

---

## Performance-Analysis

Comparing native JavaScript array methods map, reduce, filter, and find against for loop, forEach loop and lodash methods. The analysis uses basic operations and heavy data manipulation to analyze the execution speed of each method.

### To run

1. Run `npm install`
2. Generate the data for the tests by running `npm run seed`.
  - The default array is 10000 elements in length. You can create an array of a custom length by passing the desired size as an argument, like so `npm run seed 100000`.
3. For a small data set performance report run `npm run t:s`.
  - This runs the analysis on the first 5 elements of the array.
4. For a performance report on the whole array run `npm run t:l`

To test your own function create them in the formulas.js file.

## Results for small data set of array size 5 - 1000

For small data set	5-1000			
Array Size	5			
Function	Reduce	Map	Filter	Find
Methods	Time (ms)	Time (ms)	Time (ms)	Time (ms)
JS native	0.202	0.143	0.134	0.107
For loop	0.036	0.037	0.015	0.01
For each	0.556	0.105	0.102	0.032
Lodash	0.262	0.203	0.159	0.269
Array Size	100			
Function	Reduce	Map	Filter	Find
Methods	Time (ms)	Time (ms)	Time (ms)	Time (ms)
JS native	0.21	0.125	0.166	0.138
For loop	0.03	0.018	0.019	0.01
For each	0.186	0.064	0.077	0.032
Lodash	0.225	0.146	0.097	0.283
Array Size	500			
Function	Reduce	Map	Filter	Find
Methods	Time (ms)	Time (ms)	Time (ms)	Time (ms)
JS native	0.271	0.247	0.267	0.174
For loop	0.574	0.056	0.084	0.02
For each	0.246	0.186	0.269	0.18
Lodash	0.283	0.29	0.281	0.381
Array Size	1000			
Function	Reduce	Map	Filter	Find
Methods	Time (ms)	Time (ms)	Time (ms)	Time (ms)
JS native	0.941	0.486	0.387	0.174
For loop	0.074	0.128	0.152	0.103
For each	0.297	0.209	0.505	0.181
Lodash	0.392	0.205	0.286	0.587

---

## Results for mid data set of array size 3000 - 20000

For mid data set	3000-20000			
Array Size	3000			
Function	Reduce	Map	Filter	Find
Methods	Time (ms)	Time (ms)	Time (ms)	Time (ms)
JS native	0.434	1.478	0.522	0.383
For loop	0.508	0.167	7.956	0.112
For each	0.484	0.344	0.724	0.286
Lodash	0.699	0.427	0.604	0.576
Array Size	5000			
Function	Reduce	Map	Filter	Find
Methods	Time (ms)	Time (ms)	Time (ms)	Time (ms)
JS native	0.553	0.792	0.444	0.27
For loop	0.616	0.927	1.259	1.023
For each	0.69	0.277	0.403	0.31
Lodash	1.109	0.358	0.393	0.465
Array Size	10000			
Function	Reduce	Map	Filter	Find
Methods	Time (ms)	Time (ms)	Time (ms)	Time (ms)
JS native	1.298	1.917	0.527	0.368
For loop	1.527	1.225	1.615	0.83
For each	0.976	0.456	0.661	0.356
Lodash	1.758	0.494	0.456	0.933
Array Size	20000			
Function	Reduce	Map	Filter	Find
Methods	Time (ms)	Time (ms)	Time (ms)	Time (ms)
JS native	2.517	4.04	0.849	0.598
For loop	2.534	1.13	4.196	1.056
For each	1.888	0.844	0.905	0.816
Lodash	2.082	0.998	1.313	0.82

---

---

## Results for large data set of array size 50000 - 1000000

For large data se	50000-1000000			
Array Size	50000			
Function	Reduce	Map	Filter	Find
Methods	Time (ms)	Time (ms)	Time (ms)	Time (ms)
JS native	4.773	14.534	1.59	1.022
For loop	3.463	1.404	1.711	1.264
For each	3.192	1.668	1.641	2.205
Lodash	3.582	1.659	1.638	1.239
Array Size	100000			
Function	Reduce	Map	Filter	Find
Methods	Time (ms)	Time (ms)	Time (ms)	Time (ms)
JS native	7.597	29.347	3.207	1.902
For loop	7.511	2.809	2.034	1.357
For each	6.237	10.842	3.376	3.156
Lodash	6.694	4.415	2.649	1.273
Array Size	500000			
Function	Reduce	Map	Filter	Find
Methods	Time (ms)	Time (ms)	Time (ms)	Time (ms)
JS native	36.909	129.026	13.839	9.094
For loop	29.783	4.806	5.882	3.762
For each	31.79	12.79	42.116	11.521
Lodash	30.773	8.224	9.659	4.225
Array Size	1000000			
Function	Reduce	Map	Filter	Find
Methods	Time (ms)	Time (ms)	Time (ms)	Time (ms)
JS native	71.219	276.729	29.769	17.282
For loop	58.245	8.491	12.704	6.772
For each	64.462	28.129	31.082	23.71
Lodash	60.515	16.812	15.938	5.859

---

## Coming Soon

1. Ramda.js test
2. Caching (inline, warm) considerations
3. GC considerations

## Note

1. These results are computed using Node V8 v5.8.283.41
2. These result does not consider the JIT, inline caching, hidden classes, deoptimizations, garbage collection, pretenuring etc.
3. Result may vary as per env's.
4. Red colour highlight in the above images is just for reference, will soon change.

## Discussion/Posts

1. <https://news.ycombinator.com/item?id=17050798>
2. <https://medium.com/@ideepak.jsd/javascript-performance-test-for-vs-for-each-vs-map-reduce-filter-find-32c1113f19d7>