Watch: 💭 **¿Vale la pena seguir usando lodash? Analizando performance**

Tienes el código de este vídeo en GitHub.

Si has buscado información sobre performance de vanilla JavaScript vs Lodash es probable que hayas visto artículos que apuntan que los métodos de Lodash son más rápidos. Pero son de hace unos cuantos años, ¿sigue siendo así a día de hoy?

Podemos ejecutar comparativas en webs como MeasureThat y veremos que los operadores nativos suelen ser más rápidos.

Más allá de comparar lodash vs JS nativo, también podemos usar estos servicios para comparar si nuestro código sería más performant de una forma u otra. También podemos usar `[performance.now](http://performance.now)` en nuestro código para comparar dos timestamps:

```
let t1, t2;
const data = Array(1000000).fill({ option: true });
const newArray = [];

t1 = performance.now();
for (const item of data) {
  if (item.option) {
    newArray.push(item);
  }
}
t2 = performance.now();
console.log(`for loop took ${t2 - t1} milliseconds`);

t1 = performance.now();
data.forEach(function (item) {
  if (item.option) {
    newArray.push(item);
  }
});
t2 = performance.now();

console.log(`forEach took ${t2 - t1} milliseconds`);
```

Aunque si estamos tratando con una cantidad enorme de datos esto puede ser útil, en general los problemas de performance en frontend vienen de otro lado: optimización de assets, falta de lazy loading... en el código generalmente nos será más útil optimizar por legibilidad que no por performance.

Enlaces relacionados

- Tests en MeasureThat:
  - Native find vs lodash find
  - ES6 equivalent to lodash _.mapValues
  - ES6 equivalent to lodash _.mapValues without console.log
- performance.now