

JEST24.1

DocsAPIHelpBlogEnglishSearchGitHub

Expect · Jest

https://jestjs.io/docs/en/expect#expecttextendmatchers

Introduction

Getting Started

Using Matchers

Testing Asynchronous Code

Setup and Teardown

Mock Functions

Jest Platform

Jest Community

More Resources

Guides

Snapshot Testing

An Async Example

Timer Mocks

Manual Mocks

ES6 Class Mocks

expect.extend(matchers)

You can use `expect.extend` to add your own matchers to Jest. For example, let's say that you're testing a number utility library and you're frequently asserting that numbers appear within particular ranges of other numbers. You could abstract that into a `toBeWithinRange` matcher:

```
expect.extend({
  toBeWithinRange(received, floor, ceiling) {
    const pass = received >= floor && received <= ceiling;
    if (pass) {
      return {
        message: () =>
          `expected ${received} not to be within range ${floor} - ${ceiling}`,
        pass: true,
      };
    } else {
      return {
        message: () =>
          `expected ${received} to be within range ${floor} - ${ceiling}`,
        pass: false,
      };
    }
  },
});
```

`.toHaveLength(number)`

`.toHaveReturnedWith(value)`

`.toHaveLastReturnedWith(value)`

`.toHaveNthReturnedWith(nthCall, value)`

`.toBeCloseTo(number, numDigits)`

`.toBeDefined()`

`.toBeFalsy()`

`.toBeGreaterThan(number)`

`.toBeGreaterThanOrEqual(number)`

`.toBeLessThan(number)`

`.toBeLessThanOrEqual(number)`

`.toBeInstanceOf(Class)`

`.toBeNull()`

`.toBeTruthy()`

`.toBeUndefined()`

`.toBeNaN()`

`.toContain(item)`

`.toContainEqual(item)`

`.toEqual(value)`

```
●test("uncle phil", async () => {  
    const response = await BelAir.request("philip");  
  
    expect(response).toHaveFullName("Uncle Phil");  
    expect(response).toBeParent();  
});
```