Get the Flavors: Solution Review

Solution review.

So a given employee's JSON is shaped like this

```
{
    "name": "Devante",
    "username": "Devante40",
    "avatar": "https://s3.amazonaws.com/uifaces/faces/twitter/adrienths/128.jpg",
    "email": "Devante40_Trantow11@yahoo.com",
    "dob": "1958-06-01T03:53:42.127Z",
    "phone": "544.986.5264",
    "address": {
      "street": "Louie Mission",
      "suite": "Suite 504",
      "city": "Lake Kyle",
      "zipcode": "60316",
      "geo": {
        "lat": "-65.8775",
        "lng": "-66.4677"
      }
    },
    "website": "delaney.info",
    "company": {
      "name": "Keeling Group",
      "catchPhrase": "Self-enabling grid-enabled architecture",
      "bs": "real-time orchestrate interfaces"
    },
    "interests": {
      "foods": {
        "sweets": {
          "iceCream": {
            "favoriteFlavor": "White Chocolate Raspberry Truffle"
          }
        }
      }
    }
  }
```

And their flavor's tucked away here

```
"interests": {
  "foods": {
    "sweets": {
      "iceCream": {
        "favoriteFlavor": "White Chocolate Raspberry Truffle"
        }
      }


    }
  }
```

That’s quite a few levels to go through, so Ramda’s `path` function is a good start.

index.js

data.json

```
import { path } from 'ramda';
import data from './data.json';

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

const result = getFlavor(data);

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

Our solution must use lenses, so this can easily become a `lensPath` and `view` combination.

index.js

data.json

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

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

const result = view(favoriteFlavor, data);

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

But this only works for one person! Using it on a list of people returns `undefined` because your using `view` on the array.

`undefined` because your using `view` on the array.

index.js

employees.json

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

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

const result = view(favoriteFlavor, employees);

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

How do we generalize a solution to a *list*?

Map

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 });
```

As we previously discussed, lenses use `map` themselves. When they receive the data, they begin unfolding and traverse the data structure.

Adding `map` on top of that just loops through `employees` and feeds each one to `view` as data.

Our result is an array of everyone’s favorite flavor!