Introduction to Jest

Jest is a delightful JavaScript testing framework maintained by Facebook, with a focus on simplicity. It works with projects using Babel, TypeScript, Node, React, Angular, Vue.js, and more.

Installation

Install Jest using npm:

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

Basic Usage

Add a script in your package.json to run Jest:

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

Run your tests with:

```
npm test
```

Writing Tests

Test Structure Jest uses test or it functions to define tests. Each test suite can have multiple tests.

```
describe("My Test Suite", () => {
  test("should perform some action", () => {
    expect(true).toBe(true);
  });

  it("should perform another action", () => {
    expect(false).toBe(false);
  });
});
```

Matchers

Jest uses matchers to assert values in tests. Some commonly used matchers are:

- toBe (for primitive values)

- toEqual (for objects and arrays)
- toBeNull
- toBeUndefined
- toBeDefined
- toBeTruthy
- toBeFalsy

Example:

- toBe (for primitive values)

```
// toBe is used to compare primitive values
// (e.g., numbers, strings, booleans).
test("two plus two is four", () => {
  expect(2 + 2).toBe(4);
});

test("string comparison", () => {
  expect("hello").toBe("hello");
});

test("boolean comparison", () => {
  expect(true).toBe(true);
});
```

- toEqual (for objects and arrays)

```
//toEqual is used to compare the equality of objects and arrays.
test("object assignment", () => {
  const data = { one: 1 };
  data["two"] = 2;
  expect(data).toEqual({ one: 1, two: 2 });
});

test("array comparison", () => {
  const arr = [1, 2, 3];
  expect(arr).toEqual([1, 2, 3]);
});
```

- toBeNull toBeNull is used to check if a value is null.

```
test("null value", () => {
  const value = null;
  expect(value).toBeNull();
});
```

- toBeUndefined toBeUndefined is used to check if a value is undefined.

```
test("undefined value", () => {
  let value;
  expect(value).toBeUndefined();
});
```

- toBeDefined toBeDefined is the opposite of toBeUndefined and is used to check if a value is defined.

```
test("defined value", () => {
  const value = 42;
  expect(value).toBeDefined();
});
```

- toBeTruthy toBeTruthy is used to check if a value is truthy (i.e., it evaluates to true in a boolean context).

```
test("truthy value", () => {
  const value = "hello";
  expect(value).toBeTruthy();
});

test("non-empty array is truthy", () => {
  const value = [1, 2, 3];
  expect(value).toBeTruthy();
});
```

- toBeFalsy toBeFalsy is used to check if a value is falsy (i.e., it evaluates to false in a boolean context).

```
test("falsy value", () => {
  const value = 0;
  expect(value).toBeFalsy();
});

test("empty string is falsy", () => {
  const value = "";
  expect(value).toBeFalsy();
});
```

Setup and Teardown

Jest provides helper functions to run code before and after tests:

- beforeEach
- afterEach
- beforeAll

- afterAll

```
beforeEach(() => {
  // code to run before each test
});

afterEach(() => {
  // code to run after each test
});

beforeAll(() => {
  // code to run before all tests
});

afterAll(() => {
  // code to run after all tests
});
```

Example:

```
describe("beforeEach example", () => {
  let data;

  // The beforeEach function runs before each test in this suite
  beforeEach(() => {
    // Initialize the data array with [1, 2, 3] before each test
    data = [1, 2, 3];
  });

  // Define the first test case
  test("first test", () => {
    // Modify the data array by adding the number 4
    data.push(4);
    // Check if the data array now equals [1, 2, 3, 4]
    expect(data).toEqual([1, 2, 3, 4]);
  });

  // Define the second test case
  test("second test", () => {
    // Modify the data array by adding the number 5
    data.push(5);
    // Check if the data array now equals [1, 2, 3, 5]
    expect(data).toEqual([1, 2, 3, 5]);
  });
});
```

Mocking

Mock Functions

Mock functions allow you to test the links between code by erasing the actual implementation.

```
const myMock = jest.fn();
myMock.mockReturnValueOnce(10).mockReturnValueOnce("x");

console.log(myMock(), myMock()); // 10, 'x'
```

Mock Modules

You can mock entire modules using jest.mock.

```
jest.mock("axios");
const axios = require("axios");

axios.get.mockResolvedValue({ data: "some data" });
```