Longhand Lenses

Ramda's lensProp and lensIndex are convenience functions. Understanding the longhand version primes us to dive even deeper. (3 min. read)

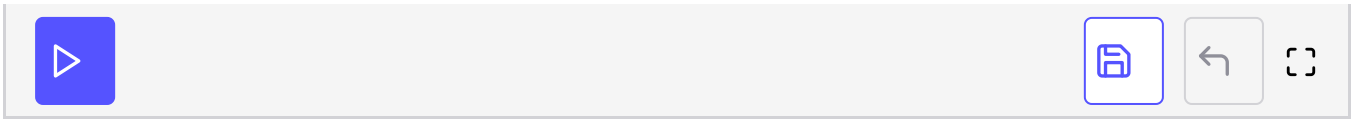
`lensProp` and `lensIndex` are actually built on top of the lower-level `lens` function.

This

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

const name = lensProp('name');
const result = view(name, { name: 'Bobo' });

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

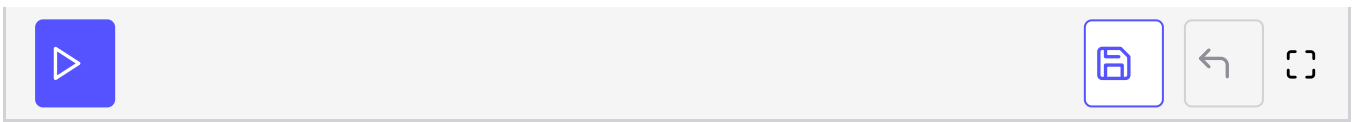


Is a convenient shorthand for this

```
import { assoc, lens, prop, view } from 'ramda';

const name = lens(prop('name'), assoc('name'));
const result = view(name, { name: 'Bobo' });

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



See the difference?

```
// shorthand
lensProp('name');

// longhand
lens(prop('name'), assoc('name'));
```

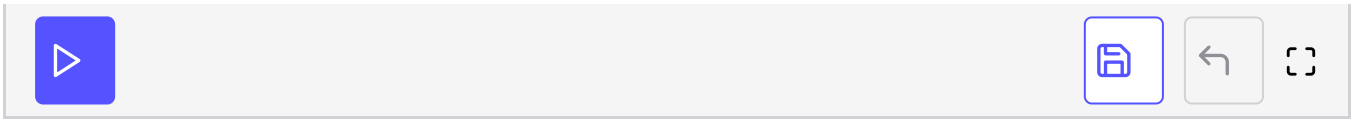
Why prop() and assoc()?

Ramda's `prop` function lets you **get** object properties.

```
import { prop } from 'ramda';

const person = { name: 'Bobo' };
const name = prop('name', person);

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

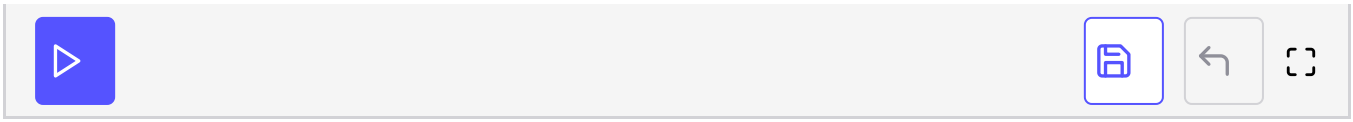


And `assoc` lets you **set** them without mutating the original object.

```
import { assoc } from 'ramda';

const person = { name: 'Bobo' };
const personWithAge = assoc('age', 25, person);

console.log({ person, personWithAge });
```



And since lenses need a **getter** and a **setter** to perform their duties, `prop` and `assoc` make the perfect combination to immutably handle and change lenses.

```
import { assoc, lens, prop, set, view } from 'ramda';

const data = { name: 'Bobo' };

const name = lens(prop('name'), assoc('name'));

const original = view(name, data);

const newObj = set(name, 'Another Name', data);

console.log({
  original,
  newObj
});
```