# Property Based Testing with TypeScript & React

Rafał Kostrzewski

## Properties are laws that subjects under test must obey

## Some examples:

- never throws errors,
- always returns positive integers,
- returns array of length equal to input

## Property based testing aims to check those properties

You declare those properties and the library check those against (not so) random data

# It all started with QuickCheck & Haskell (FP )

## yarn add fast-check



#### How does it look like?

- 1. Find out what the variables are
- 2. Create **arbitraries** aka generators those values
- 3. Create **properties** based on those arbitraries
- 4. Assert that those properties are adhered to

#### Example: Let's test a simple function

```
const reverse = <T>(array: T[]): T[] ⇒ {
  const copy = array.slice()
  return copy.reverse()
}
```

### Example: Let's test a simple function

```
import fc from 'fast-check'

it('when reversed should equal original', () ⇒ {
  const arrayArbitrary = fc.array(
    fc.anything()
)

const reversedReverseEqualsOriginal = fc.property(
    arrayArbitrary,
    (array) ⇒ expect(reverse(reverse(array))).toEqual(array)
)

fc.assert(reversedReverseEqualsOriginal)
})
```

### Example: Let's mess up

```
const reverse = <T>(array: T[]): T[] ⇒ {
  const copy = array.slice()

if (array.length == 8) {
  return array.slice(1)
  }
  return copy.reverse()
}
```

#### Example: Let's mess up

```
FAIL src/registration/_tests_/example.ts

x when reversed should equal original (98ms)
```

• when reversed should equal original

```
Property failed after 9 tests
{ seed: -1651074760, path: "8:4:6:9:12:15:17:19:20", endOnFailure: true }
Counterexample: [[[],{},[],{},false,"",[],[]]]
Shrunk 8 time(s)
Got error: Error: expect(received).toEqual(expected) // deep equality
```

### Example: fast-check is deterministic

```
fc.assert(reversedReverseEqualsOriginal, {
    seed: -1651074760,
    path: "8:4:6:9:12:15:17:19:20",
    endOnFailure: true
})
```

### Example: fast-check shrinks (and does it well)

```
FAIL src/registration/__tests__/example.ts
    x when reversed should equal original (14ms)

• when reversed should equal original

Property failed after 3 tests
    { seed: -724242980, path: "2", endOnFailure: true }
    Counterexample: [[true,0.659151591115403,["I,2?","1!~"],{"sC`pD{":{},"U$48/":{"zYc P@z`%":undefined,"Xe":"XY","3Z":""}},[],{"_tY":[true,false,false],";n+":[-105757044,0.8085733412311057,515591603],"W*>72":"fw4cjJ","05D81<":{"?>4tT~bOh":undefined,"":"67","<)fKR:vHY*":null,"WRyg7Nus$4":"dx3","o|%M3[Zni(":",%&Ik6v3?y"}},{},{" x-gB":false,"%&.":false}]]
    Shrunk 0 time(s)
    Got error: Error: expect(received).toEqual(expected) // deep equality</pre>
```

### Example: fast-check shrinks (and does it well)

```
FAIL src/registration/_tests__/example.ts
x when reversed should equal original (81ms)

• when reversed should equal original

Property failed after 36 tests
{ seed: -38383590, path: "35:4:6:9:12:15:17:19:20", endOnFailure: true }
Counterexample: [[[],[],[],[],[],{},0,[]]]
Shrunk 8 time(s)
Got error: Error: expect(received).toEqual(expected) // deep equality
```

## Where to use property based testing?

- Algorithms
- Smoke testing (as in 50 6)
- State machines
- Random interactions 🐒
- Fuzz testing

## Doesn't replace other tests!!

## 

- Random interactions with a music players (play/ pause / next / prev)
- Fuzz testing of API (never returns 5xx)
- Race conditions (Autocomplete / ToDo list)

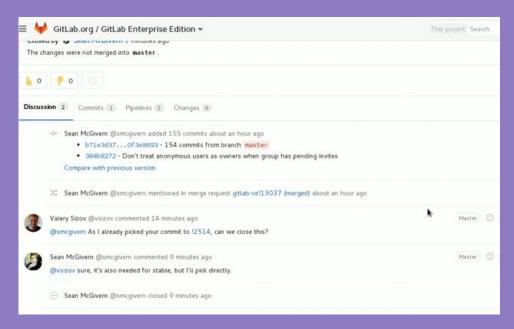
## Prevent real world bugs

## **Spotify**

- 1. Turn on playlist repeat
- 2. Prepare playlist with one track
- 3. Press play
- 4. Add a track to playlist
- 5. Click next

  First track plays again

## **Gitlab**



### JIRA / GT Dashboard / Others

1. Post a message with emoji MySQL throws an error

## Real world examples

github.com/rkostrzewski/property-based-testing

## Morse code aka testing untestable

## Forms aka testing validation

### — Q & A

## Thanks!