TDD + Code Coverage

Lorenzo Gabriel Pérez González alu0101233499
Pablo Pérez González alu0101318318

Team

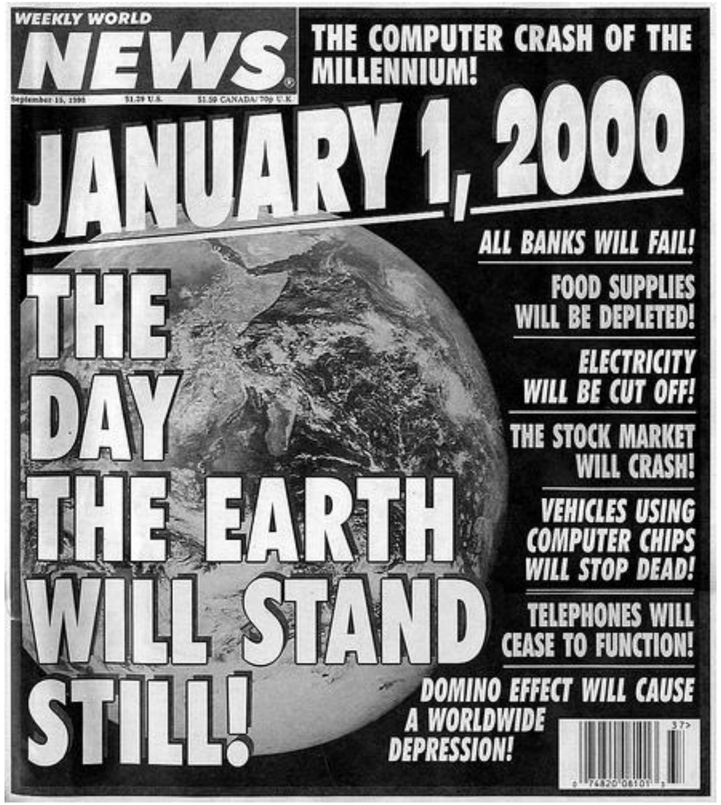
Pablo Pérez González
alu0101318318

Lorenzo Gabriel Pérez González
alu0101233499

Table of Contents

Before we start...

This is really important.



WEEKLY WORLD NEWS

September 15, 1998 $1.29 U.S. $1.59 CANADA/70p U.K.

THE COMPUTER CRASH OF THE MILLENNIUM!

JANUARY 1, 2000

THE DAY THE EARTH WILL STAND STILL!

ALL BANKS WILL FAIL!

FOOD SUPPLIES WILL BE DEPLETED!

ELECTRICITY WILL BE CUT OFF!

THE STOCK MARKET WILL CRASH!

VEHICLES USING COMPUTER CHIPS WILL STOP DEAD!

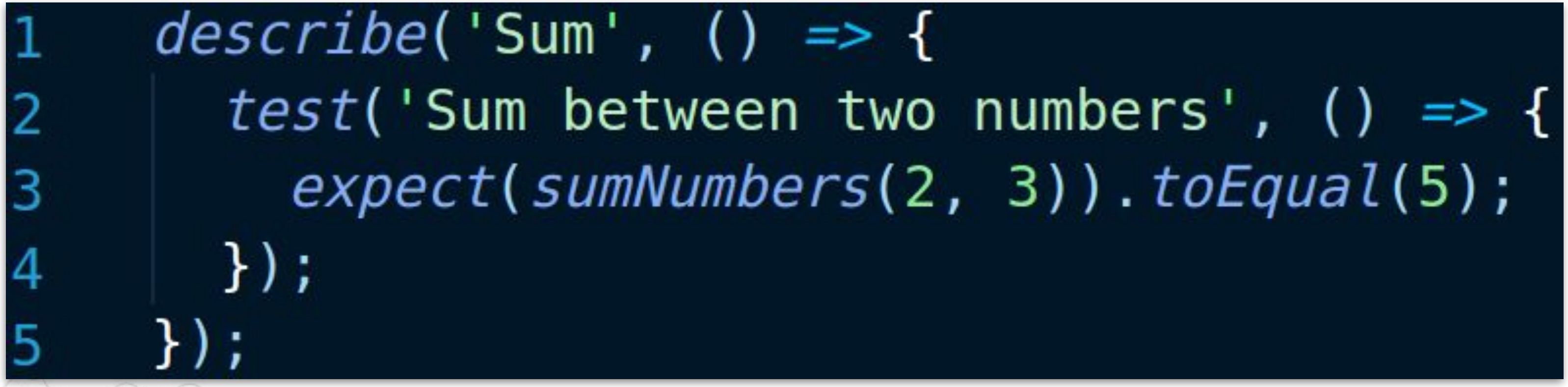
TELEPHONES WILL CEASE TO FUNCTION!

DOMINO EFFECT WILL CAUSE A WORLDWIDE DEPRESSION!

1. TDD

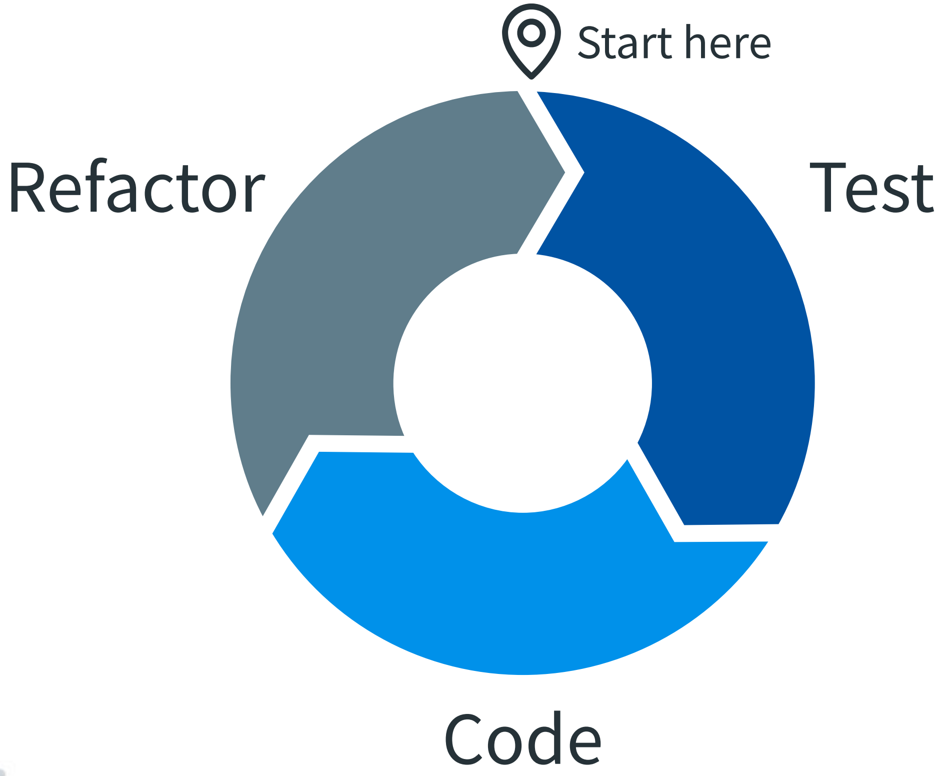
Definition

Test Driven Development is a software development process that starts with the creation of a test and continues with the implementation of the code that makes it work.



```
describe('Sum', () => {
  test('Sum between two numbers', () => {
    expect(sumNumbers(2, 3)).toEqual(5);
  });
});
```

TDD's cycle



Why is TDD better than other software development processes?

“*Focus to a single feature at a time.*”

Modularity

“Developers naturally produce a cleaner, more readable, and manageable code.”

Maintenance

“*Modular improvement.*”

Refactoring

“TDD can reduce your time-to-market speed.”

Decreasing costs

“Tests act as documentation and illustrate how the code works.”

Better documentation

“TDD produces a higher overall test coverage”

Less debugging

What about TDD’s cons?

“The team will be busy writing tests first.”

Slow development

"Requires skills, persistence, and discipline."

Difficulty

“Not every developer can make tests before having the code done”

Strange approach

“Tests could change to adapt.”

Changing tests

TDD is easier and challenging to maintain?

Tests ≠ Implementation

- Test maintenance is hard.
- Implementation maintenance is easy.

Conclusions

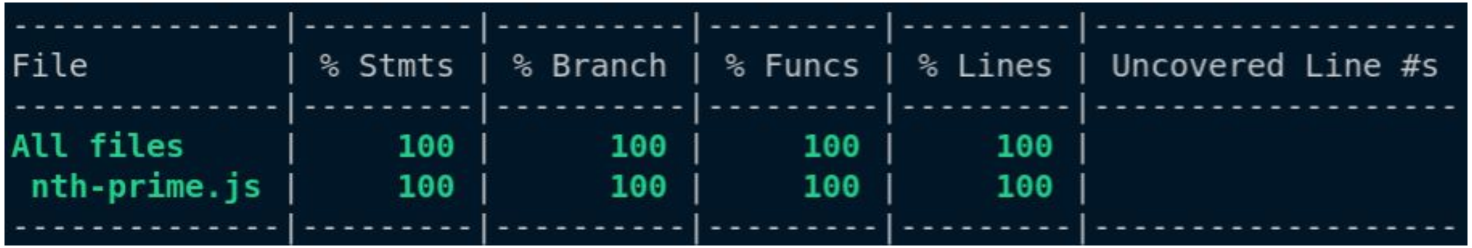
TDD: Conclusions

- Code quality.
- Difficulty.
- Use in companies.
- Code coverage.

2. Code Coverage

Definition

Represents the percentage of code that has been tested and strongly represents the completeness of our tests.



```
--------------|----------|----------|----------|----------|-------------------
File          |  % Stmts | % Branch |  % Funcs |  % Lines | Uncovered Line #s
--------------|----------|----------|----------|----------|-------------------
All files     |      100 |      100 |      100 |      100 |
 nth-prime.js |      100 |      100 |      100 |      100 |
--------------|----------|----------|----------|----------|-------------------
```

Criteria

- **Function Coverage**: Has each function been tested?
- **Statement Coverage**: Has each statement been tested?
- **Edge Coverage**: Has the control flow been tested completely?
- **Condition Coverage**: Has every condition been tested?

Why is Code Coverage useful?

“Without code coverage, we don’t know if our tests are useful”

Safety

“Higher code coverage finds more bugs”

Quality

How much Code Coverage is necessary?

Nice Code Coverage

Small projects

- Aim to 100%

Large projects

- Aim to 70-80%

High risk systems

- Aim to the highest possible value.

But, are these numbers absolute?

“Focus your time testing what is important.”

Understand your code

Improvement

1. **Make tests for general cases.**
2. **Identify critical misses in testing with code coverage.**
3. **Improve your testing.**

Code Coverage in Jest

What is Jest?

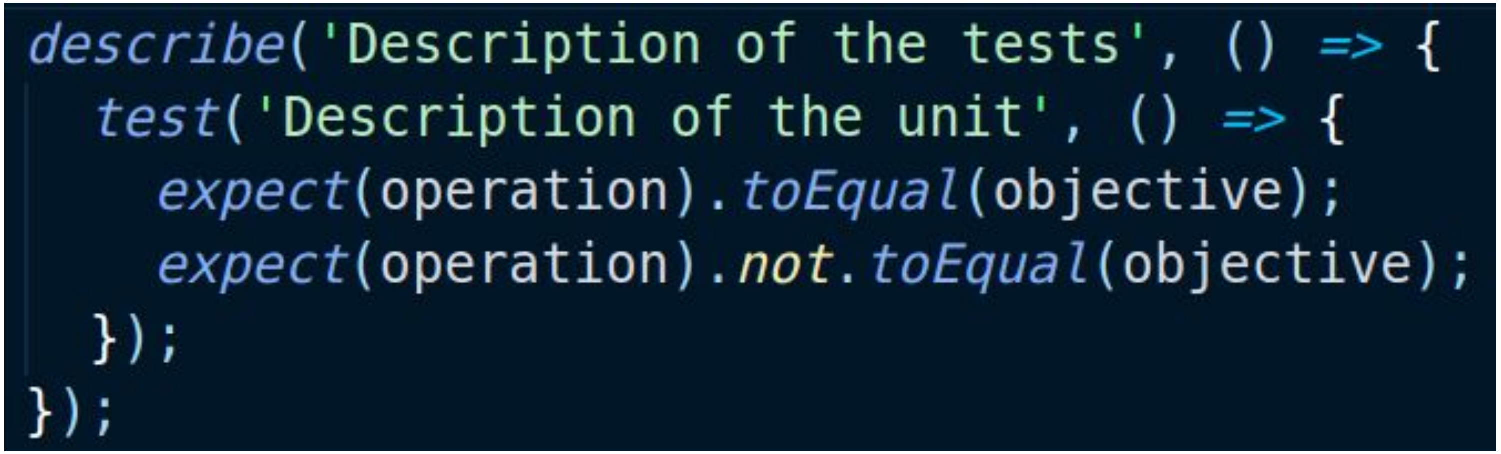
Jest is a JavaScript testing framework designed to ensure correctness of any JavaScript codebase.

Jest

“Write tests with an approachable, familiar and feature-rich API that gives you results quickly”

Simplicity

Making a test



```
describe('Description of the tests', () => {
  test('Description of the unit', () => {
    expect(operation).toEqual(objective);
    expect(operation).not.toEqual(objective);
  });
});
```

Matchers

- 📌toBeGreaterThan
- 📌toBeLessThan
- 📌toBe
- 📌toEqual

- 📌toBeNull
- 📌toBeDefined
- 📌toBeUndefined
- 📌toBeTruthy
- 📌toBeFalsy

- 📌toMatch
- 📌toContain
- 📌toThrow

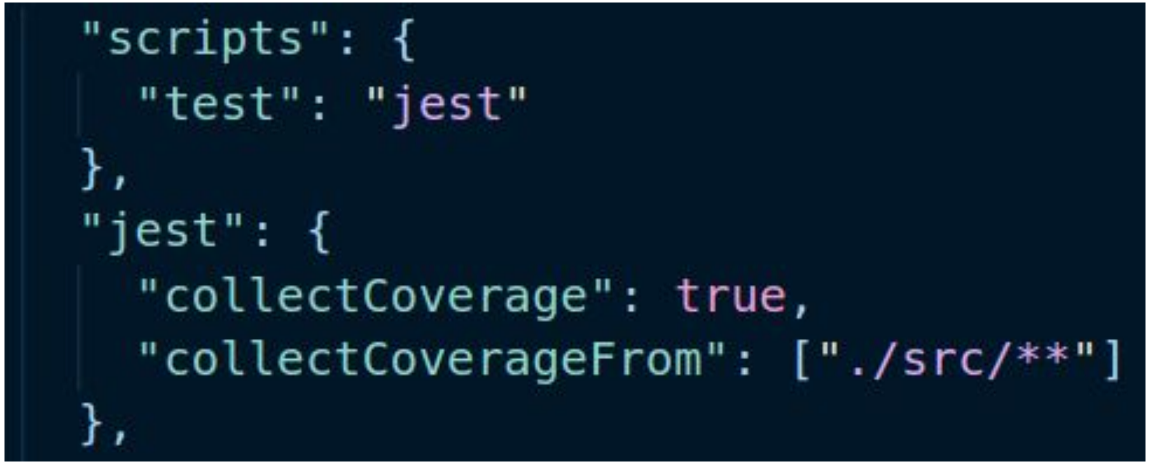
Now, how to do coverage with Jest?

Noob mode

Just add *--coverage*

Medium mode

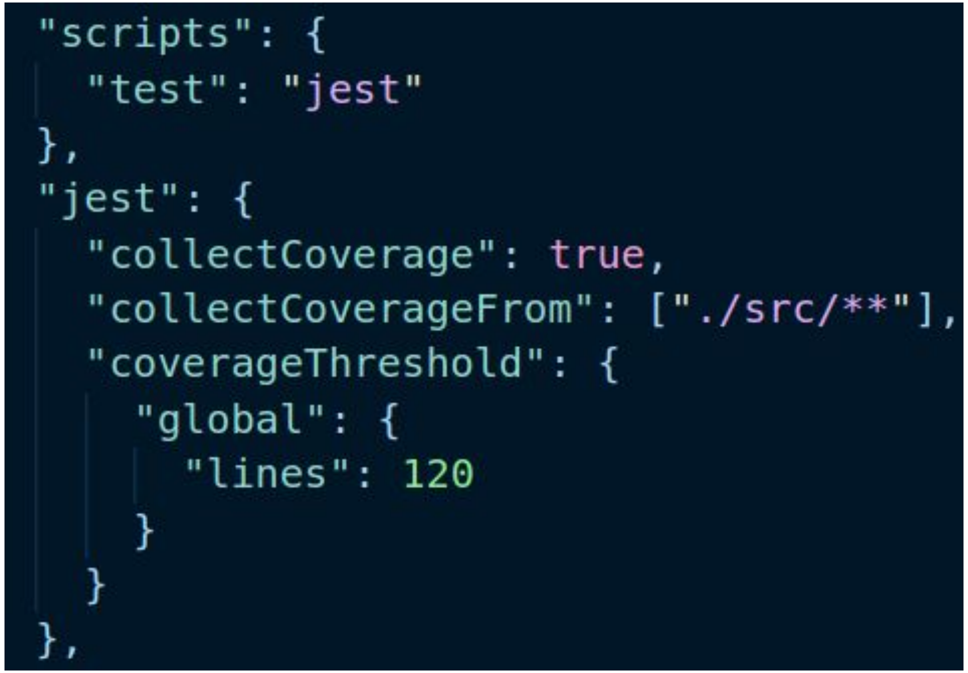
Using a *package.json*



```
"scripts": {
  "test": "jest"
},
"jest": {
  "collectCoverage": true,
  "collectCoverageFrom": ["./src/**"]
},
```

Pro mode

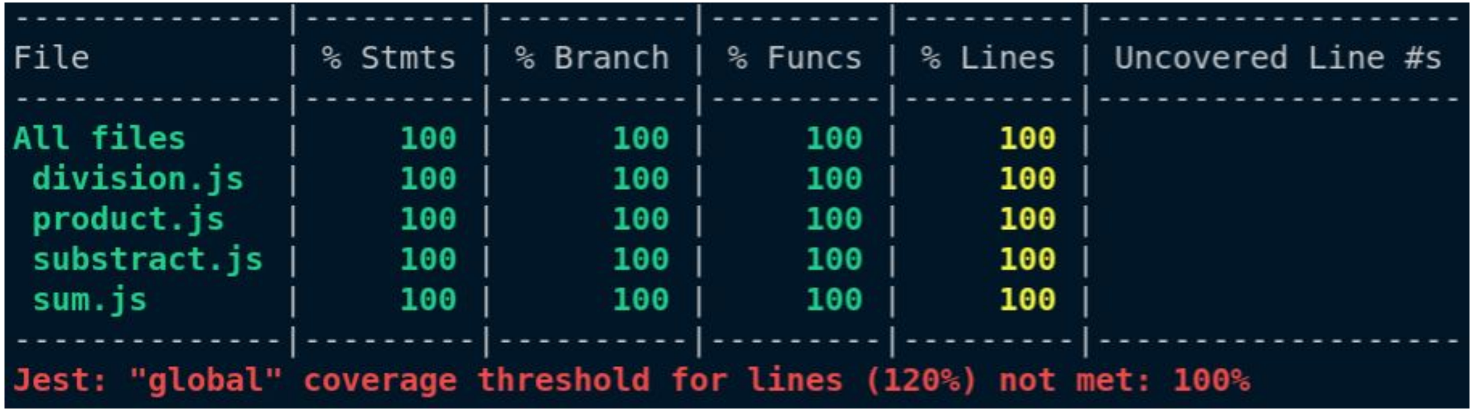
Using a good package.json.



```
"scripts": {
  "test": "jest"
},
"jest": {
  "collectCoverage": true,
  "collectCoverageFrom": ["./src/**"],
  "coverageThreshold": {
    "global": {
      "lines": 120
    }
  }
},
```

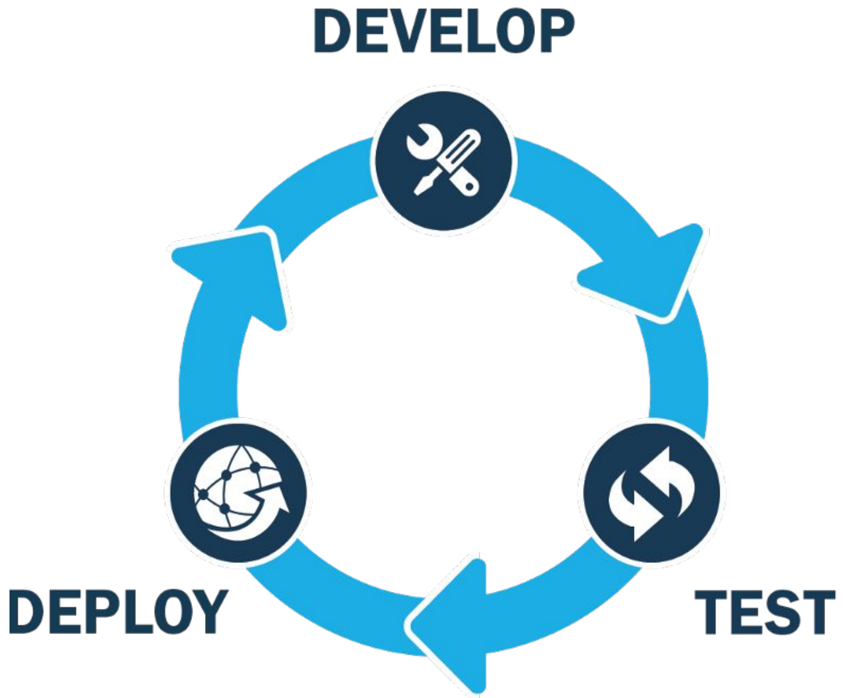
Pro mode

Using a good package.json.



```
--------------|----------|-----------|----------|----------|-------------------
File          | % Stmts  | % Branch  | % Funcs  | % Lines  | Uncovered Line #s
--------------|----------|-----------|----------|----------|-------------------
All files     |      100 |       100 |      100 |      100 |
 division.js  |      100 |       100 |      100 |      100 |
 product.js   |      100 |       100 |      100 |      100 |
 substract.js |      100 |       100 |      100 |      100 |
 sum.js       |      100 |       100 |      100 |      100 |
--------------|----------|-----------|----------|----------|-------------------
Jest: "global" coverage threshold for lines (120%) not met: 100%
```

What about continuous integration?



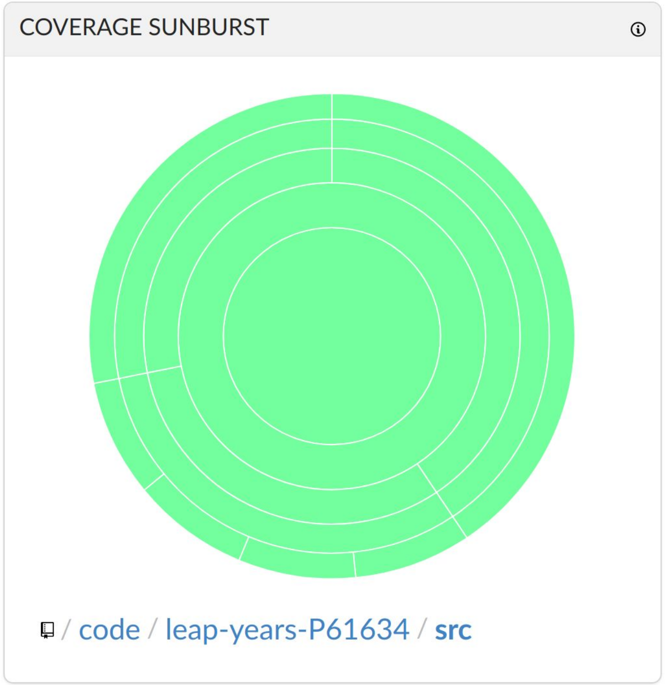
Code Coverage in CodeCov

Definition

Code coverage is one of the most important metrics companies rely on to ship healthier code, faster, and with less risk.

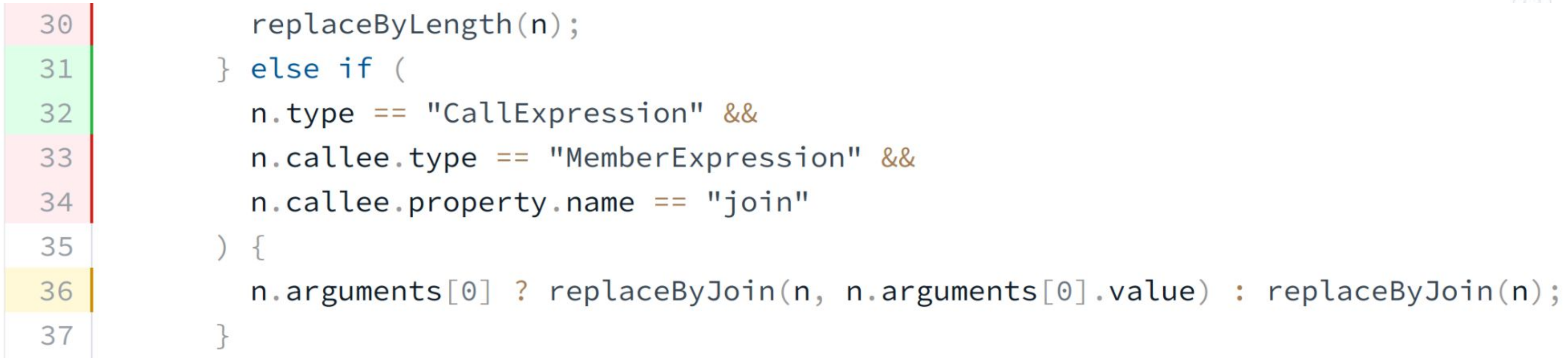
CodeCov benefits

Code coverage in a graphical representation.



CodeCov benefits

Code coverage in an easy way.



```
    replaceByLength(n);
} else if (
    n.type == "CallExpression" &&
    n.callee.type == "MemberExpression" &&
    n.callee.property.name == "join"
) {
    n.arguments[0] ? replaceByJoin(n, n.arguments[0].value) : replaceByJoin(n);
}
```

Use CodeCov

1. Give CodeCov permission to your Github.
2. Choose a repository (Token creation).

https://about.codecov.io/

Use CodeCov

3. Create the coverage directory with Jest.
4. Setup the CI with Github Actions.

Use CodeCov

5. Download the CodeCov Uploader.
6. Upload coverage with ./codecov -t [Token]

Conclusions

CodeCov: Conclusions

- **Code quality.**
- **Code safety.**
- **Simple code coverage improvement.**

Bibliography: TDD

1. **TDD Wikipedia**: https://en.wikipedia.org/wiki/Test-driven_development
2. **Learn TDD**: https://github.com/dwyl/learn-tdd
3. **Benefits of TDD**: https://fortegrp.com/test-driven-development-benefits/#:~:text=Developers%20have%20less%20debugging%20to,quality%20of%20the%20final%20product
4. **Software disasters**: https://raygun.com/blog/costly-software-errors-history/

   https://www.rankred.com/biggest-software-failures/

Bibliography: Code Coverage

1. **Code Coverage Wikipedia**: https://en.wikipedia.org/wiki/Code_coverage
2. **Code Coverage Definition**: https://confluence.atlassian.com/clover/about-code-coverage-71599496.html#:~:text=Code%20coverage%20is%20the%20percentage,and%20which%20statements%20have%20not
3. **Code Coverage Criteria**: https://www.atlassian.com/continuous-delivery/software-testing/code-coverage
4. **Why is Code Coverage important?**: https://about.codecov.io/blog/who-cares-about-code-coverage-and-why/#:~:text=Code%20coverage%20is%20a%20simple,the%20quality%20of%20your%20code
5. **Use Code Coverage with Jest**: https://www.valentinog.com/blog/jest-coverage/
6. **Jest's Code Coverage Documentation**: https://jestjs.io/docs/configuration#coveragethreshold-object
7. **CodeCov Quick Start**: https://docs.codecov.com/docs

Thanks!

Any questions?

You can find us at:

pablo.perez.gonzalez.23@ull.edu.es

gabriel.perez.10@ull.edu.es