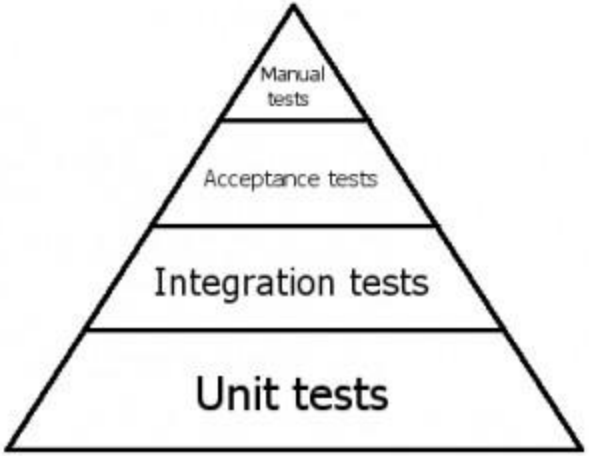
Unit testing in Node JS

Инструменты и лучшие практики

Что такое unit testing



процесс в программировании, позволяющий проверить на корректность отдельные модули исходного кода программы

Пример 1

```
function mapToArray(map) {
 let index = -1
 const result = new Array(map.size)
 map.forEach((value, key) => {
   result[++index] = [key, value]
 })
 return result
}
```

Тест 1

```
it('should return array with key/value from map', () => {
   const testMap = new Map()
   testMap.set('key', 'value')
   const expectedResult = [['key', 'value']]
   const result = mapToArray(testMap)
   assert.equal(expectedResult, result)
 })
```

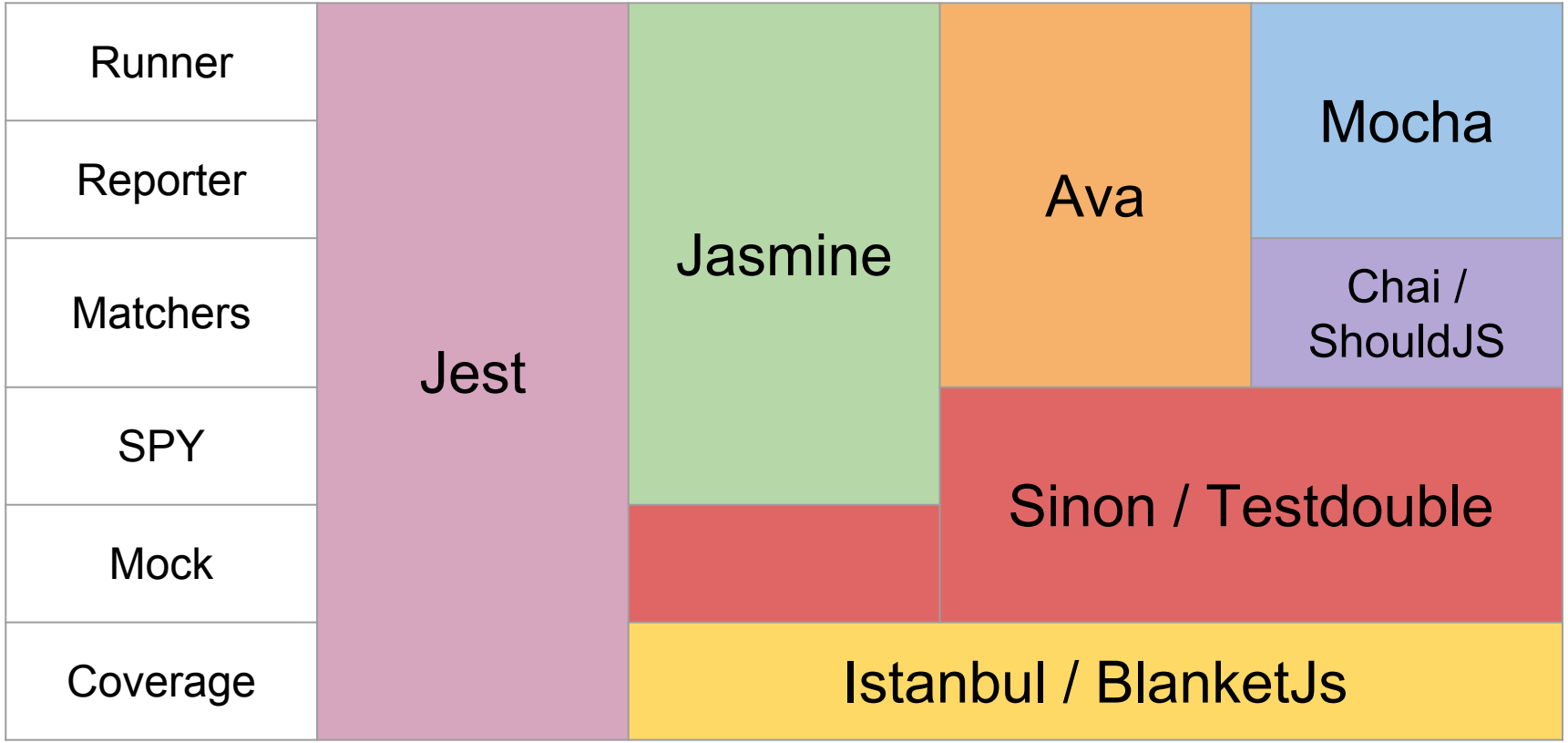
Тест 2

```
it('should return empty array for empty map', () => {
   const testMap = new Map()
   const expectedResult = []
   const result = mapToArray(testMap)
   assert.equal(expectedResult, result)
 })
```

Пример 2

```
const cache = require('./cache')
const db = require('./db')

function userMiddleware(req, res, next) {
 const userId = req.params.id
 const user = cache.get(userId)
 if (!user) {
   db.findUserById(userId).then((result) => {
     res.body = result
     next()
   })
 } else {
   res.body = user
   next()
```

Технологический стек Unit тестов



Mocha

- Используем для структурирования и запуска тестов
- Простая в освоении библиотека
- Содержит общие функции для тестирования (describe, it, assert)
- Поддержка асинхронных тестов
- Hooks
- Reporters
- Изоморфность
- ES6
- и многое другое mochajs.org

Структура тестов с Mocha

```
describe('userMiddleware', () => {
 before(() => {
     // Setup environment and dependencies
 })
 describe('working with cache', () => {
   it('should take value from cache if it exists', () => {
   })
 })
 describe('working with db', () => {
   it('should take value from db if it exists', (done) => {
   })
 })
})
```

Sinon

- mock библиотека для javascript
- Spies
- Stubs
- Mocks
- Готовые плагины для многих фреймворков (sinon-mongoose, sinon-express-mock и т.д.)
- встроенные assert или интеграция с chai
- Sandbox
- sinonjs.org

Работа с зависимостями

```
fakeUserCache = {
        name: faker.internet.userName,
        email: faker.internet.email
      }

dbStub = {
      findUserById: sinon.stub().returns(Promise.resolve(fakeUserFromDb))
    }


cacheStub = {
        get: sinon.stub().returns(fakeUserCache)
      }
```

Proxyquire и Faker

- proxyquire - библиотека для подмены require зависимостей
- работает без изменения исходного кода тестируемого модуля
- частичная подмена зависимости (например только нескольких методов)
- proxyquire

- faker - библиотека для генерации тестовых данных
- поддерживает множество различных типов генерируемых данных (адреса, имена, названия компаний, рандомные значения и т.д.)
- Faker.js

Подмена зависимостей

```
const proxyquire = require('proxyquire').noPreserveCache()

mdl = proxyquire('../src/middleware.js',
        {
          './cache': cacheStub,
          './db': dbStub
        }
      )
```

Chai

- Популярная assert библиотека
- Поддержка различных стилей BDD, TDD
- Огромный выбор assert от простого equal до doesNotHaveAnyDeepKeys
- Возможность расширения с помощью плагинов
- Большое количество уже существующих плагинов
- ES6
- chaijs.com

Тест 1

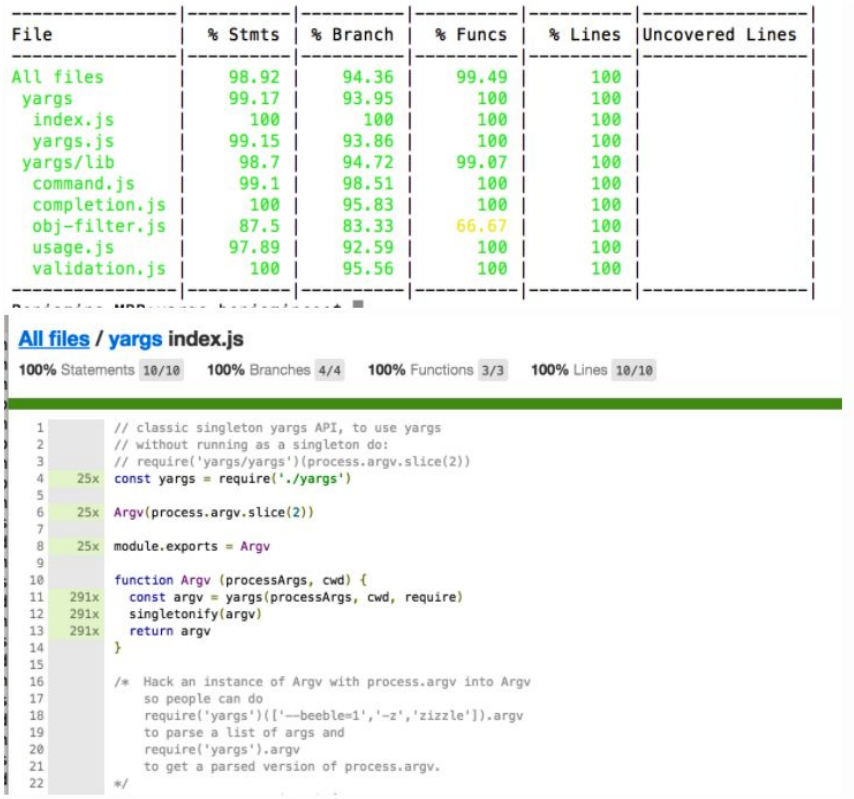
```
it('should take value from cache if it exists', () => {
     mdl.userMiddleware(fakeRequest, fakeResponse, fakeNext)

     sinon.assert.calledOnce(cacheStub.get)
     sinon.assert.calledWith(cacheStub.get, fakeRequest.params.id)
     sinon.assert.calledOnce(fakeNext)

     assert.equal(fakeResponse.body, fakeUserCache)
})
```

Тест 2

```
it('should take value from db if it exists', (done) => {
      let fakeNext
      fakeNext = () => {
        sinon.assert.calledOnce(dbStub.findUserById)
        sinon.assert.calledWith(dbStub.findUserById,
fakeRequest.params.id)
        assert.equal(fakeResponse.body, fakeUserFromDb)
        done()
      }
      mdl.userMiddleware(fakeRequest, fakeResponse, fakeNext)
})
```

Istanbul

- фреймворк для измерения code coverage
- большое количество вариантов отчетов (от консольного вывода до полноценного html отчета)
- интеграция с большим кол-вом фреймворков (webpack, mocha, jasmine)
- поддержка es6 и typescript
- istanbul.js.org



File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
All files	98.92	94.36	99.49	100	
yargs	99.17	93.95	100	100	
index.js	100	100	100	100	
yargs.js	99.15	93.86	100	100	
yargs/lib	98.7	94.72	99.07	100	
command.js	99.1	98.51	100	100	
completion.js	100	95.83	100	100	
obj-filter.js	87.5	83.33	66.67	100	
usage.js	97.89	92.59	100	100	
validation.js	100	95.56	100	100	

All files / yargs index.js

100% Statements 10/10 100% Branches 4/4 100% Functions 3/3 100% Lines 10/10

```
1       // classic singleton yargs API, to use yargs
2       // without running as a singleton do:
3       // require('yargs/yargs')(process.argv.slice(2))
4   25x const yargs = require('./yargs')
5
6   25x Argv(process.argv.slice(2))
7
8   25x module.exports = Argv
9
10      function Argv (processArgs, cwd) {
11  291x  const argv = yargs(processArgs, cwd, require)
12  291x  singletonify(argv)
13  291x  return argv
14      }
15
16      /*  Hack an instance of Argv with process.argv into Argv
17          so people can do
18          require('yargs')(['--beeble=1','-z','zizzle']).argv
19          to parse a list of args and
20          require('yargs').argv
21          to get a parsed version of process.argv.
22      */
```

Оценка покрытия тестами

```
"scripts": {
   "mocha": "mocha",
   "test": "nyc --reporter=lcov --reporter=html mocha"
 }
```

Пример отчета о покрытии

All files middleware.js

100% Statements `11/11` **100%** Branches `2/2` **100%** Functions `2/2` **100%** Lines `11/11`

```
1  2x  const cache = require('./cache')
2  2x  const db = require('./db')
3
4      function userMiddleware(req, res, next) {
5  2x    const userId = req.params.id
6  2x    const user = cache.get(userId)
7  2x    if (!user) {
8  1x      db.findUserById(userId).then((result) => {
9  1x        res.body = result
10 1x        next()
11       })
12     } else {
13 1x      res.body = user
14 1x      next()
15     }
16   }
17
18 2x  module.exports = {
19     userMiddleware
20   }
21
```

Best practices

- Полный контроль над внешними зависимостями
- Тестируем только публичные методы и интерфейсы
- Избегать ветвления внутри тестов (IF, WHILE, ELSE)
- Тест должен быть простым - последовательность вызовов + assert
- Один тест - один assert
- Соблюдать принципы наименования
- Малое время выполнения
- Независимость от окружения выполнения
- Запускаться регулярно
- Иметь метрики для оценки покрытия кода тестами

Недостатки

- Требуют дополнительного времени на создание и поддержку
- Требуют настройки окружения и технологического стека
- Требуют определенной квалификации от разработчиков
- Качество кода - сложный и плохо написанный код тяжело тестировать

Вопросы

???

github.com/kirill3333/jsnn_6