Jest Workshop

# Set up a new Jest project using npm and create a simple directory structure

**Step 1: Initialize a New npm Project**

Open a terminal and navigate to the directory where you want to create your Jest project.

# Navigate to the desired directory

cd path/to/your/directory

# Initialize a new npm project

npm init -y

This will create a **package.json** file with default values.

**Step 2: Install Jest**

Install Jest as a development dependency using npm:

npm install --save-dev jest

This command installs Jest and adds it to the **devDependencies** section in your **package.json** file.

**Step 3: Create a Simple Directory Structure**

Create a basic directory structure for your Jest project. You can customize this structure based on your project's needs.

# Create directories

src

\_\_tests\_\_

# Create sample files

src/myFunction.js

\_\_tests\_\_/myFunction.test.js

**Step 4: Write a Simple Function**

Open the **src/myFunction.js** file and write a simple JavaScript function. For example:

// src/myFunction.js

function add(a, b) {

return a + b;

}

module.exports = add;

**Step 5: Write a Jest Test**

Open the **\_\_tests\_\_/myFunction.test.js** file and write a Jest test for the function.

// \_\_tests\_\_/myFunction.test.js

const add = require('../src/myFunction');

test('adds 1 + 2 to equal 3', () => {

expect(add(1, 2)).toBe(3);

});

test('adds 0 + 0 to equal 0', () => {

expect(add(0, 0)).toBe(0);

});

test('adds -1 + 1 to equal 0', () => {

expect(add(-1, 1)).toBe(0);

});

**Step 6: Add a Jest Script to package.json**

Open your **package.json** file and add a script for running Jest tests.

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

// package.json

{… // ... other configurations

"scripts": {

"test": "jest"

}

}

**Step 7: Run Jest Tests**

Run Jest tests using the following command:

npm test

This will execute Jest and run the tests you've written.

**Summary:**

You have now set up a new Jest project, created a simple directory structure, written a basic function, and created Jest tests for that function. You will continue to add more functions and tests as your project grows, leveraging the power of Jest for JavaScript testing.

# Setting up a palindrome test with Jest

**Step 1: Create a Palindrome Function**

Create a JavaScript file (e.g., **palindrome.js**) and write a function that checks if a given string is a palindrome.

// palindrome.js

function isPalindrome(str) {

// Remove non-alphanumeric characters and convert to lowercase

const cleanStr = str.replace(/[^a-zA-Z0-9]/g, '').toLowerCase();

// Compare the cleaned string with its reverse

return cleanStr === cleanStr.split('').reverse().join('');

}

module.exports = isPalindrome;

**Step 2: Create a Jest Test File**

Create a Jest test file (e.g., **palindrome.test.js**) to test the **isPalindrome** function.

// palindrome.test.js

const isPalindrome = require('./palindrome');

test('detects a palindrome', () => {

expect(isPalindrome('A man, a plan, a canal, Panama')).toBe(true);

expect(isPalindrome('race a car')).toBe(false);

});

test('handles edge cases', () => {

expect(isPalindrome('')).toBe(true); // An empty string is a palindrome

expect(isPalindrome('a')).toBe(true); // Single characters are palindromes

});

**Step 3: Run Jest Tests**

Run Jest tests using the following command:

npm test

Jest will execute the tests, and you should see the output indicating whether the palindrome tests passed or failed.

**Summary:**

You have now set up a simple palindrome-checking function, written Jest test cases to verify its correctness, and executed the tests using Jest. This example covers basic palindrome scenarios and edge cases, but you can expand the test cases based on your specific requirements.

# Make a separate JS function to Remove non-alphanumeric characters and convert to lowercase

function siivous(str) {

    // Remove non-alphanumeric characters using a regular expression

    const cleanedStr = str.replace(/[^a-zA-Z0-9]/g, '');

    // Convert the cleaned string to lowercase

    const lowercaseStr = cleanedStr.toLowerCase();

    return lowercaseStr;

}

module.exports = siivous;

In this function:

* str.replace(/[^a-zA-Z0-9]/g, ''): This regular expression removes any characters that are not alphanumeric (letters or numbers).
* toLowerCase(): This method converts the resulting string to lowercase.

# Make a Jest test for a JS function that Remove non-alphanumeric characters and convert to lowercase.

// stringUtils.test.js

const cleanAndLowercase = require('../src/siivous');

describe('cleanAndLowercase', () => {

    test('removes non-alphanumeric characters and converts to lowercase', () => {

        const originalString = "Hello, World! 123";

        const result = cleanAndLowercase(originalString);

        expect(result).toBe('helloworld123');

    });

    test('handles empty string', () => {

        const result = cleanAndLowercase('');

        expect(result).toBe('');

    });

    test('handles string with no non-alphanumeric characters', () => {

        const result = cleanAndLowercase('abc123');

        expect(result).toBe('abc123');

    });

    // Add more test cases as needed

});

# Make a separate JS function to reverse a string

function reverseString(str) {

    // Split the string into an array of characters

    const characters = str.split('');

    // Reverse the array

    const reversedCharacters = characters.reverse();

    // Join the characters back into a string

    const reversedString = reversedCharacters.join('');

    return reversedString;

}

module.exports = reverseString;

# Make a Jest test for a JS function that reverse a string.

// reverse.test.js

const reverseString = require('../src/reverse');

describe('reverseString', () => {

    test('reverses a string', () => {

        const originalString = 'Hello, World!';

        const result = reverseString(originalString);

        expect(result).toBe('!dlroW ,olleH');

    });

    test('handles an empty string', () => {

        const result = reverseString('');

        expect(result).toBe('');

    });

    test('handles a string with a single character', () => {

        const result = reverseString('a');

        expect(result).toBe('a');

    });

    // Add more test cases as needed

});