

# Fix the Flavors: Solution Review

Solution review.

We know from the last exercise that `lensPath` is the easiest way to focus on everyone's favorite flavor.

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

const favoriteFlavor = lensPath([
  'interests',
  'foods',
  'sweets',
  'iceCream',
  'favoriteFlavor'
]);
```

Then combining it with `map` and `view` will fetch them.

index.js

employees.json

```
import { lensPath, map, view } from 'ramda';
import employees from './employees.json';

const favoriteFlavor = lensPath([
  'interests',
  'foods',
  'sweets',
  'iceCream',
  'favoriteFlavor'
]);

const result = map(view(favoriteFlavor), employees);

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

But now there's work to do in between. Before `view`, we need a *setter* lens. Unfortunately `set` is only good for static values.

```
set(favoriteFlavor, 'Chocolate');
```

## Uppercasing Flavors

`over`, however, lets us set using functions! We can use Ramda's `toUpper` function to uppercase all the flavors.

index.js

employees.json





```
import { lensPath, map, over, toUpper, view } from 'ramda';
import employees from './employees.json';

const favoriteFlavor = lensPath([
  'interests',
  'foods',
  'sweets',
  'iceCream',
  'favoriteFlavor'
]);

const modifiedFlavors = map(over(favoriteFlavor, toUpper), employees);

const result = map(view(favoriteFlavor), modifiedFlavors);

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



Again, it's a *list* of employees so `map`'s necessary here. `over` focuses on an object using the `favoriteFlavor` lens, and uppercases its value.

That returns a new object with the modified flavors.

Passing that new array to `map(view)` returns the capitalized flavors.


## Appending to Flavors

But wait there's more! We must also append "IS A GREAT FLAVOR" to each of them.

We could combine that with `toUpper` using good ol' `pipe`.

index.js

employees.json



```
import { lensPath, map, pipe, over, toUpper, view } from 'ramda';
import employees from './employees.json';

const favoriteFlavor = lensPath([
  'interests',
  'foods',
  'sweets',
  'iceCream',
  'favoriteFlavor'
]);

const modifiedFlavors = map(over(favoriteFlavor, pipe(toUpper, (flavor) => `${flavor} IS A GREAT FLAVOR`)), employees);

const result = map(view(favoriteFlavor), modifiedFlavors);

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



This does the job, but not very eloquently. Look at that nesting!

```
const modifiedFlavors = map(over(favoriteFlavor, pipe(toUpper, (flavor) => `${flavor} IS A GREAT FLAVOR`)), employees);
```



Let's create a helper function, **emphasize**.

index.js

employees.json



```
import { lensPath, map, pipe, over, toUpper, view } from 'ramda';
import employees from './employees.json';

const favoriteFlavor = lensPath([
  'interests',
  'foods',
  'sweets',
  'iceCream',
  'favoriteFlavor'
]);

const emphasize = pipe(
  toUpper,
  (flavor) => `${flavor} IS A GREAT FLAVOR`
);

const modifiedFlavors = map(over(favoriteFlavor, emphasize), employees);

const result = map(view(favoriteFlavor), modifiedFlavors);

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



Good, let's go a step further.

Remember that lenses compose too. We don't need to separate them and call `map` twice!

index.js



employees.json

```
import { lensPath, map, pipe, over, toUpper, view } from 'ramda';
import employees from './employees.json';

const favoriteFlavor = lensPath([
  'interests',
  'foods',
  'sweets',
  'iceCream',
  'favoriteFlavor'
]);

const emphasize = pipe(
  toUpper,
  (flavor) => `${flavor} IS A GREAT FLAVOR`
);

const emphasizeFlavor = pipe(
  over(favoriteFlavor, emphasize),
  view(favoriteFlavor)
);

const emphasizeFlavors = map(emphasizeFlavor);

const result = emphasizeFlavors(employees);

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

