fast-check

Property-based testing

Luis Presa Collada

Pablo Cañal Suárez

Sofía García Barbés

Daniel Finca Martínez

Introduction

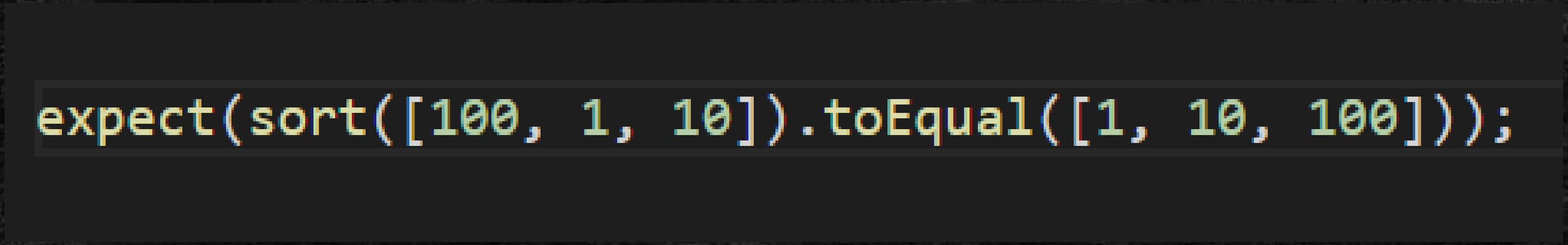
Some definitions

- Property testing
- Property, Law or Invariant
- Test generation
- Non-deterministic test execution

Property testing / Generative testing

- Writing code: Solving a problem.
- Writing tests: Checking whether or not our solution is valid

- Integer array sorting function



```
expect(sort([100, 1, 10]).toEqual([1, 10, 100]));
```

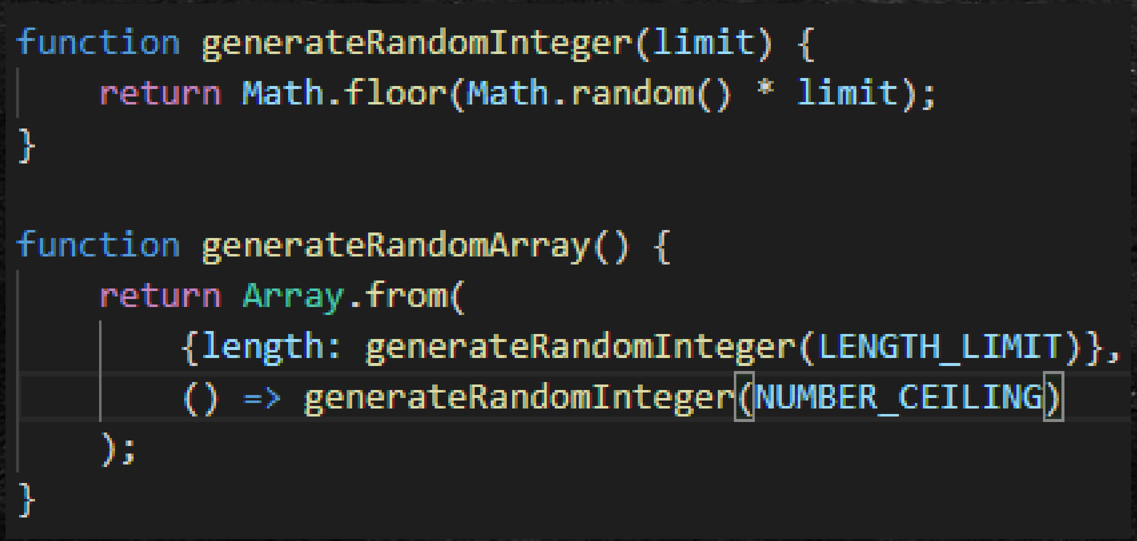
Property / Law / Invariant

- Characteristic(s) that qualify the result of the execution of certain piece of code.
- Some are valid in many cases. Some are not.

- Previous integer array sorting function properties:
  - Sorting an array would never modify its content (add, remove, change elements).
  - Sorting an already sorted array does nothing to it.
  - Sorting a sequence of numbers results in an ordered list (Quite obvious BTW).

Test generation => Ad-hoc generators

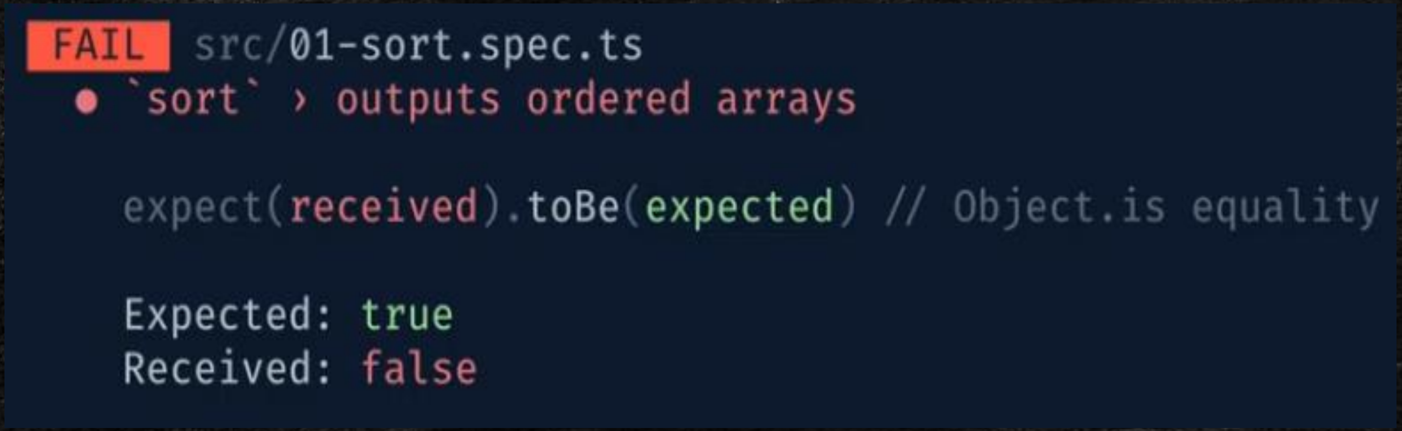
- Well, I have the properties. What do I do now?
  - Good question by the way! (:



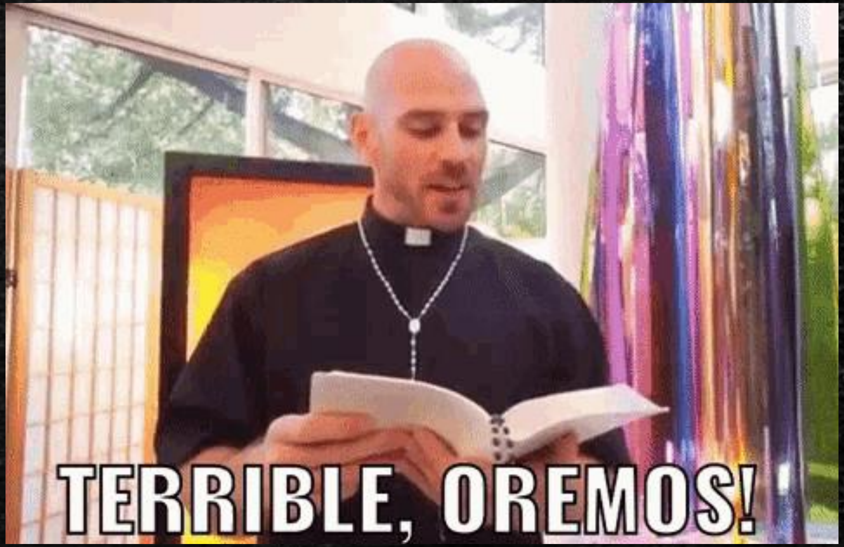
```
function generateRandomInteger(limit) {
    return Math.floor(Math.random() * limit);
}

function generateRandomArray() {
    return Array.from(
        {length: generateRandomInteger(LENGTH_LIMIT)},
        () => generateRandomInteger(NUMBER_CEILING)
    );
}
```

NDE => Non-deterministic execution

- Randomness is a pain in the… Yeah.
- Sometimes test pass. Some others… well you just simply don’t have a clue about what happened.



```
FAIL src/01-sort.spec.ts
  ● `sort` › outputs ordered arrays

    expect(received).toBe(expected) // Object.is equality

    Expected: true
    Received: false
```



Why is it awesome?

Three main features...

- Coverage
- Reproducible
- Shrink

Traditional testing...? Sure?

- Coverage

```
function serialize<T>(instance: T, params: Parameters): string
{ /* code */ }
```

Discover uncovered code paths!

- Coverage

```
test( ‘Should be able to read itself’ , () => {
        fc.assert(
                fc.property( fc. jsonObject(), (instance, params) => {
                        expect(deserialize( serialize(instance, params))).toEqual( instance);
                })
        )
});
```

Replay the same test!

- Reproducible

```
Error: Property failed after 1 tests ( seed: 1527423434693, path: "0:0:0“ ): ["","",""]
Shrunk 2 time(s)

Got error: Property failed by returning false
```

Replay the same test!

- Reproducible

```
test('the failing test', () => fc.assert ( fc.property (
                    // check method
          ), {
          seed: 1527423434693,           // seed and path taken from the previous slide
          path: "0:0:0" }
));
```

Understand your errors...

- Shrink

```
test('the failing test', () => fc.assert (
        fc.property (
            // check method
        ), { verbose: true }
));
```

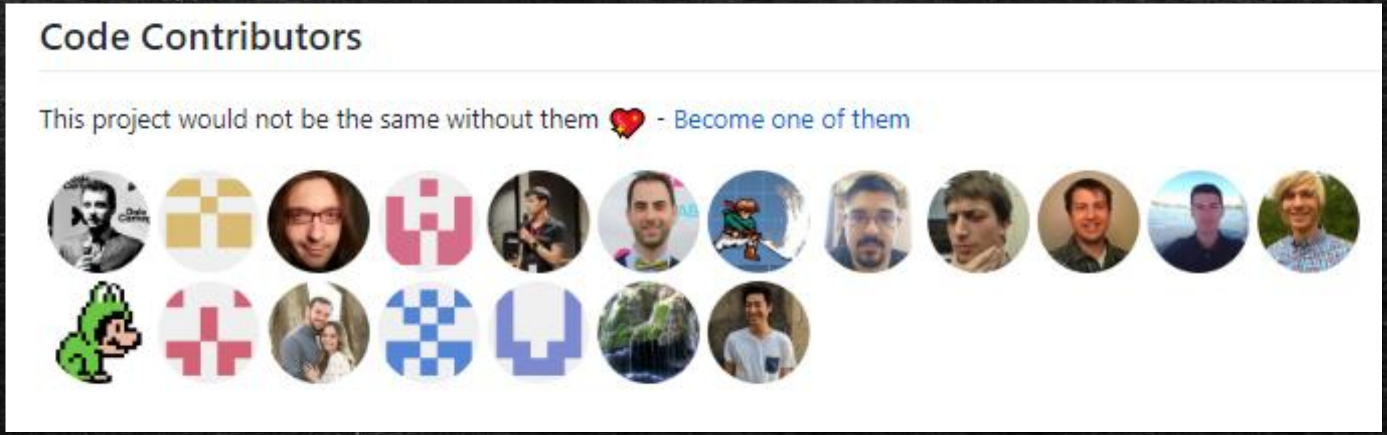
Understand your errors...

- Shrink

```
Error: Property failed after 1 tests ( seed: 1527423434693, path: "0:0:0“ ): ["","",""]
Shrunk 2 time(s)

Got error: Property failed by returning false

Encountered failures were:

 - ["", "JeXPqlQ6”, ">q"]

 - ["", "", ">q"]

 - ["", "", ""]
```

Who could be interested?

Stakeholders



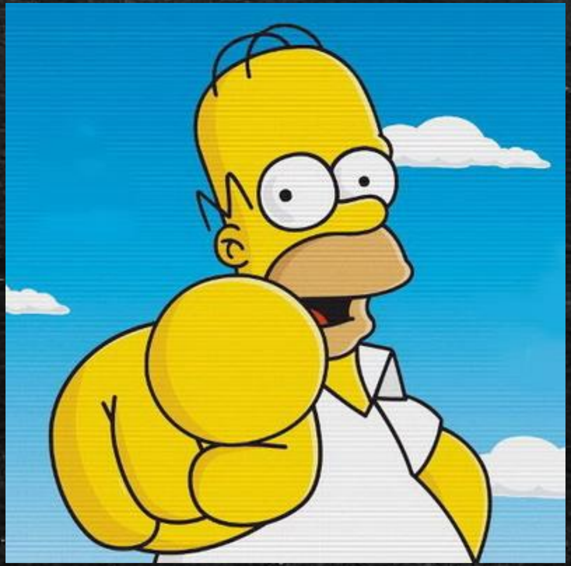
Code Contributors

This project would not be the same without them - Become one of them

Nicolas Dubien

The GitHub team

Stakeholders



You as a developer!

Packages using Fast-check!

Jasmine
Behavior-Driven JavaScript

Regulex

query-string build passing

List

Jest

How to Fast-Check

Hang on! Configure jest script

```
"scripts": {
        "test": "jest"
},
"jest": {
        "moduleFileExtensions": ["ts", "tsx", "js"],
        "globals": {"ts-jest": {"tsConfig": "tsconfig.json"}},
        "transform": {"^.+\\.(ts|tsx)$": "ts-jest"},
        "testMatch": ["**/specs/*.+(ts|tsx|js)"]
},
```

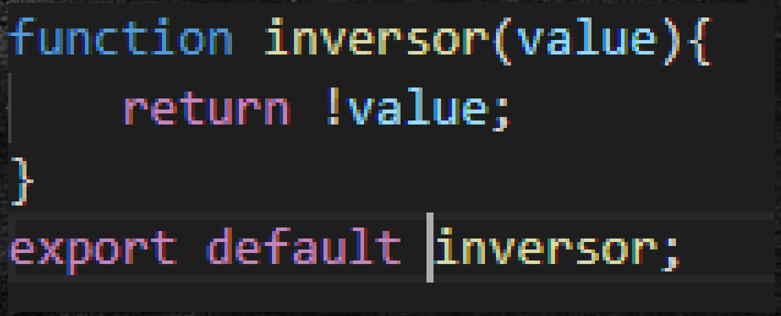
And finally…

```
npm install --save-dev fast-check
```

LET THE GAMES
BEGIN

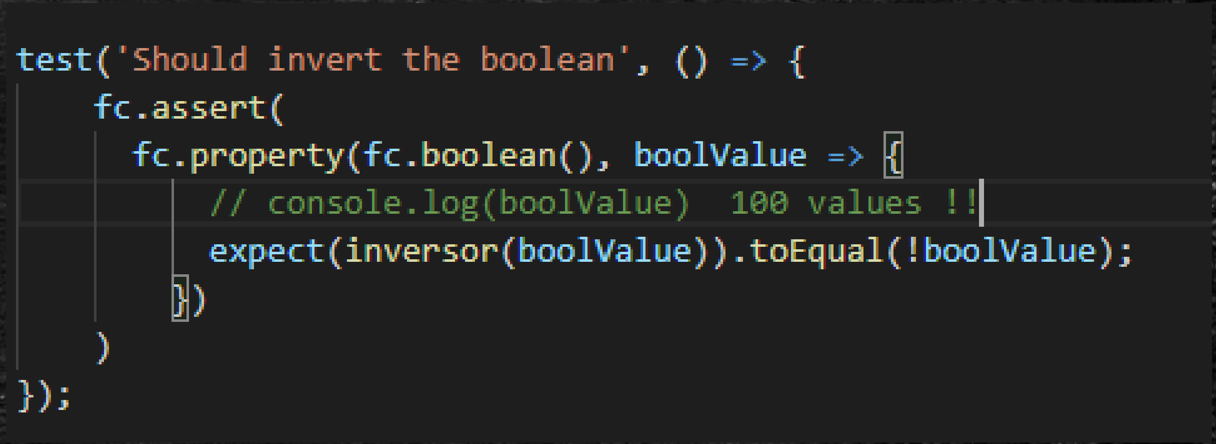
Let’s go slow, for now

Method to test



```
function inversor(value){
    return !value;
}
export default inversor;
```

Test Method



```
test('Should invert the boolean', () => {
    fc.assert(
      fc.property(fc.boolean(), boolValue => {
          // console.log(boolValue)  100 values !!
          expect(inversor(boolValue)).toEqual(!boolValue);
        })
    )
});
```

Boring stuff, right? Fast! Check this!

- Numbers: (and you can bound them!)
  - Natural
  - Negative
  - Floating point
- Random strings:
  - Ascii, Unicode.
  - Length bounded
  - Char bit size

Not convinced yet? More random Strings!!

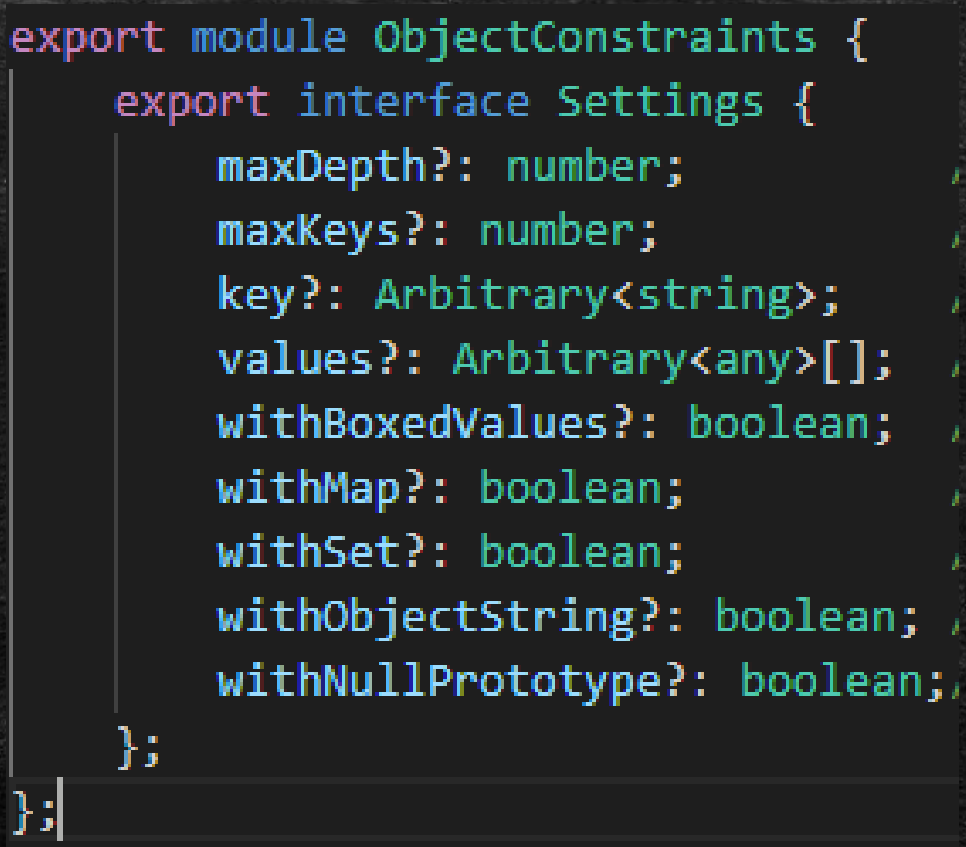
Json structure	->	.json(maxLength)
Ipv4, Ipv6	->	.ipv4() .ipv6()
URLs	->	.webUrl()
Email addresses	->	.emailAddress()
UUID	->	.uuid() .uuidV(version)

More “normal” things

- Fixed contant -> .constant(value)
- Picked constant -> .oneOf(array[value])
- Array: of an Arbitrary.
- Subarray: picks elements from an array. You may shuffle.
- Set: Unique values of Arbitrary.

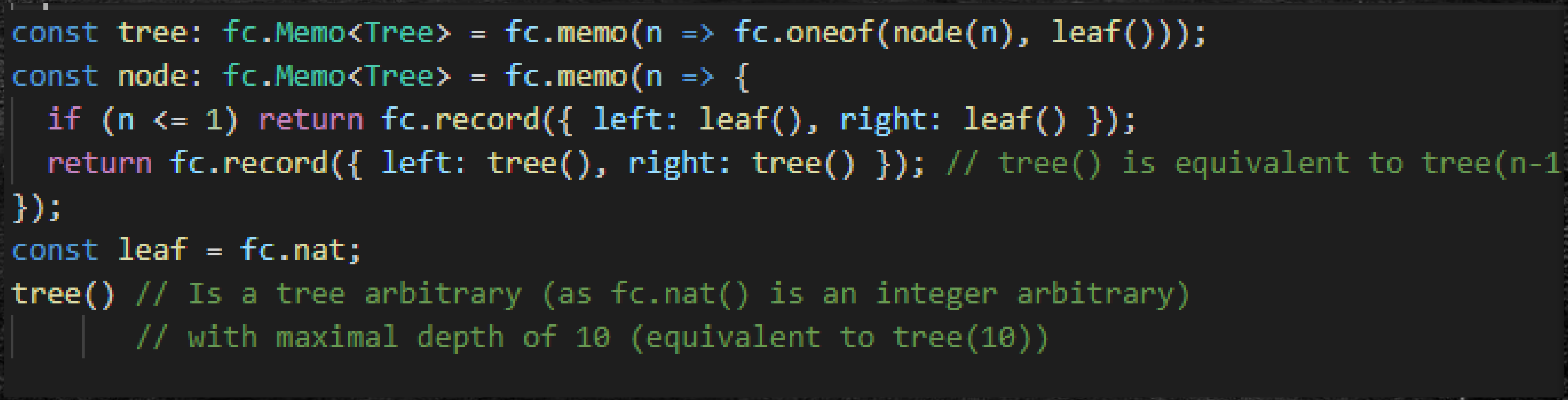
anything(settings:ObjectConstraints.Settings)

- You need an input
- You configure it
- You use the function
- You have your test :D



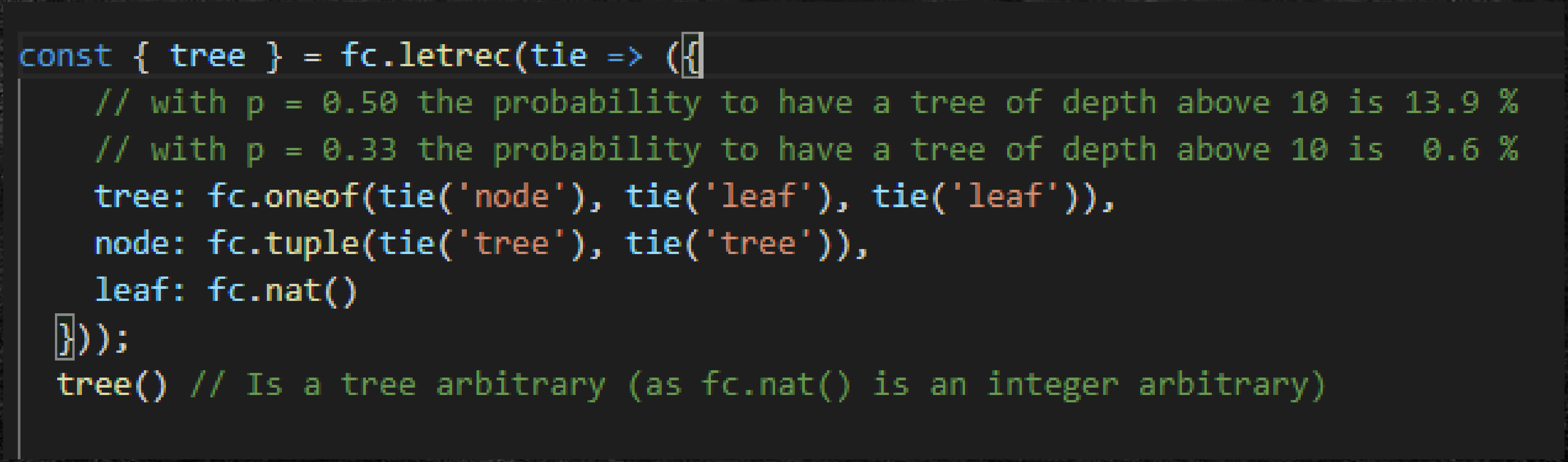
```
export module ObjectConstraints {
    export interface Settings {
        maxDepth?: number;
        maxKeys?: number;
        key?: Arbitrary<string>;
        values?: Arbitrary<any>[];
        withBoxedValues?: boolean;
        withMap?: boolean;
        withSet?: boolean;
        withObjectString?: boolean;
        withNullPrototype?: boolean;
    };
};
```

RECUR_RECUR__RECURSION__SION_SION



```
const tree: fc.Memo<Tree> = fc.memo(n => fc.oneof(node(n), leaf()));
const node: fc.Memo<Tree> = fc.memo(n => {
  if (n <= 1) return fc.record({ left: leaf(), right: leaf() });
  return fc.record({ left: tree(), right: tree() }); // tree() is equivalent to tree(n-1
});
const leaf = fc.nat;
tree() // Is a tree arbitrary (as fc.nat() is an integer arbitrary)
       // with maximal depth of 10 (equivalent to tree(10))
```

Crazy, random, and recursive.



```
const { tree } = fc.letrec(tie => ({
    // with p = 0.50 the probability to have a tree of depth above 10 is 13.9 %
    // with p = 0.33 the probability to have a tree of depth above 10 is  0.6 %
    tree: fc.oneof(tie('node'), tie('leaf'), tie('leaf')),
    node: fc.tuple(tie('tree'), tie('tree')),
    leaf: fc.nat()
  }));
  tree() // Is a tree arbitrary (as fc.nat() is an integer arbitrary)
```

Don’t understimate its power

- Transform and derive Arbitraries:
  - Map
  - Filter

- Commands (PreCondition -> Execution -> PostCondition)
- Support for asynchronous :
  - Command
  - Arbitraries
  - Handle Race Conditions
    - Shedulers
    - Wrap calls.

Someone said API?

{ api }

{api}

- **Arbitraty**

```
export declare abstract class Arbitrary<T>
```

- **Shrinkable**

```
export declare class Shrinkable<T, TShrink extends T = T>
```

- anything(settings?)
- asciiString(minLength?, maxLength?)
- double()
- float()
- …

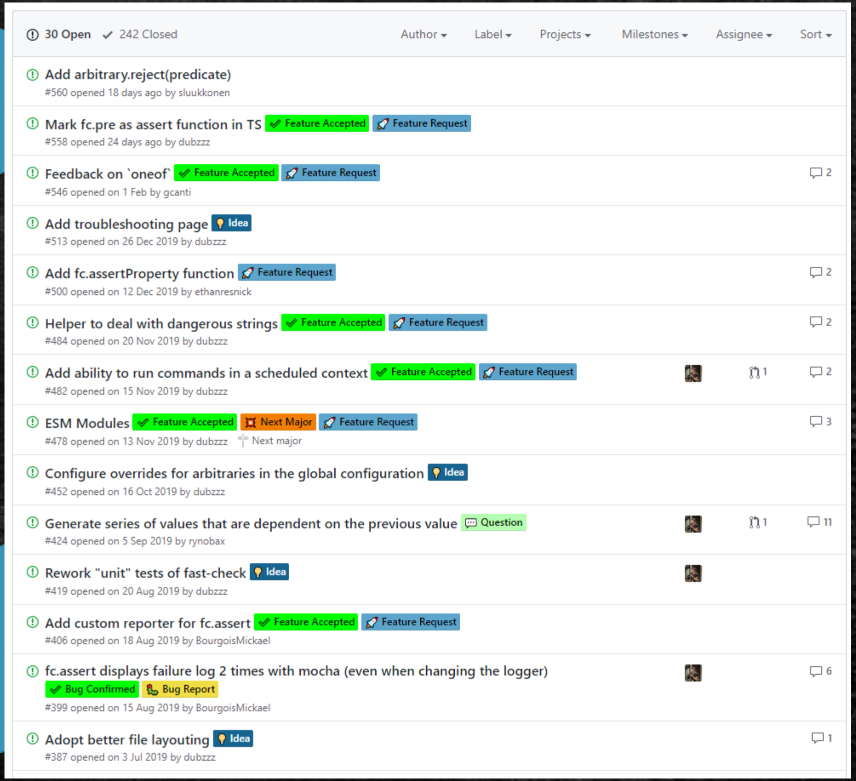
Quality Attributes

Why use it?

- Offers plenty of types
- Extendable
- Filtering options
- Adds debugging advantages
- Race conditions detection

Fast (and active) Check

Fast (and active) Check


30 Open ✓ 242 Closed | Author | Label | Projects | Milestones | Assignee | Sort

Add arbitrary.reject(predicate)
#560 opened 18 days ago by sluukkonen

Mark fc.pre as assert function in TS Feature Accepted Feature Request
#558 opened 24 days ago by dubzzz

Feedback on `oneof` Feature Accepted Feature Request 2
#546 opened on 1 Feb by gcanti

Add troubleshooting page Idea
#513 opened on 26 Dec 2019 by dubzzz

Add fc.assertProperty function Feature Request 2
#500 opened on 12 Dec 2019 by ethanresnick

Helper to deal with dangerous strings Feature Accepted Feature Request 2
#484 opened on 20 Nov 2019 by dubzzz

Add ability to run commands in a scheduled context Feature Accepted Feature Request 1 2
#482 opened on 15 Nov 2019 by dubzzz

ESM Modules Feature Accepted Next Major Feature Request 3
#478 opened on 13 Nov 2019 by dubzzz Next major

Configure overrides for arbitraries in the global configuration Idea
#452 opened on 16 Oct 2019 by dubzzz

Generate series of values that are dependent on the previous value Question 1 11
#424 opened on 5 Sep 2019 by rynobax

Rework "unit" tests of fast-check Idea
#419 opened on 20 Aug 2019 by dubzzz

Add custom reporter for fc.assert Feature Accepted Feature Request
#406 opened on 18 Aug 2019 by BourgoisMickael

fc.assert displays failure log 2 times with mocha (even when changing the logger) 6
Bug Confirmed Bug Report
#399 opened on 15 Aug 2019 by BourgoisMickael

Adopt better file layouting Idea 1
#387 opened on 3 Jul 2019 by dubzzz

Fast Releases

1.23.0

Thanks for fast-checking with us!!