- Lenses
  - compassable pair of **pure** getter and setters.
  - Focus on particular field inside the object
  - Obey a set of axioms known as **lens laws**
  - Think Object as the whole, field as the part
  - The getter takes the a whole and returns the part of the object the lens is focused on
    - *view = whole => part*
  - The setter takes a whole, and a value to set the part to, and returns a new whole with the part updated. Unlike a function which simply sets a value into an object's member field, Lens setters are pure functions:
    - set = whole => part => whole
  - Lens decouples state shape dependency from the rest of the system.
    - You need not know at what hierarchical depth the part of the object is present.
    - Lenses allow you to abstract state shape behind getters and setters. Instead of littering your codebase with code that dives deep into the shape of a particular object, import a lens.
    - If you later need to change the state shape, you can do so in the lens, and none of the code that depends on the lens will need to change.
  - **Lens Laws**
    - *view(lens, set(lens, value, store))* ≡ *value* — If you set a value into the store, and immediately view the value through the lens, you get the value that was set.
    - *set(lens, b, set(lens, a, store))* ≡ *set(lens, b, store)* — If you set a lens value to a and then immediately set the lens value to b, it's the same as if you'd just set the value to b.
    - *set(lens, view(lens, store), store)* ≡ *store* — If you get the lens value from the store, and then immediately set that value back into the store, the value is unchanged.