JavaScript - Les opérateurs

[Mise à jour le 25/9/2020]

- **Sources** et compléments sur **MDN Web Docs**
  - Les opérateurs
- **Lectures connexes**
  - Wikis WebPEM : "Préparer un projet de site Web avec l'IDE VSCode"

---

1. Opérations sur les nombres

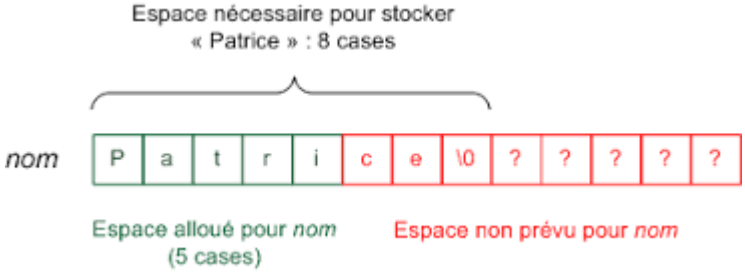
1	2	3
4	5	6
7	8	9

- **Opérations arithmétiques** : **+, -, *, /, %**
- **Incrémenter** (ajouter 1) et **décrémenter** (enlever 1) une variable de type nombre

Exemple

*.js

```
var score = 1 ;
score++ ; // Ecrire score dans la console donne 2
score-- ; // Ecrire score dans la console donne 1
```



2. Opérations sur les chaînes

- **Concaténer des chaînes**

L'opérateur **+** permet d'assembler des chaînes de caractères.

Exemple

*.js

```
var accueil = "Bonjour " ;
var nom = "Nina" ; // Ecrire accueil + nom dans la console donne
"Bonjour Nina"
```

- **Trouver la longueur d'une chaîne**

Pour connaître la longueur d'une chaîne de caractères, il suffit de lier l'attribut ***length*** à la fin.

Exemple

*.js

```
"anticonstitutionnellement".length; // résultat dans la console : 25
```

- **Extraire un caractère d'une chaîne**

On accède à un caractère dans une chaîne à partir de sa position comme dans l'exemple ci-dessous ;

Exemple

*.js

```
var nom = "Nicolas" ;
nom[0] ; // résultat dans la console : "N"
nom[3] ; // résultat dans la console : "o"
```

- **Découper des chaînes**

Pour couper un bout d'une chaîne de caractères, on utilise la méthode ***slice(x,y)***. Avec x, le début de la chaîne à extraire et y la fin.

Exemple

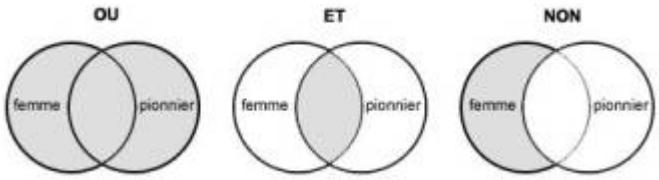
*.js

```
"Une chaîne".slice(1,5); // résultat : "ne c"
```

- **Transformer des chaînes de caractères tout en majuscules ou tout en minuscules**
  - La méthode ***toUpperCase()*** affiche un texte avec tous ses caractères en majuscule.
  - La méthode ***toLowerCase()*** affiche un texte avec tous ses caractères en minuscule.

3. Opérations sur les booléens

Un booléen prend soit la valeur vraie (**true**) soit la valeur faux (**false**).



Les opérateurs logiques

- **&&** signifie (**ET**). Cet opérateur s'emploie avec deux valeurs booléennes pour savoir si elles sont toutes les deux vraies.

Exemple

*.js

```
var a = true ;
var b = false ;
var c = a && b ; // Ecrire c dans la console donne false
```

- **||** signifie (**OU**). Cet opérateur s'emploie avec deux valeurs booléennes pour savoir si l'une des deux ou les deux sont vraies.

Exemple

*.js

```
var a = true ;
var b = false ;
var c = a && b ; // Ecrire c dans la console donne true
```

- **!** signifie (**NON**)

Exemple

*.js

```
var a = true ;
```

```
var c = !a ;  // Ecrire c dans la console donne false
```

Les opérateurs de comparaison

Les opérateurs de comparaison sont : **<, >, < =, >=, ==, ===**

Exemple

*.js

```
var hauteur = 165 ;
var hauteurMin = 150 ;
hauteur > hauteurMin; // résultat dans la console : true
```

L'opérateur "**===**" est l'opérateur de stricte égalité. Les valeurs comparées doivent être de même type. L'opérateur "**==**" ne tient pas compte des types.

Exemple

*.js

```
var chaine = "5";
var nombre = 5;
chaine === nombre; // résultat dans la console : false
chaine == nombre; // résultat dans la console : true
```

Attention à l'utilisation du double égal.

On pourrait penser que son utilisation est plus simple ! Mais, par exemple :

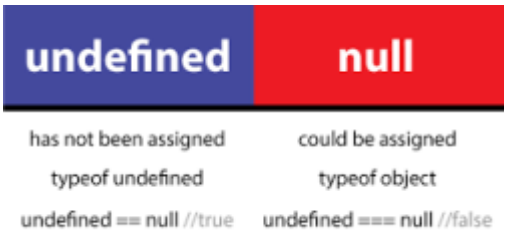
*.js

```
0 == false; // résultat dans la console : true
"false" == false; // résultat dans la console : false
```

Si on compare deux valeurs avec le double égal, JavaScript essaie d'abord de les convertir dans le même type.

4 Les valeurs undefined et null

Ces deux valeurs signifient : rien.



undefined est la valeur que JavaScript utilise s'il n'a pas de valeur à assigner à quelque chose.

Exemple

*.js

```
var variableSansValeur;
variableSansValeur; // résultat dans la console : undefined
```

null est habituellement utilisé pour dire clairement que la variable est vide.

Exemple

*.js

```
var variableSansValeur = null;
variableSansValeur; // résultat dans la console : null
```

Ressources

- Bibliographie
- Webographie

- Lexique