JS : Nombres et dates

L2 MPCIE - UE Développement Web

David Lesaint

david.lesaint@univ-angers.fr

Janvier 2019

Les nombres

Implémentation en JS

Sur 64 bits à précision double au format IEEE-754

- Bit de poids fort pour le signe, puis 11 bits pour l'exposant, puis 52 bits pour la mantisse.

Pas de type spécifique pour les entiers.
Trois valeurs spéciales : `+Infinity`, `-Infinity`, `NaN`.

Quatre types de littéraux numériques

- Décimal.
- Binaire.
- Octal.
- Hexadécimal.

Littéraux numériques

nombres-litteraux.js

```
1  // Nombres décimaux
2  // !! analyse en base octale si 0 en début de nombre et mode non-strict
3  1234567980;
4  42;
5  // Nombres binaires : 0b ou 0B puis chiffres dans {0,1}
6  var x = 0b101; // 5
7  // Nombres octaux : 0 puis chiffres dans {0...7}
8  var n = 0755; // 493 en base 10
9  var m = 0644; // 420 en base 10
10 // Nombres hexadécimaux : 0x ou 0X puis caractères dans {0...9,A...F}
11 0xFFFFFFFFFFFFFFFF // 295147905179352830000
12 0x123456789ABCDEF  // 81985529216486900
13 0XA                // 10
14 // Notation scientifique
15 1E3   // 100
16 2e6   // 2000000
17 0.1e2 // 10
```

L'objet natif `Number`

Enveloppe objet (wrapper) autour du type primitif numérique

Permet de manipuler les nombres comme des objets.
Fournit constantes et méthodes pour :

- Convertir des chaînes.
- Tester type et valeur.
- Formatter des nombres sous forme de chaînes.

Pour des raisons de performances et de simplicité

Préférer les littéraux numériques à la construction d'objets `Number`.
Utiliser les méthodes de `Number` comme des méthodes "statiques" de classe.

L'objet natif `Number`

Propriétés	Description
`Number.MAX_VALUE`	Le plus grand nombre qu'on peut représenter en JS.
`Number.MIN_VALUE`	Le plus petit nombre qu'on peut représenter en JS.
`Number.NaN`	Une valeur spéciale signifiant que la valeur n'est pas un nombre.
`Number.NEGATIVE_INFINITY`	L'infini négatif, renvoyé lorsqu'on dépasse la valeur minimale.
`Number.POSITIVE_INFINITY`	L'infini positif, renvoyé lorsqu'on dépasse la valeur maximale.
`Number.EPSILON`	La différence entre 1 et la première valeur supérieure à 1 qui puisse être représentée comme `Number`.
`Number.MIN_SAFE_INTEGER`	Le plus petit entier qu'on puisse représenter en JS.
`Number.MAX_SAFE_INTEGER`	Le plus grand entier qu'on puisse représenter en JS.

L'objet natif `Number`

Méthodes	Description
`Number.parseFloat()`	Analyse une chaîne et renvoie un nombre décimal. Equivalente à `parseFloat()`.
`Number.parseInt()`	Analyse une chaîne et renvoie un entier exprimé dans une base donnée. Equivalente à `parseInt()`.
`Number.isFinite()`	Détermine si l'argument est un nombre fini.
`Number.isInteger()`	Détermine si l'argument est un nombre entier.
`Number.isNaN()`	Détermine si l'argument est `NaN`. Version plus robuste que `isNaN()`.
`Number.isSafeInteger()`	Détermine si l'argument est un nombre qu'il est possible de représenter comme un entier sans perdre d'information.
`toExponential()`	Renvoie une chaîne représentant le nombre en notation exponentielle.
`toFixed()`	Renvoie une chaîne représentant le nombre en notation à point fixe.
`toPrecision()`	Renvoie une chaîne représentant le nombre en notation à point fixe avec une précision donnée.

L'objet natif `Number`

nombres-conversions.js

```
1  var d = new Date('December 17, 1970 03:24:00');
2  console.log(Number(d) / (1000 * 60 * 60 * 24)); // 350.1
3
4  Number("123");                // 123
5  Number("12.3");               // 12.3
6  Number("");                   // 0
7  Number("0x11");               // 17
8  Number("0b11");               // 3
9  Number("0o11");               // 9
10 Number("toto");               // NaN
11 Number("100a");               // NaN
12 Number.parseInt("100a");   // 100
```

L'objet natif `Math`

Fournit constantes et méthodes de calcul

- Trigonométrie.
- Logarithmes, ...

On ne peut pas créer d'objets `Math`

Utiliser l'objet natif comme l'instance d'une classe "singleton".

L'objet natif `Math`

Méthodes	Description
`abs()`	Valeur absolue.
`sin(),cos(),tan()`	Fonctions trigonométriques (arguments en radians).
`asin(),acos(),atan(),atan2()`	Fonctions trig. inverses (valeurs renvoyées en radians).
`sinh(),cosh(),tanh()`	Fonctions trig. hyperboliques (arguments en radians).
`asinh(),acosh(),atanh()`	Fonctions trig. hyperboliques inverses (valeurs renvoyées en radians).
`pow(),exp(),expm1()` `log10(),log1p(),log2()`	Fonctions exponentielles et logarithmiques.
`floor(),ceil()`	Renvoie le plus petit/grand entier inférieur/supérieur ou égal à l'argument.
`min(),max()`	Renvoie le plus petit/grand nombre d'une liste de nombres séparés par des virgules.
`random()`	Renvoie un nombre aléatoire entre 0 et 1.
`round(),fround(),trunc()`	Fonctions d'arrondis et de troncature.
`sqrt(),cbrt(),hypot()`	Racine carrée, cubique et racine carrée de la somme des carrés des arguments.
`sign()`	Indique si la valeur est négative, positive ou nulle.
`clz32()`	Le nombre de zéros qui commencent un nombre sur 32 bits en représentation binaire.
`imul()`	La multiplication de deux arguments sur 32 bits effectuée comme en C.

Les dates

Manipulées à l'aide d'objets `Date`

Chaque objet `Date` stocke une durée en millisecondes (Epoch UNIX) relative au 1/1/1970 :00 :00 :00.

Construction de date avec `new Date([paramètres])`

Sans paramètres, l'objet représentera la date et l'heure courante.
Avec paramètres et selon le format, l'objet représentera la date indiquée :

- (année,mois,jour) : `new Date(2018,2,31).`
- (a,m,j,h,mn,s) : `new Date(2018,2,31,22,44,55).`
- (m,j,a h :mn :s") : `new Date("March 31, 2018 22:44:55").`

Objets `Date`

Fournissent accesseurs et mutateurs pour

- Définir et modifier la date.
- Obtenir des informations sur la date (l'année, le mois, ...).
- Convertir la date en différents formats.
- Convertir une chaîne en date.

nombres-dates.js

```
1 var today = new Date();
2 //Janvier = 0
3 var eoy = new Date(2019, 0, 1, 00, 00, 00, 000);
4 // Millisecondes par jour
5 var msPerDay = 24 * 60 * 60 * 1000;
6 // Jours restants dans l'année
7 var daysLeft = (eoy.getTime() - today.getTime()) /
msPerDay;
8 console.log(daysLeft = Math.round(daysLeft));
```