Jest: Javascript Testing Framework

Speakers

Alexander G. Covic
- *alu0101397627@ull.edu.es*

José Lozano Armas
- *alu0101392561@ull.edu.es*

Table of contents

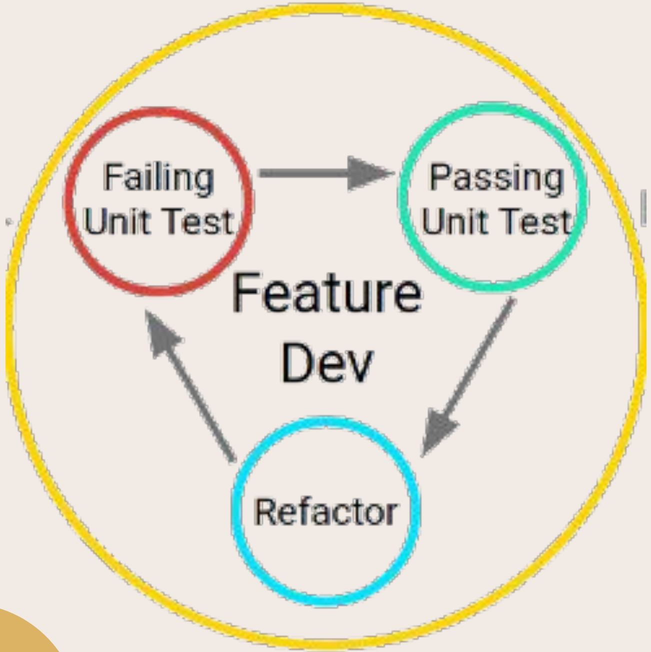
01

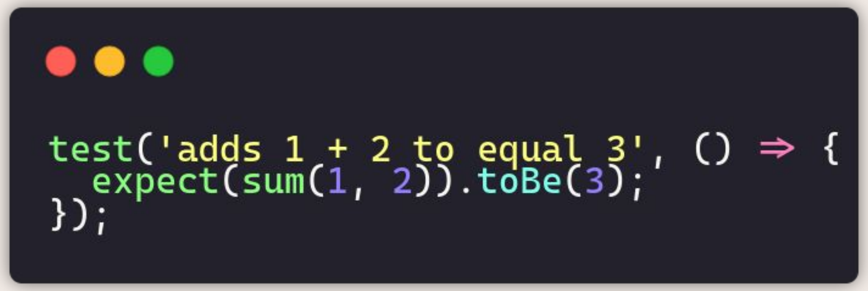
Introduction

Testing, Unit Testing, Jest…

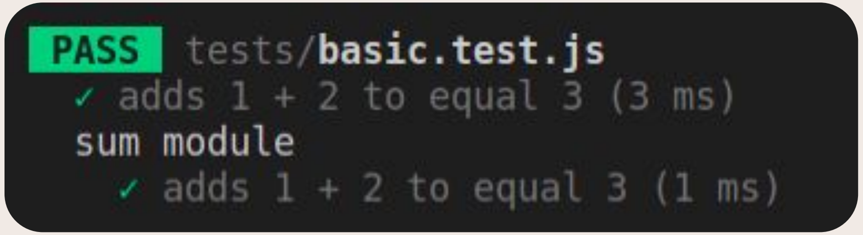
What is testing?

Unit Testing



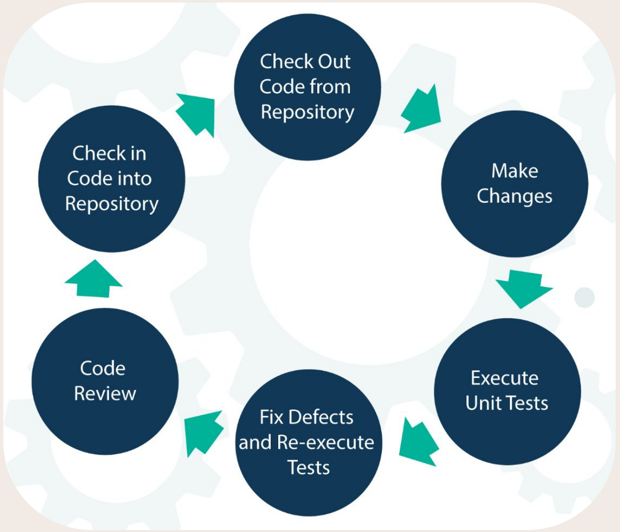


```
test('adds 1 + 2 to equal 3', () => {
  expect(sum(1, 2)).toBe(3);
});
```



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

Jest


Check Out Code from Repository
Make Changes
Execute Unit Tests
Fix Defects and Re-execute Tests
Code Review
Check in Code into Repository

Features of JEST

01 FAST AND SAFE

02 EASY MOCKING

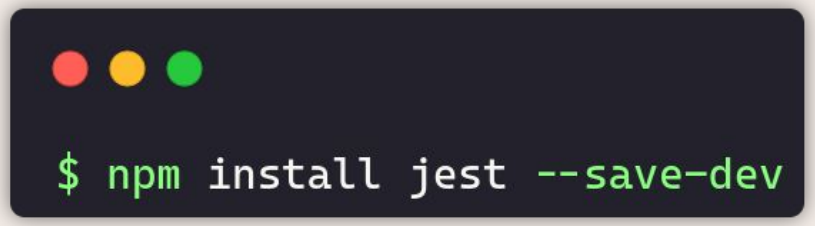
03 PHILOSOPHY

02

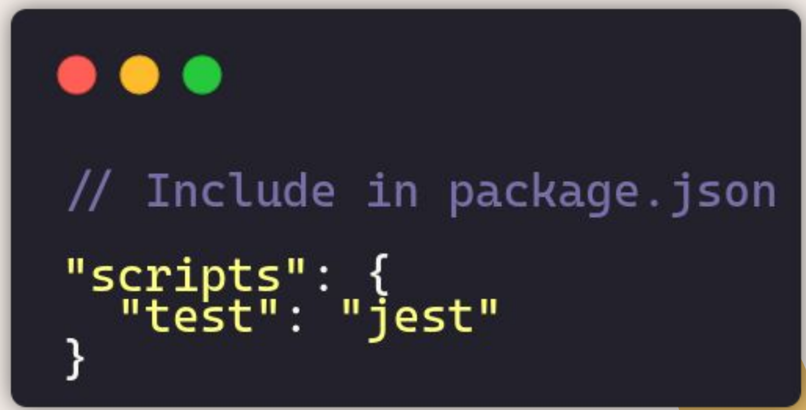
Installation

Quick guide on how to install and configure Jest

How to Install: Jest



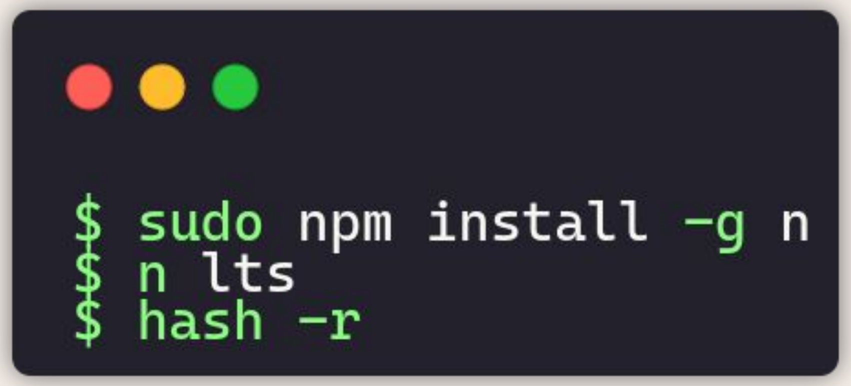
```
$ npm install jest --save-dev
```



```
// Include in package.json

"scripts": {
   "test": "jest"
}
```

Common Error

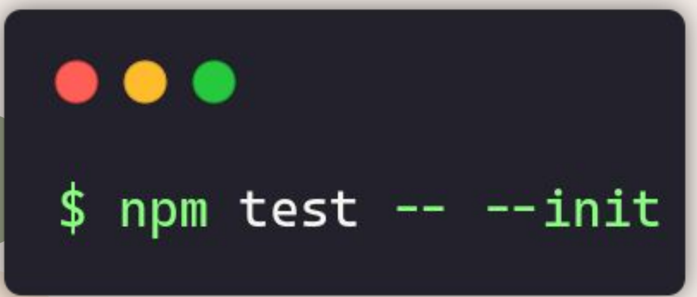
if (error?.stack)



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

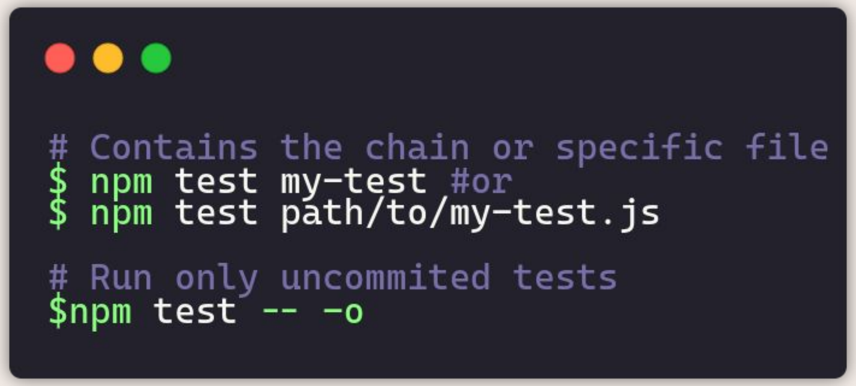
Jest Project: Config & CLI

Create Jest Configuration File



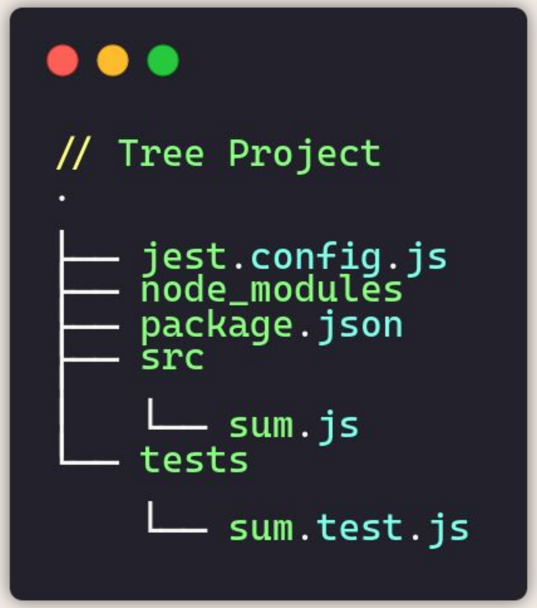
```
$ npm test -- --init
```

Few CLI Options



```
# Contains the chain or specific file
$ npm test my-test #or
$ npm test path/to/my-test.js

# Run only uncommited tests
$npm test -- -o
```

Jest Project: Directory Structure

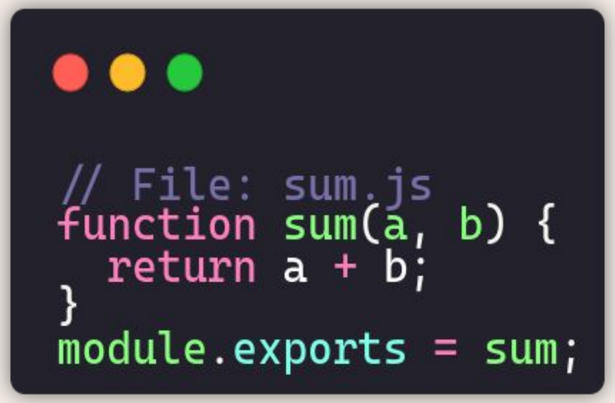


```
// Tree Project
.
├── jest.config.js
├── node_modules
├── package.json
├── src
│   └── sum.js
└── tests
    └── sum.test.js
```

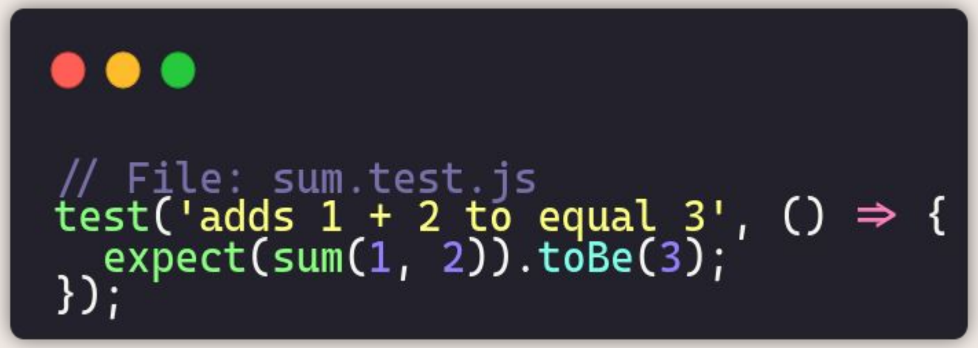
03

Topics

Jest Basics

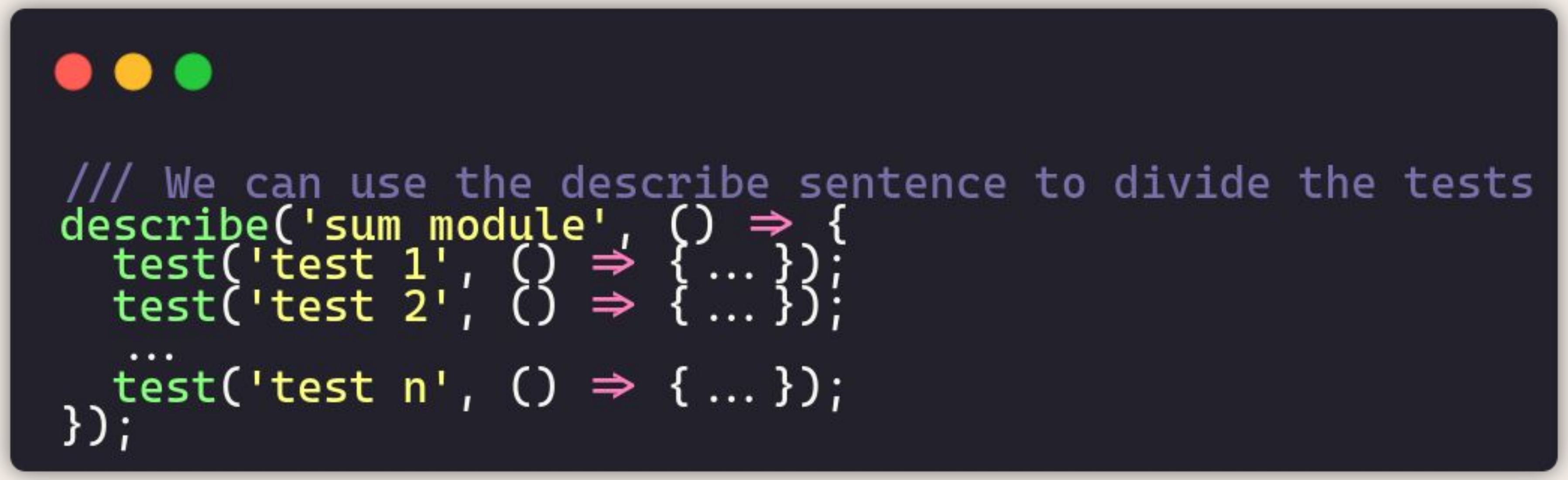


```js
// File: sum.js
function sum(a, b) {
  return a + b;
}
module.exports = sum;
```



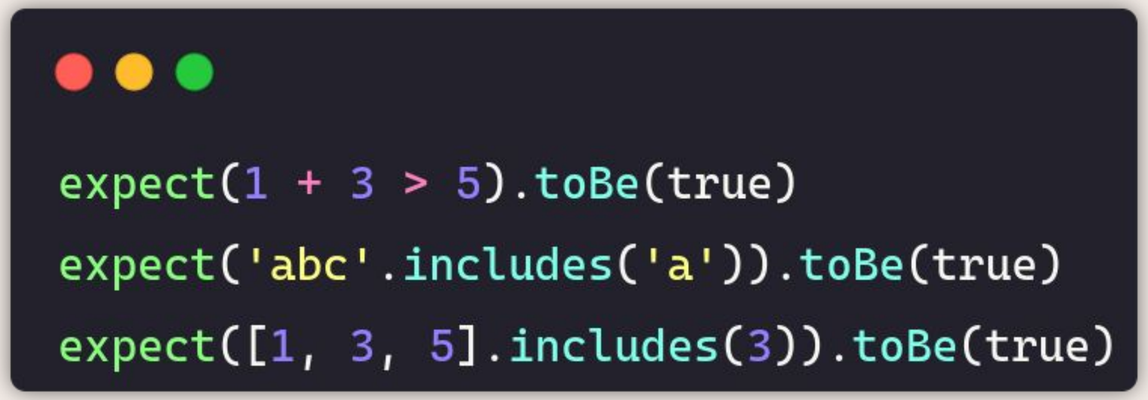
```js
// File: sum.test.js
test('adds 1 + 2 to equal 3', () => {
  expect(sum(1, 2)).toBe(3);
});
```

Basics: Group Tests



```
/// We can use the describe sentence to divide the tests
describe('sum module', () => {
  test('test 1', () => {...});
  test('test 2', () => {...});
  ...
  test('test n', () => {...});
});
```

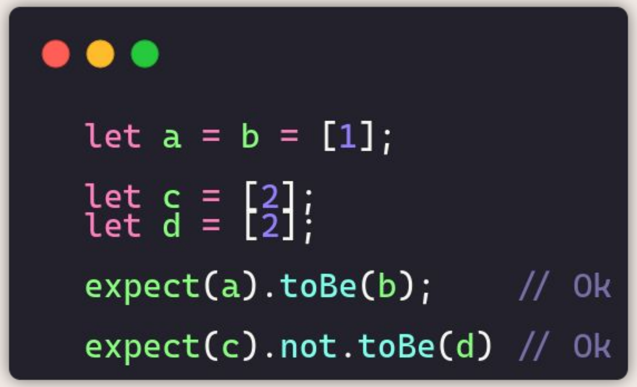
Jest Matchers



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

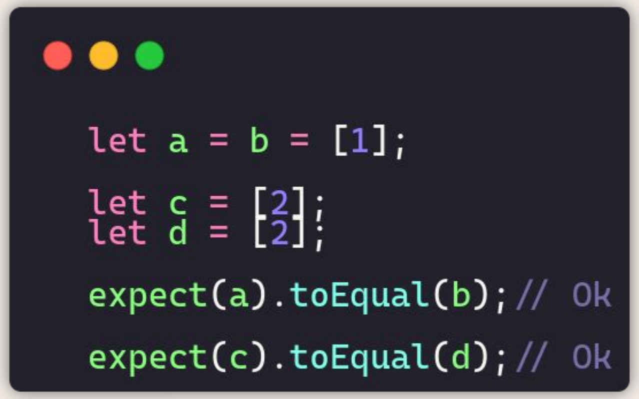
Matchers: Most Common

ToBe



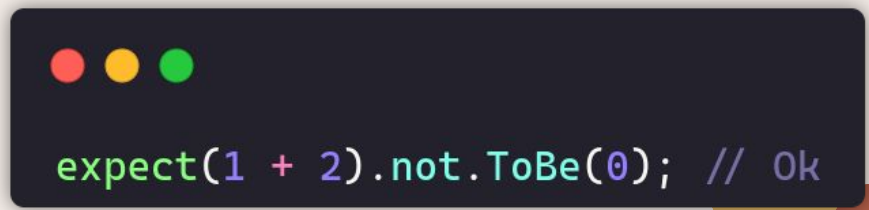
```
let a = b = [1];

let c = [2];
let d = [2];

expect(a).toBe(b);      // Ok

expect(c).not.toBe(d)  // Ok
```

ToEqual



```
let a = b = [1];

let c = [2];
let d = [2];

expect(a).toEqual(b);// Ok

expect(c).toEqual(d);// Ok
```

Not



```
expect(1 + 2).not.ToBe(0);  // Ok
```

Matchers: Truthiness

toBeNull

Check if the value is **null.**

toBeDefined

Check if the value is **defined.**

toBeUndefined

Check if the value is **undefined..**

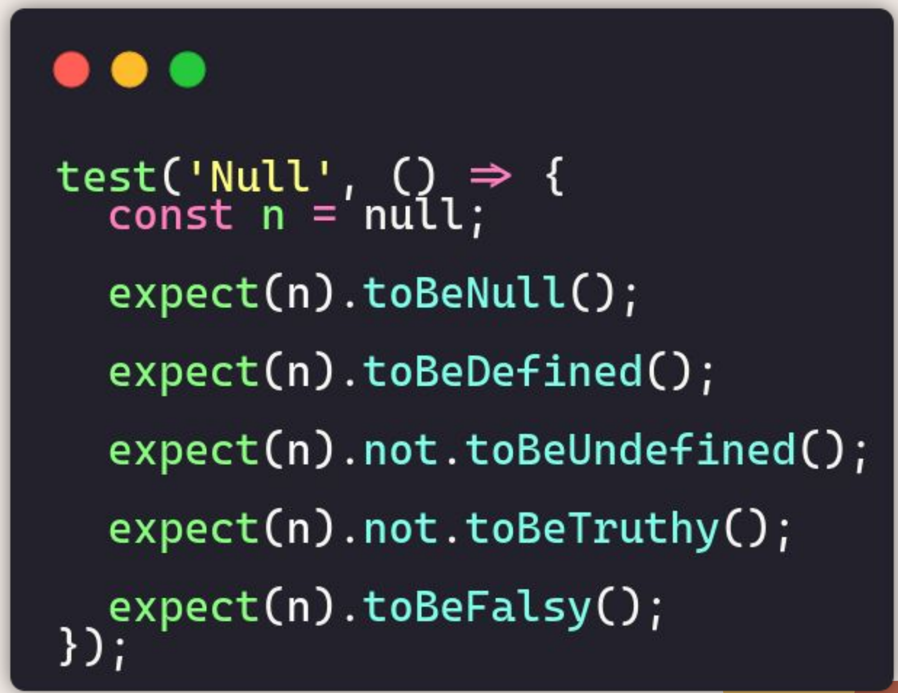
toBeTruthy

Check if the value is **true.**

toBeFalsy

Check if the value is **false.**

Matchers: Truthiness



```
test('Null', () => {
  const n = null;

  expect(n).toBeNull();

  expect(n).toBeDefined();

  expect(n).not.toBeUndefined();

  expect(n).not.toBeTruthy();

  expect(n).toBeFalsy();
});
```

Matchers: Numbers

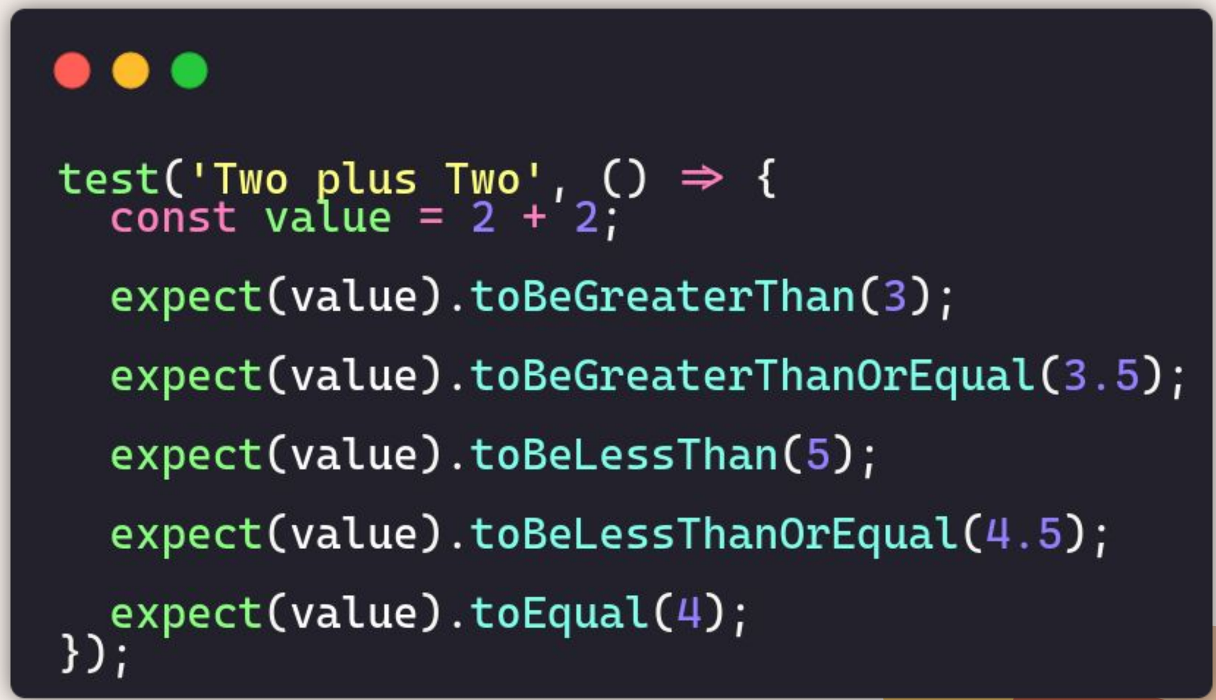
toBeGreaterThan
Check if the value is **bigger.**

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.**

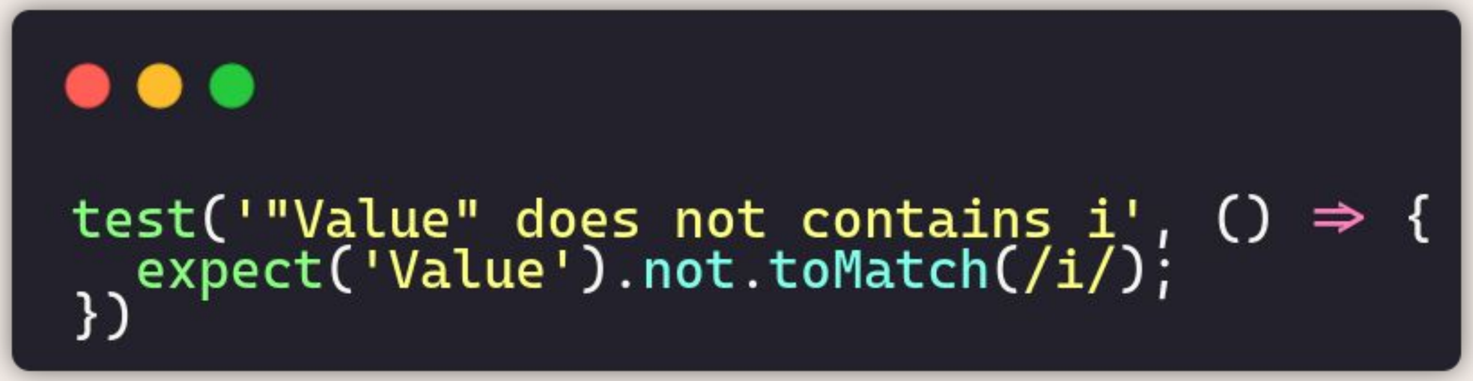
Matchers: Numbers



```
test('Two plus Two', () => {
  const value = 2 + 2;

  expect(value).toBeGreaterThan(3);

  expect(value).toBeGreaterThanOrEqual(3.5);

  expect(value).toBeLessThan(5);

  expect(value).toBeLessThanOrEqual(4.5);

  expect(value).toEqual(4);
});
```

Matchers: Strings

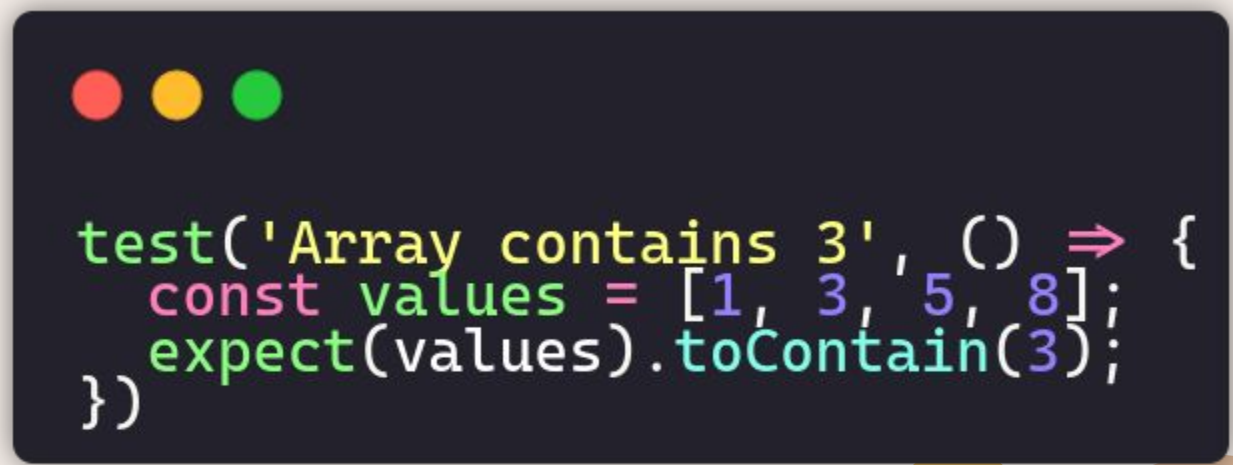
Check Strings with Regex



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

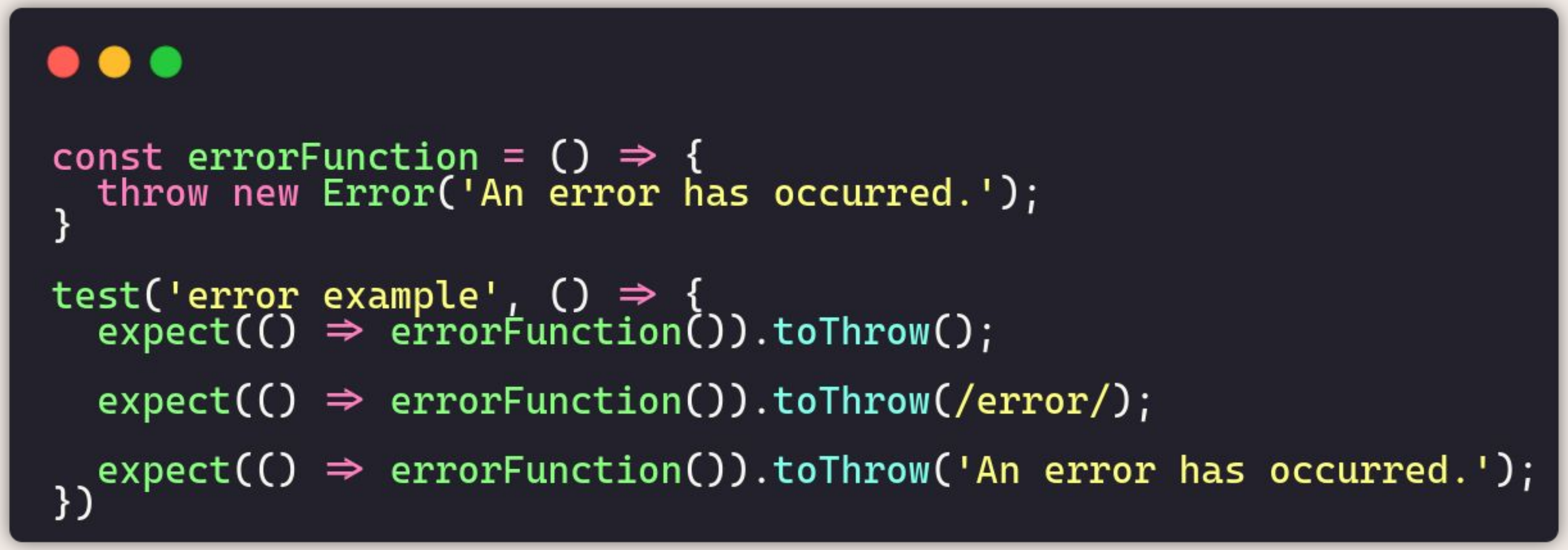
Matchers: Array and Iterables

Check if contains a value



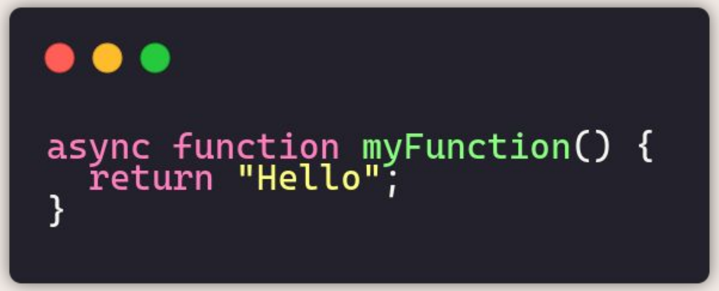
```
test('Array contains 3', () => {
   const values = [1, 3, 5, 8];
   expect(values).toContain(3);
})
```

Matchers: Exceptions



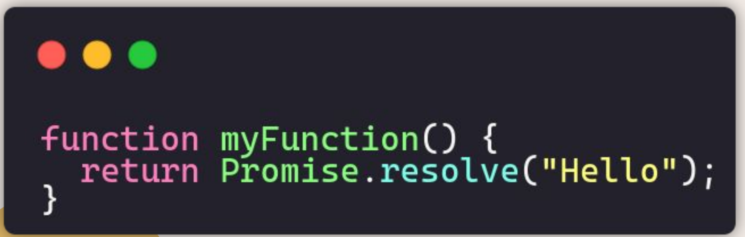
```
const errorFunction = () => {
  throw new Error('An error has occurred.');
}

test('error example', () => {
  expect(() => errorFunction()).toThrow();

  expect(() => errorFunction()).toThrow(/error/);

  expect(() => errorFunction()).toThrow('An error has occurred.');
})
```

Jest Async

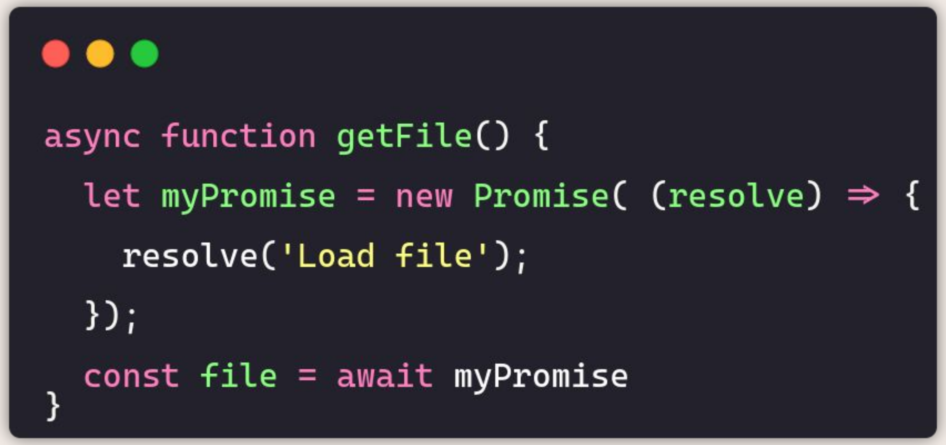


```
async function myFunction() {
  return "Hello";
}
```

Is the same as:

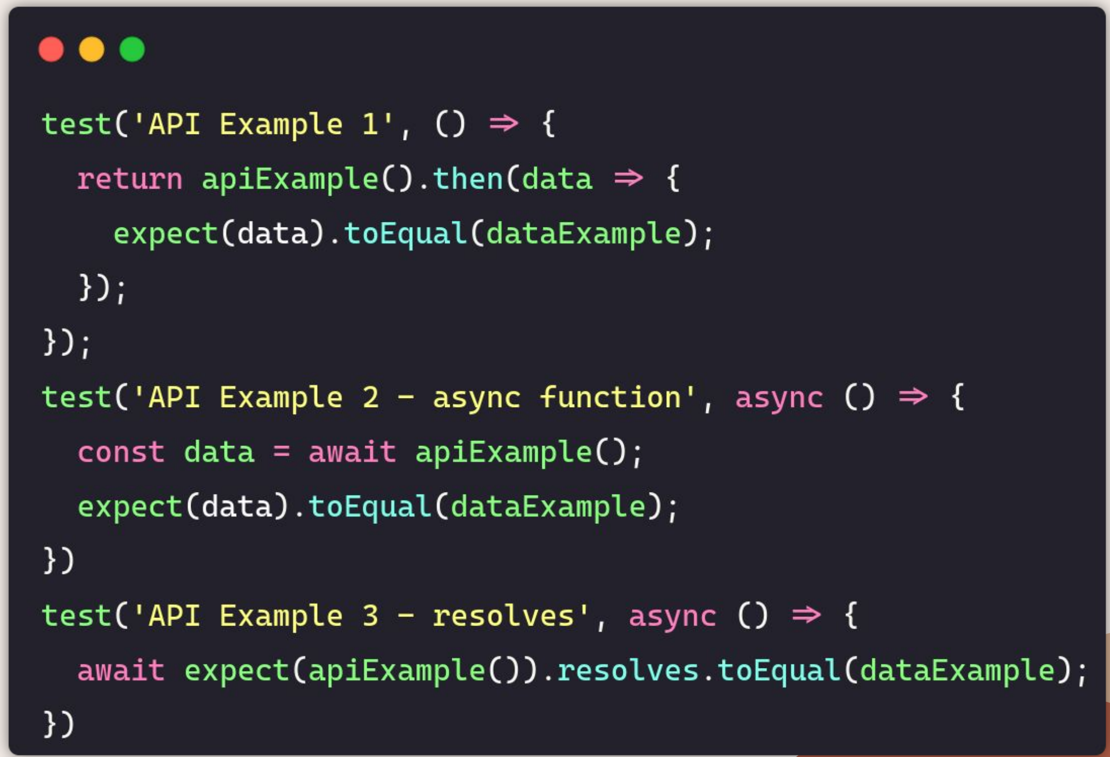


```
function myFunction() {
  return Promise.resolve("Hello");
}
```



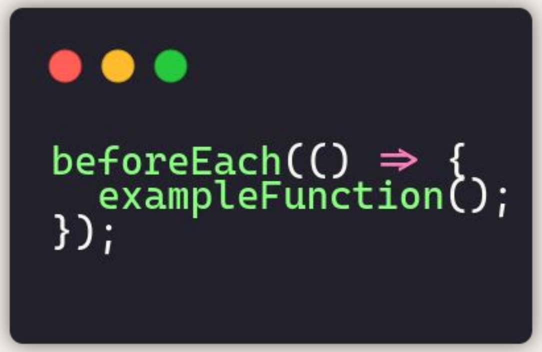
```
async function getFile() {
  let myPromise = new Promise( (resolve) ⇒ {
    resolve('Load file');
  });
  const file = await myPromise
}
```

Async: 3 Ways

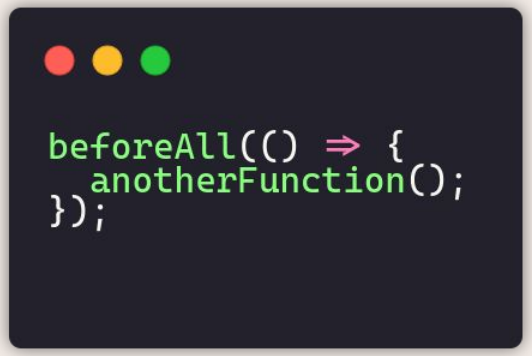


```
test('API Example 1', () => {
  return apiExample().then(data => {
    expect(data).toEqual(dataExample);
  });
});
test('API Example 2 - async function', async () => {
  const data = await apiExample();
  expect(data).toEqual(dataExample);
})
test('API Example 3 - resolves', async () => {
  await expect(apiExample()).resolves.toEqual(dataExample);
})
```

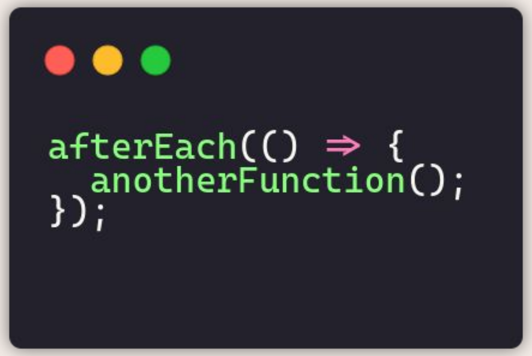
Setup and teardown



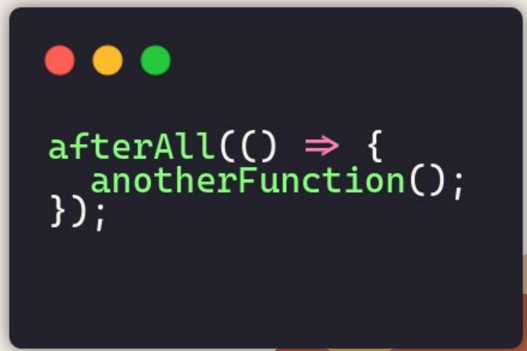
```
beforeEach(() => {
  exampleFunction();
});
```



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

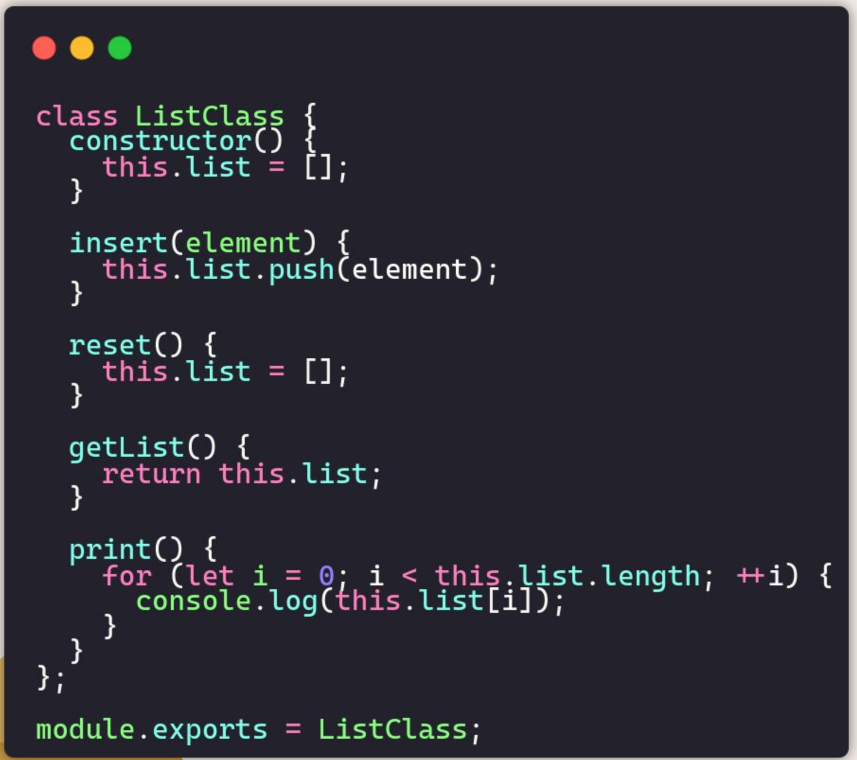


```
afterEach(() => {
  anotherFunction();
});
```

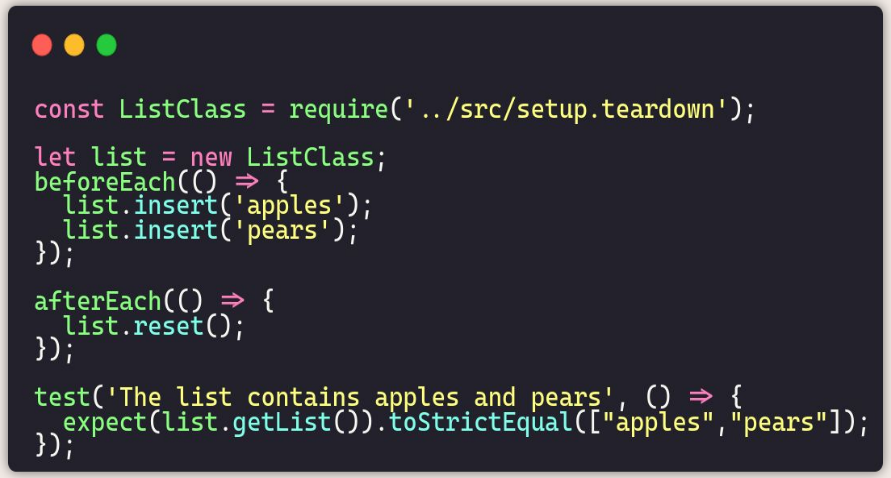


```
afterAll(() => {
  anotherFunction();
});
```

Basic Setup and teardown

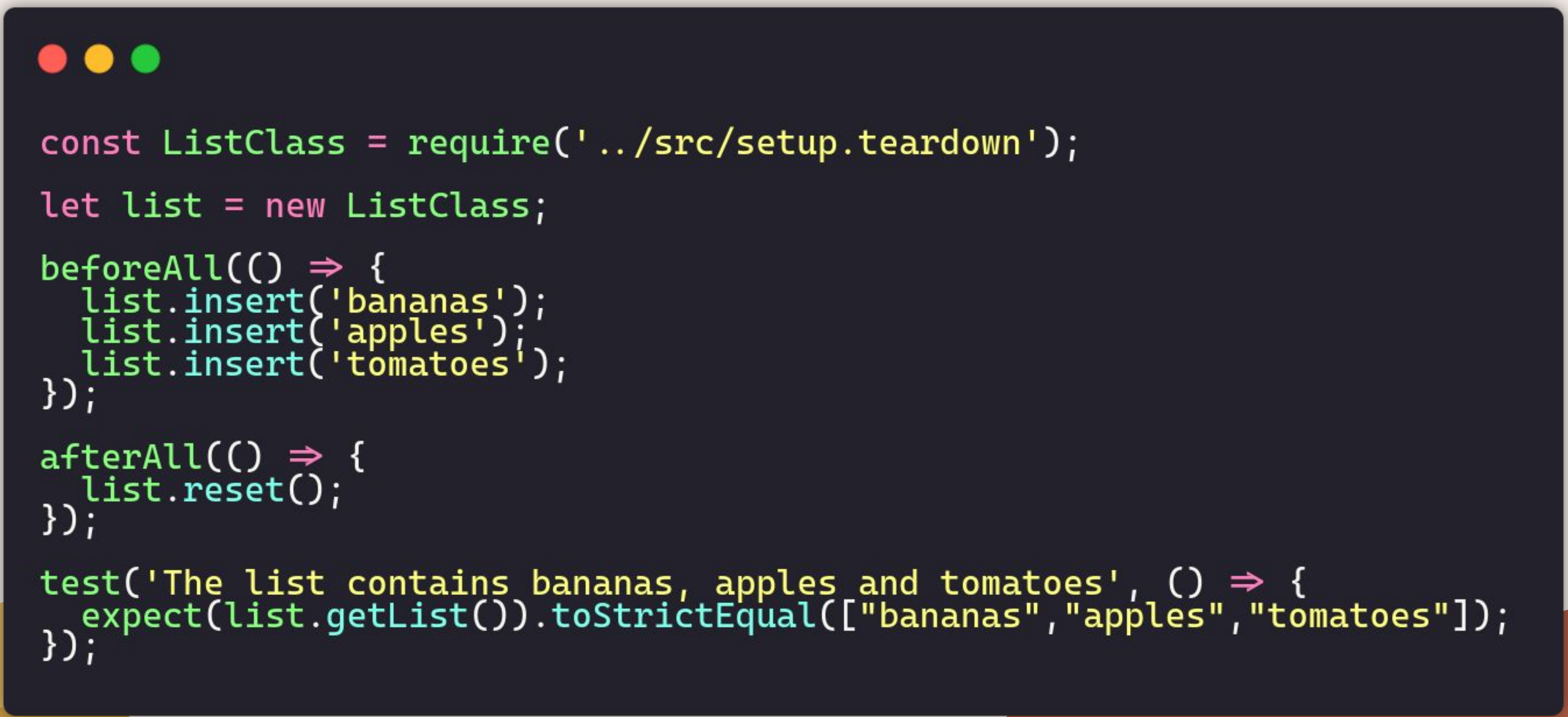


```
class ListClass {
  constructor() {
    this.list = [];
  }

  insert(element) {
    this.list.push(element);
  }

  reset() {
    this.list = [];
  }

  getList() {
    return this.list;
  }

  print() {
    for (let i = 0; i < this.list.length; ++i) {
      console.log(this.list[i]);
    }
  }
};

module.exports = ListClass;
```



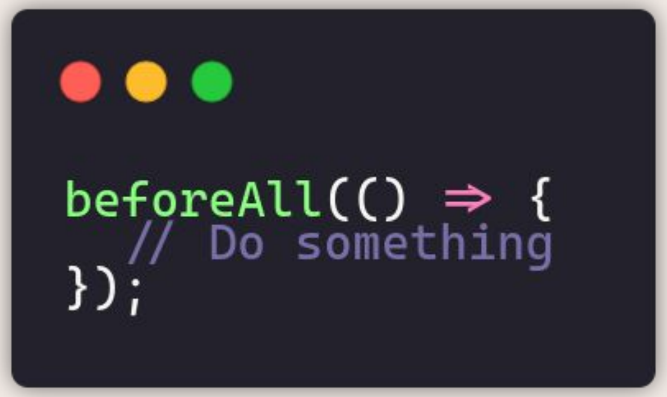
```
const ListClass = require('../src/setup.teardown');

let list = new ListClass;
beforeEach(() => {
  list.insert('apples');
  list.insert('pears');
});

afterEach(() => {
  list.reset();
});

test('The list contains apples and pears', () => {
  expect(list.getList()).toStrictEqual(["apples","pears"]);
});
```

The use of before and after all

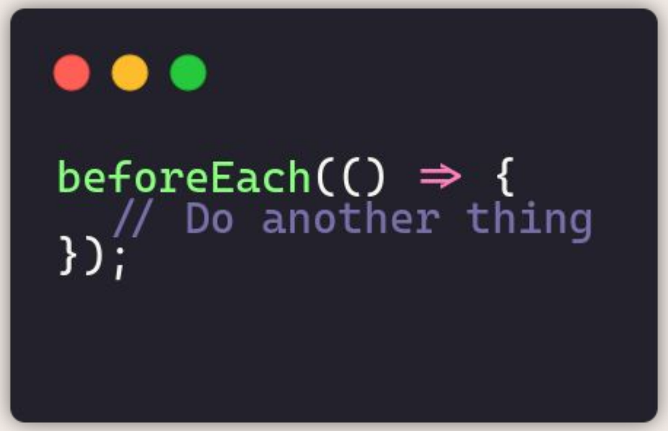


```
const ListClass = require('../src/setup.teardown');

let list = new ListClass;

beforeAll(() => {
  list.insert('bananas');
  list.insert('apples');
  list.insert('tomatoes');
});

afterAll(() => {
  list.reset();
});

test('The list contains bananas, apples and tomatoes', () => {
  expect(list.getList()).toStrictEqual(["bananas","apples","tomatoes"]);
});
```

¿Before each or before all?

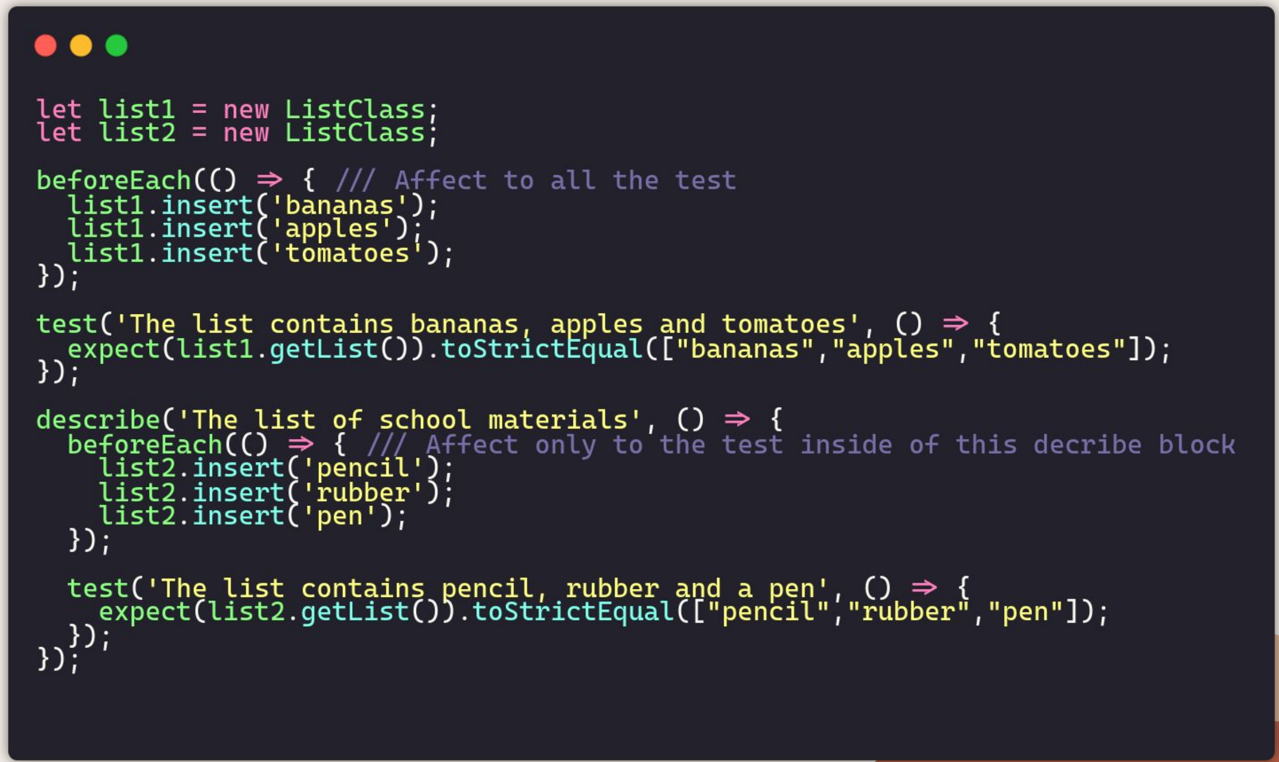


```
beforeAll(() => {
   // Do something
});
```



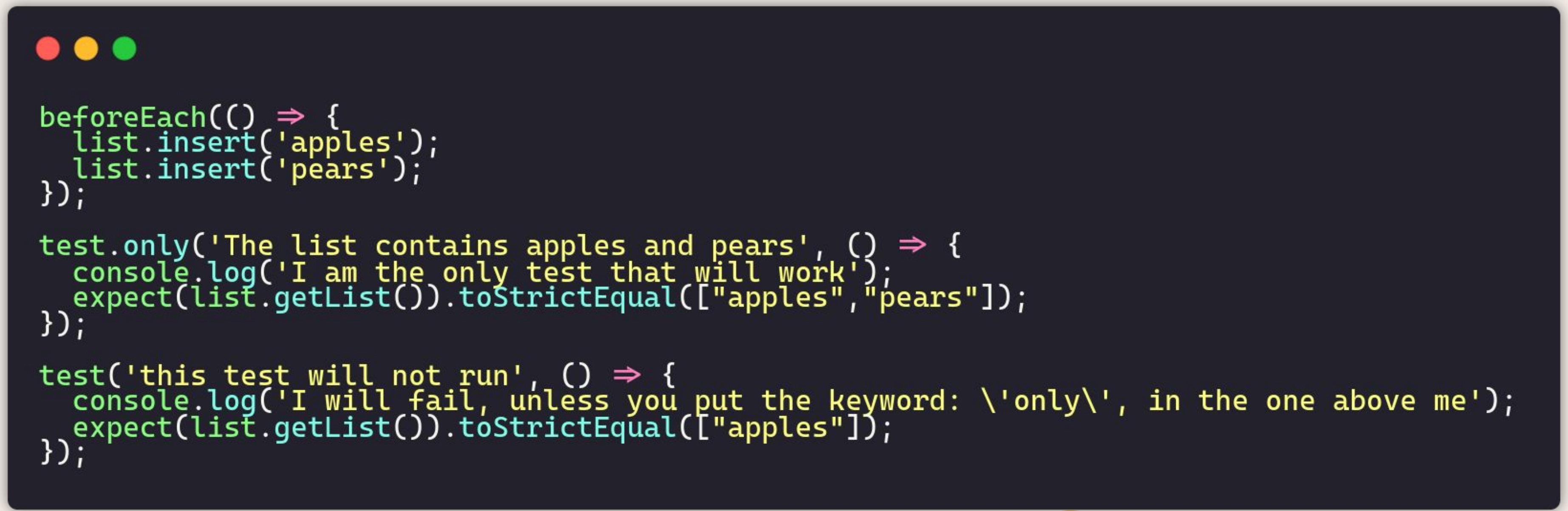
```
beforeEach(() => {
   // Do another thing
});
```

Scoping



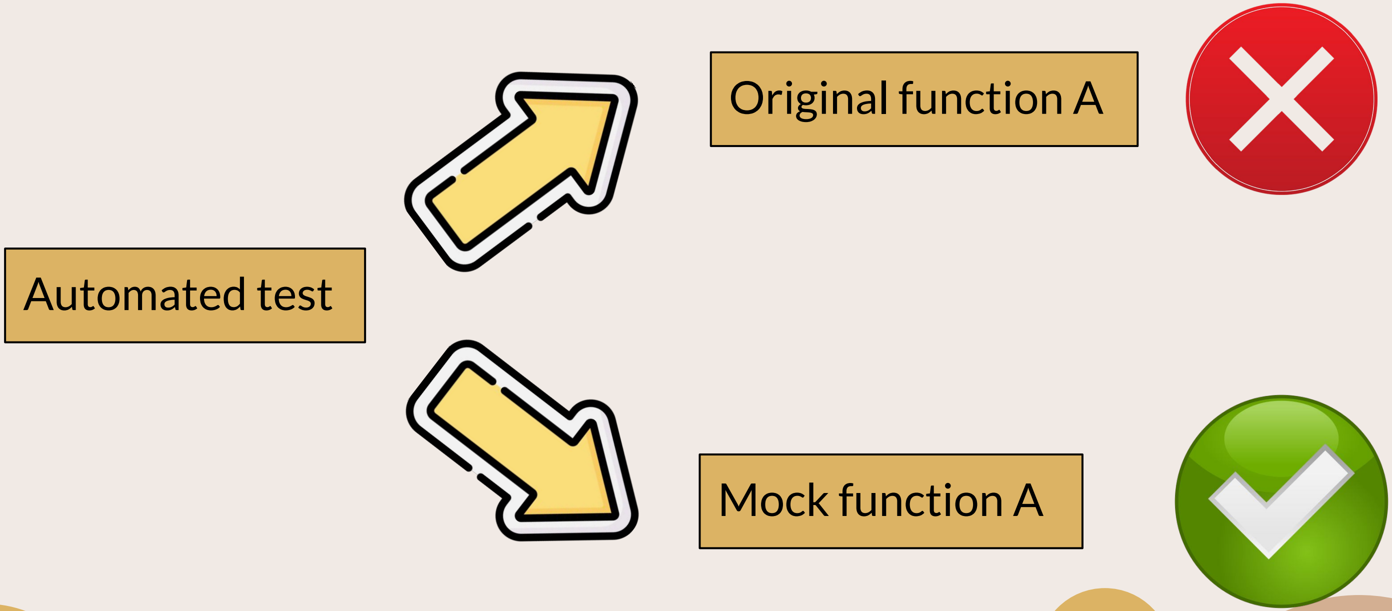
```
let list1 = new ListClass;
let list2 = new ListClass;

beforeEach(() => { /// Affect to all the test
  list1.insert('bananas');
  list1.insert('apples');
  list1.insert('tomatoes');
});

test('The list contains bananas, apples and tomatoes', () => {
  expect(list1.getList()).toStrictEqual(["bananas","apples","tomatoes"]);
});

describe('The list of school materials', () => {
  beforeEach(() => { /// Affect only to the test inside of this decribe block
    list2.insert('pencil');
    list2.insert('rubber');
    list2.insert('pen');
  });

  test('The list contains pencil, rubber and a pen', () => {
    expect(list2.getList()).toStrictEqual(["pencil","rubber","pen"]);
  });
});
```

The use of only

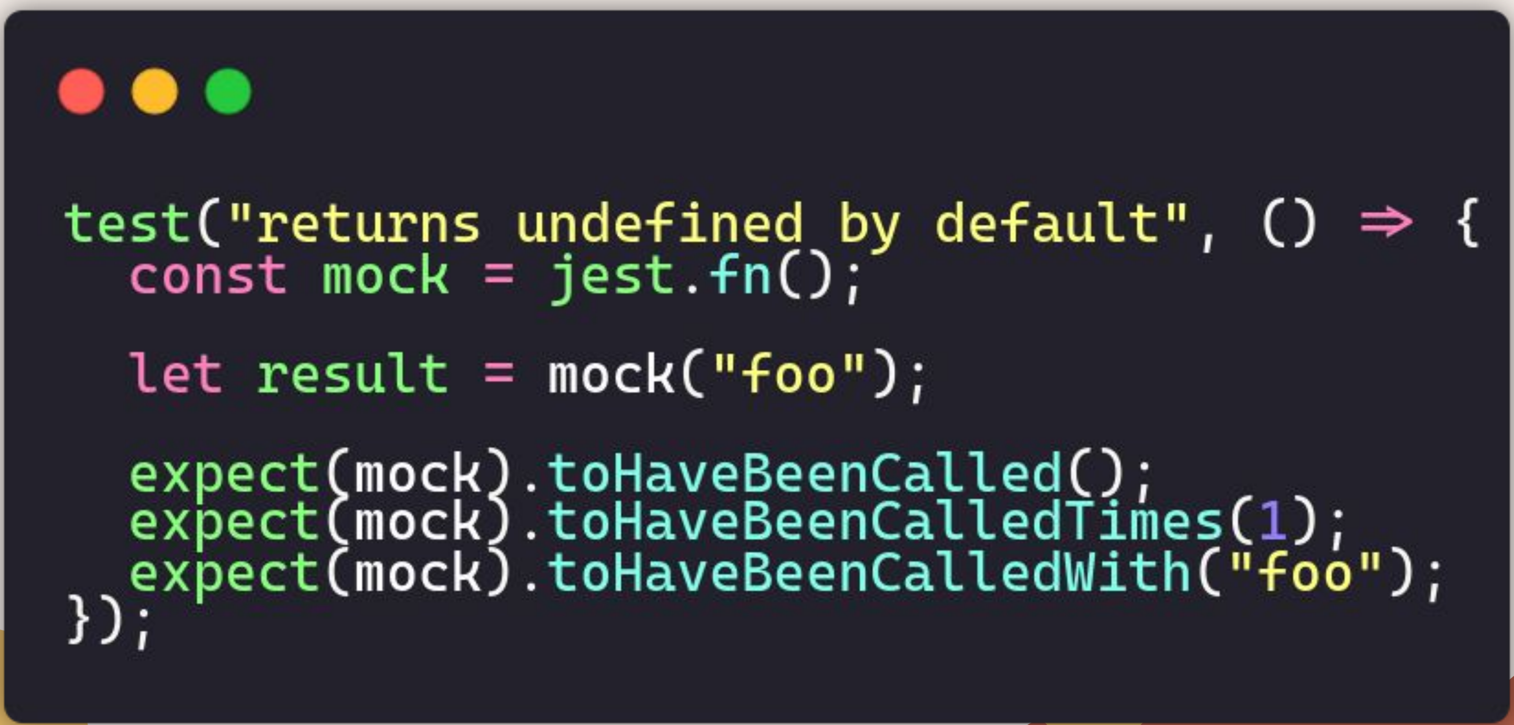


```
beforeEach(() => {
  list.insert('apples');
  list.insert('pears');
});

test.only('The list contains apples and pears', () => {
  console.log('I am the only test that will work');
  expect(list.getList()).toStrictEqual(["apples","pears"]);
});

test('this test will not run', () => {
  console.log('I will fail, unless you put the keyword: \'only\', in the one above me');
  expect(list.getList()).toStrictEqual(["apples"]);
});
```

Mock functions

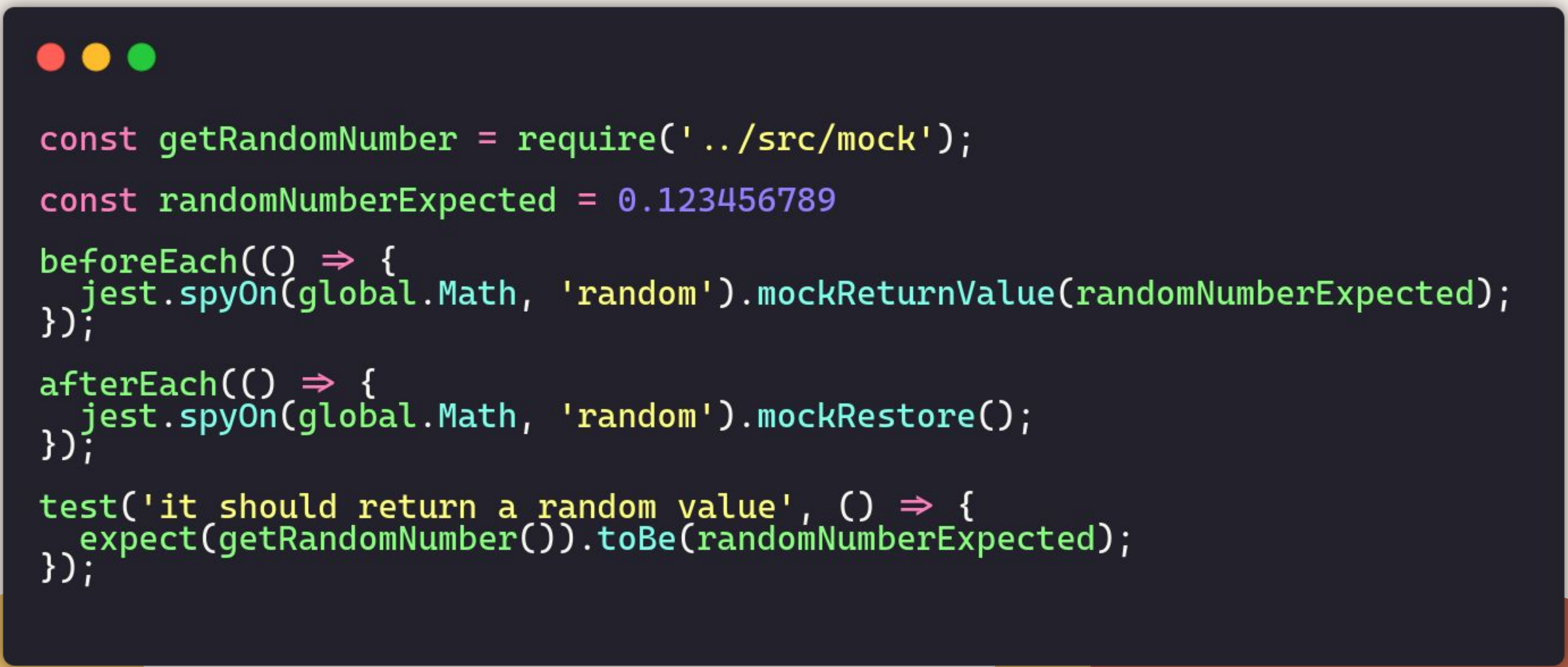


Basic use of mock



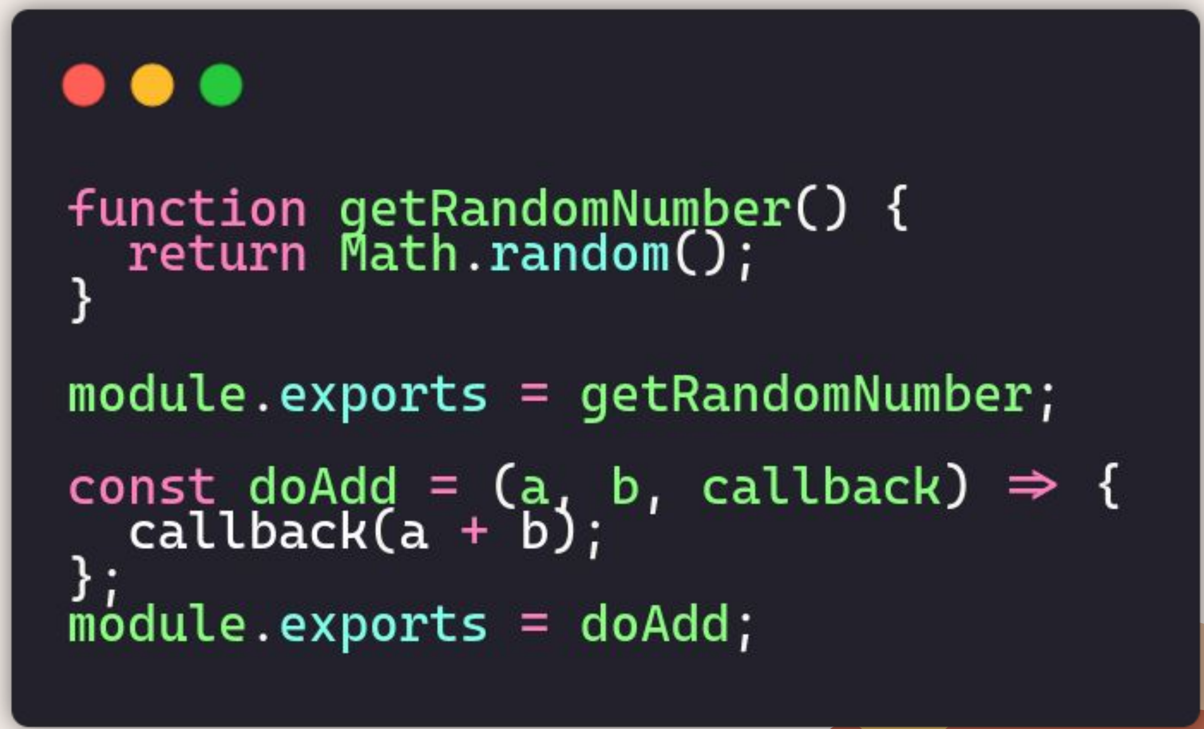
```
test("returns undefined by default", () => {
  const mock = jest.fn();

  let result = mock("foo");

  expect(mock).toHaveBeenCalled();
  expect(mock).toHaveBeenCalledTimes(1);
  expect(mock).toHaveBeenCalledWith("foo");
});
```

Manipulation using jest.spyON



```javascript
const getRandomNumber = require('../src/mock');

const randomNumberExpected = 0.123456789

beforeEach(() => {
  jest.spyOn(global.Math, 'random').mockReturnValue(randomNumberExpected);
});

afterEach(() => {
  jest.spyOn(global.Math, 'random').mockRestore();
});

test('it should return a random value', () => {
  expect(getRandomNumber()).toBe(randomNumberExpected);
});
```

Using functions



```
function getRandomNumber() {
  return Math.random();
}

module.exports = getRandomNumber;

const doAdd = (a, b, callback) => {
  callback(a + b);
};
module.exports = doAdd;
```

04

Examples

References

- Jest
- Jest Expect Object
- W3Schools Async
- Mocks functions
- Exercism