Les types primitifs

Number

Les types primitifs

Déclaration d'une variable de type « Number »

Le type « Number » permet de stocker un nombre réel.

- Exemple de déclaration de variable

```
const val1 = 42;
const val2 = 4_357;
const val3 = 9.99;
const val4 = 2e3;  // → 2000
```

Pour la lisibilité, le symbole « underscore » peut être utilisé comme séparateur.

Il est possible d'écrire un nombre sous la forme d'exposant décimal.
Exemple : 5e2 ⟶ 5 x 10^2

Convertir en nombre

Le JavaScript possède deux méthodes pour convertir une variable de type « String »

- parseInt(..., ...)
- parseFloat(...)

En cas d'échec de la conversion, ces méthodes renvoient le résultat « NaN »

Rappel :
Pour tester que la valeur est « NaN », il faut utiliser la méthode « isNaN(...) »

Convertir en nombre - Entier

La méthode « parseInt » permet d'obtenir la valeur entière d'un chaîne de caractères.

Celle-ci prend en paramètres :

- ➔ La chaîne de caractères à convertir
- ➔ La base (par défaut : 10)

```
const textNb = '42';
const nb1 = parseInt(textNb);      // → 42
const nb2 = parseInt(textNb, 10); // → 42
const nb3 = parseInt(textNb, 16); // → 66
const nb4 = parseInt('9.999');     // → 9

const textError = 'Dix';
const nb5 = parseInt(textError);  // → NaN
```

Convertir en nombre - Réel

La méthode « parseFloat » permet d'obtenir la valeur réel.

Celle-ci prend ne prend qu'un seul paramètre :

➔ La chaîne de caractères à convertir.

```
const textReal = '3.14159';
const nb6 = parseFloat(textReal);     // → 3.14159

const textBad = '1,2345';
const nb7 = parseFloat(textBad);      // → 1

const textError2 = 'Demo';
const nb8 = parseFloat(textError2);   // → NaN
```

Les opérateurs arithmétiques

Le langage JavaScript support les opérateurs arithmétiques suivants :

- Addition
- Soustraction
- Multiplication
- Division
- Modulo
- Puissance

```
const nb1 = 14;
const nb2 = 3;

// Addition
const r1 = nb1 + nb2;   // → 17

// Soustraction
const r2 = nb1 - nb2;   // → 11

// Multiplication
const r3 = nb1 * nb2;   // → 42

// Division
const r4 = nb1 / nb2;   // → 4,66666...

// Modulo (Reste d'une division entière)
const r5 = nb1 % nb2;   // → 2

// Puissance
const r6 = nb1 ** nb2;  // → 2744
```

Les opérateurs arithmétiques

Le langage JavaScript permet de combiner un opérateur d'affectation avec un opérateur arithmétique.

```
let result = 14;
const nb = 3;

// Addition
result += nb;

// Soustraction
result -= nb;

// Multiplication
result *= nb;

// Division
result /= nb;

// Modulo
result %= nb;

// Puissance
result **= nb;
```

Les opérateurs d'incrément et de décrément

Le langage JavaScript permet de réaliser des actions d'incrémentation et de décrémentation à l'aide d'un raccourci d'écriture.

Deux possibilité :

- Post-Action :
  Après l'utilisation de la variable.
- Pré-Action :
  Avant l'utilisation de la variable.

```
// Post-Incrementation & Post-Decrementation
let nbA = 5, nbB = 3;
const r1 = nbA++;          // → r1: 5 / nbA: 6
const r2 = nbB--;          // → r2: 3 / nbB: 2

// Pré-Incrementation & Pré-Decrementation
let nbC = 5, nbD = 3;
const r3 = ++nbC;          // → r3: 6 / nbC: 6
const r4 = --nbD;          // → r4: 2 / nbD: 2

// Exemple
let nbE = 5, nbF = 3;
const r5 = (nbE-- * ++nbF) + nbE;    // → r5 ?
```

Les constantes de « Number »

Quelques constantes utiles du pseudo-objet« Number » :

```
// La plus petite et la grande valeur numérique positive
Number.MIN_VALUE;
Number.MAX_VALUE;

// Les valeurs entieres minimum et maximum représentable en JS
Number.MIN_SAFE_INTEGER;   // -(2⁵³ - 1)
Number.MAX_SAFE_INTEGER;   //  (2⁵³ - 1)

// Le plus petit intervalle possible entre deux valeurs numériques
Number.EPSILON;
```

Les méthodes de « Number »

Quelques méthodes utiles du pseudo-objet« Number » :

```
// Détermine si la valeur est un entier
const r1 = Number.isInteger(42_000_000_000_000_123);      // → true

// Déterminer si la valeur peut être correctement représentée comme un entier
const r2 = Number.isSafeInteger(42_000_000_000_000_123);  // → false


const v1 = 42.13;

// Détermine si la valeur est un nombre fini
const r3 = Number.isFinite(v1);       // → true

// Détermine si la valeur vaut NaN (Not a Number)
const r4 = Number.isNaN(v1);          // → false
```

Remarque : Contrairement aux méthodes globales, les méthodes “isFinite” et “isNaN” du pseudo-objet « Number » ne réalise pas de conversion avant de tester la valeur.

L'objet « Math »

L'objet « Math » est un élément natif du Javascript qui permet d'utiliser :

- Des constantes
- Des fonctions mathématiques

Quelques exemples :

```
// La valeur de PI (π)
const v1 = Math.PI;          // → 3.141592653589793

// Arrondir un nombre
const v2 = 4.2;
const a1 = Math.round(v1);  // → 4 (l'arrondi mathématique)
const a2 = Math.floor(v1);  // → 4 (l'entier inférieur ou égal)
const a3 = Math.ceil(v1);   // → 5 (l'entier supérieur ou égal)

// Générer un nombre pseudo-aléatoire
const v3 = Math.random();
```