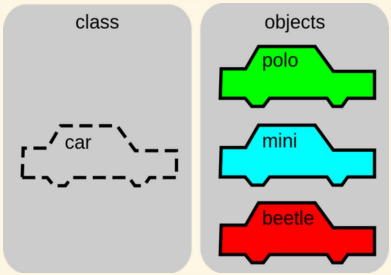
TYPESCRIPT : LES CLASSES

RAPPELS : CLASSES



- Une classe est un modèle permettant de créer des objets
- C'est le premier pilier de la POO

RAPPELS : LES INSTANCES

Le terme **instance** désigne un objet créé à l'aide d'une classe

RAPPELS : BONNES PRATIQUES

- On utilise le '**camelCase**' pour les fonctions et variables
- On utilise le **PascalCase** pour les classes

CRÉER UNE CLASSE

SYNTAXE

Pour déclarer une classe, on utilise la syntaxe suivante :

```
class NomClasse {
    // ...
}
```

Exemple :

```
class Formateur {
    // ...
}
```

Lorsque l'on créé une classe, TypeScript considère qu'il s'agit d'un type à part entière

INSTANCES

On peut créer des instances grâce au mot clef `new` :

```
1 class Formateur {
2     // ...
3 }
4 const alex: Formateur = new Formateur()
```

LES PROPRIÉTÉS

DÉFINITION

- 'Propriété' désigne les attributs des objets de notre classe
- Chaque instance disposera donc des mêmes propriétés

SYNTAXE

```
class Formateur {
    nom: string
    prenom: string
}
```

- On déclare les noms des attributs ainsi que leurs types
- On pourra définir les propriétés une fois l'instance créée

EXEMPLE

```
1  class Formateur {
2      nom: string;
3      prenom: string;
4  }
5
6  let formateur1: Formateur = new Formateur()
7
8  formateur1.nom = 'Devos'
9  formateur1.prenom = 'Alexandre'
10
11 console.log(formateur1.prenom + " " + formateur1.nom)
```

Essayez ceci et vous aurez une erreur !

LES PARAMÈTRES

- Bien que le code soit fonctionnel, l'IDE renvoit une erreur
- C'est du aux propriétés qui ne disposent de valeurs par défaut
- On risque de créer par inadvertence des instances avec des propriétés **undefined**

LES PARAMÈTRES PAR DÉFAUT

On peut préciser un paramètre par défaut comme suit :

```
class Formateur {
    nom: string = 'Doe';
    prenom: string = 'John';
}
```

DEFINITE ASSIGNMENT ANSERTION OPERATOR

- Il nous est possible d'ajouter `!` pour indiquer à TS que le paramètre sera rempli plus tard

```
1 class Formateur {
2     nom!: string
3     prenom!: string
4 }
```

- Notez qu'il est préférable d'utilsier un **constructeur** pour éviter cette erreur

LES MÉTHODES

MÉTHODES

- **méthode** désigne une fonction propre à une classe
- Ces méthodes seront accessibles à toutes les instances
- Elle contiendra souvent de la logique afin de réaliser des opérations

SYNTAXE

Une méthode peut être déclarée comme suit :

```
class Formateur {
    nom: string = 'Doe'
    prenom: string = 'John'

    getFormateurFullName(): string {
        return `${this.nom} ${this.prenom}`
    }
}
```

LA DOCUMENTATION

Notez que, comme dans tous les langages :

LA DOCUMENTATION EST OBLIGATOIRE

DOCUMENTATION

On précise la documentation directement au dessus de la fonction :

```
class Formateur {
    nom: string = 'Doe'
    prenom: string = 'John'

    /**
     * Affiche le nom complet du formateur
     * @returns Nom complet du formateur
     */
    getFormateurFullName(): string {
        return `${this.nom} ${this.prenom}`
    }
}
```

THIS ??

- le mot `this` désigne la propriété de l'instance
- Il nous permet donc d'indiquer que l'on souhaite réaliser une opération sur le paramètre de l'instance

DÉMONSTRATION

Classe 'employé'

EXERCICE !

Réaliser l'exercice 2

LA SUITE

Par ici !