

Strings et affichage formaté

On a vu dans le chapitre sur les [types primitifs](#) que l'on peut faire des chaînes de caractères avec `&str`, mais ici on va voir la version plus évoluée avec `String`

Ce type plus complexe a beaucoup de méthodes et de possibilités. D'abord voyons comment le créer :

```
// On peut créer un nouveau String vide comme ceci:
let mon_string = String::new();

// On peut aussi créer un String à partir d'un littéral (exemple "hello")
let mon_autre_string = String::from("Hello World");

// Sinon on peut aussi convertir un &str en String avec la méthode de &str 'to_string'
let troisieme_string = "Foo Bar".to_string();

// Affichons tout ça
println!("{}", mon_string, mon_autre_string, troisieme_string);
```

Contrairement aux `&str`, on peut aussi concaténer des `String` avec des `&str`

```
let un = String::from("hello ");
let deux = "world";
let trois = un + deux;
println!("{}", trois);
```

Je ne vais pas passer en revue toutes les méthodes de `String`, car après avoir compris les types `Option` que l'on va voir dans [un futur chapitre](#) et celui sur les [fonctions](#), vous serez en mesure de comprendre toutes les méthodes à partir de la documentation.

Parmi d'autres, voici quelques exemples de choses qui peuvent être faites avec `String`: remplacer des chaînes de caractères, remplacer le début ou la fin de chaîne de caractère (attention peut nécessiter d'utiliser la méthode `trim()` avant), etc.

Toutes la documentation est disponible [ici](#).

Affichage formaté

Maintenant pour afficher des choses dans le terminal ou formater des choses d'une certaine manière. On peut utiliser des *macros*.

La première que j'ai déjà utilisé avant est `println!()`, l'autre pour formater est `format!()` et enfin une troisième pour print sans retour à la ligne est `print!()`

```
// On peut print à partir d'un littéral
println!("Hello World");

// Mais on peut aussi print avec une sorte de template
let nombre = 42;
let chaine = "La réponse de la vie est ";

// Pour que l'un type puisse être affiché comme ceci il faut qu'il implémente le type `Display`
// Ne faites pas attention au & maintenant, on va y venir plus tard
println!("{}", {}, &chaine, nombre);

let result = format!("Même chose mais avec format d'abord → {} {}", chaine, nombre);
println!("{}", result);

// En revanche on peut utiliser {:?} et {:#?} pour afficher certains types qui ne sont normalement pas affichables
let monvecteur = vec![chaine, "42"];
println!("{:?}", &monvecteur);
println!("{:?}", monvecteur); // Celui-ci est appelé "pretty print"
```

Revision #1

Created 27 April 2023 06:10:28 by SnowCode

Updated 27 April 2023 06:12:49 by SnowCode