✓

TESTING

From 2+2 to fetch

ARIANE JURADO DE BILBAO

wefox GROUP

wallapop

ThoughtWorks

ManoMano

XING

New Relic.

Adevinta

vueling

autentia

holaluz

netcentric
A Cognizant Digital Business

Typeform

GFT

Adaptive

marfeel

WOMEN WHO CODE BARCELONA

BCN ENGINEERING

Agile-Spain

DEVOPS.BARCELONA

joppy

CodelyTV

I'm Ari =)

NURSE
Pediatric ICU and urgencies

EDUCATIONAL PSYCHOLOGIST
Learning disabilities

DEVELOPER
At S|ngular

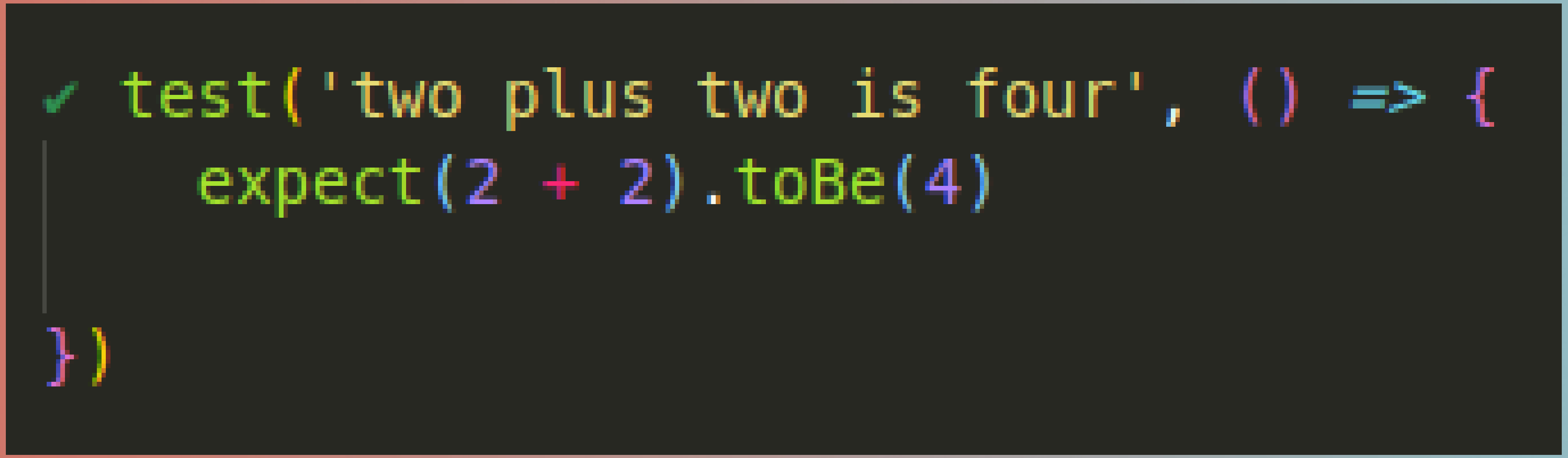
MOM
Of Sebas

@Ari_Reinventada

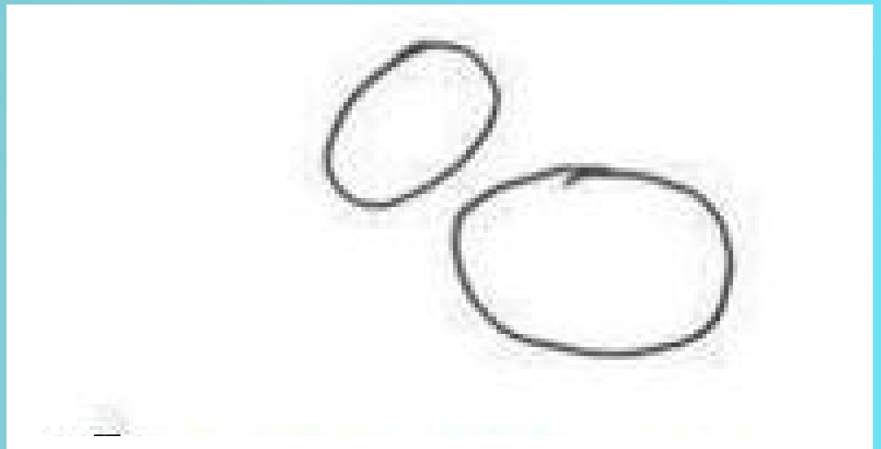
FROM 2+2 TO FETCH

I AM NOT AN EXPERT

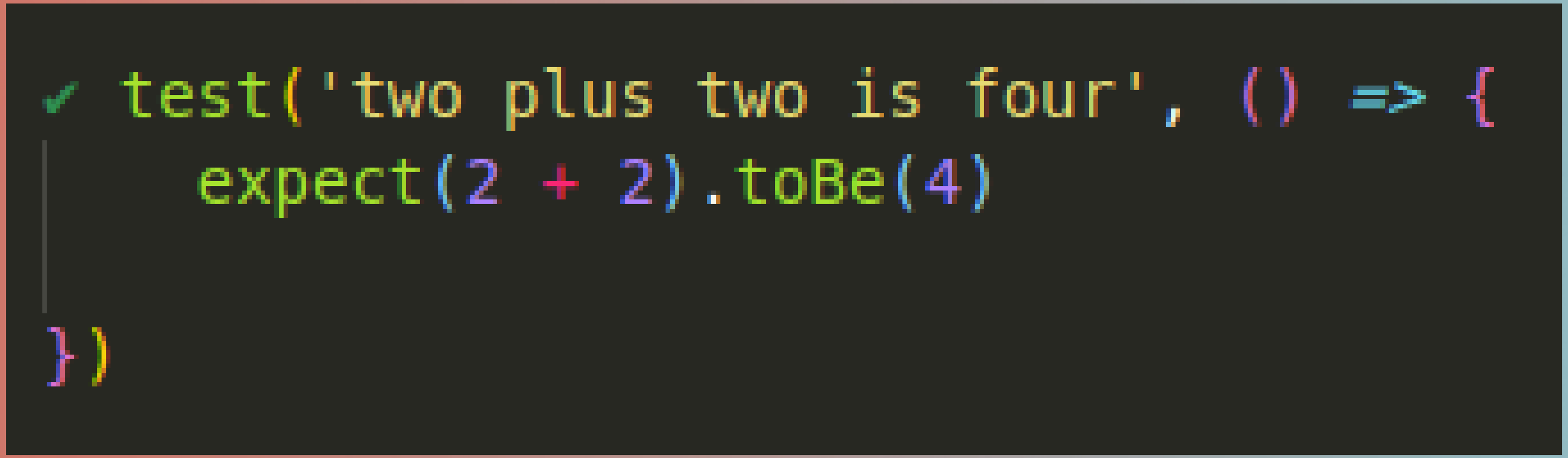
THIS IS MY PROCESS



```
✓ test('two plus two is four', () => {
    expect(2 + 2).toBe(4)

})
```

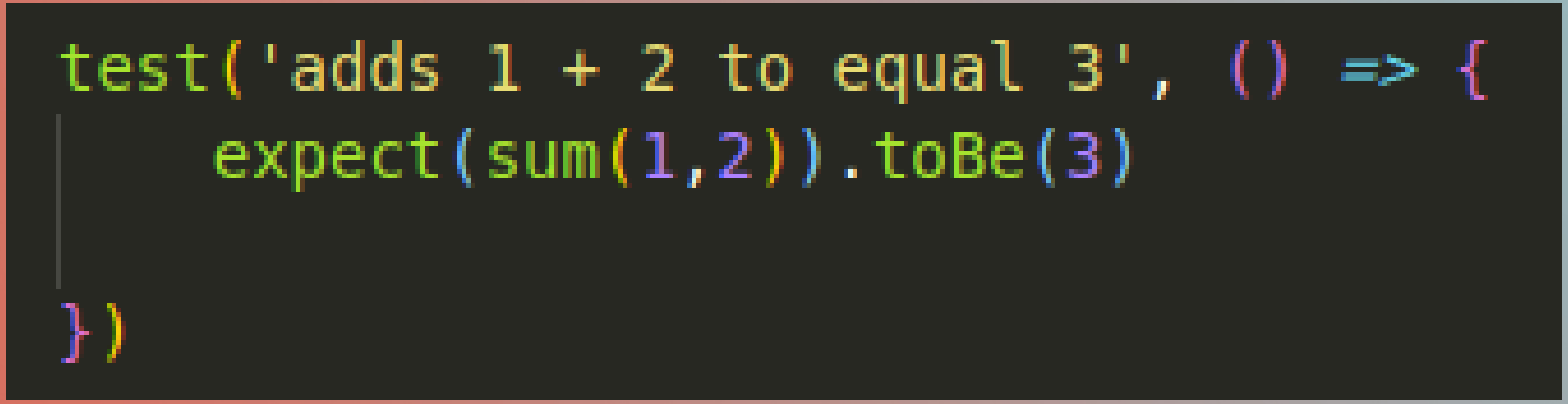


Draw 2 circles

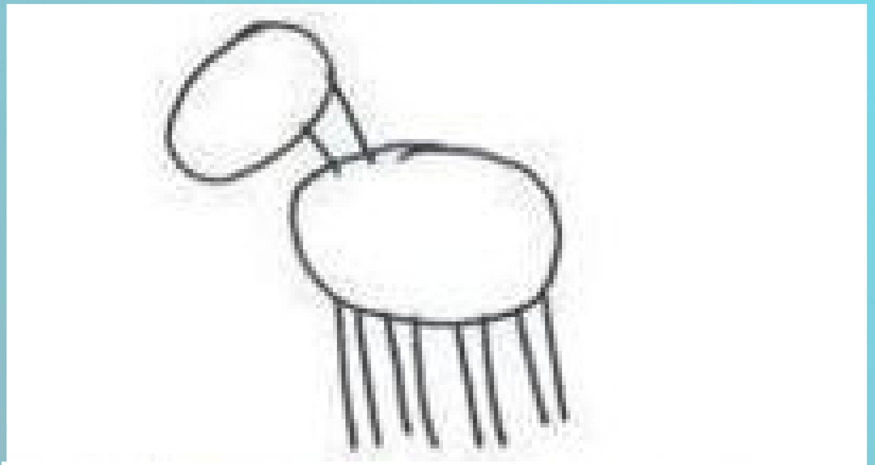


```
✓ test('two plus two is four', () => {
    expect(2 + 2).toBe(4)

})
```

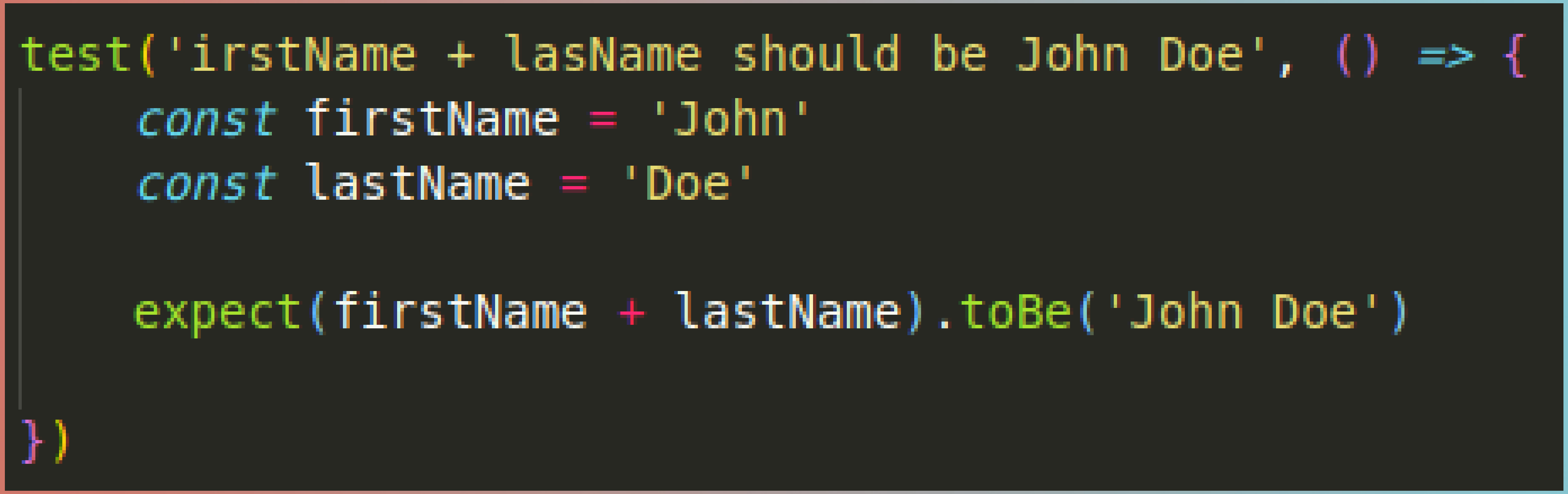
Draw 2 circles



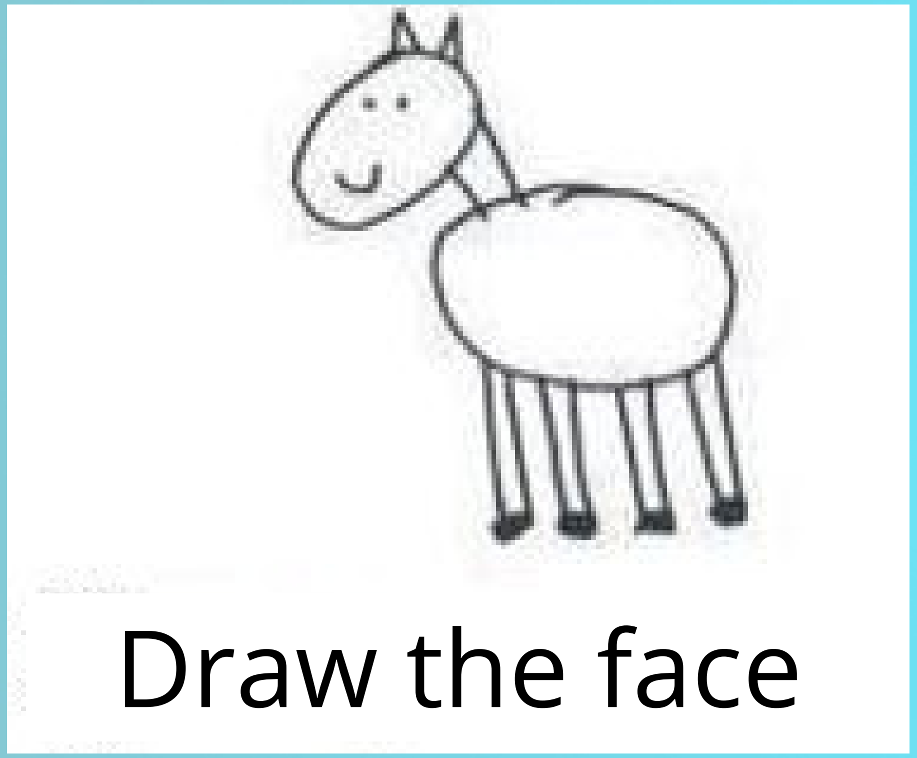
```
test('adds 1 + 2 to equal 3', () => {
    expect(sum(1,2)).toBe(3)

})
```

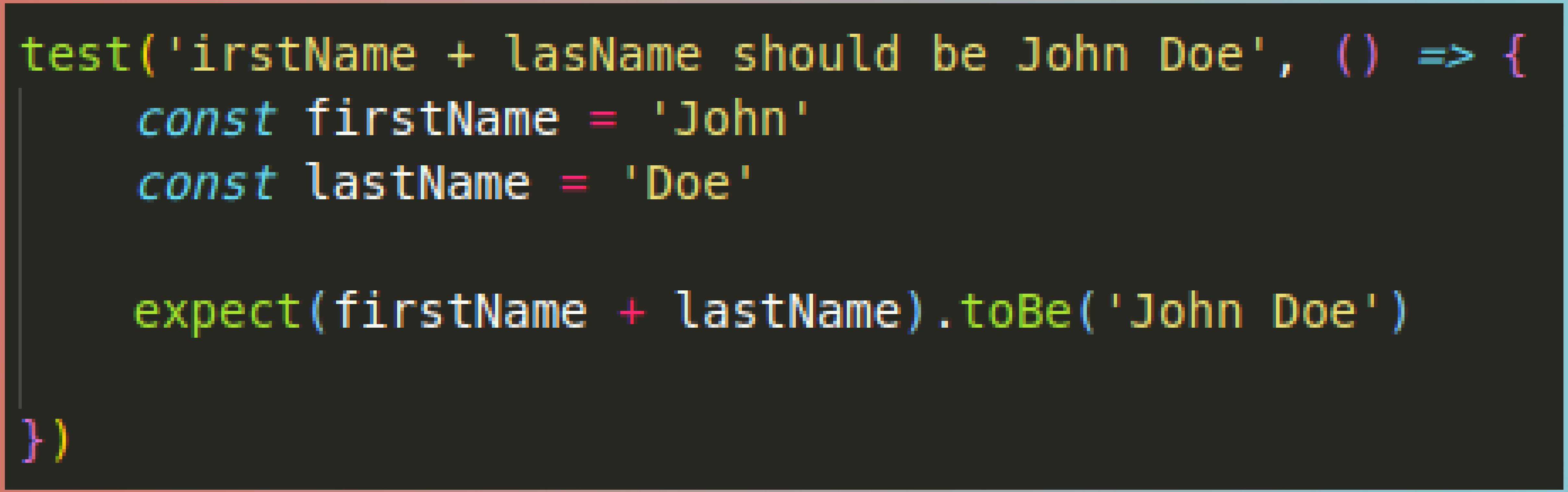


Draw the legs

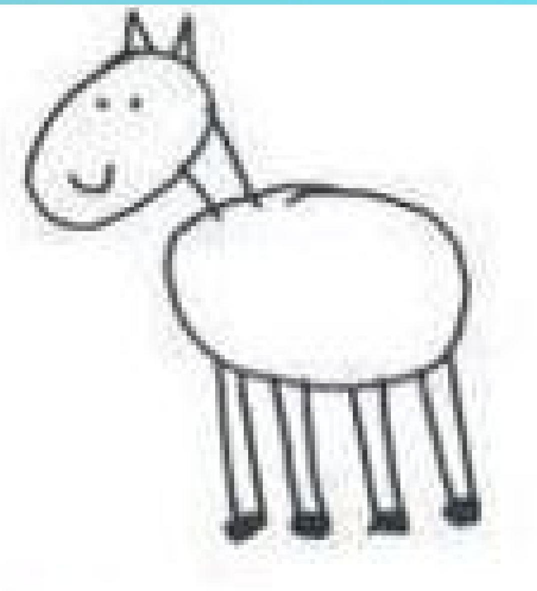


```
test('irstName + lasName should be John Doe', () => {
    const firstName = 'John'
    const lastName = 'Doe'

    expect(firstName + lastName).toBe('John Doe')

})
```

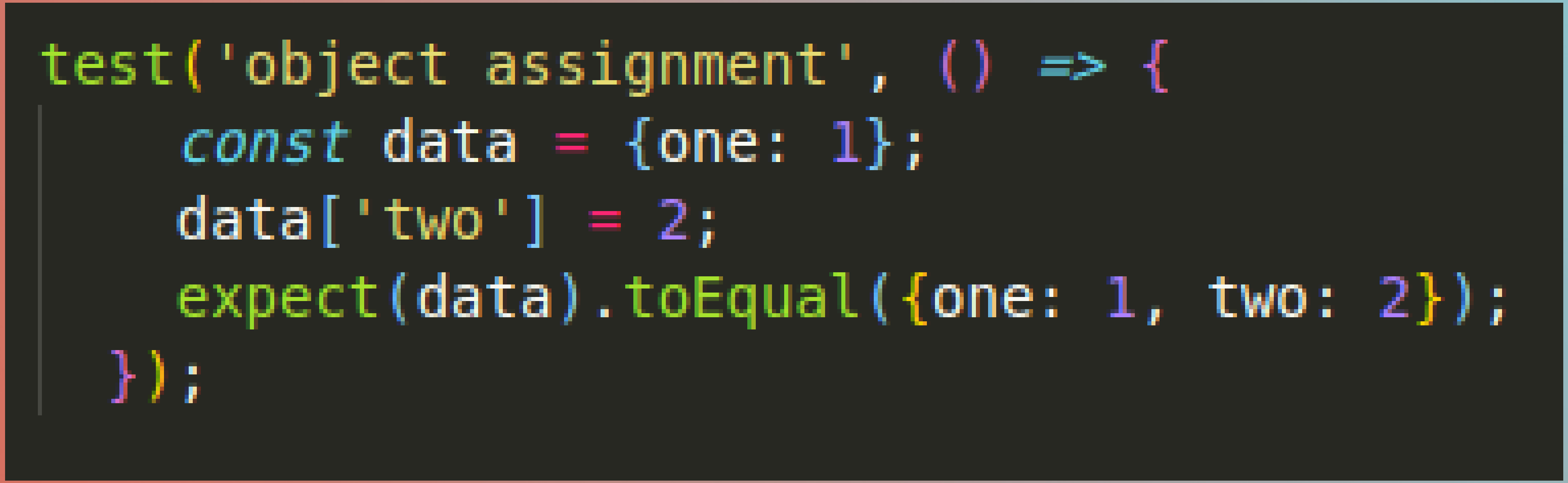




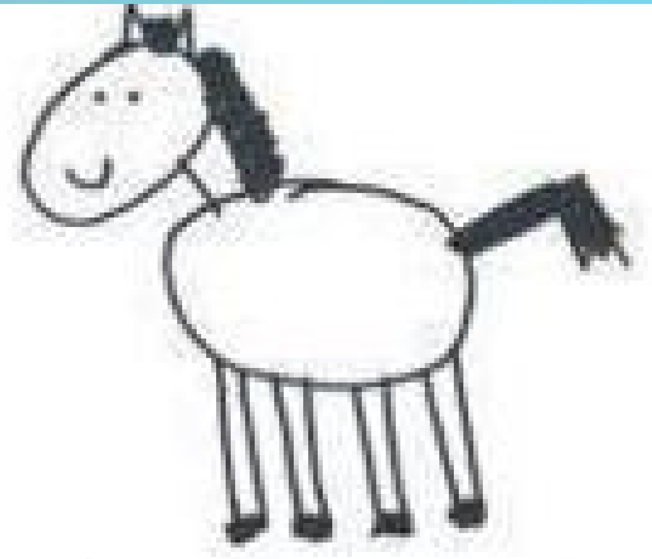
```
test('irstName + lasName should be John Doe', () => {
    const firstName = 'John'
    const lastName = 'Doe'

    expect(firstName + lastName).toBe('John Doe')

})
```



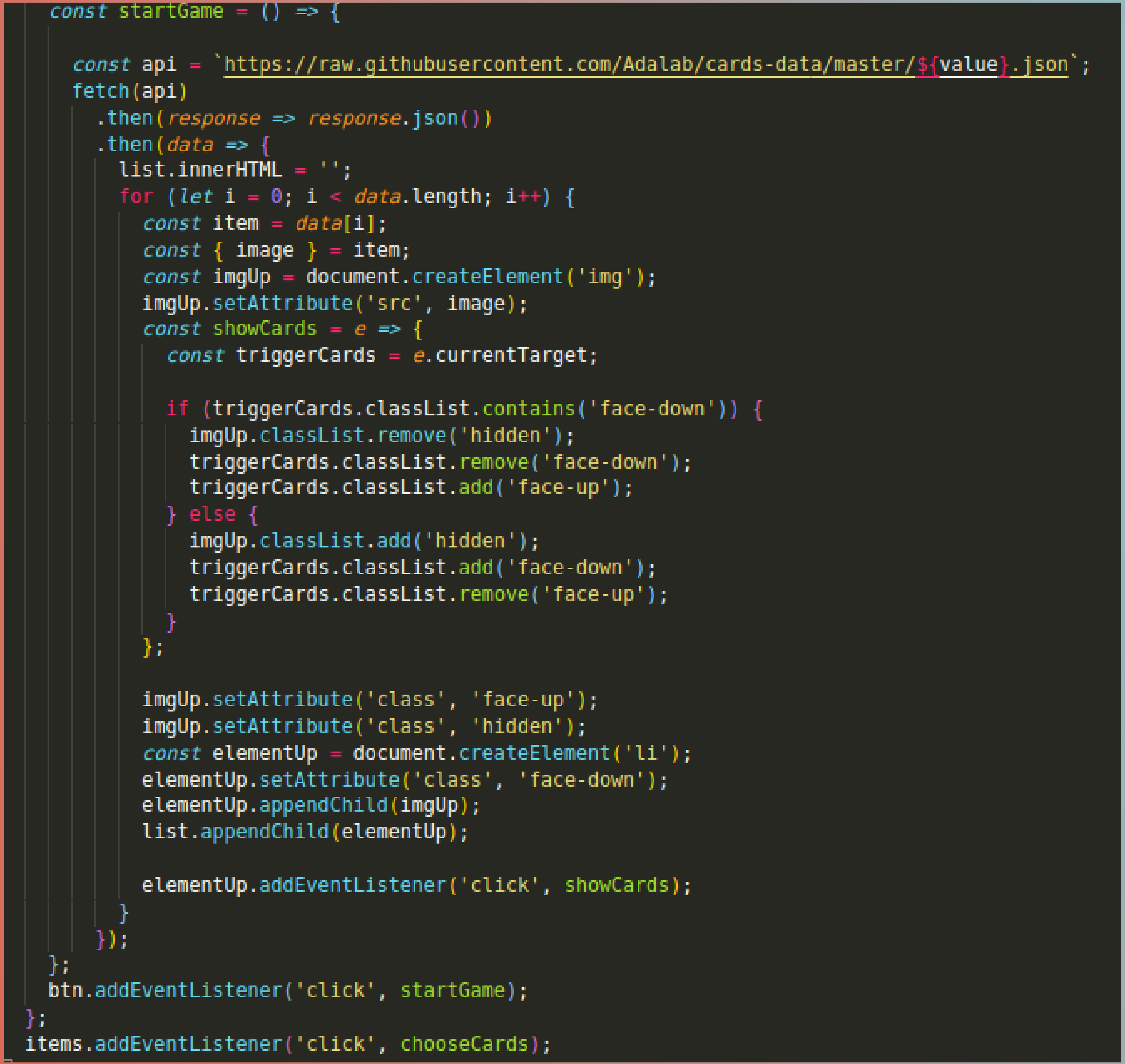
Draw the face



```
test('object assignment', () => {
    const data = {one: 1};
    data['two'] = 2;
    expect(data).toEqual({one: 1, two: 2});
  });
```



Draw the hair



```
    const startGame = () => {

      const api = `https://raw.githubusercontent.com/Adalab/cards-data/master/${value}.json`;
      fetch(api)
        .then(response => response.json())
        .then(data => {
          list.innerHTML = '';
          for (let i = 0; i < data.length; i++) {
            const item = data[i];
            const { image } = item;
            const imgUp = document.createElement('img');
            imgUp.setAttribute('src', image);
            const showCards = e => {
              const triggerCards = e.currentTarget;

              if (triggerCards.classList.contains('face-down')) {
                imgUp.classList.remove('hidden');
                triggerCards.classList.remove('face-down');
                triggerCards.classList.add('face-up');
              } else {
                imgUp.classList.add('hidden');
                triggerCards.classList.add('face-down');
                triggerCards.classList.remove('face-up');
              }
            };

            imgUp.setAttribute('class', 'face-up');
            imgUp.setAttribute('class', 'hidden');
            const elementUp = document.createElement('li');
            elementUp.setAttribute('class', 'face-down');
            elementUp.appendChild(imgUp);
            list.appendChild(elementUp);

            elementUp.addEventListener('click', showCards);
          }
        });
    };
    btn.addEventListener('click', startGame);
  };
  items.addEventListener('click', chooseCards);
```

Add some small details

Some types of test

UNIT

A small part, a basic element, method, function, class, component, module...

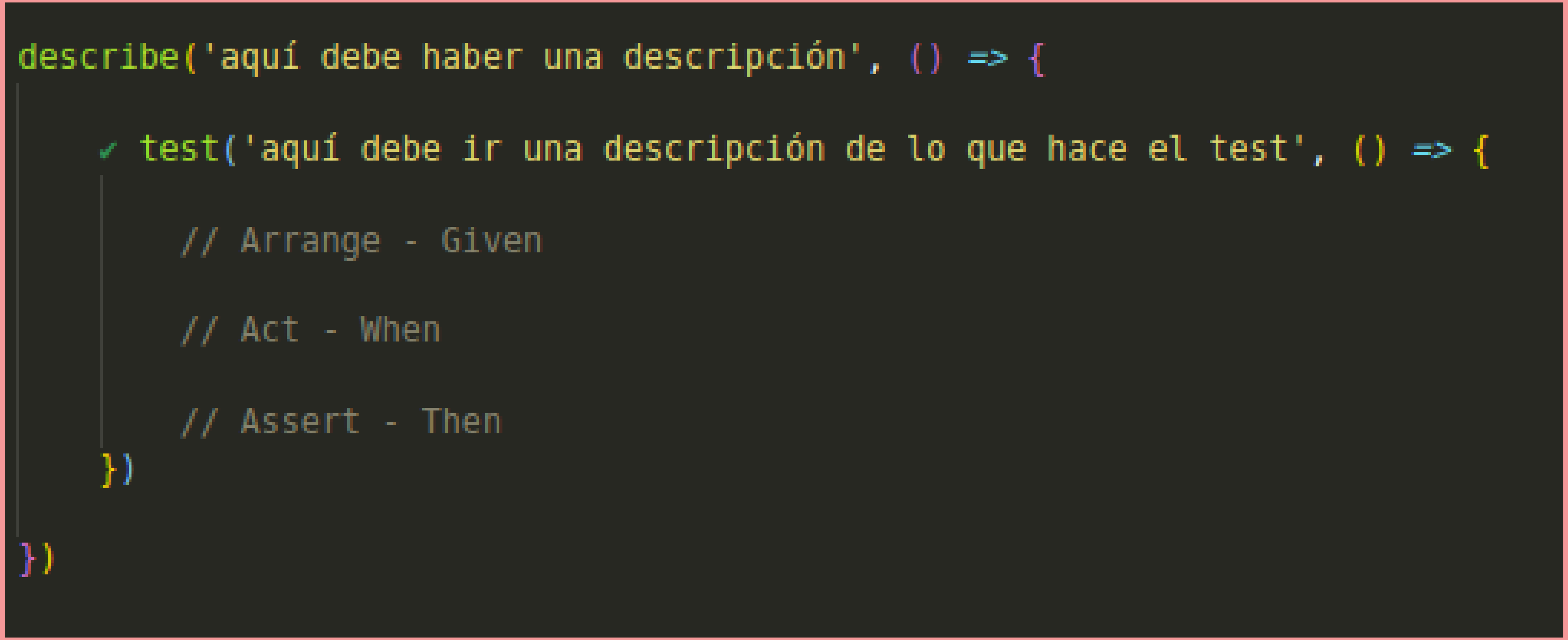
INTEGRATION

A group of elements interacting with each other communication between them.

END TO END

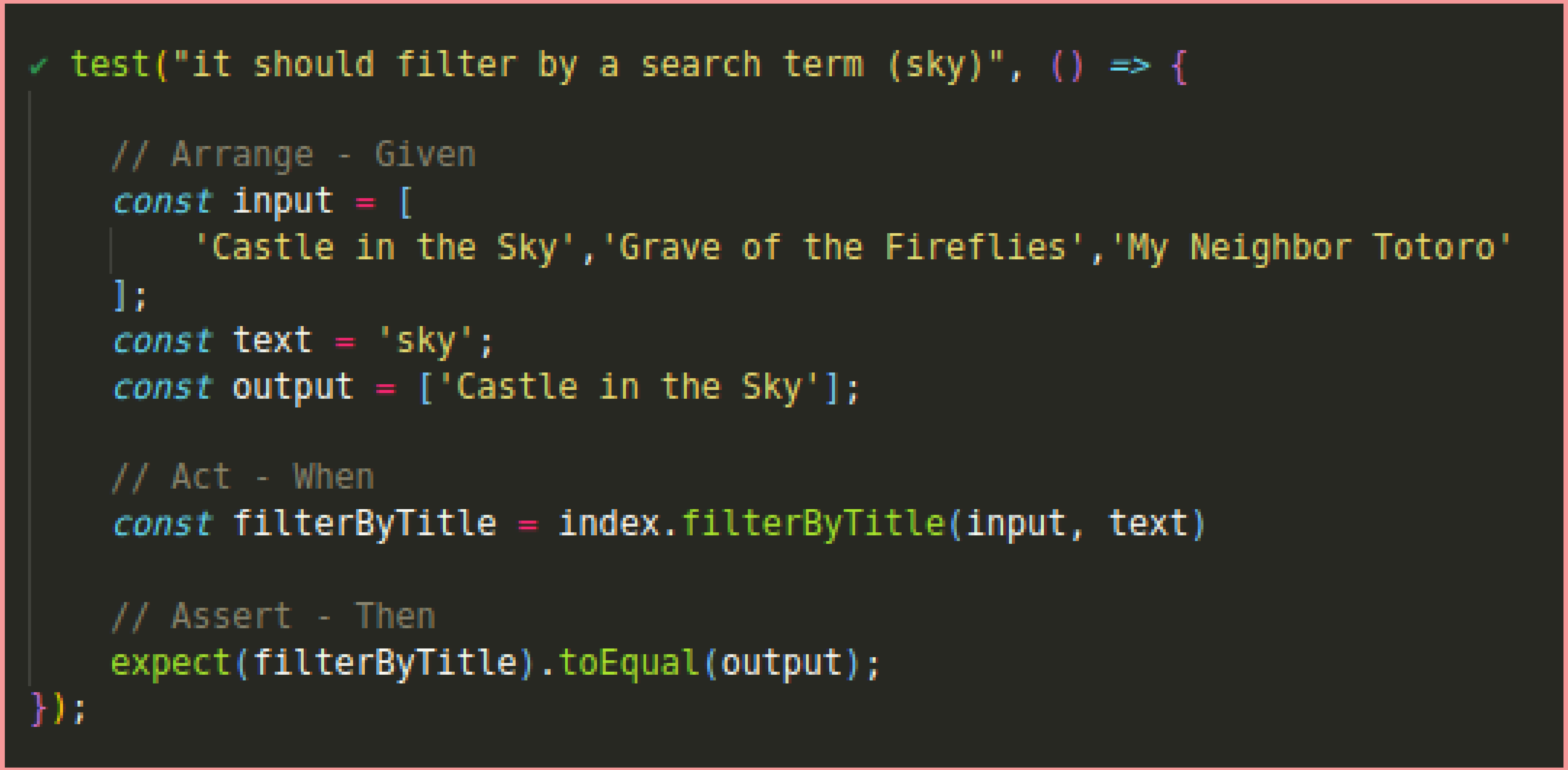
Check if the software behaves as expected. The complete system

HOW TO WRITE A TEST



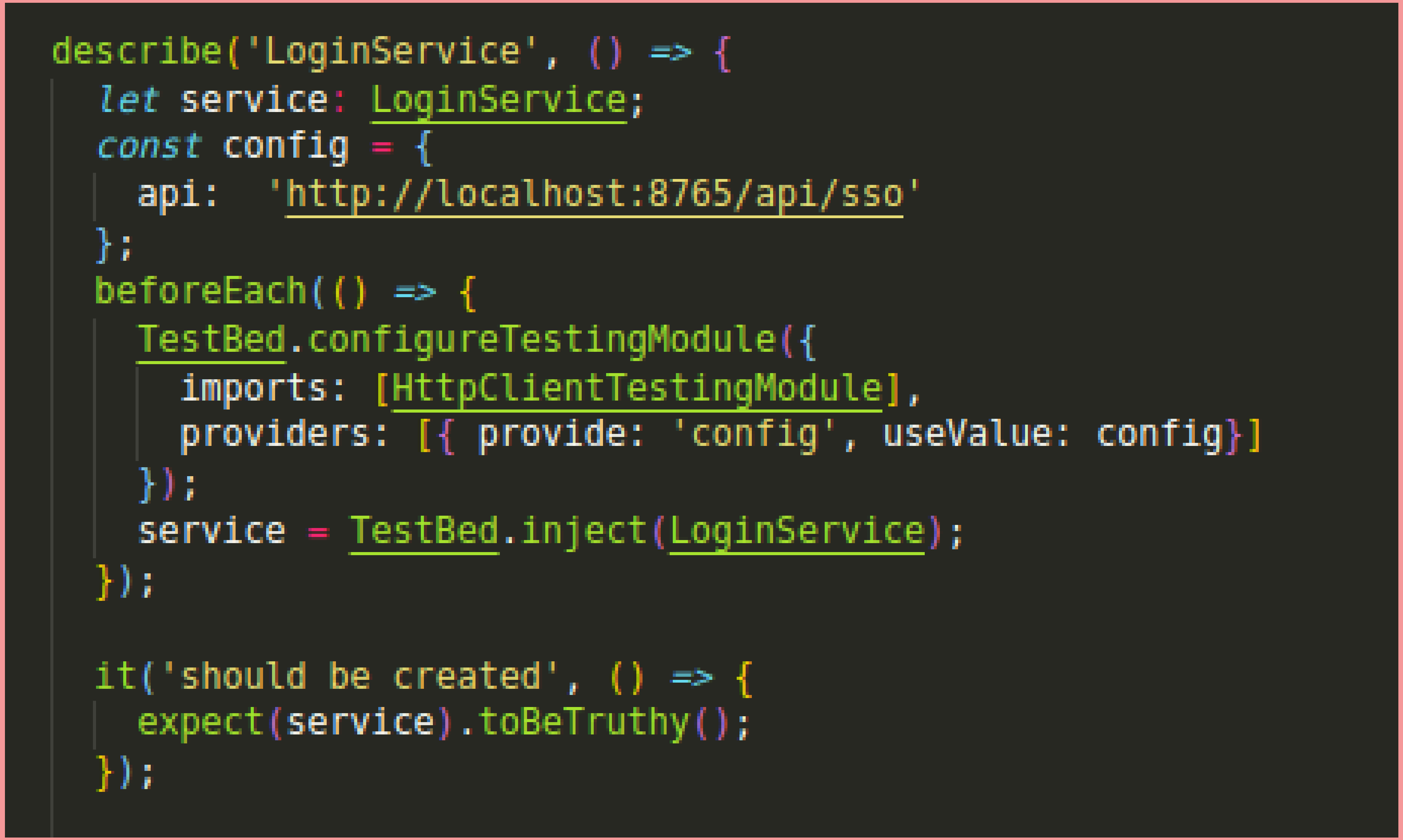
```
describe('aquí debe haber una descripción', () => {

    ✓ test('aquí debe ir una descripción de lo que hace el test', () => {

        // Arrange - Given

        // Act - When

        // Assert - Then
    })

})
```

HOW TO WRITE A TEST



```
✓ test("it should filter by a search term (sky)", () => {

    // Arrange - Given
    const input = [
        'Castle in the Sky','Grave of the Fireflies','My Neighbor Totoro'
    ];
    const text = 'sky';
    const output = ['Castle in the Sky'];

    // Act - When
    const filterByTitle = index.filterByTitle(input, text)

    // Assert - Then
    expect(filterByTitle).toEqual(output);
});
```

HOW TO WRITE A TEST



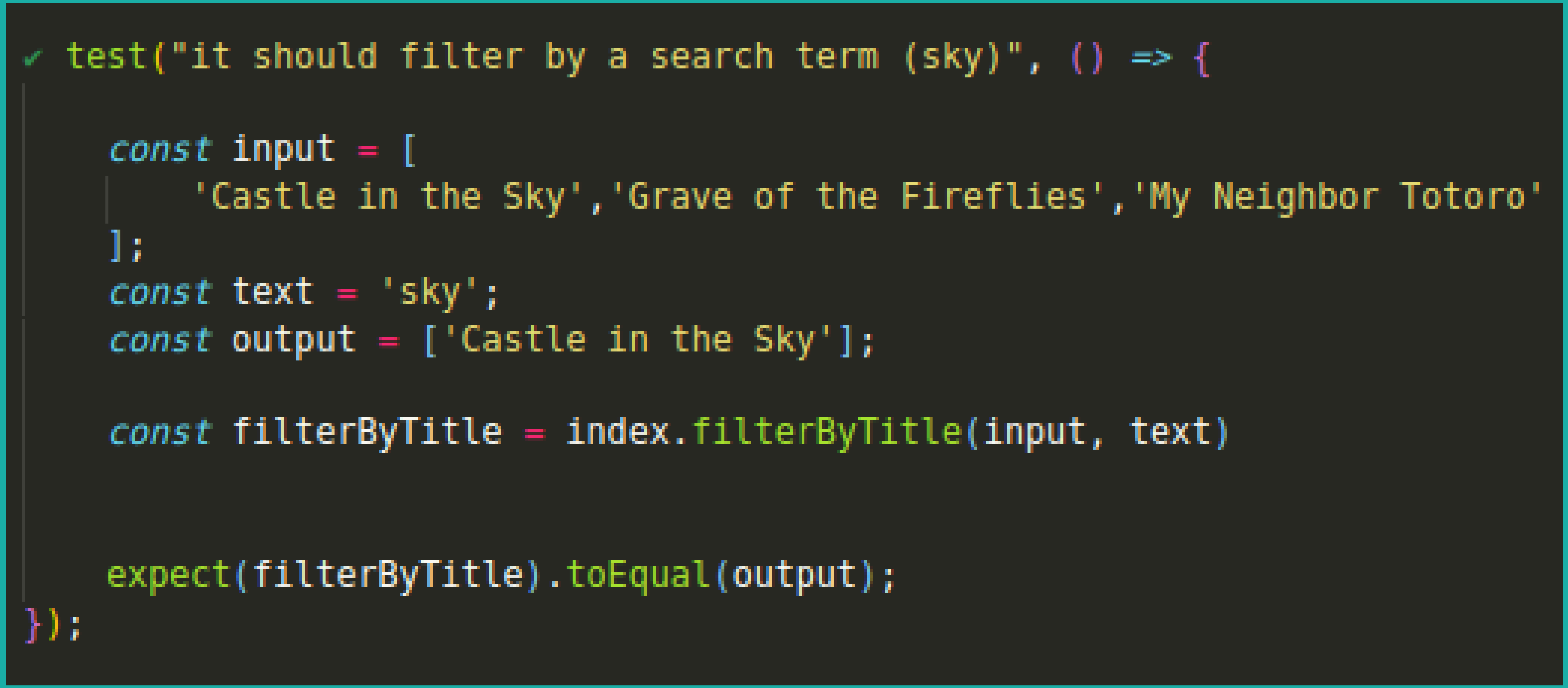
```
describe('LoginService', () => {
  let service: LoginService;
  const config = {
    api:  'http://localhost:8765/api/sso'
  };
  beforeEach(() => {
    TestBed.configureTestingModule({
      imports: [HttpClientTestingModule],
      providers: [{ provide: 'config', useValue: config}]
    });
    service = TestBed.inject(LoginService);
  });

  it('should be created', () => {
    expect(service).toBeTruthy();
  });
```

DOUBLES

OR MOCKS?

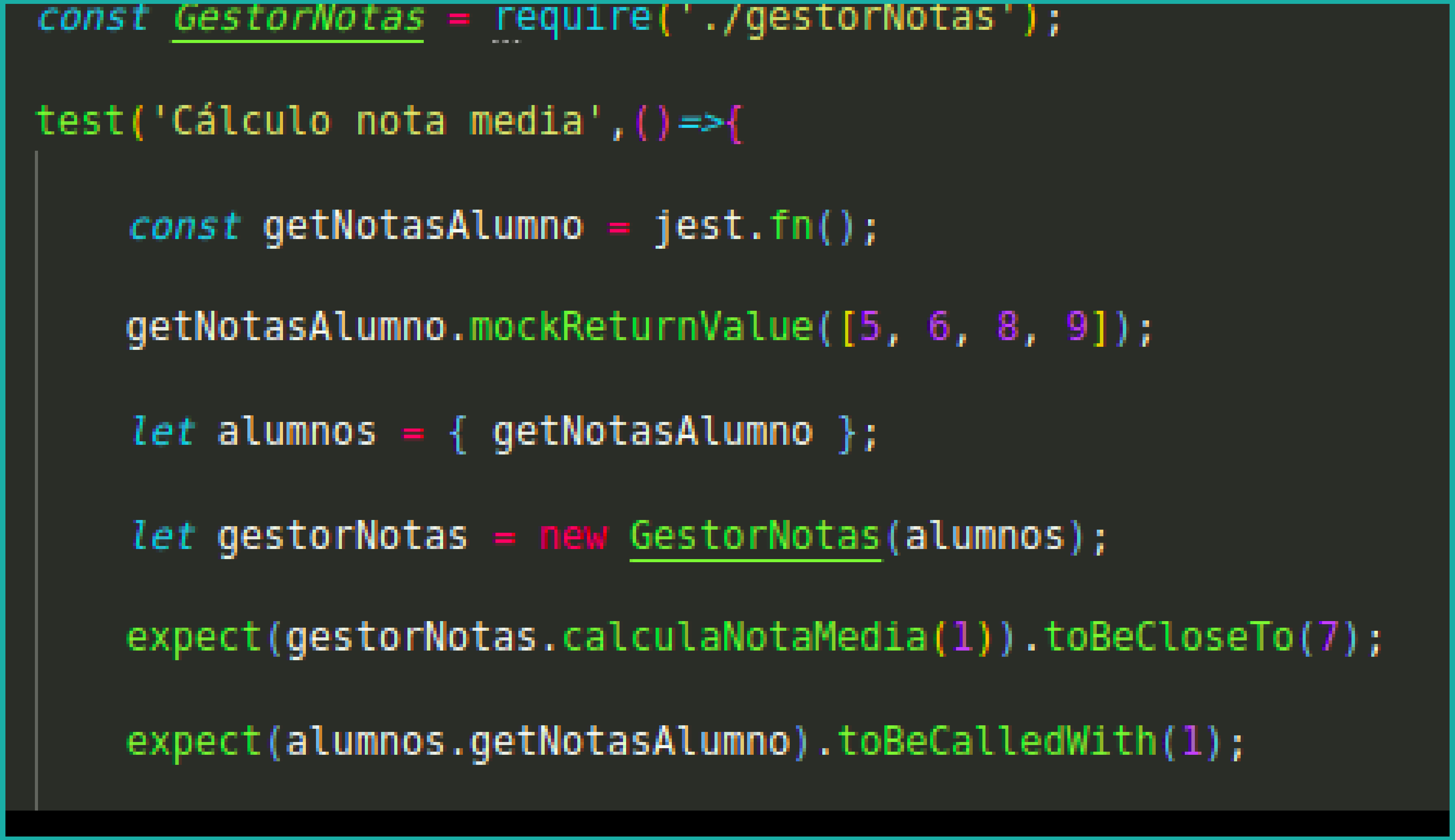
FAKE

DOUBLES



```
✓ test("it should filter by a search term (sky)", () => {

    const input = [
        'Castle in the Sky','Grave of the Fireflies','My Neighbor Totoro'
    ];
    const text = 'sky';
    const output = ['Castle in the Sky'];

    const filterByTitle = index.filterByTitle(input, text)


    expect(filterByTitle).toEqual(output);
});
```

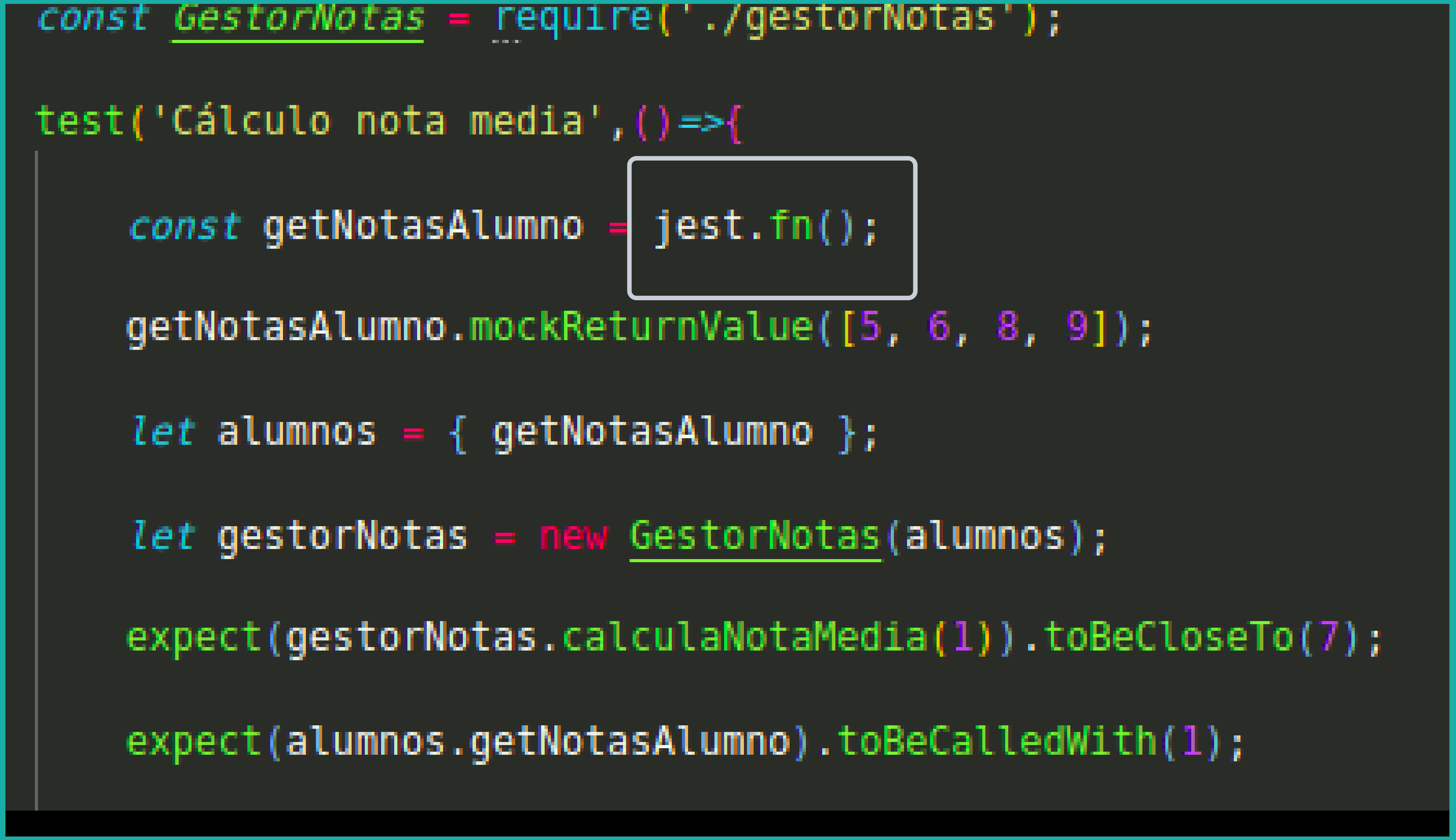
MOCK



```
const GestorNotas = require('./gestorNotas');

test('Cálculo nota media',()=>{

    const getNotasAlumno = jest.fn();

    getNotasAlumno.mockReturnValue([5, 6, 8, 9]);

    let alumnos = { getNotasAlumno };

    let gestorNotas = new GestorNotas(alumnos);

    expect(gestorNotas.calculaNotaMedia(1)).toBeCloseTo(7);

    expect(alumnos.getNotasAlumno).toBeCalledWith(1);
```

DOUBLES

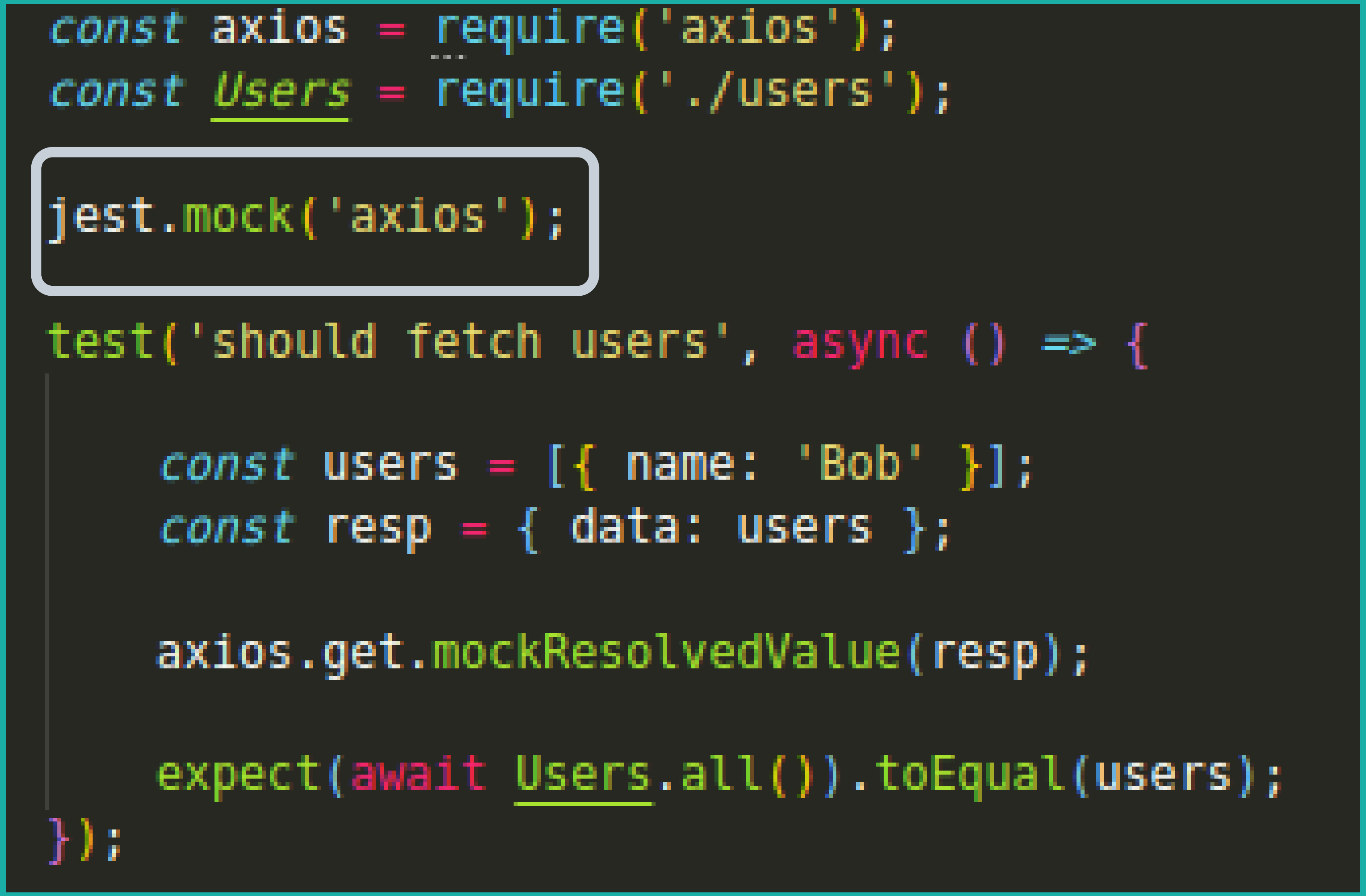
MOCK FUNCTIONS



```
const GestorNotas = require('./gestorNotas');

test('Cálculo nota media',()=>{

    const getNotasAlumno = jest.fn();

    getNotasAlumno.mockReturnValue([5, 6, 8, 9]);

    let alumnos = { getNotasAlumno };

    let gestorNotas = new GestorNotas(alumnos);

    expect(gestorNotas.calculaNotaMedia(1)).toBeCloseTo(7);

    expect(alumnos.getNotasAlumno).toBeCalledWith(1);
```

DOUBLES

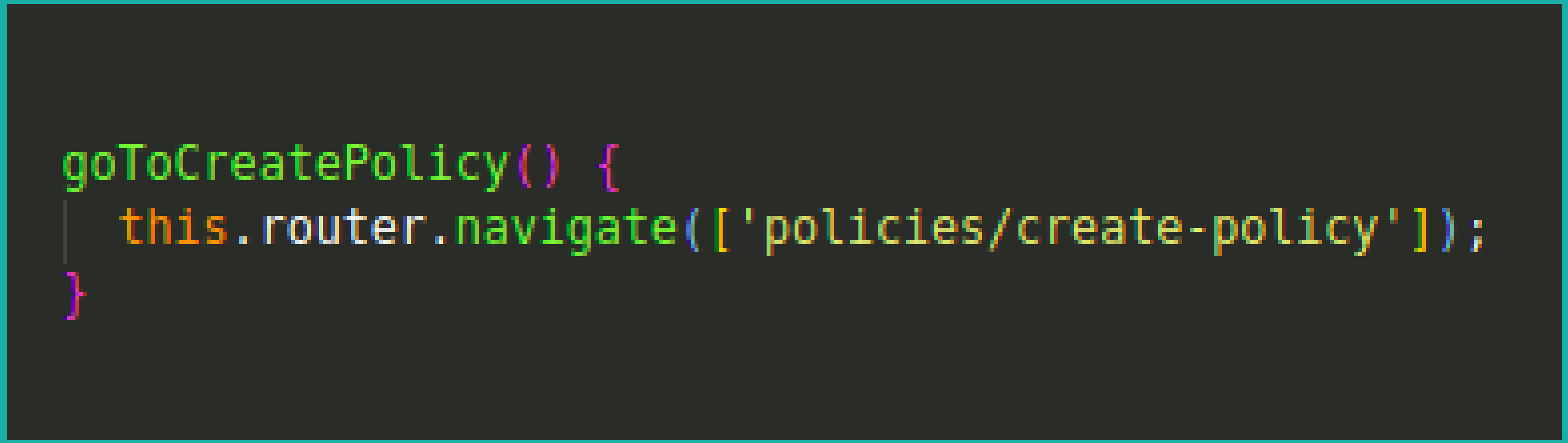
MOCK MODULES



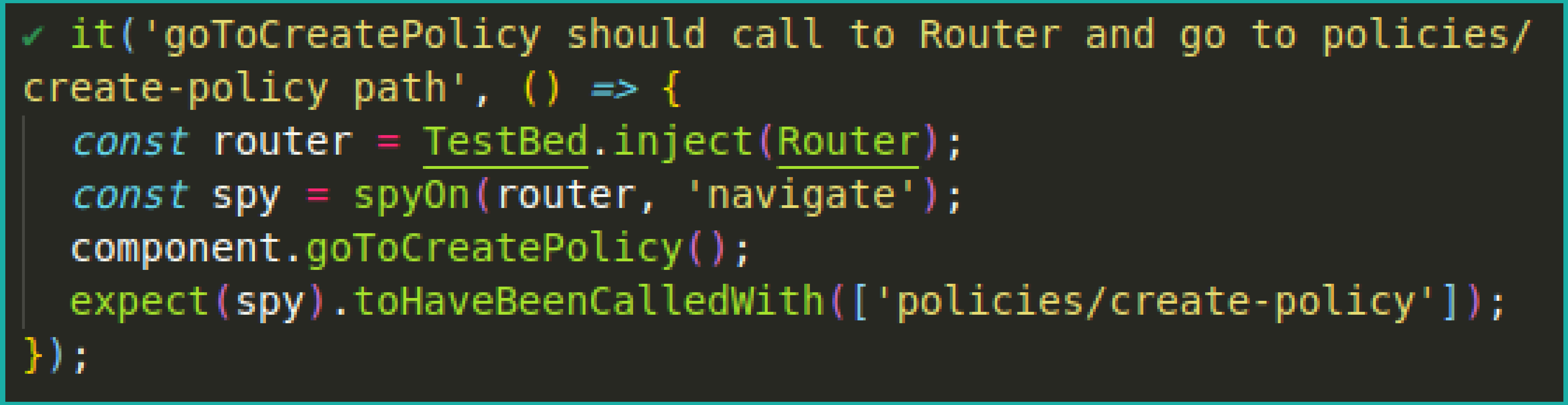
```
const axios = require('axios');
const Users = require('./users');

jest.mock('axios');

test('should fetch users', async () => {

    const users = [{ name: 'Bob' }];
    const resp = { data: users };

    axios.get.mockResolvedValue(resp);

    expect(await Users.all()).toEqual(users);
});
```

DOUBLES

SPY



```
goToCreatePolicy() {
  this.router.navigate(['policies/create-policy']);
}
```

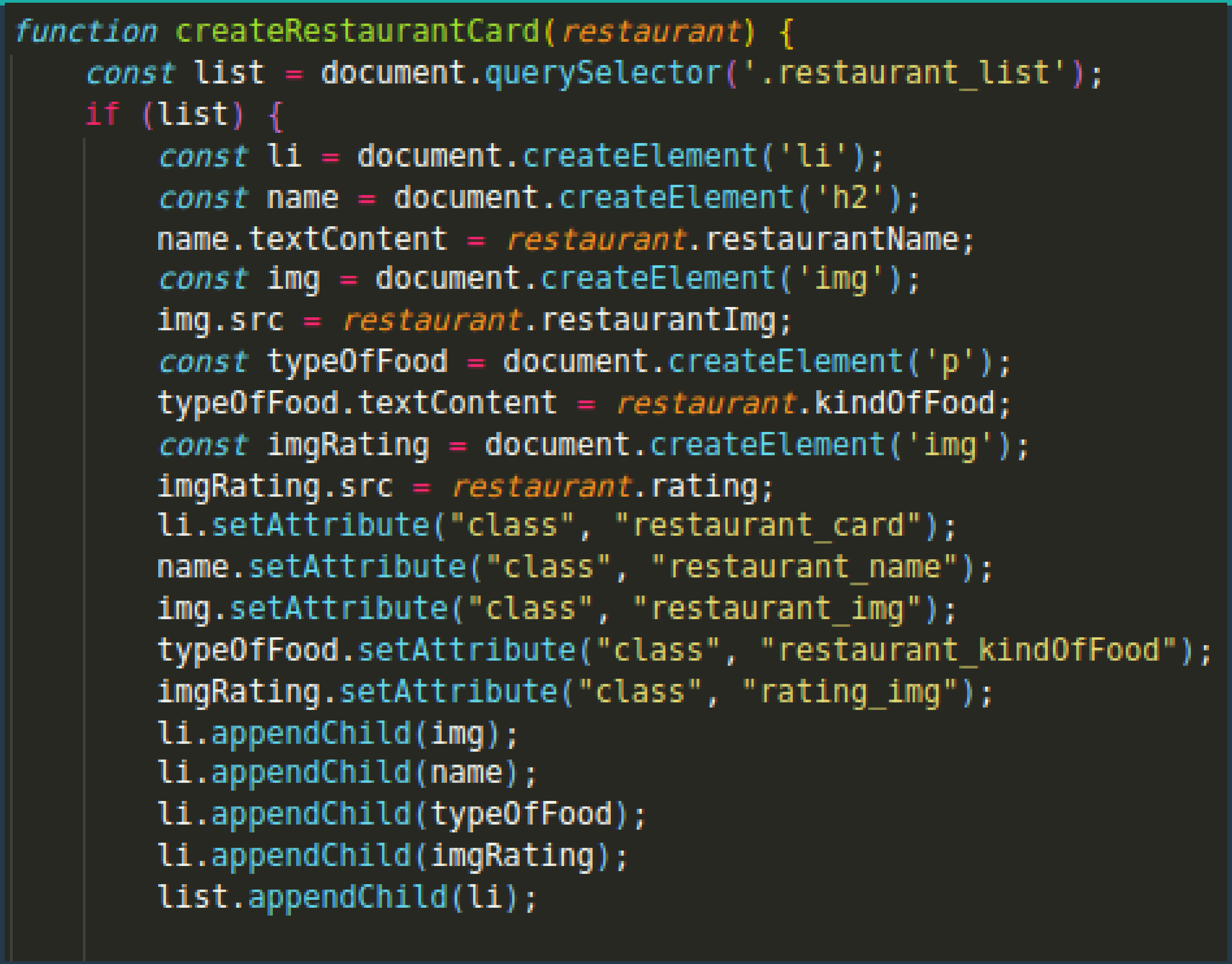


```
✓ it('goToCreatePolicy should call to Router and go to policies/
create-policy path', () => {
  const router = TestBed.inject(Router);
  const spy = spyOn(router, 'navigate');
  component.goToCreatePolicy();
  expect(spy).toHaveBeenCalledWith(['policies/create-policy']);
});
```

DOUBLES

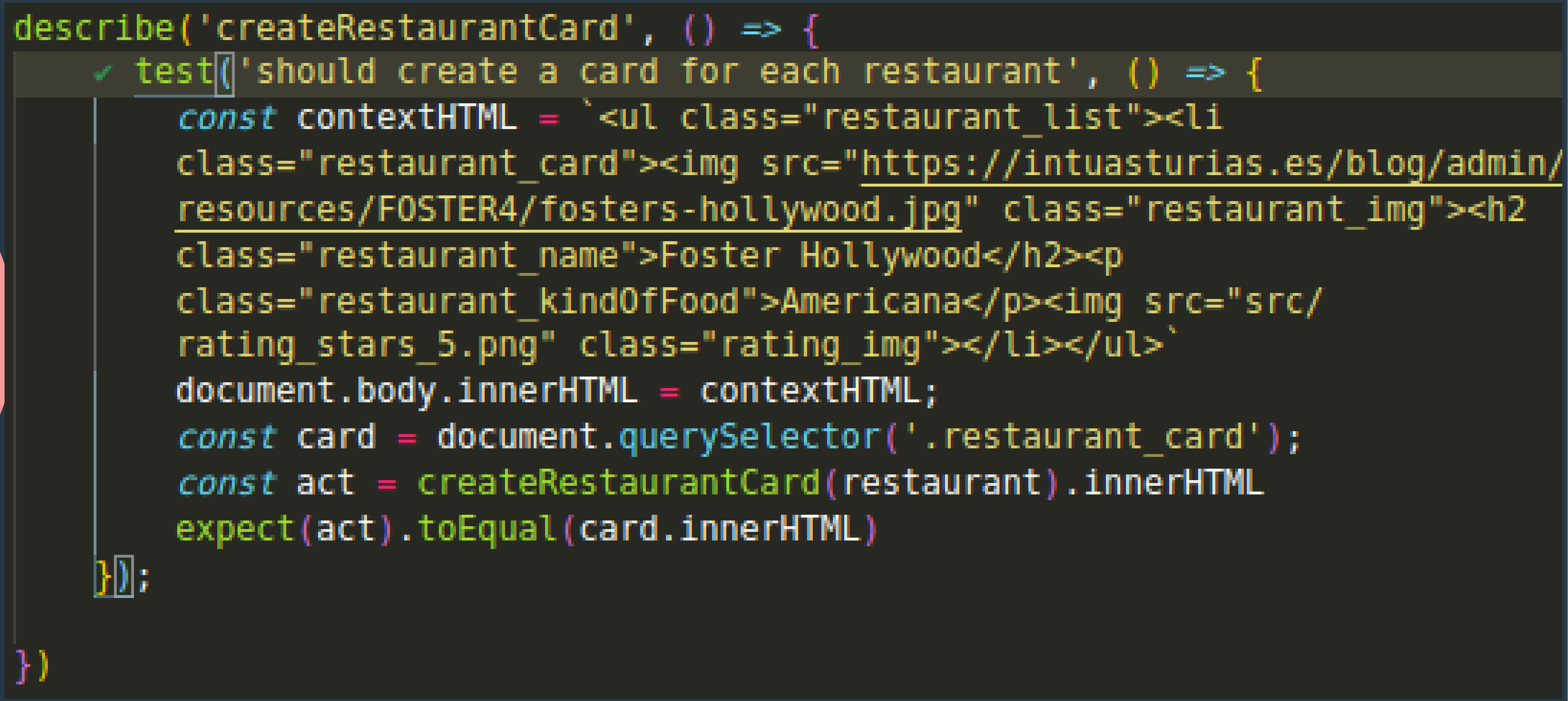
DOM

DOM



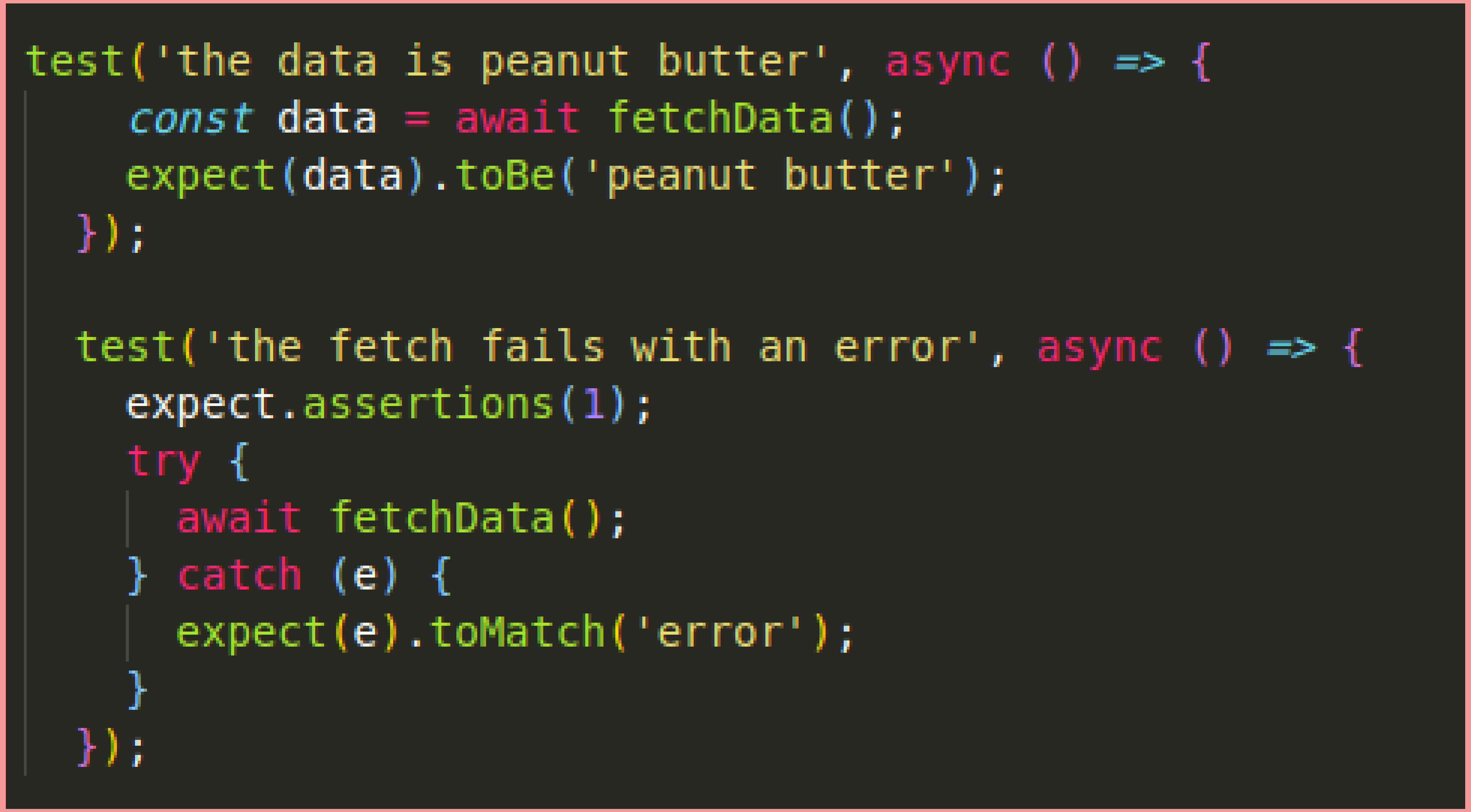
```
function createRestaurantCard(restaurant) {
    const list = document.querySelector('.restaurant_list');
    if (list) {
        const li = document.createElement('li');
        const name = document.createElement('h2');
        name.textContent = restaurant.restaurantName;
        const img = document.createElement('img');
        img.src = restaurant.restaurantImg;
        const typeOfFood = document.createElement('p');
        typeOfFood.textContent = restaurant.kindOfFood;
        const imgRating = document.createElement('img');
        imgRating.src = restaurant.rating;
        li.setAttribute("class", "restaurant_card");
        name.setAttribute("class", "restaurant_name");
        img.setAttribute("class", "restaurant_img");
        typeOfFood.setAttribute("class", "restaurant_kindOfFood");
        imgRating.setAttribute("class", "rating_img");
        li.appendChild(img);
        li.appendChild(name);
        li.appendChild(typeOfFood);
        li.appendChild(imgRating);
        list.appendChild(li);
```

DOM

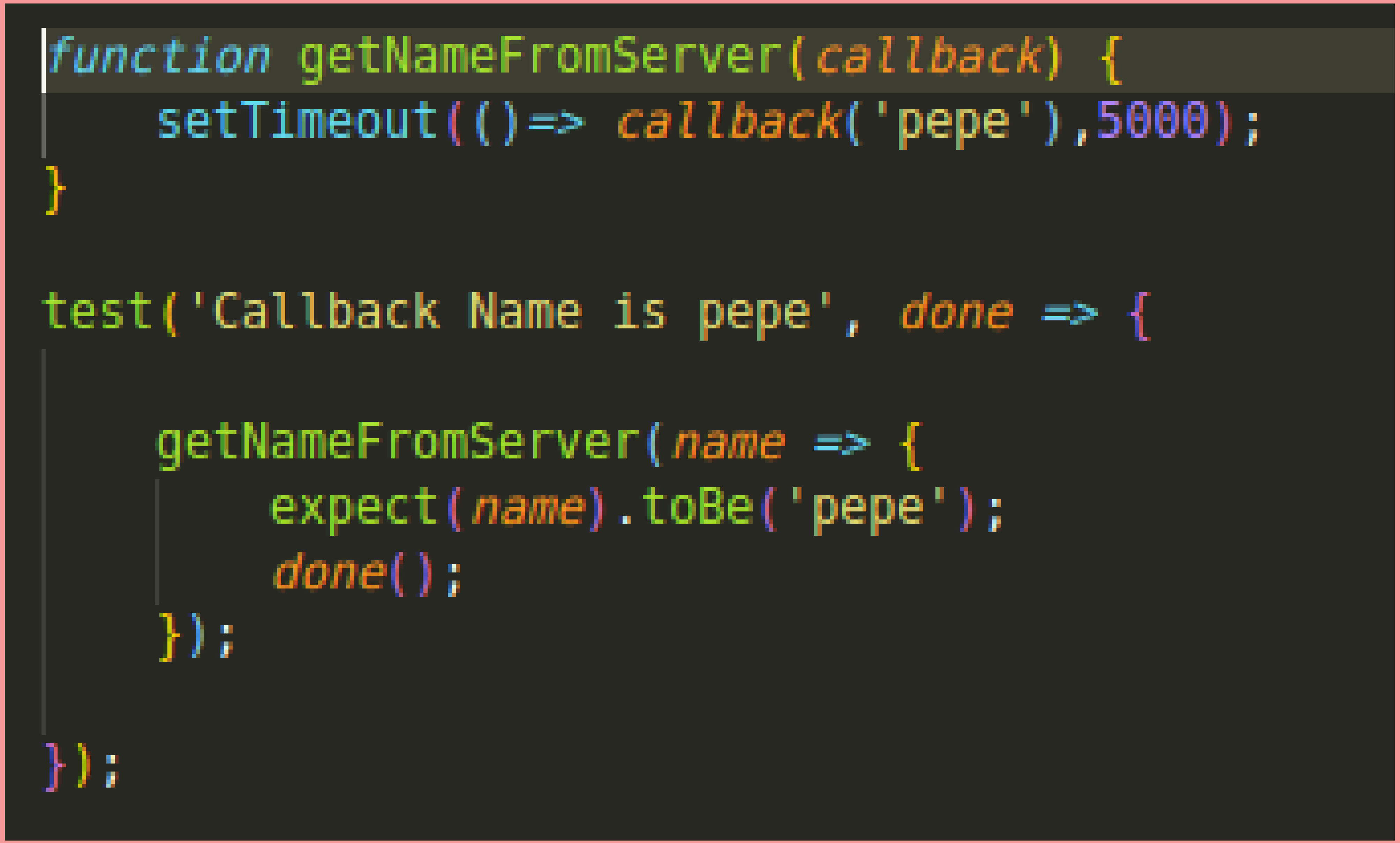


```
describe('createRestaurantCard', () => {
    test('should create a card for each restaurant', () => {
        const contextHTML = `<ul class="restaurant_list"><li
        class="restaurant_card"><img src="https://intuasturias.es/blog/admin/
        resources/FOSTER4/fosters-hollywood.jpg" class="restaurant_img"><h2
        class="restaurant_name">Foster Hollywood</h2><p
        class="restaurant_kindOfFood">Americana</p><img src="src/
        rating_stars_5.png" class="rating_img"></li></ul>`
        document.body.innerHTML = contextHTML;
        const card = document.querySelector('.restaurant_card');
        const act = createRestaurantCard(restaurant).innerHTML
        expect(act).toEqual(card.innerHTML)
    });

})
```

ASYNCRONOUS TEST

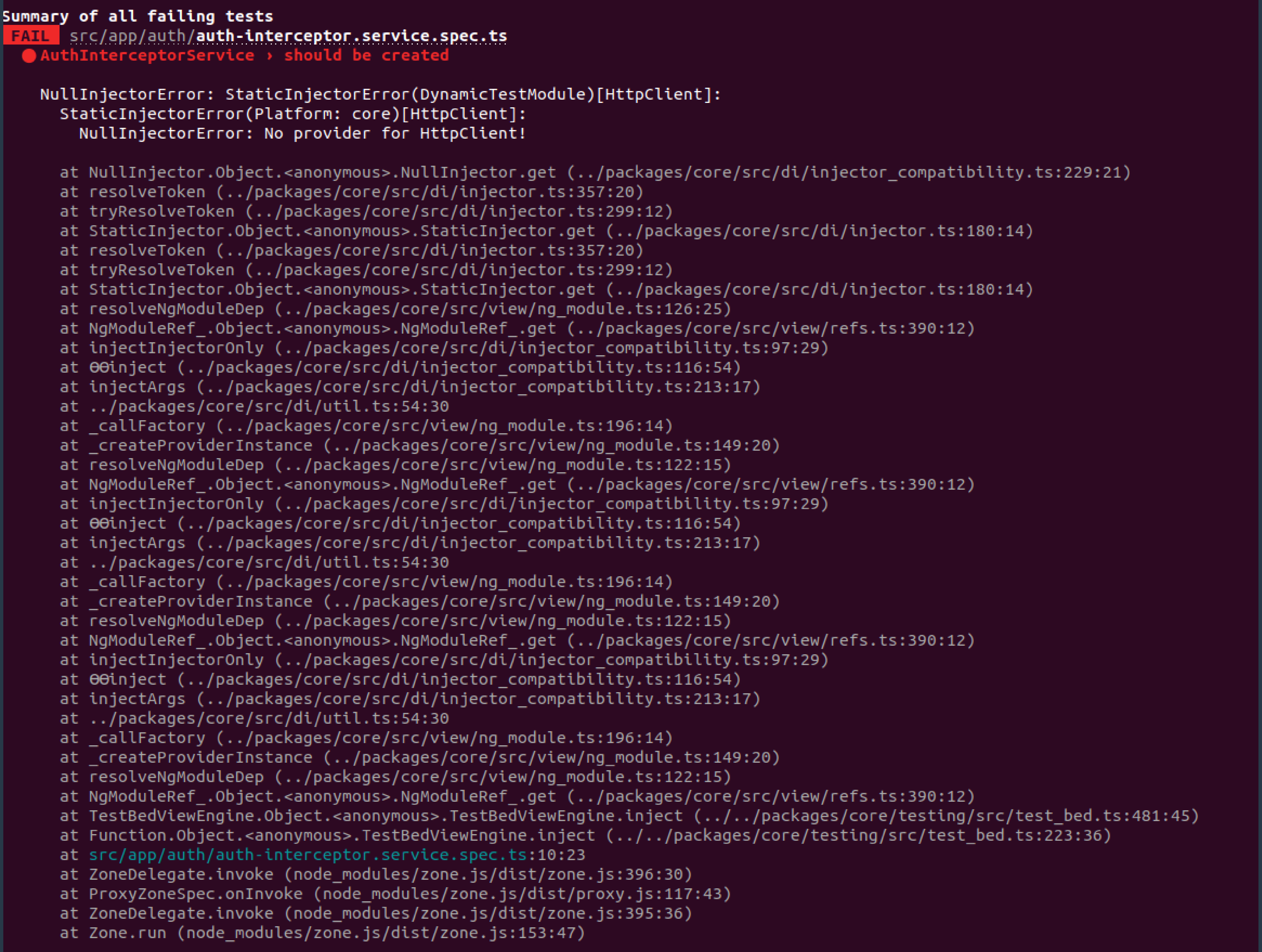


```
test('the data is peanut butter', async () => {
    const data = await fetchData();
    expect(data).toBe('peanut butter');
  });

  test('the fetch fails with an error', async () => {
    expect.assertions(1);
    try {
      await fetchData();
    } catch (e) {
      expect(e).toMatch('error');
    }
  });
```

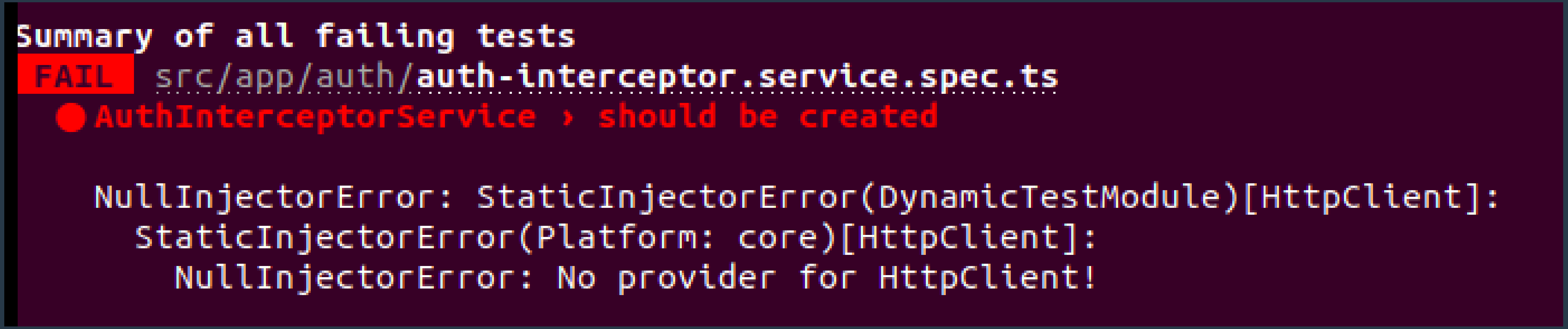


```
function getNameFromServer(callback) {
    setTimeout(()=> callback('pepe'),5000);
}

test('Callback Name is pepe', done => {

    getNameFromServer(name => {
        expect(name).toBe('pepe');
        done();
    });

});
```

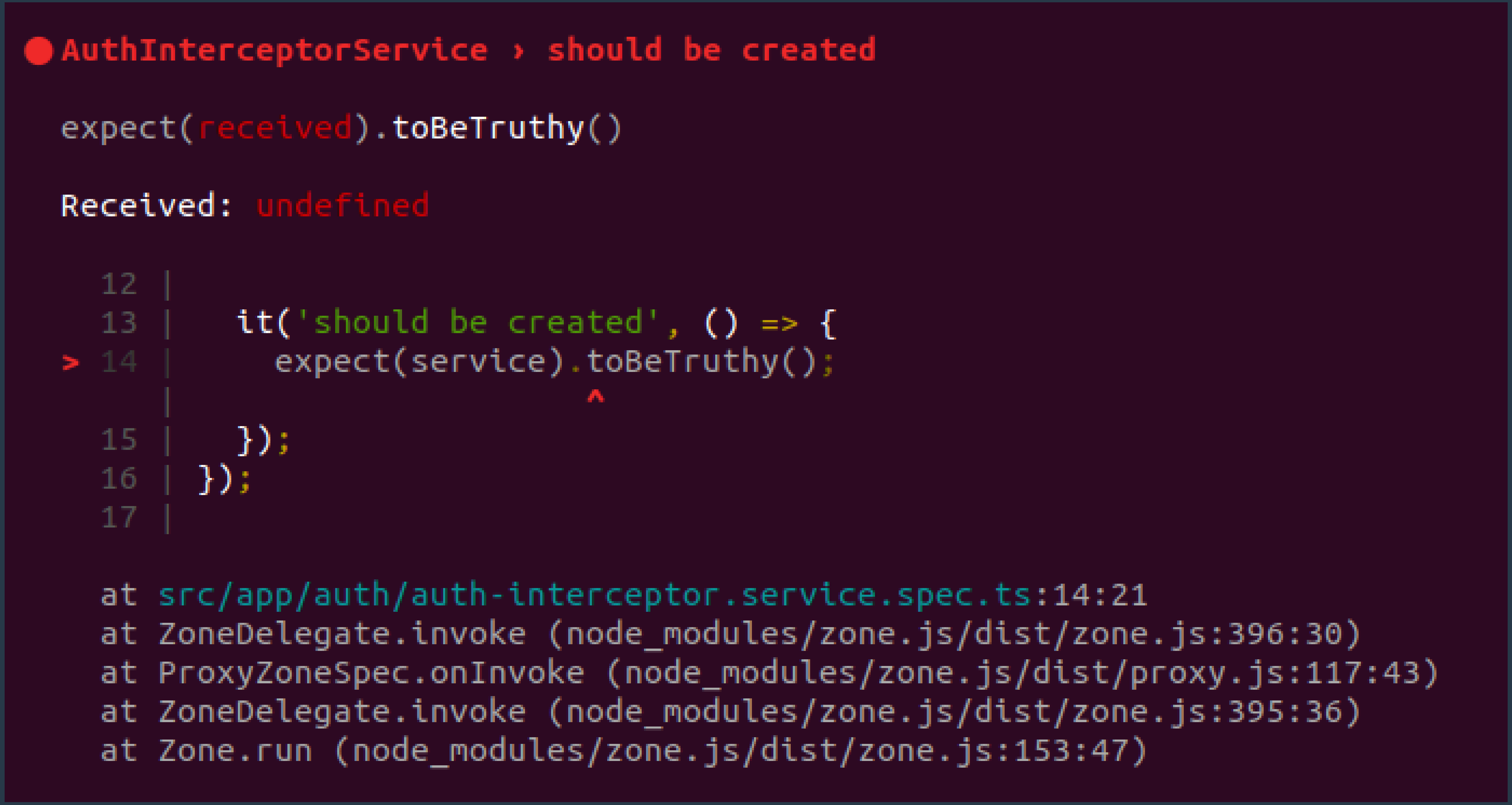
WHY MY TEST FAILS?



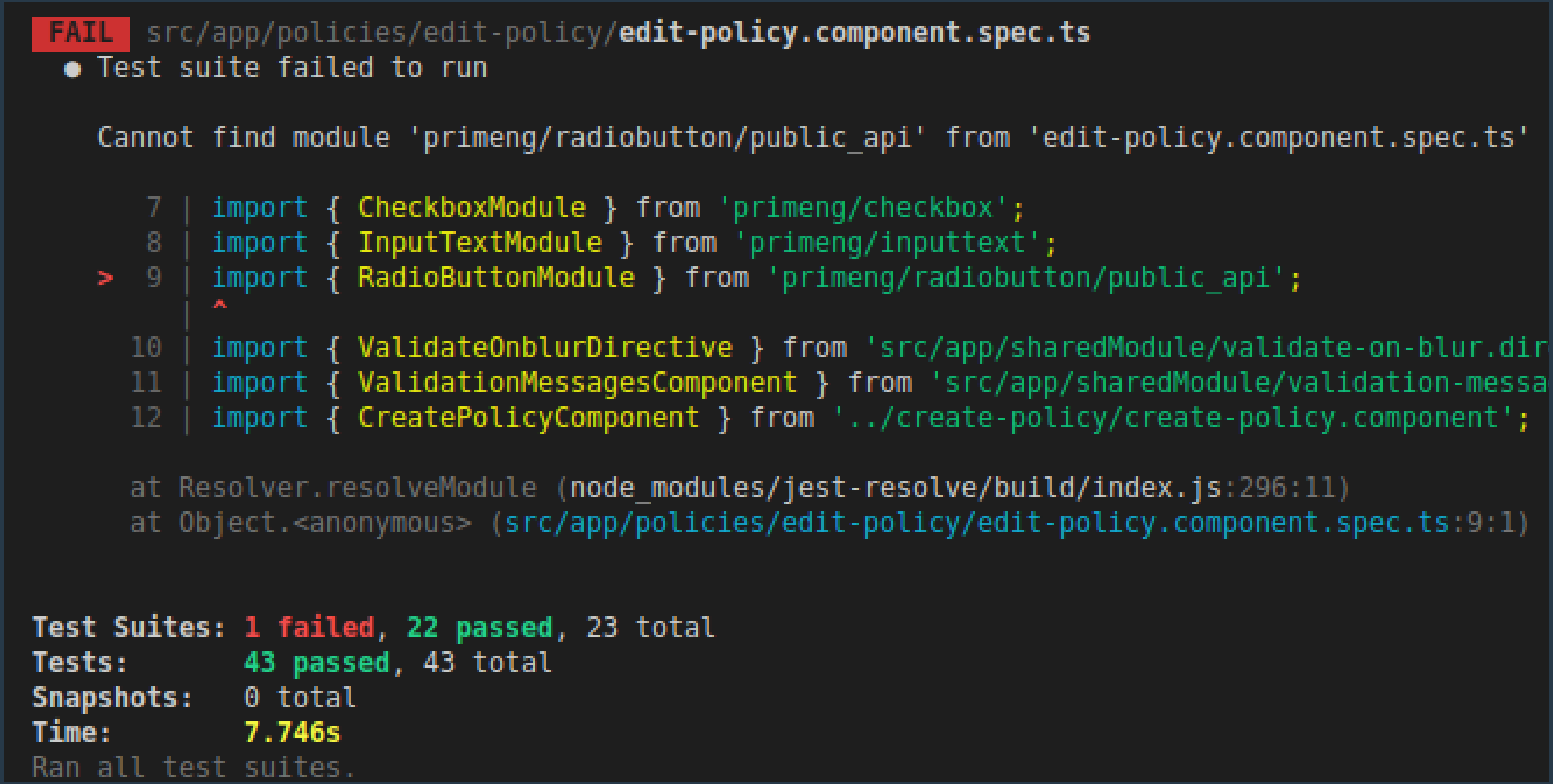
```
Summary of all failing tests
 FAIL  src/app/auth/auth-interceptor.service.spec.ts
  ● AuthInterceptorService › should be created

    NullInjectorError: StaticInjectorError(DynamicTestModule)[HttpClient]:
      StaticInjectorError(Platform: core)[HttpClient]:
        NullInjectorError: No provider for HttpClient!

      at NullInjector.Object.<anonymous>.NullInjector.get (../packages/core/src/di/injector_compatibility.ts:229:21)
      at resolveToken (../packages/core/src/di/injector.ts:357:20)
      at tryResolveToken (../packages/core/src/di/injector.ts:299:12)
      at StaticInjector.Object.<anonymous>.StaticInjector.get (../packages/core/src/di/injector.ts:180:14)
      at resolveToken (../packages/core/src/di/injector.ts:357:20)
      at tryResolveToken (../packages/core/src/di/injector.ts:299:12)
      at StaticInjector.Object.<anonymous>.StaticInjector.get (../packages/core/src/di/injector.ts:180:14)
      at resolveNgModuleDep (../packages/core/src/view/ng_module.ts:126:25)
      at NgModuleRef_.Object.<anonymous>.NgModuleRef_.get (../packages/core/src/view/refs.ts:390:12)
      at injectInjectorOnly (../packages/core/src/di/injector_compatibility.ts:97:29)
      at ɵɵinject (../packages/core/src/di/injector_compatibility.ts:116:54)
      at injectArgs (../packages/core/src/di/injector_compatibility.ts:213:17)
      at ../packages/core/src/di/util.ts:54:30
      at _callFactory (../packages/core/src/view/ng_module.ts:196:14)
      at _createProviderInstance (../packages/core/src/view/ng_module.ts:149:20)
      at resolveNgModuleDep (../packages/core/src/view/ng_module.ts:122:15)
      at NgModuleRef_.Object.<anonymous>.NgModuleRef_.get (../packages/core/src/view/refs.ts:390:12)
      at injectInjectorOnly (../packages/core/src/di/injector_compatibility.ts:97:29)
      at ɵɵinject (../packages/core/src/di/injector_compatibility.ts:116:54)
      at injectArgs (../packages/core/src/di/injector_compatibility.ts:213:17)
      at ../packages/core/src/di/util.ts:54:30
      at _callFactory (../packages/core/src/view/ng_module.ts:196:14)
      at _createProviderInstance (../packages/core/src/view/ng_module.ts:149:20)
      at resolveNgModuleDep (../packages/core/src/view/ng_module.ts:122:15)
      at NgModuleRef_.Object.<anonymous>.NgModuleRef_.get (../packages/core/src/view/refs.ts:390:12)
      at injectInjectorOnly (../packages/core/src/di/injector_compatibility.ts:97:29)
      at ɵɵinject (../packages/core/src/di/injector_compatibility.ts:116:54)
      at injectArgs (../packages/core/src/di/injector_compatibility.ts:213:17)
      at ../packages/core/src/di/util.ts:54:30
      at _callFactory (../packages/core/src/view/ng_module.ts:196:14)
      at _createProviderInstance (../packages/core/src/view/ng_module.ts:149:20)
      at resolveNgModuleDep (../packages/core/src/view/ng_module.ts:122:15)
      at NgModuleRef_.Object.<anonymous>.NgModuleRef_.get (../packages/core/src/view/refs.ts:390:12)
      at TestBedViewEngine.Object.<anonymous>.TestBedViewEngine.inject (../../packages/core/testing/src/test_bed.ts:481:45)
      at Function.Object.<anonymous>.TestBedViewEngine.inject (../../packages/core/testing/src/test_bed.ts:223:36)
      at src/app/auth/auth-interceptor.service.spec.ts:10:23
      at ZoneDelegate.invoke (node_modules/zone.js/dist/zone.js:396:30)
      at ProxyZoneSpec.onInvoke (node_modules/zone.js/dist/proxy.js:117:43)
      at ZoneDelegate.invoke (node_modules/zone.js/dist/zone.js:395:36)
      at Zone.run (node_modules/zone.js/dist/zone.js:153:47)
```



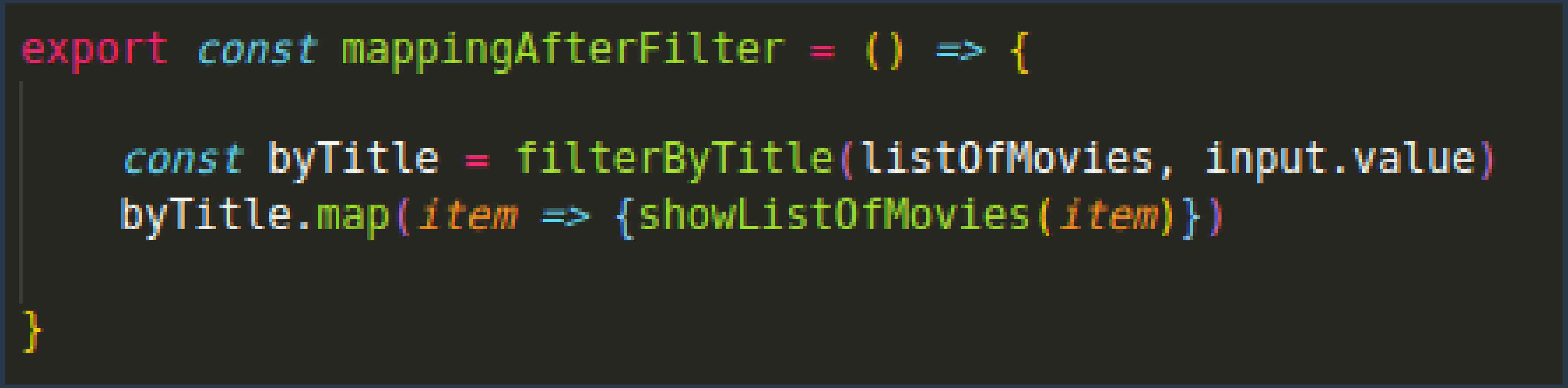
```
Summary of all failing tests
 FAIL  src/app/auth/auth-interceptor.service.spec.ts
  ● AuthInterceptorService › should be created

    NullInjectorError: StaticInjectorError(DynamicTestModule)[HttpClient]:
      StaticInjectorError(Platform: core)[HttpClient]:
        NullInjectorError: No provider for HttpClient!
```



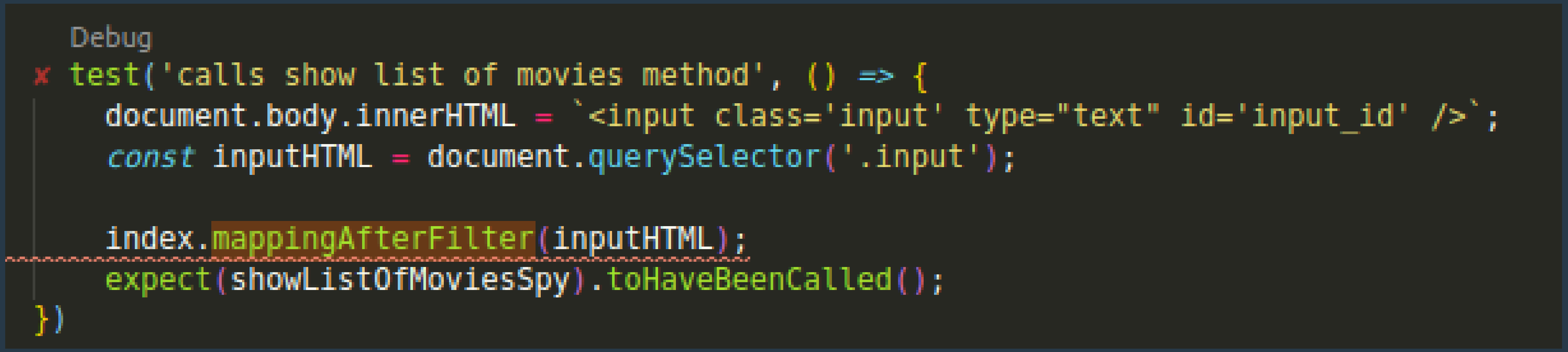
```
● AuthInterceptorService › should be created

  expect(received).toBeTruthy()

  Received: undefined

    12 |
    13 |   it('should be created', () => {
  > 14 |     expect(service).toBeTruthy();
       |                     ^
    15 |   });
    16 | });
    17 |

    at src/app/auth/auth-interceptor.service.spec.ts:14:21
    at ZoneDelegate.invoke (node_modules/zone.js/dist/zone.js:396:30)
    at ProxyZoneSpec.onInvoke (node_modules/zone.js/dist/proxy.js:117:43)
    at ZoneDelegate.invoke (node_modules/zone.js/dist/zone.js:395:36)
    at Zone.run (node_modules/zone.js/dist/zone.js:153:47)
```



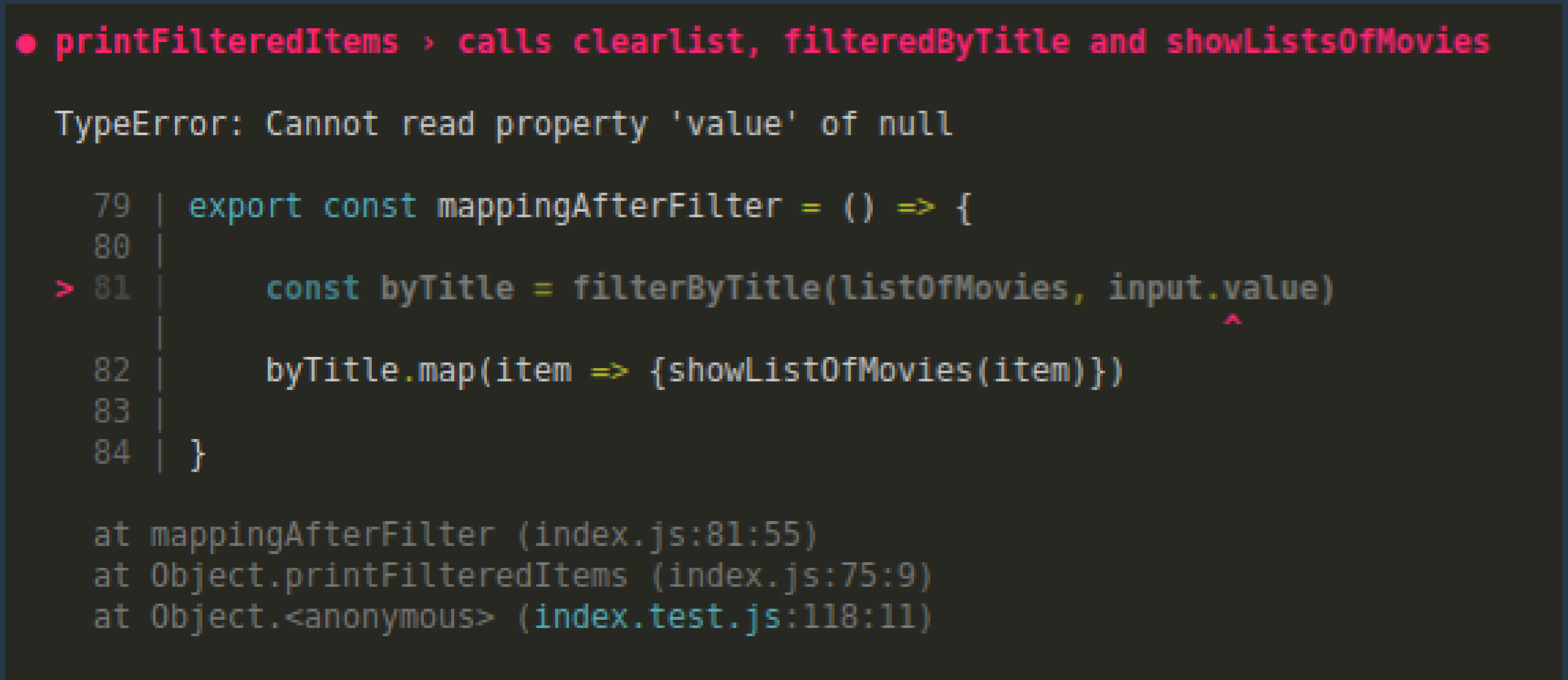
```
FAIL  src/app/policies/edit-policy/edit-policy.component.spec.ts
  ● Test suite failed to run

    Cannot find module 'primeng/radiobutton/public_api' from 'edit-policy.component.spec.ts'

       7 | import { CheckboxModule } from 'primeng/checkbox';
       8 | import { InputTextModule } from 'primeng/inputtext';
    >  9 | import { RadioButtonModule } from 'primeng/radiobutton/public_api';
         | ^
      10 | import { ValidateOnblurDirective } from 'src/app/sharedModule/validate-on-blur.dir
      11 | import { ValidationMessagesComponent } from 'src/app/sharedModule/validation-messa
      12 | import { CreatePolicyComponent } from '../create-policy/create-policy.component';

      at Resolver.resolveModule (node_modules/jest-resolve/build/index.js:296:11)
      at Object.<anonymous> (src/app/policies/edit-policy/edit-policy.component.spec.ts:9:1)


Test Suites: 1 failed, 22 passed, 23 total
Tests:       43 passed, 43 total
Snapshots:   0 total
Time:        7.746s
Ran all test suites.
```



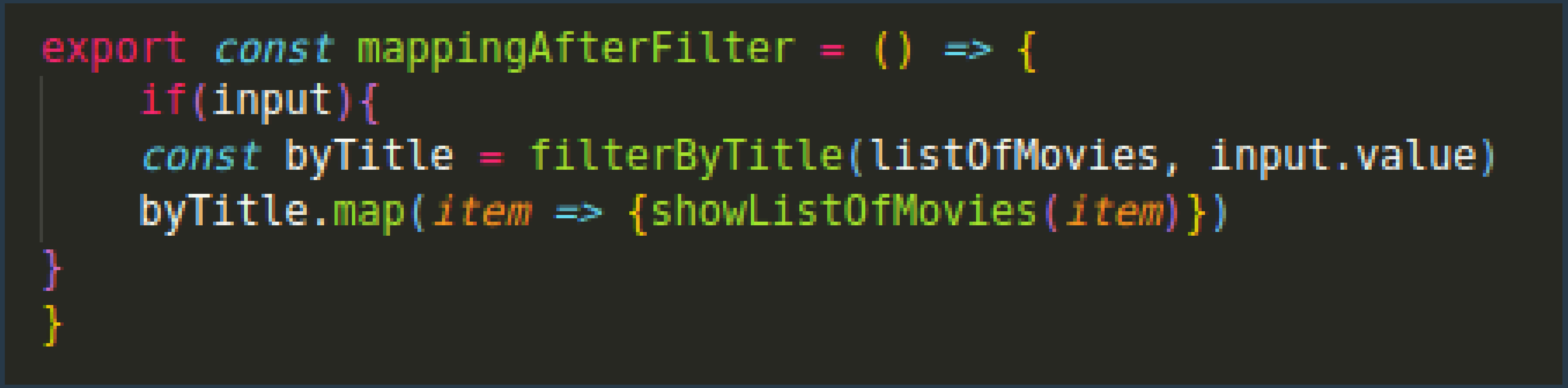
```
export const mappingAfterFilter = () => {

    const byTitle = filterByTitle(listOfMovies, input.value)
    byTitle.map(item => {showListOfMovies(item)})

}
```



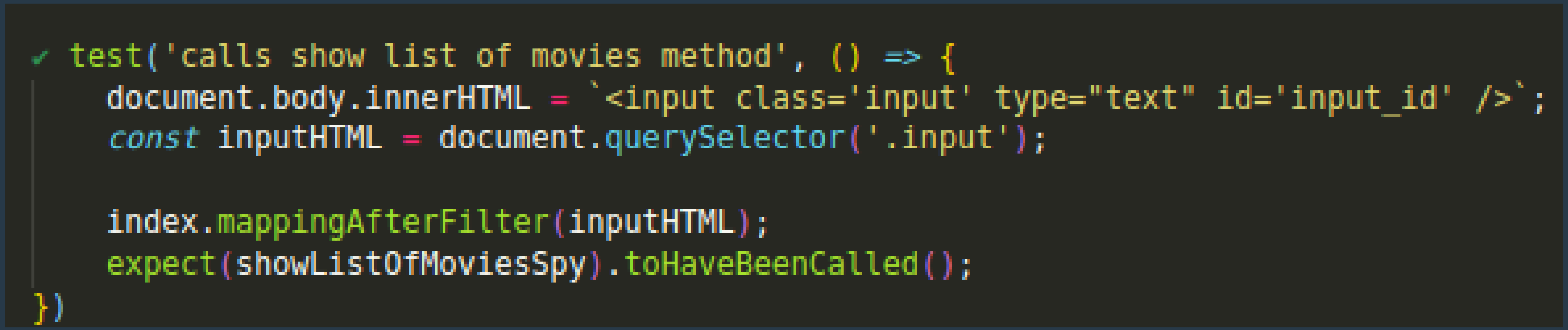
```
  Debug
✘ test('calls show list of movies method', () => {
    document.body.innerHTML = `<input class='input' type="text" id='input_id' />`;
    const inputHTML = document.querySelector('.input');

    index.mappingAfterFilter(inputHTML);
    expect(showListOfMoviesSpy).toHaveBeenCalled();
 })
```



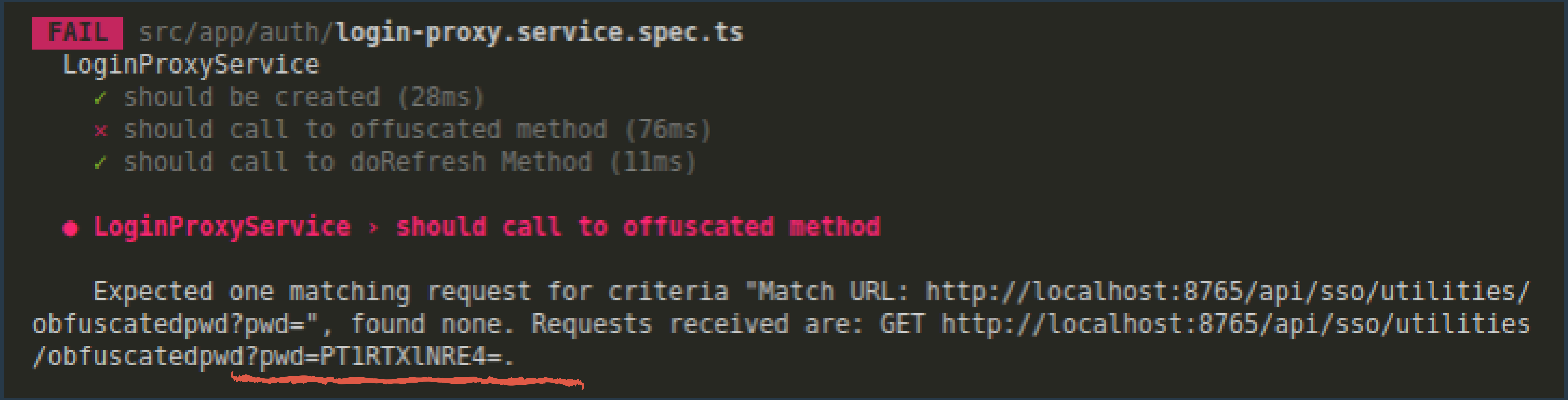
```
● printFilteredItems › calls clearlist, filteredByTitle and showListsOfMovies

  TypeError: Cannot read property 'value' of null

    79 | export const mappingAfterFilter = () => {
    80 |
  > 81 |     const byTitle = filterByTitle(listOfMovies, input.value)
       |                                                       ^
    82 |     byTitle.map(item => {showListOfMovies(item)})
    83 |
    84 | }

    at mappingAfterFilter (index.js:81:55)
    at Object.printFilteredItems (index.js:75:9)
    at Object.<anonymous> (index.test.js:118:11)
```



```javascript
export const mappingAfterFilter = () => {
    if(input){
    const byTitle = filterByTitle(listOfMovies, input.value)
    byTitle.map(item => {showListOfMovies(item)})
}
}
```



```javascript
✓ test('calls show list of movies method', () => {
    document.body.innerHTML = `<input class='input' type="text" id='input_id' />`;
    const inputHTML = document.querySelector('.input');

    index.mappingAfterFilter(inputHTML);
    expect(showListOfMoviesSpy).toHaveBeenCalled();
})
```



```
FAIL  src/app/auth/login-proxy.service.spec.ts
  LoginProxyService
    ✓ should be created (28ms)
    × should call to offuscated method (76ms)
    ✓ should call to doRefresh Method (11ms)

  ● LoginProxyService › should call to offuscated method

    Expected one matching request for criteria "Match URL: http://localhost:8765/api/sso/utilities/
obfuscatedpwd?pwd=", found none. Requests received are: GET http://localhost:8765/api/sso/utilities
/obfuscatedpwd?pwd=PT1RTXlNRE4=.
```

COVERAGE

100% FAKE

THINGS TO KEEP IN MIND

CONCLUSIONS

THE TEST DOESN`T KNOW

You have to give it the same context as your app to do that test

BUGS FREE...NOT

The circumstances are defined by us and our apps have errors.
tests > bugs

GREEN IS NOT ALWAYS GOOD

Remember false positives. Break your code to check the test
Ask for a code review.

RED IS NOT BAD

Learn from it!
Search the error, understand it and turn it in green.

JUST ONE ASSERTION

But it has to exist! A test without an assertion passes even though it's doing nothing.

YOU ARE ALLOW TO DUPLICATE

It's ok if it's easier to read and to understand.

COVERAGE

The coverage should ever be your goal. Ever.

REFACTOR IN GREEN

Do not refactor with failing tests, because afterwards you don't know what is broken in your code.

NOT ONLY HAPPY PATH

Try the sad path and all the possible options

ASSUME THE LEARNING CURVE

It's neither easy nor fast. But assume it and enjoy the path!

PRACTISE

A lot. Do testing in old side projects.

WHEN YOU FEEL CONFORTABLE

Start doing them in your sprint

Thank you! =)

I HOPE YOU LIKED IT AND MOTIVATED YOU TO LEARN

Recursos:

Versión larga de esta charla en español: 'Del 2+2 al fetch', Ari.

Charla: "it('should be easier'): Testing automatizado en el mundo real, Paqui Calabria

Charla: "El testing ya no es para gurús", RIcardo Borillo

Charla: "Mocks & Stubs, Ken Scambler

Charla: "Testing, CI and CD in the real world", Roc Boronat

Introducción al testing, Jaime Barrio (openWebinars)

Charla: 'Testing práctico con JavaScript", Ramón Guijarro

Testing: Introducción y buenas prácticas, CodelyTV

Testing con Javascript, Jorge Baumann

100% Code Coverage is Useless, Jorge Baumann

Recursos:

Learn the smart, efficient way to test any Javascript application, Kent Dodds.

Write tests. Not too many. Mostly integration, Kent Dodds.

The Merits of Mocking, Kent Dodds.

Testing JavaScript with Jest, Flavio Copes

Libro: Clean code, solid y testing aplicado a JavaScript, Miguel A. Gómez (Software crafters)

Documentación oficial de Jest

Jest cheat sheet

Documentación oficial de Cypress

SuperTest