

Тестирование Jest



Обычно, когда мы пишем функцию, мы легко можем представить, что она должна делать, и как она будет вести себя в зависимости от переданных параметров.

Во время разработки мы можем проверить правильность работы функции, просто вызвав её, например, из консоли и сравнив полученный результат с ожидаемым.

При тестировании кода ручными перезапусками легко упустить что-нибудь важное.

Jest

Это фреймворк для тестирования JavaScript кода

Работает с проектами, использующими Babel, TypeScript, Node, React, Angular, Vue и др.

Установка Jest

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

Jest

sum.js

```
function sum(a, b) {  
  return a + b;  
}  
module.exports = sum;
```

sum.test.js

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

Jest

Добавить в
package.json:

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

Запустить:

```
npm run test
```

Jest

`.toBe()` - проверяет на точное равенство

`.not.toBe()` - противоположность равенству

Jest - логические значения

- `toBeNull()` соответствует только `null`
- `toBeUndefined()` соответствует только `undefined`
- `toBeDefined()` является противоположностью `toBeUndefined`
- `toBeTruthy()` соответствует всему, что `if` инструкция рассматривает как `true`
- `toBeFalsy()` соответствует всему, что `if` инструкция рассматривает как `false`

Jest - числа

- `expect(value).toBeGreaterThan(3);`
- `expect(value).toBeGreaterThanOrEqual(3.5);`
- `expect(value).toBeLessThan(5);`
- `expect(value).toBeLessThanOrEqual(4.5);`

Jest

Для сопоставления строк с регулярными выражениями, используйте `toMatch`

содержит ли массив или итерируемый объект конкретное значение, используя `toContain`

Jest

Документация:

<https://jestjs.io/docs/ru/getting-started>

Другие JS библиотеки для тестов:

- Jasmine - <https://jasmine.github.io/>
- Mocha - <https://mochajs.org/>
- Chai - <https://www.chaijs.com/>