TYPESCRIPT : LES CONSTRUCTEURS

DÉFINITION

- Un constructeur est une fonction 'spéciale' faisant partie d'un objet
- Il permet de saisir les paramètres d'une instance lors de sa création

SYNTAXE

```
class NomClasse {
    _param1: string
    _param2: string

    constructor(param1: string, param2: string) {
        this._param1 = param1
        this._param2 = param2
    }
}
```

EXEMPLE

```
class Formateur {
    // On peut retirer le "!", qui n'a plus aucun intérêt
    _nom: string
    _prenom: string

    // Déclaration :
    constructor(nom: string, prenom: string) {
        this._nom = nom
        this._prenom = prenom
    }
}

// Appel lors de la création de l'instance
let formateur1 = new Formateur('Polnareff', 'Michel')
```

EXEMPLE

```
class Formateur {
    // On peut retirer le "!", qui n'a plus aucun intérêt
    _nom: string
    _prenom: string

    // Déclaration :
    constructor(nom: string, prenom: string) {
        this._nom = nom
        this._prenom = prenom
    }
}

// Appel lors de la création de l'instance
let formateur1 = new Formateur('Polnareff', 'Michel')
```

L'ENCAPSULATION

L'ENCAPSULATION

- Second pilier de la POO
- Stipule qu'il est préférable de modifier/afficher un objet depuis l'intérieur que l'extérieur
- Implique de définir la portée des propriétés comme étant privée ou protégée

LES ACCESSEURS

- Plutôt que de laisser accès aux paramètres de l'objet, nous lui laisseron l'accès aux getters/setters
- Nous pourrons ainsi y ajouter de la logique et traitement.
- Pour ce faire, nous définirons une portée pour chaque propriété.

PORTÉE : PUBLIC

- Par défaut
- La propriété/fonction est accessible partout

PORTÉE : PRIVATE

- La propriété n'est accessible que par la classe
- Une fois l'instance créé, on ne peut ni accéder ni modifier la propriété
- La propriété/fonction ne sera pas héritée en cas d'héritage de classe

PORTÉE : PROTECTED

- Idem que 'Private'
- La propriété/fonction peut être héritée
- Voir chapitre sur l'héritage

PORTÉE : READONLY

- La portée readonly permet d'autoriser la lecture
- L'écriture sera donc impossible

DÉMONSTRATION

Jouer avec les portée

LES GETTERS

DÉFINITON

- Un getter est une fonction 'classique' qui a pour but de présenter de l'information
- La fonction est prefixée de '**get**'
- S'agissant d'une fonction, on pourra y intégrer de la logque
- On parle d'accesseur

POURQUOI ?

- La portée privée contribue à limiter le nombre d'erreurs, et renforce la sécurité du code
- Les getters/setters permettent l'utilisation de logique, contrairement à une simple variable
- On pourra par exemple transformer le format de la date US vers EU

L'ENCAPSULATION

- L'encapsulation ne désigne donc pas le simple fait de modifier les portées
- Il désigne le fait de présenter des informations pertinentes à l'utilisateur, sans qu'il n'ait à savoir comment elles sont stockées.
- Vous n'avez pas besoin de savoir comment fonctionne votre porte de voiture : vous voulez juste pouvoir l'ouvrir.

SYNTAXE

```
class Formateur {
    private _nom: string
    private _prenom: string

    constructor(nom: string, prenom: string) {
        this._nom = nom
        this._prenom = prenom
    }

    get fullName(): string {
        return `${this._prenom} ${this._nom}`
    }

    get nom(): string {
        return `${this._nom}`
    }
```

SYNTAXE

```
 8      }
 9
10      get fullName(): string {
11          return `${this._prenom} ${this._nom}`
12      }
13
14      get nom(): string {
15          return `${this._nom}`
16      }
17  }
18
19  let formateur: Formateur = new Formateur('Polnareff', 'Michel')
20
21  console.log(formateur.fullName)
22  console.log(formateur.nom)
```

SYNTAXE

```
 8      }
 9
10      get fullName(): string {
11          return `${this._prenom} ${this._nom}`
12      }
13
14      get nom(): string {
15          return `${this._nom}`
16      }
17  }
18
19  let formateur: Formateur = new Formateur('Polnareff', 'Michel')
20
21  console.log(formateur.fullName)
22  console.log(formateur.nom)
```

UNDERSCORE

- Ajouter un '_' devant le nom du paramètre nous permet de différencier le paramètre du getter
- En faisant ainsi, on peut donner le même nom à notre setter et notre propriété
- Il n'y a aucun risque de confusion, ni besoin de trouver de nom alambiqué

LES SETTERS

DÉFINITION

- Comme pour les getters, mais pour définir/redéfinir un paramètre
- Il contiendra notre fonction et sa logique pour modifier les paramètres.

```
class Formateur {
    private _nom: string
    private _prenom: string

    constructor(nom: string, prenom: string) {
        this._nom = nom
        this._prenom = prenom
    }

    get fullName(): string {
        return `${this._prenom} ${this._nom}`
    }

    set nom(nouveauNom: string) {
        if (nouveauNom.length > 1 ) {
            this._nom = nouveauNom
```

DÉMONSTRATION

Ajout des getter/setters à Employee

EXERCICE !

Réalisez l'exercice 3

LA SUITE !

La suite ici!