4 minutes

Les conversions

La conversion est simplement le fait de changer le type d'une valeur. Par exemple en changeant une chaîne de caractères en nombre.

Il existe deux types de conversion en `JavaScript` : la conversion explicite et la conversion implicite.

La conversion explicite

La conversion explicite utilise des objets permettant de réaliser la conversion vers le type spécifié :

```
String(1);  // "1"
Number("22"); // 22
Boolean("true"); // true
```

Copier

Conversion en chaîne de caractères

Pour les conversions en chaîne de caractères avec `String()` les résultats sont extrêmement prévisibles :

```
String(null); // 'null'
String(undefined); // 'undefined'
String(true); // 'true'
String(false); // 'false'
String(Symbol('test')); // 'Symbol(test)'
```

Copier

Conversion en booléen

Pour les conversions en booléen c'est un peu moins prévisible, mais cela reste simple :

```
Boolean({}); // true
Boolean([]); // true
Boolean(Symbol());  // true
Boolean(function() {}); // true
Boolean('test'); // true
Boolean(22); // true
Boolean(''); // false
Boolean(0); // false
Boolean(NaN); // false
Boolean(null); // false
```

```
Boolean(undefined); // false
Boolean(false); // false
```

Copier

Conversion en nombre

Pour les nombres c'est un peu plus complexe :

```
Number(null); // 0
Number(undefined); // NaN
Number(true); // 1
Number(false);// 0
Number(" 22 "); // 22
Number("-12.32"); // -12.32
Number("
"); // 0
Number(" 55s "); // NaN
Number(42); // 42
Number(Symbol('bonjourl')); // TypeError
```

Copier

`NaN` est une valeur spéciale indiquant que la valeur n'a pas pu être converti en nombre. Elle est donc `Not a Number`, littéralement, pas un nombre.

Remarquons également que les espaces sont retirés lors de la conversion et ne pose pas de problème.

Notez que si `null` se convertit en `0` ce n'est pas le cas pour `undefined`.

Étonnamment `Symbol` renvoi une erreur plutôt que d'être converti en `NaN`.

La conversion implicite ou automatique (`type coercion`)

L'interpréteur effectue également des **conversions automatiques dans certains cas**, c'est ce qu'on appelle la conversion implicite.

Les conversions automatiques se font uniquement vers trois types : nombre, chaîne de caractères et booléens.

Premièrement, elles se produisent lors **des comparaisons**, notamment avec **l'opérateur de comparaison d'égalité faible** == que nous verrons dans la prochaine leçon.

Un exemple :

```
2 == '3'; // false
2 == '2'; // true
```

Copier

Deuxièmement, elles se produisent lors de **l'évaluation des conditions et les opérateurs logiques**, que nous verrons dans le prochain chapitre.

Un exemple :

```
2 > '3'; // false
```

Copier

Troisièmement, elles se produisent **avec l'opérateur +** qui va convertir automatiquement en chaîne de caractères si au moins une chaîne de caractères est présente, sinon en nombre si possible :

```
42 + ''; // "42"
true + false; // 1
12 / '6'; // 2
"a" + 15 + 3; // "a153"
15 + 3 + "a"; // "18a"
```

Copier

Les deux dernières lignes se comprennent aisément avec **l'associativité** que nous avons vu. Pour l'opérateur d'addition la règle d'associativité est une évaluation de gauche vers la droite.

Donc pour `"a" + 15 + 3;` nous avons `"a" + 15` puis `"a15" + 3`.

Alors que pour `15 + 3 + "a";` nous avons `15 + 3` puis `18 + "a"`.

Nous verrons d'autres exemples au fur et à mesure des chapitres.