compass.uol

Modern Applications & Mobile Apps

compass.uol

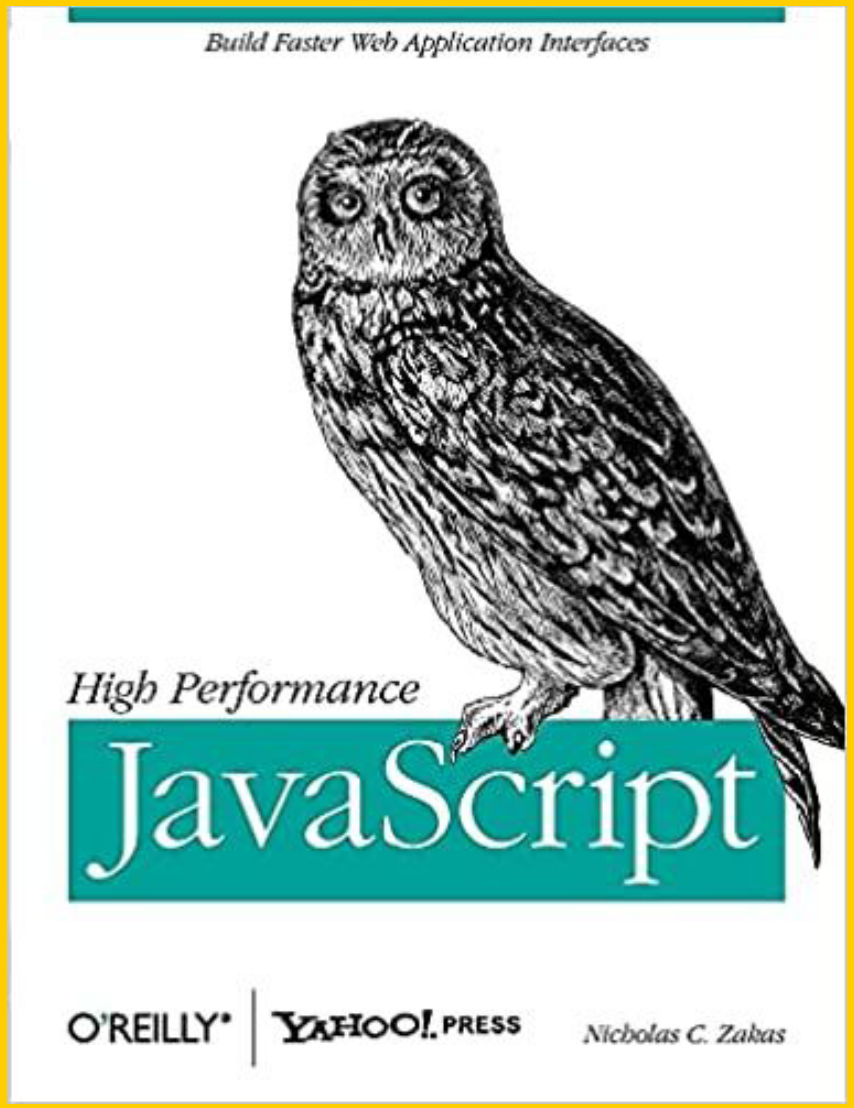
JavaScript de alto desempenho

compass.uol

JavaScript

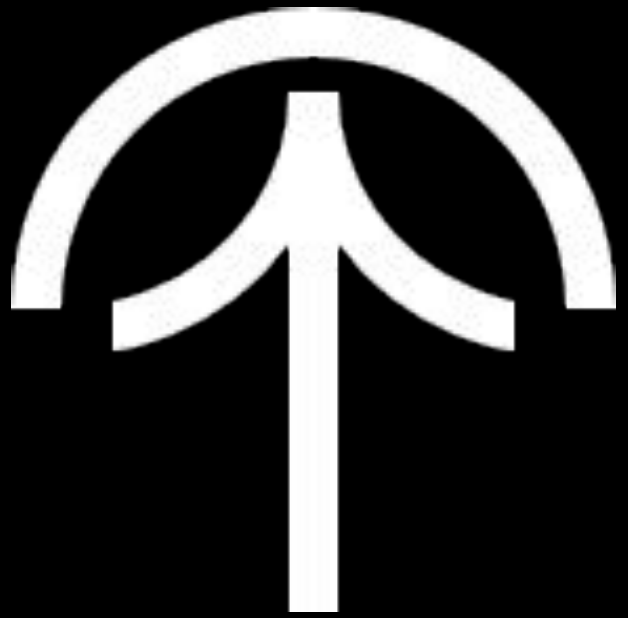
de Alto Desempenho

Com a liberdade que o código escrito em javascript dá ao desenvolvedor, muitas vezes acabamos ignorando diversos fatores especialmente quando estamos iniciando na programação, um deles e que pode vir a ser um gargalo no código a curto/médio e longo prazo é a Performance. Em geral, a Performance mede a eficácia de um sistema de software em relação às restrições de tempo e alocação de recursos. Um bom desempenho contribui para uma boa experiência do usuário. E uma boa experiência do usuário faz com que os usuários voltem, seja de uma API ou seja de uma aplicação que rode no browser.



compass.uol

Afinal para que serve teste de *Performance*?



compass.uol

Mesclando arrays sem causar carga no servidor

```
var array1 = [1, 2, 3];
var array2 = [4, 5, 6];
console.log(array1.concat(array2)); // [1,2,3,4,5,6];
```

```
var array1 = [1, 2, 3];
var array2 = [4, 5, 6];
console.log(array1.push.apply(array1, array2)); // [1,2,3,4,5,6];
```

compass.uol

Converta para número flutuante sem matar o desempenho

Usamos **math.floor** , **math.ceil** e **math.round** . Em vez de usá-los, **use “~~”**

```
Use

~~ (math.random*100)

Instead of

math.round(math.random*100)
```

compass.uol

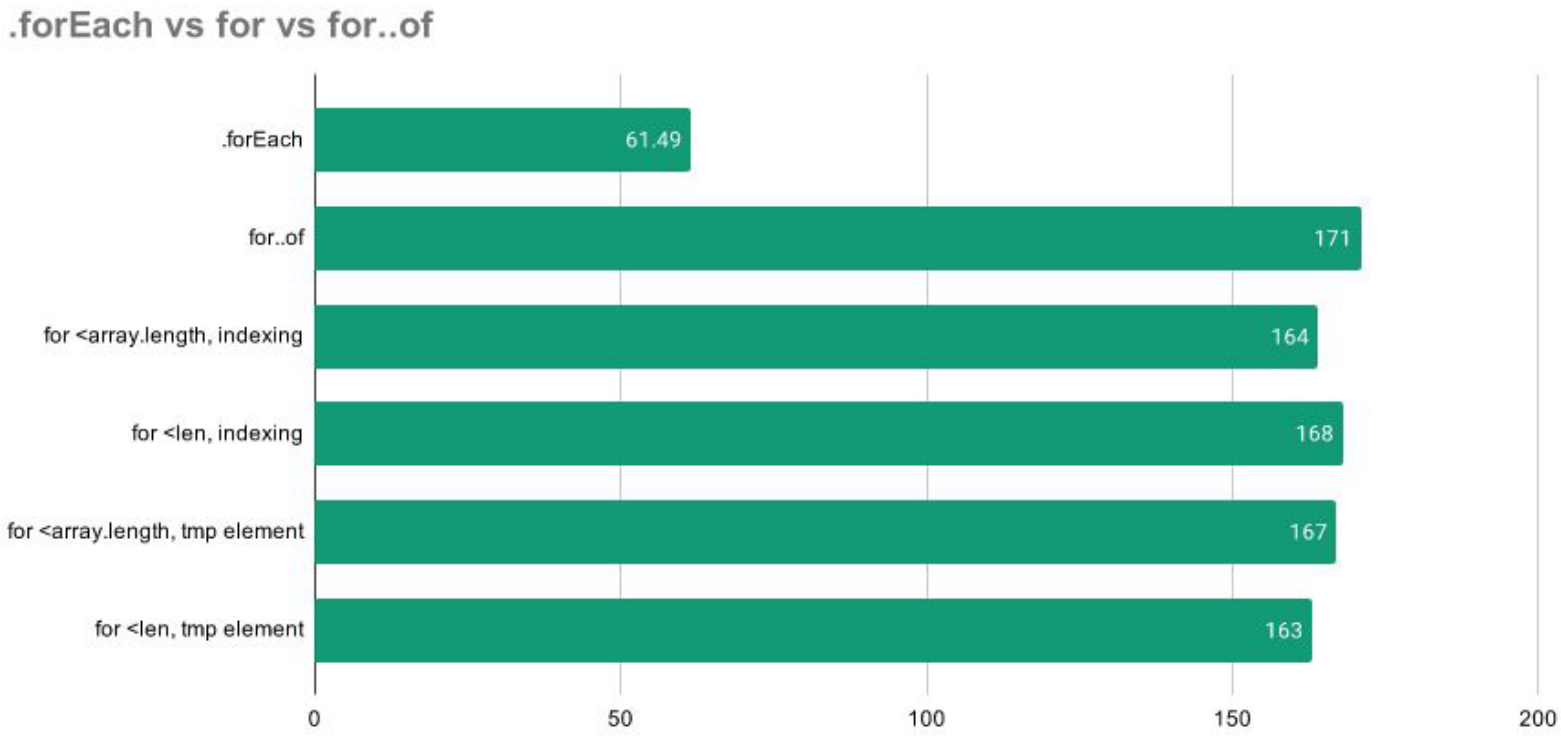
Desempenho de JavaScript .forEach, .map e .reduce vs for e for..of

```
function generateTestArray() {
  const result = [];
  for (let i = 0; i < 1000000; ++i) {
    result.push({
      a: i,
      b: i / 2,
      r: 0,
    });
  }
  return result;
}
```

compass.uol

→

Array.forEach vs for e for..of



Operações por segundo, quanto mais alto melhor

→

compass.uol

Array.forEach vs for e for..of

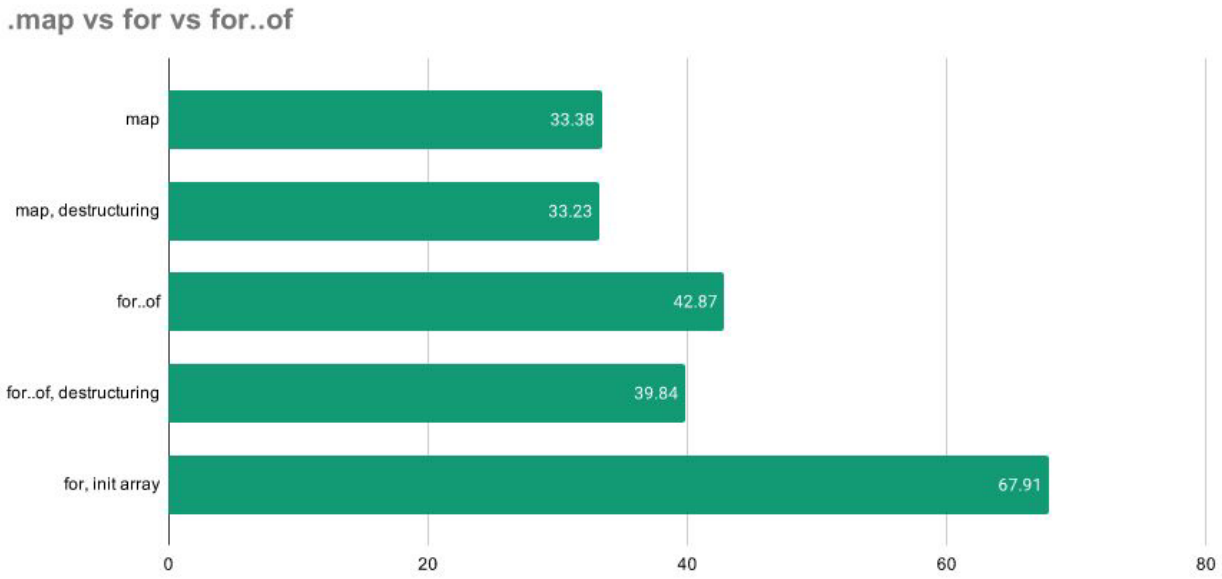
```
return new Benchmark.Suite("forEach")
  .add("Array.forEach", function () {
    array.forEach((x) => {
      x.r = x.a + x.b;
    });
  })
  .add("for of", function () {
    for (const obj of array) {
      obj.r = obj.a + obj.b;
    }
  })
  .add("for <array.length, indexing", function () {
    for (let i = 0; i < array.length; ++i) {
      array[i].r = array[i].a + array[i].b;
    }
  })
```

```
  .add("for <len, indexing", function () {
    const len = array.length;
    for (let i = 0; i < len; ++i) {
      array[i].r = array[i].a + array[i].b;
    }
  })
  .add("for <array.length, tmp element", function () {
    for (let i = 0; i < array.length; ++i) {
      const x = array[i];
      x.r = x.a + x.b;
    }
  })
  .add("for <len, tmp element", function () {
    const len = array.length;
    for (let i = 0; i < len; ++i) {
      const x = array[i];
      x.r = x.a + x.b;
    }
  });
```

compass.uol

→

Array.map vs for vs for..of



Operações por segundo, quanto mais alto melhor

→

compass.uol

→

Array.map vs for vs for..of

```
return array.map((x) => x.a + x.b);
```

```
const result = [];
for (const { a, b } of array) {
   result.push(a + b);
}
return result;
```

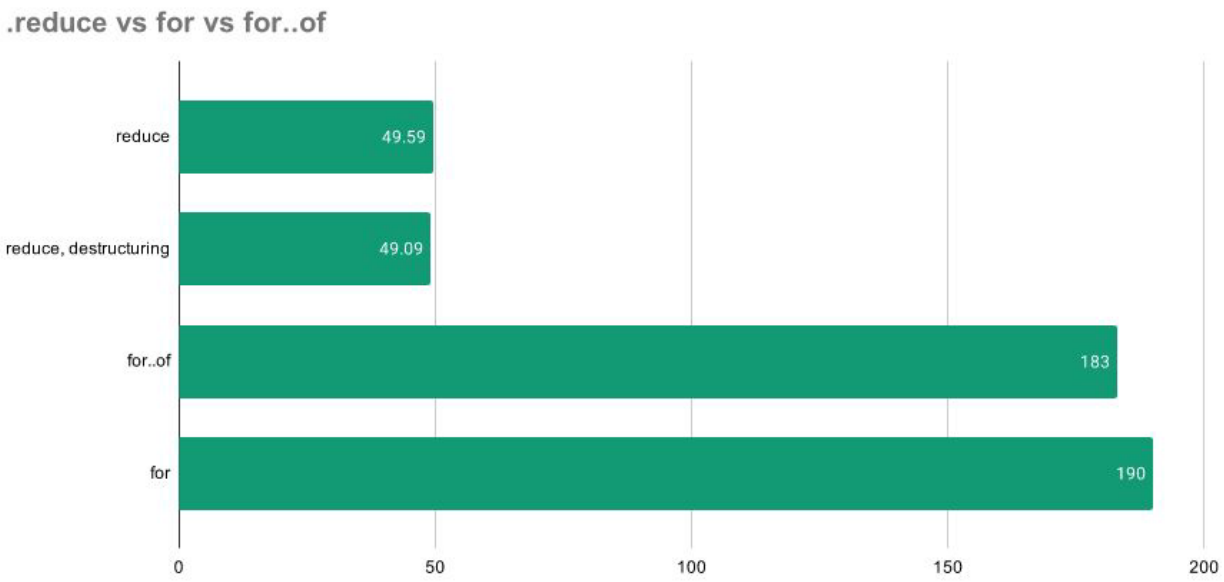
Operações por segundo, quanto mais alto melhor

→

compass.uol

→

Array.reduce vs for e for..of



Operações por segundo, quanto mais alto melhor

→

compass.uol

Array.map vs for vs for..of

```
.add("Array.reduce", function () {
  return array.reduce((p, x) => p + x.a + x.b, 0);
})
Complexity is 3 Everything is cool!
.add("Array.reduce, destructuring", function () {
  return array.reduce((p, { a, b }) => p + a + b, 0);
})
Complexity is 3 Everything is cool!
.add("for of", function () {
  let result = 0;
  for (const { a, b } of array) {
    result += a + b;
  }
  return result;
})
Complexity is 3 Everything is cool!
.add("for", function () {
  let result = 0;
  for (let i = 0; i < array.length; ++i) {
    result += array[i].a + array[i].b;
  }
  return result;
});
```

compass.uol

String() vs .toString() vs `${num}`

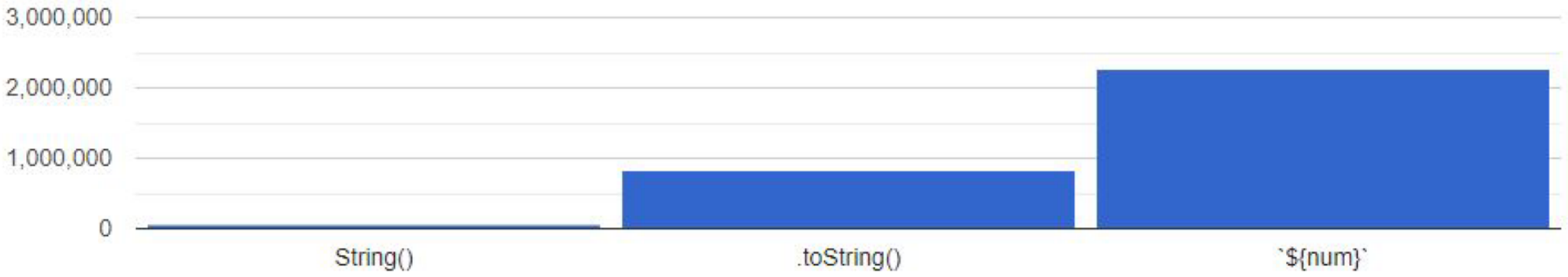
```
let num = 500;
let nums = [];
for(let i = 0; i < 100; ++i) {
  nums.push(String(num));
}
```

```
let num = 500;
let nums = [];
for(let i = 0; i < 100; ++i) {
  nums.push(num.toString());
}
```

```
let num = 500;
let nums = [];
for(let i = 0; i < 100; ++i) {
  nums.push(`${num}`);
}
```

compass.uol

String() vs .toString() vs `${num}`



Operações por segundo, quanto mais alto melhor

compass.uol

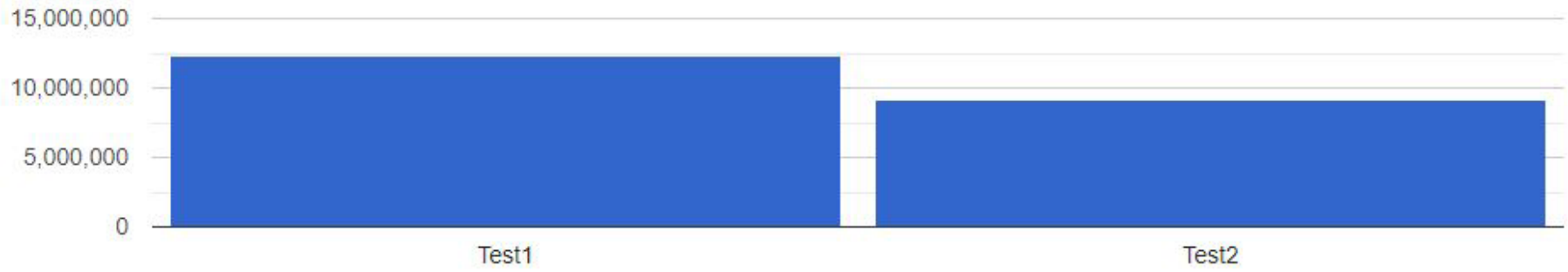
→

Boolean vs !!2

```var newArr = ['1', '2', '3', '4'];

!!newArr.length

newArr.length > 0```




Operações por segundo, quanto mais alto melhor

→
```

compass.uol

Reverse vs Unshift

```
1  var arr = Array.from({length: 100000}).map((_, i) => ({i}));
2
3  var arr2 = [];
4  for (let i = 0; i < arr.length; i++){
5    arr2.unshift({ ...arr[i]});
6  }
```

```
1  var arr = Array.from({length: 100000}).map((_, i) => ({i}));
2
3  var arr3 = arr.map(o=>({ ...o})).reverse();
```

```
$ node index.js
for: 1.145s
map: 3.876ms
```

compass.uol

O desempenho não é a única coisa que importa. A legibilidade do código geralmente é mais importante, portanto, o padrão é o estilo que se adapta ao seu aplicativo.

compass.uol

Bibliografia

http://rocha.la/JavaScript-bitwise-operators-in-practice

https://levelup.gitconnected.com/which-is-faster-for-for-of-foreach-loops-in-javascript-18dbd9ffbca9

https://github.com/dg92/Performance-Analysis-JS

https://www.measurethat.net/Benchmarks

https://www.linkedin.com/posts/erickwendel_top-5-dicas-para-resolver-problemas-de-lentid%C3%A3o-activity-6930144599441039360-j1wp/?utm_source=linkedin_share&utm_medium=member_desktop_web

https://www.loginradius.com/blog/engineering/16-javascript-hacks-for-optimization/

https://leanylabs.com/blog/js-forEach-map-reduce-vs-for-for_of/

compass.uol

Modern Applications & Mobile Apps

compass.uol

JavaScript de alto desempenho