Jest (2 Units)

Unit 9: Introduction

Overview

Why test:

- Make sure that changes doesn’t break the whole app.
- Help other devs to understand your code.

What is Jest:

Testing Framework

Overview

why use Jest:

- Official supported by FB, creator of React, React native
- Less configuration compares to Mocha
- Support testing React, React Native by default.
- Snapshot testing
- Auto mocking

Getting started with jest:

Install:

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

Overview

Package.json

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

.babelrc

```
{
  "presets": ["es2015", "react"]
}
```

Overview

Basic example

Sum.js

```
function sum(a, b) {
  return a + b;
}
Export default sum;
```

Sum.test.js

```
Import sum from ‘./sum/

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

Overview

Common matchers:
toBe

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

toBe uses === to test exact equality. If you want to check the value of an object, use toEqual instead:
toEqual

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

toEqual recursively checks every field of an object or array.

Overview

toBeNull matches only null

toBeUndefined matches only undefined

toBeDefined is the opposite of toBeUndefined

toBeTruthy matches anything that an if statement treats as true

toBeFalsy matches anything that an if statement treats as false

Overview

```
test('null', () => {
  const n = null;
  expect(n).toBeNull();
  expect(n).toBeDefined();
  expect(n).not.toBeUndefined();
  expect(n).not.toBeTruthy();
  expect(n).toBeFalsy();
});
```

Overview

```
test('zero', () => {
  const z = 0;
  expect(z).not.toBeNull();
  expect(z).toBeDefined();
  expect(z).not.toBeUndefined();
  expect(z).not.toBeTruthy();
  expect(z).toBeFalsy();
});
```

Overview

Numbers:

```
test('two plus two', () => {
  const value = 2 + 2;
  expect(value).toBeGreaterThan(3);
  expect(value).toBeGreaterThanOrEqual(3.5);
  expect(value).toBeLessThan(5);
  expect(value).toBeLessThanOrEqual(4.5);

  // toBe and toEqual are equivalent for numbers
  expect(value).toBe(4);
  expect(value).toEqual(4);
});
```

Overview

For floating point equality, use toBeCloseTo instead of toEqual, because you don't want a test to depend on a tiny rounding error.

```
test('adding floating point numbers', () => {
  const value = 0.1 + 0.2;
  expect(value).not.toBe(0.3);    // It isn't! Because rounding error
  expect(value).toBeCloseTo(0.3); // This works.
});
```

Overview

Strings

```
test('there is no I in team', () => {
  expect('team').not.toMatch(/I/);
});

test('but there is a "stop" in Christoph', () => {
  expect('Christoph').toMatch(/stop/);
});
```

Overview

Arrays

```
const shoppingList = [
  'diapers',
  'kleenex',
  'trash bags',
  'paper towels',
  'beer',
];

test('the shopping list has beer on it', () => {
  expect(shoppingList).toContain('beer');
});
```

Overview

Exceptions

Overview

```
function compileAndroidCode() {
  throw new ConfigError('you are using the wrong JDK');
}

test('compiling android goes as expected', () => {
  expect(compileAndroidCode).toThrow();
  expect(compileAndroidCode).toThrow(ConfigError);

  // You can also use the exact error message or a regexp
  expect(compileAndroidCode).toThrow('you are using the wrong JDK');
  expect(compileAndroidCode).toThrow(/JDK/);
});
```