Ce qu'il faut savoir sur les conversions de type.

Concernant les conversions de type.

Nous avons vu que :

`var myString = "1";` est de type string.

`var myInt = 1;` est de type int.

Si je souhaite faire une simple addition avec ma variable myString je dois au préalable la convertir en type int.

Pour cela, je peux utiliser `int.Parse(myString);`

👉 "Mais qu'en est-il des autres conversions de types ?"

Si vous souhaitez convertir un `int` en `string` vous pouvez utiliser l'extension de méthode .ToString()

Par exemple :

```
var result = myInt.ToString();
```

La variable `result` vaudra bien **"1"**.

Mais c'est également valable **pour convertir les autres types** en `string` !

Par exemple

var myBool = true;var result = myBool.ToString();

La variable `result` vaudra **"true" et non plus true (sans parenthèses)**.

👉 "Et si je veux convertir un `bool` en `string` ?"

Vous allez pouvoir utiliser, de la même manière que `int.Parse()` la variable `bool.Parse(`) avec entre parenthèses la variable ou la chaine de caractère que vous souhaitez convertir.

Ainsi:

var myString = "false";var result = bool.Parse(myString);

La variable `result` donnera `false`.

Et, bonne nouvelle, il en est de même pour convertir en `float` avec l'instruction `float.Parse()` !

En résumé

Je veux convertir en `string`, je rajoute :

`.ToString()` à ma variable, je n'ai pas besoin de mettre quoi que ce soit entre parenthèses.

Et si je veux convertir respectivement en `int`, `bool` ou `float`, je peux utiliser respectivement :

int.Parse();
bool.Parse();
float.Parse();

Avec entre parenthèses, l'argument à convertir. Et bien évidemment si le type n'est pas convertible, le programme crachera lamentablement ! Et pour pallier à cette erreur nous pouvons utiliser l'instruction TryParse au lieu de Parse. Mais cette commande est plus complexe et sera vu plus loin dans le cours.

Pour aller plus loin

Sachez qu'il existe aussi la possibilité de faire, ce qu'on appelle, **une conversion explicite ou cast** mais au risque d'une perte d'information.

Par exemple, si j'ai le code suivant :

`double x = 1234.7;` `double` étant un nombre décimal similaire à `float`.

Supposons que je souhaite convertir `x` en `int`. Je peux faire `int.Parse` comme nous l'avons vu mais je peux aussi simplement écrire:

```
int y = (int)x;
```

En écrivant le code ci dessus, **j'indique explicitement à mon code que je souhaite convertir la variable x en type double en variable y en type int.** Le résultat sera 1234 car comme nous l'avons vu int ne permet de stocker uniquement les nombres entiers.

Donc **si vous voyez un type entre parenthèses préfixer un autre élément vous savez qu'il s'agit d'une conversion de type.**