Cours sur les bases de TypeScript pour débutant.

TypeScript est un sur-ensemble typé de JavaScript qui apporte une meilleure gestion des types, des fonctionnalités orientées objets (POO), ainsi que des concepts comme les décorateurs. Il est souvent utilisé pour développer des applications web complexes tout en offrant une plus grande robustesse et des outils de développement avancés.

Dans ce cours, nous allons explorer les bases de TypeScript en commençant par les variables, les types, la syntaxe, la programmation orientée objet (POO) et les décorateurs. Ce cours inclut également des exemples simples pour mieux comprendre chaque concept.

1. Introduction à TypeScript

TypeScript permet aux développeurs de spécifier les types de données, ce qui aide à attraper les erreurs avant même que le code ne soit exécuté (grâce à la compilation). Le code TypeScript est ensuite compilé en JavaScript.

Installation de TypeScript

Si vous ne l'avez pas encore installé, vous pouvez le faire en utilisant npm (le gestionnaire de paquets de Node.js) :

```
bash
Copier le code
npm install -g typescript
```

Vous pouvez créer un fichier `.ts` (ex: `index.ts`) et le compiler en JavaScript en utilisant la commande :

```
bash
Copier le code
tsc index.ts
```

---

2. Les Variables et Types de base

2.1 Déclaration de Variables

En TypeScript, les variables sont déclarées avec `let`, `const`, ou `var` (similaire à JavaScript). Cependant, TypeScript permet de spécifier des types de données pour les variables.

Exemple de variables avec types :

```
typescript
Copier le code
```


contact@logonedigital.com www.logonedigital.com (+237) 6 52 10 98 85 / 6 59 48 47 72

Adresse : TotalEnergie Nsimeyong
RC RC/YAO/2024/B/1127
NIU: M052416809892A

```
let nom: string = "Marie";
let age: number = 25;
let isEnrolled: boolean = true;
```

Ici, `nom` est de type `string`, `age` est de type `number`, et `isEnrolled` est de type `boolean`.

2.2 Types de base

- `string` : pour les chaînes de caractères.
- `number` : pour les nombres (flottants et entiers).
- `boolean` : pour les valeurs vraies ou fausses (true/false).
- `any` : peut prendre n'importe quel type. Ce type est à éviter autant que possible, car il désactive les vérifications de type.
- `void` : utilisé principalement pour les fonctions qui ne retournent rien.

Exemple :

```
typescript
Copier le code
let ville: string = "Paris";
let temperature: number = 22.5;
let estConnecte: boolean = false;
let quelconque: any = "Peut être n'importe quoi";
```

2.3 Type `Array` (Tableau)

Un tableau peut être typé en définissant quel type d'éléments il contient.

Exemple :

```
typescript
Copier le code
let nombres: number[] = [1, 2, 3, 4];
let noms: string[] = ["Jean", "Marie", "Paul"];
```

---

3. Les Fonctions

TypeScript permet de définir les types des paramètres et du retour de la fonction.

Exemple de fonction avec types :

```
typescript
Copier le code
function addition(a: number, b: number): number {
    return a + b;
}
```


contact@logonedigital.com www.logonedigital.com (+237) 6 52 10 98 85 / 6 59 48 47 72

Adresse : TotalEnergie Nsimeyong
RC RC/YAO/2024/B/1127
NIU: M052416809892A

```
let resultat: number = addition(5, 3);
console.log(resultat); // Affiche : 8
```

Dans cet exemple, les paramètres `a` et `b` sont des nombres (`number`) et la fonction renvoie également un nombre.

---

4. Types personnalisés : `Interfaces` et `Type Aliases`

4.1 Interfaces

Les interfaces permettent de définir la structure d'un objet.

Exemple :

```
typescript
Copier le code
interface Produit {
    nom: string;
    prix: number;
    enStock: boolean;
}

let produit: Produit = {
    nom: "Ordinateur",
    prix: 999.99,
    enStock: true
};
```

4.2 Type Aliases

Les alias de types permettent de donner un nom à un type. Ils sont similaires aux interfaces, mais sont plus flexibles.

Exemple :

```
typescript
Copier le code
type Point = {
    x: number;
    y: number;
};

let p: Point = {
    x: 10,
    y: 20
};
```

---

contact@logonedigital.com
www.logonedigital.com
(+237) 6 52 10 98 85 / 6 59 48 47 72

Adresse : TotalEnergie Nsimeyong
RC RC/YAO/2024/B/1127
NIU: M052416809892A

5. Programmation Orientée Objet (POO)

TypeScript prend en charge la POO avec des concepts comme les **classes**, **héritage**, **interfaces**, **constructeurs**, etc.

5.1 Classes

Les classes sont au cœur de la POO. Elles définissent des objets en termes de propriétés et de méthodes.

Exemple de Classe simple :

```
typescript
Copier le code
class Personne {
    nom: string;
    age: number;

    constructor(nom: string, age: number) {
        this.nom = nom;
        this.age = age;
    }

    direBonjour(): string {
        return `Bonjour, je m'appelle ${this.nom} et j'ai ${this.age}
ans.`;
    }
}

let personne1 = new Personne("Jean", 30);
console.log(personne1.direBonjour()); // Affiche : Bonjour, je m'appelle
Jean et j'ai 30 ans.
```

5.2 Héritage

Une classe peut hériter d'une autre classe en utilisant le mot-clé `extends`.

Exemple d'héritage :

```
typescript
Copier le code
class Employe extends Personne {
    salaire: number;

    constructor(nom: string, age: number, salaire: number) {
        super(nom, age);  // Appelle le constructeur de la classe parente
        this.salaire = salaire;
    }

    afficherSalaire(): string {
```


contact@logonedigital.com
www.logonedigital.com
(+237) 6 52 10 98 85 / 6 59 48 47 72

Adresse : TotalEnergie Nsimeyong
RC RC/YAO/2024/B/1127
NIU: M052416809892A

```
        return `${this.nom} gagne ${this.salaire} € par an.`;
    }
}

let employe1 = new Employe("Marie", 28, 50000);
console.log(employe1.afficherSalaire());  // Affiche : Marie gagne 50000 €
par an.
```

5.3 Modificateurs d'accès : `public`, `private` et `protected`

- `public` : accessible partout.
- `private` : accessible seulement dans la classe.
- `protected` : accessible dans la classe et les classes dérivées.

Exemple :

```
typescript
Copier le code
class Banque {
    private solde: number;

    constructor(solde: number) {
        this.solde = solde;
    }

    public consulterSolde(): string {
        return `Votre solde est de ${this.solde} €.`;
    }

    private deposer(montant: number): void {
        this.solde += montant;
    }
}
```

---

6. Les Décorateurs (Décoration de classes et méthodes)

Les décorateurs sont des fonctions spéciales qui peuvent être utilisées pour annoter ou modifier des classes et des méthodes. Pour utiliser les décorateurs, vous devez activer la fonctionnalité dans `tsconfig.json` (`experimentalDecorators: true`).

Exemple de décorateur de classe :

```
typescript
Copier le code
function journalisation(constructor: Function) {
    console.log(`Instance de ${constructor.name} créée`);
}

@journalisation
```


contact@logonedigital.com
www.logonedigital.com
(+237) 6 52 10 98 85 / 6 59 48 47 72

Adresse : TotalEnergie Nsimeyong
RC RC/YAO/2024/B/1127
NIU: M052416809892A

```
class Voiture {
    constructor(public marque: string) {}
}

let voiture = new Voiture("Toyota");
// Affichera : Instance de Voiture créée
```

Dans cet exemple, le décorateur `journalisation` est utilisé pour journaliser lorsqu'une instance de la classe `Voiture` est créée.

Exemple de décorateur de méthode :

```
typescript
Copier le code
function temporisation(ms: number) {
    return function(target: any, propertyKey: string, descriptor:
PropertyDescriptor) {
        const originalMethod = descriptor.value;
        descriptor.value = function(...args: any[]) {
            console.log(`Appel différé de ${ms}ms`);
            setTimeout(() => originalMethod.apply(this, args), ms);
        };
    };
}

class Appareil {
    @temporisation(1000)
    allumer() {
        console.log("L'appareil est allumé");
    }
}

let appareil = new Appareil();
appareil.allumer();  // Affiche : "Appel différé de 1000ms", puis après 1
seconde : "L'appareil est allumé"
```

Conclusion

TypeScript est un puissant sur-ensemble de JavaScript qui apporte la gestion des types et des concepts orientés objets. Il permet d'écrire un code plus robuste et maintenable tout en restant compatible avec le vaste écosystème de JavaScript.

Voici un résumé des concepts que nous avons couverts :

1. **Variables et types de base** (string, number, boolean, any, array).
2. **Fonctions** avec typage des paramètres et du retour.
3. **Types personnalisés** avec `Interfaces` et `Type Aliases`.
4. **POO en TypeScript** avec les classes, l'héritage, et les modificateurs d'accès.
5. **Les Décorateurs** pour annoter et modifier les classes et méthodes.

contact@logonedigital.com
www.logonedigital.com
(+237) 6 52 10 98 85 / 6 59 48 47 72

Adresse : TotalEnergie Nsimeyong
RC RC/YAO/2024/B/1127
NIU: M052416809892A

En maîtrisant ces concepts, vous aurez une bonne base pour développer des applications modernes en TypeScript !

contact@logonedigital.com
www.logonedigital.com
(+237) 6 52 10 98 85 / 6 59 48 47 72

Adresse : TotalEnergie Nsimeyong
RC RC/YAO/2024/B/1127
NIU: M052416809892A