Protocol Witnesses

— Brandon Williams

— @mbrandonw

— mbw234@gmail.com

POINT·FREE

www.pointfree.co

w/ Stephen Celis (@stephencelis)

Protocols

Protocols

🛑 *Protocol can only be used as a generic constraint because it has Self or associated type requirements*

Protocols

```
extension Optional: Equatable where Wrapped: Equatable {
  static func ==(lhs: Wrapped?, rhs: Wrapped?) -> Bool {
    switch (lhs, rhs) {
    case let (.some(lhs), .some(rhs)):
      return lhs == rhs
    case (.none, .none):
      return true
    case (.some, .none), (.none, .some):
      return false
    }
  }
}
```

Protocols

```
extension Optional: Equatable where Wrapped: Equatable {
  static func ==(lhs: Wrapped?, rhs: Wrapped?) -> Bool {
    switch (lhs, rhs) {
    case let (.some(lhs), .some(rhs)):
      return lhs == rhs
    case (.none, .none):
      return true
    case (.some, .none), (.none, .some):
      return false
    }
  }
}
```

Protocols

```
extension Void: Equatable {
  static func ==(lhs: Void, rhs: Void) -> Bool {
    return true
  }
}

extension (A, B): Equatable where A: Equatable, B: Equatable {
  static func ==(lhs: (A, B), rhs: (A, B)) -> Bool {
    return lhs.0 == rhs.0 && lhs.1 == rhs.1
  }
}

extension ((A) -> B): Equatable where A: CaseIterable, B: Equatable {
  static func ==(lhs: (A) -> B, rhs: (A) -> B) -> Bool {
    return A.allCases.reduce(true) { isEqual, a in
      isEqual && lhs(a) == rhs(a)
    }
  }
}
```

Protocols

```
extension Void: Equatable {
  static func ==(lhs: Void, rhs: Void) -> Bool {
    return true
  }
}

extension (A, B): Equatable where A: Equatable, B: Equatable {
  static func ==(lhs: (A, B), rhs: (A, B)) -> Bool {
    return lhs.0 == rhs.0 && lhs.1 == rhs.1
  }
}

extension ((A) -> B): Equatable where A: CaseIterable, B: Equatable {
  static func ==(lhs: (A) -> B, rhs: (A) -> B) -> Bool {
    return A.allCases.reduce(true) { isEqual, a in
      isEqual && lhs(a) == rhs(a)
    }
  }
}
```

Protocols

```
extension Void: Equatable {
  static func ==(lhs: Void, rhs: Void) -> Bool {
    return true
  }
}

extension (A, B): Equatable where A: Equatable, B: Equatable {
  static func ==(lhs: (A, B), rhs: (A, B)) -> Bool {
    return lhs.0 == rhs.0 && lhs.1 == rhs.1
  }
}

extension ((A) -> B): Equatable where A: CaseIterable, B: Equatable {
  static func ==(lhs: (A) -> B, rhs: (A) -> B) -> Bool {
    return A.allCases.reduce(true) { isEqual, a in
      isEqual && lhs(a) == rhs(a)
    }
  }
}
```

Protocols

```
extension Void: Equatable {
  static func ==(lhs: Void, rhs: Void) -> Bool {
    return true
  }
}

extension (A, B): Equatable where A: Equatable, B: Equatable {
  static func ==(lhs: (A, B), rhs: (A, B)) -> Bool {
    return lhs.0 == rhs.0 && lhs.1 == rhs.1
  }
}

extension ((A) -> B): Equatable where A: CaseIterable, B: Equatable {
  static func ==(lhs: (A) -> B, rhs: (A) -> B) -> Bool {
    return A.allCases.reduce(true) { isEqual, a in
      isEqual && lhs(a) == rhs(a)
    }
  }
}
```

Protocols

```
import XCTest

XCTAssertEqual((1, 1), (2, 2))
```

🛑 *Global function 'XCTAssertEqual(_:_:_:file:line:)' requires that '(_, _)' conform to 'Equatable'*

Protocols

```
indirect enum Tree<A> {
  case empty
  case node(left: Tree<A>, value: A, right: Tree<A>)
}

extension Tree: Sequence {
  // depth first?
  //   in-order?
  //   pre-order?
  //   post-order?
  // breadth first?
}
```

Protocols

```
indirect enum Tree<A> {
  case empty
  case node(left: Tree<A>, value: A, right: Tree<A>)
}

extension Tree: Sequence {
  // depth first?
  //   in-order?
  //   pre-order?
  //   post-order?
  // breadth first?
}
```

Protocols

```
protocol ProtocolA {}
protocol ProtocolB {}

extension ProtocolA: ProtocolB {}
```

🛑 ***Extension of protocol 'ProtocolA' cannot have an inheritance clause***

Non-protocol forms of abstraction

De-protocolization

Combinable

```
protocol Combinable {
  func combine(_ other: Self) -> Self
}
```

Combinable

```
extension String: Combinable {
  func combine(_ other: String) -> String {
    return self + other
  }
}
```

Combinable

```
extension Array: Combinable {
  func combine(_ other: Array) -> Array {
    return self + other
  }
}
```

Combinable

```
extension RangeReplaceableCollection: Combinable {
  func combine(_ other: Self) -> Self {
    return self + other
  }
}
```

🛑 ***Extension of protocol 'RangeReplaceableCollection' cannot have an inheritance clause***

Combinable

```
extension Int: Combinable {
  func combine(_ other: Int) -> Int {
    return self + other
  }
}

extension Double: Combinable {
  func combine(_ other: Int) -> Int {
    return self + other
  }
}
```

Combinable

```
extension Numeric: Combinable {
  func combine(_ other: Self) -> Self {
    return self + other
  }
}
```

🛑 ***Extension of protocol 'Numeric' cannot have an inheritance clause***

```
extension Int: Combinable {
  func combine(_ other: Int) -> Int {
    return self * other
  }
}
```

🛑 ***Redundant conformance of 'Int' to protocol 'Combinable'***

```
extension Double: Combinable {
  func combine(_ other: Int) -> Int {
    return self * other
  }
}
```

🛑 ***Redundant conformance of 'Double' to protocol 'Combinable'***

Combinable

```
extension (A, B): Combinable where A: Combinable, B: Combinable {
  func combine(_ other: (A, B)) -> (A, B) {
    return (
      self.0.combine(other.0),
      self.1.combine(other.1)
    )
  }
}

extension ((A) -> B): Combinable where B: Combinable {
  func combine(_ other: (A) -> B) -> (A) -> B {
    return { a in
      self(a).combine(other(a))
    }
  }
}
```

Combinable

```
extension (A, B): Combinable where A: Combinable, B: Combinable {
  func combine(_ other: (A, B)) -> (A, B) {
    return (
      self.0.combine(other.0),
      self.1.combine(other.1)
    )
  }
}

extension ((A) -> B): Combinable where B: Combinable {
  func combine(_ other: (A) -> B) -> (A) -> B {
    return { a in
      self(a).combine(other(a))
    }
  }
}
```

Combinable

```
extension (A, B): Combinable where A: Combinable, B: Combinable {
  func combine(_ other: (A, B)) -> (A, B) {
    return (
      self.0.combine(other.0),
      self.1.combine(other.1)
    )
  }
}

extension ((A) -> B): Combinable where B: Combinable {
  func combine(_ other: (A) -> B) -> (A) -> B {
    return { a in
      self(a).combine(other(a))
    }
  }
}
```

Combinable

```
extension (A, B): Combinable where A: Combinable, B: Combinable {
  func combine(_ other: (A, B)) -> (A, B) {
    return (
      self.0.combine(other.0),
      self.1.combine(other.1)
    )
  }
}

extension ((A) -> B): Combinable where B: Combinable {
  func combine(_ other: (A) -> B) -> (A) -> B {
    return { a in
      self(a).combine(other(a))
    }
  }
}
```

Combinable

```
extension (A, B): Combinable where A: Combinable, B: Combinable {
  func combine(_ other: (A, B)) -> (A, B) {
    return (
      self.0.combine(other.0),
      self.1.combine(other.1)
    )
  }
}

extension ((A) -> B): Combinable where B: Combinable {
  func combine(_ other: (A) -> B) -> (A) -> B {
    return { a in
      self(a).combine(other(a))
    }
  }
}
```

Combinable

```
[1, 2, 3, 4].reduce(0, +)

extension Collection where Element: Combinable {
  func reduce(_ initial: Element) -> Element {
    return self.reduce(initial) { $0.combine($1) }
  }
}

[1, 2, 3, 4].reduce(0) // 10

["Hello", " ", "World"].reduce("") // "Hello World"

[[1, 2], [3, 4], [5, 6]].reduce([]) // [1, 2, 3, 4, 5, 6]
```

Combinable

```
[1, 2, 3, 4].reduce(0, +)

extension Collection where Element: Combinable {
  func reduce(_ initial: Element) -> Element {
    return self.reduce(initial) { $0.combine($1) }
  }
}

[1, 2, 3, 4].reduce(0) // 10

["Hello", " ", "World"].reduce("") // "Hello World"

[[1, 2], [3, 4], [5, 6]].reduce([]) // [1, 2, 3, 4, 5, 6]
```

Combinable

```
[1, 2, 3, 4].reduce(0, +)

extension Collection where Element: Combinable {
  func reduce(_ initial: Element) -> Element {
    return self.reduce(initial) { $0.combine($1) }
  }
}

[1, 2, 3, 4].reduce(0) // 10

["Hello", " ", "World"].reduce("") // "Hello World"

[[1, 2], [3, 4], [5, 6]].reduce([]) // [1, 2, 3, 4, 5, 6]
```

De-protocolizing

De-protocolizing

```
protocol Combinable {
  func combine(_ other: Self) -> Self
}
```

👇

```
struct Combining<A> {
  let combine: (A, A) -> A
}
```

De-protocolizing

```
protocol Combinable {
  func combine(_ other: Self) -> Self
}
```

👇

```
struct Combining<A> {
  let combine: (A, A) -> A
}
```

De-protocolizing

```
protocol Combinable {
  func combine(_ other: Self) -> Self
}
```

👇

```
struct Combining<A> {
  let combine: (A, A) -> A
}
```

De-protocolizing

```
extension Int: Combinable {
  func combine(_ other: Int) -> Int {
    return self + other
  }
}

let sum = Combining<Int> { $0 + $1 }

extension String: Combinable {
  func combine(_ other: String) -> String {
    return self + other
  }
}

let concat = Combining<String> { $0 + $1 }
```

De-protocolizing

```
extension Int: Combinable {
  func combine(_ other: Int) -> Int {
    return self + other
  }
}

let sum = Combining<Int> { $0 + $1 }

extension String: Combinable {
  func combine(_ other: String) -> String {
    return self + other
  }
}

let concat = Combining<String> { $0 + $1 }
```

De-protocolizing

```
extension Int: Combinable {
  func combine(_ other: Int) -> Int {
    return self + other
  }
}

let sum = Combining<Int> { $0 + $1 }

extension String: Combinable {
  func combine(_ other: String) -> String {
    return self + other
  }
}

let concat = Combining<String> { $0 + $1 }
```

De-protocolizing

```
extension Array {
  func reduce(
    _ initial: Element,
    _ combining: Combining<Element>
    ) -> Element {

      return self.reduce(initial, combining.combine)
  }
}

[1, 2, 3, 4].reduce(0, sum) // 10

[[1, 2], [3, 4]].reduce(0, concat) // [1, 2, 3, 4]
```

De-protocolizing

```
extension Array {
  func reduce(
    _ initial: Element,
    _ combining: Combining<Element>
    ) -> Element {

      return self.reduce(initial, combining.combine)
  }
}

[1, 2, 3, 4].reduce(0, sum) // 10

[[1, 2], [3, 4]].reduce(0, concat) // [1, 2, 3, 4]
```

De-protocolizing

```
let sum = Combining<Int> { $0 + $1 }

let prod = Combining<Int> { $0 * $1 }

[1, 2, 3, 4].reduce(1, sum) // 10

[1, 2, 3, 4].reduce(1, prod) // 24
```

De-protocolizing

```
extension Combining where A: Numeric {
  static var sum: Combining {
    return Combining { $0 + $0 }
  }

  static var prod: Combining {
    return Combining { $0 * $0 }
  }
}

[1, 2, 3, 4].reduce(0, .sum) // 10 as Int

[1, 2, 3, 4].reduce(1, .prod) // 24 as Int

[1.0, 2, 3, 4].reduce(1, .prod) // 24.0 as Double

[CGFloat(1), 2, 3, 4].reduce(1, .prod) // 24.0 as CGFloat
```

De-protocolizing

```
extension Combining where A: Numeric {
  static var sum: Combining {
    return Combining { $0 + $0 }
  }

  static var prod: Combining {
    return Combining { $0 * $0 }
  }
}

[1, 2, 3, 4].reduce(0, .sum) // 10 as Int

[1, 2, 3, 4].reduce(1, .prod) // 24 as Int

[1.0, 2, 3, 4].reduce(1, .prod) // 24.0 as Double

[CGFloat(1), 2, 3, 4].reduce(1, .prod) // 24.0 as CGFloat
```

De-protocolizing

```
extension Combining where A: RangeReplaceableCollection {
  static var concat: Combining {
    return Combining { $0 + $1 }
  }
}

["Hello", " ", "World"].reduce("", .concat)

[[1, 2], [3, 4]].reduce([], .concat)
```

De-protocolizing

```
func zip<A, B>(
  _ a: Combining<A>,
  _ b: Combining<B>
  ) -> Combining<(A, B)> {

    return Combining { lhs, rhs in
      (
        a.combine(lhs.0, rhs.0),
        b.combine(lhs.1, rhs.1)
      )
    }
}
```

De-protocolizing

```
func zip<A, B>(
  _ a: Combining<A>,
  _ b: Combining<B>
  ) -> Combining<(A, B)> {

    return Combining { lhs, rhs in
      (
        a.combine(lhs.0, rhs.0),
        b.combine(lhs.1, rhs.1)
      )
    }
}
```

De-protocolizing

```
func zip<A, B>(
  _ a: Combining<A>,
  _ b: Combining<B>
  ) -> Combining<(A, B)> {

    return Combining { lhs, rhs in
      (
        a.combine(lhs.0, rhs.0),
        b.combine(lhs.1, rhs.1)
      )
    }
}
```

De-protocolizing

```
func zip<A, B>(
  _ a: Combining<A>,
  _ b: Combining<B>
  ) -> Combining<(A, B)> {

    return Combining { lhs, rhs in
      (
        a.combine(lhs.0, rhs.0),
        b.combine(lhs.1, rhs.1)
      )
    }
}
```

De-protocolizing

```
func zip<A, B>(
  _ a: Combining<A>,
  _ b: Combining<B>
  ) -> Combining<(A, B)> {

    return Combining { lhs, rhs in
      (
        a.combine(lhs.0, rhs.0),
        b.combine(lhs.1, rhs.1)
      )
    }
}
```

De-protocolizing

```
[
  (1, "Hello"),
  (2, " "),
  (3, "World"),
  (4, "!")
  ]
  .reduce(zip(.sum, .concat))

// (10, "Hello World!")
```

De-protocolizing

```
func pointwise<A, B>(
  _ b: Combining<B>
  ) -> Combining<(A) -> B> {

    return Combining { f, g in
      return { a in
        b.combine(f(a), g(a))
      }
    }
}
```

De-protocolizing

```
func pointwise<A, B>(
  _ b: Combining<B>
  ) -> Combining<(A) -> B> {

    return Combining { f, g in
      return { a in
        b.combine(f(a), g(a))
      }
    }
}
```

De-protocolizing

```
func pointwise<A, B>(
  _ b: Combining<B>
  ) -> Combining<(A) -> B> {

    return Combining { f, g in
      return { a in
        b.combine(f(a), g(a))
      }
    }
}
```

De-protocolizing

```
func pointwise<A, B>(
  _ b: Combining<B>
  ) -> Combining<(A) -> B> {

    return Combining { f, g in
      return { a in
        b.combine(f(a), g(a))
      }
    }
}
```

Case Study

Case Study: Snapshot Testing

https://github.com/pointfreeco/swift-snapshot-testing

pointfreeco / swift-snapshot-testing

Unwatch 19 | Unstar 900 | Fork 62

Delightful Swift snapshot testing. https://www.pointfree.co/episodes/ep4...

Edit

swift | testing | snapshot-testing | screenshot-testing | Manage topics

165 commits | 19 branches | 6 releases | 25 contributors | MIT

Branch: master | New pull request | Create new file | Upload files | Find File | Clone or download

pavel-y-ivanov and stephencelis Disable bitcode for target xcconfig (#221) ··· Latest commit 75eaeb1 6 days ago

.circleci	Swift 5 (#200)	a month ago
.github	Update README (#118)	5 months ago
Documentation	Documentation fixes	13 days ago
SnapshotTesting.xcodeproj	Inline Snapshot Testing (#199)	a month ago
Sources	Allow dynamic size of views based on content sizes (#217)	18 days ago
Tests	Allow dynamic size of views based on content sizes (#217)	18 days ago
.dockerignore	Add Travis CI for linux testing! (#33)	2 years ago
.gitignore	Swift 5 (#200)	a month ago
.swift-version	1.4.0	a month ago
.travis.yml	Strategies (#51)	7 months ago
CODE_OF_CONDUCT.md	Update README (#118)	5 months ago

What is snapshot testing?

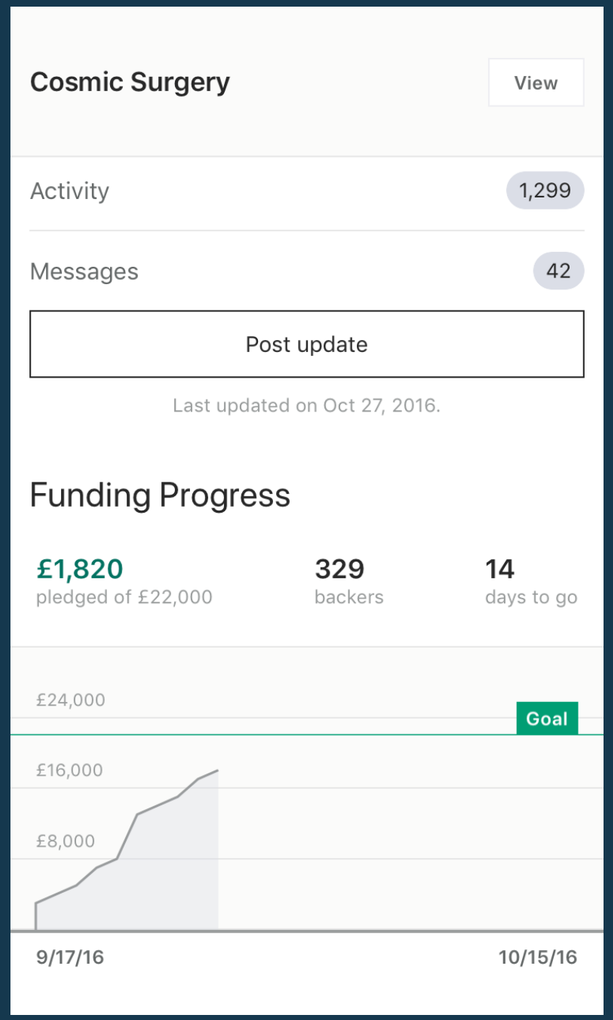
What is snapshot testing?

```
import XCTest

XCTAssertEqual(
  42,
  compute(3)
)
```

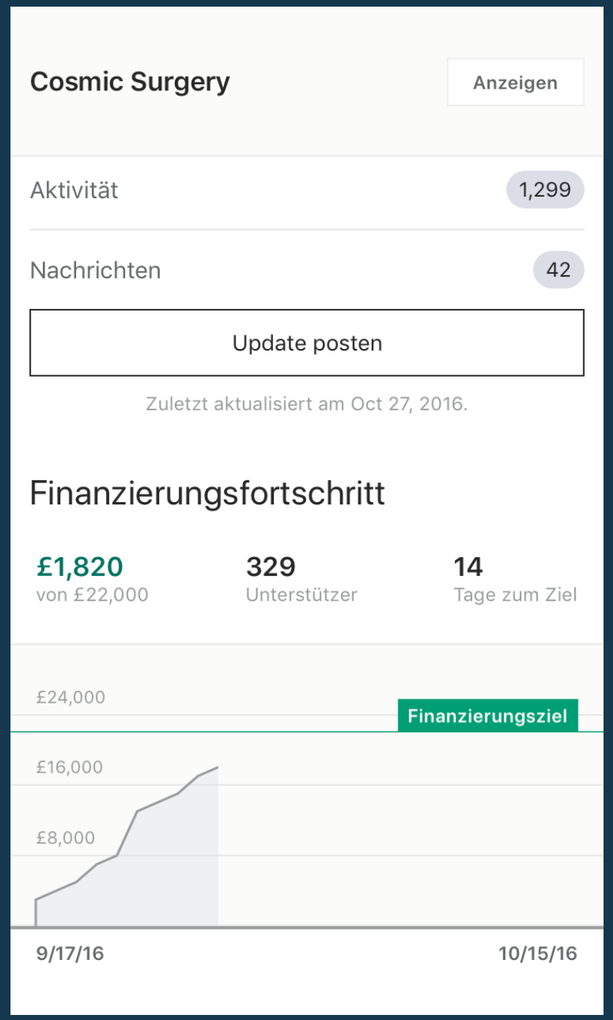
What is snapshot testing?

```
import SnapshotTesting

assertSnapshot(matching: compute(3))
```

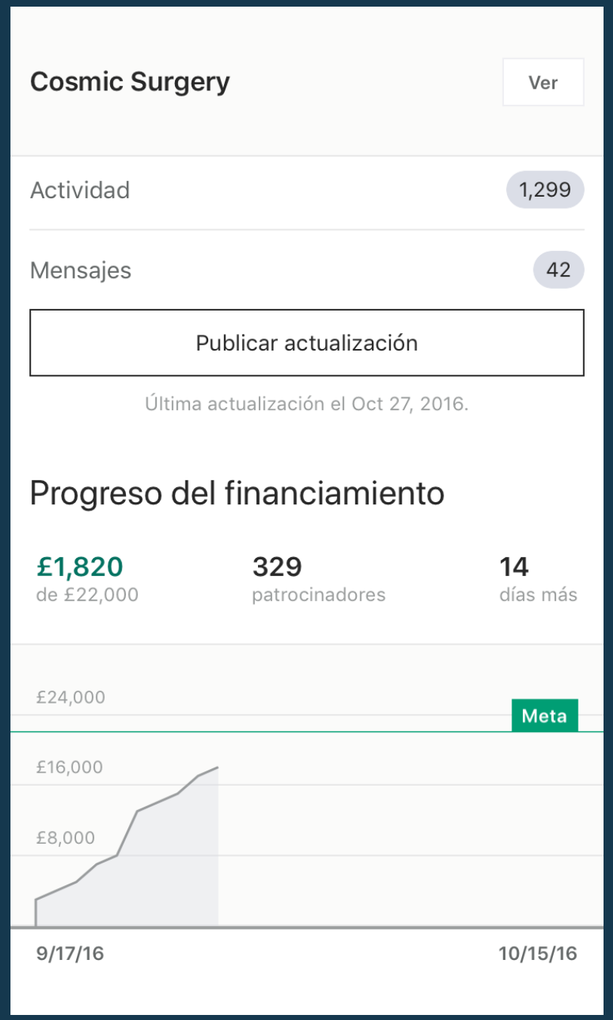
What is snapshot testing?



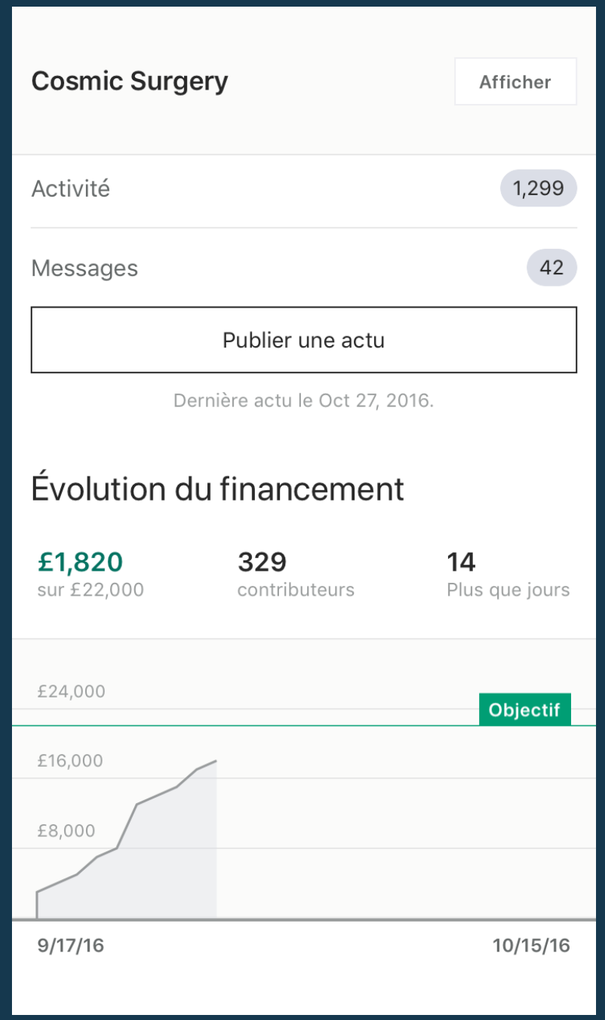
What is snapshot testing?



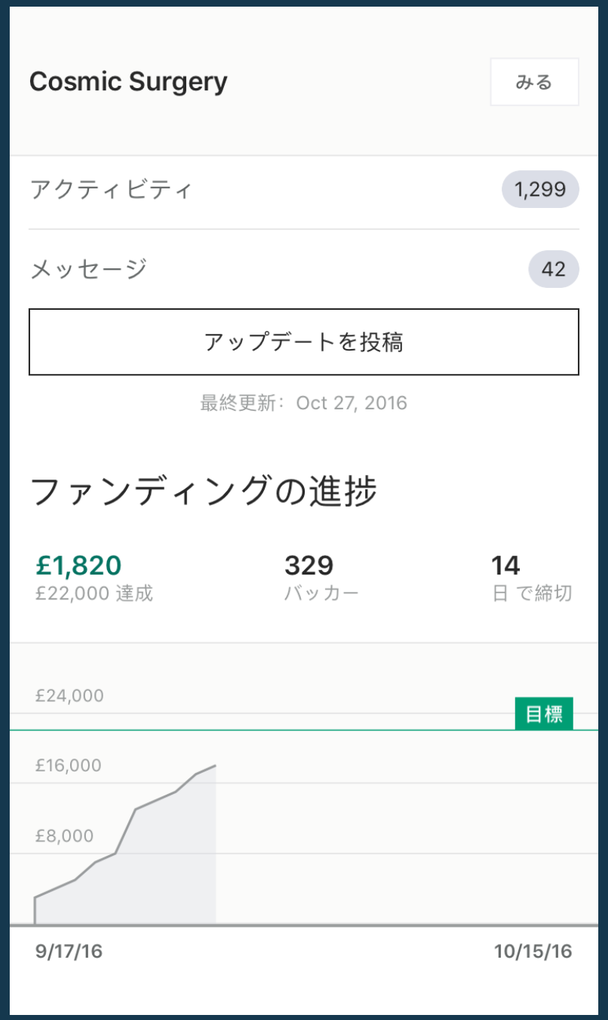
What is snapshot testing?



What is snapshot testing?



What is snapshot testing?



What is snapshot testing?

```
assertSnapshot(matching: request)
```

```
POST https://api.stripe.com/v1/subscriptions/sub_test?expand%5B%5D=customer
Authorization: Basic aHR0cHM6Ly93d3cucG9pbnRmcmVlLmNv

coupon=&items[0][id]=si_test&items[0][plan]=individual-yearly&items[0][quantity]=1
```

Protocol-oriented snapshot testing

Protocol-oriented snapshot testing

```
protocol Diffable {
  static func diff(old: Self, new: Self) -> (String, [XCTAttachment])?
  var data: Data { get }
  static func from(data: Data) -> Self
}

protocol Snapshottable {
  associatedtype Format: Diffable
  static var pathExtension: String { get }
  var snapshot: Format { get }
}
```

Protocol-oriented snapshot testing

```
protocol Diffable {
  static func diff(old: Self, new: Self) -> (String, [XCTAttachment])?
  var data: Data { get }
  static func from(data: Data) -> Self
}

protocol Snapshottable {
  associatedtype Format: Diffable
  static var pathExtension: String { get }
  var snapshot: Format { get }
}
```

Protocol-oriented snapshot testing

```
protocol Diffable {
  static func diff(old: Self, new: Self) -> (String, [XCTAttachment])?
  var data: Data { get }
  static func from(data: Data) -> Self
}

protocol Snapshottable {
  associatedtype Format: Diffable
  static var pathExtension: String { get }
  var snapshot: Format { get }
}
```

Protocol-oriented snapshot testing

```
protocol Diffable {
  static func diff(old: Self, new: Self) -> (String, [XCTAttachment])?
  var data: Data { get }
  static func from(data: Data) -> Self
}

protocol Snapshottable {
  associatedtype Format: Diffable
  static var pathExtension: String { get }
  var snapshot: Format { get }
}
```

Protocol-oriented snapshot testing

```
protocol Diffable {
  static func diff(old: Self, new: Self) -> (String, [XCTAttachment])?
  var data: Data { get }
  static func from(data: Data) -> Self
}

protocol Snapshottable {
  associatedtype Format: Diffable
  static var pathExtension: String { get }
  var snapshot: Format { get }
}
```

Protocol-oriented snapshot testing

```
protocol Diffable {
  static func diff(old: Self, new: Self) -> (String, [XCTAttachment])?
  var data: Data { get }
  static func from(data: Data) -> Self
}

protocol Snapshottable {
  associatedtype Format: Diffable
  static var pathExtension: String { get }
  var snapshot: Format { get }
}
```

Protocol-oriented snapshot testing

```
protocol Diffable {
  static func diff(old: Self, new: Self) -> (String, [XCTAttachment])?
  var data: Data { get }
  static func from(data: Data) -> Self
}

protocol Snapshottable {
  associatedtype Format: Diffable
  static var pathExtension: String { get }
  var snapshot: Format { get }
}
```

Protocol-oriented snapshot testing

```
protocol Diffable {
  static func diff(old: Self, new: Self) -> (String, [XCTAttachment])?
  var data: Data { get }
  static func from(data: Data) -> Self
}

protocol Snapshottable {
  associatedtype Format: Diffable
  static var pathExtension: String { get }
  var snapshot: Format { get }
}
```

Protocol-oriented snapshot testing

```
protocol Diffable {
  static func diff(old: Self, new: Self) -> (String, [XCTAttachment])?
  var data: Data { get }
  static func from(data: Data) -> Self
}

protocol Snapshottable {
  associatedtype Format: Diffable
  static var pathExtension: String { get }
  var snapshot: Format { get }
}
```

Protocol-oriented snapshot testing

```
func assertSnapshot<A: Snapshottable>(
  matching: A
) {
  // ...
}
```

Witness-oriented snapshot testing

```
protocol Diffable {
  static func diff(old: Self, new: Self) -> (String, [XCTAttachment])?
  var data: Data { get }
  static func from(data: Data) -> Self
}
```

👇

```
struct Diffing<Value> {
  let diff: (Value, Value) -> (String, [XCTAttachment])?
  let data: (Value) -> Data
  let from: (Data) -> Value
}
```

Witness-oriented snapshot testing

```
protocol Diffable {
  static func diff(old: Self, new: Self) -> (String, [XCTAttachment])?
  var data: Data { get }
  static func from(data: Data) -> Self
}
```

👇

```
struct Diffing<Value> {
  let diff: (Value, Value) -> (String, [XCTAttachment])?
  let data: (Value) -> Data
  let from: (Data) -> Value
}
```

Witness-oriented snapshot testing

```
protocol Diffable {
  static func diff(old: Self, new: Self) -> (String, [XCTAttachment])?
  var data: Data { get }
  static func from(data: Data) -> Self
}
```

👇

```
struct Diffing<Value> {
  let diff: (Value, Value) -> (String, [XCTAttachment])?
  let data: (Value) -> Data
  let from: (Data) -> Value
}
```

Witness-oriented snapshot testing

```
protocol Diffable {
  static func diff(old: Self, new: Self) -> (String, [XCTAttachment])?
  var data: Data { get }
  static func from(data: Data) -> Self
}
```

👇

```
struct Diffing<Value> {
  let diff: (Value, Value) -> (String, [XCTAttachment])?
  let data: (Value) -> Data
  let from: (Data) -> Value
}
```

Witness-oriented snapshot testing

```
protocol Diffable {
  static func diff(old: Self, new: Self) -> (String, [XCTAttachment])?
  var data: Data { get }
  static func from(data: Data) -> Self
}
```

👇

```
struct Diffing<Value> {
  let diff: (Value, Value) -> (String, [XCTAttachment])?
  let data: (Value) -> Data
  let from: (Data) -> Value
}
```

Witness-oriented snapshot testing

```
protocol Snapshottable {
  associatedtype Format: Diffable
  static var pathExtension: String { get }
  var snapshot: Format { get }
}
```

👇

```
struct Snapshotting<Value, Format> {
  let diffing: Diffing<Format>
  let pathExtension: String
  let snapshot: (A) -> Format
}
```

Witness-oriented snapshot testing

```
protocol Snapshottable {
  associatedtype Format: Diffable
  static var pathExtension: String { get }
  var snapshot: Format { get }
}
```

👇

```
struct Snapshotting<Value, Format> {
  let diffing: Diffing<Format>
  let pathExtension: String
  let snapshot: (A) -> Format
}
```

Witness-oriented snapshot testing

```
protocol Snapshottable {
  associatedtype Format: Diffable
  static var pathExtension: String { get }
  var snapshot: Format { get }
}
```

👇

```
struct Snapshotting<Value, Format> {
  let diffing: Diffing<Format>
  let pathExtension: String
  let snapshot: (A) -> Format
}
```

Witness-oriented snapshot testing

```
protocol Snapshottable {
  associatedtype Format: Diffable
  static var pathExtension: String { get }
  var snapshot: Format { get }
}
```

👇

```
struct Snapshotting<Value, Format> {
  let diffing: Diffing<Format>
  let pathExtension: String
  let snapshot: (A) -> Format
}
```

Witness-oriented snapshot testing

```
protocol Snapshottable {
  associatedtype Format: Diffable
  static var pathExtension: String { get }
  var snapshot: Format { get }
}
```

👇

```
struct Snapshotting<Value, Format> {
  let diffing: Diffing<Format>
  let pathExtension: String
  let snapshot: (A) -> Format
}
```

Witness-oriented snapshot testing

```
protocol Snapshottable {
  associatedtype Format: Diffable
  static var pathExtension: String { get }
  var snapshot: Format { get }
}
```

👇

```
struct Snapshotting<Value, Format> {
  let diffing: Diffing<Format>
  let pathExtension: String
  let snapshot: (A) -> Format
}
```

Witness-oriented snapshot testing

```
func assertSnapshot<A>(
  matching: A,
  as: Snapshotting<A>
) {
  // ...
}
```

Snapshot Strategies

Snapshot Strategies: dump

```
assertSnapshot(matching: user, as: .dump)
```

```
▽ User
  - bio: "Blobbed around the world."
  - id: 1
  - name: "Blobby"
```

Snapshot Strategies: `json`

```
assertSnapshot(matching: user, as: .json)
```

```
{
  "bio" : "Blobbed around the world.",
  "id" : 1,
  "name" : "Blobby"
}
```

Snapshot Strategies: URLRequest raw

```
assertSnapshot(matching: request, as: .raw)

POST http://localhost:8080/account
Cookie: pf_session={"userId":"1"}

email=blob%40pointfree.co&name=Blob
```

Snapshot Strategies: URLRequest curl

```
assertSnapshot(matching: request, as: .curl)

curl \
  --request POST \
  --header "Accept: text/html" \
  --data 'pricing[billing]=monthly&pricing[lane]=individual' \
  --cookie "pf_session={\"user_id\":\"1\"}" \
  "https://www.pointfree.co/subscribe"
```

Snapshot Strategies: `image`

```
assertSnapshot(
  matching: view,
  as: .image(traits: .init(horizontalSizeClass: .regular))
)

assertSnapshot(matching: vc, on: .iPhoneX(.portrait))

assertSnapshot(matching: vc, on: .iPhoneX(.landscape))
```

Snapshot Strategies: image

```
assertSnapshot(
  matching: view,
  as: .image(traits: .init(horizontalSizeClass: .regular))
)

assertSnapshot(matching: vc, on: .iPhoneX(.portrait))

assertSnapshot(matching: vc, on: .iPhoneX(.landscape))
```

Snapshot Strategies: image

```
assertSnapshot(
  matching: view,
  as: .image(traits: .init(horizontalSizeClass: .regular))
)

assertSnapshot(matching: vc, on: .iPhoneX(.portrait))

assertSnapshot(matching: vc, on: .iPhoneX(.landscape))
```

Snapshot Strategies: image

```
assertSnapshot(
  matching: view,
  as: .image(traits: .init(horizontalSizeClass: .regular))
)

assertSnapshot(matching: vc, on: .iPhoneX(.portrait))

assertSnapshot(matching: vc, on: .iPhoneX(.landscape))
```

Snapshot Strategies: `recursiveDescription`

```
assertSnapshot(matching: view, as: .recursiveDescription)
```

```
<UIView; frame = (0 0; 1024 768); autoresize = W+H; layer = <CALayer>>
   | <UILabel; frame = (484.5 20; 55.5 20.5); text = 'What's'; userInteractionEnabled = NO; layer = <_UILabelLayer>>
   | <UILabel; frame = (0 384; 25 20.5); text = 'the'; userInteractionEnabled = NO; layer = <_UILabelLayer>>
   | <UILabel; frame = (985 384; 39 20.5); text = 'point'; userInteractionEnabled = NO; layer = <_UILabelLayer>>
   | <UILabel; frame = (508.5 750; 7.5 18); text = '?'; userInteractionEnabled = NO; layer = <_UILabelLayer>>
```

Snapshot Strategies: `hierarchy`

```
assertSnapshot(matching: vc, as: .hierarchy)
```

```
<UITabBarController>, state: appeared, view: <UILayoutContainerView>
   | <UINavigationController>, state: appeared, view: <UILayoutContainerView>
   |    | <UIPageViewController>, state: appeared, view: <_UIPageViewControllerContentView>
   |    |    | <UIViewController>, state: appeared, view: <UIView>
   | <UINavigationController>, state: disappeared, view: <UILayoutContainerView>
   |    | <UIViewController>, state: disappeared, view: (view not loaded)
   | <UINavigationController>, state: disappeared, view: <UILayoutContainerView>
   |    | <UIViewController>, state: disappeared, view: (view not loaded)
   | <UINavigationController>, state: disappeared, view: <UILayoutContainerView>
   |    | <UIViewController>, state: disappeared, view: (view not loaded)
   | <UINavigationController>, state: disappeared, view: <UILayoutContainerView>
   |    | <UIViewController>, state: disappeared, view: (view not loaded)
```

Snapshot Strategies: hierarchy

```
assertSnapshot(matching: vc, as: .hierarchy)
```

```
<UITabBarController>, state: appeared, view: <UILayoutContainerView>
   | <UINavigationController>, state: appeared, view: <UILayoutContainerView>
   |    | <UIPageViewController>, state: appeared, view: <_UIPageViewControllerContentView>
   |    |    | <UIViewController>, state: appeared, view: <UIView>
   | <UINavigationController>, state: disappeared, view: <UILayoutContainerView>
   |    | <UIViewController>, state: disappeared, view: (view not loaded)
   | <UINavigationController>, state: disappeared, view: <UILayoutContainerView>
   |    | <UIViewController>, state: disappeared, view: (view not loaded)
   | <UINavigationController>, state: disappeared, view: <UILayoutContainerView>
   |    | <UIViewController>, state: disappeared, view: (view not loaded)
   | <UINavigationController>, state: disappeared, view: <UILayoutContainerView>
   |    | <UIViewController>, state: disappeared, view: (view not loaded)
```

Snapshot Strategies: hierarchy

```
assertSnapshot(matching: vc, as: .hierarchy)
```

```
<UITabBarController>, state: appeared, view: <UILayoutContainerView>
   | <UINavigationController>, state: appeared, view: <UILayoutContainerView>
   |    | <UIPageViewController>, state: appeared, view: <_UIPageViewControllerContentView>
   |    |    | <UIViewController>, state: appeared, view: <UIView>
   | <UINavigationController>, state: disappeared, view: <UILayoutContainerView>
   |    | <UIViewController>, state: disappeared, view: (view not loaded)
   | <UINavigationController>, state: disappeared, view: <UILayoutContainerView>
   |    | <UIViewController>, state: disappeared, view: (view not loaded)
   | <UINavigationController>, state: disappeared, view: <UILayoutContainerView>
   |    | <UIViewController>, state: disappeared, view: (view not loaded)
   | <UINavigationController>, state: disappeared, view: <UILayoutContainerView>
   |    | <UIViewController>, state: disappeared, view: (view not loaded)
```

Snapshot Strategies: hierarchy

```
assertSnapshot(matching: vc, as: .hierarchy)
```

```
<UITabBarController>, state: appeared, view: <UILayoutContainerView>
   | <UINavigationController>, state: appeared, view: <UILayoutContainerView>
   |    | <UIPageViewController>, state: appeared, view: <_UIPageViewControllerContentView>
   |    |    | <UIViewController>, state: appeared, view: <UIView>
   | <UINavigationController>, state: disappeared, view: <UILayoutContainerView>
   |    | <UIViewController>, state: disappeared, view: (view not loaded)
   | <UINavigationController>, state: disappeared, view: <UILayoutContainerView>
   |    | <UIViewController>, state: disappeared, view: (view not loaded)
   | <UINavigationController>, state: disappeared, view: <UILayoutContainerView>
   |    | <UIViewController>, state: disappeared, view: (view not loaded)
   | <UINavigationController>, state: disappeared, view: <UILayoutContainerView>
   |    | <UIViewController>, state: disappeared, view: (view not loaded)
```

Snapshot Strategies: pdf

https://github.com/WeirdMath/SwiftyHaru

```
assertSnapshot(matching: document, as: .pdf)
```

Snapshot Strategies: `gif`

```
assertSnapshot(
  matching: canvas,
  as: .gif(of: animation, duration: 1, framesPerSecond: 60)
)
```

Snapshot Strategies: `gif`

```
assertSnapshot(
  matching: canvas,
  as: .gif(of: animation, duration: 1, framesPerSecond: 60)
)
```

Snapshot Strategies: `gif`

Snapshot Strategies: `gif`

Snapshot Strategies: onion

```
assertSnapshot(
  matching: canvas,
  as: .onion(of: circleAnimation, frames: 20)
)
```

Transforming existing strategies into new strategies

Pullbacks

```
extension Snapshotting {
  func pullback<NewValue>(
    _ f: (NewValue) -> Value
    ) -> Snapshotting<NewValue, Format> {

      return Snapshotting(
        diffing: self.diffing,
        pathExtension: self.pathExtension,
        snapshot: { newValue in
          self.snapshot(f(newValue))
        }
      )
  }
}
```

Pullbacks

```
extension Snapshotting {
  func pullback<NewValue>(
    _ f: (NewValue) -> Value
    ) -> Snapshotting<NewValue, Format> {

      return Snapshotting(
        diffing: self.diffing,
        pathExtension: self.pathExtension,
        snapshot: { newValue in
          self.snapshot(f(newValue))
        }
      )
  }
}
```

Pullbacks

```
extension Snapshotting {
  func pullback<NewValue>(
    _ f: (NewValue) -> Value
    ) -> Snapshotting<NewValue, Format> {

      return Snapshotting(
        diffing: self.diffing,
        pathExtension: self.pathExtension,
        snapshot: { newValue in
          self.snapshot(f(newValue))
        }
      )
  }
}
```

Pullbacks

```
extension Snapshotting {
  func pullback<NewValue>(
    _ f: (NewValue) -> Value
    ) -> Snapshotting<NewValue, Format> {

      return Snapshotting(
        diffing: self.diffing,
        pathExtension: self.pathExtension,
        snapshot: { newValue in
          self.snapshot(f(newValue))
        }
      )
  }
}
```

Pullbacks

```
extension Snapshotting {
  func pullback<NewValue>(
    _ f: (NewValue) -> Value
    ) -> Snapshotting<NewValue, Format> {

      return Snapshotting(
        diffing: self.diffing,
        pathExtension: self.pathExtension,
        snapshot: { newValue in
          self.snapshot(f(newValue))
        }
      )
  }
}
```

Pullbacks

```
extension Snapshotting {
  func pullback<NewValue>(
    _ f: (NewValue) -> Value
    ) -> Snapshotting<NewValue, Format> {

      return Snapshotting(
        diffing: self.diffing,
        pathExtension: self.pathExtension,
        snapshot: { newValue in
          self.snapshot(f(newValue))
        }
      )
  }
}
```

Pullbacks

```
let image: Snapshotting<UIImage, UIImage> = ...

let layer: Snapshotting<CALayer, UIImage> =
  image.pullback { layer in
    UIGraphicsImageRenderer(size: layer.bounds.size)
      .image { ctx in layer.render(in: ctx.cgContext) }
}

let view: Snapshotting<UIView, UIImage> =
  layer.pullback { $0.layer }

let viewController: Snapshotting<UIViewController, UIImage> =
  view.pullback { $0.view }
```

Pullbacks

```
let image: Snapshotting<UIImage, UIImage> = ...

let layer: Snapshotting<CALayer, UIImage> =
  image.pullback { layer in
    UIGraphicsImageRenderer(size: layer.bounds.size)
      .image { ctx in layer.render(in: ctx.cgContext) }
}

let view: Snapshotting<UIView, UIImage> =
  layer.pullback { $0.layer }

let viewController: Snapshotting<UIViewController, UIImage> =
  view.pullback { $0.view }
```

Pullbacks

```
let image: Snapshotting<UIImage, UIImage> = ...

let layer: Snapshotting<CALayer, UIImage> =
  image.pullback { layer in
    UIGraphicsImageRenderer(size: layer.bounds.size)
      .image { ctx in layer.render(in: ctx.cgContext) }
}

let view: Snapshotting<UIView, UIImage> =
  layer.pullback { $0.layer }

let viewController: Snapshotting<UIViewController, UIImage> =
  view.pullback { $0.view }
```

Pullbacks

```
let image: Snapshotting<UIImage, UIImage> = ...

let layer: Snapshotting<CALayer, UIImage> =
  image.pullback { layer in
    UIGraphicsImageRenderer(size: layer.bounds.size)
      .image { ctx in layer.render(in: ctx.cgContext) }
}

let view: Snapshotting<UIView, UIImage> =
  layer.pullback { $0.layer }

let viewController: Snapshotting<UIViewController, UIImage> =
  view.pullback { $0.view }
```

Pullbacks

```
let image: Snapshotting<UIImage, UIImage> = ...

let layer: Snapshotting<CALayer, UIImage> =
  image.pullback { layer in
    UIGraphicsImageRenderer(size: layer.bounds.size)
      .image { ctx in layer.render(in: ctx.cgContext) }
}

let view: Snapshotting<UIView, UIImage> =
  layer.pullback { $0.layer }

let viewController: Snapshotting<UIViewController, UIImage> =
  view.pullback { $0.view }
```

Pullbacks

```
let image: Snapshotting<UIImage, UIImage> = ...

let layer: Snapshotting<CALayer, UIImage> =
  image.pullback { layer in
    UIGraphicsImageRenderer(size: layer.bounds.size)
      .image { ctx in layer.render(in: ctx.cgContext) }
}

let view: Snapshotting<UIView, UIImage> =
  layer.pullback { $0.layer }

let viewController: Snapshotting<UIViewController, UIImage> =
  view.pullback { $0.view }
```

Pullbacks

```
let image: Snapshotting<UIImage, UIImage> = ...

let layer: Snapshotting<CALayer, UIImage> =
  image.pullback { layer in
    UIGraphicsImageRenderer(size: layer.bounds.size)
      .image { ctx in layer.render(in: ctx.cgContext) }
}

let view: Snapshotting<UIView, UIImage> =
  layer.pullback { $0.layer }

let viewController: Snapshotting<UIViewController, UIImage> =
  view.pullback { $0.view }
```

Pullbacks

```
let image: Snapshotting<UIImage, UIImage> = ...

let layer: Snapshotting<CALayer, UIImage> =
  image.pullback { layer in
    UIGraphicsImageRenderer(size: layer.bounds.size)
      .image { ctx in layer.render(in: ctx.cgContext) }
}

let view: Snapshotting<UIView, UIImage> =
  layer.pullback { $0.layer }

let viewController: Snapshotting<UIViewController, UIImage> =
  view.pullback { $0.view }
```

Conclusion

Conclusion

— Protocols are the de facto abstraction tool in Swift

Conclusion

- Protocols are the de facto abstraction tool in Swift
- Protocols still have problems, but they are getting better

Conclusion

- Protocols are the de facto abstraction tool in Swift
- Protocols still have problems, but they are getting better
- There is a process to turn protocols into concrete data types

Conclusion

- Protocols are the de facto abstraction tool in Swift
- Protocols still have problems, but they are getting better
- There is a process to turn protocols into concrete data types
- Doing so can fix some protocols problems and expose interesting transformations

— Pros

- Pros
  - ✅ Gets around many protocol shortcomings

- Pros
  - ✅ Gets around many protocol shortcomings
  - ✅ Encourages multiple conformances

- Pros
  - ✅ Gets around many protocol shortcomings
  - ✅ Encourages multiple conformances
  - ✅ Build new conformances from existing ones

- Pros
  - ✅ Gets around many protocol shortcomings
  - ✅ Encourages multiple conformances
  - ✅ Build new conformances from existing ones
  - ✅ Reveals transformations previously hidden

- Pros
  - ✅ Gets around many protocol shortcomings
  - ✅ Encourages multiple conformances
  - ✅ Build new conformances from existing ones
  - ✅ Reveals transformations previously hidden
- Cons

- Pros
  - ✅ Gets around many protocol shortcomings
  - ✅ Encourages multiple conformances
  - ✅ Build new conformances from existing ones
  - ✅ Reveals transformations previously hidden
- Cons
  - 💔 Ubiquitous protocols

- Pros
  - ✅ Gets around many protocol shortcomings
  - ✅ Encourages multiple conformances
  - ✅ Build new conformances from existing ones
  - ✅ Reveals transformations previously hidden
- Cons
  - 💔 Ubiquitous protocols
  - 💔 Deep protocol hiearchies

- Pros
  - ✅ Gets around many protocol shortcomings
  - ✅ Encourages multiple conformances
  - ✅ Build new conformances from existing ones
  - ✅ Reveals transformations previously hidden
- Cons
  - 💔 Ubiquitous protocols
  - 💔 Deep protocol hiearchies
  - 💔 Advanced protocols

Thanks!

POINT·FREE

— Brandon Williams

— @mbrandonw

— www.pointfree.co