

# Tableaux et Boucles

## (JavaScript)

```
var beatles = [];
```



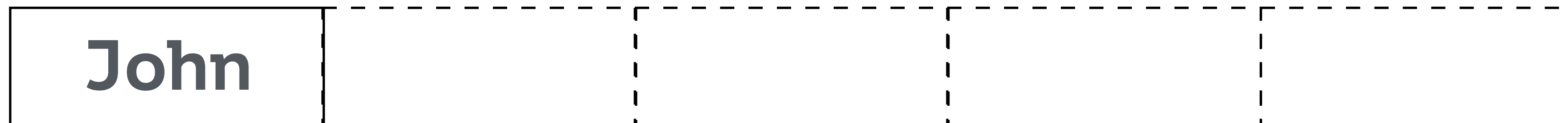
```
var beatles = [];
```

```
beatles.length;
```

```
// => 0
```

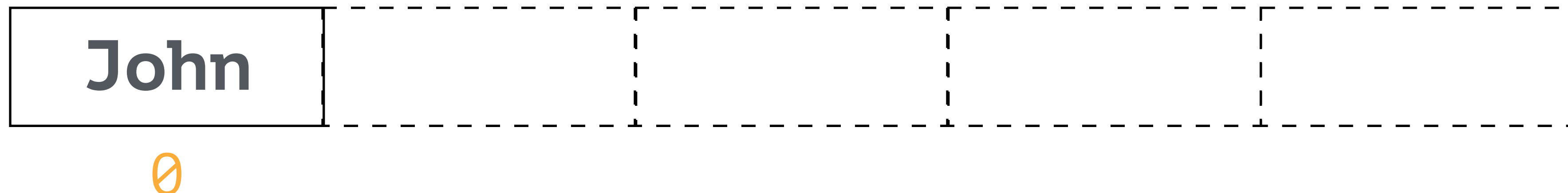


```
var beatles = [];  
beatles.push("John");
```



```
var beatles = [];  
beatles.push("John");
```

```
beatles[0];  
// => "John"
```



```
var beatles = ["John"];  
beatles.push("Paul");  
beatles.push("Ringo");
```

<b>John</b>	<b>Paul</b>	<b>Ringo</b>		
-------------	-------------	--------------	--	--

```
var beatles = ["John"];  
beatles.push("Paul");  
beatles.push("Ringo");
```

```
beatles[beatles.length - 1];  
// => "Ringo"
```



Array - JavaScript | MDN

Mozilla Foundation (US)https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets\_globaux/ArrayRechercher

Se connecter avec

mozilla

MDNMOZILLA DEVELOPER NETWORK

ZONES

PLATEFORME WEB

OUTILS

DÉMOS

PARTICIPER

MDN > Technologies Web pour développeurs > JavaScript > Référence JavaScript > Objets globaux > Array

LANGUES

MODIFIER

# Array

MASQUER LA BARRE LATÉRALE

VOIR AUSSI

Objets standards

Array

Propriétés

Array.prototype

Array.length

Méthodes

Array.from()

Array.isArray()

Array.observe()

Array.of()

Array.prototype.concat()

Les objets globaux **Array** sont des constructeurs pour tableaux, qui sont des objets de haut-niveau (en termes de complexité homme-machine) et qui sont semblables à des listes.

## Syntaxe

```
[element0, element1, ..., elementN]
new Array(element0, element1[, ..., [, elementN]])
new Array(arrayLength)
```

**element0, element1, ..., elementN**

Un tableau est initialisé avec les éléments donnés, sauf dans le cas où un argument seul est passé au constructeur **Array** et que l'argument est un nombre. (Voir ci-après.) Notez que ce cas spécial s'applique aux tableaux créés avec le constructeur **Array**, pas avec des tableaux créés avec des littéraux qui utilisent les crochets.

**arrayLength**

### DANS CET ARTICLE

Syntaxe

Description

Accéder aux éléments d'un tableau

Relation entre `length` et les propriétés numériques

Création d'un tableau utilisant le résultat d'une correspondance

Propriétés

Méthodes

Instances d'Array

Les propriétés

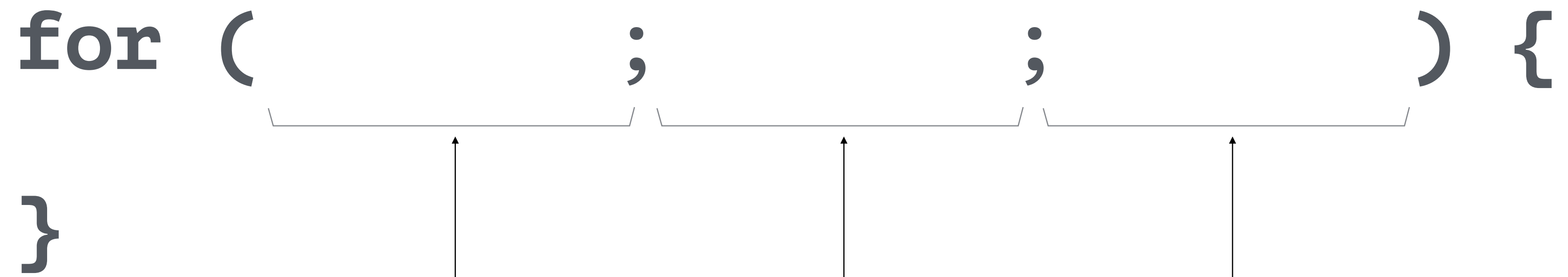
Les méthodes

Les mutateurs

Les accesseurs



```
for (      ;      ;      ) {  
}
```



The diagram shows the syntax of a for loop: `for ( ; ; ) { }`. Brackets are placed under the opening parenthesis, the first semicolon, the second semicolon, and the closing parenthesis. Three vertical arrows point from the labels below to these brackets: 'Initialisation' points to the first bracket, 'Condition d'arrêt' points to the second bracket, and 'Code exécuté à chaque fin d'itération' points to the third bracket.

Initialisation

Condition d'arrêt

Code exécuté à chaque fin d'itération

```
var grades = [12, 18, 15, 9, 13];
```

<b>12</b>	<b>18</b>	<b>15</b>	<b>9</b>	<b>13</b>
-----------	-----------	-----------	----------	-----------

```
var grades = [12, 18, 15, 9, 13];
```

```
var sum = 0;
```

```
for (var i = 0; i < grades.length; i++) {  
    sum = sum + grades[i];  
}
```

**sum**



0

12

18

15

9

13

0

1

2

3

4

```
var grades = [12, 18, 15, 9, 13];
```

```
var sum = 0;
```

```
for (var i = 0; i < grades.length; i++) {  
    sum = sum + grades[i];  
}
```

Étape 0 - Initialisation (une seule fois)



```
var grades = [12, 18, 15, 9, 13];
```

```
var sum = 0;
```

```
for (var i = 0; i < grades.length; i++) {  
    sum = sum + grades[i];  
}
```

Étape 1 - Test de la condition d'arrêt



```
var grades = [12, 18, 15, 9, 13];
```

```
var sum = 0;
```

```
for (var i = 0; i < grades.length; i++) {  
    sum = sum + grades[i];  
}
```

Étape 2 - Exécution du bloc



```
var grades = [12, 18, 15, 9, 13];
```

```
var sum = 0;
```

```
for (var i = 0; i < grades.length; i++) {  
    sum = sum + grades[i];  
}
```

Étape 3 - Exécution du code de fin d'itération



```
var grades = [12, 18, 15, 9, 13];
```

```
var sum = 0;
```

```
for (var i = 0; i < grades.length; i++) {  
    sum = sum + grades[i];  
}
```

On recommence à l'étape 1 !





```
var grades = [12, 18, 15, 9, 13];
```

```
var sum = 0;
```

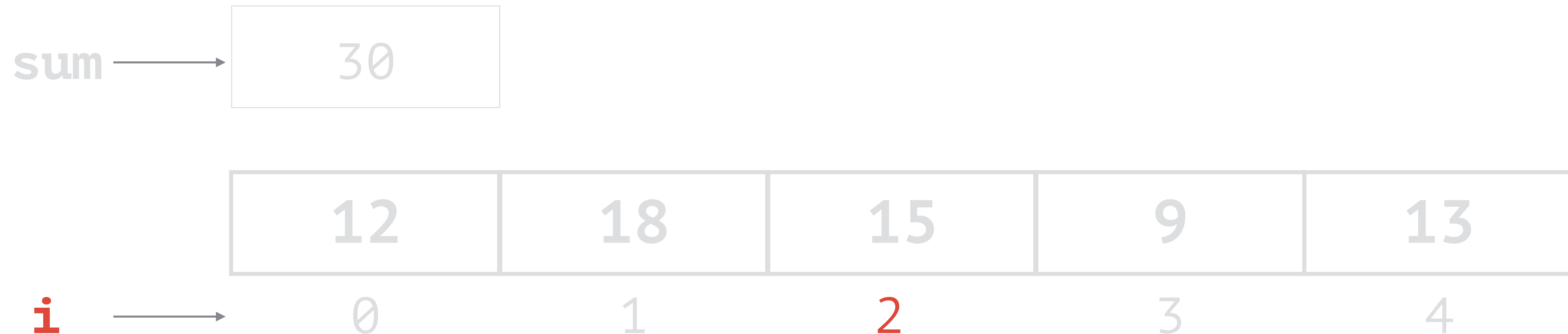
```
for (var i = 0; i < grades.length; i++) {  
    sum = sum + grades[i];  
}
```



```
var grades = [12, 18, 15, 9, 13];
```

```
var sum = 0;
```

```
for (var i = 0; i < grades.length; i++) {  
    sum = sum + grades[i];  
}
```



[...]

```
var grades = [12, 18, 15, 9, 13];
```

```
var sum = 0;
```

```
for (var i = 0; i < grades.length; i++) {  
    sum = sum + grades[i];  
}
```



```
var grades = [12, 18, 15, 9, 13];
```

```
var sum = 0;
```

```
for (var i = 0; i < grades.length; i++) {  
    sum = sum + grades[i];  
}
```



```
var grades = [12, 18, 15, 9, 13];
```

```
var sum = 0;
```

```
for (var i = 0; i < grades.length; i++) {  
    sum = sum + grades[i];  
}
```



**false**

**sum**



**67**

La boucle s'arrête !



**i**



0

1

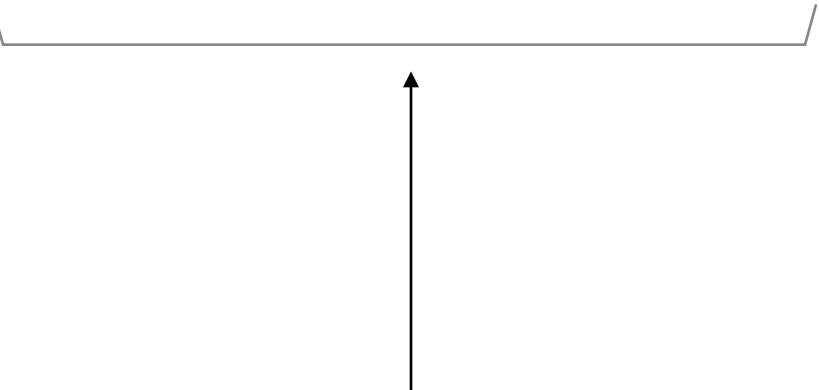
2

3

4

**5**

```
while (      ) {  
    }  
}
```

A horizontal bracket is positioned below the opening parenthesis of the while loop. A vertical arrow points upwards from the text 'Condition d'arrêt' to the center of this bracket.

Condition d'arrêt

```
var grades = [12, 18, 15, 9, 13];
```

```
var sum = 0;
```



```
var grades = [12, 18, 15, 9, 13];
```

```
var sum = 0;
```

```
var i = 0;
```

```
var grades = [12, 18, 15, 9, 13];
```

```
var sum = 0;
```

```
var i = 0;
```

```
while (i < grades.length) {
```

```
}
```

```
var grades = [12, 18, 15, 9, 13];
```

```
var sum = 0;
```

```
var i = 0;
```

```
while (i < grades.length) {
```

```
    sum = sum + grades[i];
```

```
}
```

```
var grades = [12, 18, 15, 9, 13];
```

```
var sum = 0;
```

```
var i = 0;
```

```
while (i < grades.length) {
```

```
    sum = sum + grades[i];
```

```
    i++;
```

```
}
```