Bang for Your Buck: Solution Review

Solution review.

Review

Let's review the steps

1. Get everything `maxPrice` or less
2. Sort by rating
3. Get the top 3

Here's a vanilla solution

index.js

menu.js

```
import menu from './menu';

const getTop3MealsFor = (maxPrice, menu) => menu
    .filter((item) => item.price <= maxPrice)
    .sort((a, b) => b.rating - a.rating)
    .slice(0, 3);

const result = getTop3MealsFor(12, menu);

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

Ramda has `slice` and `sort` functions. Using them lets us `pipe` the entire function to make it point-free.

index.js

menu.js

```
import { pipe, slice, sort } from 'ramda';
import menu from './menu';

const getTop3MealsFor = pipe(
  (maxPrice, menu) => menu.filter((item) => item.price <= maxPrice),
```

```
  sort((a, b) => b.rating - a.rating),
  slice(0, 3)
);

const result = getTop3MealsFor(12, menu);

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

We can point-free up `filter`'s predicate function using `propSatisfies`. It tests an object's property against a function to return true or false.

index.js

menu.js

```
import { gte, pipe, propSatisfies, slice, sort } from 'ramda';
import menu from './menu';

const getTop3MealsFor = pipe(
  (maxPrice, menu) => menu.filter(propSatisfies(gte(maxPrice), 'price')),
  sort((a, b) => b.rating - a.rating),
  slice(0, 3)
);

const result = getTop3MealsFor(12, menu);

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

I personally wouldn't do that, but it's fun to experiment.

Comparators

I *would*, however, play around with comparators. These are functions that compare data for sorting.

Ramda's `descend` function is a *descending comparator*, meaning a function that sorts entries from biggest to smallest. The flip side would be `R.ascend`, an *ascending comparator*.

We can break the sorting logic into a function

```
const byPrice = descend(prop('rating'));
```

Then use it like so

```
sort(byPrice);
```

It reads like a sentence!

index.js

menu.js

```
import { descend, gte, pipe, prop, propSatisfies, slice, sort } from 'ramda';
import menu from './menu';

const byPrice = descend(prop('rating'));
const getTop3MealsFor = pipe(
  (maxPrice, menu) => menu.filter(propSatisfies(gte(maxPrice), 'price')),
  sort(byPrice),
  slice(0, 3)
);

const result = getTop3MealsFor(12, menu);

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