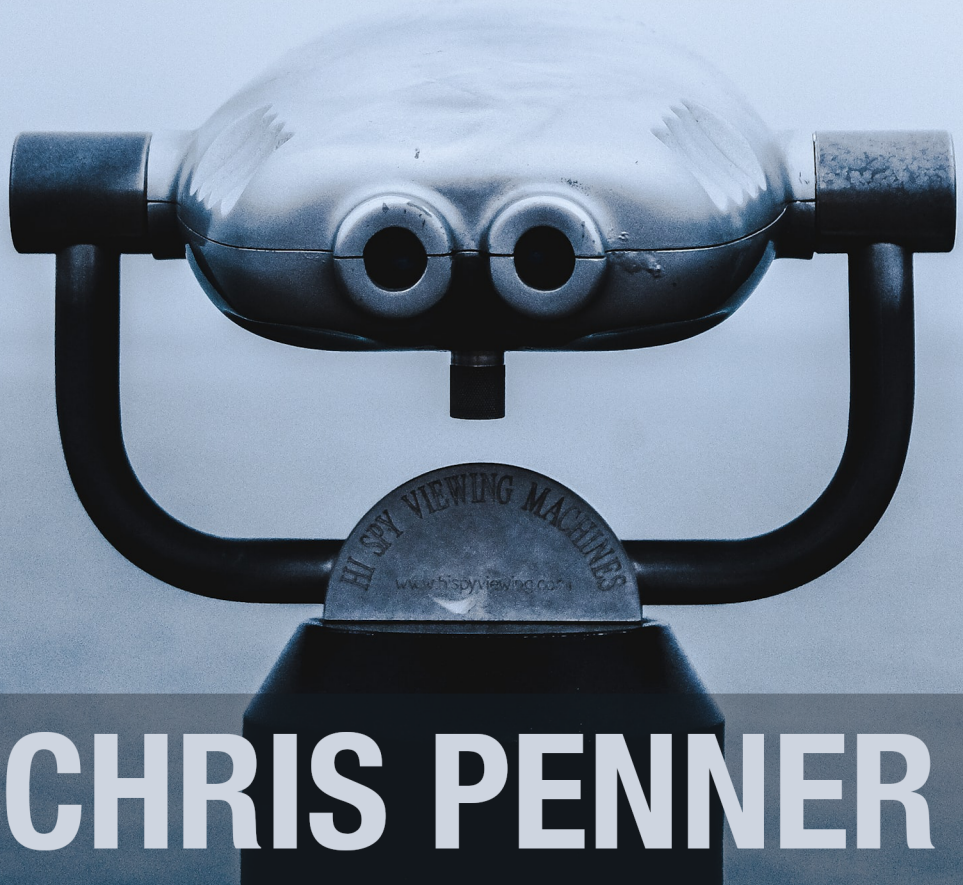
OPTICS
BY EXAMPLE
FUNCTIONAL LENSES IN HASKELL



CHRIS PENNER

Optics By Example

Functional lenses in Haskell

Chris Penner

This book is for sale at http://leanpub.com/optics-by-example

This version was published on 2020-04-22

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Tweet This Book!

Please help Chris Penner by spreading the word about this book on Twitter!

The suggested tweet for this book is:

Interested in lenses or optics? You seriously need to check out @OpticsByExample by @chrislpenner! #OpticsByExample

The suggested hashtag for this book is #OpticsByExample.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#OpticsByExample

Contents

1. Obligatory Preamble

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

1.1 Why should I read this book?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

1.2 How to read this book

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

1.3 Chosen language and optics encodings

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

1.4 Your practice environment

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

1.5 Following examples

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

1.6 About the type signatures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

1.7 About the exercises

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

2. Optics

2.1 What are optics?

Optics in its most general sense is a full field of study! In a slightly more concrete sense, **optics** are a **family** of **tools** which are interoperable with one another. **Lenses**, **Folds**, **Traversals**, **Prisms** and **Isos** are all **types of optics** which we'll explore throughout the book! This isn't a comprehensive list of all optics, in fact new types are still being discovered all the time!

You'll gain an intuition for what the more general concept of an **optic** actually *is* as you learn about each concrete type and begin to understand what they have in common, but to put it in a nutshell: optics are a family of inter-composable combinators for building bidirectional data transformations.

2.2 Strengths

So why do we actually care about these bidirectional transformation things? The short answer is that they solve a *lot* of very common data-manipulation problems in a **composable**, **performant**, and **concise** way, the long answer is what follows from here to the end of the book!

I'll take just a moment to expand on a few strengths:

Composition
: Each optic *focuses* on some subset of data, when composing with other optics they can pick up from where the previous optic left off and dive down even further. This means that each optic you learn becomes a member of your growing **vocabulary** of optics. Just as words in natural language can be strung together to form a sentence which communicates any intent you might want, a sufficiently complete vocabulary of optics can be arranged to effectively manipulate data to achieve your goals.

Separation of concerns
: Optics are the abstraction most programmers didn't know they needed. They allow us to cleanly separate concerns in stronger ways than *either* of Object-Oriented or Functional-Programming styles allow on their own. Optics allow us to specify which portions of data we wish to work with **separately** from the operations we wish to perform on them. We could, for example, encode a pre-order traversal of a tree structure, and combine it with a behaviour which prints the elements. We can swap out either of the **data-selector** or the **action** without affecting the other. Say goodbye to the Visitor Pattern!

Concision
: Although there are *many* other good reasons to love optics, they have the nice property of

being *very* succinct. **Most** tasks can be expressed in a single line of code, and in many cases the resulting code even reads like a simple sentence. For instance: `sumOf (key "transactions" . values . key "cost" . _Number)` will accept a JSON blob, will look into the "transactions" key, will then dive into each of the elements of the array, and in each of those objects will collect the "cost" of the transaction as a **number**, summing them all up into a total. In a typical imperative language this operation would likely take a few lines of code, or would at the very least require some ugly nested brackets. If we wish to instead take the **average** of these numbers we need only swap the `sumOf` action without fussing about with any variables and loops.

Enforcing interface boundaries

Optics can serve as an external interface which remains consistent despite changes to your data layer. They provide an abstraction layer similar to getters & setters which one might write in Java or Python. This allows you to alter your underlying data structures without breaking external consumers. You can even enforce data-consistency invariants when both getting and setting values! This replaces the idea of class-based "getters" and "setters" while also covering significant ground which used to require interfaces.

A principled and mature ecosystem

Optics have been around for long enough now that the ecosystem has ironed out most bugs and performance issues. There are a wide variety of libraries available, and many popular libraries provide optics-based interfaces (e.g. there are optics wrappers around JSON and XML libraries). Optics have a simple universal construction with the surprising benefit that writing an optics combinator **does not** require a dependency on any optics libraries! This allows us to use optics as a primitive building block or interface across libraries without worrying about large transitive or cyclic dependencies.

Hopefully all of this is sounding "too good to be true"! I assure you that optics can deliver on these promises. I can also guarantee that it'll take a little work and more than a few terrible, horrible, no good, very bad type errors to get there, but we're all in this together!

2.3 Weaknesses

Can't always have your cake and eat it too; here are a few areas where optics aren't perfect **yet**:

Type Errors

Most optics libraries (especially `lens`) can spew out some pretty ugly type errors when something goes wrong. This is one of the **most common** complaints, however it's a problem which is not easily solved. A great deal of type-level complexity is required to keep these libraries polymorphic and performant! We'll approach new types carefully and will address some common mistakes as well as talking about how to read these terrible beasts, so hopefully we can mitigate this one slightly.

Complex Implementation

Most (all?) optics implementations have their fair share of magic (see: complex category theory

and/or dirty hacks) going on behind the scenes. To make matters worse, most libraries are implemented in completely different ways! The Scala implementation is different from the Haskell implementation which is different from the Purescript implementation! They all follow a lot of the same theories and encode the same concepts, but a *perfect* backing implementation hasn't been discovered yet, though the profunctor encoding is looking pretty good so far. Luckily most libraries provide helpers which abstract over the underlying implementation so you won't typically need to worry about it.

Vast collection of combinators

This is one of those "weaknesses" you put on your CV that's actually a strength in disguise. There are a **LOT** of helpers and combinators provided in most optics libraries, it's overwhelming at first, but you'll learn how to search through them and better find the ones you need; and when you can do that effectively it means you'll usually be able to find a helper for performing almost any optics task! Just have a little patience, and finish reading this book of course!

2.4 Practical optics at a glance

I've talked at a high level about how optics help you perform actions over portions of data. Simple **actions** you can perform involve variants of **view**ing, **modify**ing or **traversing** the selected data.

Here are a few examples of varying difficulty and usefulness which represent a few things we'll see:

```
-- View nested fields of some record type
>>> view (address . country) person
"Canada"

-- Update portions of immutable data structures
>>> set _3 False ('a', 'b', 'c')
('a', 'b', False)

-- These selectors compose!
-- We can perform a task over deeply nested subsets of data.
-- Let's sum all numbers wrapped in a 'Left' within the right half of each tuple
>>> sumOf (folded . _2 . _Left)
      [(True, Left 10), (False, Right "pepperoni"), (True, Left 20)]
30

-- Truncate any stories longer than 10 characters, leaving shorter ones alone.
>>> let stories =
      ["This one time at band camp", "Nuff said.", "This is a short story"]
>>> over
      (traversed . filtered ((>10) . length))
      (\story -> take 10 story ++ "...")
```

```
      stories
["This one t...","Nuff said.","This is a ..."]
```

2.5 Impractical optics at a glance

Here are a few of the more arcane and *interesting* things optics can do. It's not important that you understand how these work or what they're doing, they're just here to help demonstrate the sheer **adaptability** of optics. Note how each operation is only one line of code! Hopefully they spark a bit of curiosity!

```
-- Summarize a list of numbers, subtracting the 'Left's, adding the 'Right's!
>>> import Numeric.Lens (negated)
>>> sumOf (folded . beside negated id) [Left 1, Right 10, Left 2, Right 20]
27

-- Capitalize each word in a sentence
>>> "why is a raven like a writing desk" & worded . _head %~ toUpper
"Why Is A Raven Like A Writing Desk"

-- Multiply every Integer by 100 no matter where they are in the structure:
>>> import Data.Data.Lens (biplate)
>>> (Just 3, Left ("hello", [13, 15, 17])) & biplate *~ 100
(Just 300,Left ("hello",[1300,1500,1700]))

-- Reverse the ordering of all even numbers in a sequence.
-- We leave the odd numbers alone!
>>> [1, 2, 3, 4, 5, 6, 7, 8] & partsOf (traversed . filtered even) %~ reverse
[1,8,3,6,5,4,7,2]

-- Sort all the characters in all strings, across word boundaries!
>>> import Data.List (sort)
>>> ("one", "two", "three") & partsOf (each . traversed) %~ sort
("eee","hno","orttw")

-- Flip the 2nd bit of each number
>>> import Data.Bits.Lens (bitAt)
>>> [1, 2, 3, 4] & traversed . bitAt 1 %~ not
[3,0,1,6]

-- Prompt the user with each question in a tuple,
-- then return the tuple with each prompt replaced with the user's input,
>>> let prompts = ( "What is your name?"
```

```
                  , "What is your quest?"
                  , "What is your favourite color?"
                  )
>>> prompts & each  %%~ (\prompt -> putStrLn prompt >> getLine)
What is your name?
> Sir Galahad
What is your quest?
> To seek the holy grail
What is your favourite color?
> Blue I think?
("Sir Galahad","To seek the holy grail","Blue I think?")
```

I hope that was a sufficiently strange list of examples to spark some wonder and creativity. These were meant to show the versatility, expressivity, and concision of optics! These examples are contrived and complex of course, but we'll see some more practical examples as we go on.

3. Lenses

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

3.1 Introduction to Lenses

We'll start our journey with lenses!

I mentioned in the **optics** section that optics allow us to separate concerns; i.e. split up the **action** we perform on data from the **selection** of data we want to perform it on. To be clear I'll refer to operations which can be performed on data as **actions**, whereas the data selectors are the actual **optics**. Each type of **optic** comes with a set of compatible **actions**.

Each type of optic has a different balance of **constraint** vs **flexibility**, moving to and fro on this spectrum results in several different but useful behaviours. Lenses lean closer to the **constrained** side of things, which means you have a lot of **guarantees** about their behaviour, but also means that you need to prove those guarantees to make a lens, so there are fewer **lenses** in the world than there are of the more flexible optics.

Lenses have the following concrete guarantees:

- A Lens **focuses** (i.e. **selects**) a **single** piece of data within a larger **structure**.
- A Lens must **never fail** to **get** or **modify** that focus.

These constraints unlock a few **actions** we can perform on lenses:

- We can use a lens to **view** the **focus** within a structure.
- We can use a lens to **set** the **focus** within a structure.
- We can use a lens to **modify** the **focus** within a structure.

Before we talk too much at a high level, let's take a look at a concrete usage of a lens and understand the different parts.

Anatomy

Here's a simple snippet which gets the `String` "hello" out from a couple nested tuples:

```
>>> view (_2 . _1) (42, ("hello", False))
"hello"
```

We won't worry about *exactly* what it's doing or how it works yet; for now we'll pick out and name individual pieces of this construction so that I can save myself some typing for the rest of the book.

Let's break it down into its anatomy:

```
      +-> The Action         +-> The Structure
      |                        |
    /-+-\           /----------+----------\
>>> view  (_2 . _1) (42, ("hello", False)))
          \---+---/       \--+--/
              |              |
              |              +-> The Focus
              +-> The Path
```

Note that the names for these things aren't really standardized yet, so you may have poor luck searching for them on Google; but if you start using them confidently around the water cooler I'm sure they'll catch on eventually.

The Action™
: An **action** executes some **operation** over the **focus** of a **path**. E.g. `view` is an action which **gets** the **focus** of a **path** from a **structure**. Actions are often written as an **infix operator**; e.g. `%~`, `^.` or even `<<%@=`!

The Path™
: The **path** indicates which data to **focus** and where to find it within the **structure**. A path can be a single optic, or several optics chained together through *composition*. If you consider dot-notation from most Object-Oriented languages you'll see similarities.

The Structure™
: The **structure** is the hunk of data that we want to work with. The **path** selects data from within the **structure**, and that data will be passed to the **action**.

The Focus™
: The smaller piece of the **structure** indicated by the **path**. The **focus** will be passed to the **action**. E.g. we may want to *get*, *set*, or *modify* the focus.

Exercises – Optic Anatomy

Jump to answers

For each of the following, identify the **action**, **path** and **structure**, don't worry about understanding how they actually work just yet. If you want a real challenge, try to identify the **focus** too! Note that certain optics allow **multiple focuses**, and some actions accept **parameters** other than the **focus**.

```
>>> view (_1 . _2)  ((1, 2), 3)
2

>>> set (_2 . _Left) "new" (False, Left "old")
(False, Left "new")

>>> over (taking 2 worded . traversed) toUpper "testing one two three"
"TESTING ONE two three"

>>> foldOf (both . each) (["super", "cali"],["fragilistic", "expialidocious"])
"supercalifragilisticexpialidocious"
```

3.2 Lens actions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Viewing through lenses

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Setting through a lens

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises - Lens Actions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

3.3 Lenses and records

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Lenses subsume the “accessor” pattern

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Building a lens for a record field

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises - Records Part One

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Getting and setting with a field lens

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Modifying fields with a lens

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Automatically generating field lenses

Writing our field accessors manually taught us about the relationship between **lenses**, **getters**, and **setters**, but writing them by hand is **mechanical**, **boring**, and **error-prone**! Did I hear someone yell boilerplate from the back!? There's only one correct way to get or set a record field, let's let the computer figure it out for us.

We can use Template Haskell to write our lenses for us! It's basically a macro system for generating Haskell code. To use it we'll need to enable the GHC extension by adding the following pragma to the top of our Haskell module:

```
{-# LANGUAGE TemplateHaskell #-}
```

With that enabled, we can add the appropriate Template Haskell expression right **after** our data declaration:

```
data Ship =
    Ship { _name    :: String
         , _numCrew :: Int
         }
    deriving (Show)

makeLenses ''Ship
```

The double-ticks `''` aren't a typo, they're how we pass an identifier name to the `makeLenses` macro!

This will generate the appropriate lens for each field in our `Ship` record type. By default `makeLenses` chooses names for the lenses by stripping the leading underscore _ from the field name. It'll generate the exact same lens we wrote by hand and will even have the same name! You'll need to delete, move, or rename your lens for the `numCrew` field if you still have that sitting around.

Note that it **won't** generate lenses for fields that aren't named with underscores, so don't forget to add it!

In general `makeLenses` does "The Right Thing"™, so I recommend taking advantage of it whenever you can. Make sure you do try writing a few lenses by hand though, it's a very good exercise when you're learning.

Template Haskell runs at compile-time, so the lenses produced by `makeLenses` won't ever show up in your source code. This can be a bit confusing at first, so if you're having trouble tracking down where a lens is coming from, it's very possible it's being generated by Template Haskell somewhere, make sure to take a look for a `makeLenses` call!

`makeLenses`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises - Records Part Two

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

3.4 Limitations

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Is it a Lens?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Is it a Lens? – Answers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

3.5 Lens Laws

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Why do optics have laws?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

The Laws

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

You get back what you set (set-get)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Setting back what you got doesn't do anything (get-set)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Setting twice is the same as setting once (set-set)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Case Study: `_1`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

You get back what you set (set-get)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Setting back what you got doesn't do anything (get-set)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Setting twice is the same as setting once (set-set)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Case Study: `msg`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

You get back what you set (set-get)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Case Study: `lensProduct`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

You get back what you set (set-get)

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises - Laws

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

3.6 Virtual Fields

I mentioned earlier how lenses subsume the Accessor Pattern from Object-Oriented programming. We've already seen how lenses take care of the **getters** and **setters** for record fields, this chapter covers how to represent "virtual fields" with lenses.

What's a virtual field

I'm using the term **virtual field** to mean any conceptual piece of data that doesn't exist as an actual field in your record definition. These are sometimes called "computed properties" or "managed attributes" in lanuages like Java, Python, etc. They're often used to present the data from concrete fields in a more convenient or enriched way. Sometimes they combine several concrete fields together, other times they're just used to avoid breaking changes when refactoring the structure of the record.

At the end of the day; they're really just normal lenses! Let's look at a few examples.

Writing a virtual field

For a simple example let's look at the following type:

```
data Temperature =
    Temperature { _location :: String
                , _celsius  :: Float
                }
    deriving (Show)
```

This generates the field lens:

```
celsius :: Lens' Temperature Float
```

Which we can use to get or set the temperature in celsius.

```
>>> let temp = Temperature "Berlin" 7.0
>>> view celsius temp
7.0

>>> set celsius 13.5 temp
Temperature {_location = "Berlin", _celsius = 13.5}

-- Bump the temperature up by 10 degrees Celsius
>>> over celsius (+10) temp
Temperature {_location = "Berlin", _celsius = 17.0}
```

But what about our American colleagues who'd prefer **Fahrenheit**? It'd be easy enough to write a function which convert **Celsius** to **Fahrenheit** and call that on the result, but you'd still need to **set** new temperatures using **Celsius**!

First we'll define our conversion functions back and forth, nothing too interesting there:

```
celsiusToFahrenheit :: Float -> Float
celsiusToFahrenheit c = (c * (9/5)) + 32

fahrenheitToCelsius :: Float -> Float
fahrenheitToCelsius f = (f - 32) * (5/9)
```

Here's how we *could* get and set using Fahrenheit:

```
>>> let temp = Temperature "Berlin" 7.0
>>> celsiusToFahrenheit . view celsius temp
44.6

>>> set celsius (fahrenheitToCelsius 56.3) temp
Temperature {_location = "Berlin", _celsius = 13.5}

-- Bump the temp by 18 degrees Fahrenheit
>>> over celsius (fahrenheitToCelsius . (+18) . celsiusToFahrenheit) temp
Temperature {_location = "Berlin", _celsius = 17.0}
```

The first two aren't **too** bad, but the `over` example is getting a bit clunky and error prone!

If we instead encode the **Fahrenheit** version of the temperature as a virtual field we gain better usability, cleaner code, and avoid a lot of possible mistakes.

Now for the fun part! We can write a `fahrenheit` lens **using** the existing `celsius` lens! We simply convert back and forth when getting and setting.

```
fahrenheit :: Lens' Temperature Float
fahrenheit = lens getter setter
  where
    getter = celsiusToFahrenheit . view celsius
    setter temp f = set celsius (fahrenheitToCelsius f) temp
```

Look how it cleans up the call site:

```
>>> let temp = Temperature "Berlin" 7.0
>>> view fahrenheit temp
44.6

>>> set fahrenheit 56.3 temp
Temperature {_location = "Berlin", _celsius = 13.5}

>>> over fahrenheit (+18) temp
Temperature {_location = "Berlin", _celsius = 17.0}
```

Much cleaner! Even though our `Temperature` record doesn't have a field for **Fahrenheit** we faked it using lenses to create a **virtual field**!

Breakage-free refactoring

Another great benefit of using lenses instead of field accessors for interacting with our data is that we gain more freedom when refactoring. To continue with the Temperature example, let's say as we've developed our wonderful weather app further we've discovered that Kelvin is a much better canonical representation for temperature data. We'd love to swap our `_celsius` field for a `_kelvin` field instead.

We'll consider two possible universes, in one this book was never written, so we didn't use lenses to access our fields. In the other (the one you're living in) we finished the book and decided to use lenses as our external interface instead.

The universe without lenses

In the sad universe without lenses we had the following code scattered throughout our app:

```
updateTempReading :: Temperature -> IO Temperature
updateTempReading temp = do
  newTempInCelsius <- readTemp
  return temp{_celsius=newTempInCelsius}
```

Then we refactored our `Temperature` object to the following:

```
data Temperature =
    Temperature { _location :: String
                , _kelvin  :: Float
                }
    deriving (Show)
```

And unfortunately every file that used record update syntax now fails to compile, because the `_celsius` field no longer exists. If we had instead used pattern matching, the situation would be even worse:

```
updateTempReading :: Temperature -> IO Temperature
updateTempReading (Temperature location _) = do
  newTempInCelsius <- readTemp
  return (Temperature location newTempInCelsius)
```

In this case the code will still compile, but we've completely switched units, this will behave completely incorrectly!

The glorius utopian lenses universe

Come with me now to the happy universe. In this universe we decided to use lenses as our interface for interacting with `Temperatures`, meaning we didn't expose the field accessors and thus disallowed fragile record-update syntax. We used the `celsius` lens to perform the update instead:

```
updateTempReading :: Temperature -> IO Temperature
updateTempReading temp = do
  newTempInCelsius <- readTemp
  return $ set celsius newTempInCelsius temp
```

Now when we refactor, we can simply export a replacement `celsius` lens in place of the old one:

```
data Temperature =
    Temperature { _location :: String
                , _kelvin  :: Float
                }
    deriving (Show)
makeLenses ''Temperature

celsius :: Lens' Temperature Float
celsius = lens getter setter
  where
    getter = (subtract 273.15) . view kelvin
    setter temp c = set kelvin (c + 273.15) temp
```

By adding the replacement lens we avoid breaking any external users of the type! Even our `fahrenheit` lens was defined in terms of `celsius`, so it will continue to work perfectly.

This is a simple example, but this idea works for more complex refactorings as well. When adopting this style it's important to avoid exporting the data type constructor or field accessors. Export a "smart constructor" function and the lenses for each field instead.

Exercises – Virtual Fields

Jump to answers

Consider this data type for the following exercises:

```
data User =
  User { _firstName :: String
       , _lastName :: String
       , _username :: String
       , _email :: String
       } deriving (Show)
makeLenses ''User
```

1. We've decided we're no longer going to have separate usernames and emails; now the email will be used in place of a username. Your task is to delete the `_username` field and write a replacement `username` lens which reads and writes from/to the `_email` field instead. The change should be unnoticed by those importing the module.
2. Write a lens for the user's `fullName`. It should append the first and last names when "getting". When "setting" treat everything till the first space as the first name, and everything following it as the last name.

It should behave something like this:

```
>>> let user = User "John" "Cena" "invisible@example.com"
>>> view fullName user
"John Cena"
>>> set fullName "Doctor of Thuganomics" user
User
    { _firstName = "Doctor"
    , _lastName = "of Thuganomics"
    , _email = "invisible@example.com"
    }
```

3.7 Data correction and maintaining invariants

We just learned about using lenses for computed and virtual fields; there's an extension to this idea where we can use lenses to perform certain types of data correction to ensure our data remains in a valid state. This is easiest explained with an example so we'll jump right in.

Including correction logic in lenses

Imagine we've got a rudimentary data type for storing clock time:

```
data Time =
    Time { _hours :: Int
         , _mins  :: Int
         }
    deriving (Show)
```

We want to allow users to edit the time of the clock, so we'll expose some lenses! However, we want to make sure that no matter what, our `hours` value remains between 0-23, and the minutes remain between 0-59. If the user tries to set the values outside of that range we'll simply clamp the value to fit the range instead. We can do this pretty easily by adding some simple logic to our **setter**s

```
clamp :: Int -> Int -> Int -> Int
clamp minVal maxVal a = min maxVal . max minVal $ a

hours :: Lens' Time Int
hours = lens getter setter
  where
    getter (Time h _) = h
    setter (Time _ m) newHours = Time (clamp 0 23 newHours) m

mins :: Lens' Time Int
mins = lens getter setter
  where
    getter (Time _ m) = m
    setter (Time h _) newMinutes = Time h (clamp 0 59 newMinutes)
```

These custom lenses clamp any new values we're setting to be within the expected range.

```
>>> let time = Time 3 10
>>> time
Time {_hours = 3, _mins = 10}

>>> set hours 40 time
Time {_hours = 23, _mins = 10}

>>> set mins (-10) time
Time {_hours = 3, _mins = 0}
```

This ensures that the values are within the expected ranges when setting! If you're you're a bit paranoid you could also clamp the getters.

Hopefully at some point during this section you thought "wait a minute, is this lawful"? The answer is **no, these are not lawful lenses**. If we `set` a bad value, we'll `get` the corrected value instead. This is usually fine, but it's good to think carefully about whether this behaviour is acceptable to you or not.

This isn't the only type of correction we could make in this scenario. If we wanted we could actually have the "minutes" and "hours" fields *roll over* when out of bounds. This makes it possible to do operations like adding 90 minutes to a time and still getting a sensible answer. Let's see how that would look:

```
hours :: Lens' Time Int
hours = lens getter setter
  where
    getter (Time h _) = h
    -- Take the hours 'mod' 24 so we always end up in the right range
    setter (Time _ m) newHours = Time (newHours `mod` 24) m

mins :: Lens' Time Int
mins = lens getter setter
  where
    getter (Time _ m) = m
    -- Minutes overflow into hours
    setter (Time h _) newMinutes
      = Time ((h + (newMinutes `div` 60)) `mod` 24) (newMinutes `mod` 60)
```

In this new configuration we can add or subtract minutes and hours from the clock time and the lens will automatically **normalize** the minutes and hours!

```
>>> let time = Time 3 10
>>> time
Time {_hours = 3, _mins = 10}

>>> over mins (+ 55) time
Time {_hours = 4, _mins = 5}

>>> over mins (subtract 20) time
Time {_hours = 2, _mins = 50}

>>> over mins (+1) (Time 23 59)
Time {_hours = 0, _mins = 0}
```

Nifty! Again; these lenses are **unlawful**, but still useful!

You're probably wondering whether there are ways to provide an error message on invalid input rather than silently correcting it; and indeed there are! We'll just need to learn a few more things before we're ready to take that on.

Exercises – Self-Correcting Lenses

Jump to answers

Consider the following:

```
data ProducePrices =
    ProducePrices { _limePrice  :: Float
                  , _lemonPrice :: Float
                  }
    deriving Show
```

1. We're handling a system for pricing our local grocery store's citrus produce! Our first job is to write lenses for setting the prices of limes and lemons. Write lenses for `limePrice` and `lemonPrice` which prevent **negative** prices by rounding up to `0` (we're okay with given produce out for free, but certainly aren't going to pay others to take it).
2. The owner has informed us that it's VERY important that the prices of limes and lemons must NEVER be further than 50 cents apart or the produce world would descend into total chaos. Update your lenses so that when setting lime-cost the lemon-cost is rounded to within 50 cents; (and vice versa).

It should behave something like this; don't worry if you can't get it exactly right, this one is tricky!

```
>>> let prices = ProducePrices 1.50 1.48
>>> set limePrice 2 prices
ProducePrices
    { _limePrice = 2.0
    , _lemonPrice = 1.5
    }
>>> set limePrice 1.8 prices
ProducePrices
    { _limePrice = 1.8
    , _lemonPrice = 1.48
    }
>>> set limePrice 1.63 prices
ProducePrices
    { _limePrice = 1.63
    , _lemonPrice = 1.48
    }
>>> set limePrice  (-1.00) prices
ProducePrices
    { _limePrice = 0.0
    , _lemonPrice = 0.5
    }
```

4. Polymorphic Optics

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

4.1 Introduction to polymorphic optics

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Simple vs Polymorphic optics

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

4.2 When do we need polymorphic lenses

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Type-changing focuses

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Changing type variables with polymorphic lenses

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Polymorphic Lenses

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

4.3 Composing Lenses

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

How do I update fields in deeply nested records?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Composing update functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Composing Lenses

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

How do Lens Types Compose?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Lens Composition

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

5. Operators

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

5.1 Lens Operators

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

5.2 `view` a.k.a. `^.`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

5.3 `set` a.k.a. `.~`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

5.4 Chaining many operations

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

5.5 Using `%~` a.k.a. `over`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

5.6 Learning Hieroglyphics

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

5.7 Modifiers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

5.8 When to use operators vs named actions?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

5.9 Exercises – Operators

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

6. Folds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

6.1 Introduction to Folds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Focusing all elements of a container

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Collapsing the Set

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Collecting focuses as a list

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Using lenses as folds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Composing folds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Foundational fold combinators

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Simple Folds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

6.2 Custom Folds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Mapping over folds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Combining multiple folds on the same structure

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Custom Folds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

6.3 Fold Actions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Writing queries with folds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Does my fold contain a given element?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Do ANY focuses match a predicate?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Do ALL focuses match a predicate?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Find the first element matching a predicate

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Does my fold have any elements or not?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

How many focuses are there?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

What's the sum or product of my focuses?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

What's the first or last focus?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Find the minimum or maximum focus

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Queries case study

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Folding with effects

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Combining fold results

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Using 'view' on folds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Customizing monoidal folds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Fold Actions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

6.4 Higher Order Folds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Taking, Dropping

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Backwards

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

TakingWhile, DroppingWhile

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Higher Order Folds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

6.5 Filtering folds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Filtered

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Filtering

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

6.6 Fold Laws

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

7. Traversals

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

7.1 Introduction to Traversals

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

How do Traversals fit into the hierarchy?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

A bit of Nostalgia

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

From fold to traversal

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

7.2 Traversal Combinators

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Traversing each element of a container

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

More Combinators

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Traversing multiple paths at once

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Focusing a specific traversal element

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

7.3 Traversal Composition

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Simple Traversals

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

7.4 Traversal Actions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

A Primer on Traversable

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Traverse on Traversals

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Infix traverseOf

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Using Traversals directly

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Traversal Actions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

7.5 Custom traversals

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Optics look like `traverse`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Our first custom traversal

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Traversals with custom logic

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Case Study: Transaction Traversal

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Custom Traversals

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

7.6 Traversal Laws

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Law One: Respect Purity

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Law Two: Consistent Focuses

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Good Traversal Bad Traversal

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Traversal Laws

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

7.7 Advanced manipulation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

partsOf

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Polymorphic partsOf

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

partsOf and other data structures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – partsOf

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

8. Indexable Structures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

8.1 What's an "indexable" structure?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

8.2 Accessing and updating values with 'Ixed'

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

The `Ixed` Class

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Accessing and setting values with `ix`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Indexed Structures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Indexing monomorphic types

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Indexing stranger structures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

8.3 Inserting & Deleting with ‘At’

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Map-like structures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Manipulating Sets

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Indexable Structures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

8.4 Custom Indexed Data Structures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Custom Ixed: Cyclical indexing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Custom At: Address indexing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Custom Indexed Structures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

8.5 Handling missing values

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Checking whether updates succeed

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Fallbacks with 'failing'

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Default elements

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Checking fold success/failure

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Missing Values

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

9. Prisms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

9.1 Introduction to Prisms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

How do Prisms fit into the hierarchy?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Simple Pattern-Matching Prisms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Checking pattern matches with prisms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Generating prisms with makePrisms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Embedding values with prisms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Other types of patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Prisms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

9.2 Writing Custom Prisms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Rebuilding `_Just` and `_Nothing`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Matching String Prefixes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Cracking the coding interview: Prisms style!

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Custom Prisms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

9.3 Laws

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Law One: Review-Preview

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Law Two: Prism Complement

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Law Three: Pass-through Reversion

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Prism Laws

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

9.4 Case Study: Simple Server

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Path prefix matching

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Altering sub-sets of functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Matching on HTTP Verb

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

10. Isos

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

10.1 Introduction to Isos

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

How do Isos fit into the hierarchy?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

There and back again

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

10.2 Building Isos

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

10.3 Flipping isos with `from`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

10.4 Modification under isomorphism

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

10.5 Varieties of isomorphisms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Composing isos

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Intro to Isos

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

10.6 Projecting Isos

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Projected Isos

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

10.7 Isos and newtypes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Coercing with isos

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Newtype wrapper isos

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

10.8 Laws

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

The one and only law: Reversibility

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Iso Laws

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

11. Indexed Optics

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

11.1 What are indexed optics?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

11.2 Index Composition

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Custom index composition

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Indexed Optics

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

11.3 Filtering by index

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Index Filters

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

11.4 Custom indexed optics

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Custom IndexedFolds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Custom IndexedTraversals

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Index helpers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Exercises – Custom Indexed Optics

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

11.5 Index-preserving optics

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

12. Dealing with Type Errors

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

12.1 Interpreting expanded optics types

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

12.2 Type Error Arena

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

First Foe: Level 1 Lenslion

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Level 2 Tuplicant

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Level 3 Settersiren

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Level 4 Composicore

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Level 5 Foldasaurus

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Level 6 Higher Order Beast

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Level 7 Traversacula

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

13. Optics and Monads

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

13.1 Reader Monad and View

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

13.2 State Monad Combinators

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

13.3 Magnify & Zoom

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

14. Classy Lenses

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

14.1 What are classy lenses and when do I need them?

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

No duplicate record fields

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Separating logic and minimizing global knowledge

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Granular dependencies with `makeFields`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Field requirements compose

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

14.2 `makeFields` vs `makeClassy`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

15. JSON

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

15.1 Introspecting JSON

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

15.2 Diving deeper into JSON structures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

15.3 Traversing into multiple JSON substructures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Traversing Arrays

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Traversing Objects

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

15.4 Filtering JSON Queries

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

15.5 Serializing & Deserializing within an optics path

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

15.6 Exercises: Kubernetes API

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

BONUS Questions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

16. Uniplate - Manipulating recursive data

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

16.1 A Brief History

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

16.2 Control.Lens.Plated

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Children

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Rewrite

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Universe

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Transform

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Deep

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

16.3 Overriding `plate`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

16.4 The magic of `biplate`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

16.5 Exercises – Uniplate

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

17. `generic-lens`

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

17.1 Generic Lenses

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Record Fields

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Positional Lenses

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Typed Lenses

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Altogether Now

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Subtype Lenses

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

17.2 Generic Traversals

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Types Traversal

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Parameter Traversals

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

17.3 Generic Prisms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Constructor Prisms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Typed Prisms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

17.4 Exercises – Generic Lens

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

18. Appendices

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

18.1 Optic Composition Table

This table is adapted from the documentation of the Scala optics library **Monocle**[1]. I've simply altered it to match the `lens` library.

The value of each cell denotes the most general type you can achieve by composing the column header with the row header.

The type of an optic is determined by collecting all the constraints of all composed optics in a path. Since constraints collection acts as a set union (which is commutative) the order of composition has no effect on the resulting optic type. Therefore the following table is symmetric across its diagonal.

"–" signifies that the optics are incompatible and do not compose.

	Fold	**Getter**	**Setter**	**Traversal**	**Prism**	**Lens**	**Iso**
Fold	Fold	Fold	–	Fold	Fold	Fold	Fold
Getter	Fold	Getter	–	Fold	Fold	Getter	Getter
Setter	–	–	Setter	Setter	Setter	Setter	Setter
Traversal	Fold	Fold	Setter	Traversal	Traversal	Traversal	Traversal
Prism	Fold	Fold	Setter	Traversal	Prism	Traversal	Prism
Lens	Fold	Getter	Setter	Traversal	Traversal	Lens	Lens
Iso	Fold	Getter	Setter	Traversal	Prism	Lens	Iso

For example, to determine which type we get by composing `traverse` with `_Just` we first check each of their types to discover that `traverse` is a `Traversal` and `_Just` is a `Prism`. We then look up the column (or row) with the header `Traversal`, then find the cell with the corresponding header `Prism` on the other axis. Performing this look up we see that the composition `traversed . _Just` results in a `Traversal`.

18.2 Optic Compatibility Chart

The following chart details which optics are valid substitutions for one another.

As an example, let's say we were curious if all **Prism**s are a valid **Traversal**; we first find the

[1] https://julien-truffaut.github.io/Monocle/optics.html

row with **Prism** in the first column; then find the corresponding **Traversal** column and find a `Yes`; meaning that a Prism **is** a valid substitution for a Traversal.

	Fold	Getter	Setter	Traversal	Lens	Review	Prism	Iso
Fold	Yes	No	No	No	No	No	No	No
Getter	Yes	Yes	No	No	No	No	No	No
Setter	No	No	Yes	No	No	No	No	No
Traversal	Yes	Yes	Yes	Yes	No	No	No	No
Lens	Yes	Yes	Yes	Yes	Yes	No	No	No
Review	No	No	No	No	No	Yes	No	No
Prism	Yes	No	Yes	Yes	No	Yes	Yes	No
Iso	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

18.3 Operator Cheat Sheet

Operators may look like an earthquake hit the mechanical keyboard factory, but there's actually a bit of a language to the whole thing which starts to make sense after a bit of practice.

Once you get used to the ideas you can usually *guess* the name of a symbol which does what you need, and it'll usually exist!

Legend for Getters

Symbol	Description
`^`	Denotes a **Getter**
`@`	Include the **index** with the result
`.`	Get a **single** value
`..`	Get a **List of** values
`?`	**Maybe** get the first value
`!`	**Force** a result or throw an exception if missing

Examples

```
(^@..) :: s -> IndexedFold i s a -> [(i, a)]
```

A **getter** (`^`) which **includes the index** (`@`) in a **list of all focuses** (`..`).

```
"Yarrr" ^@.. folded
[(0,'Y'),(1,'a'),(2,'r'),(3,'r'),(4,'r')]
```

```
(^?!) :: s -> Traversal' s a -> a
```

A **getter** (`^`) which **forcibly gets** (`!`) a **possibly missing** (`?`) value.

```
>>> Just "Nemo" ^?! _Just
"Nemo"
>>> Nothing ^?! _Just
*** Exception: (^?!): empty Fold
CallStack (from HasCallStack):
  error, called at src/Control/Lens/Fold.hs:1285:28 in lens
E:Control.Lens.Fold
  ^?!, called at <interactive>:1:1 in interactive:Ghci4
```

Legend for Setters/Modifiers

Symbol	Description
`.`	**Set** the focus
`%`	**Mod**ify the focus
~	Denotes a **Setter**/**Modifier**
`=`	Denotes a **Setter**/**Modifier** over a `MonadState` context
`<`	Include the **altered** focus with the result
`<<`	Include the **unaltered** focus with the result
`%%`	Perform a **traversal** over the focus
`<>`	`mappend` over the focus
`?`	Wrap in `Just` before setting
`+`	**Add** to the focus
`-`	**Subtract** from the focus
`*`	**Multiply** the focus
`//`	**Divide** the focus
`||`	Logically **`or`** the focus
`&&`	Logically **`and`** the focus
`@`	Pass the **index** to the modification function

Examples

`(<>~) :: Monoid a => Traversal' s a -> a -> s -> s`

A **setter** (~) which `mappends` (`<>`) a new value to the focus.

```
>>> ["Polly want a", "Salty as a soup"] & traverse <>~ " cracker!"
["Polly want a cracker!","Salty as a soup cracker!"]
```

`(<<%@=) :: MonadState s m => Traversal s s a b -> (i -> a -> b) -> m a`

Modify (`%`) the focus from within a `MonadState` (`=`), passing the **index** (`@`) to the function as well. Also return **unaltered** (`<<`) original value.

This one's a bit tricky:

```
>>> runState (itraversed <<%@= \k v -> "item: " <> k <> ", quantity: " <> v) (M.sing\
leton "bananas" "32")
("32",fromList [("bananas","item: bananas, quantity: 32")])
```

(%%~) :: Traversal s t a b -> (a -> f b) -> s -> f t

A **setter** (~) which **traverses** (%%) a function over the focus.

```
>>> (1, 2) & both %%~ (\n -> if even n then Just n else Nothing)
Nothing

>>> (2, 4) & both %%~ (\n -> if even n then Just n else Nothing)
Just (2,4)
```

18.4 Optic Ingredients

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19. Answers to Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.1 Optic Anatomy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.2 Lens Actions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.3 Records Part One

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.4 Records Part Two

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Laws

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.5 Virtual Fields

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.6 Self-Correcting Lenses

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.7 Polymorphic Lenses

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.8 Lens Composition

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.9 Operators

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.10 Simple Folds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.11 Writing Custom Folds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.12 Querying Using Folds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.13 Higher Order Folds

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.14 Filtering

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.15 Simple Traversals

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.16 Traversal Actions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.17 Custom Traversals

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.18 Traversal Laws

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.19 partsOf

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.20 Indexable Structures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.21 Custom Indexed Structures

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.22 Missing Values

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.23 Prisms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.24 Custom Prisms

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.25 Prism Laws

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.26 Intro to Isos

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.27 Projected Isos

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.28 Iso Laws

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.29 Indexed Optics

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.30 Index Filters

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.31 Custom Indexed Optics

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.32 Type Errors

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

First Foe: Level 1 Lenslion

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Level 2 Tuplicant

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Level 3 Settersiren

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Level 4 Composicore

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Level 5 Foldasaurus

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Level 6 Foldasaurus

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

Level 7 Traversacula

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.33 Kubernetes API

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

BONUS Questions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.34 Uniplate

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

19.35 Generic Lens

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

20. Thanks

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

20.1 Patreon Supporters

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.

20.2 Book Cover

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/optics-by-example.