# Travail Pratique – Les classes

Ce TP a pour but de vous permettre de découvrir et créer vos propres classes en JavaScript. C'est un élément fondamental de la programmation OBJET qui est la bonne pratique.

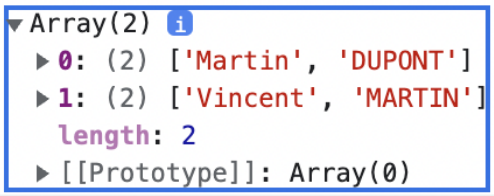
Pour commencer, créez un fichier HTML de base ainsi qu'un script javascript associé appelé main.js. N'oubliez pas le DEFER !

## Etape n°1. Créer une liste de stagiaires.

Les stagiaires ont un nom et un prénom. Les tableaux multi-dimensionnels n'existent pas en Java-Script. Comment faire ?
1ère méthode : créer un tableau par stagiaire. Créez un tableau de tableau.

```
let stagiaire1 = ['Martin','DUPONT'];
let stagiaire2 = ['Vincent','MARTIN'];
let stagiaires = [stagiaire1,stagiaire2];
console.log(stagiaires);
```

Vous pouvez vérifier ce que cela affiche et observer.




Pour afficher uniquement le prénom du deuxième stagiaire :

```
console.log(stagiaires[1][0]);
```

Quelles sont les faiblesses de la méthode ?

- Est-ce que ce que l'écriture permet de savoir facilement ce que l'on affiche ?
- Est-ce que l'on peut empêcher d'éventuelles inversions dans l'ordre des informations ?

## Etape n°2. Création de notre première classe

Il nous serait pratique de créer une variable davantage structurée qui indiquerait clairement le nom et le prénom. Cela s'appelle un OBJET. Et un objet est une INSTANCE d'une classe.
Il nous faut définir le « patron » de l'objet, au sens couturier du terme.
Dans un fichier séparé que nous appelons stagiaire.js. N'oubliez pas de le référencer dans le fichier index.html AVANT l'autre fichier.

Dans le fichier stagiaire.js

```
class Stagiaire
{
   nom;
   prenom;
}
```

Pour utiliser un objet de cette classe, il faut l'instancier, il faut le fabriquer avec le mot-clé new.
Dans le fichier main.js

```
let stagiaireA = new Stagiaire();
```

```
stagiaireA.nom = "DUPONT";
stagiaireA.prenom = "Martin";
```

De la même manière, créez un « stagiaireB » appelé Vincent MARTIN.

Créez ensuite un tableau contenant les deux stagiaires. Affichez-le avec console.log.
Enfin, affichez le prénom du deuxième stagiaire. Voici la syntaxe.

```
console.log(stagiaires[1].prenom);
```

Plus clair non ?
Mais on peut mieux faire.

## Etape n°3. Créer des méthodes

Les méthodes, c'est comme une fonction. La différence ? C'est qu'elle est rattachée à une classe. Une méthode ne peut pas être appelée s'il n'y a pas un objet instancié. Créons une méthode qui retourne un petit texte de salutation.

Dans le fichier stagiaire.js

```
class Stagiaire
{
    nom;
    prenom;

    salutation()
    {
        return `je m'appelle ${this.prenom} ${this.nom}`;
    }
}
```

Le mot clé « this » n'est utilisable que dans la définition d'une METHODE dans la définition de la CLASSE. Cela désigne l'instance de l'objet à partir duquel la méthode est appelée.

Si on utilise cette méthode dans le fichier main.js :

```
console.log(stagiaires[0].salutation());
```

Le « this » est interprété suite à cet appel et donc fera le lien avec notres stagiaires[0].
This.prenom contiendra bien « Martin ».

## Etape n°4. Construire un objet

Les méthodes sont comme les fonctions elles ont des paramètres. Lorsque l'on utilise le mot-clé « new » on appelle une méthode spéciale, définie par défaut, le « constructeur », sans paramètres.
Nous pouvons personnaliser notre constructeur.

```
    constructor(prenom, nom)
    {
        this.prenom = prenom;
        this.nom = nom;
    }
```

Pour l'utiliser ?

```
let anotherStagiaire = new Stagiaire("Lucette","Anderson");
```

Affichez le résultat avec console.log.

Vous vous demandez peut-être pourquoi s'embêter ? Eh bien, le nom de famille, vous l'avez remarqué, n'est pas mis en majuscule. Cela peut être fait directement dans le constructeur !

```
this.nom = pNom.toUpperCase();
```

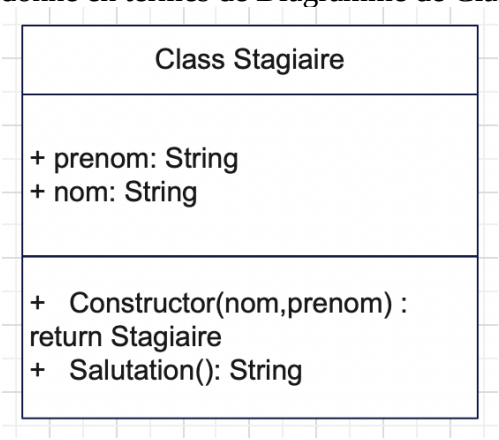
Attention, cela peut bugger si vous utilisez le constructeur sans paramètres (vous devez mettre en commentaire ou amender le code précédent pour débugger !).

Vous pouvez également donner des valeurs par défaut aux paramètres dans le prototype de la méthode Constructor.

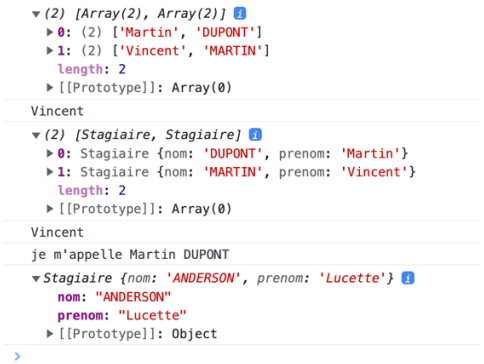
```
constructor(pPrenom="",pNom="")
```

# Bilan

Vous avez créé une classe Stagiaire avec un constructeur personnalisé et une première méthode. Voici ce que cela donne en termes de Diagramme de Classe.

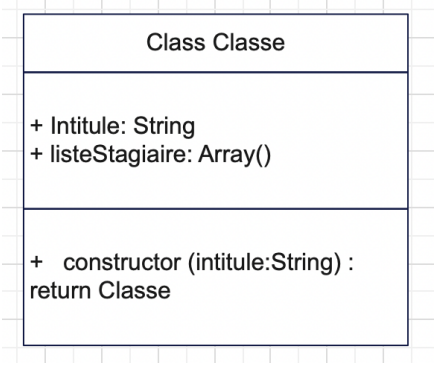



Au niveau du LOG, vous devez voir quelque chose comme ceci :



# Classes et relations entre classes

Vous avez probablement remarqué que votre classe (votre groupe de stagiaire) était composée de… stagiaires.
Nous allons donc créer une classe Classe qui sera définie ainsi :




Créez bien entendu un nouveau fichier classe.js à référencer également dans le index.html.

## Etape n°1. Création des propriétés et du constructeur

Créez la classe Classe ainsi que son constructeur. Par défaut le champ « listeStagiaire » contiendra un tableau vide ( new Array()).

## Etape n°2. Ajout d'un stagiaire

Créez la méthode ajouterStagiaire à l'aide de la fonction « push » utilisable avec les tableaux.
Array - push
Testez le code en créant une classe de deux stagiaires. Vous devez, avec console.log obtenir quelque chose ainsi :

```
▼Classe {intitule: 'DWWM-14', listeStagiaire: Array(2)} i
  intitule: "DWWM-14"
  ▼listeStagiaire: Array(2)
    ▶0: Stagiaire {nom: 'ANDERSON', prenom: 'Lucette'}
    ▶1: Stagiaire {nom: 'LUTHER', prenom: 'Martin'}
     length: 2
    ▶[[Prototype]]: Array(0)
  ▶[[Prototype]]: Object
>
```

## Etape n°3. Ajout d'une date de début et de fin

Vous pouvez ajouter deux champs à Classe : dateDebut et dateFin. Pour créer ces dates, utilisez la classe Date fournie avec JS.

**Création d'une date**

La classe Date permet de manipuler les dates avec facilité. Cela gère également les questions de fuseau horaire

```
let dateNaissance = new  Date (1977,10,24);
console.log(dateNaissance.toString());
```

*Lors de la création de la date, la date est configurée, avec l'heure, suivant le fuseau horaire de l'ordinateur de l'utilisateur. En faisant ce code, cela indique logiquement votre fuseau étant comme GMT+2. Attention aux heures d'été et d'hiver.*

Modifiez le code dans main.js pour donner des dates en paramètre au constructeur. Vous devez obtenir ceci.


# Formulaires, Dates et Classes

Créez un formulaire de création d'une Classe : intitule, dateDebut et dateFin. Ajoutez un bouton « créer » et « afficher dans log ».
Dans le fichier main.js, créez une variable stage (avec le mot-clé VAR pour changer) et la fonction creerStage et afficherStage.

CreerStage : récupère les valeurs des inputs et crée un stage, stocké dans la variable stage)
afficherStage : affiche le contenu de la variable stage.

**Saisir des dates**

*Les formulaires « input » peuvent gérer pour nous les dates : utilisez le type Date.*

```
<input type="date" name="dateDebut" id="dateDebut">
```

*La valeur récupérée depuis le input peut être utilisé comme paramètre du constructeur de Date()*

```
let dateDebut = document.querySelector('#dateDebut').value;
console.log(new Date(dateDebut).toString());
```

*Vous obtiendrez bien une date en GMT+2.*

# Aller plus loin.

Ajoutez un affichage du stage directement dans le HTML (dans un DIV).
A la création du stage, ajoutez l'affichage d'un formulaire pour ajouter des stagiaires.
Affichez la liste des stagiaires dans un SELECT par exemple.