Where There's a nil, There's a Way

Detroit.rb

Mike Schutte

May 22, 2019

— 

— 

—   

—  

— TED

computer science in every high school

Microsoft Philanthropies

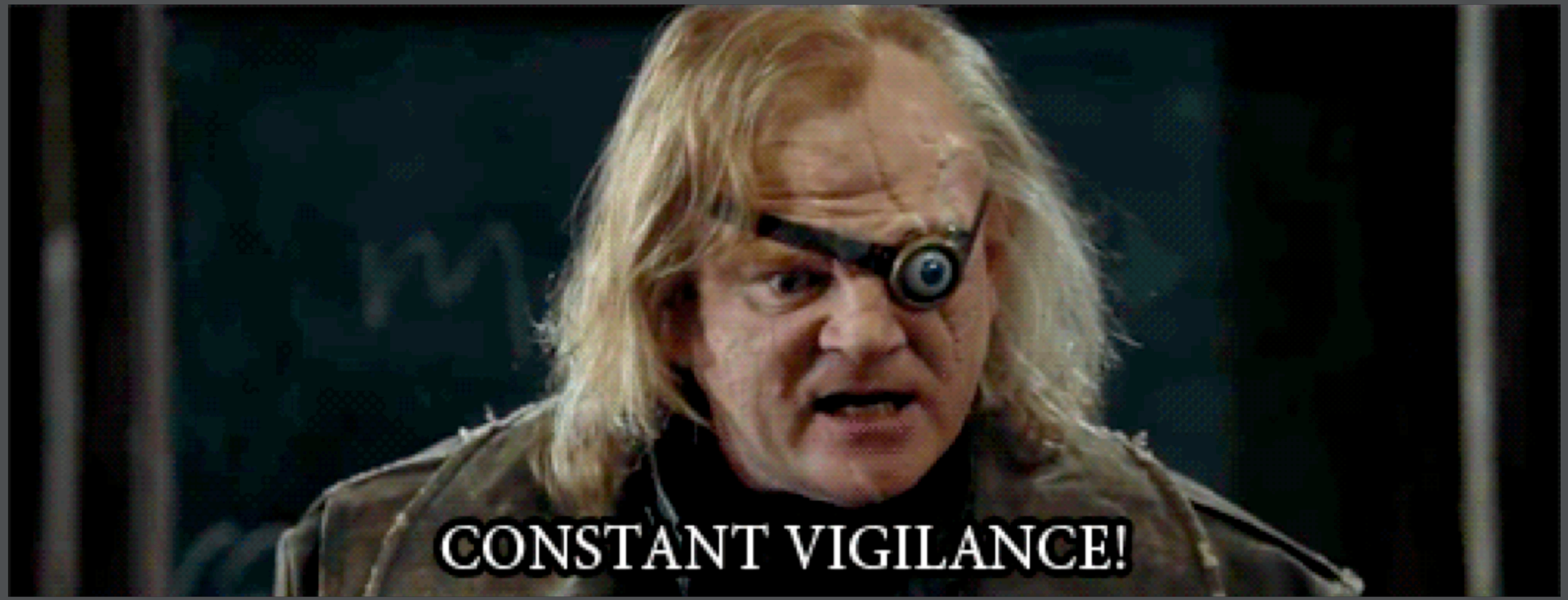
tealsk12.org

Victory Conditions

— Know how/when to use fetch and dig

— Understand nothingness as a concept more generally

— Feel more suspect of nil entering your Ruby programs



`nil` 👀

Roadmap

- API of fetch and dig
- Maybe values (as a concept)
- Message Passing
- Type Coercion
- Review

fetch and dig

- Array
- Hash

Basic API

```
lost = [4, 8, 15, 16, [23, 42]]

lost[1] == lost.fetch(1)

lost[4][1] == lost.dig(4, 1)

user = {
  name: "John Locke",
  physical_attributes: {
    hair: :bald,
    height: :tall
  },
  bio: "looking for meaning",
}

user[:name] == user.fetch(:name)

user[:physical_attributes][:hair] == user.dig(:physical_attributes, :hair)
```

Defaults for fetch

```
lost = [4, 8, 15, 16, [23, 42]]

# lost[10] || 0
(10 < lost.length ? lost[10] : 0) == lost.fetch(10, 0)

user = {
  name: "John Locke",
  physical_attributes: {
    hair: :bald,
    height: :tall
  },
  bio: "looking for meaning",
}

# user[:nickname] || "no nickname"
(user.has_key?(:nickname) ? user[:nickname] : "no nickname") == user.fetch(:nickname, "no nickname")
```

Index and Key Errors for fetch

```
lost = [4, 8, 15, 16, [23, 42]]

lost.fetch(10)
# => IndexError (index 10 outside of array bounds: -5...5)

user = {
  name: "John Locke",
  physical_attributes: {
    hair: :bald,
    height: :tall
  },
  bio: "looking for meaning",
}

user.fetch(:height)
# => KeyError (key not found: :height)
```

What about dig's defaults and errors?

— 😢

```ruby
user = {
  name: "John Locke",
  physical_attributes: {
    hair: :bald,
    height: :tall
  },
  bio: "looking for meaning",
}

user.fetch(:height)
# => KeyError (key not found: :height)

user.dig(:height)
# => nil

user.dig(:bio, :about)
# => TypeError: String does not have #dig method

user.dig(:physical_attributes, :eye_color) || "no eyes"
user.dig(:physical_attributes, :eye_color).to_s
```

Performance

- fetch: negligible difference
  - https://keepthecodesimple.com/ruby-fetch/
- dig: 1.5x slower (still fast)
  - https://tiagoamaro.com.br/2016/08/27/ruby-2-3-dig/

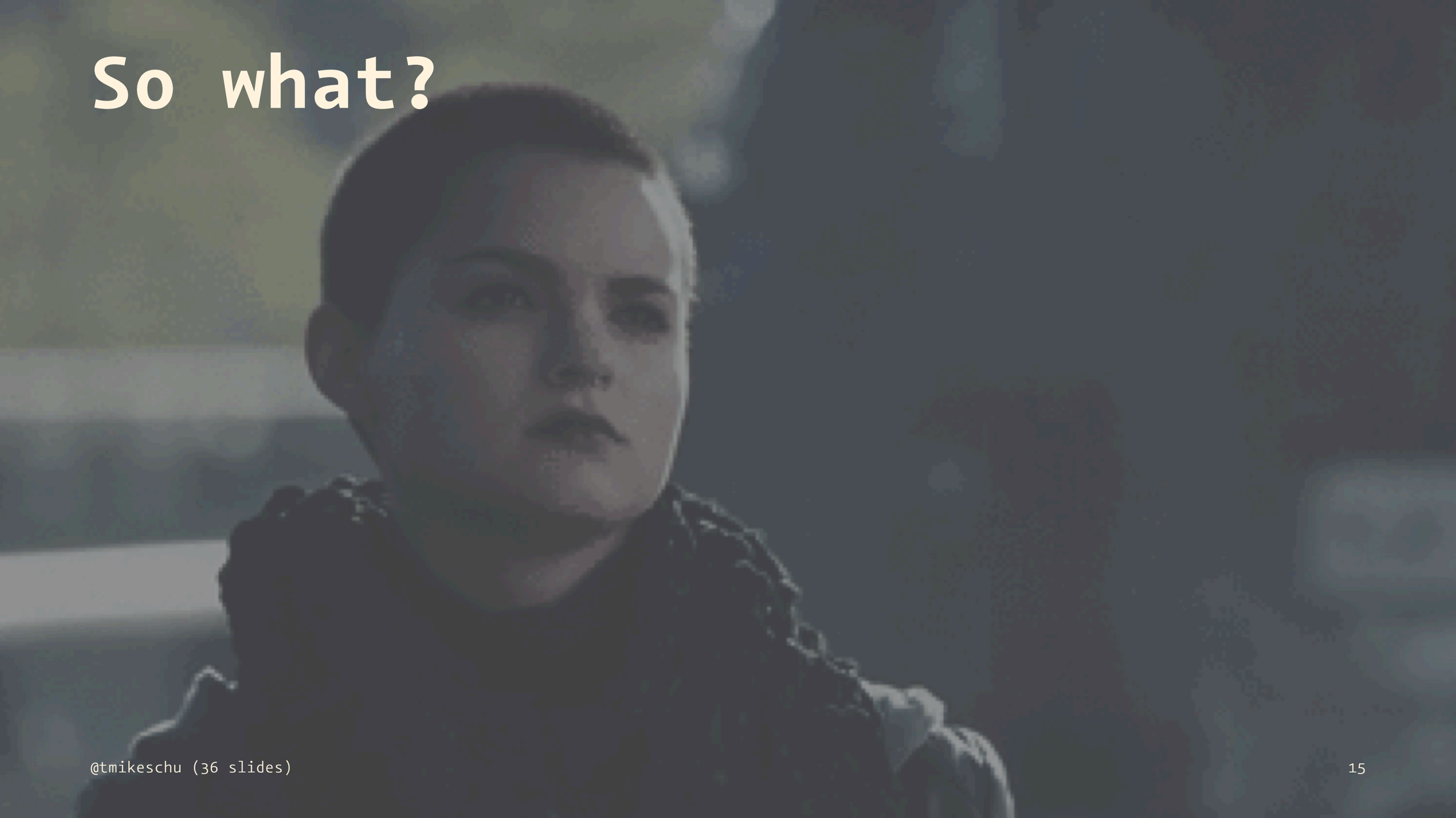
When is this useful?

— Rails controllers or services parsing params

— Getting the first item in an array

— Parsing config and option hashes

So what?



Maybe

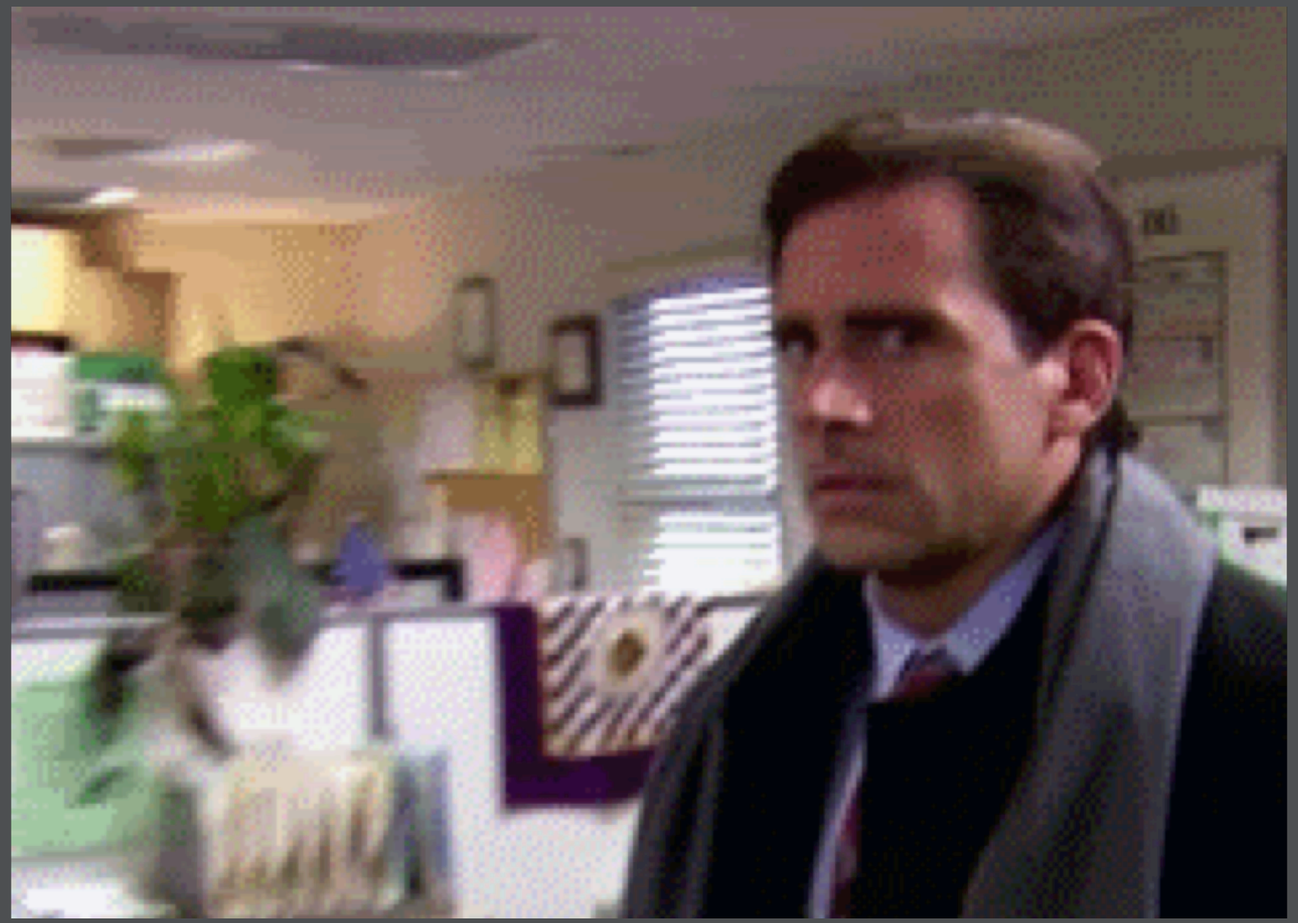
```elm
find : (a -> Bool) -> List a -> Maybe a
find predicate xs =
    xs
    |> List.filter predicate
    |> List.head


ages : List Int
ages = [10, 15, 21, 22]

firstAdult =
  find (\age -> age >= 18) ages
-- => Just 21

firstBaby =
  find (\age -> age < 3) ages
-- => Nothing
```

```elm
find : (a -> Bool) -> List a -> Maybe a
find predicate xs =
    xs -- List a
    |> List.filter predicate -- List a (elements that pass predicate)
    |> List.head -- Maybe a
```

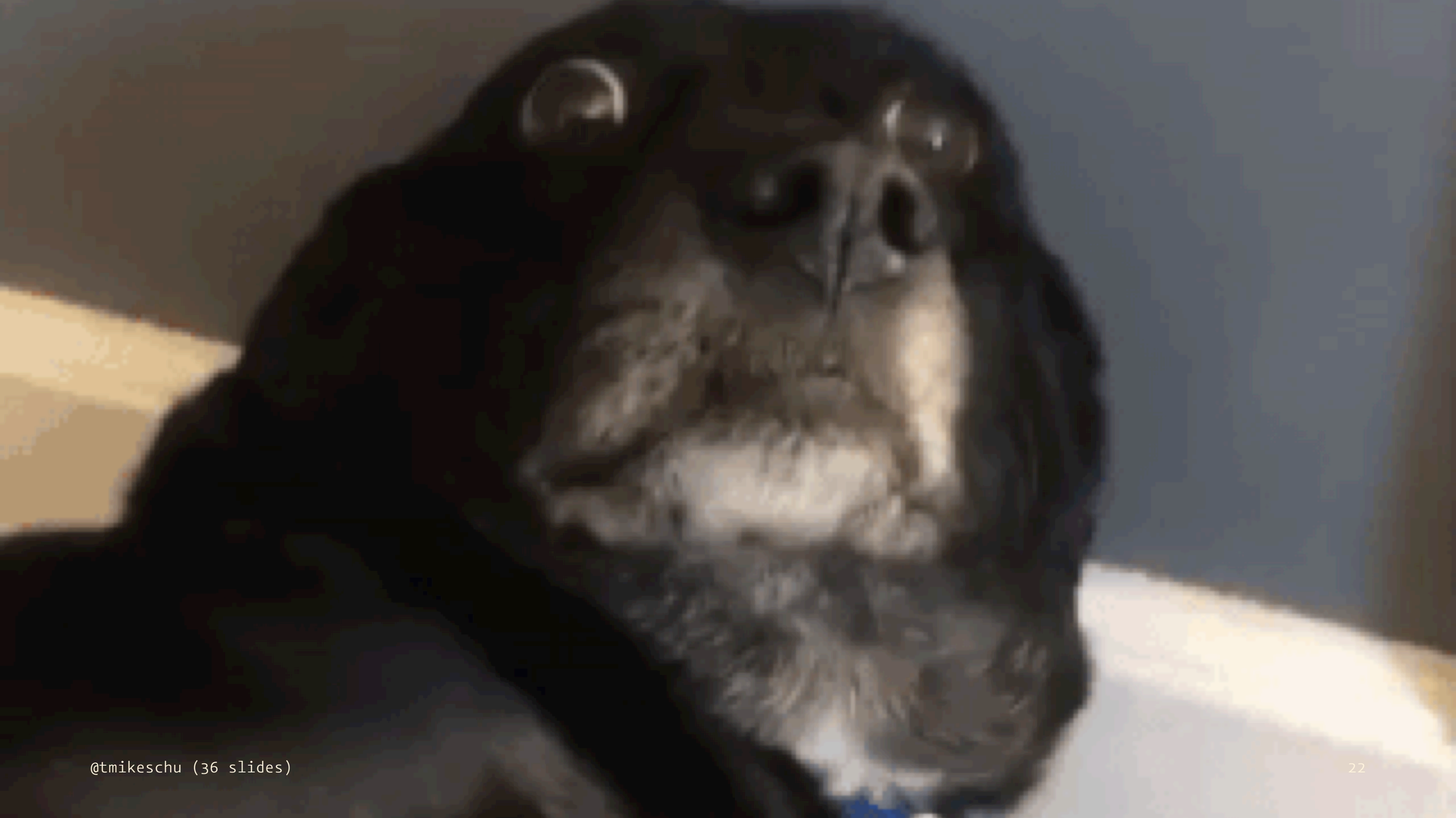
```elm
ages : List Int
ages = [10, 15, 21, 22]

firstAdult =
  find (\age -> age >= 18) ages
-- => Just 21

firstBaby =
  find (\age -> age < 3) ages
-- => Nothing
```

```elm
case firstBaby of
  Just age ->
    age

  Nothing ->
    0

-- OR

Maybe.withDefault 0 firstBaby
-- => 0

Maybe.withDefault 0 firstAdult
-- => 21
```

Overcomplicated?



Back to Ruby...



```
user.fetch(:please_dont_blow_up, "🙏")
```

Sending Messages

I’m sorry that I long ago coined the term "objects" for this topic because it gets many people to focus on the lesser idea. The big idea is messaging.

-- Alan Kay

```ruby
lost = [4, 8, 15, 16, [23, 42]]

lost[0] == lost.[](0) # => true

lost.[](0) == lost.send(:[], 0) # => true
```

```
100[0]
# => ?
```

```
100[0]
# => 0
```

Interact with [] and {} with a more consistent pattern.

— i.e., messages with explicit names

Safe Navigation Operators

A test might belong to a classroom
A classroom might belong to a teacher
A teacher might belong to a school

```
test&.classroom&.teacher
```

Meaningful Nothingness

```
coercers = (NilClass.instance_methods - Object.methods).
  select { |method| method.to_s.match(/^to_/) }.
  concat([:to_s])
# => [:to_a, :to_f, :to_i, :to_h, :to_r, :to_c]

coercers.map { |coercer| nil.send(coercer) }
# => [[], 0.0, 0, {}, (0/1), (0+0i), ""]
```

```ruby
# => [[], 0.0, 0, {}, (0/1), (0+0i), ""]

0.round => 0 # => 0

"".upcase # => ""

[].map { |x| x + x } # => []

{}.keys # => []

10 + 0 # => 10

"Harry Potter" + "" # => "Harry Potter"

[1, 2, 3] + [] # => [1, 2, 3]

{ a: 1 }.merge({}) # => { a: 1 }
```

What about other classes?

```
class Animal
  def self.find(id)
    # fetch animal from from data store
  end

  # ...
end

class MissingAnimal
  def name
    "missing animal"
  end

  # ... any other public methods on the Animal class
end

class GuaranteedAnimal
  def self.find(id)
    Animal.find(id) || MissingAnimal.new
  end
end
```

Review

- fetch: defaults, errors
- dig
- Maybe
- Message passing
- Type coercion
  - Null object pattern

What You Can Do Tomorrow

- Use fetch with defaults
  - params.fetch(:username, "")
- Use dig with trailing type coercion
  - params.dig(:user, :password).to_s

Thank you!

Resources

- Nothing is Something by Sandi Metz: https://www.youtube.com/watch?v=29MAL8pJImQ
- Elm Maybe: https://package.elm-lang.org/packages/elm/core/latest/Maybe
- Hash#fetch: https://apidock.com/ruby/Hash/fetch
- Hash#dig: https://apidock.com/ruby/Hash/dig
- Array#fetch: https://apidock.com/ruby/Array/fetch