

Typescript

langage "javascript" compilé statiquement



POLYTECH
MONTPELLIER



Christophe Fiorio

Typescript vs Javascript

Différences entre Javascript et Typescript

3



Typescript est une surcouche de Javascript développée par Microsoft

Typescript est transcompilé en Javascript

- ❑ Typescript est un langage de programmation orienté objet
- ❑ Typescript est un langage compilé
- ❑ Typescript est fortement typé
- ❑ Typescript prend en charge l'annotation statique de types

Typescript facilite la maintenabilité du code et la prévention des bugs

Typescript permet déployer des applications javascript plus robustes

De javascript à Typescript....

4



Exemple tiré de la page officielle : typescriptlang.org

```
function compact(arr){  
  if (arr.length > 10)  
    return arr.trim(0,10)  
  return arr  
}
```

Aucun warning

Ce code provoque une erreur à l'exécution

De javascript à Typescript....

5



Exemple tiré de la page officielle : typescriptlang.org

```
function compact(arr){  
  if (orr.length > 10)  
    return arr.trim(0,10)  
  return arr  
}
```

orr à la place de arr

Ce code provoque une erreur à l'exécution

De javascript à Typescript....

6



Exemple tiré de la page officielle : typescriptlang.org

```
/** @param {any[] arr} */  
function compact(arr){  
    if (arr.length > 10)  
        return arr.trim(0,10)  
    return arr  
}
```

javadoc pour préciser le type

Ce code provoque une erreur à l'exécution

De javascript à Typescript....

7



Exemple tiré de la page officielle : typescriptlang.org

```
/** @param {any[] arr} */  
function compact(arr){  
    if (arr.length > 10)  
        return arr.trim(0,10)  
    return arr  
}
```

trim n'existe pas pour les tableaux

A red curved arrow pointing from the text 'trim n'existe pas pour les tableaux' to the 'trim' method call in the code snippet.

Ce code provoque une erreur à l'exécution

De javascript à Typescript....

8

Exemple tiré de la page officielle : typescriptlang.org

```
/** @param {any[]} arr */  
function compact(arr: string[]){  
    if (arr.length > 10)  
        return arr.slice(0,10)  
    return arr  
}
```

Typescript ajoute une syntaxe habituelle précisant les types

Exemple de détection de bug

9



En javascript

```
function add (nbr1, nbr2) {  
    return nbr1 + nbr2;  
}  
add(2, 2) //4  
add(2, "two") //2two
```

En Typescript

```
function add (nbr1: number , nbr2: number ) {  
    //nbr1 et nbr2 doit être uniquement de type "number"  
    return nbr1 + nbr2;  
}  
add(2, 2) //4  
add(2, "two") //Erreur lors de la compilation
```

Définition de types

10



```
interface User{  
  name = string;  
  id: number;  
  age: number;  
  occupation: string;  
}
```

```
const user : User = {  
  name = "Fiorio";  
  id: 1;  
  age: 25;  
  occupation: "EC";  
}
```

```
interface User{  
11   name = string;  
    id: number;  
    age: number;  
    occupation: string;  
}
```

```
class user : User = {  
    name = string;  
    id: number;  
    age: number;  
    occupation: string;  
    constructor(name: string, id: number, age: number, occ: string){  
        this.name = name;  
        this.id = id;  
        this.age = age;  
        this.occupation = occ;  
    }  
}
```

Types composés

12



```
type MyBool = true | false;
```

```
type WindowStates = "open" | "closed" | "minimized";
```

```
type LockStates = "locked" | "unlocked";
```

```
type PositiveOddNumbersUnderTen = 1 | 3 | 5 | 7 | 9;
```

```
function getLength(obj: string | string[]) {  
    return obj.length;  
}
```

```
function wrapInArray(obj: string | string[]) {  
    if (typeof obj === "string") {  
        return [obj];  
    }  
    return obj;  
}
```

Types génériques

13



```
type StringArray = Array<string>;
type NumberArray = Array<number>;
type ObjectWithNameArray = Array<{ name: string }>;

interface Backpack<Type> {
  add: (obj: Type) => void;
  get: () => Type;
}
// déclaration d'une constante (la def est ailleurs)
declare const backpack: Backpack<string>;
// object est une string puisque backpack instance avec string
const object = backpack.get();

// Impossible de passer un nombre à add !
backpack.add(23);
Argument of type 'number' is not assignable to parameter of type
'string'.
```

Système de types structurés

14



```
interface Point {  
  x: number;  
  y: number;  
}  
  
function logPoint(p: Point) {  
  console.log(`${p.x}, ${p.y}`);  
}  
  
// logs "12, 26"  
const point = { x: 12, y: 26 };  
logPoint(point);
```

point est identifié au type `Point` de par sa structure

l'appel à `logPoint` est donc valide

```
class VirtualPoint {  
  x: number;  
  y: number;  
  
  constructor(x: number, y: number) {  
    this.x = x;  
    this.y = y;  
  }  
}  
  
const newVPoint = new VirtualPoint(13, 56);  
logPoint(newVPoint); // logs "13, 56"
```

La classe `VirtualPoint` est aussi identifiée au type `Point`

Le langage Typescript

Exercices :

<https://www.typescriptlang.org/play/>

Soit le code suivant :

```
function logPerson(user: User) {  
    console.log(` - ${user.name}, ${user.age}`);  
}
```

```
console.log('Users:');  
users.forEach(logPerson);
```

1. Définissez le type User
2. Définissez et initialisez un tableau users de User
3. Exécutez votre code

Soit le code suivant :

```
interface Admin {  
    name: string;  
    age: number;  
    role: string;  
}  
  
type Person = User | Admin;
```

1. Modifiez la fonction `logPerson` précédente pour

- prendre en paramètre une `Person`
- afficher le rôle ou l'occupation de la `Person`

2. Exécutez votre code