Nombre

`Number` est un [objet wrapper primitif](#) utilisé pour représenter et manipuler des nombres comme `37` ou `-9.25`.

Le `Number` constructeur contient des constantes et des méthodes pour travailler avec des nombres. Les valeurs d'autres types peuvent être converties en nombres à l'aide de la `Number()` fonction.

Le type JavaScript `Number` est un[format binaire double précision 64 bits IEEE 754](#) value, comme `double` en Java ou C#. Cela signifie qu'il peut représenter des valeurs fractionnaires, mais il y a des limites à ce qu'il peut stocker. A `Number` ne conserve qu'environ 17 décimales de précision; l'arithmétique est soumise à[arrondir](#). La plus grande valeur que `Number` peut contenir a est d'environ 1,8E308. Les valeurs supérieures sont remplacées par la `Number` constante spéciale `Infinity`.

Un nombre littéral comme `37` dans le code JavaScript est une valeur à virgule flottante, pas un entier. Il n'y a pas de type entier séparé dans l'usage quotidien courant. (JavaScript a maintenant un `BigInt` type, mais il n'a pas été conçu pour remplacer Number pour les utilisations quotidiennes. `37` est toujours un `Number`, pas un BigInt.)

`Number` peut également être exprimé sous des formes littérales telles que `0b101`, `0o13`, `0x0A`. [En savoir plus sur la grammaire lexicale](#) numérique ici.

La description

Lorsqu'il est utilisé en tant que fonction, `Number(value)` convertit une chaîne ou une autre valeur en type Number. Si la valeur ne peut pas être convertie, elle renvoie `NaN`.

Syntaxe littérale

```
123    // one-hundred twenty-three
123.0  // same
123 === 123.0  // true
```

Syntaxe de la fonction

```
Number('123')  // returns the number 123
```

```
Number('123') === 123  // true

Number("unicorn")  // NaN
Number(undefined)  // NaN
```

Constructeur

`Number()`
Crée une nouvelle `Number` valeur.

Propriétés statiques

`Number.EPSILON`
Le plus petit intervalle entre deux nombres représentables.

`Number.MAX_SAFE_INTEGER`
L'entier sûr maximum en JavaScript (`2^53 - 1`).

`Number.MAX_VALUE`
Le plus grand nombre représentable positif.

`Number.MIN_SAFE_INTEGER`
L'entier sûr minimum en JavaScript (`-(2^53 - 1)`).

`Number.MIN_VALUE`
Le plus petit nombre représentable positif, c'est-à-dire le nombre positif le plus proche de zéro (sans être réellement zéro).

`Number.NaN`
Valeur spéciale " Pas **un numéro** ".

`Number.NEGATIVE_INFINITY`
Valeur spéciale représentant l'infini négatif. Renvoyé sur débordement.

`Number.POSITIVE_INFINITY`
Valeur spéciale représentant l'infini. Renvoyé sur débordement.

`Number.prototype`
Permet d'ajouter des propriétés à l' `Number` objet.

Méthodes statiques

`Number.isNaN()`
Déterminez si la valeur transmise est `NaN` .

`Number.isFinite()`
Déterminez si la valeur transmise est un nombre fini.

`Number.isInteger()`
Déterminez si la valeur transmise est un entier.

`Number.isSafeInteger()`
Déterminez si la valeur transmise est un entier sûr (nombre compris entre `-(2^53 - 1)` et `2^53 - 1`).

`Number.parseFloat(string)`
C'est la même chose que la `parseFloat()` fonction globale.

`Number.parseInt(string, [radix])`
C'est la même chose que la `parseInt()` fonction globale.

Méthodes d'instance

`Number.prototype.toExponential(fractionDigits)`
Renvoie une chaîne représentant le nombre en notation exponentielle.

`Number.prototype.toFixed(digits)`
Renvoie une chaîne représentant le nombre en notation à virgule fixe.

`Number.prototype.toLocaleString([locales [, options]])`
Renvoie une chaîne avec une représentation sensible à la langue de ce nombre. Remplace la `Object.prototype.toLocaleString()` méthode.

`Number.prototype.toPrecision(precision)`
Renvoie une chaîne représentant le nombre avec une précision spécifiée en notation à virgule fixe ou exponentielle.

`Number.prototype.toString([radix])`
Renvoie une chaîne représentant l'objet spécifié dans la base spécifiée *(* "base"). Remplace la `Object.prototype.toString()` méthode.

`Number.prototype.valueOf()`
Renvoie la valeur primitive de l'objet spécifié. Remplace la `Object.prototype.valueOf()` méthode.

Exemples

Utilisation de l'objet Number pour affecter des valeurs à des variables numériques

L'exemple suivant utilise les `Number` propriétés de l'objet pour affecter des valeurs à plusieurs variables numériques :

```
const biggestNum     = Number.MAX_VALUE
const smallestNum    = Number.MIN_VALUE
const infiniteNum    = Number.POSITIVE_INFINITY
const negInfiniteNum = Number.NEGATIVE_INFINITY
const notANum        = Number.NaN
```

Plage d'entiers pour Number

L'exemple suivant montre les valeurs entières minimales et maximales qui peuvent être représentées en tant `Number` qu'objet. (Plus de détails à ce sujet sont décrits dans la norme ECMAScript, chapitre *6.1.6 Le type de numéro* .)

```
const biggestInt  = Number.MAX_SAFE_INTEGER  //  (2**53 - 1) =>  900719
const smallestInt = Number.MIN_SAFE_INTEGER  // -(2**53 - 1) => -900719
```

Lors de l'analyse de données qui ont été sérialisées en JSON, on peut s'attendre à ce que les valeurs entières se trouvant en dehors de cette plage soient corrompues lorsque l'analyseur JSON les force à `Number` taper.

Une solution de contournement possible consiste à utiliser à la `String` place.

Des nombres plus grands peuvent être représentés en utilisant le `BigInt` type.

Utilisation de Number pour convertir un objet Date

L'exemple suivant convertit l' `Date` objet en une valeur numérique en utilisant `Number` comme fonction :

```
let d = new Date('December 17, 1995 03:24:00')
console.log(Number(d))
```

Cela enregistre `819199440000` .

Convertir des chaînes numériques et null en nombres

```
Number('123')     // 123
Number('123') === 123 // true
Number('12.3')    // 12.3
Number('12.00')   // 12
Number('123e-1')  // 12.3

Number('')        // 0
Number(null)      // 0
Number('0x11')    // 17
Number('0b11')    // 3
Number('0o11')    // 9
Number('foo')     // NaN
Number('100a')    // NaN
Number('-Infinity') // -Infinity
```

Caractéristiques

spécification
Spécification du langage ECMAScript (ECMAScript) # sec-number-objects

Compatibilité du navigateur

Signaler les problèmes avec ces données de compatibilité sur GitHub

Number	
Chrome	1
Edge	12
Firefox	1
Internet Explorer	3
Opera	3
Safari	1
WebView Android	1
Chrome Android	18
Firefox for Android	4
Opera Android	10.1
Safari on iOS	1