ECMAScript 6

Rafael Escalfoni

adaptado de ECMAScript6 – Entre de cabeça no futuro do JavaScript

Iteração

- Iteradores:
  - Objeto que sabe como acessar, os itens de um iterável
  - Método: next()
  - Propriedades: `done` e `value`
- Iteráveis: define como o elemento será percorrido

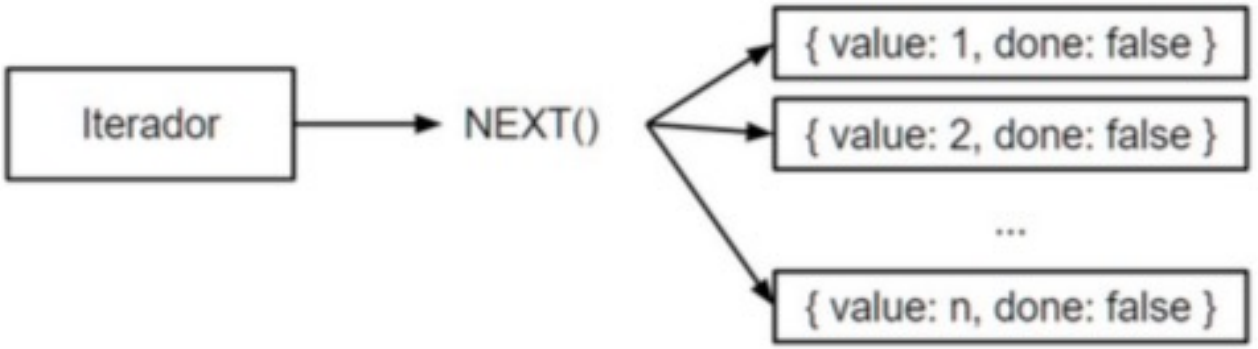


Figura 4.1: Relação entre iterador e iterável

Iteráveis

- Um objeto é definido como iterável se ele define seu comportamento de iteração.
  - Deve implementar o iterador na propriedade de chave `Symbol.iterator`
  - Tipos iteráveis:
    - Arrays
    - Strings
    - Maps
    - Sets

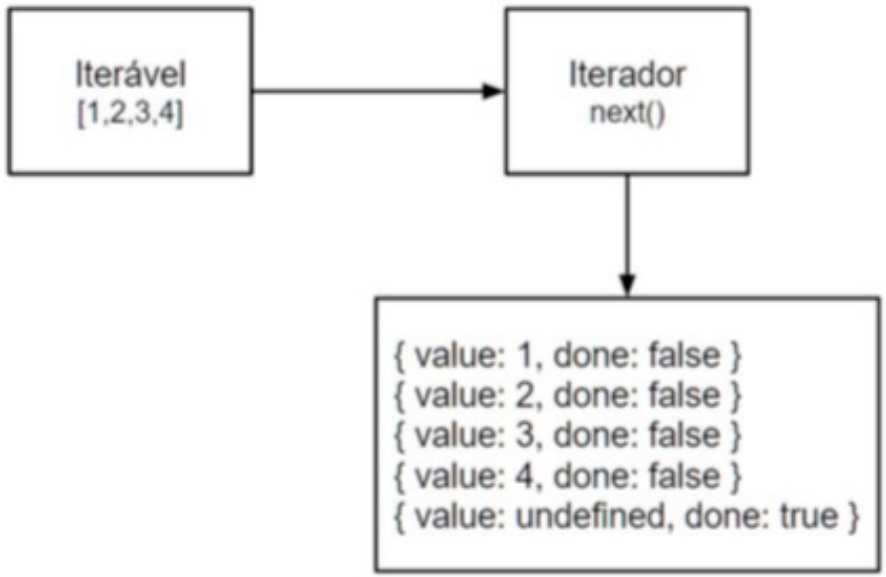
Exemplo

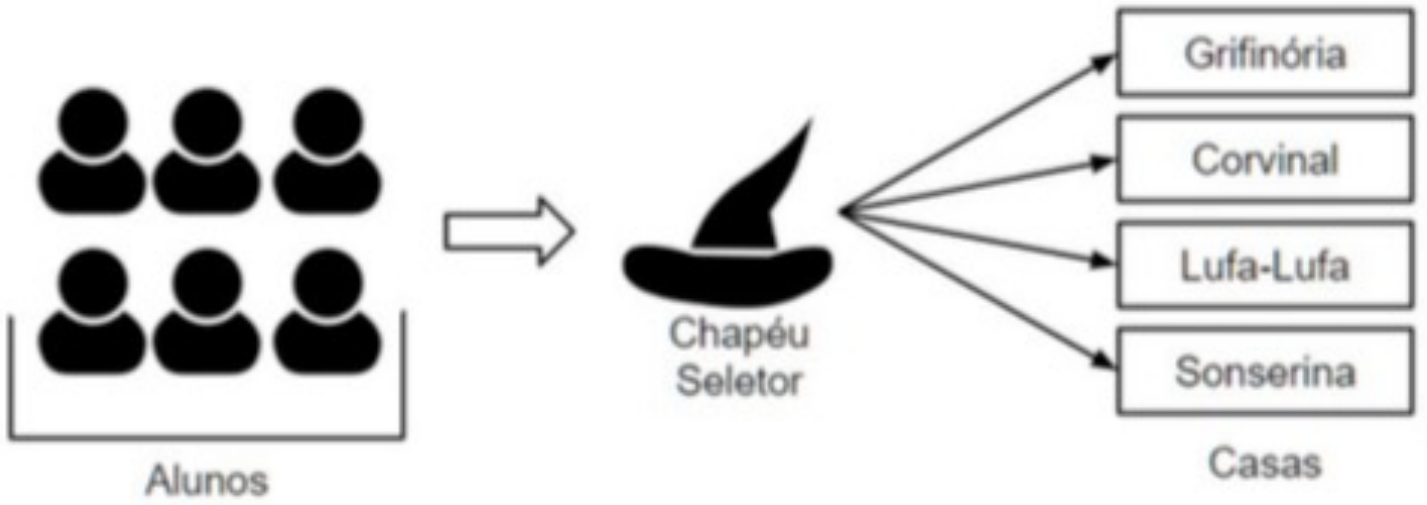
```
var bruxos = [‘Harry Potter', ‘Hermione Granger', ‘Rony Weasley’];
// obtém o iterador
var iteradorBruxos = bruxos[Symbol.iterator]();
iteradorBruxos.next(); // {value: ‘Harry Potter’, done: false}
iteradorBruxos.next(); // {value: ‘Hermione Granger’, done: false}
iteradorBruxos.next(); // {value: ‘Rony Weasley’, done: false}

iteradorBruxos.next(); // {value: undefined, done: true}
```

Mecânica da coisa

Um objeto iterável está ligado ao iterador, que define como o objeto será percorrido





Implementando o Chapéu Seletor de Hogwarts

- Para cada bruxo, o Chapéu Seletor deve fazer a seleção de sua casa de acordo com seus critérios misteriosos
  - ***Sempre haverá pelo menos um bruxo em cada escola...***

```
var iterador = bruxos[Symbol.iterator]();
var done = false;
var proximo = iterador.next();
do {
  let bruxo = proximo.value;
  chapeuSeletor.fazerSelecaoDaCasa(bruxo);
  proximo = iterador.next();
} while (!proximo.done);
```

Iterando com FOR... OF

```
for (variável of iteravel) {
// corpo
}
```

Exemplo:

```
var numeros = [1,2,3,4,5];
for (var num of numeros) {
  console.log(num);
}
```

Behind the scene

- O for...of acessa o iterador da estrutura a cada passo da iteração e carrega na variável
  for...of NÃO FUNCIONA EM OBJETOS NÃO ITERÁVEIS!!!

```
let perfilDoFacebook = {nome: “Carlos”
         , idade:22
         // ...outras propriedades
};
for (let dado of perfilDoFacebook) {
  console.log(dado);
}
```

```
ERROR!!!!
Uncaught TypeError: perfilDoFacebook is not iterable
```

Diferenças entre FOR...OF e FOR...IN

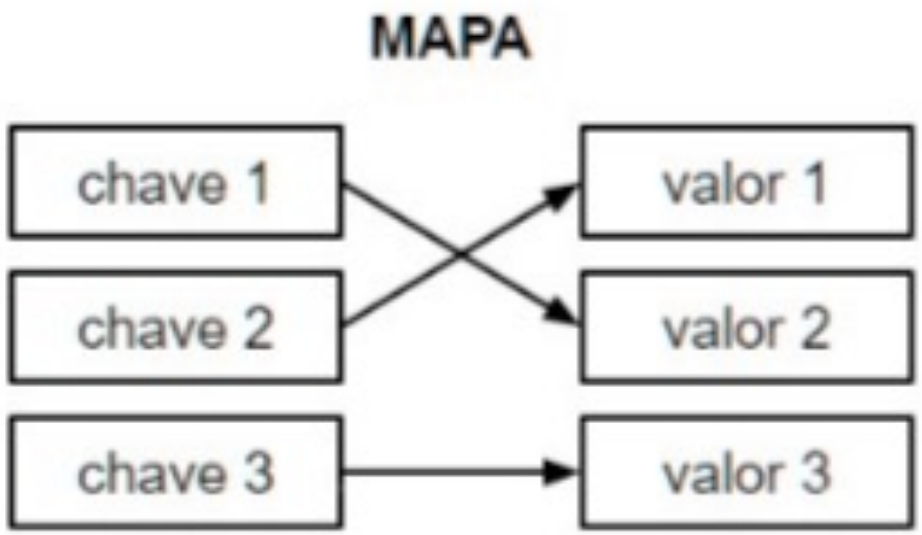
- FOR...OF – percorre apenas objetos iteráveis
- FOR...IN – percorre os atributos de um objeto

```
let perfilDoFacebook = {nome: “Carlos”
        , idade:22
        // ...outras propriedades
};
for (let dado in perfilDoFacebook) {
  console.log(dado);
}
```

```
Carlos
22
...
```

Voltando à escolha dos bruxos

```
for (var bruxo of bruxos) {
  chapeuSeletor.fazerSelecaoDaCasa(bruxo);
}
```



OBJETO

```
var obj = {
    chave1 : valor2,
    chave2 : valor1,
    chave3 : valor3
}
```

Estruturas MAP e WEAKMAP

Mapas são estruturas de dados com coleções de chave-valor, também chamados de dicionários

Novidades

- Adicionar elementos pelo par (chave- valor)
- Remover elementos pela chave
- Acessar elementos dada uma chave
- Pesquisar elementos, descobrindo se ele pertence ou não a coleção através da chave
- Indagar sobre atributos, por exemplo, número de elementos.

MAP

- Em um `Map` do JavaScript, qualquer valor (`objetos, funções ou valores primitivos`) pode ser usado como chave ou valor.
- Para definir esses valores, basta usar o método `set`:

```
var map = new Map();
function funcao(){};
var objeto = {};
map.set("string", "sou uma string");
map.set(objeto, "sou um objeto");
map.set(funcao, "sou uma função");

console.log(map.get("string")); // sou uma string
console.log(map.get(objeto)); // sou um objeto
console.log(map.get(funcao)); // sou uma função
```

```
console.log("tamanho: " + map.size); // tamanho: 3
console.log(map.has("string"); // true
console.log(map.has("abc"); // false

map.delete("string");
console.log(map.has("string"); // false

map.clear();
console.log("tamanho: " + map.size); // tamanho: 0
```

```
var mapa = new Map();
map.set("um", 1);
map.set("dois", 2);
map.set("tres", 3);

for (var chave of mapa.keys()) {
  console.log(chave); // um dois tres
}

for (var valor of mapa.values()) {
  console.log(valor); // 1  2  3
}

for (var entrada of mapa.entries()) {
  console.log(entrada);}
}
// saída:
// [ 'um', 1 ]
// [ 'dois', 2 ]
// [ 'tres', 3 ]
```

Mapas ou Objetos? Quando usar cada um?

- Maps são úteis SOMENTE para coleções
- Na dúvida, verifique:
  - As chaves são desconhecidas até o tempo de execução ou você precisa procurá-las dinamicamente?
  - Todos os valores sempre são do mesmo tipo?
  - Você precisa de chaves que não são strings?
  - Pares chave/valor são adicionados ou removidos frequentemente?
  - Você tem uma quantidade de pares chave/valor arbitrária?
  - A coleção é iterada?
    - SIM >> Map

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Map

Weakmap???

- Coleção de pares chave/valor na qual as chaves só podem ser objetos

```
var weakMap = new WeakMap();
var elemento1 = window;
var elemento2 = document.querySelector('body');

map.set(elemento1, "sou o elemento1");
map.set(elemento2, "sou o elemento2");

console.log(weakMap.get(elemento1));
console.log(weakMap.get(elemento2));

// saída:
// sou o elemento 1
// sou o elemento 2
```

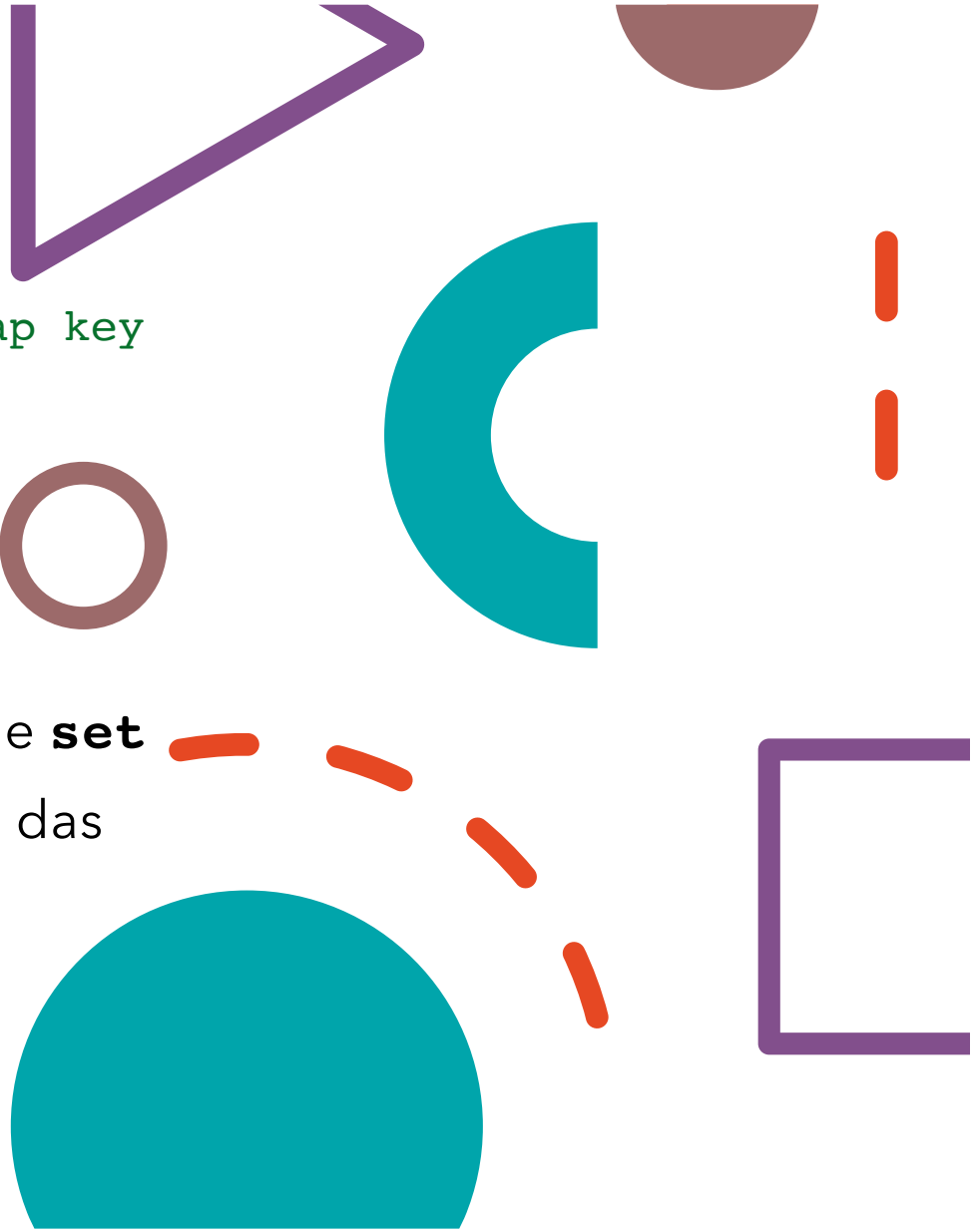
```
elemento1.parentNode.removeChild(elemento2);
elemento2 = null; // removendo referência local

console.log(weakMap.get(elemento2)); // undefined
```

```
var weakMap = new WeakMap();
function funcao(){};
var objeto = {};

// TypeError: Invalid value used as weak map key
weakMap.set("string", "isso é uma string");

weakMap.set(funcao, "isso é uma função");
weakMap.set(objeto, "isso é um objeto");
```



- Possui apenas os métodos **`delete`**, **`has`**, **`get`** e **`set`** (não tem o **`clear`**, nem o **`entries`** por conta das ligações fracas)

Quando usar WeakMap?

- Armazenando dados em objetos que podem ser destruídos com o passar do tempo.
  - O armazenamento por referência poupará memória

  - *Não haverá problemas de vazamento de memória (memory leak)*
  - *Poderemos manter dados privados dentro da aplicação, resguardando o máximo*

```
function Pessoa(nome){
  this._nome = nome;
};

Pessoa.prototype.getNome = function() {
  return this._nome;
};

var roberto = new Pessoa("Roberto");
console.log(roberto.getNome()); // Roberto
console.log(roberto._nome); // Roberto
```

```
var Pessoa = (function(){
  var dadosPrivados = new WeakMap();
  function Pessoa(nome){
    dadosPrivados.set(this, {nome: nome});
  };
  Pessoa.prototype.getNome = function(){
    return dadosPrivados.get(this).nome;
  }
  return Pessoa;
}());

var paulo = new Pessoa("Paulo");
console.log(paulo.getNome()); // Paulo
console.log(paulo._nome); // undefined
```