Median Paycheck: Solution Review

Solution review.

I hope you didn't cheat and look at the HOFs section to find this snippet. :D

Either way, let's list the steps

1. Get salaries
2. Reject anything below $100,000
3. Get the median
4. Calculate monthly paycheck (amount / 12 months)
5. Format dollars (USD)

I think we're comfortable enough to start with a Ramda solution

index.js

employees.js

```
import { filter, map, median, pipe, prop } from 'ramda';
import employees from './employees';

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

const getMedianPaycheck = pipe(
  map(prop('salary')),
  filter((amount) => amount >= 100000),
  median,
  (amount) => amount / 12,
  toUSD
);

const result = getMedianPaycheck(employees);

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

Remember, `pluck('salary')` is equivalent to `map(prop('salary'))`.

index.js

employees.js

```
import { filter, median, pipe, pluck } from 'ramda';
import employees from './employees';

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

const getMedianPaycheck = pipe(
  pluck('salary'),
  filter((amount) => amount >= 100000),
  median,
  (amount) => amount / 12,
  toUSD
);

const result = getMedianPaycheck(employees);

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

And `R.lte` is great for filtering the salaries.

index.js

employees.js

```
import { filter, lte, median, pipe, pluck } from 'ramda';
import employees from './employees';

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

const getMedianPaycheck = pipe(
  pluck('salary'),
  filter(lte(100000)),
  median,
  (amount) => amount / 12,
  toUSD
);

const result = getMedianPaycheck(employees);

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

Ramda has a `divide` function, but it doesn't work as expected.

index.js

employees.js

```
import { divide, filter, lte, median, pipe, pluck } from 'ramda';
import employees from './employees';

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

const getMedianPaycheck = pipe(
  pluck('salary'),
  filter(lte(100000)),
  median,
  divide(12),
  toUSD
);

const result = getMedianPaycheck(employees);

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

$0.00?! That doesn't look right. Let's inspect with `tap`.

index.js

employees.js

```
import { divide, filter, lte, median, pipe, pluck, tap } from 'ramda';
import employees from './employees';

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

const getMedianPaycheck = pipe(
  pluck('salary'),
  filter(lte(100000)),
  median,
  tap((value) => {
    console.log('Before divide:', value);
  }),
  divide(12),
  tap((value) => {
    console.log('After divide:', value);
  }),
  toUSD
);

const result = getMedianPaycheck(employees);

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

Aha! We're dividing 12 by 608702.5 and getting a tiny decimal that rounds to $0.00! But we want to flip that division! Sounds like a job for Ramda's `flip` function.

index.js

employees.js

```
import { divide, filter, flip, lte, median, pipe, pluck, tap } from 'ramda';
import employees from './employees';

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

const flippedDivide = flip(divide);

const getMedianPaycheck = pipe(
  pluck('salary'),
  filter(lte(100000)),
  median,
  flippedDivide(12),
  toUSD
);

const result = getMedianPaycheck(employees);

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

Looks good to me! `flip` takes a function and returns a new one with the first two arguments reversed.

https://ramdajs.com/docs/#flip

Again, I wouldn't do this in the real world. The point's to expose you to Ramda's toolkit and let you decide what's best for your application.