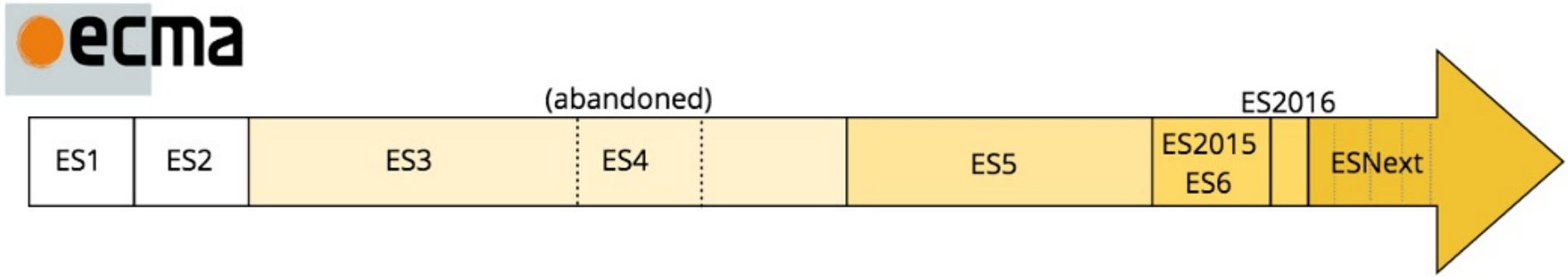
ECMAScript 6

Rafael Escalfoni

adaptado de ECMAScript6 – Entre de cabeça no futuro do JavaScript

ECMAScript vs JavaScript

- ECMAScript é a especificação da linguagem
- JavaScript foi o nome comercial dado a implementação


ecma
(abandoned)
ES2016
ES1
ES2
ES3
ES4
ES5
ES2015
ES6
ESNext


JS


1995
2000
2005
2010
2015
2020+


iOS

Como lidar com tantas implementações???

BABEL!!! https://babeljs.io/

- Tradutor de JavaScript

Arrays

Arrays em JavaScript

- São estruturas usadas para armazenar múltiplos valores em uma única variável

```
const cars = ["Volks", "Volvo", "BMW"];
```

- Um array é uma variável especial que pode guardar mais de um valor ao mesmo tempo

```
const car1 = "Volks";
const car2 = "Volvo";
const car3 = "BMW";
```

- ***E se tivéssemos 300 carros???***

Criando um Array

- Array literal:

```
const cars = [“Volks”, “Volvo”, “BMW”];
```

- Criando o Array e em seguida inserindo elementos:

```
const cars = [];
cars[0] = “Volks”;
cars[1] = “Volvo”;
cars[2] = “BMW”;
```

- Usando a palavra reservada **new**:

```
const cars = new Array(“Volks”, “Volvo”, “BMW”);
```

Acessando elementos

```
const cars = [“Volks”, “Volvo”, “BMW”];
let x = cars[0]; // x = “Volks”;
```

Alterando um elemento

```
const cars = [“Volks”, “Volvo”, “BMW”];
let cars[0] = “Gurgel”; // [“Gurgel”, “Volvo”, “BMW”]
```

Arrays são objetos!

- Arrays são um tipo especial de objetos. Usam números para acessar seus elementos e possuem um conjunto de métodos para iteração próprios.
- Possui Propriedades e Métodos:

```
cars.length; //propriedade que retorna o número de elementos
cars.sort(); //método que organiza o array
//acessando o último elemento
cars[cars.length - 1];
```

Adicionando elementos no Array

```
const fruits = [“Banana”, “Orange”, “Apple”];
fruits.push(“Lemon”); // adiciona o elemento Lemon ao array fruits

// ou:
const fruits = [“Banana”, “Orange”, “Apple”];
fruits[fruits.length] = “Lemon”;
```

Cuidado! Índices muito grandes podem criar buracos no array:

```
const fruits = [“Banana”, “Orange”, “Apple”];
fruits[6] = “Lemon”; // cria algumas posições undefined no array
```

CUIDADO!

- Se você usar strings nos índices, o array será convertido em um objeto simples e o método length para de funcionar.

```
const person = [];
person["firstName"] = "John";
person["lastName"] = "Doe";
person["age"] = 46;
person.length;     // retornará 0
person[0];         // retornará undefined
```

```
const fruits = [“Banana”, “Orange”, “Apple”];
```

Métodos de Arrays

- join() – transforma um array numa string, com um separador definido como parâmetro:

```
fruits.join(“ / ”); // Banana / Orange / Apple
```

- push e pop: adicionar e remover elementos

```
fruits.pop(); // Remove “Apple”
```

- shift e unshift: shift é um pop no início do array; unshift adiciona no início e retorna o novo length
- splice: adiciona novos elementos em um array numa posição específica

```
fruits.splice(2, 0, ”Lemon”, “Kiwi”); /* [“Banana”, “Orange”, ”Lemon“,
                                          “Kiwi”, “Apple”] */
```

Arrays

Métodos auxiliares para Arrays

- forEach
- map
- filter
- find
- every
- some
- reduce

Forma tradicional de iterar um array

```
var frutas = ['abacaxi', 'maça', 'uva']; }
for(var i=0; i < frutas.length; frutas++) {
   // corpo da iteração
}
```

Para que serve a iteração???

```
var frutas = ['abacaxi', 'maça', 'uva']; }
frutas.forEach(function(fruta) {
   // corpo da iteração
}
```

E agora?

forEach

```
var frutas = ['abacaxi', 'maça', 'uva']; }
for(var i=0; i < frutas.length; frutas++) {
   // corpo da iteração
}
```

Para que serve a iteração???

```
var frutas = ['abacaxi', 'maça', 'uva']; }
frutas.forEach(function(fruta) {
   // corpo da iteração
}
```

E agora?

Map

- Quando precisarmos iterar elementos de um array fazendo alguma modificação
- Jeito antigo:

```
var numeros = [1,2,3];
var dobro = [];
for( var i = 0; i < numeros.length; i++) {
        dobro.push(numeros[i] *2);
}
console.log(numeros); // [1,2,3]
console.log(dobro); // [2,4,6]
```

Map

- Usando map:

```
var numeros = [1,2,3];
var dobro = numeros.map(function(numero){
    return numero * 2;
}
console.log(numeros); // [1,2,3]
console.log(dobro); // [2,4,6]
```

- O map executa a função de callback recebida por parâmetro para cada elemento iterado e constrói um novo array.

Filter

- Quando precisarmos filtrar elementos de uma lista.
- Jeito antigo:

```
var alunos = [
    {nome:'joão', idade: 15},
    {nome:'josé', idade: 18},
    {nome:'maria', idade: 20}];

var alunosMaiores = [];
for (var i = 0; i < alunos.length; i++) {
    if(alunos[i].idade >=18) {
        alunosMaiores.push(alunos[i]);
    }
}
console.log(alunosMaiores); // [{nome:'josé', idade:18}, {nome:'maria', idade:20}]
```

Filter

- Quando precisarmos filtrar elementos de uma lista.
- Jeito antigo:

```
var alunos = [
    {nome:'joão', idade: 15},
    {nome:'josé', idade: 18},
    {nome:'maria', idade: 20}];

var alunosMaiores = alunos.filter(function(aluno){
    return aluno.idade >= 18;
});
console.log(alunosMaiores); // [{nome:'josé', idade:18}, {nome:'maria', idade:20}]
```

Find

- Para encontrar um item específico dentro de um array.
- Jeito antigo

```
var alunos = [
    {nome:'joão', idade: 15},
    {nome:'josé', idade: 18},
    {nome:'maria', idade: 20}];

var aluno;
for(var i = 0; i < alunos.length; i++){
    if(alunos[i].nome === 'josé'){
        aluno = alunos[i];
        break; // evita percorrer o restante da lista
});
console.log(aluno); // {nome:'josé', idade:18}
```

Find

- Para encontrar um item específico dentro de um array.
- Usando o Find

```
var alunos = [
    {nome:'joão', idade: 15},
    {nome:'josé', idade: 18},
    {nome:'maria', idade: 20}];

var aluno = alunos.find(function(aluno){
    return aluno.nome === 'josé';
});
console.log(aluno); // {nome:'josé', idade:18} só a primeira ocorrência
```

Every

- Verificar se um Array respeita uma condição
- Jeito antigo:

```
var alunos = [
    {nome:'joão', idade: 18},
    {nome:'josé', idade: 20},
    {nome:'maria', idade: 24}];

var todosAlunosMaiores = true;
for(var i = 0; i < alunos.length; i++){
    if(alunos[i].idade < 18){
        todosAlunosMaiores = false;
        break; // evita percorrer o restante da lista
});
console.log(aluno); // true
```

Every

- Verificar se um Array respeita uma condição
- Com every()

```
var alunos = [
    {nome:'joão', idade: 18},
    {nome:'josé', idade: 20},
    {nome:'maria', idade: 24}];

var todosAlunosMaiores = alunos.every(function(aluno){
    return aluno.idade > 18;
});
console.log(todosAlunosMaiores); // true
```

Some

- Verificar se algum elemento do Array satisfaz a condição
- Jeito antigo

```
var pesoDasMalas = [12,32,21,29];
var temMalaAcimaDoPeso = false;
for(var i = 0; i < pesoDasMalas.length; i++) {
    if(pesoDasMalas[i] > 30) {
        temMalaAcimaDoPeso = true;
    }
}
console.log(temMalaAcimaDoPeso); //true
```

Some

- Verificar se algum elemento do Array satisfaz a condição
- Usando some

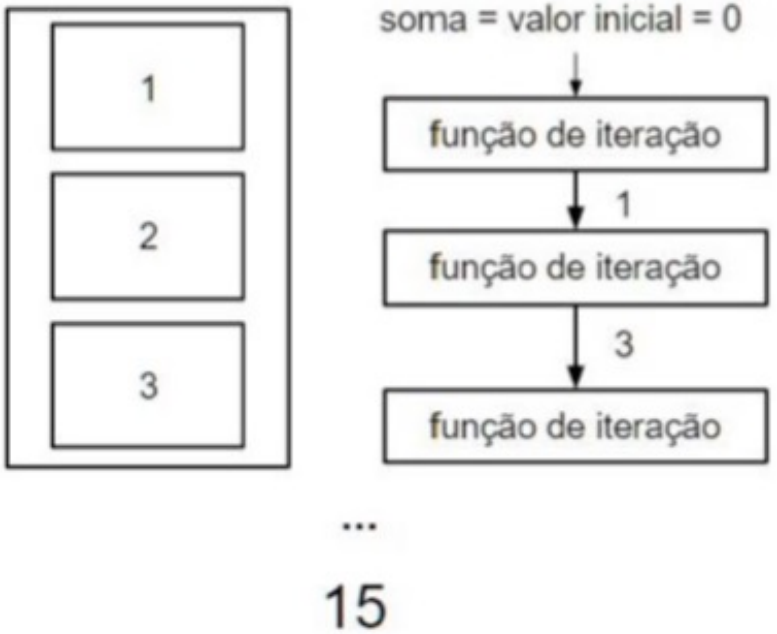
```
var pesoDasMalas = [12,32,21,29];
var temMalaAcimaDoPeso = pesoDasMalas.some(function(pesoDaMala) {
    return pesoDaMala > 30;
});

console.log(temMalaAcimaDoPeso); //true
```

Reduce

- Soma ou concatenação, definindo um valor inicial

```
var numeros = [1,2,3,4,5];
var soma = 0;
soma = numeros.reduce(function(soma, numero){
    return soma + numero;
}, 0);
console.log(soma); // 15
```



- function(soma, numero): função de iteração com acumulador e param iteração
- 0 : um valor inicial qualquer

Reduce

- Aplicação

```
var alunos = [
    {nome:'joão', idade: 10},
    {nome:'josé', idade: 20},
    {nome:'maria', idade: 30}];

var nomes = alunos.reduce(function(arrayNomes, aluno){
    arrayNomes.push(aluno.nome);
    return arrayNomes;
}, []);
console.log(nomes); // ['joão', 'josé', 'maria']
```