Generating tests with SwiftCheck

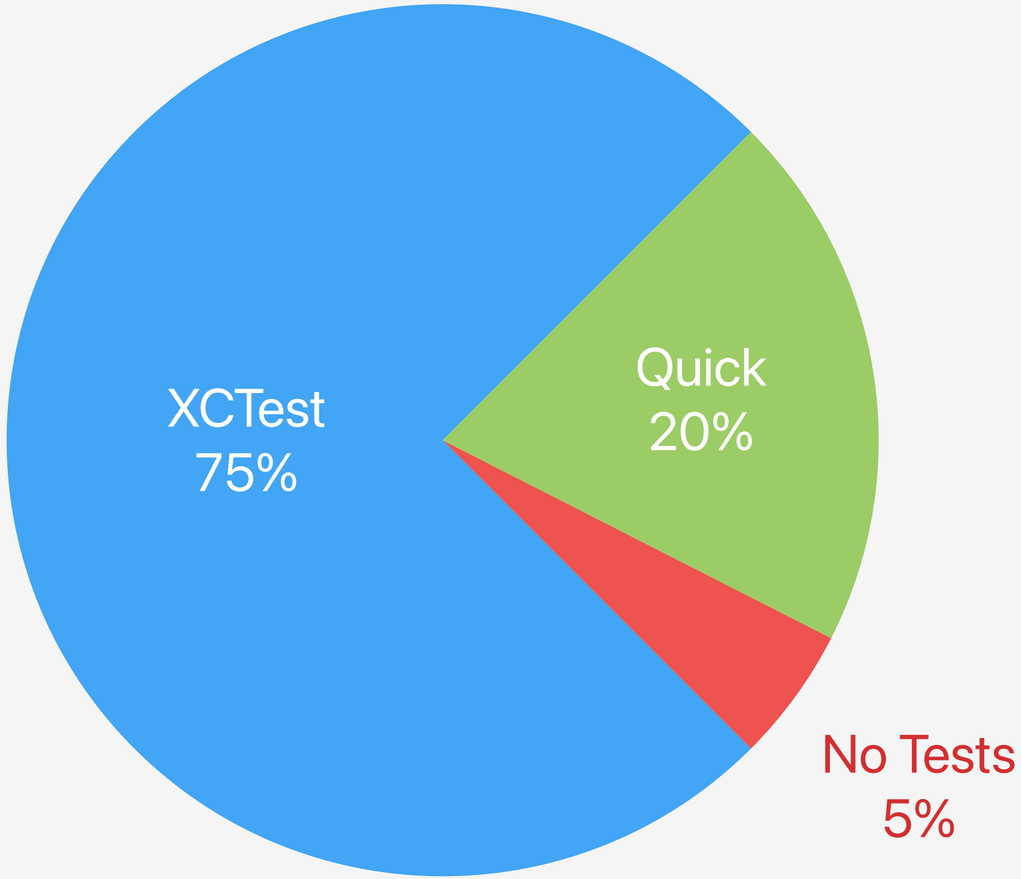
Adrian Kashivskyy @ Netguru

How many of you use XCTest for testing?

How many of you use Quick for testing?

How many of you don’t write tests? 😏

Testing frameworks in top popular projects



Source: 20 most starred Swift repositories on GitHub

```
override func setUp() {
  sut = Rectangle(width: 1, height: 2)
}

func testMultiplication() {
  sut.scale(by: 2)
  XCTAssertEqual(sut.width, 2)
  XCTAssertEqual(sut.height, 4)
}
```

Good tests

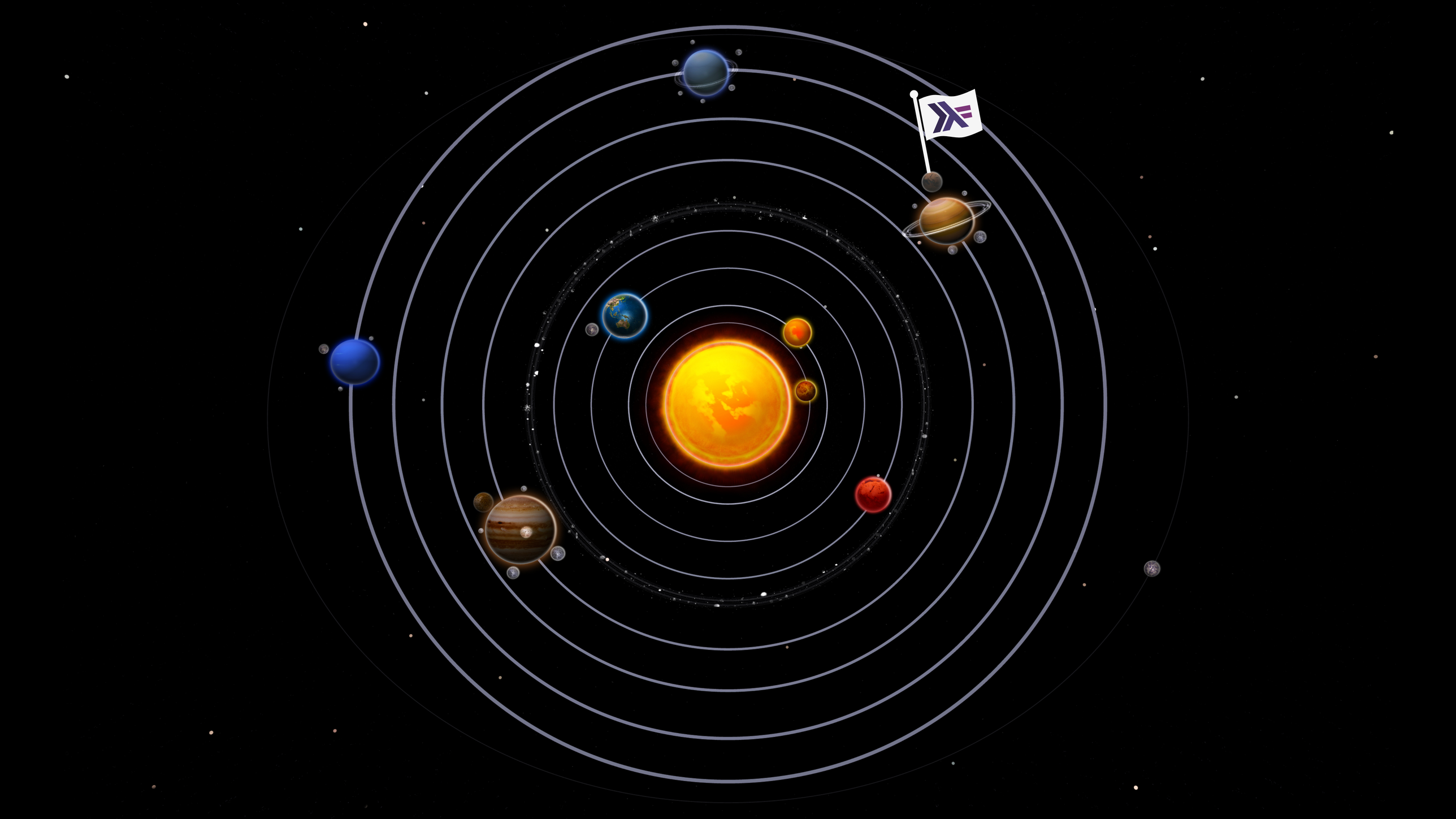
- Prove correctness of your code
- Catch regressions during development
- Find edge cases before your users do

```
override func setUp() {
  sut = Rectangle(width: 1, height: 2)
}

func testMultiplication() {
  sut.scale(by: 2)
  XCTAssertEqual(sut.width, 2)
  XCTAssertEqual(sut.height, 4)
}
```

```
func testMultiplicationByZero() {
  sut.scale(by: 0)
  XCTAssertEqual(sut.width, 0)
  XCTAssertEqual(sut.height, 0)
}

func testMultiplicationByFraction() {
  sut.scale(by: 0.5)
  XCTAssertEqual(sut.width, 0.5)
  XCTAssertEqual(sut.height, 1)
}
```

Such tests are limited to developer's imagination

Let computers come up with test cases



They introduced
QuickCheck

```haskell
allEqual  x y z = x == y && y == z
allEqual' x y z = 2 * x == y + z

prop_st x y (z :: Int) = allEqual x y z == allEqual' x y z

prop_rev_rev :: Eq a => [a] -> Bool
prop_rev_rev xs = reverse (reverse xs) == xs

main = $quickCheckAll
```

SwiftCheck

https://github.com/typelift/SwiftCheck

```
property("contains same elements as input")
```

```
property("elements are in ascending order")
```

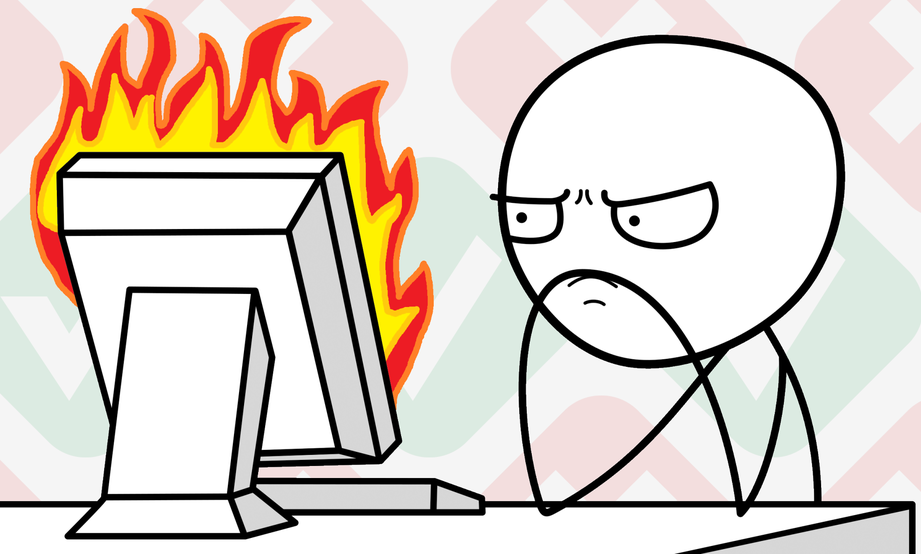
```
property("contains same elements as input")
<- forAll { (a: ArrayOf<Int>) in

}

property("elements are in ascending order")
<- forAll { (a: ArrayOf<Int>) in

}
```

```
property("contains same elements as input")
<- forAll { (a: ArrayOf<Int>) in
  a.getArray.sorted().counts() == a.getArray.counts()
}

property("elements are in ascending order")
<- forAll { (a: ArrayOf<Int>) in
  a.getArray.sorted().elementsAreInAscendingOrder()
}
```

Testing with SwiftCheck

- You write assertions about logical properties
- SwiftCheck looks for a failing case
- If found, SwiftCheck shrinks input to smallest reproducible value that causes failure

Demo



SwiftCheck building blocks

- struct Gen<A>
- protocol Arbitrary

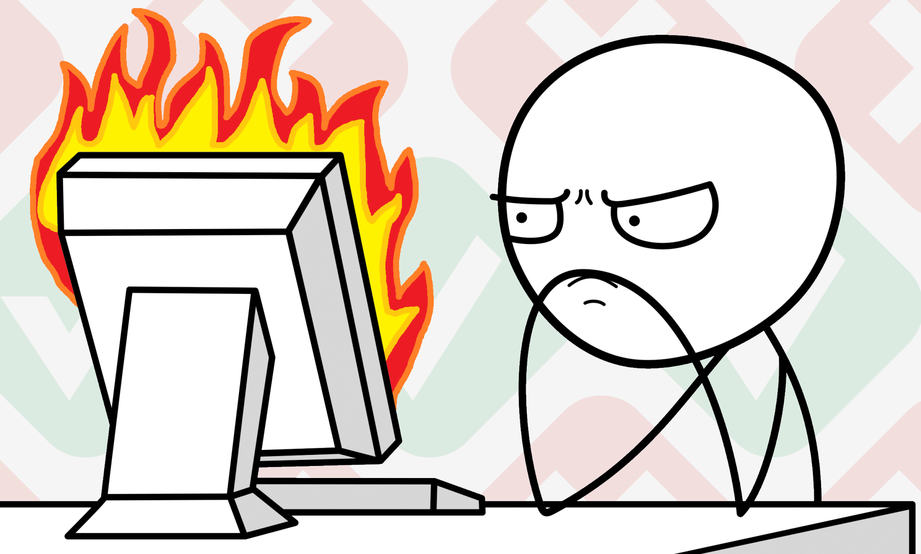
struct Gen<A>

- Responsible for generating random inputs
- Just a wrapper around a closure
- Controls distirbution of values
- Supports functional composition

protocol Arbitrary

- Denote which types can be generated
- Responsible for shrinking
- Has one requirement: a static variable of Gen

Demo



QuickSwiftCheck

https://github.com/akashivskyy/QuickSwiftCheck

Which properties to test?

- Round trip
- Relation to other equivalent implementations
- Relation to input

Question time!

Thanks,

have a great evening!