Intro to the Haskell `lens` library

Who am I?

- Jason Stolaruk
- Android developer at Detroit Labs
- Haskell hacker on the side
- jason@detroitlabs.com
- @JasonStolaruk
- https://github.com/jasonstolaruk

First, some questions from me…

What are lenses?

- Simply put, lenses are functional getters and setters.
- They provide a compelling way to "poke around" inside data structures.
- You can drill down into lists, maps, and nested record data types.

Advantages of `lens`

- Concise and expressive.
- Flexible: may be used with a variety of data structures in a variety of ways.
- Lenses may be composed, allowing one to easily drill down into deeply nested data structures.
- Better than Haskell's unwieldy record update syntax.✨
- Other (they play nicely with the state monad…)

Disadvantages of `lens`

- The library is cumbersome (has many dependencies).
- The library has too many operators.
  https://hackage.haskell.org/package/lens-4.15.4/docs/Control-Lens-Operators.html
- The library implementation is difficult to understand.
  https://hackage.haskell.org/package/lens
- The types are often very complex - `Lens` has 4 type parameters!
  https://www.stackage.org/haddock/lts-9.2/lens-4.15.4/Control-Lens-Type.html#t:Lens
- Incomprehensible error messages when your types don't line up.
  For example: `(1, 2) & _1 <>~ "x"`
- Those who haven’t made the effort to learn `lens` won't be able to read your code.