Introduction à Typescript

Le TypeScript est un sur-ensemble (un “superset”) de JavaScript qui est transcompilé (transcompilation : "traduction" d'un langage de programmation vers un autre - différent de la compilation, qui transforme généralement le code vers un format exécutable) en JavaScript pour être compréhensible par les navigateurs. Il ajoute des fonctionnalités extrêmement utiles comme :

- le typage strict, qui permet de s’assurer qu’une variable ou une valeur passée vers ou retournée par une fonction soit du type prévu ;
- les fonctions dites lambda ou arrow, permettant un code plus lisible et donc plus simple à maintenir ;
- les classes et interfaces, permettant de coder de manière beaucoup plus modulaire et robuste.

Que faut t'il connaitre pour suivre ce cours?

Pré-requis :

- La programmation orienté objet
- Le JS

Connaissance très utiles pour ce cours :

- ES6 / ES7 (*voir ECMAScript dans le cours Javascript*)
- Techno web

I - Le Typage

A - Les différents types

Les types de bases sont les même qu'en Javascript c'est à dire:

- number (entier, float, etc...)
- string
- boolean
- any (qui est le type 'objet' JS)

B - Les variables

Pour typer une variable c'est très simple :

```
var maVariableNombre :number;
```
> *variable de type nombre*

```
var maVariableString :string;
```
> *variable de type chaine de caractères*

Les variables peuvent être typé implicitement à la déclaration:

```
var maVarNombre = 1;
```
> variable de type number

```
var maVarString = "hello world";
```
> variable de type string

```
var maVarAny = null;
```
> variable de type any

Vous savez maintenant typer des variables en Typescript.

C - Les fonctions

Pour les fonctions c'est le même concept ":type" pour indiquer le type:

```
function triple(n: number): number { return 3 * n; }
```

II - Les classes

Comme dans Javascript ES6, il y a la notion de class et d'interface dans typescript.
Les notion d'héritage existe,
le concept de public privé protected, en gros tous les grands principes écriture d'une clas
se :

```
class Lion extends Animal { // class Lion herite de Animal

  sex: string; // attribut

  constructor(name: string, sex: string) { // contructeur
    super(name); // call constructeur de la class mère
    this.sex = sex;
  }

  shout(): string { // une methode
    return "Rooooaarrr!"
  }
}

//instanciation
var myLion = new Lion('pablo','female');

//call d'un methode
myLion.shout();
```

On voit donc comment déclarer une class avec héritage. Le fonctionnement est très proche de JS ES6 la grande différence se situe au niveau des attributs qui sont déclarés dans le corps de la class.

Les interfaces

La nuance avec la class est qu'on ne définit l'intérieur des méthodes:

```
interface I3 extends I2 {
     c: boolean;
     f(truc :I2):string;
}
```

III - Autre beauté du langage

A - Fonction anonymes

Il existe en Typescript une façon plus réduite d'écrire une fonction anonyme
(la version javascript reste toujours disponible). Cette nouvelle écriture permet d'améliorer
lisibilité du code et de réduire l'écriture.

```
function (a, b) { return a * b; } //fonction anonyme JS
```

```
(a, b) => a * b //fonction anonyme fléché
```

B - Typage anonyme

On peut en Typescript créé des types anonymes lors de la déclaration d'une variable ou
typer un d'une fonction de callback par exemple.

```
var monObjetAnonyme: { a: number, b: string, c: string[] };
```

Fonctionnalité similaire a ES6

- L'utilisation de let et const.
- Les Promises
- le generator (yield et function*) en fonction de la version de translation utilisé
- async, await sur les fonction / méthode avec ES7

[Source : https://www.supinfo.com/articles/single/3916-cours-n2-introduction-typescript#le-typage]