Use With Objects

Lenses focus on a piece of a data structure. They're composable and provide great ways to read/update the focused piece. (5 min. read)

Hopefully the last few exercises challenged you. Now let's sit back and soak in some more theory.

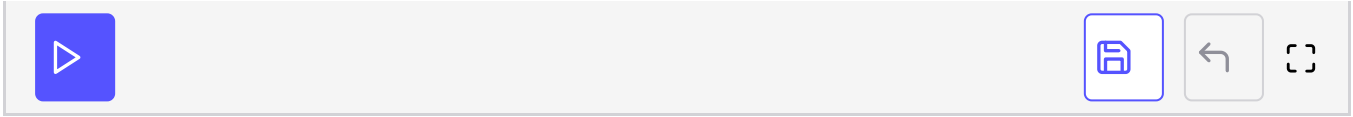
Lenses

Like the name implies, lenses let you "zoom in" on a particular piece of a data structure.

Getters

Let's use good ol' Bobo as an example.

```
import { lensProp, view } from 'ramda';

const person = {
  firstName: 'Bobo',
  lastName: 'Flakes'
};

const fNameLens = lensProp('firstName');
const result = view(fNameLens, person);

console.log({ result });
```

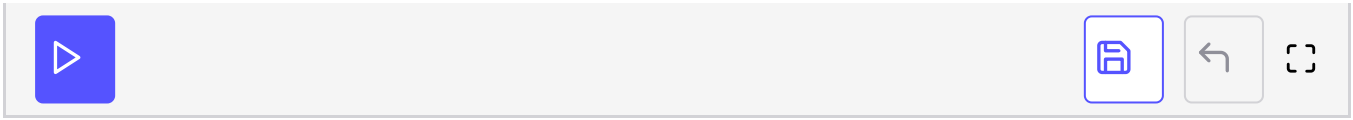


`lensProp` creates a lens focused on an object's property. In this case, `fNameLens` will find any object's `firstName` property. Passing it to `view` with our `person` returns Bobo's first name.

Setters

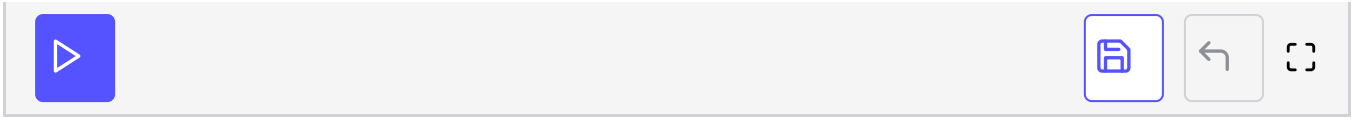
Changing Bobo's first name is just as easy. Use `set` with your desired change.

```
import { lensProp, set } from 'ramda';

const person = {
  firstName: 'Bobo',
  lastName: 'Flakes'
};

const fNameLens = lensProp('firstName');
const result = set(fNameLens, 'Bobo Jr.', person);

console.log({ person });
console.log({ result });
```



If you'd like to change it using a function, `over` can help.

```
import { concat, lensProp, over } from 'ramda';

const person = {
  firstName: 'Bobo',
  lastName: 'Flakes'
};

const fNameLens = lensProp('firstName');
const result = over(fNameLens, concat('Mr. '), person);

console.log({ person });
console.log({ result });
```



Note that `set` and `over` didn't mutate the original `person` object. Ramda functions don't mutate their inputs but instead return a new output.

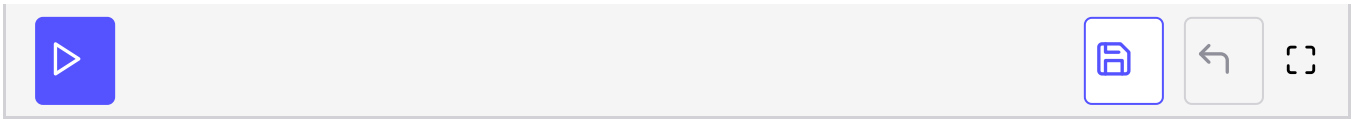
Nested Properties

Lenses are great for safely changing nested properties without a ton of merging code.

Using plain JS, how would you immutably change Bobo's manager's last name to "Flakes"? Probably something like this

```
const person = {
  firstName: 'Bobo',
  lastName: 'Flakes',
  company: 'Fake Inc.',
  position: {
    title: 'Front-End Developer'
```

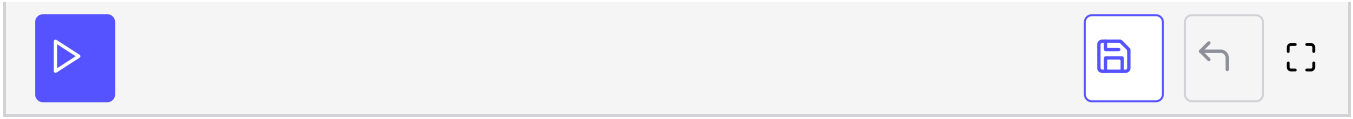
```
    department: {
      name: 'Product',
      departmentManager: {
        firstName: 'Bobo Sr.',
        lastName: 'Flax'
      }
    }
  }
};

const correctPerson = {
  ...person,
  position: {
    ...person.position,
    department: {
      ...person.position.department,
      departmentManager: {
        ...person.position.department.departmentManager,
        lastName: 'Flakes'
      }
    }
  }
};

const correctedLastName = correctPerson.position.department.departmentManager.lastName;

console.log({ correctedLastName });
```



Not too pretty, even with ES6 spread. Let's try lenses.

```
import { lensPath, set, view } from 'ramda';

const person = {
  firstName: 'Bobo',
  lastName: 'Flakes',
  company: 'Fake Inc.',
  position: {
    title: 'Front-End Developer',
    department: {
      name: 'Product',
      departmentManager: {
        firstName: 'Bobo Sr.',
        lastName: 'Flax'
      }
    }
  }
};

const managerLastNameLens = lensPath([
  'position',
  'department',
  'departmentManager',
  'lastName'
]);
```

```
const correctPerson = set(managerLastNameLens, 'Flakes', person);
const correctedLastName = view(managerLastNameLens, correctPerson);



console.log({ correctedLastName });
```



Much cleaner. One bonus is that since it's curried, a single lens can be used to view and update a property. We used `managerLastNameLens` in both `set` and `view`.