

Nuestro primer test unitario con JavaScript

DigitalHouse>



**Certified Tech
Developer**

The Ultimate Degree

Índice

1. [Configurar el framework](#)
2. [Nuestro primer test](#)
3. [Agrupar unit tests](#)

1 | Configurar el Framework

Framework para JavaScript

Para JavaScript vamos a utilizar el framework **Jest**. Si queremos configurarlo, debemos instalar:

1. Un IDE (el recomendado es **Visual Studio Code** (<https://code.visualstudio.com>)
2. **Node.js** (<https://nodejs.org>).

Debemos tener en cuenta que si el código brindado no posee el archivo **package.json** debemos crearlo, ya que aquí se van a guardar todas las configuraciones de nuestro proyecto. Para crearlo debemos correr en la terminal el comando: **npm init -y**



3. JEST (<https://jestjs.io/>)

Para incluir Jest dentro de nuestro proyecto debemos correr en la terminal el comando: **npm install --save-dev jest** (tener en cuenta que esto debemos realizarlo en cada uno de los proyectos que vamos a trabajar).



2 | **Nuestro primer test**

Crear nuestro primer unit test

1

Bajar el código fuente del siguiente repositorio:

<https://github.com/academind/js-testing-introduction/tree/s tarting-setup>.

Este programa solicita el ingreso de **Nombre** y **Edad** y al presionar el botón **Agregar Usuario** arma una lista con los datos ingresados.

Nombre

Tutor

Edad

25

Agregar Usuario

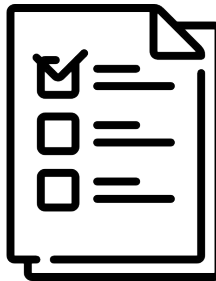
Profesor (30 years old)

Tutor (25 years old)

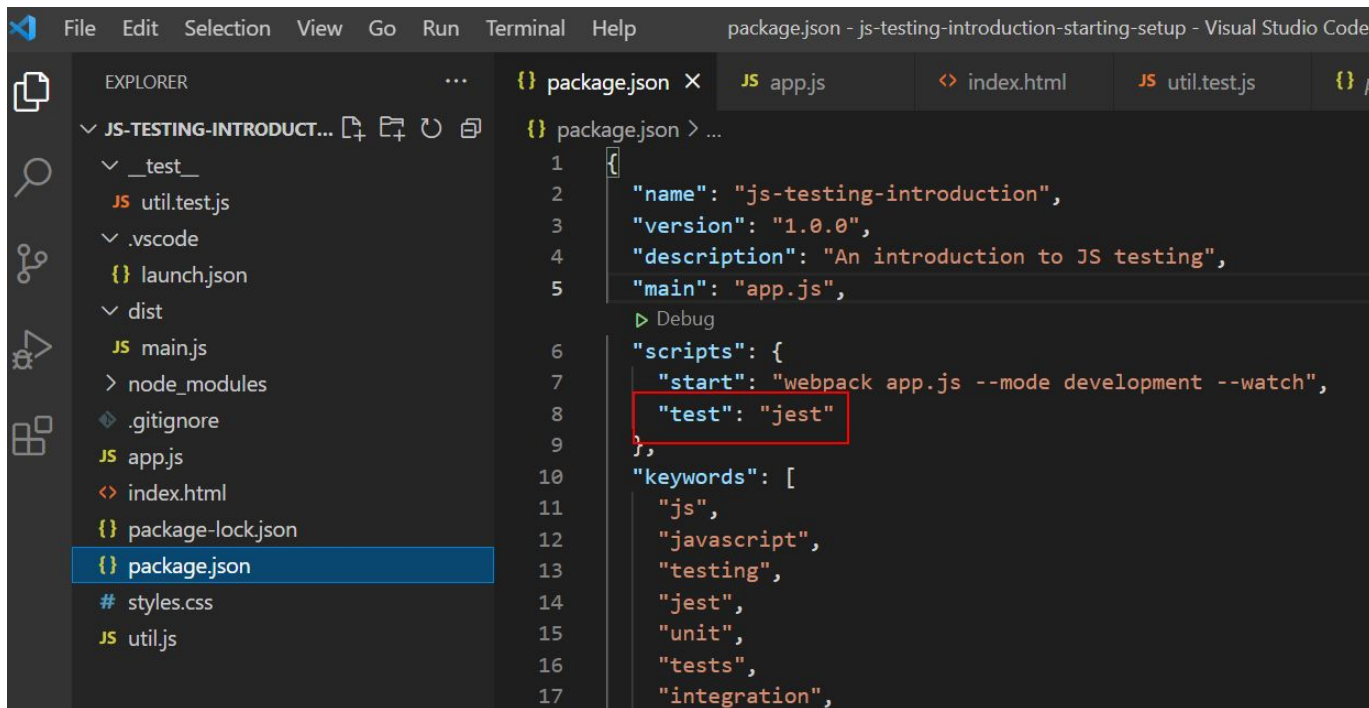
2

Configurar Jest como ejecutor de las pruebas:

1. Ir al archivo **package.json**.
2. En la parte de scripts/test debemos reemplazar **"echo "Error: no test specified" && exit 1"** por **"jest"**. De esta forma indicamos que vamos a correr test con nuestro framework Jest.



2



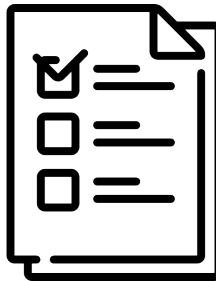
The screenshot shows the Visual Studio Code interface with the 'package.json' file open in the editor. The Explorer sidebar on the left shows the project structure, with 'package.json' selected. The editor displays the following JSON content:

```
1 {  
2   "name": "js-testing-introduction",  
3   "version": "1.0.0",  
4   "description": "An introduction to JS testing",  
5   "main": "app.js",  
6   "scripts": {  
7     "start": "webpack app.js --mode development --watch",  
8     "test": "jest"  
9   },  
10  "keywords": [  
11    "js",  
12    "javascript",  
13    "testing",  
14    "jest",  
15    "unit",  
16    "tests",  
17    "integration",
```

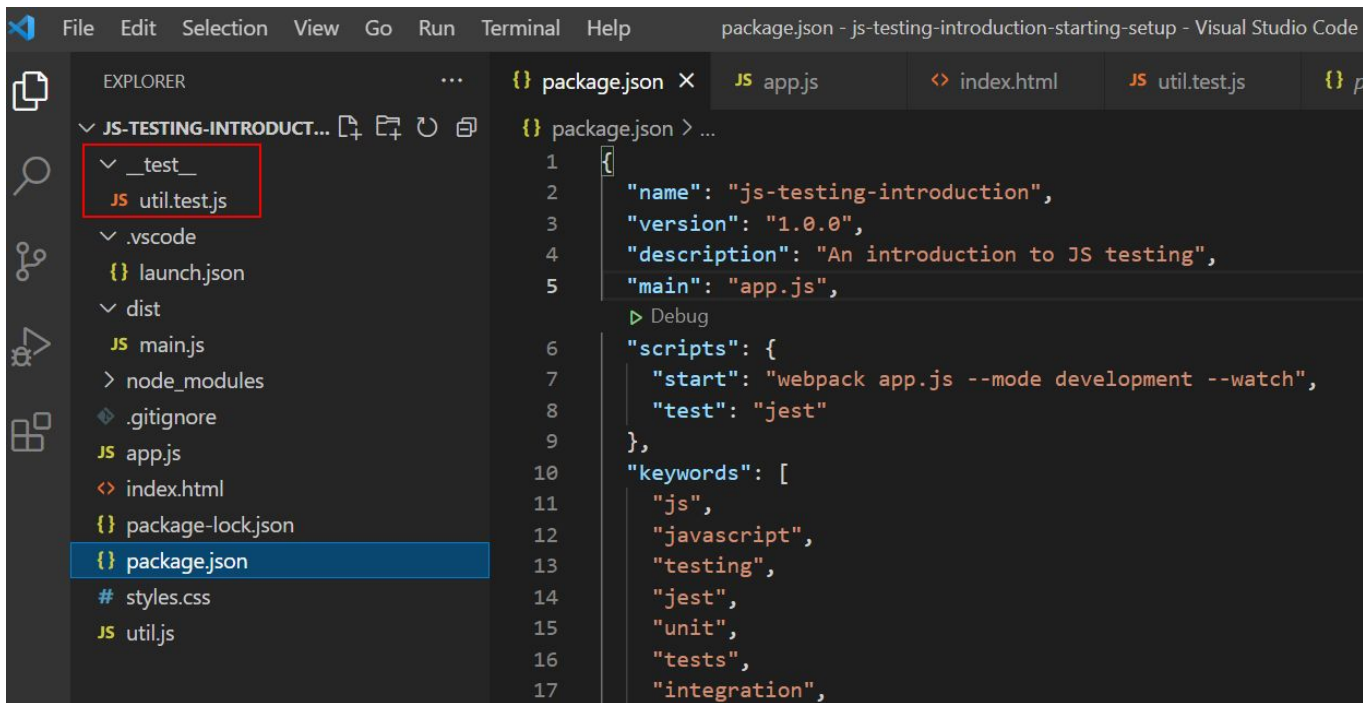
3

Crear la carpeta y archivo de prueba:

1. Crear la carpeta **__test__** donde se encontrarán todos los archivos de prueba.
2. Crear el archivo de prueba ***NombreArchivo.test.js***. Para que Jest reconozca los archivos de prueba deben terminar en **.test.js**. En este caso nuestro archivo se llamará **util.test.js**.

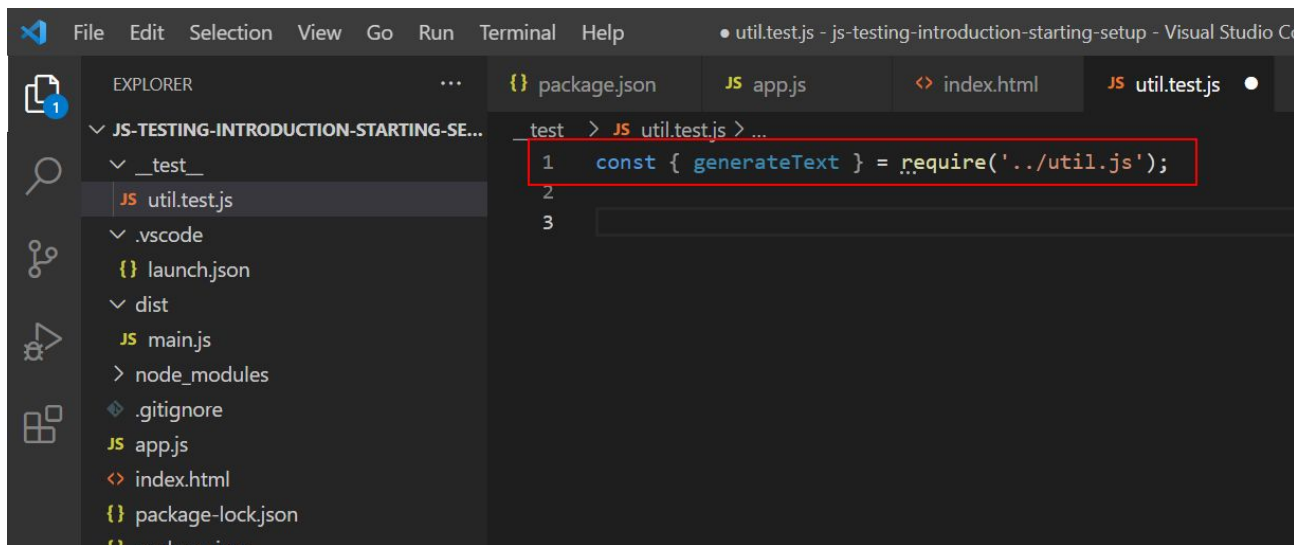


3



4

Importar el código a probar: debemos importar la sección del código que queremos probar. En este caso vamos a probar el que se encuentra en el archivo **util.js**, ya que ahí se encuentra la lógica de nuestro programa.



```
File Edit Selection View Go Run Terminal Help • util.test.js - js-testing-introduction-starting-setup - Visual Studio Co

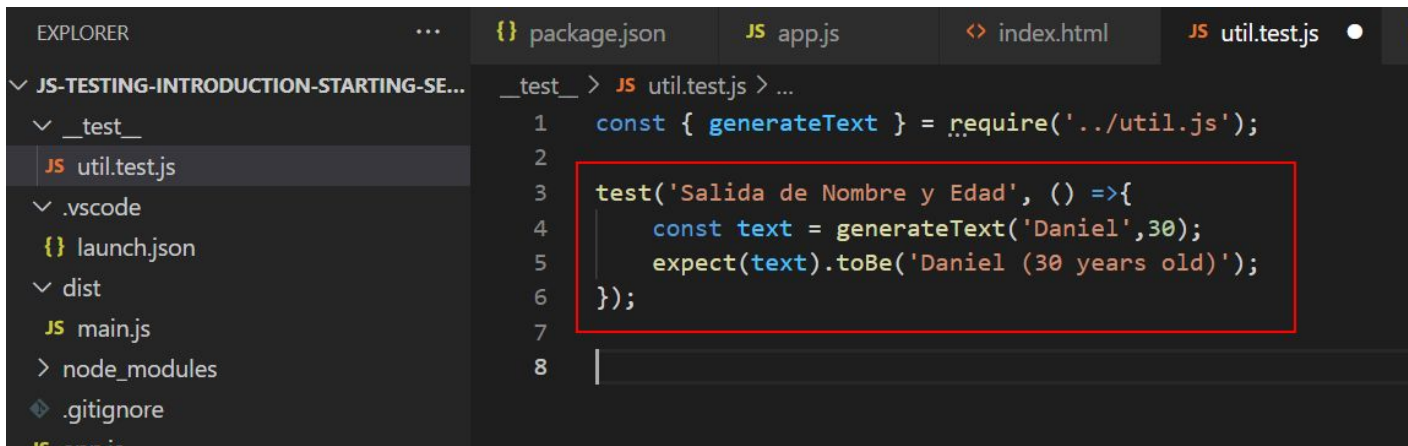
EXPLORER
JS-TESTING-INTRODUCTION-STARTING-SE...
  _test_
    JS util.test.js
  .vscode
    {} launch.json
  dist
    JS main.js
  > node_modules
  .gitignore
  JS app.js
  <> index.html
  {} package-lock.json
  {} package.json

test > JS util.test.js > ...
1 const { generateText } = require('../util.js');
2
3
```

5

Escribir el código del unit test. Las funciones claves a utilizar son:

- **test()** o **it()**: donde se define el test. Se debe ingresar un nombre representativo.
- **expect()**: lo que se desea verificar es parte de la *assertion library*. En el ejemplo se quiere verificar la salida que se genera al presionar el botón **Agregar Usuario**.

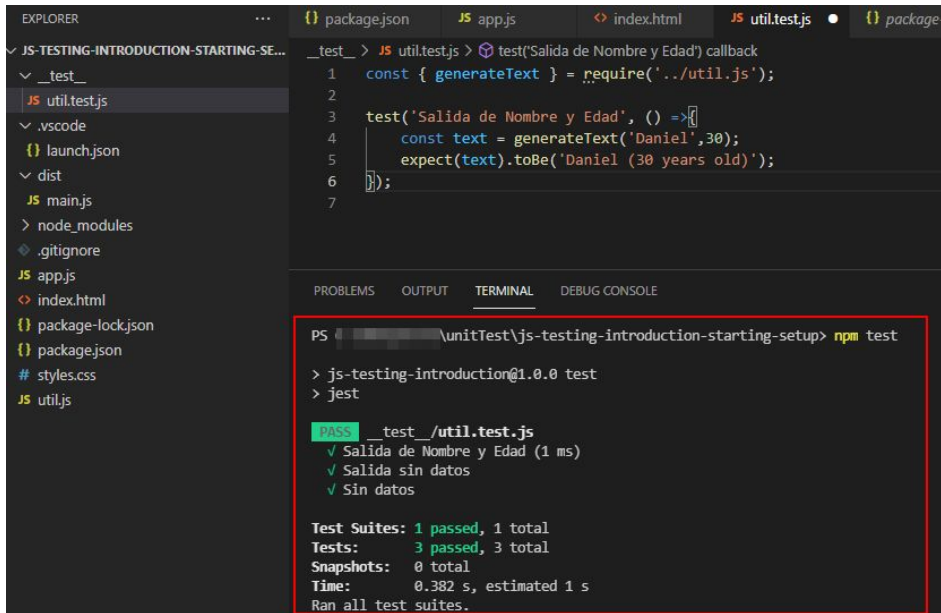


The screenshot shows the VS Code editor interface. The Explorer sidebar on the left shows a project structure with folders like .vscode, dist, and node_modules, and files like package.json, launch.json, main.js, and util.test.js. The main editor area shows the content of util.test.js. The code includes a require statement for '../util.js' and a test function. The test function is highlighted with a red box. The test function is named 'Salida de Nombre y Edad' and takes an empty array as an argument. Inside the test function, a variable 'text' is assigned the value of generateText('Daniel', 30), and then expect(text).toBe('Daniel (30 years old)') is called to verify the output.

```
__test__ > JS util.test.js > ...
1  const { generateText } = require('../util.js');
2
3  test('Salida de Nombre y Edad', () =>{
4      const text = generateText('Daniel',30);
5      expect(text).toBe('Daniel (30 years old)');
6  });
7
8  |
```

6

Para ejecutar el test, primero debemos abrir la terminal y, luego, ejecutar el comando **npm test**.



The screenshot shows the VS Code interface. The Explorer sidebar on the left displays the file structure of a project named 'JS-TESTING-INTRODUCTION-STARTING-SE...'. The file 'util.test.js' is selected. The main editor area shows the content of 'util.test.js', which includes a Jest test suite with three tests: 'Salida de Nombre y Edad', 'Salida sin datos', and 'Sin datos'. The test suite is named 'Salida de Nombre y Edad' and uses the 'jest' runner. The terminal at the bottom shows the command 'npm test' being executed, resulting in a 'PASS' status for the test suite. The terminal output includes the following details:

```
PS C:\Users\user\unitTest\js-testing-introduction-starting-setup> npm test

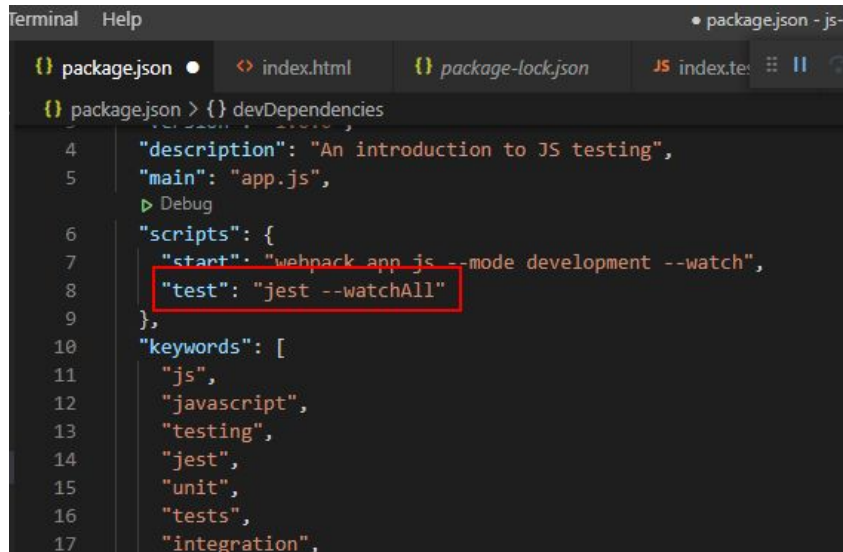
> js-testing-introduction@1.0.0 test
> jest

PASS __test_/util.test.js
  ✓ Salida de Nombre y Edad (1 ms)
  ✓ Salida sin datos
  ✓ Sin datos

Test Suites: 1 passed, 1 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        0.382 s, estimated 1 s
Ran all test suites.
```

7

Se puede configurar que los tests se ejecuten cada vez que se genera un cambio en el código. Para ello, debemos ir al archivo **package.json** e ingresar **jest --watchAll** en el nodo "test" que está dentro de "scripts".



The screenshot shows a code editor with a dark theme. The top bar indicates the file is 'package.json' and the editor is in 'Terminal' mode. The file content is as follows:

```
{} package.json > {} devDependencies
{} package.json > {} devDependencies
4  "description": "An introduction to JS testing",
5  "main": "app.js",
   ▶ Debug
6  "scripts": {
7    "start": "webpack app.js --mode development --watch",
8    "test": "jest --watchAll"
9  },
10 "keywords": [
11   "js",
12   "javascript",
13   "testing",
14   "jest",
15   "unit",
16   "tests",
17   "integration",
```

The line `"test": "jest --watchAll"` is highlighted with a red rectangular box.

Luego, ejecutar el siguiente comando: **npm test**.

Allí, se debe realizar alguna modificación en el código del programa (**util.js**) y, por último, al momento de guardar el archivo modificado, se ejecutarán los tests.

```
EXPLORER package.json JS app.js index.html JS util.test.js package-lock
JS TESTING-INTRODUCTION-STARTING-SE...
  _test_
  util.test.js
  .vscode
  launch.json
  dist
  JS main.js
  node_modules
  .gitignore
  JS app.js
  index.html
  package-lock.json
  package.json
  styles.css
  JS util.js

JS util.js generateText generateText
1 exports.generateText = (name, age) => {
2   // Returns output text
3   return `${name} (${age} años)`;
4 }
5
6 exports.createElement = (type, text, className) => {
7   // Creates a new HTML element and returns it
8   const newElement = document.createElement(type);
9   newElement.classList.add(className);
10  newElement.textContent = text;
11 }

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
[Error] _test_/util.test.js
  Salida de Nombre y Edad (1 ms)
  • Salida de Nombre y Edad

expect(received).toBe(expected) // Object.is equality

Expected: "Daniel (30 years old)"
Received: "Daniel (30 años)"

3 | test('Salida de Nombre y Edad', () =>{
4 |   const text = generateText('Daniel',30);
5 |   expect(text).toBe('Daniel (30 years old)');
6 | });
7 |

at Object.<anonymous> (_test_/util.test.js:5:18)

Test Suites: 1 failed, 1 total
Tests: 1 failed, 1 total
Snapshots: 0 total
Time: 0.242 s, estimated 1 s
Ran all test suites.

Watch Usage
  > Press f to run only failed tests.
  > Press o to only run tests related to changed files.
  > Press p to filter by a filename regex pattern.
  > Press t to filter by a test name regex pattern.
  > Press q to quit watch mode.
  > Press i to run failing tests interactively.
  > Press Enter to trigger a test run.
```


3 | Agrupar unit tests

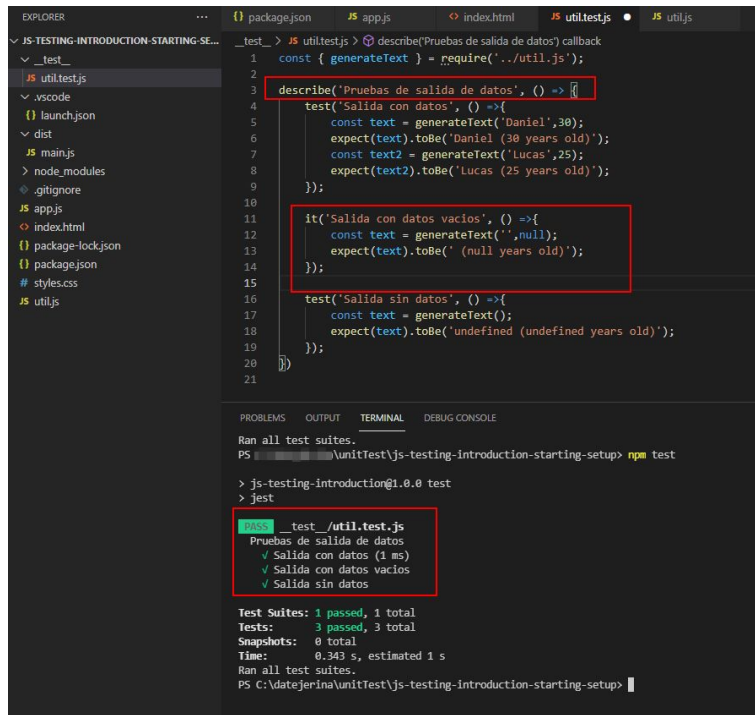
Agrupar unit tests

8

Para generar agrupaciones de unit tests se utiliza la siguiente palabra clave:

describe()

Esto será útil para organizar las pruebas de acuerdo a las funcionalidades a probar.



The screenshot shows a VS Code editor with a file named `util.test.js` open. The file contains three test blocks, each enclosed in a red box:

```
describe('Pruebas de salida de datos', () => {  
  test('Salida con datos', () => {  
    const text = generateText('Daniel', 30);  
    expect(text).toBe('Daniel (30 years old)');  
    const text2 = generateText('Lucas', 25);  
    expect(text2).toBe('Lucas (25 years old)');  
  });  
  
  it('Salida con datos vacios', () => {  
    const text = generateText('', null);  
    expect(text).toBe(' (null years old)');  
  });  
  
  test('Salida sin datos', () => {  
    const text = generateText();  
    expect(text).toBe('undefined (undefined years old)');  
  });  
});
```

Below the editor, the terminal shows the command `npm test` being executed. The output displays the test results for `util.test.js`, with the test suite name and file path highlighted in a red box:

```
PS C:\datejerina\unitTest\js-testing-introduction-starting-setup> npm test  
  
> js-testing-introduction@1.0.0 test  
> jest  
  
PASS _test_/util.test.js  
  Pruebas de salida de datos  
    ✓ Salida con datos (1 ms)  
    ✓ Salida con datos vacios  
    ✓ Salida sin datos  
  
Test Suites: 1 passed, 1 total  
Tests: 3 passed, 3 total  
Snapshots: 0 total  
Time: 0.343 s, estimated 1 s  
Ran all test suites.  
PS C:\datejerina\unitTest\js-testing-introduction-starting-setup>
```

DigitalHouse>