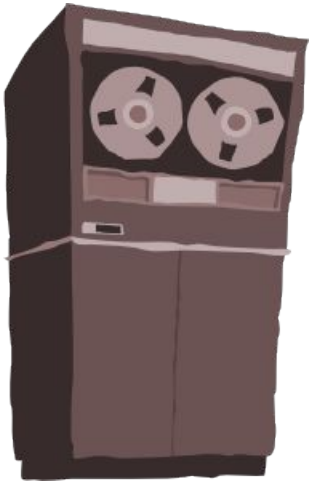
Lesson - 17
Javascript unit testing

SkillUp, by Vitali Cernomschi

План занятия

1. Решение домашнего задания
2. ESLint demo: https://eslint.org/demo/
3. TDD
4. Unit tests
5. Add custom script to NPM
6. Linters: SASS lint

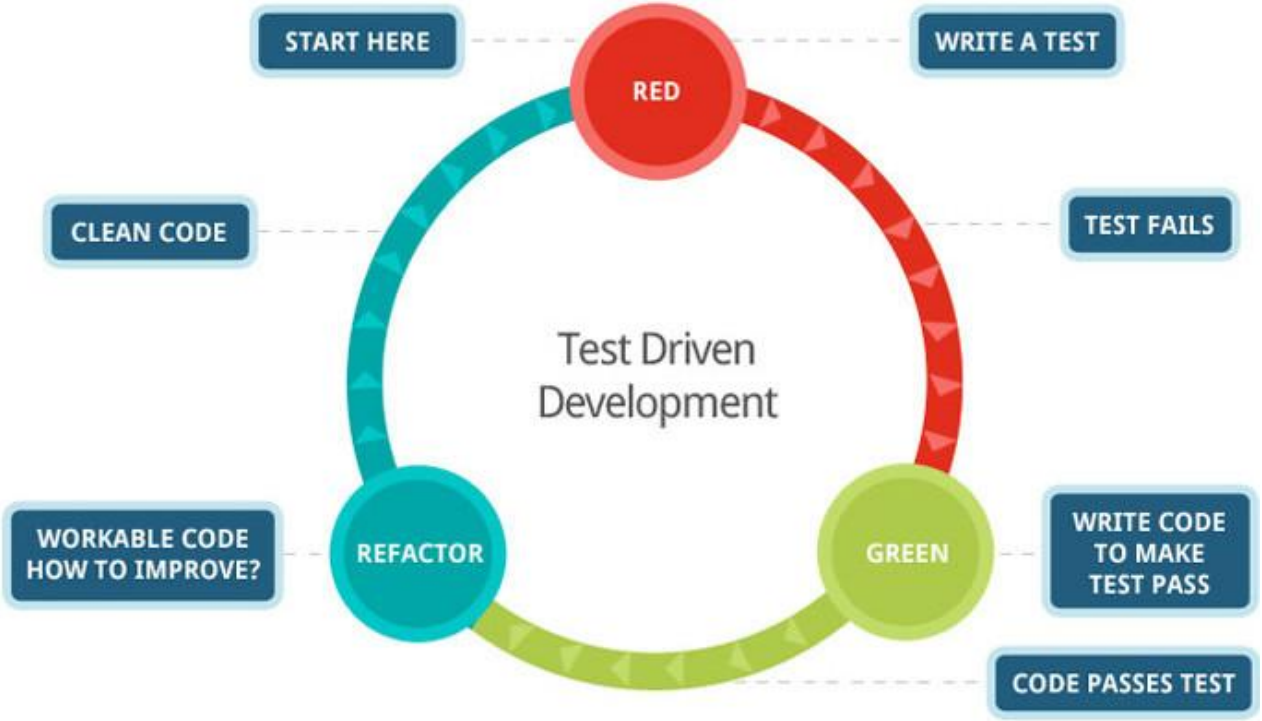
Javascript Unit Testing Frameworks



Test tool types

- Testing structure
- Assertions functions
- Display and watch
- Snapshots
- Mock, spies, stubs
- Code coverage
- Browser like environment

TDD


START HERE
RED
WRITE A TEST
CLEAN CODE
TEST FAILS
Test Driven
Development
WORKABLE CODE
HOW TO IMPROVE?
REFACTOR
GREEN
WRITE CODE
TO MAKE
TEST PASS
CODE PASSES TEST

Grouping test, setup, teardown

```
describe('first set', () => {
  beforeEach(() => {
    //do something
  })
  afterAll(() => {
    //do something
  })
  test(/*...*/)
  test(/*...*/)
})

describe('second set', () => {
  beforeEach(() => {
    //do something
  })
  beforeAll(() => {
    //do something
  })
  test(/*...*/)
  test(/*...*/)
})
```

Stub vs Mock vs Spy

- **Dummy** objects are passed around but never actually used. Usually they are just used to fill parameter lists.
- **Fake** objects actually have working implementations, but usually take some shortcut which makes them not suitable for production (an in memory database is a good example).
- **Stubs** provide canned answers to calls made during the test, usually not responding at all to anything outside what's programmed in for the test.
- **Spies** are stubs that also record some information based on how they were called. One form of this might be an email service that records how many messages it was sent.
- **Mocks** are what we are talking about here: objects pre-programmed with expectations which form a specification of the calls they are expected to receive.

AAA Pattern

- Arrange
- Act
- Assert

NPM scripts

pre- and post-

```
{
  "scripts: {
  "potato": "potato --mash ./index.js"
  "prepotato": "echo SO HUNGRY",
  "postpotato": "echo YUM YUM"
}
```

```
npm run potato
```

```
> prepotato
SO HUNGRY
> potato
mashing...
done.
> postpotato
YUM YUM
```

References

1. JS code quality:
   EN: https://javascript.info/code-quality
   RU: https://learn.javascript.ru/writing-js
4. Recommended rules JavaScript Style Guide:
   https://eslint.org/docs/rules/
5. TDD vs BDD:
   https://joshldavis.com/2013/05/27/difference-between-tdd-and-bdd/
6. Javascript Unit Testing:
   https://designmodo.com/test-javascript-unit/
   https://medium.com/welldone-software/an-overview-of-javascript-testing-in-2018-f68950900bc3
7. Jest:
   https://jestjs.io/
   https://flaviocopes.com/jest/
8. Comparing JavaScript unit testing frameworks:
   https://raygun.com/blog/javascript-unit-testing-frameworks/
9. Stub vs Mock vs Spy:
   https://martinfowler.com/articles/mocksArentStubs.html
10. AAA pattern:
   https://medium.com/@pjbgf/title-testing-code-ocd-and-the-aaa-pattern-df453975ab80
   http://wiki.c2.com/?ArrangeActAssert
11. NPM Scripts:
   https://docs.npmjs.com/misc/scripts
12. SASS Lint:
   https://www.npmjs.com/package/sass-lint

Домашнее задание - теоретическая часть

1. Прочитать Качество кода - Code quality главу:

   EN: https://javascript.info/testing-mocha
   RU: https://learn.javascript.ru/testing

Домашнее задание - практическая часть (* - advanced)

1. Написать unit tests для вычисления дискриминанта.
2. *Написать unit tests для вычисления корней уравнения.

Домашнее задание - advanced теоретическая часть