Déployer

L' **`Array`** objet, comme les tableaux dans d'autres langages de programmation, permet de stocker une collection de plusieurs éléments sous un seul nom de variable et possède des membres pour effectuer des opérations de tableau communes .

La description

En JavaScript, les tableaux ne sont pas des primitives , mais plutôt `Array` des objets avec les caractéristiques principales suivantes :

- **Les tableaux JavaScript sont redimensionnables** et **peuvent contenir un mélange de différents types de données** . (Lorsque ces caractéristiques ne sont pas souhaitables, utilisez plutôt des tableaux typés .)
- **Les tableaux JavaScript ne sont pas des tableaux associatifs** et, par conséquent, les éléments du tableau ne sont pas accessibles en utilisant des chaînes comme index , mais doivent être accessibles en utilisant des entiers comme index.
- **Les tableaux JavaScript sontindexé à zéro** : le premier élément d'un tableau est à index `0` , le second est à index `1` , et ainsi de suite — et le dernier élément est à la valeur de la propriété du tableau `length` minus `1` .
- **Les opérations de copie de tableau JavaScript créent des copies superficielles** . (Toutes les opérations de copie intégrées standard avec *des* objets JavaScript créent des copies superficielles, plutôt que des copies profondes).

Constructeur

`Array()`
Crée un nouvel `Array` objet.

Propriétés statiques

`get Array[@@species]`
Renvoie le `Array` constructeur.

Méthodes statiques

`Array.from()`

Crée une nouvelle `Array` instance à partir d'un objet de type tableau ou d'un objet itérable.

`Array.isArray()`
Retourne `true` si l'argument est un tableau, ou `false` autrement.

`Array.of()`
Crée une nouvelle `Array` instance avec un nombre variable d'arguments, quel que soit le nombre ou le type des arguments.

Propriétés d'occurrence

`Array.prototype.length`
Reflète le nombre d'éléments dans un tableau.

`Array.prototype[@@unscopables]`
Contient des noms de propriété qui n'étaient pas inclus dans la norme ECMAScript avant la version ES2015 et qui sont ignorés à `with` des fins de liaison d'instructions.

Méthodes d'instance

`Array.prototype.at()`
Renvoie l'élément de tableau à l'index donné. Accepte les entiers négatifs, qui comptent à partir du dernier élément.

`Array.prototype.concat()`
Renvoie un nouveau tableau qui est le tableau appelant joint à d'autres tableaux et/ou valeurs.

`Array.prototype.copyWithin()`
Copie une séquence d'éléments de tableau dans un tableau.

`Array.prototype.entries()`
Renvoie un nouvel objet *itérateur de tableau* qui contient les paires clé/valeur pour chaque index d'un tableau.

`Array.prototype.every()`

Renvoie `true` si chaque élément du tableau appelant satisfait la fonction de test.

`Array.prototype.fill()`
Remplit tous les éléments d'un tableau d'un index de début à un index de fin avec une valeur statique.

`Array.prototype.filter()`
Renvoie un nouveau tableau contenant tous les éléments du tableau appelant pour lequel la fonction de filtrage fournie renvoie `true`.

`Array.prototype.find()`
Renvoie le trouvé `element` dans le tableau appelant, si un élément du tableau satisfait la fonction de test, ou `undefined` s'il n'est pas trouvé.

`Array.prototype.findIndex()`
Renvoie l'index trouvé dans le tableau appelant, si un élément du tableau satisfait la fonction de test, ou `-1` s'il n'est pas trouvé.

`Array.prototype.flat()`
Renvoie un nouveau tableau avec tous les éléments du sous-tableau concaténés de manière récursive jusqu'à la profondeur spécifiée.

`Array.prototype.flatMap()`
Retourne un nouveau tableau formé en appliquant une fonction de rappel donnée à chaque élément du tableau appelant, puis en aplatissant le résultat d'un niveau.

`Array.prototype.forEach()`
Appelle une fonction pour chaque élément du tableau appelant.

`Array.prototype.groupBy()`
Regroupe les éléments d'un tableau en fonction des résultats d'une fonction de test. Les groupes résultants sont accessibles à l'aide des propriétés de l'objet.

`Array.prototype.includes()`
Détermine si le tableau appelant contient une valeur, retour `true` ou `false` selon le cas.

`Array.prototype.indexOf()`
Renvoie le premier (le plus petit) indice auquel un élément donné peut être trouvé dans le tableau appelant.

`Array.prototype.join()`
Joint tous les éléments d'un tableau dans une chaîne.

`Array.prototype.keys()`
Renvoie un nouvel *itérateur de tableau* contenant les clés de chaque index du tableau appelant.

`Array.prototype.lastIndexOf()`
Renvoie le dernier (le plus grand) index auquel un élément donné peut être trouvé dans

le tableau appelant, ou `-1` si aucun n'est trouvé.

`Array.prototype.map()`
Renvoie un nouveau tableau contenant les résultats de l'appel d'une fonction sur chaque élément du tableau appelant.

`Array.prototype.pop()`
Supprime le dernier élément d'un tableau et renvoie cet élément.

`Array.prototype.push()`
Ajoute un ou plusieurs éléments à la fin d'un tableau et renvoie le nouveau `length` du tableau.

`Array.prototype.reduce()`
Exécute une fonction de rappel "réducteur" fournie par l'utilisateur sur chaque élément du tableau (de gauche à droite), pour le réduire à une seule valeur.

`Array.prototype.reduceRight()`
Exécute une fonction de rappel "réducteur" fournie par l'utilisateur sur chaque élément du tableau (de droite à gauche), pour le réduire à une seule valeur.

`Array.prototype.reverse()`
Inverse l'ordre des éléments d'un tableau *en place* . (Le premier devient le dernier, le dernier devient le premier.)

`Array.prototype.shift()`
Supprime le premier élément d'un tableau et renvoie cet élément.

`Array.prototype.slice()`
Extrait une section du tableau appelant et renvoie un nouveau tableau.

`Array.prototype.some()`
Renvoie `true` si au moins un élément du tableau appelant satisfait la fonction de test fournie.

`Array.prototype.sort()`
Trie les éléments d'un tableau en place et renvoie le tableau.

`Array.prototype.splice()`
Ajoute et/ou supprime des éléments d'un tableau.

`Array.prototype.toLocaleString()`
Renvoie une chaîne localisée représentant le tableau appelant et ses éléments. Remplace la `Object.prototype.toLocaleString()` méthode.

`Array.prototype.toString()`
Renvoie une chaîne représentant le tableau appelant et ses éléments. Remplace la `Object.prototype.toString()` méthode.

`Array.prototype.unshift()`
Ajoute un ou plusieurs éléments au début d'un tableau et renvoie le nouveau `length` du tableau.

`Array.prototype.values()`
Renvoie un nouvel objet *itérateur de tableau* qui contient les valeurs de chaque index du tableau.

`Array.prototype[@@iterator]()`
Renvoie la `values()` fonction par défaut.

Exemples

Cette section fournit quelques exemples d'opérations courantes sur les tableaux en JavaScript.

> **Remarque :** si vous n'êtes pas encore familiarisé avec les bases des tableaux, pensez à lire d' abord JavaScript First Steps: Arrays , qui explique ce que sont les tableaux et comprend d'autres exemples d'opérations courantes sur les tableaux.

Créer un tableau

Cet exemple montre trois façons de créer un nouveau tableau : d'abord en utilisant la notation littérale de tableau , puis en utilisant le `Array()` constructeur et enfin en utilisant `String.prototype.split()` pour construire le tableau à partir d'une chaîne.

```
// 'fruits' array created using array literal notation.
const fruits = ['Apple', 'Banana'];
console.log(fruits.length);
// 2

// 'fruits' array created using the Array() constructor.
const fruits = new Array('Apple', 'Banana');
console.log(fruits.length);
// 2

// 'fruits' array created using String.prototype.split().
const fruits = 'Apple, Banana'.split(', ');
console.log(fruits.length);
```

```
// 2
```

Créer une chaîne à partir d'un tableau

Cet exemple utilise la `join()` méthode pour créer une chaîne à partir du `fruits` tableau.

```
const fruits = ['Apple', 'Banana'];
const fruitsString = fruits.join(', ');
console.log(fruitsString);
// "Apple, Banana"
```

Accéder à un élément du tableau par son index

Cet exemple montre comment accéder aux éléments du `fruits` tableau en spécifiant le numéro d'index de leur position dans le tableau.

```
const fruits = ['Apple', 'Banana'];

// The index of an array's first element is always 0.
fruits[0]; // Apple

// The index of an array's second element is always 1.
fruits[1]; // Banana

// The index of an array's last element is always one
// less than the length of the array.
fruits[fruits.length - 1]; // Banana

// Using a index number larger than the array's length
// returns 'undefined'.
fruits[99]; // undefined
```

Trouver l'index d'un élément dans un tableau

Cet exemple utilise la `indexOf()` méthode pour trouver la position (index) de la chaîne " `Banana` " dans le `fruits` tableau.

```
const fruits = ['Apple', 'Banana'];
console.log(fruits.indexOf('Banana'));

// 1
```

Vérifier si un tableau contient un certain élément

Cet exemple montre deux façons de vérifier si le `fruits` tableau contient " `Banana` " et " `Cherry` " : d'abord avec la `includes()` méthode, puis avec la `indexOf()` méthode pour tester

une valeur d'index qui n'est pas `-1`.

```
const fruits = ['Apple', 'Banana'];

fruits.includes('Banana'); // true
fruits.includes('Cherry'); // false

// If indexOf() doesn't return -1, the array contains the given item.
fruits.indexOf('Banana') !== -1; // true
fruits.indexOf('Cherry') !== -1; // false
```

Ajouter un élément à un tableau

Cet exemple utilise la `push()` méthode pour ajouter une nouvelle chaîne au `fruits` tableau.

```
const fruits = ['Apple', 'Banana'];
const newLength = fruits.push('Orange');
console.log(fruits);
// ["Apple", "Banana", "Orange"]
console.log(newLength);
// 3
```

Supprimer le dernier élément d'un tableau

Cet exemple utilise la `pop()` méthode pour supprimer le dernier élément du `fruits` tableau.

```
const fruits = ['Apple', 'Banana', 'Orange'];
const removedItem = fruits.pop();
console.log(fruits);
// ["Apple", "Banana"]
console.log(removedItem);
// Orange
```

> **Remarque :** `pop()` ne peut être utilisé que pour supprimer le dernier élément d'un tableau. Pour supprimer plusieurs éléments à la fin d'un tableau, consultez l'exemple suivant.

Supprimer plusieurs éléments de la fin d'un tableau

Cet exemple utilise la `splice()` méthode pour supprimer les 3 derniers éléments du `fruits` tableau.

```
const fruits = ['Apple', 'Banana', 'Strawberry', 'Mango', 'Cherry'];
```

```
const start = -3;
const removedItems = fruits.splice(start);
console.log(fruits);
// ["Apple", "Banana"]
console.log(removedItems);
// ["Strawberry", "Mango", "Cherry"]
```

Tronquer un tableau à ses *N* premiers éléments

Cet exemple utilise la `splice()` méthode pour tronquer le `fruits` tableau à ses 2 premiers éléments uniquement.

```
const fruits = ['Apple', 'Banana', 'Strawberry', 'Mango', 'Cherry'];
const start = 2;
const removedItems = fruits.splice(start);
console.log(fruits);
// ["Apple", "Banana"]
console.log(removedItems);
// ["Strawberry", "Mango", "Cherry"]
```

Supprimer le premier élément d'un tableau

Cet exemple utilise la `shift()` méthode pour supprimer le premier élément du `fruits` tableau.

```
const fruits = ['Apple', 'Banana'];
const removedItem = fruits.shift();
console.log(fruits);
// ["Banana"]
console.log(removedItem);
// Apple
```

> **Remarque :** `shift()` ne peut être utilisé que pour supprimer le premier élément d'un tableau. Pour supprimer plusieurs éléments du début d'un tableau, consultez l'exemple suivant.

Supprimer plusieurs éléments du début d'un tableau

Cet exemple utilise la `splice()` méthode pour supprimer les 3 premiers éléments du `fruits` tableau.

```
const fruits = ['Apple', 'Strawberry', 'Cherry', 'Banana', 'Mango'];
const start = 0;
const deleteCount = 3;
```

```
const removedItems = fruits.splice(start, deleteCount);
console.log(fruits);
// ["Banana", "Mango"]
console.log(removedItems);
// ["Apple", "Strawberry", "Cherry"]
```

Ajouter un nouveau premier élément à un tableau

Cet exemple utilise la `unshift()` méthode pour ajouter, à index `0`, un nouvel élément au `fruits` tableau, ce qui en fait le nouveau premier élément du tableau.

```
const fruits = ['Banana', 'Mango'];
const newLength = fruits.unshift('Strawberry');
console.log(fruits);
// ["Strawberry", "Banana", "Mango"]
console.log(newLength);
// 2
```

Supprimer un seul élément par index

Cet exemple utilise la `splice()` méthode pour supprimer la chaîne " `Banana` " du `fruits` tableau — en spécifiant la position d'index de " `Banana` ".

```
const fruits = ['Strawberry', 'Banana', 'Mango'];
const start = fruits.indexOf('Banana');
const deleteCount = 1;
const removedItems = fruits.splice(start, deleteCount);
console.log(fruits);
// ["Strawberry", "Mango"]
console.log(removedItems);
// ["Banana"]
```

Supprimer plusieurs éléments par index

Cet exemple utilise la `splice()` méthode pour supprimer les chaînes " `Banana` " et " `Strawberry` " du `fruits` tableau - en spécifiant la position d'index de " `Banana` ", ainsi qu'un décompte du nombre total d'éléments à supprimer.

```
const fruits = ['Apple', 'Banana', 'Strawberry', 'Mango'];
const start = 1;
const deleteCount = 2;
const removedItems = fruits.splice(start, deleteCount);
console.log(fruits);
// ["Apple", "Mango"]
```

```
console.log(removedItems);
// ["Banana", "Strawberry"]
```

Remplacer plusieurs éléments dans un tableau

Cet exemple utilise la `splice()` méthode pour remplacer les 2 derniers éléments du `fruits` tableau par de nouveaux éléments.

```
const fruits = ['Apple', 'Banana', 'Strawberry'];
const start = -2;
const deleteCount = 2;
const removedItems = fruits.splice(start, deleteCount, 'Mango', 'Cherry');
console.log(fruits);
// ["Apple", "Mango", "Cherry"]
console.log(removedItems);
// ["Banana", "Strawberry"]
```

Itérer sur un tableau

Cet exemple utilise une `for...of` boucle pour itérer sur le `fruits` tableau, en enregistrant chaque élément dans la console.

```
const fruits = ['Apple', 'Mango', 'Cherry'];
for (const fruit of fruits) {
  console.log(fruit);
}
// Apple
// Mango
// Cherry
```

Mais `for...of` ce n'est qu'une des nombreuses façons d'itérer sur n'importe quel tableau ; pour plus de façons, voir Boucles et itération , et voir la documentation des méthodes `every()` , `filter()` , `flatMap()` , `map()` , `reduce()` et `reduceRight()` — et voir l'exemple suivant, qui utilise la `forEach()` méthode.

Appeler une fonction sur chaque élément d'un tableau

Cet exemple utilise la `forEach()` méthode pour appeler une fonction sur chaque élément du `fruits` tableau ; la fonction entraîne l'enregistrement de chaque élément dans la console, ainsi que le numéro d'index de l'élément.

```
const fruits = ['Apple', 'Mango', 'Cherry'];
fruits.forEach(function(item, index, array) {
  console.log(item, index);
});
```

```
// Apple 0
// Mango 1
// Cherry 2
```

Fusionner plusieurs baies ensemble

Cet exemple utilise la `concat()` méthode pour fusionner le `fruits` tableau avec un `moreFruits` tableau, pour produire un nouveau `combinedFruits` tableau. Notez cela `fruits` et `moreFruits` restez inchangé.

```
const fruits = ['Apple', 'Banana', 'Strawberry'];
const moreFruits = ['Mango', 'Cherry'];
const combinedFruits = fruits.concat(moreFruits);
console.log(combinedFruits);
// ["Apple", "Banana", "Strawberry", "Mango", "Cherry"]

// The 'fruits' array remains unchanged.
console.log(fruits);
// ["Apple", "Banana", "Strawberry"]

// The 'moreFruits' array also remains unchanged.
console.log(moreFruits);
// ["Mango", "Cherry"]
```

Copier un tableau

Cet exemple montre trois manières de créer un nouveau tableau à partir du `fruits` tableau existant : d'abord en utilisant la syntaxe spread , puis en utilisant la `from()` méthode , puis en utilisant la `slice()` méthode .

```
const fruits = ['Strawberry', 'Mango'];

// Create a copy using spread syntax.
const fruitsCopy = [...fruits];
// ["Strawberry", "Mango"]

// Create a copy using the from() method.
const fruitsCopy = Array.from(fruits);
// ["Strawberry", "Mango"]


// Create a copy using the slice() method.
const fruitsCopy = fruits.slice();
// ["Strawberry", "Mango"]
```

Toutes les opérations de copie de tableau intégrées (syntaxe d'étalement , `Array.from()` ,

`Array.prototype.slice()` et `Array.prototype.concat()`) créent des copies superficielles . Si vous souhaitez plutôt une copie complète d'un tableau, vous pouvez utiliser `JSON.stringify()` pour convertir le tableau en une chaîne JSON, puis `JSON.parse()` reconvertir la chaîne en un nouveau tableau complètement indépendant du tableau d'origine.

```
const fruitsDeepCopy = JSON.parse(JSON.stringify(fruits));
```

Vous pouvez également créer des copies complètes à l'aide de la `structuredClone()` méthode, qui a l'avantage de permettre aux objets transférables de la source d'être *transférés* vers la nouvelle copie, plutôt que simplement clonés.

Enfin, il est important de comprendre que l'affectation d'un tableau existant à une nouvelle variable ne crée pas de copie du tableau ou de ses éléments. Au lieu de cela, la nouvelle variable est simplement une référence, ou un alias, au tableau d'origine ; c'est-à-dire que le nom du tableau d'origine et le nouveau nom de la variable ne sont que deux noms pour exactement le même objet (et seront donc toujours évalués comme strictement équivalents). Par conséquent, si vous apportez la moindre modification à la valeur du tableau d'origine ou à la valeur de la nouvelle variable, l'autre changera également :

```
const fruits = ['Strawberry', 'Mango'];
const fruitsAlias = fruits;
// 'fruits' and 'fruitsAlias' are the same object, strictly equivalent.
fruits === fruitsAlias // true
// Any changes to the 'fruits' array change 'fruitsAlias' too.
fruits.unshift('Apple', 'Banana');
console.log(fruits);
// ['Apple', 'Banana', 'Strawberry', 'Mango']
console.log(fruitsAlias);
// ['Apple', 'Banana', 'Strawberry', 'Mango']
```

Autres exemples

Création d'un tableau à deux dimensions

Ce qui suit crée un échiquier sous la forme d'un tableau de chaînes à deux dimensions. Le premier mouvement est effectué en copiant le `'p'` in `board[6][4]` dans `board[4][4]`. L'ancienne position à `[6][4]` est rendue vide.

```
const board = [
  ['R','N','B','Q','K','B','N','R'],
  ['P','P','P','P','P','P','P','P'],
  [' ',' ',' ',' ',' ',' ',' ',' '],
  [' ',' ',' ',' ',' ',' ',' ',' '],
  [' ',' ',' ',' ',' ',' ',' ',' '],
```

```
  [' ',' ',' ',' ',' ',' ',' ',' '],
  ['p','p','p','p','p','p','p','p'],
  ['r','n','b','q','k','b','n','r'] ];

console.log(board.join('\n') + '\n\n');

// Move King's Pawn forward 2
board[4][4] = board[6][4];
board[6][4] = ' ';
console.log(board.join('\n'));
```

Voici la sortie :

```
R,N,B,Q,K,B,N,R
P,P,P,P,P,P,P,P
 , , , , , , ,
 , , , , , , ,
 , , , , , , ,
 , , , , , , ,
p,p,p,p,p,p,p,p
r,n,b,q,k,b,n,r

R,N,B,Q,K,B,N,R
P,P,P,P,P,P,P,P
 , , , , , , ,
 , , , , , , ,
 , , , ,p, , ,
 , , , , , , ,
p,p,p,p, ,p,p,p
r,n,b,q,k,b,n,r
```

Utilisation d'un tableau pour tabuler un ensemble de valeurs

```
const values = [];
for (let x = 0; x < 10; x++) {
  values.push([
    2 ** x,
    2 * x ** 2,
  ]);

}
console.table(values);
```

Résulte en

```
// The first column is the index
```

```
0  1    0
1  2    2
2  4    8
3  8    18
4  16   32
5  32   50
6  64   72
7  128  98
8  256  128
9  512  162
```

Remarques

`Array` les objets ne peuvent pas utiliser de chaînes comme index d'éléments (comme dans un tableau associatif) mais doit utiliser des entiers. La définition ou l'accès via des nombres non entiers à l'aide de la notation par crochets (ou notation par points) ne définira ni ne récupèrera un élément de la liste de tableaux elle-même, mais définira ou accédera à une variable associée à la collection de propriétés d'objet de ce tableau . Les propriétés d'objet du tableau et la liste des éléments du tableau sont distinctes, et les opérations de parcours et de mutation du tableau ne peuvent pas être appliquées à ces propriétés nommées.

Les éléments de tableau sont des propriétés d'objet de la même manière qu'une `toString` propriété (pour être précis, cependant, `toString()` c'est une méthode). Néanmoins, essayer d'accéder à un élément d'un tableau comme suit génère une erreur de syntaxe car le nom de la propriété n'est pas valide :

```
console.log(arr.0); // a syntax error
```

Il n'y a rien de spécial dans les tableaux JavaScript et les propriétés qui en sont la cause. Les propriétés JavaScript qui commencent par un chiffre ne peuvent pas être référencées avec la notation par points et doivent être accessibles à l'aide de la notation par crochets.

Par exemple, si vous aviez un objet avec une propriété nommée `3d` , il ne peut être référencé qu'à l'aide d'une notation entre parenthèses.

```
const years = [1950, 1960, 1970, 1980, 1990, 2000, 2010];
console.log(years.0);   // a syntax error
console.log(years[0]);  // works properly
```

```
renderer.3d.setTexture(model, 'character.png');     // a syntax error
renderer['3d'].setTexture(model, 'character.png');  // works properly
```

Dans l' `3d` exemple, `'3d'` *devait* être entre guillemets (car il commence par un chiffre). Mais il est également possible de citer également les index du tableau (par exemple, `years['2']` au lieu de `years[2]`), bien que ce ne soit pas nécessaire.

Le `2` in `years[2]` est converti en chaîne par le moteur JavaScript via une `toString` conversion implicite. Par conséquent, `'2'` et `'02'` ferait référence à deux emplacements différents sur l' `years` objet, et l'exemple suivant pourrait être `true` :

```
console.log(years['2'] != years['02']);
```

Relation entre la longueur et les propriétés numériques

`length` La propriété et les propriétés numériques d'un tableau JavaScript sont connectées.

Plusieurs des méthodes de tableau intégrées (par exemple, `join()`, `slice()`, `indexOf()`, etc.) prennent en compte la valeur de la `length` propriété d'un tableau lorsqu'elles sont appelées.

D'autres méthodes (par exemple, `push()`, `splice()`, etc.) entraînent également des mises à jour de la propriété d'un tableau `length`.

```
const fruits = [];
fruits.push('banana', 'apple', 'peach');
console.log(fruits.length); // 3
```

Lors de la définition d'une propriété sur un tableau JavaScript lorsque la propriété est un index de tableau valide et que cet index est en dehors des limites actuelles du tableau, le moteur mettra à jour la `length` propriété du tableau en conséquence :

```
fruits[5] = 'mango';
console.log(fruits[5]);           // 'mango'
console.log(Object.keys(fruits)); // ['0', '1', '2', '5']
console.log(fruits.length);       // 6
```

Augmenter le `length`.

```
fruits.length = 10;
console.log(fruits);              // ['banana', 'apple', 'peach', empty
console.log(Object.keys(fruits)); // ['0', '1', '2', '5']
console.log(fruits.length);       // 10
console.log(fruits[8]);           // undefined
```

Cependant, la diminution de la `length` propriété supprime des éléments.

```
fruits.length = 2;
console.log(Object.keys(fruits)); // ['0', '1']
```

```
console.log(fruits.length);       // 2
```

Ceci est expliqué plus loin sur la `Array.length` page.

Créer un tableau en utilisant le résultat d'une correspondance

Le résultat d'une correspondance entre a `RegExp` et une chaîne peut créer un tableau JavaScript qui a des propriétés et des éléments qui fournissent des informations sur la correspondance. Un tel tableau est retourné par `RegExp.exec()` et `String.match()`.

Pour vous aider à expliquer ces propriétés et éléments, consultez l'exemple suivant, puis reportez-vous au tableau ci-dessous :

```
// Match one d followed by one or more b's followed by one d
// Remember matched b's and the following d
// Ignore case

const myRe = /d(b+)(d)/i;
const myArray = myRe.exec('cdbBdbsbz');
```

Les propriétés et les éléments renvoyés par cette correspondance sont les suivants :

Propriété/élément	La description	Exemple
`input` Lecture seulement	La chaîne d'origine à laquelle l'expression régulière a été comparée.	`"cdbBdbsbz"`
`index` Lecture seulement	Index de base zéro de la correspondance dans la chaîne.	`1`
`[0]` Lecture seulement	Les derniers caractères correspondants.	`"dbBd"`
`[1], ...[n]` Lecture seulement	Éléments qui spécifient les correspondances de sous-chaîne entre parenthèses (si elles sont incluses) dans l'expression régulière. Le nombre de sous-chaînes entre parenthèses possibles est illimité.	`[1]: "bB"` `[2]: "d"`

parentheses possibles est illimité.

Propriété/élément	La description	Exemple

Caractéristiques

spécification
Spécification du langage ECMAScript # sec-array-objects

Compatibilité du navigateur

Signaler les problèmes avec ces données de compatibilité sur GitHub

`Array`	
Chrome	1
Bord	12
Firefox	1
Internet Explorer	4
Opéra	4
Safari	1
WebAfficher Android	37
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`Array()` constructeur	
Chrome	1
Bord	12
Firefox	1

Internet Explorer	4
Opéra	4
Safari	1
WebAfficher Android	37
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`at`

Chrome	92
Bord	92
Firefox	90
Internet Explorer	**Non**
Opéra	78
Safari	15.4
WebAfficher Android	92
ChromeAndroid	92
Firefox pour Androïd	90
Opéra Android	**Non**
Safari sur iOS	15.4
Internet Samsung	16.0
Déno	1.12
Node.js	16.6.0

`concat`

Chrome	1
Bord	12

Firefox	1
Internet Explorer	5.5
Opéra	4
Safari	1
WebAfficher Android	1
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`copyWithin`

Chrome	45
Bord	12
Firefox	32
Internet Explorer	**Non**
Opéra	32
Safari	9
WebAfficher Android	45
ChromeAndroid	45
Firefox pour Androïd	32
Opéra Android	32
Safari sur iOS	9
Internet Samsung	5.0
Déno	1.0
Node.js	4.0.0

`entries`

Chrome	38

Bord	12
Firefox	28
Internet Explorer	**Non**
Opéra	25
Safari	8
WebAfficher Android	38
ChromeAndroid	38
Firefox pour Androïd	28
Opéra Android	25
Safari sur iOS	8
Internet Samsung	3.0
Déno	1.0
Node.js	0.12.0

`every`

Chrome	1
Bord	12
Firefox	1.5
Internet Explorer	9
Opéra	9.5
Safari	3
WebAfficher Android	37
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`fill`

Chrome	45
Bord	12
Firefox	31
Internet Explorer	**Non**
Opéra	32
Safari	8
WebAfficher Android	45
ChromeAndroid	45
Firefox pour Androïd	31
Opéra Android	32
Safari sur iOS	8
Internet Samsung	5.0
Déno	1.0
Node.js	4.0.0

`filter`

Chrome	1
Bord	12
Firefox	1.5
Internet Explorer	9
Opéra	9.5
Safari	3
WebAfficher Android	37
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`find`	
Chrome	45
Bord	12
Firefox	25
Internet Explorer	**Non**
Opéra	32
Safari	8
WebAfficher Android	45
ChromeAndroid	45
Firefox pour Androïd	4
Opéra Android	32
Safari sur iOS	8
Internet Samsung	5.0
Déno	1.0
Node.js	4.0.0

`findIndex`	
Chrome	45
Bord	12
Firefox	25
Internet Explorer	**Non**
Opéra	32
Safari	8
WebAfficher Android	45
ChromeAndroid	45
Firefox pour Androïd	4
Opéra Android	32
Safari sur iOS	8
Internet Samsung	5.0
Déno	1.0

Node.js	4.0.0
`findLast`	
Chrome	97
Bord	97
Firefox	**Non**
Internet Explorer	**Non**
Opéra	83
Safari	15.4
WebAfficher Android	97
ChromeAndroid	97
Firefox pour Androïd	**Non**
Opéra Android	**Non**
Safari sur iOS	15.4
Internet Samsung	**Non**
Déno	1.16
Node.js	**Non**
`findLastIndex`	
Chrome	97
Bord	97
Firefox	**Non**
Internet Explorer	**Non**
Opéra	83
Safari	15.4
WebAfficher Android	97
ChromeAndroid	97
Firefox pour Androïd	**Non**
Opéra Android	**Non**
Safari sur iOS	15.4
Internet Samsung	**Non**

Déno	1.16
Node.js	**Non**

`flat`	
Chrome	69
Bord	79
Firefox	62
Internet Explorer	**Non**
Opéra	56
Safari	12
WebAfficher Android	69
ChromeAndroid	69
Firefox pour Androïd	62
Opéra Android	48
Safari sur iOS	12
Internet Samsung	10.0
Déno	1.0
Node.js	11.0.0

`flatMap`	
Chrome	69
Bord	79
Firefox	62
Internet Explorer	**Non**
Opéra	56
Safari	12
WebAfficher Android	69
ChromeAndroid	69
Firefox pour Androïd	62
Opéra Android	48
Safari sur iOS	12

Internet Samsung	10.0
Déno	1.0
Node.js	11.0.0

`forEach`

Chrome	1
Bord	12
Firefox	1.5
Internet Explorer	9
Opéra	9.5
Safari	3
WebAfficher Android	37
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`from`

Chrome	45
Bord	12
Firefox	32
Internet Explorer	**Non**
Opéra	32
Safari	9
WebAfficher Android	45
ChromeAndroid	45
Firefox pour Androïd	32
Opéra Android	32

Safari sur iOS	9
Internet Samsung	5.0
Déno	1.0
Node.js	4.0.0

`groupBy`

Chrome	**Non**
Bord	**Non**
Firefox	Chaque nuit
Internet Explorer	**Non**
Opéra	**Non**
Safari	**Non**
WebAfficher Android	**Non**
ChromeAndroid	**Non**
Firefox pour Androïd	**Non**
Opéra Android	**Non**
Safari sur iOS	**Non**
Internet Samsung	**Non**
Déno	**Non**
Node.js	**Non**

`groupByToMap`

Chrome	**Non**
Bord	**Non**
Firefox	Chaque nuit
Internet Explorer	**Non**
Opéra	**Non**
Safari	**Non**
WebAfficher Android	**Non**
ChromeAndroid	**Non**
Firefox pour Androïd	**Non**

Opéra Android	**Non**
Safari sur iOS	**Non**
Internet Samsung	**Non**
Déno	**Non**
Node.js	**Non**

`includes`	
Chrome	47
Bord	14
Firefox	43
Internet Explorer	**Non**
Opéra	34
Safari	9
WebAfficher Android	47
ChromeAndroid	47
Firefox pour Androïd	43
Opéra Android	34
Safari sur iOS	9
Internet Samsung	5.0
Déno	1.0
Node.js	6.0.0

`indexOf`	
Chrome	1
Bord	12
Firefox	1.5
Internet Explorer	9
Opéra	9.5
Safari	3
WebAfficher Android	37
ChromeAndroid	18

Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`isArray`

Chrome	5
Bord	12
Firefox	4
Internet Explorer	9
Opéra	10.5
Safari	5
WebAfficher Android	1
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	14
Safari sur iOS	5
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`join`

Chrome	1
Bord	12
Firefox	1
Internet Explorer	5.5
Opéra	4
Safari	1
WebAfficher Android	1

ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`keys`

Chrome	38
Bord	12
Firefox	28
Internet Explorer	**Non**
Opéra	25
Safari	8
WebAfficher Android	38
ChromeAndroid	38
Firefox pour Androïd	28
Opéra Android	25
Safari sur iOS	8
Internet Samsung	3.0
Déno	1.0
Node.js	0.12.0

`lastIndexOf`

Chrome	1
Bord	12
Firefox	1.5
Internet Explorer	9
Opéra	9.5
Safari	3

WebAfficher Android	37
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`length`

Chrome	1
Bord	12
Firefox	1
Internet Explorer	4
Opéra	4
Safari	1
WebAfficher Android	37
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`map`

Chrome	1
Bord	12
Firefox	1.5
Internet Explorer	9
Opéra	9.5

Safari	3
WebAfficher Android	37
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

of

Chrome	45
Bord	12
Firefox	25
Internet Explorer	**Non**
Opéra	26
Safari	9
WebAfficher Android	39
ChromeAndroid	39
Firefox pour Androïd	25
Opéra Android	26
Safari sur iOS	9
Internet Samsung	4.0
Déno	1.0
Node.js	4.0.0

pop

Chrome	1
Bord	12
Firefox	1
Internet Explorer	5.5

Opéra	4
Safari	1
WebAfficher Android	1
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`push`

Chrome	1
Bord	12
Firefox	1
Internet Explorer	5.5
Opéra	4
Safari	1
WebAfficher Android	1
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`reduce`

Chrome	3
Bord	12
Firefox	3

Internet Explorer	9
Opéra	10.5
Safari	5
WebAfficher Android	37
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	14
Safari sur iOS	4
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`reduceRight`

Chrome	3
Bord	12
Firefox	3
Internet Explorer	9
Opéra	10.5
Safari	5
WebAfficher Android	37
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	14
Safari sur iOS	4
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`reverse`

Chrome	1
Bord	12

Firefox	1
Internet Explorer	5.5
Opéra	4
Safari	1
WebAfficher Android	1
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`shift`

Chrome	1
Bord	12
Firefox	1
Internet Explorer	5.5
Opéra	4
Safari	1
WebAfficher Android	1
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`slice`

Chrome	1

Bord	12
Firefox	1
Internet Explorer	4
Opéra	4
Safari	1
WebAfficher Android	1
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`some`

Chrome	1
Bord	12
Firefox	1.5
Internet Explorer	9
Opéra	9.5
Safari	3
WebAfficher Android	37
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`sort`

Chrome	1
Bord	12
Firefox	1
Internet Explorer	5.5
Opéra	4
Safari	1
WebAfficher Android	1
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0
Tri stable	
Chrome	70
Bord	79
Firefox	3
Internet Explorer	**Non**
Opéra	57
Safari	10.1
WebAfficher Android	70
ChromeAndroid	70
Firefox pour Androïd	4
Opéra Android	49
Safari sur iOS	10.3
Internet Samsung	10.0
Déno	1.0
Node.js	12.0.0

`splice`	
Chrome	1
Bord	12
Firefox	1
Internet Explorer	5.5
Opéra	4
Safari	1
WebAfficher Android	1
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`toLocaleString`	
Chrome	1
Bord	12
Firefox	1
Internet Explorer	5.5
Opéra	4
Safari	1
WebAfficher Android	37
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0

Node.js	0.10.0
`locales` paramètre	
Chrome	24
Bord	79
Firefox	52
Internet Explorer	**Non**
Opéra	15
Safari	7
WebAfficher Android	37
ChromeAndroid	25
Firefox pour Androïd	56
Opéra Android	14
Safari sur iOS	7
Internet Samsung	2.0
Déno	1.8
Node.js	13.0.0
`options` paramètre	
Chrome	24
Bord	79
Firefox	52
Internet Explorer	**Non**
Opéra	15
Safari	7
WebAfficher Android	37
ChromeAndroid	25
Firefox pour Androïd	56
Opéra Android	14
Safari sur iOS	7
Internet Samsung	2.0

Déno	1.0
Node.js	0.12.0

`toSource`

Chrome	**Non**
Bord	**Non**
Firefox	**1 – 74**
Internet Explorer	**Non**
Opéra	**Non**
Safari	**Non**
WebAfficher Android	**Non**
ChromeAndroid	**Non**
Firefox pour Androïd	4
Opéra Android	**Non**
Safari sur iOS	**Non**
Internet Samsung	**Non**
Déno	**Non**
Node.js	**Non**

`toString`

Chrome	1
Bord	12
Firefox	1
Internet Explorer	4
Opéra	4
Safari	1
WebAfficher Android	37
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1

Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`unshift`

Chrome	1
Bord	12
Firefox	1
Internet Explorer	5.5
Opéra	4
Safari	1
WebAfficher Android	1
ChromeAndroid	18
Firefox pour Androïd	4
Opéra Android	10.1
Safari sur iOS	1
Internet Samsung	1.0
Déno	1.0
Node.js	0.10.0

`values`

Chrome	66
Bord	12
Firefox	60
Internet Explorer	**Non**
Opéra	53
Safari	9
WebAfficher Android	66
ChromeAndroid	66
Firefox pour Androïd	60
Opéra Android	47

Safari sur iOS	9
Internet Samsung	9.0
Déno	1.0
Node.js	10.9.0

`@@iterator`	
Chrome	38
Bord	12
Firefox	36
Internet Explorer	**Non**
Opéra	25
Safari	dix
WebAfficher Android	38
ChromeAndroid	38
Firefox pour Androïd	36
Opéra Android	25
Safari sur iOS	dix
Internet Samsung	3.0
Déno	1.0
Node.js	0.12.0

`@@species`	
Chrome	51
Bord	79
Firefox	48
Internet Explorer	**Non**
Opéra	38
Safari	dix
WebAfficher Android	51
ChromeAndroid	51
Firefox pour Androïd	48

Opéra Android	41
Safari sur iOS	dix
Internet Samsung	5.0
Déno	1.0
Node.js	6.5.0

`@@unscopables`

Chrome	38
Bord	12
Firefox	48
Internet Explorer	**Non**
Opéra	25
Safari	dix
WebAfficher Android	38
ChromeAndroid	38
Firefox pour Androïd	48
Opéra Android	25
Safari sur iOS	dix
Internet Samsung	3.0
Déno	1.0
Node.js	0.12.0

Plein soutien

Prise en charge partielle

En développement. Pris en charge dans une version préliminaire.

Pas de support

Expérimental. Attendez-vous à ce que le comportement change à l'avenir.

Non standard. Vérifiez la prise en charge de plusieurs navigateurs avant de l'utiliser.

Voir les notes d'implémentation.

L'utilisateur doit explicitement activer cette fonctionnalité.

Utilise un nom non standard.

Voir également

- À partir du guide JavaScript :
  - "Indexation des propriétés de l'objet"
  - "Collections indexées : `Array` objet"
- Tableaux typés
- RangeError : longueur de tableau non valide

Dernière modification : 20 février 2022, par les contributeurs MDN