



**MAIL.RU
CLOUD
SOLUTIONS**

Очень странное тестирование

Гафаров Назим
Разработчик интерфейсов

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

Example-based testing

```
const {equal} = require('assert')  
  
const actual = sum(1, 2)  
const expected = 3  
  
equal(actual, expected)
```


Сказочный программист



Сцена из фильма «Даун Хауз», 2001 г.

```
function sum (a, b) {  
    return 3  
}
```

```
equal(  
    sum(4, 8),  
    12  
)
```

```
function sum (a, b) {  
    if (a == 4 && b == 8) return 12  
    return 3  
}
```

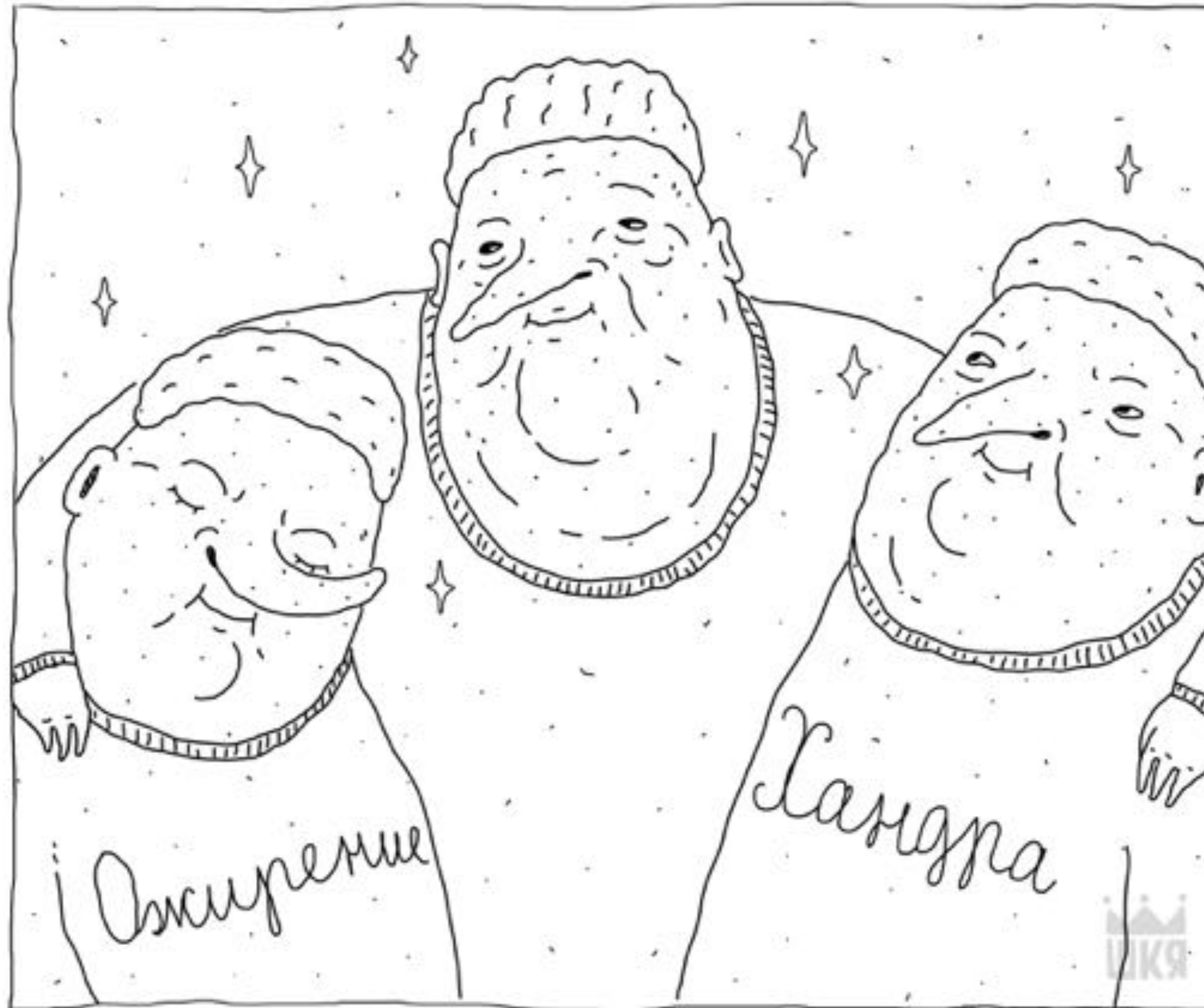


```
const a = Math.random()  
const b = Math.random()  
const actual = sum(a, b)  
const expected = a + b  
  
equal(actual, expected)
```

```
equal( sum(1, 2), 3 )
```

```
equal( sum(4, 8), 12 )
```

The Enterprise Developer From Hell



A + B

Коммутативность

$$A + B = B + A$$


```
const actual = sum(1, 2)
const expected = sum(2, 1)

equal(actual, expected)
```

```
const rand = Math.random
const [n1, n2] = [ rand(), rand() ]

const actual = sum( n1, n2 )
const expected = sum( n2, n1 )

equal(actual, expected)
```

```
function div (dividend, divisor) {  
    return dividend / divisor  
}
```

Дистрибутивность справа

$$(n1 + n2) / n3 = (n1 / n3) + (n2 / n3)$$

```
const [n1, n2, n3] = [rand(), rand(), rand()]

const left = div(n1 + n2, n3)
const right = div(n1, n3) + div(n2, n3)

equal(left, right)
```



```
const [n1, n2, n3] = [0, 0, 0]
```

```
assert.js:85
```

```
  throw new AssertionError(obj);
```

```
    ^
```

```
AssertionError [ERR_ASSERTION]: NaN == NaN
```

```
const [n1, n2, n3] = [2, 1, -347]
```

```
assert.js:85
```

```
  throw new AssertionError(obj);
```

```
    ^
```

```
AssertionError [ERR_ASSERTION]:
```

```
-0.008645533141210375 == -0.008645533141210374
```

```
const [n1, n2, n3] = [rand(), rand(), rand()]

const left = div(n1 + n2, n3)
const right = div(n1, n3) + div(n2, n3)

equal(left, right)
```

Property-based testing

- Генерация случайных входных данных
- Описание ожидаемых свойств у выходных данных
- Запуск в цикле N раз

Как выявлять свойства?

Как выявлять свойства?

Бизнес-требования => Спецификация => Коллекция свойств

$$(\forall x \in X) P(x)$$

`X.every(x => p(x))`

Фреймворки

QuickCheck

- Hypothesis (Python)
- Rubycheck
- ScalaCheck
- Gopter
- JSVerify
- fast-check

JSVerify

```
npm install jsverify
```




```
const jsc = require('jsverify')

jsc.assertForall(
  jsc.integer, jsc.integer,
  (a, b) => a + b === b + a
)
```

```
const subtractionIsCommutative = jsc.checkForall(  
  jsc.integer, jsc.integer,  
  (a, b) => a - b === b - a  
)  
  
console.log(subtractionIsCommutative)
```

```
{  
  counterexample: [ 0, 1 ],  
  tests: 1,  
  shrinks: 4,  
  rngState: '0e168f30eac572b94d'  
}
```

Детерминированная случайность

```
mocha test.js \  
--jsverifyRngState 0e168f30eac572b94d
```

DSL

```
jsc.assertForall(  
    'integer', 'integer',  
    (a, b) => a + b === b + a  
)
```


DSL

```
jsc.assert(jsc.forall(  
    'integer', 'nat', 'number', 'bool', 'falsy', 'char',  
(int, nat, num, bool, fls, chr) => {  
    console.log(int, nat, num, bool, fls, chr)  
    return true  
}  
))
```


DSL

```
jsc.assert(jsc.forall(  
  '{ name: asciinestring; age: nat }',  
  (obj) => {  
    console.log(obj) // { name: '91fpy', age: 34 }  
    return true  
  })  
))
```

```
jsc.record({  
  name: jsc.asciinestring,  
  age: jsc.nat,  
})
```

```
jsc.integer(-2, 2),  
jsc.nat(5),  
jsc.elements([true, null]),
```

```
const emailGenerator = jsc
  .asciiinestring.generator
  .map(str => `${str}@example.com`)
```

В реальной жизни

left-pad

```
leftPad('foo', 5)  
  
// => "   foo"
```


left-pad

```
> npm i left-pad
```

↓ weekly downloads

2 729 866



version

1.3.0

license

WTFPL

```

test('spaces for ch', function (assert) {
  assert.plan(12);
  // default to space if not specified
  assert.strictEqual(leftPad('foo', 2), 'foo');
spaces for ch', function (assert) {
  plan(7);
  strictEqual(leftPad(1, 2, 0), '01');
  strictEqual(leftPad(1, 2, '-'), '-1');
  strictEqual(leftPad('foo', 4, '*'), '*foo', '0b1 len');
  strictEqual(leftPad('foo', 5, '*'), '**foo', '0b10 len');
  strictEqual(leftPad('foo', 6, '*'), '***foo', '0b11 len');
  strictEqual(leftPad('foo', 7, '*'), '****foo', '0b001 len');
  strictEqual(leftPad('foo', 103, '*'), '*****');

  assert.strictEqual(leftPad('foo', 5, ' '), '   foo');
  assert.strictEqual(leftPad('foo', 12, ' '), '          foo');
  assert.strictEqual(leftPad('foo', 13, ' '), '         foo');
});

```


left-pad

```
assert.strictEqual(
  leftPad('foobar', 8, false),
  'foohar'
)
assert.strictEqual(
  leftPad(0, 3, 1),
  '110',
  'integer for str is converted to string'
)
```

```
fc.property(  
  fc.fullUnicodeString(), fc.nat(100),  
  (str, additionalPad) =>  
    length( leftPad(str, length(str) + additionalPad) ) ===  
    length(str) + additionalPad  
)
```

Wrong size when left padding a unicode string #58

 Closed

dubzzz opened this issue on 27 Mar 2018 · 3 comments



dubzzz commented on 27 Mar 2018

Contributor

...

There is an inconsistency when padding strings containing unicode characters out of BMP plan (ie. code points encoded on two chars in UTF-16).

```
leftPad('a\u{1f431}b', 4, 'x') => 'a\u{1f431}b' // in: 3 code points, out: 3 code points
leftPad('abc', 4, '\u{1f431}') => '\u{1f431}abc' // in: 3 code points, out: 4 code points
```

You should maybe specify that left-pad does not handle code points out of BMP plan as single characters.

Failure found using property based testing:

<https://runkit.com/dubzzz/5ab9f3d8cc861f0012852eff>


```
'a\u{1f431}b'.padStart(4, 'x') => "a🐱b"  
'abc'.padStart(4, '\u{1f431}') => "\ud83dabc"
```


query-string

install

```
> npm i query-string
```

↓ weekly downloads

4 802 132



version

6.5.0

license

MIT

packages depending on **query-string**

npm

A package manager for JavaScript

zkat published 6.9.0 • 2 months ago

```
queryString.parseUrl('https://foo.bar?foo=bar')  
//=> {url: 'https://foo.bar', query: {foo: 'bar'}}
```

```
queryString.parse('likes=cake&likes=icecream&name=bob')  
//=> {likes: ['cake', 'icecream'], name: 'bob'}
```

```
queryString.stringify({b: 1, c: 2, a: 3})  
//=> 'b=1&c=2&a=3'
```

```
test('array order', t => {
  t.is(queryString.stringify({
    abc: 'abc',
    foo: ['baz', 'bar']
  }), 'abc=abc&foo=baz&foo=bar');
});
```

```
test('handle empty array value', t => {
  t.is(queryString.stringify({
    abc: 'abc',
    foo: []
  }), 'abc=abc');
});
```

```
test('should not encode undefined values', t => {
  t.is(queryString.stringify({
    abc: undefined,
```

```
> {
  bar'), {foo: 'bar'}));
```

```
fy({foo: 'bar'}), 'foo
fy({
```

```
), {a: [null, null]});
a'), {a: ['', null]});
```



```
fc.property(  
  queryParamsArbitrary, optionsArbitrary,  
  (obj, opts) => deepEqual(  
    m.parse(m.stringify(obj, opts), opts),  
    obj  
  )  
)
```

```
const opts = {arrayFormat: 'bracket'}

m.stringify({bar: ['a', null, 'b']}, opts)
//=> "bar[]=a&bar&bar[]=b"

m.parse('bar[]=a&bar&bar[]=b', opts)
//=> {bar: [null, 'b']}
```

Популярные свойства для тестирования

Инверсия

```
const string = 'ANY_STRING'  
const encrypted = encrypt(string)  
  
expect( decrypt(encrypted) ).toBe( string )
```


Инверсия

```
const obj = {any: 'object'}  
  
_.isEqual(  
  JSON.parse( JSON.stringify(obj) ),  
  obj,  
)
```

Обратимость

```
_.isEqual(  
  array.reverse().reverse(),  
  array,  
)
```

Инвариантность

```
equal(  
    array.sort().length,  
    array.length,  
)
```

Идемпотентность

```
_.isEqual(  
  array.sort().sort(),  
  array.sort(),  
)
```

Идемпотентность

```
string.padStart(10) ===  
string.padStart(10).padStart(10)
```

Идемпотентность

- Форматирование текста
- Поиск уникальных значений
- Нормализация данных
- Повторное добавление элемента в множество

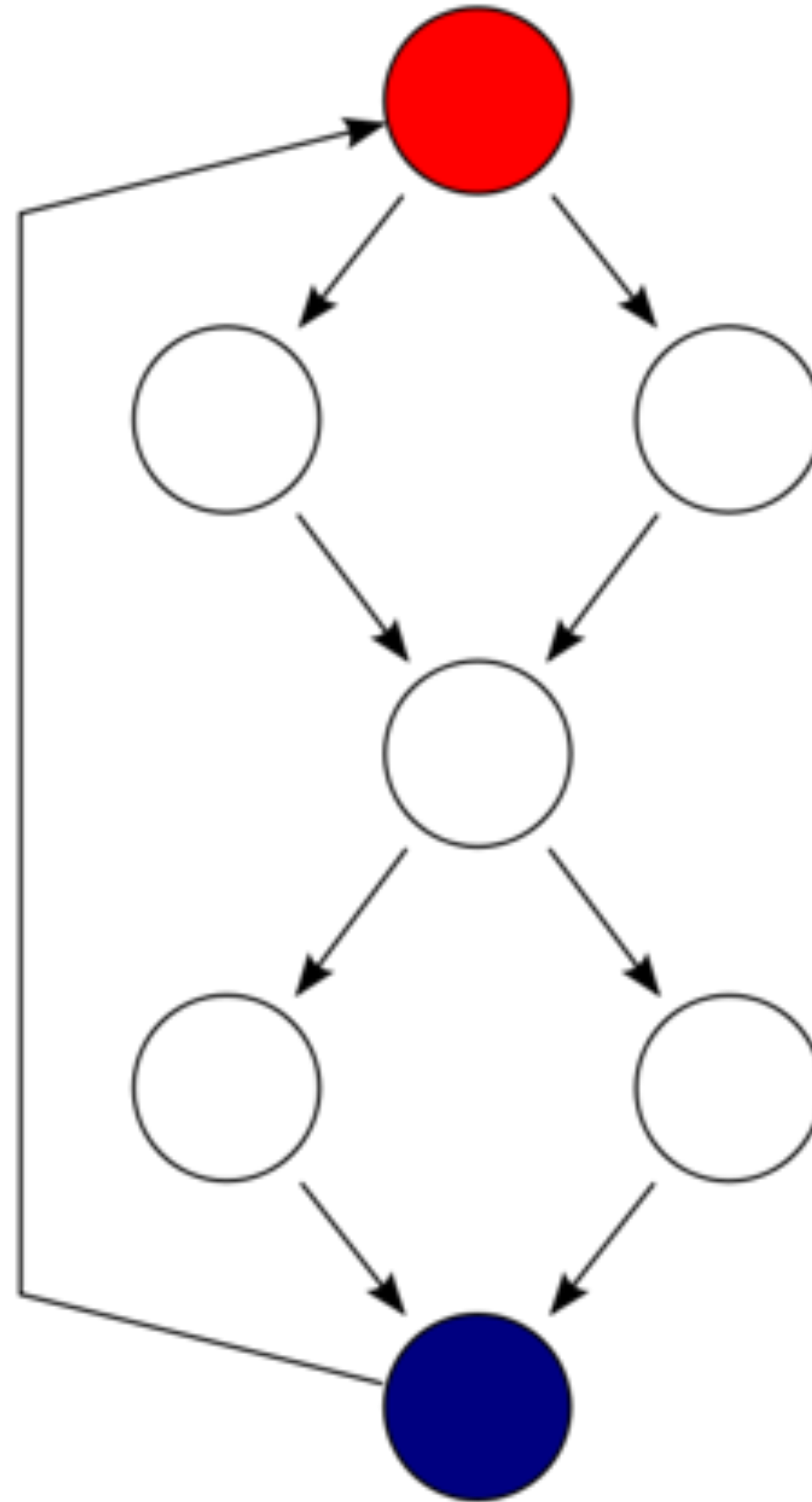
Эталонная реализация

```
_.isEqual(  
  [...array].sort(),  
  fastestSortingAlgorithm(array),  
)
```

Эталонная реализация

```
equal(  
    legacyFunc(data),  
    iBelieveICanFly(data),  
)
```

Только не падай



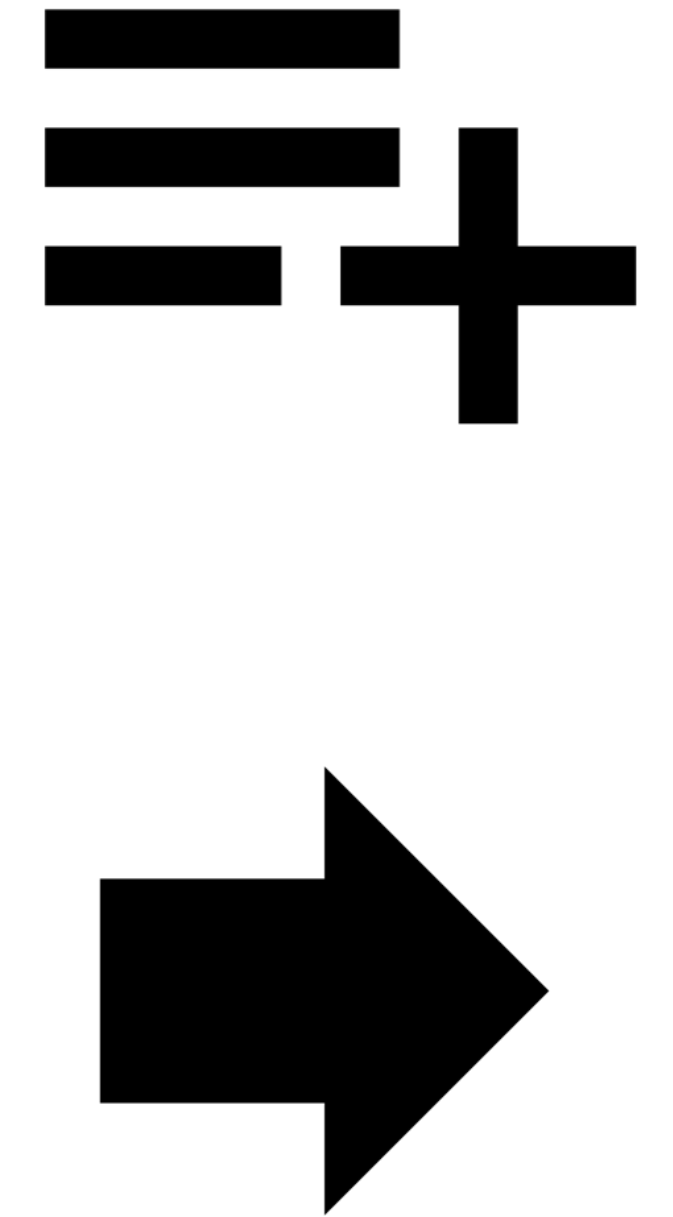
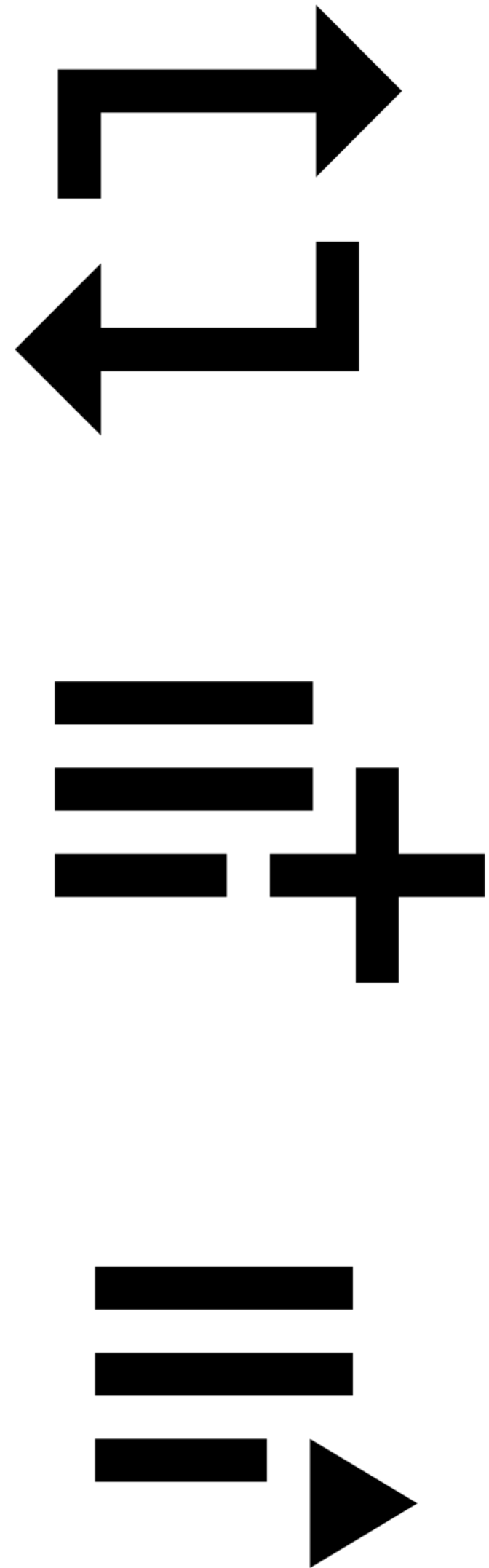
Тестирование UI



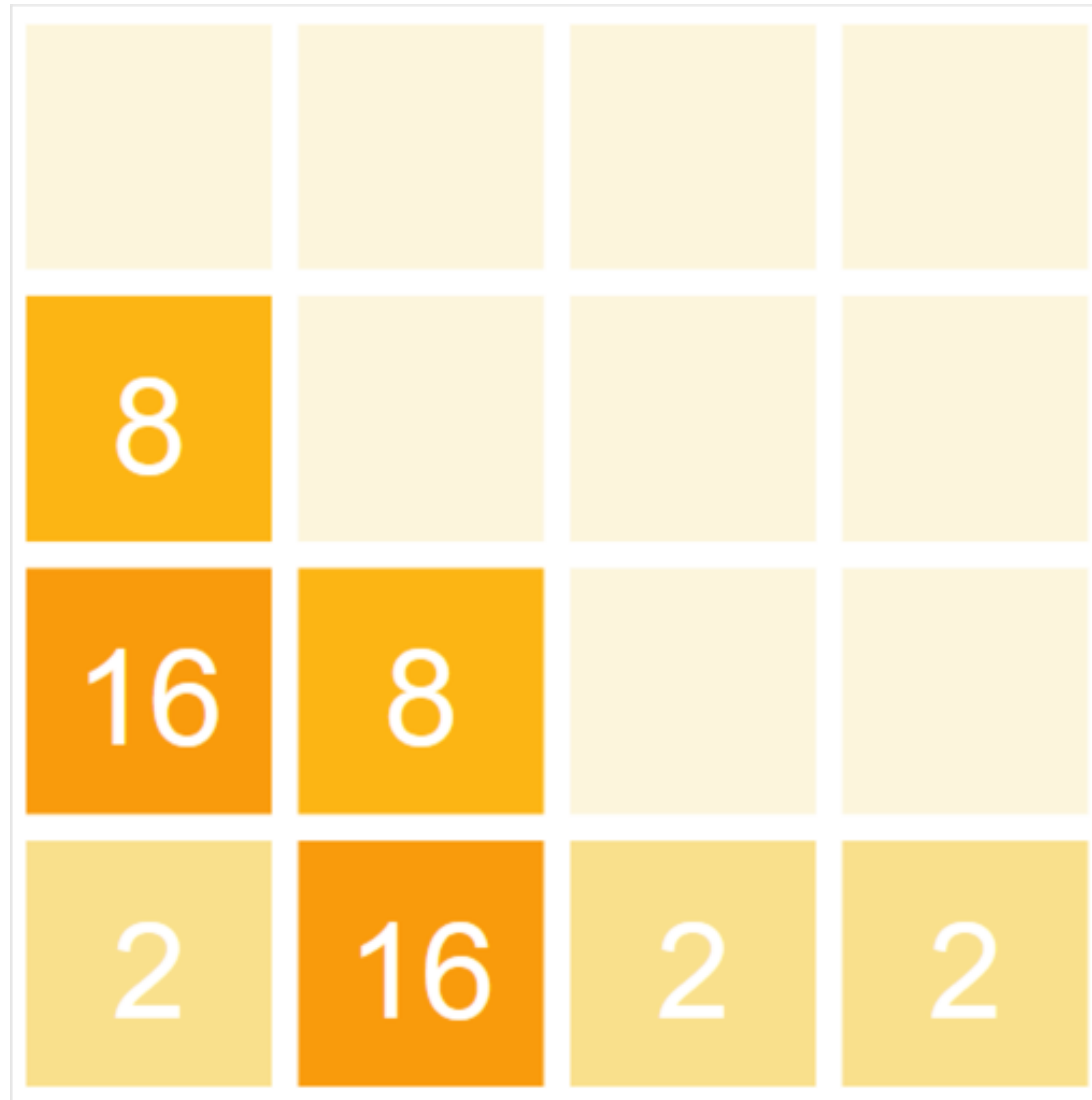
Корзина ≥ 0

Корзина \leq Каталог

$\text{sum}(\text{Корзина}) \geq \text{max}(\text{Товары})$



$$\text{view} = f(\text{model})$$





Итоги

Плюсы

- Лаконичность
- Больше покрытие
- Находят крайние случаи
- Легче поддерживать
- Писать интереснее!
- Заставляют думать

Минусы

- Требуется больше усилий и времени
- Сложнее в понимании
- Увеличивается время выполнения
- Ложное ощущение безопасности

Когда использовать

- суперважные функции
- валидаторы
- агрегаторы
- мапперы
- редьюсеры

Выводы

- Пишите тесты
- Не пишите тесты
- Пишите тесты на основе свойств

СПАСИБО ЗА ВНИМАНИЕ!

Гафаров Назим

 ngafarov