

Total Cart Price: Solution Review

Solution review.

Getting Prices

Since point-free takes some getting used to, it can help to first write a plain solution and make it point-free later.

If you need a cart's total price and each item's shaped like this...

```
{
  name: 'apples',
  price: 2.49
}
```

...start by planning how you'd initially prepare the data. It's a collection so `map` can get the prices and `reduce` can add them up for us.

index.js

cart.js

```
import cart from './cart';

const getTotalPrice = (items) => items
  .map((item) => item.price)
  .reduce((acc, value) => acc + value, 0)

const result = getTotalPrice(cart);

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




A Touch of Formatting

We'd like a dollar figure, so 36.410000000000004 should be \$36.41.

index.js

cart.js



```
import cart from './cart';

const toUSD = (amount) => amount.toLocaleString('en-US', {
  style: 'currency',
  currency: 'USD',
});

const getTotalPrice = (items) => {
  const total = items
    .map((item) => item.price)
    .reduce((acc, value) => acc + value, 0);

  return toUSD(total);
};

const result = getTotalPrice(cart);

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



Refactoring to Point-Free

What were our steps?

1. Get each price
2. Add them up
3. Format as dollars (USD)

We know **pipe**'s great for multiple steps, so that'll definitely help us get a point-free solution. And **map** / **reduce** are provided by Ramda.

index.js



cart.js

```
import { map, pipe, reduce } from 'ramda';
import cart from './cart';

const toUSD = (amount) => amount.toLocaleString('en-US', {
  style: 'currency',
  currency: 'USD',
});

const getTotalPrice = pipe(
  map((item) => item.price),
  reduce((acc, value) => acc + value, 0),
  toUSD
);
```

```
const result = getTotalPrice(cart);

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



Even though the `map / reduce` functions aren't point-free, `getTotalPrice` is. This solution's perfectly fine so pat yourself on the back if you got it.

We can make `map / reduce` point-free though. Ramda's `prop` function grabs a given property of an object.

```
map(prop('price'))
```

And Ramda's `add` function can be put inside `reduce`.

```
reduce(add, 0);
```

index.js



cart.js

```
import { add, map, pipe, prop, reduce } from 'ramda';
import cart from './cart';

const toUSD = (amount) => amount.toLocaleString('en-US', {
  style: 'currency',
  currency: 'USD',
});

const getTotalPrice = pipe(
  map(prop('price')),
  reduce(add, 0),
  toUSD
);

const result = getTotalPrice(cart);

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



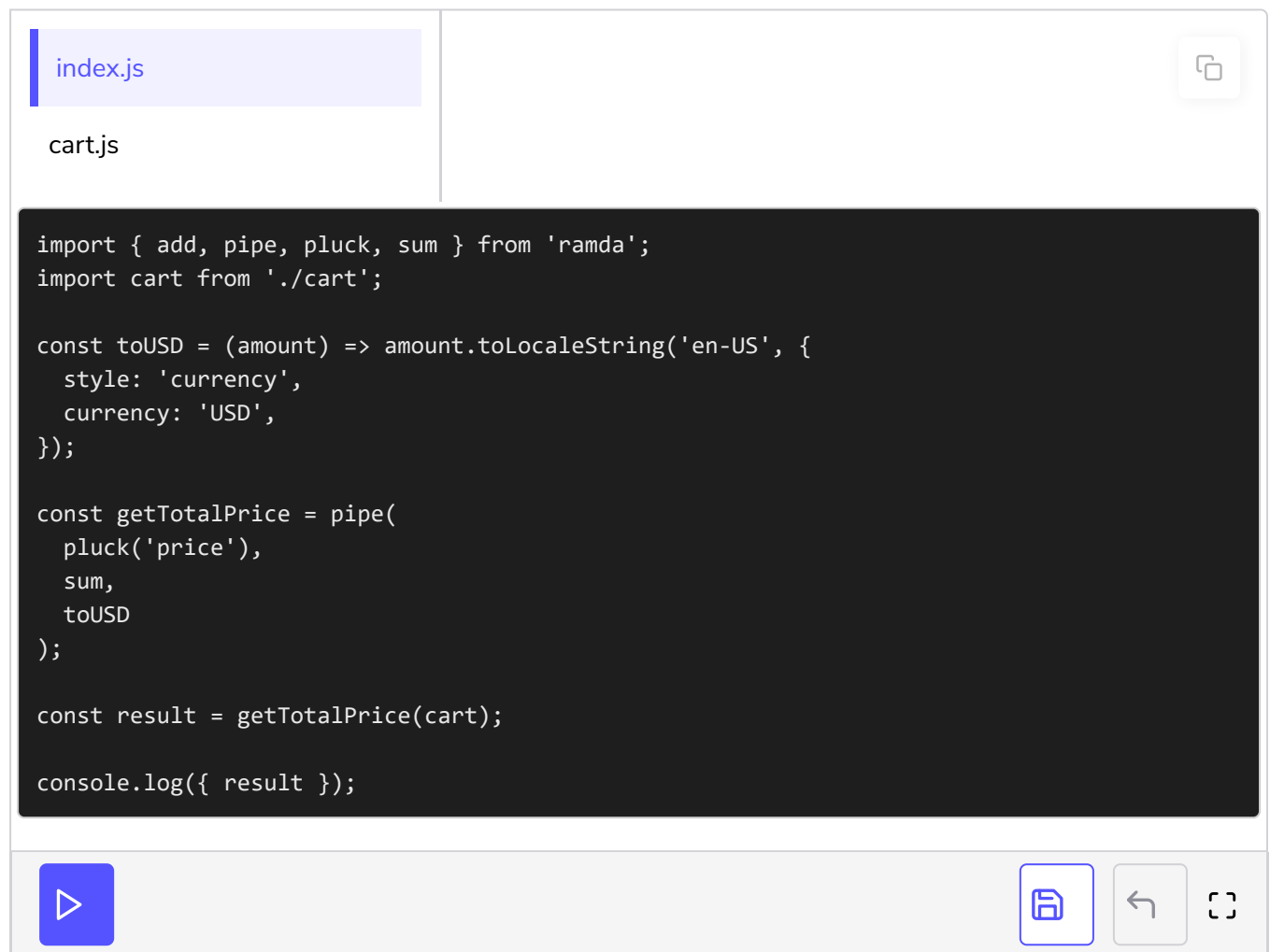
Wait There's More!

Plucking a property and summing a list are such common use cases that Ramda provides functions for them!

Check out `pluck` and `sum`.

<https://ramdajs.com/docs/#pluck>

<https://ramdajs.com/docs/#sum>



```
index.js
cart.js

import { add, pipe, pluck, sum } from 'ramda';
import cart from './cart';

const toUSD = (amount) => amount.toLocaleString('en-US', {
  style: 'currency',
  currency: 'USD',
});

const getTotalPrice = pipe(
  pluck('price'),
  sum,
  toUSD
);

const result = getTotalPrice(cart);

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

I personally liked this solution the most as it's so concise and expressive. If you got this one then double-pat yourself on the back.