

 Rechercher sur MDN

Technologies ▼

Guides et références ▼

Votre avis ▼

# RegExp

Français ▼

Le constructeur **RegExp** crée un objet expression rationnelle pour la reconnaissance d'un modèle dans un texte.

Pour une introduction aux expressions rationnelles, lire le chapitre [Expressions rationnelles](#) dans le Guide JavaScript.

## JavaScript Demo: RegExp Constructor

```
1 var regex1 = /\w+/;
2 var regex2 = new RegExp('\\w+');
3
4 console.log(regex1);
5 // expected output: /\w+/
6
7 console.log(regex2);
8 // expected output: /\w+/
9
10 console.log(regex1 === regex2);
11 // expected output: false
12
```

Run ›

Reset

# Syntaxe

Les notations littérales, par constructeur ou de base sont possibles :

```
/modèle/marqueurs  
new RegExp(modèle[, marqueurs])  
RegExp(modèle[, marqueurs])
```

## Paramètres

### modèle

Le texte de l'expression rationnelle ou, à partir d'ES5, un autre objet ou littéral `RegExp` à copier. Ce motif peut inclure certains caractères spéciaux pour correspondre à un ensemble de valeurs plus large (qu'une simple chaîne littérale).

### marqueurs

Si cet argument est utilisé, il indique les marqueurs à utiliser pour l'expression rationnelle. Ces valeurs remplaceront celles de l'objet à copier si `modèle` est un objet `RegExp` (`lastIndex` sera réinitialisé à 0 à partir d'ECMAScript 2015 / ES6). Cet argument est une chaîne de caractères qui peut contenir une combinaison des valeurs suivantes:

#### g

recherche globale ; retrouve toutes les correspondances plutôt que de s'arrêter après la première.

#### i

la casse est ignorée. Si le marqueur `u` est également activé, les caractères Unicode équivalents pour la casse correspondent.

#### m

multiligne : les caractères de début et de fin (^ et \$) sont traités comme travaillant sur des lignes multiples (i.e, ils correspondent au début et à la fin de *chaque* ligne (délimitée par `\n` ou `\r`), pas seulement au début ou à la fin de la chaîne d'entrée complète).

#### u

unicode : traite le modèle comme une séquence de points de code Unicode (voir également les chaînes binaires).

**y**

adhérence : n'établit de correspondance qu'à partir de l'indice dans la chaîne cible indiqué par la propriété `lastIndex` de l'expression rationnelle (et ne cherche pas à établir de correspondance à partir d'indices au delà).

**s**

"dotAll" : permet d'indiquer que `.` peut correspondre à un saut de ligne.

---

## Description

Il existe deux façons de créer un objet `RegExp` : une notation littérale ou un constructeur. La notation littérale est délimitée par des barres obliques (*slashes*) tandis que le constructeur utilise des apostrophes. Ainsi, les expressions suivantes créent la même expression rationnelle :

```
1 | /ab+c/i;           // notation littérale
2 | new RegExp('ab+c', 'i'); // constructeur
3 | new RegExp(/ab+c/, 'i'); // notation littérale dans un constructeur
```

La notation littérale effectue la compilation de l'expression rationnelle lorsque l'expression est évaluée. Utilisez la notation littérale lorsque l'expression rationnelle reste constante. Par exemple, si vous utilisez la notation littérale pour construire une expression rationnelle utilisée dans une boucle, l'expression rationnelle ne sera pas recompilée à chaque itération.

Le constructeur de l'objet expression rationnelle, par exemple `new RegExp('ab+c')`, effectue la compilation de l'expression rationnelle au moment de l'exécution. Utilisez la fonction constructeur quand vous savez que le modèle d'une expression rationnelle sera variable, ou si vous ne connaissez pas le modèle et que vous l'obtiendrez d'une autre source, telle qu'une saisie utilisateur.

À partir d'ECMAScript 6, `new RegExp(/ab+c/, 'i')` ne déclenche plus d'exception `TypeError` ("can't supply flags when constructing one RegExp from another") lorsque le premier argument est une `RegExp` et que le second argument `marqueurs` est présent. Une nouvelle `RegExp` sera créée à la place à partir des arguments.

Lorsqu'on utilise le constructeur, les règles normales d'échappement de chaîne (le fait de faire précéder d'un `\` les caractères spéciaux à l'intérieur d'une chaîne) sont requises. Par exemple, les définitions suivantes sont équivalentes :

```
1 | var re = /\w+/;  
2 | var re = new RegExp('\\w+');
```

---

## Propriétés

### **RegExp.prototype**

Cette propriété permet d'ajouter des propriétés à tous les objets qui sont des expressions rationnelles.

### **RegExp.length**

La valeur de longueur pour le constructeur dont la valeur est 2.

### **get RegExp[*@@species*]**

La fonction de construction utilisée pour créer les objets dérivés.

### **RegExp.lastIndex**

L'indice à partir duquel rechercher la prochaine correspondance.

---

## Méthodes

L'objet global `RegExp` ne possède pas de méthode propre. En revanche, il hérite de certaines méthodes via sa chaîne de prototypes.

---

## Le prototype de `RegExp` et les instances

### Propriétés

Voir également la page sur les propriétés dépréciées de `RegExp`.

On notera que plusieurs des propriétés de `RegExp` ont un nom court et un nom long (semblable aux noms Perl). Le nom court et le nom long font référence à la même propriété. La modélisation des expressions rationnelles JavaScript est basée sur celle de Perl, un autre langage de programmation.

### **`RegExp.prototype.constructor`**

Définit la fonction qui crée le prototype d'un objet.

### **`RegExp.prototype.flags`**

Une chaîne qui contient les drapeaux (*flags*) utilisés pour l'objet `RegExp`.

### **`RegExp.prototype.dotAll`**

Indique si `.` peut correspondre à des sauts de ligne.

### **`RegExp.prototype.global`**

Définit si l'expression rationnelle doit relever la première correspondance d'une chaîne ou toutes les correspondances.

### **`RegExp.prototype.ignoreCase`**

Définit si l'expression rationnelle doit ignorer la casse ou non pour détecter une correspondance.

### **`RegExp.prototype.multiline`**

Définit si la recherche de la correspondance s'effectue sur plusieurs lignes ou sur une seule.

### **`RegExp.prototype.source`**

Le texte du motif (*pattern*) à rechercher.

### **`RegExp.prototype.sticky`**

Définit si la recherche s'effectue uniquement à partir de `lastIndex` ou non.

### **`RegExp.prototype.unicode`**

Cette propriété indique si les fonctionnalités Unicode sont activées ou non.

## Méthodes

### **`RegExp.prototype.compile()`**

(Re)compile une expression rationnelle lors de l'exécution d'un script.

### **`RegExp.prototype.exec()`**

Exécute une recherche de correspondance sur la chaîne de caractères fournie en paramètre.

### **RegExp.prototype.test()**

Teste s'il y a une correspondance dans la chaîne de caractères fournie en paramètre.

### **RegExp.prototype[@@match]()**

Teste une correspondance sur une chaîne de caractères donnée et renvoie le résultat du test.

### **RegExp.prototype[@@matchAll]()**

Renvoie l'ensemble des correspondances d'une expression rationnelle sur une chaîne.

### **RegExp.prototype[@@replace]()**

Remplace les correspondances d'une chaîne de caractères avec une nouvelle sous-chaînes.

### **RegExp.prototype[@@search]()**

Recherche la correspondance dans une chaîne de caractères donnée et renvoie la position où est trouvé le motif.

### **RegExp.prototype[@@split]()**

Découpe une chaîne de caractères en un tableau de sous-chaînes.

### **RegExp.prototype.toString()**

Renvoie un littéral objet représentant l'objet spécifié. Cette méthode peut être utilisée pour créer un nouvel objet. Elle surcharge la méthode `Object.prototype.toString()`.

### **RegExp.prototype.toString()**

Renvoie une chaîne de caractères représentant l'objet spécifié. Cette méthode surcharge `Object.prototype.toString()`.

---

## **Exemples**

Utiliser une expression rationnelle pour modifier un format de données

Dans le script suivant, on utilise la méthode `replace()` de `String` pour effectuer une correspondance sur le prénom et le nom pour les inverser. On utilise des parenthèses capturantes pour pouvoir utiliser les correspondances dans la construction du résultat (avec `$1` et `$2`).

```
1 | var re = /(\w+)\s(\w+)/;  
2 | var chaîne = 'Alain Dupont';  
3 | var nouvelleChaîne = chaîne.replace(re, '$2, $1');  
4 | console.log(nouvelleChaîne);
```

Cela affichera "Dupont, Alain".

## Utiliser une expression rationnelle pour découper des lignes avec différents sauts de ligne/fins de ligne

La fin de ligne par défaut dépend de la plateforme (Unix, Windows, etc.). Cette méthode de découpage fournie permet de découper indépendamment de la plateforme utilisée.

```
1 | var texte = 'Un texte\net un autre\r\npuis ensuite\r\nla fin';  
2 | var lignes = texte.split(/\r\n|\r|\n/);  
3 | console.log(lignes); // affiche [ 'Un texte', 'et un autre', 'puis ensui
```

On voit ici que l'ordre des modèles dans l'expression rationnelle importe.

## Utiliser une expression rationnelle sur plusieurs lignes

```
1 | var s = 'Et voici\nune autre ligne !';  
2 | s.match(/voici.*ligne/);  
3 | // Renvoie null  
4 | s.match(/voici[^]*ligne/);  
5 | // Renvoie ['voici\nune autre ligne']
```

## Utiliser une expression rationnelle avec le marqueur d'adhérence

Cet exemple illustre comment on peut utiliser ce marqueur qui recherche une correspondance après `RegExp.prototype.lastIndex`.

```
1  var str = '#toto#';
2  var regex = /toto/y;
3
4  regex.lastIndex; // 0
5  regex.test(str); // true
6  regex.lastIndex = 1;
7  regex.test(str); // true
8  regex.lastIndex = 5;
9  regex.test(str); // false (lastIndex est pris en compte avec ce marqueur
10 regex.lastIndex; // 0 (réinitialisation suite à l'échec)
```

## Les expressions rationnelles et les caractères Unicode

Comme mentionné ci-avant, les classes `\w` ou `\W` ne correspondent qu'à des caractères ASCII "a" à "z", "A" à "Z", "0" à "9" et `_`. Pour effectuer des correspondances sur d'autres caractères (comme par exemple les caractères cyrilliques), on utilisera `\uhhhh`, où "hhhh" représente la valeur Unicode exprimée en hexadécimal. Cet exemple illustre comment il est possible de séparer les caractères Unicode d'un mot.

```
1  var texte = 'Образец text на русском языке';
2  var regex = /[\u0400-\u04FF]+/g;
3
4  var corresp = regex.exec(texte);
5  console.log(corresp[0]); // affiche 'Образец'
6  console.log(regex.lastIndex); // affiche '7'
7
8  var corresp2 = regex.exec(texte);
9  console.log(corresp2[0]); // affiche 'на' (n'affiche pas text
10 console.log(regex.lastIndex); // affiche '15'
11
12 // et ainsi de suite
```





Voici une ressource tierce pour obtenir les différents intervalles Unicode des différents alphabets : [Regexp-unicode-block](https://github.com/Regexp-Unicode/block).

## Extraire un sous-domaine d'une URL



```
1 | var url = 'http://xxx.domaine.com';  
2 | console.log(/^[^.]*/.exec(url)[0].substr(7)); // affiche 'xxx'
```

## Spécifications

Spécification	État	Commentaires
ECMAScript 1st Edition (ECMA-262)	 ST Standard	Définition initiale. Implémentée avec JavaScript 1.1.
ECMAScript 5.1 (ECMA- 262) La définition de 'RegExp' dans cette spécification.	 ST Standard	
ECMAScript 2015 (6th Edition, ECMA-262) La définition de 'RegExp' dans cette spécification.	 ST Standard	Le constructeur RegExp ne renvoie plus d'exception lorsqu'il est utilisé avec un objet RegExp et que le second argument est utilisé. Ajout du marqueur d'adhérence et du marqueur Unicode.
ECMAScript Latest Draft (ECMA-262) La définition de 'RegExp' dans cette spécification.	 D Projet	

# Compatibilité des navigateurs

[Update compatibility data on GitHub](#)

## RegExp

Chrome	Oui
Edge	Oui
Firefox	1
IE	Oui
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	4
Opera Android	Oui
Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	Oui

## compile

Chrome	Oui
Edge	12
Firefox	1
IE	Oui
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	4
Opera Android	Oui

Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	Oui

**dotAll**

Chrome	62
Edge	Non
Firefox	Non
IE	Non
Opera	49
Safari	12
WebView Android	62
Chrome Android	62
Firefox Android	Non
Opera Android	46
Safari iOS	12
Samsung Internet Android	Oui
nodejs	8.10.0

**exec**

Chrome	Oui
Edge	12
Firefox	1
IE	Oui
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	4
Opera Android	Oui

Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	Oui

### flags

Chrome	Oui
Edge	Oui
Firefox	37
IE	Non
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	37
Opera Android	Oui
Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	6.0.0

### global

Chrome	Oui
Edge	12
Firefox	1
IE	Oui
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	4
Opera Android	Oui

Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	Oui

### ignoreCase

Chrome	Oui
Edge	12
Firefox	1
IE	Oui
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	4
Opera Android	Oui
Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	Oui

### RegExp.input (\$\_)

Chrome	Oui
Edge	12
Firefox	1
IE	Oui
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	4
Opera Android	Oui

Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	Oui

**lastIndex**

Chrome	Oui
Edge	12
Firefox	1
IE	Oui
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	4
Opera Android	Oui
Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	Oui

**RegExp.lastMatch (\$&)**

Chrome	Oui
Edge	12
Firefox	1
IE	Oui
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	4
Opera Android	Oui

Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	Oui

### RegExp.lastParen (\$+)

Chrome	Oui
Edge	12
Firefox	1
IE	Oui
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	4
Opera Android	Oui
Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	Oui

### RegExp.leftContext (\$`)

Chrome	Oui
Edge	12
Firefox	1
IE	Oui
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	4
Opera Android	Oui

Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	Oui

lookbehind assertions ( (?<= ) and (?<! ) )

Chrome	62
Edge	Non
Firefox	Non
IE	Non
Opera	49
Safari	Non
WebView Android	62
Chrome Android	62
Firefox Android	Non
Opera Android	46
Safari iOS	Non
Samsung Internet Android	Oui
nodejs	8.10.0

### multiline

Chrome	Oui
Edge	12
Firefox	1
IE	Oui
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	4
Opera Android	Oui



Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	Oui

### RegExp.\$1-\$9

Chrome	Oui
Edge	Oui
Firefox	1
IE	Oui
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	4
Opera Android	Oui
Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	Oui

### Named capture groups

Chrome	64
Edge	Non
Firefox	Non
IE	Non
Opera	51
Safari	11.1
WebView Android	64
Chrome Android	64
Firefox Android	Non
Opera Android	47

Safari iOS	11.3
Samsung Internet Android	9.0
nodejs	10.0.0

## Unicode property escapes ( \p{...} )

Chrome	64
Edge	?
Firefox	Non
IE	?
Opera	51
Safari	?
WebView Android	64
Chrome Android	64
Firefox Android	Non
Opera Android	47
Safari iOS	?
Samsung Internet Android	Oui
nodejs	10.0.0

## prototype

Chrome	Oui
Edge	Oui
Firefox	1
IE	Oui
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	4
Opera Android	Oui

Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	Oui

### RegExp.rightContext (\$')

Chrome	Oui
Edge	12
Firefox	1
IE	Oui
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	4
Opera Android	Oui
Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	Oui

### source

Chrome	Oui
Edge	12
Firefox	1
IE	Oui
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	4
Opera Android	Oui

Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	Oui

**sticky**

Chrome	49
Edge	13
Firefox	3
IE	Non
Opera	36
Safari	10
WebView Android	49
Chrome Android	49
Firefox Android	4
Opera Android	36
Safari iOS	10
Samsung Internet Android	5.0
nodejs	Oui

**test**

Chrome	Oui
Edge	12
Firefox	1
IE	Oui
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	4
Opera Android	Oui

Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	Oui

**toSource**

Chrome	Non
Edge	Non
Firefox	1
IE	Non
Opera	Non
Safari	Non
WebView Android	Non
Chrome Android	Non
Firefox Android	4
Opera Android	Non
Safari iOS	Non
Samsung Internet Android	Non
nodejs	Non

**toString**

Chrome	Oui
Edge	Oui
Firefox	1
IE	Oui
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	4
Opera Android	Oui

Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	Oui

**unicode**

Chrome	50
Edge	12
Firefox	46
IE	Non
Opera	37
Safari	10
WebView Android	Oui
Chrome Android	50
Firefox Android	46
Opera Android	37
Safari iOS	10
Samsung Internet Android	5.0
nodejs	Oui

**@@match**

Chrome	Oui
Edge	Oui
Firefox	49
IE	Non
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	49
Opera Android	Oui

Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	6.0.0

**@@matchAll**

Chrome	73
Edge	Non
Firefox	67
IE	Non
Opera	60
Safari	Non
WebView Android	73
Chrome Android	73
Firefox Android	67
Opera Android	Oui
Safari iOS	Non
Samsung Internet Android	Oui
nodejs	12.0.0

**@@replace**

Chrome	Oui
Edge	Oui
Firefox	49
IE	Non
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	49
Opera Android	Oui

Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	6.0.0

**@@search**

Chrome	Oui
Edge	Oui
Firefox	49
IE	Non
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	49
Opera Android	Oui
Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	6.0.0

**@@species**

Chrome	Oui
Edge	Oui
Firefox	49
IE	Non
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	49
Opera Android	Oui



Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	6.5.0

**@@split**

Chrome	Oui
Edge	Oui
Firefox	49
IE	Non
Opera	Oui
Safari	Oui
WebView Android	Oui
Chrome Android	Oui
Firefox Android	49
Opera Android	Oui
Safari iOS	Oui
Samsung Internet Android	Oui
nodejs	6.0.0

What are we missing?

☐

Support complet

☐

Aucun support

☐

Compatibilité inconnue

Fonctionnalité non-standard. Celle-ci peut être incorrectement supportée par les autres navigateurs.

Obsolète. Les nouveaux sites web ne doivent pas utiliser cette fonctionnalité.

Voir les notes d'implémentation.

Une action explicite de l'utilisateur est nécessaire pour activer cette fonctionnalité.

## Notes spécifiques à Firefox

À partir de Firefox 34, dans le cas où on utilise un groupe capturant avec des quantificateurs qui l'invalident, le texte correspondant au groupe est désormais `undefined` et non la chaîne vide :

```
1 // Firefox 33 ou antérieur
2 'x'.replace(/x(.)?/g, function(m, group) {
3   console.log("'group:" + group + "'");
4 }); // 'group:'
5
6 // Firefox 34 ou supérieur
7 'x'.replace(/x(.)?/g, function(m, group) {
8   console.log("'group:" + group + "'");
9 }); // 'group:undefined'
```

Pour des raisons de compatibilité web, `RegExp.$N` renverra une chaîne vide au lieu de `undefined` (bug 1053944).

---

## Voir aussi

- Le chapitre Expressions rationnelles du Guide JavaScript
- `String.prototype.match()`
- `String.prototype.replace()`

---

Dernière modification : 19 juil. 2019, by MDN contributors

Syntaxe

Description

Propriétés

Méthodes

Le prototype de `RegExp` et les instances

[Exemples](#)[Spécifications](#)[Compatibilité des navigateurs](#)[Voir aussi](#)

## Sujets associés

### Objets standards

#### RegExp

#### Propriétés

`RegExp.$1-$9``RegExp.input ($_)``RegExp.lastMatch ($&)``RegExp.lastParen ($+)``RegExp.leftContext ($`)``RegExp.prototype``RegExp.prototype.dotAll``RegExp.prototype.flags``RegExp.prototype.global``RegExp.prototype.ignoreCase``RegExp.prototype.multiline``RegExp.prototype.source``RegExp.prototype.sticky``RegExp.prototype.unicode``RegExp.rightContext ($')``get RegExp[@@species]``regExp.lastIndex`

#### Méthodes

`RegExp.prototype.compile()``RegExp.prototype.exec()``RegExp.prototype.test()``RegExp.prototype.toSource()``RegExp.prototype.toString()`

```
RegExp.prototype[@@matchAll]()
```

```
RegExp.prototype[@@match]()
```

```
RegExp.prototype[@@replace]()
```

```
RegExp.prototype[@@search]()
```

```
RegExp.prototype[@@split]()
```

## Héritage :

### Function

### Propriétés

```
Function.arguments
```

```
Function.arity
```

```
Function.caller
```

```
Function.displayName
```

```
Function.length
```

```
Function.name
```

```
Function.prototype
```

### Méthodes

```
Function.prototype.apply()
```

```
Function.prototype.bind()
```

```
Function.prototype.call()
```

```
Function.prototype.isGenerator()
```

```
Function.prototype.toSource()
```

```
Function.prototype.toString()
```

### Object

### Propriétés

```
Object.prototype.__count__
```

```
Object.prototype.__noSuchMethod__
```

```
Object.prototype.__parent__
```

```
Object.prototype.__proto__
```

```
Object.prototype.constructor
```

### Méthodes

```
Object.prototype.__defineGetter__()  
Object.prototype.__defineSetter__()  
Object.prototype.__lookupGetter__()  
Object.prototype.__lookupSetter__()  
Object.prototype.hasOwnProperty()  
Object.prototype.isPrototypeOf()  
Object.prototype.propertyIsEnumerable()  
Object.prototype.toLocaleString()  
Object.prototype.toSource()  
Object.prototype.toString()  
    Object.prototype.unwatch()  
Object.prototype.valueOf()  
    Object.prototype.watch()  
Object.setPrototypeOf()
```



# Recevez le meilleur du développement web

Get the latest and greatest from MDN delivered straight to your inbox.

*Cette lettre d'information est uniquement disponible en anglais pour l'instant.*

**Sign up now**