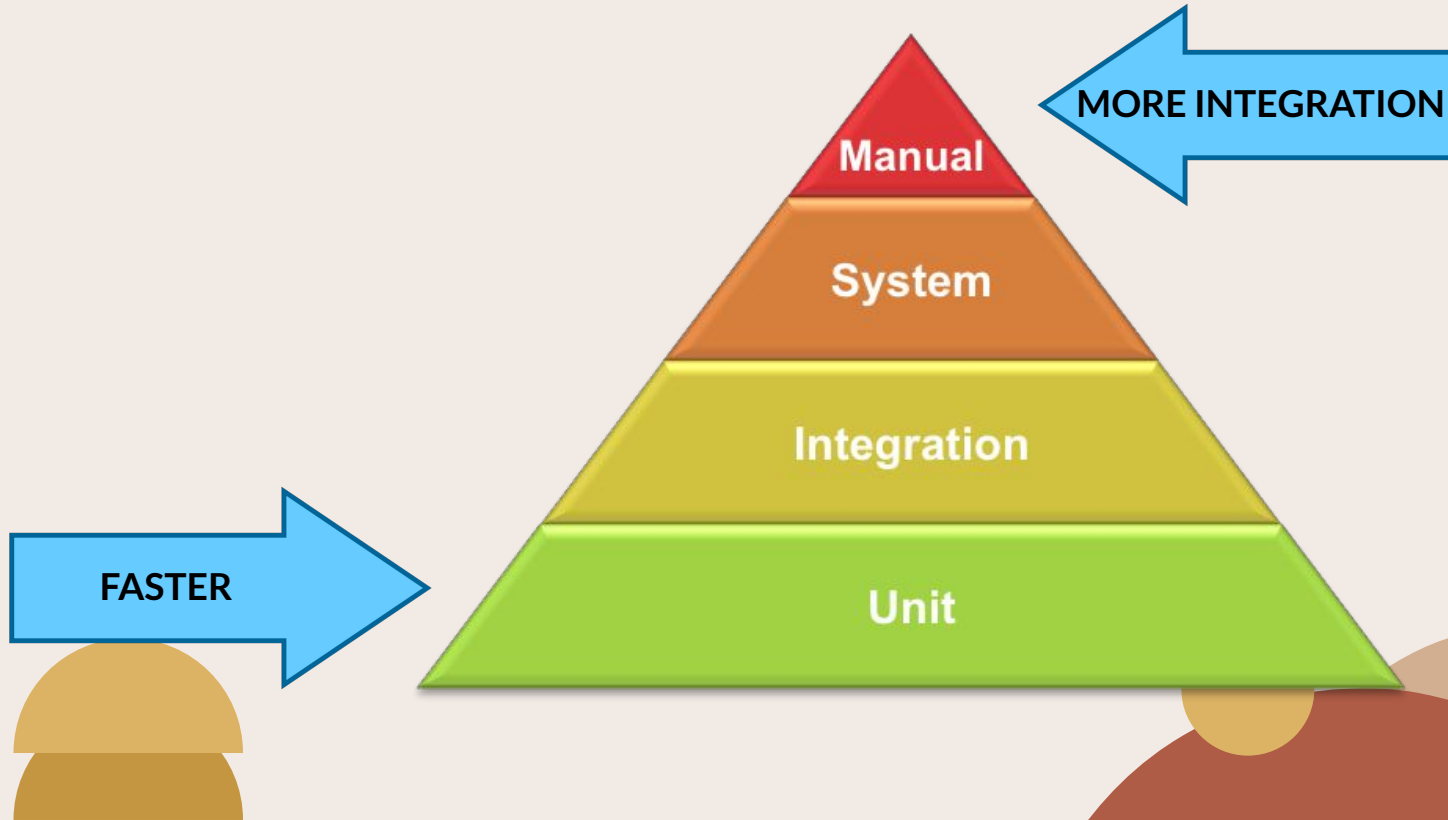
Introduction

Testing, Unit Testing, Jest...

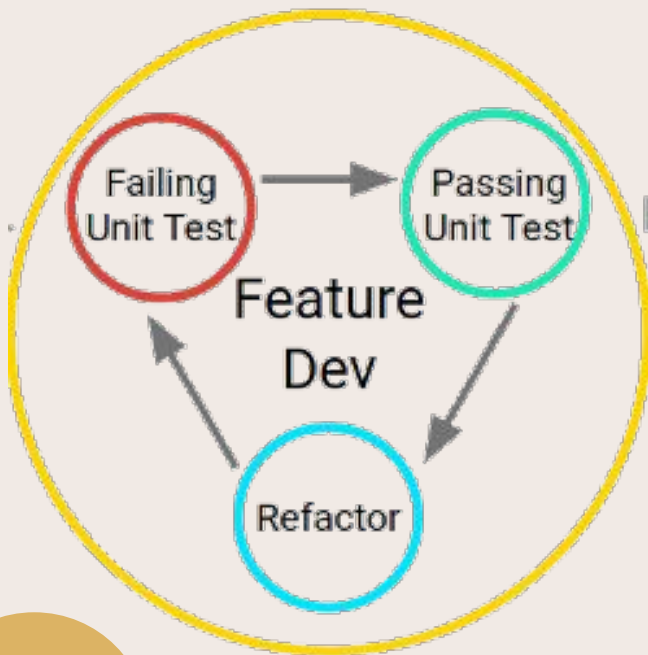
What is testing?



Testing Pyramid



Unit Testing



```
1 test('adds 1 + 2 equal 3'), () => {  
2   expect(sum(1, 2)).toBe(3);  
3 });
```

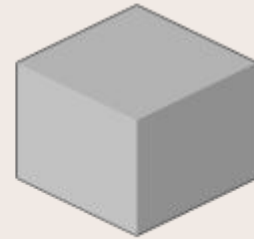
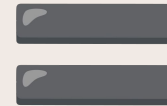
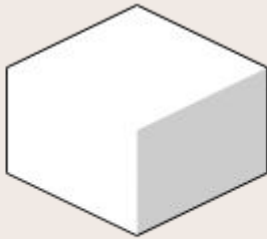
```
PASS tests/basic.test.js  
✓ adds 1 + 2 to equal 3 (3 ms)  
sum module  
  ✓ adds 1 + 2 to equal 3 (1 ms)
```

Unit testing types

White-Box
Testing

Black-Box
Testing

Gray-Box
Testing



Advantages of unit testing



What makes testing good?

Quality

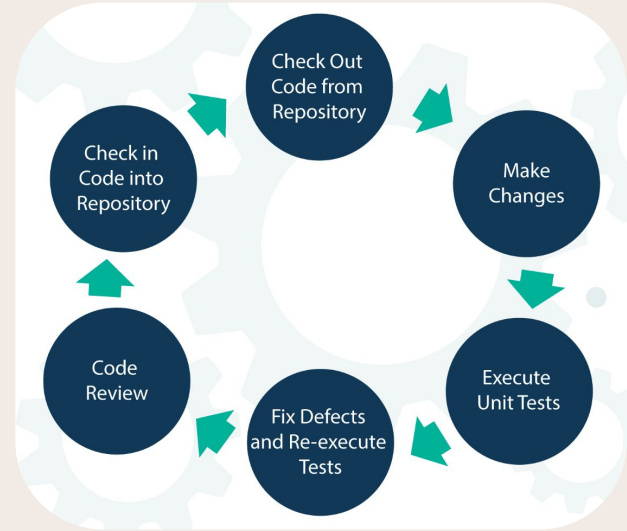
Representative
Edge cases



Quantity

Code coverage
Number of tests

Jest



Features of JEST

01

FAST

02

INTUITIVE INTERFACE

03

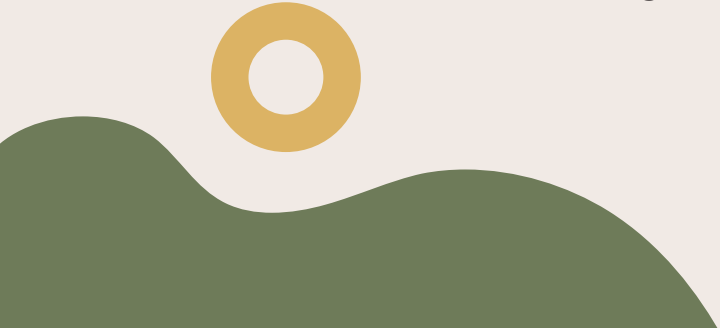
GREAT EXCEPTIONS



02

Installation

Quick guide on how to install and configure Jest



How to Install: Jest

```
$ npm install jest --save-dev
```

```
// package.json  
"scripts": {  
  "test": "jest"  
}
```

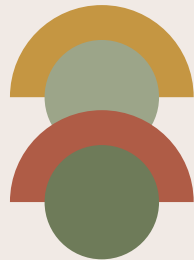
Common Error

if (error?.stack)

```
$ sudo npm install -g n  
$ n lts  
$ hash -r
```

```
$ nvm latest  
$ source ~/.bashrc
```

Jest Project: Config & CLI



Create Jest Configuration File

```
$ npm test -- --init
```

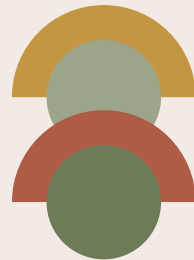
Few CLI Options

```
$ npm test regex  
$ npm test dir/file.test.js  
  
$ npm test -- -o
```


Jest Project: Directory Structure

```
.  
├── LICENSE  
├── README.md  
├── node_modules/  
├── jest.config.js  
├── package.json  
├── package-lock.json  
├── src  
│   └── basic.js  
└── tests  
    └── basic.test.js
```

Recommended Extensions



Jest

Orta | 📦 1,404,367 installs | ★★★★★ (72) | Free

Use Facebook's Jest With Pleasure.

Installation

Launch VS Code Quick Open (Ctrl+P), paste the following command, and press enter.

```
ext install orta.vscode-jest
```

Copy

| [More Info](#)



03

Topics

Basic, Matchers, setup and
teardown...

Jest Basics

```
1 test('short description of the test', () => {  
2   expect(true).toBe(true);  
3 }, 1000);
```

- **String:** “short description”
- **Function** containing the expects
- **(Optional) Int:** Maximum time to abort (*milliseconds*)

Jest Basics

```
1 // File: sum.js
2 const sum = (opOne, opTwo)
3 => {
4   return opOne + opTwo;
5 };
6
7 module.exports = sum;
```

```
1 // File: sum.test.js
2 const sum = require('sum');
3
4 test('adds 1 + 2 equal 3', () => {
5   expect(sum(1, 2)).toBe(3);
6 });
```

Basics: Group Tests

```
1 // We can use the describe sentence to divide the tests
2 describe('sum module', () => {
3     test('test 1', () => {expect(...)}),
4     test('test 2', () => {expect(...)}),
5     ...
6+n   test('test n', () => {expect(...)}),
7+n   });
```

Jest Matchers

```
1 expect(1 + 3 < 5).toBe(true);  
2 expect('abc'.includes('a')).toBe(true);  
3 expect([1, 3, 5].includes(3)).toBe(true)
```



Matchers: Most Common

toBe

```
1 const SAME_OBJECT_ONE = [1];  
2 const SAME_OBJECT_TWO = SAME_OBJECT_ONE;  
3  
4 const DIFF_OBJECT_ONE = [1];  
5 const DIFF_OBJECT_TWO = [1];  
6  
7 expect(SAME_OBJECT_ONE).toBe(SAME_OBJECT_TWO); // OK!  
8 expect(DIFF_OBJECT_ONE).toBe(DIFF_OBJECT_TWO); // Not OK
```


Matchers: Most Common

toStrictEqual

```
1 const SAME_OBJECT_ONE = [1];  
2 const SAME_OBJECT_TWO = SAME_OBJECT_ONE;  
3  
4 const DIFF_OBJECT_ONE = [1];  
5 const DIFF_OBJECT_TWO = [1];  
6  
7 expect(SAME_OBJECT_ONE).toStrictEqual(SAME_OBJECT_TWO); // OK!  
8 expect(DIFF_OBJECT_ONE).toStrictEqual(DIFF_OBJECT_TWO); // OK!
```

Matchers: Most Common

Not

```
1 expect('apple').not.ToBe('banana'); // OK!
```

Matchers: Truthiness

toBeNull

Check if the value is **null**.

toBeDefined

Check if the value is
defined.

toBeUndefined

Check if the value is
undefined.

toBeTruthy

Check if the value is **true**.

toBeFalsy

Check if the value is **false**.

Matchers: Truthiness

```
1 const NULL_EXAMPLE = null;
2 expect(NULL_EXAMPLE).toBeNull();
3 expect(NULL_EXAMPLE).toBeDefined();
4 expect(NULL_EXAMPLE).not.toBeUndefined();
5 expect(NULL_EXAMPLE).not.toBeTruthy();
6 expect(NULL_EXAMPLE).toBeFalsy();
```



Matchers: **Numbers**

toBeGreaterThan

Check if the value is **bigger**.

toBeLessThan

Check if the value is **smaller**.

ToBeGreaterThanOrEqual

Check if the value is **bigger or equal**.

ToBeLessThanOrEqual

Check if the value is **smaller or equal**.

Matchers: Numbers

```
1  const VALUE = 2 + 2;  
2  expect(VALUE).toBeGreaterThan(3);  
3  expect(VALUE).toBeGreaterThanOrEqual(3.5);  
4  expect(VALUE).toBeLessThan(5);  
5  expect(VALUE).toBeLessThanOrEqual(4.5);  
6  expect(VALUE).toEqual(4);
```

Matchers: Float

```
> console.log(0.1 + 0.2);  
0.30000000000000004
```

```
1 test('Floating values', () => {  
2   const VALUE = 0.1 + 0.2;  
3   //expect(VALUE).toStrictEqual(0.3);  
4   expect(VALUE).toBeCloseTo(0.3, 3);  
5 })
```

Matchers: Strings

Check Strings with Regex

```
1 test('Value does not contains i', () => {  
2   expect('Value').not.toMatch(/i/);  
3 });
```


Matchers: **Array** and Iterables

Check if contains a value

```
1 // Arrays and Iterables
2 test('Array contains 3', () => {
3   const EXAMPLE_VALUES = [1, 3, 5, 8];
4   expect(EXAMPLE_VALUES).toContain(3);
5 });
```

Matchers: Exceptions

```
1 // Exceptions
2 const errorFunction = () => {
3   throw new Error('Throw error.');
```

```
4 };
5
6 test('Error example', () => {
7   expect(() => errorFunction()).toThrow();
8   expect(() => errorFunction()).toThrow(/error/);
9   expect(() => errorFunction()).toThrow('Throw error');
10 });
```

Setup **and** teardown

```
beforeEach(() => {  
  exampleFunction();  
});
```

```
beforeAll(() => {  
  anotherFunction();  
});
```

```
afterEach(() => {  
  exampleFunction();  
});
```

```
afterAll(() => {  
  exampleFunction();  
});
```

The use of **before** and **after all**

```
1 beforeAll(() => {
2   list.insert('bananas');
3   list.insert('apples');
4   list.insert('pears');
5 });
6
7 afterAll(() => {
8   list.reset();
9 });
10
11 test('The list contains bananas, apples and tomatoes', () => {
12   expect(list.getList()).toEqual(['bananas', 'apples', 'pears']);
13 });});
```

Basic Setup and teardown

```
1 beforeEach(() => {  
2   list.insert('apples');  
3   list.insert('pears');  
4 });  
5  
6 afterEach(() => {  
7   list.reset();  
8 });  
9  
10 test('The list contains apples and pears', () => {  
11   expect(list.getList()).toEqual(['apples', 'pears']);  
12 });
```

¿Before each **or** before all?

```
beforeAll(() => {  
  anotherFunction();  
});
```



```
afterAll(() => {  
  exampleFunction();  
});
```

Scoping

```
1 beforeEach(() => {
2   list1.insert('bananas');
3 });
4 test('The list contains bananas, apples and tomatoes', () => {
5   expect(list1.getList()).toStrictEqual(['bananas']);
6 });
7 describe('The list of school materials', () => {
8   beforeEach(() => {
9     list2.insert('pencil');
10  });
11  test('The list contains pencil, rubber and a pen', () => {
12    expect(list2.getList()).toStrictEqual(['pencil']);
13  });
14 });
```

The use of **only**

```
1 beforeEach(() => {
2   list.insert('apples');
3   list.insert('pears');
4 });
5
6 test.only('The list contains apples and pears', () => {
7   console.log('I am the only test that will work');
8   expect(list.getList()).toStrictEqual(['apples', 'pears']);
9 });
10
11 test('this test will not run', () => {
12   console.log('I will fail, unless you put the keyword:
13     \'only\', in the one above me');
14   expect(list.getList()).toStrictEqual(['apples']);
15 });
```


References

- Jest
- Jest Expect Object
- Jest Timeout Not Async Functions - Facebook Open Issue
- Jest Matchers
- Jest Setup and Teardown
- Exercism
- Repository