



Cette page a été traduite à partir de l'anglais par la communauté. Vous pouvez également contribuer en rejoignant la communauté francophone sur MDN Web Docs.

for...in

L'instruction **for...in** permet d'itérer sur les [propriétés énumérables](#) d'un objet qui ne sont pas [des symboles](#). Pour chaque propriété obtenue, on exécute une instruction (ou plusieurs grâce à un [bloc](#) d'instructions).

JavaScript Demo: Statement - For...In

```
1 const object = { a: 1, b: 2, c: 3 };
2
3 for (const property in object) {
4   console.log(`${property}: ${object[property]}`);
5 }
6
7 // expected output:
8 // "a: 1"
```

```
8 // a: 1
9 // "b: 2"
10 // "c: 3"
11
```

Run ›Reset

Syntaxe

```
for (variable in objet) {
  instructions
}
```

variable

Un nom de propriété différent est assigné à la variable à chaque itération de la boucle.

objet

L'objet dont les propriétés énumérables et qui ne sont pas des symboles sont parcourues par la boucle.

Description

Une boucle `for...in` ne parcourt que les propriétés énumérables et qui ne sont pas des symboles. Les objets qui ont été créés par des constructeurs intégrés comme `Array` et `Object` ont hérité de propriétés non énumérables de `Object.prototype` et `String.prototype` comme les méthodes `indexOf()` du type `String` ou `toString()` depuis `Object`. La boucle parcourera toutes les propriétés énumérables de l'objet et aussi celles dont il hérite du prototype du constructeur (les propriétés les plus proches de l'objet dans la chaîne des prototypes primeront sur les propriétés des prototypes).

Les propriétés ajoutées, modifiées ou supprimées

Une boucle `for...in` parcourt les propriétés d'un objet dans un ordre arbitraire (voir l'opérateur `delete` pour plus d'explications quant à l'impossibilité de se fier à un tel ordre, au moins dans le cas où on souhaite gérer plusieurs navigateurs).

Si une propriété est modifiée dans une des itérations de la boucle et que la boucle itère ensuite sur cette propriété, sa valeur sera celle qui a été modifiée. Une propriété qui a été supprimée avant que la boucle n'itère sur celle-là ne sera pas dans la boucle. Les propriétés

qui ont été ajoutées à l'objet pendant la boucle peuvent être ou ne pas être pris en compte.

Une bonne pratique consiste à ne pas ajouter, modifier ou supprimer une propriété d'un objet lors d'une itération qui ne concerne pas cette propriété. Il n'y a aucune certitude concernant la prise en compte d'une propriété ajoutée lors d'une telle boucle et il en va de même pour savoir si on a visité une propriété avant ou après qu'elle ait été modifiée ou de savoir si une itération de la boucle concernera une propriété avant que celle-ci soit supprimée.

Utiliser `for...in` et parcourir un tableau

Note : `for...in` ne doit pas être utilisée pour parcourir un [Array](#) lorsque l'ordre des éléments est important.

Les éléments des indices d'un tableau sont des propriétés énumérables dont les noms sont des entiers, excepté cela, elles sont en tout point identiques aux propriétés des objets en général. Ici aussi, il n'y a aucune certitude que `for...in` renvoie les indices dans un ordre particulier. Cette instruction listera également les propriétés énumérables dont les noms ne sont pas des entiers et celles qui sont héritées.

L'ordre dans lequel le parcours est effectué dépend de l'implémentation. Dans le cas d'un parcours de tableau utilisant `for...in`, on pourrait très bien avoir un ordre qui ne soit pas le même entre les différents environnements. Pour cette raison, il est préférable d'utiliser une boucle [for](#) utilisant un compteur numérique (ou [Array.forEach\(\)](#) ou encore [for...of](#)) lorsqu'on souhaite parcourir des tableaux dans un ordre bien défini.

Itérer uniquement sur les propriétés non héritées

Si on souhaite ne parcourir que les propriétés propres d'un objet et pas celles rattachées à ses prototypes, on peut utiliser la méthode [Object.getOwnPropertyNames\(\)](#) ou bien effectuer un test grâce à la méthode [Object.prototype.hasOwnProperty\(\)](#) voire avec [Object.prototype.propertyIsEnumerable\(\)](#)

Exemples

La boucle `for...in` qui suit utilise parcourt l'objet `obj` et ses propriétés énumérables qui ne sont pas des symboles en fournissant la chaîne de caractères qui décrit le nom de la propriété et sa valeur.

```
var obj = {a:1, b:2, c:3};

for (var prop in obj) {
  console.log(`obj.${prop} = ${obj[prop]}`);
}
```



```
// Affiche dans la console :  
// "obj.a = 1"  
// "obj.b = 2"  
// "obj.c = 3"
```

La fonction qui suit utilise [hasOwnProperty\(\)](#) pour ne pas afficher les propriétés héritées :

```
var triangle = {a:1, b:2, c:3};  
  
function TriangleCouleur() {  
  this.couleur = "rouge";  
}  
  
TriangleCouleur.prototype = triangle;  
  
var obj = new TriangleCouleur();  
  
for (var prop in obj) {  
  if( obj.hasOwnProperty( prop ) ) {  
    console.log(`obj.${prop} = ${obj[prop]}`);  
  }  
}  
  
// Affichera dans la console :  
// "obj.couleur = rouge"
```



Spécifications

Spécification	État	Commentaires
ECMAScript 1st Edition (ECMA-262) La définition de 'for...in statement' dans cette spécification.	Standard	Définition initiale.

ECMAScript 5.1 (ECMA-262) La définition de 'for...in statement' dans cette spécification.	Standard	
ECMAScript 2015 (6th Edition, ECMA-262) La définition de 'for...in statement' dans cette spécification.	Standard	
ECMAScript (ECMA-262)		

La définition de 'for...in statement' dans cette spécification.	Standard évolutif	
Spécification	État	Commentaires

Compatibilité des navigateurs

[Report problems with this compatibility data on GitHub](#)

for...in		
Chrome		1
Edge		12
Firefox		1
Internet Explorer		6
Opera		2
Safari		1
WebView Android		1
Chrome Android		18
Firefox for Android		4
Opera Android		10.1
Safari on iOS		1
Samsung Internet		1.0
Deno		1.0
Node.js		0.10.0

☐

 Full support

Expressions avec initialiseur

Avant SpiderMonkey 40 (Firefox 40 / Thunderbird 40 / SeaMonkey 2.37), il était possible d'utiliser un initialiseur (`i=0`) dans un boucle `for...in` :


```
var obj = {a:1, b:2, c:3};
for(var i=0 in obj) {
  console.log(obj[i]);
}
```

```
}  
// 1  
// 2  
// 3
```

Ce comportement non-standard a été retiré avec la version 40. Cela provoquera désormais une exception [SyntaxError](#) ("*for-in loop head declarations may not have initializers*") en [mode strict](#) (cf. [bug 748550](#) et [bug 1164741](#)).

Les autres moteurs, tels que v8 (Chrome), Chakra (IE/Edge) et JSC (WebKit/Safari) recherchent également comment retirer ce comportement non standard.

Voir aussi

- [for...of](#) : une instruction semblable qui permet d'itérer sur les valeurs des propriétés
- [for](#)
- [Le rattachement et le caractère énumérable des propriétés](#)
- [Object.getOwnPropertyNames\(\)](#)
- [Object.prototype.hasOwnProperty\(\)](#)
- [Array.prototype.forEach\(\)](#)
- [for each...in](#)  : une instruction semblable, dépréciée, qui parcourt les valeurs des propriétés d'un objet plutôt que les noms.

Last modified: 24 févr. 2022, [by MDN contributors](#)