Property Testing

 Find More Bugs! 

'sup?

This is the part of the talk where I say my company is hiring.

We're gonna cover...

1. What's a property test?

2. When are they useful?

3. How can I write good properties?

4. 💀 Demo 💀

5. Takeaways & Recommendations

Do Questions As We Go!

1. What's a property test?

1. What's a property ~~test~~?

A property is a rule that describes a program's behavior.

$$5*3 = 3*5$$

Unit Testing

```
multiplication : Test
multiplication =
    describe "multiplication"
        [ test "is commutative" <|
            \_ ->
                Expect.equal (3 * 5) (5 * 3)
        ]
```

$$5*3 = 3*5$$

$$x * y = y * x$$

Unit Testing

```
multiplication : Test
multiplication =
    describe "multiplication"
        [ test "is commutative" <|
            \_ ->
                Expect.equal (3 * 5) (5 * 3)
        ]
```

Property Testing

```
multiplication : Test
multiplication =
    describe "multiplication"
        [ fuzz2 Fuzz.int Fuzz.int "is commutative" <|
            \x y ->
                Expect.equal (x * y) (y * x)
        ]
```

What's going on here?

1. Generate a random integer x
2. Generate a random integer y
3. Feed those to the assertion
4. Repeat 100ish times

Property Testing

```
multiplication : Test
multiplication =
    describe "multiplication"
        [ fuzz2 Fuzz.int Fuzz.int "is commutative" <|
            \x y ->
                Expect.equal (x * y) (y * x)
        ]
```

Won't I get garbage input on failure?

1. Generate a random integer `x`
2. Generate a random integer `y`
3. Feed those to the assertion, **which fails!** 😢
4. **Shrink the input to the simplest failing case**

```
fuzz (Fuzz.list Fuzz.int) "no five items" <|
    \xs ->
        Expect.notEqual 5 (List.length xs)
```

❌ [26495, 20, -8, -47, 7]

❌ [0, 0, 0, -47, 7]

❌ [0, 0, 0, 0, 0] 👈

✅ [0, 0, 0, 0]

2. When are they useful?

When a property (invariant) is known.

When two functions do symmetric things.

When the system under test does a complex thing, but the user sees a simple thing.

When you can use an "oracle" to verify behavior.

2b. When are they *not* useful?

When the code under test is super simple.

When the code under test is extremely expensive.

3. How can I write good properties?

It's really hard to write properties. Sorry.

Write down, in natural language, what your code should do.

The numbers in this list are ordered from least to greatest.

Each number in this list is less than or equal to the one after.

4. 💀 Demo 💀

5. Takeaways & Recommendations

Don't try to force property testing everywhere

Built-in generators are often insufficient.

Write your own!

Language Support

Language	Library
Elixir	stream_data
Elm	elm-test
Erlang	Quviq QuickCheck, PropEr
Haskell	QuickCheck Hedgehog
Python	Hypothesis
Scala	ScalaCheck, Hedgehog
your favorite language here	probably. Google it.

Further Reading

→ **Proper Testing:** propertesting.com

→ **Hypothesis blog:** hypothesis.works

Thank You!

brian@brianthicks.com

github.com/BrianHicks/property-testing-presentation