JEST

can do

whaaat?

@robinpokorny

React Open Source Berlin
1 November 2017

JES

can

who

INFO

Slides accompany a talk.

Here, the talk is missing.

For the full experience
see the recording.

I welcome any feedback!

@robinpokorny

React Open Source Berlin
1 November 2017

WHAT IS JEST

delightful, zero configuration testing platform

Jasmine's and Expect's successor

practical utilities for awesome DX

http://facebook.github.io/jest/


SNAPSHOT TESTING

{JSON}

{JSON}


Polaroid
OneStep
Polaroid Land Camera


A

{JSON}


Polaroid
OneStep
Polaroid Land Camera


B

{JSON}


Polaroid
OneStep
Polaroid Land Camera


A

≠ !

{JSON}


Polaroid
OneStep
Polaroid Land Camera


B

TEST

```
describe('method', () ⇒ {
  test('works', () ⇒ {
    const full = {
      number: 1975,
      bool: true,
      string: 'Hello, Vanek',
      promise: Promise.resolve('Better job'),
      symbol: Symbol('brewery'),
      [Symbol.for(‘brewmaster')]: 'Next beer!',
      …
    };

    expect(full).toMatchSnapshot();
    expect(1936).toMatchSnapshot();
  });
});
```

SNAPSHOT

```
// Jest Snapshot v1, https://goo.gl/fbAQLP

exports[`method works 1`] = `
Object {
  "bool": true,
  …
  "undefined": undefined,
  Symbol(brewmaster): "Next beer!",
}
`;

exports[`method works 2`] = `1936`;
```

SNAPSHOT

```
Object {
  "bool": true,
  "func": [Function],
  "map": Map {
    "position1" ⇒ "workman",
    "position2" ⇒ "stockkeeper",
  },
  "null": null,
  "number": 1975,
  "promise": Promise {},
```

SNAPSHOT

```
  "set": Set {
    "think",
    "write",
    "snitch",
  },
  "string": "Hello, Vanek",
  "symbol": Symbol(brewery),
  "undefined": undefined,
  Symbol(brewmaster): "Next beer!",
}
`
```

TEST

```
describe('method', () ⇒ {
  test('works', () ⇒ {
    const full = { … };

    expect(full).toMatchSnapshot('new name');
 });
});
```

SNAPSHOT

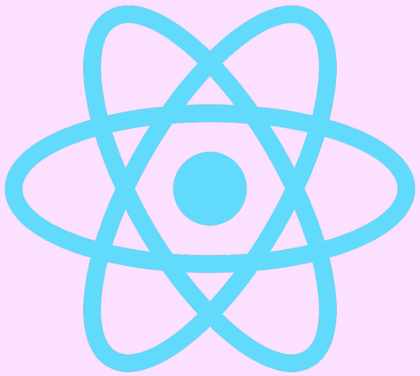
```
exports[`new name 1`] = `…`
```

TEST

```
test('mock function', () ⇒ {
  const fn = jest.fn();

  fn('Vanek');
  fn('Ferdinand', 'Vanek');

  expect(fn).toHaveBeenCalledWith('Vanek');
  expect(fn).toHaveBeenCalledWith('Ferdinand', 'Vanek');
  expect(fn).toHaveBeenCalledTimes(2);

  // vs

  expect(fn.mock.calls).toMatchSnapshot();
});
```

SNAPSHOT

```
exports[`method mock function 1`] = `
Array [
  Array [
    "Vanek",
  ],
  Array [
    "Ferdinand",
    "Vanek",
  ],
]
`;
```



IMMUTABLE

?

SNAPSHOT

```
exports[`component with defaults`] = `
<div
  style={
    Object {
      "backgroundColor": "black",
      "display": "flex",
      "flexDirection": “column",
    }
  }>
  <h1>
…
```

TEST

```
expect(Immutable.Map({ a: 1, b: 2 })).toMatchSnapshot();
```

SNAPSHOT

```
exports[`method immutable 1`] = `
Immutable.Map {
  "a": 1,
  "b": 2,
}
`;
```

SERIALISER

```
const plugin = {
  test(val) {
    return val && val.isPlay;
  },
  serialize(val, config, indent, depth, refs, printer) {
    const name = val.constructor.name;
    const newIndent = indent + config.indent;
    return (
      `Play <${val.title}>: ` +
      printer(val.content, config, newIndent, depth++, refs)
    );
  }
};
```

SERIALISER

```
expect.addSnapshotSerializer(plugin);
```

or

```
// package.json
{
  ...
  "jest": {
    "snapshotSerializers": ["plugin.js"]
  }
}
```

TEST

```
test('play', () ⇒ {
  const play = {
    isPlay: true,
    title: 'Audience',
    content: { scenes: 1 }
  };

  expect([ play ]).toMatchSnapshot();
});
```

SNAPSHOT

```
exports[`method play 1`] = `
Array [
  Play <Audience>: Object {
      "scenes": 1,
    },
]
`;
```

TEST

```
test('diff', () ⇒ {
  const play = {
    title: 'Audience',
    characters: 2
  };

  expect(play).toMatchSnapshot();
  expect({ ...play, characters: 3 }).toMatchSnapshot();
});
```

SNAPSHOT

```
exports[`diff 1`] = `
Object {
  "characters": 2,
  "title": "Audience",
}
`;

exports[`diff 2`] = `
Object {
  "characters": 3,
  "title": "Audience",
}
`;
```

```
const { toMatchDiffSnapshot } = require('snapshot-diff');
expect.extend({ toMatchDiffSnapshot });

test('diff', () ⇒ {
  const play = {
    title: 'Audience',
    characters: 2
  };

  expect(play)
    .toMatchDiffSnapshot({  ...play, characters: 3 });
});
```

SNAPSHOT

```
exports[`diff 1`] = `
"Snapshot Diff:
- First value
+ Second value

  Object {
-   \\"characters\\": 2,
+   \\"characters\\": 3,
    \\"title\\": \\"Audience\\",
  }"
`;
```

TDD × **SNAPSHOTS**

algorithms

write before

part

structures

concurrent or after

whole

inside codebase


Daniel Wellington
watch

```
$ jest --watch

$ npm test -- --watch
```

understands dependencies

*filter by **P**ath or **T**est name*

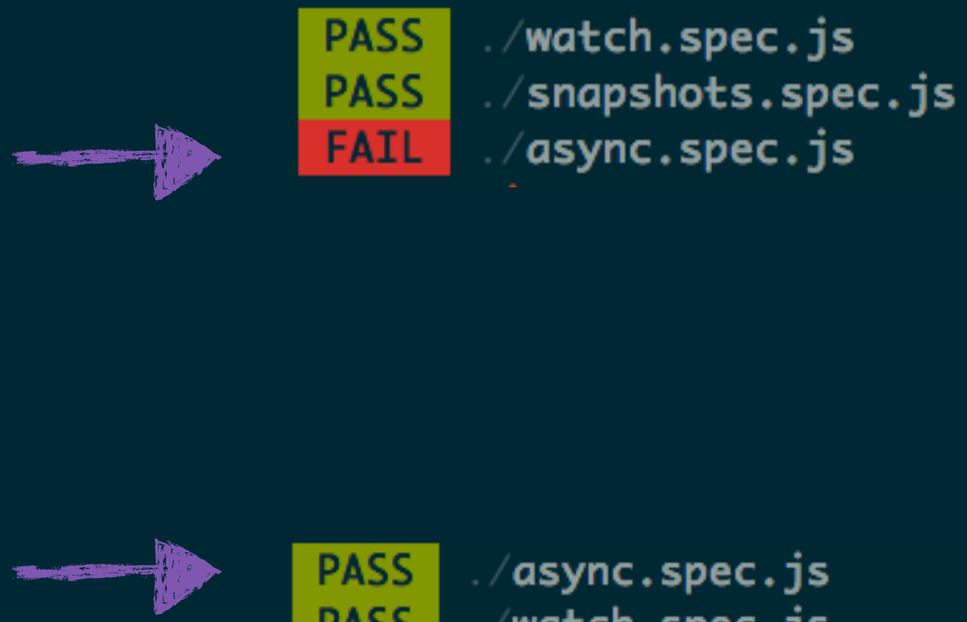
***U**pdate snapshots*

failed re-run first

DEPS

```
// moduleA.spec.js
const moduleA = require('./moduleA');

test('moduleA', () => {
  expect(moduleA()).toBe(true);
});

// moduleA.js
const moduleB = require('./moduleB');
```

```
expect(value).toMatchSnapshot()

Received value does not match stored snapshot 1.

- Snapshot
+ Received

@@ -4,11 +4,11 @@
    "map": Map {
      "position1" => "workman",
      "position2" => "stockkeeper",
    },
    "null": null,
-   "number": 1975,
+   "number": 1976,
    "promise": Promise {},
    "set": Set {
      "think",
      "write",
      "snitch",

  at Object.test (snapshots.spec.js:37:18)
      at new Promise (<anonymous>)
      at <anonymous>

 › 1 snapshot test failed.
Snapshot Summary
 › 1 snapshot test failed in 1 test suite. Inspect your code changes or press `u` to up
date them.
```


PASS ./watch.spec.js
PASS ./snapshots.spec.js
FAIL ./async.spec.js
PASS ./async.spec.js
PASS ./watch.spec.js
PASS ./snapshots.spec.js

ASYNC *in* JEST *testing*

CALLBACK

```
test('promise', done ⇒ {
  Promise.resolve(7)
    .then(n ⇒ {
      expect(n).toBe(7);
    })
    .then(done)
    .catch(done.fail);
});
```

PROMISES

```
test('promises', () ⇒ {
  Promise.resolve(8).then(n ⇒ {
    expect(n).toBe(7);
  });

  return Promise.resolve(7).then(n ⇒ {
    expect(n).toBe(7);
  });
});
```

```
robin at robins-mac in ~/projects/jest-can-do-whaaat on master [?]
$ npm test

> jest-can-do-whaaat@ test /Users/robin/projects/jest-can-do-whaaat
> jest

(node:37199) UnhandledPromiseRejectionWarning: Unhandled promise rejection (rejection id:
1): Error: expect(received).toBe(expected)

Expected value to be (using ===):
  7
Received:
  8
(node:37199) [DEP0018] DeprecationWarning: Unhandled promise rejections are deprecated. In
 the future, promise rejections that are not handled will terminate the Node.js process wi
th a non-zero exit code.
 PASS  ./async.spec.js
 PASS  ./snapshots.spec.js

Test Suites: 2 passed, 2 total
Tests:       8 passed, 8 total
Snapshots:   9 passed, 9 total
Time:        0.318s, estimated 1s
Ran all test suites.
```

```
robin at robins-mac in ~/projects/jest-can-do-whaaat on master [?]
$ npm test

> jest-can-do-whaaat@ test /Users/robin/projects/jest-can-do-whaaat
> jest

(node:37199) UnhandledPromiseRejectionWarning: Unhandled promise rejection (rejection id:
1): Error: expect(received).toBe(expected)

Expected value to be (using ===):
  7
Received:
  8
(node:37199) [DEP0018] DeprecationWarning: Unhandled promise rejections are deprecated. In
 the future, promise rejections that are not handled will terminate the Node.js process wi
th a non-zero exit code.
 PASS  ./async.spec.js
 PASS  ./snapshots.spec.js

Test Suites: 2 passed, 2 total
Tests:       8 passed, 8 total
Snapshots:   9 passed, 9 total
Time:        0.318s, estimated 1s
Ran all test suites.

robin at robins-mac in ~/projects/jest-can-do-whaaat on master [?]
$ echo $?
0
```

REJECTION

```
test('reject', () ⇒ {
  return Promise.reject(0).catch(n ⇒ {
    expect(n).toBe(0);
  });
});
```

REJECTION

```
test('reject', () => {
  return Promise.reject(0).catch(n => {
    expect(n).toBe(0);
  });
});

test('reject', () => {
  return Promise.resolve(7).catch(n => {
    expect(n).toBe(0);
  });
});
```

REJECTION

```
test('reject', () ⇒ {
  expect.assertions(1);

  return Promise.resolve(7)
    .then(() ⇒ {
      throw new Error('Not rejected!');
    })
    .catch(n ⇒ {
      expect(n).toBe(0);
    });
});
```

```
FAIL ./async.spec.js
  • reject

    expect(received).toBe(expected)

    Expected value to be (using ===):
      0
    Received:
      [Error: Not rejected!]

    Difference:

      Comparing two different types of values. Expected number but received object.

      at Promise.resolve.then.catch.n (async.spec.js:29:15)
          at <anonymous>
      at process._tickCallback (internal/process/next_tick.js:188:7)

PASS ./snapshots.spec.js

Test Suites: 1 failed, 1 passed, 2 total
Tests:       1 failed, 7 passed, 8 total
Snapshots:   9 passed, 9 total
Time:        0.167s, estimated 1s
Ran all test suites.
```

ASYNC AWAIT

```
test('async', async () ⇒ {
  const n = await Promise.resolve(7)
  const m = await Promise.resolve(42)

  expect(n).toBe(7)
  expect(m).toBe(42)
})
```

ASYNC AWAIT

```
test('async', async () ⇒ {
  const n = Promise.resolve(7)
  const m = await Promise.resolve(42)

  expect(n).toBe(7)
  expect(m).toBe(42)
})
```

```
FAIL ./async.spec.js
  • async

    expect(received).toBe(expected)

    Expected value to be (using ===):
      7
    Received:
      {}

    Difference:

      Comparing two different types of values. Expected number but received object.

      at Object.<anonymous>.test (async.spec.js:37:13)
          at <anonymous>
      at process._tickCallback (internal/process/next_tick.js:188:7)

PASS ./snapshots.spec.js

Test Suites: 1 failed, 1 passed, 2 total
Tests:       1 failed, 8 passed, 9 total
Snapshots:   9 passed, 9 total
Time:        0.17s, estimated 1s
Ran all test suites.
```

ASYNC REJECTION

```
test('async rejection', async () => {
  try {
    await Promise.reject(0);
 } catch (e) {
    expect(e).toBe(0);
  }
});
```

ASYNC REJECTION

```
test('async rejection', async () ⇒ {
  try {
    await Promise.resolve(7);
 } catch (e) {
    expect(e).toBe(0);
  }
});
```

ASYNC REJECTION

```
test('async rejection', async () ⇒ {
  try {
    await Promise.resolve(0);
    expect(true).toBe(false);
  } catch (e) {
    expect(e).toBe(0);
  }
});
```

.RESOLVES & .REJECTS

```
test('resolves/rejects', async () ⇒ {
  await expect(Promise.resolve(7)).resolves.toBe(7);

  await expect(Promise.reject(0)).rejects.not.toBe(7);
});
```

```
FAIL ./async.spec.js
  • resolves/rejects

    expect(received).resolves.toBe()

    Expected received Promise to resolve, instead it rejected to value
      0

      at Object.<anonymous> (node_modules/expect/build/index.js:131:11)
          at Generator.throw (<anonymous>)
          at <anonymous>
      at process._tickCallback (internal/process/next_tick.js:188:7)

PASS ./snapshots.spec.js

Test Suites: 1 failed, 1 passed, 2 total
Tests:       1 failed, 10 passed, 11 total
Snapshots:   9 passed, 9 total
Time:        0.14s, estimated 1s
Ran all test suites.
```

JEST

~~*can do*~~ *will do*

whaaat?

MOCK NAME

```
test('mockName', () ⇒ {
  const mockFn = jest.fn();

  expect(mockFn).toHaveBeenCalled();
});
```

- mockName

```
expect(jest.fn()).toHaveBeenCalled()

Expected mock function to have been called.

  at Object.<anonymous>.test (future.spec.js:4:18)
      at new Promise (<anonymous>)
      at <anonymous>
  at process._tickCallback (internal/process/next_tick.js:188:7)
```

MOCK NAME

```
test('mockName', () => {
  const mockFn = jest.fn();

  expect(mockFn).toHaveBeenCalled();
});
```

```
test('mockName', () => {
  const mockFn = jest.fn().mockName('mockedFunction');

  expect(mockFn).toHaveBeenCalled();
});
```

```
1  /**
2   * Sample React Native Snapshot Test
3   */
4  'use strict';
5
6  import 'react-native';
7  import React from 'react';
8  import Intro from '../Intro';
9
10 // Note: test renderer must be required after react-native.
11 import renderer from 'react-test-renderer';
12
13 it('renders correctly', () => {
14   const tree = renderer.create(
15     <Intro />
16   ).toJSON();
17   expect(tree).toMatchSnapshot(); +
18 });
19
20 // These serve as integration tests for the jest-react-native preset.
21 it('renders the ActivityIndicator component', () => {
22   const ActivityIndicator = require('ActivityIndicator');
23   const tree = renderer.create(
24     <ActivityIndicator animating={true} size="small" />
25   ).toJSON();
26   expect(tree).toMatchSnapshot(); +
27 });
28
29 it('renders the Image component', done => {
30   const Image = require('Image');
31   Image.getSize('path.jpg', (width, height) => {
32     const tree = renderer.create(
33       <Image style={{width, height}} +/>
34     ).toJSON();
```

INFO

GIF not supported in PDF

```
$ jest packages/moduleA
```

```
$ jest packages/moduleA --passWithNoTests
```

INFO

GIF not supported in PDF

by @kentcdodds

RELATED

- github.com/robinpokorny/jest-can-do-whaaat
- Jest Snapshots and Beyond by Rogelio Guzman (recording)
- Writing snapshot plugins in the docs
- Effective Snapshot Testing by Kent C. Dodds
- Async testing in Jest (recording)
- Snapshot testing in Jest (recording)
- Async testing Koa with Jest

INFO

Full links in the description

Help others to go

JEST
can do
whaaat?

@robinpokorny