Let’s talk about unit tests.

js-Montreal

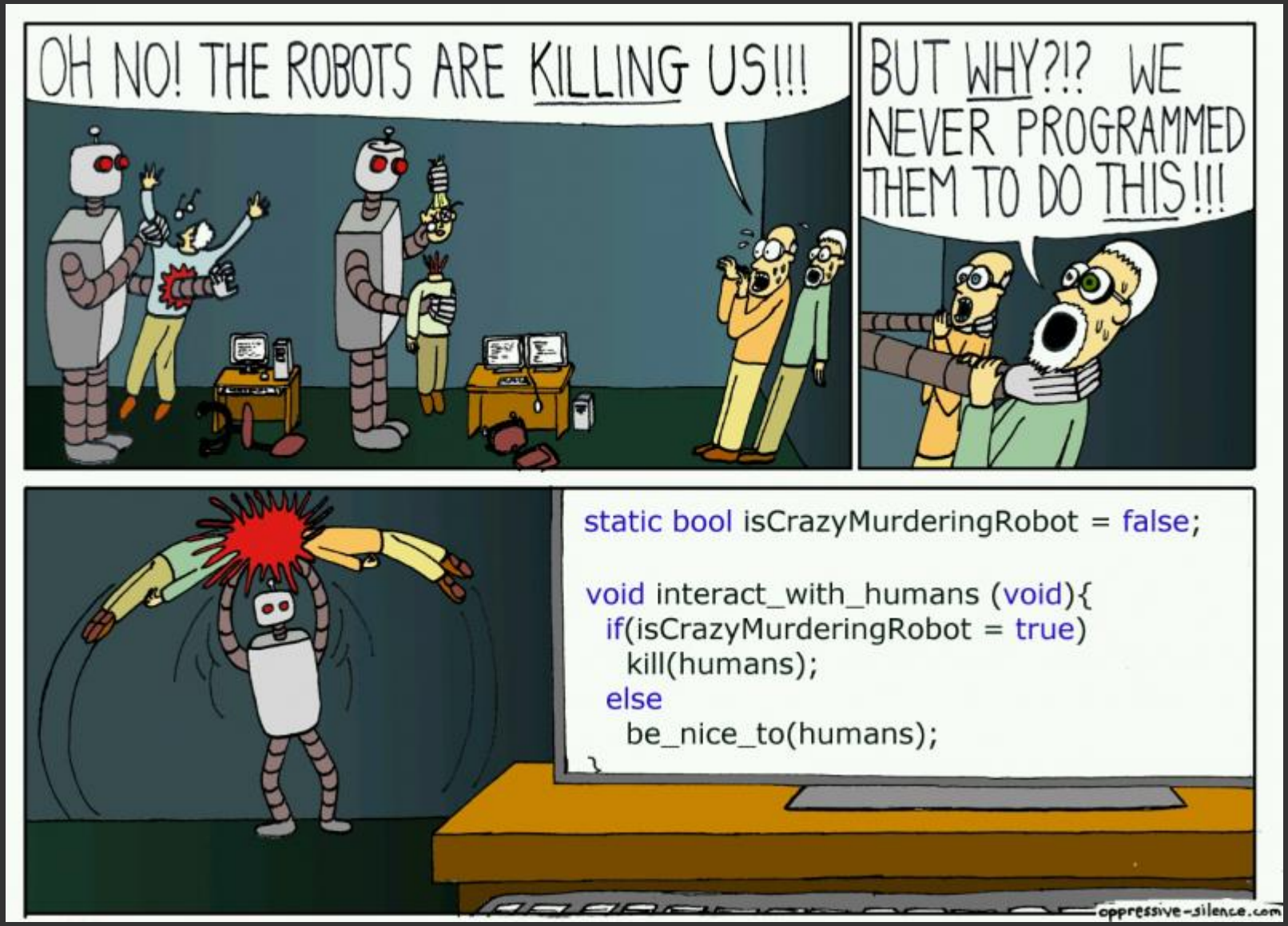
April 2018

Danielo **JEAN-LOUIS**
Front-end developer

What’s unit tests?

*“Method of testing each **unit** of code to **test** if they work well”.*

Why unit tests?



What’s unit tests?

- Avoid dumb tickets / bugs
- Improve code quality / reduce complexity
- Encourage modularity
- Ensure functionalities behaviors even if the code change (!)
- Document code

Sources

- https://medium.com/javascript-scene/what-every-unit-test-needs-f6cd34d9836d

But the code has to be ready for it

Pure function is the first step

Pure function

- No side effects
  - Updates var outside its own scope
- Always return the same results
  - E.g.: Math.random() is impure

Example

Impure function

```
let a = 2;
export const sumImpure = (b) => {
    a = a + b; // Access var outsite func
    return a
}
```

Let’s test it...

with Jest

Jest

- Created by facebook
- Agnostic but made to work well with React
- Inspired by Jasmine
- Allows snapshot tests
- Jest = Jasmine + sinon + istanbul

Sources

- https://facebook.github.io/jest/

Now test our function

Example

Pure function

```
export const sumPure = (a = 0, b) => {
    return a + b;
}
```

It works well but...

Did we manage every case?

Code coverage

- Works hand in hand with unit tests
- Detects useless code
- Forces to manage every case
- Returns the percentage of code executed

- “--coverage” param with Jest
- Jest relies on instanbul for it

Let’s coverage our code

Code coverage details

Statements (Stmts):

Proportion of statements executed

Example:

```
const a = 42; // It's a statement
```

Branches (Branch):

Conditional statements executed (if / else, switch...)

Sources

- http://2ality.com/2012/09/expressions-vs-statements.html
- https://github.com/dwyl/learn-istanbul

Code coverage details

Functions (Funcs):

Proportion of statements executed

Lines (lines):

Proportion of lines executed

Sources

- http://2ality.com/2012/09/expressions-vs-statements.html
- https://github.com/dwyl/learn-istanbul

Let’s improve the coverage

Code coverage details

Functions (Funcs):

Proportion of statements executed

Lines (lines):

Proportion of lines executed

Sources

- http://2ality.com/2012/09/expressions-vs-statements.html
- https://github.com/dwyl/learn-istanbul

100 % it’s not a dream.
It can be a reality.

Unit testing best practices

- Be explicit (You know what you’re testing)
- Don’t try to replace QA / functional tests
- Add them to your precommit
- TDD (Write your test then code)
- Define a naming convention
- Create a mocks folder for your mocked data

Conclusion

- **Unit testing** allows to improve code quality / reduce number of bugs
- **Code coverage** detects useless code
- Unit tests **mustn’t** replace QA. Functional tests exist for this.

Questions ?

More sources

- Presentation + example :
https://github.com/DanYellow/presentations/tree/master/unit-tests