Ramda vs. Lodash and Underscore

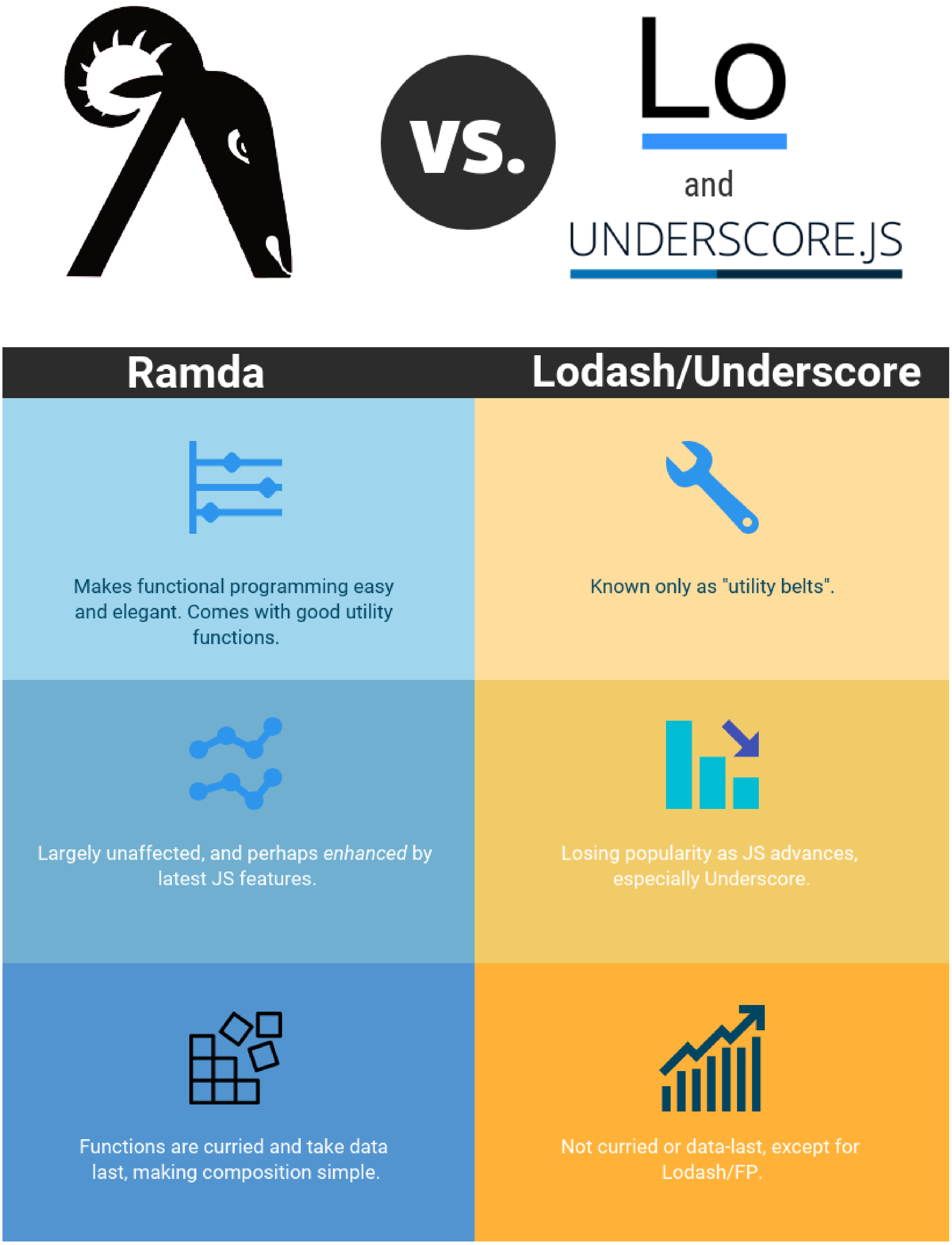
Lodash and Underscore are great utility libraries that began dying after ES6 went mainstream. Ramda wasn't just another utility, it was the precedent of practical FP in JavaScript. Underscore faded, but Lodash bounced back and released its own FP derivative. (3 min. read)

Lodash and Underscore are JavaScript's two best utility libraries. They're popular because we're more productive using them. At least, we *were*.

Before ES5 and ES6, they were perfect for transforming lists and merging objects, and even worked in Internet Explorer!

But today's tools let us write futuristic JavaScript everywhere, so their usefulness has diminished. Many of their common functions have vanilla alternatives here.

Ramda took a different path, coincidentally avoiding this threat of obsolescence.


vs.
Lo
and
UNDERSCORE.JS

Ramda	Lodash/Underscore
Makes functional programming easy and elegant. Comes with good utility functions.	Known only as "utility belts".
Largely unaffected, and perhaps *enhanced* by latest JS features.	Losing popularity as JS advances, especially Underscore.
Functions are curried and take data last, making composition simple.	Not curried or data-last, except for Lodash/FP.

Ramda vs. Lodash and Underscore Infographic

Being a utility belt simply isn't enough. `map`, `filter`, and `reduce` were among the best reasons to buy in, but they've been around since ES5. And we don't need help merging objects anymore either, as spread syntax is much easier.

This

```
import { merge } from 'lodash';

const obj1 = { name: 'Bobo' };
const obj2 = { shoeSize: 400 };

const merged = merge(obj1, obj2);
// { name: 'Bobo', 'shoeSize: 400 }
```

Has become this

```
const obj1 = { name: 'Bobo' };
const obj2 = { shoeSize: 400 };

const merged = { ...obj1, ...obj2 };
// { name: 'Bobo', shoeSize: 400 }
```

And the JS community won't hesitate to turn on you. Once ES6 went mainstream, repos and articles began sprouting, saying you might not need Lodash.

They took a big hit, especially Underscore, but Lodash adjusted pretty well.

How? What's your saving grace if the language by itself can replace you?

Support Functional Programming

Allow JavaScript developers to embrace FP and take full advantage of pure functions, higher-order functions, currying, and point-free syntax. Make this style a prime focus and watch swarms of users, old and new, put you back in their `package.json`s.

Which Ramda did, way before anyone else.

While it shares many utilities with Lodash/Underscore (`map`, `filter`, `reduce`, `merge`), Ramda never tried to be another toolkit. That saturated market was on the brink of bursting.

Instead, Ramda made FP as painless as possible for JavaScript developers–something no one had succeeded in doing at the time.

It borrows fundamental FP ideas, including

- Emphasis on pure functions
- Everything's curried
- Pipe/Compose are included and encouraged
- Functions take their data last

Which will become crystal clear throughout the course, and make a huge difference in how you express your data-manipulation logic.

Lodash FP eventually came out, which works like Ramda. So if you're interested in Lodash the patterns we'll cover easily transfer over.

Onwards!