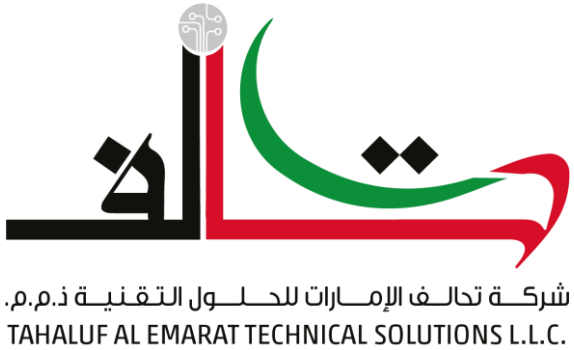
TypeScript

Tahaluf Training Center 2021



Day 05

1. Inheritance
2. Static member

Inheritance

- In TypeScript, we can use common object-oriented patterns.

- One of the most fundamental patterns in class based programming is being able to extend existing classes to create new ones using inheritance.

Inheritance

- Inheritance is one of the fundamental attributes of object-oriented programming.

- Inheritance is the ability of a program to create new classes from an existing class.

Inheritance

- The class whose members are inherited is called the base class/parent class/super class.

- The class that inherits those members is called the derived/child/subclass.

Inheritance

- TypeScript supports only single inheritance

- A class inherits from another class using the ‘extends’ keyword.

Inheritance

```
 class Marks {
 private mark: number;
  get Mark() {
 return this.mark;
                  }
 set Mark(mark: number) {
 this.mark = mark;
                  }
public fun(s1: string | number) {
if (typeof (s1) == "string")
console.log("invalid value");
Else
  this.Mark = s1;
                  }
```

Inheritance

```
var mark = new Marks();
mark.fun("h");
mark.fun(40);
console.log(mark.Mark);
```

❖ Example

```
class person {
name: string;
age: number;
constructor(name?: string, age?: number) {
this.name = name;
this.age = age;
console.log("Person class");
                }
speak() {
console.log("speaking arabic and english
language
    s");
    }
```

```
 display() {
console.log("Name : " + this.name);
console.log("Age : " + this.age);
                    }
                }
```

Inheritance

```
class teacher extends person {
constructor(name?: string, age?: number) {
super(name, age); console.log("Teacher
class"); } explian() { console.log("explains
math course"); } } Inheritance(Example) var
t = new teacher("Noor", 33); t.display();
t.speak(); t.explian();
```

Inheritance

```
class masterStudent extends teacher {
score: number; constructor(name?:
string, age?: number) { super(name,
age); console.log("Master Student
class"); } set Score(s: number) {
this.score = s; } get Score() { return
this.score; }
```

Inheritance

```
 class student extends person {
score: number; constructor(name?:
string, age?: number) {
super(name, age);
console.log("Student class"); }
set Score(s: number) { this.score
= s; } get Score() { return
this.score; } mark() {
console.log("score = " +
this.score); } }
```

Static member

- Unlike an instance property, a static property is shared among all instances of a class.

- The static members of a class are accessed using the class name and dot notation, without creating an object

Static member

```
class Employees {
static headcount: number = 0;
constructor(
private firstName: string,
private lastName: string,
            private jobTitle: string) {

            Employees.headcount++;
        }
}
```

Static member

```
let john = new Employees('John', 'Doe',
'Front-end Developer');
let jane = new Employees('Jane', 'Doe',
'Back-end Developer');


console.log(Employees.headcount); // 2
```

Demo

Demo

```
 class Address{
treetName ?: string;
city ?: string;
postalCode ?: string;
constructor(){
console.log('Address class just initiated new
instance');
        };
    }
```

Demo

```
 class Employee {
employeeName: string;
employeeAge: string;
employeeLevel: number;
salary: number;
constructor(name: string, age: string, level:
number, salary: number) {
this.employeeAge = age;
this.employeeName = name;
this.salary = salary;
this.employeeLevel = level;
                    }
               }
```

Demo

```
class Business {
static numberOfRetaurants: number = 0;
public name?: string;
public logo?: string;
public slogan?: string;
private businessEmail?: string[];
private mobileNo?: string;
private employeesNumber?: number;
protected workHours?: number;
protected employees?: Employee[] = [];
constructor() {
console.log('Business class just initiated new instance')
;
                }
```

Demo

```
//  set Mobile(value: string){
//       this.mobileNo=value;
            //  }
            //  get
mobile():string{
            //     return this
.mobileNo||'';
            //  }
        }
```

Demo

```
 class Restaurant extends Business {
mealsOffered?: string[];
discount?: number[];
address: Address;
addEmployee(name: string, age: string, level:
number, salary: number) {
const emp = new Employee(name, age, level,
salary);
this.employees.push(emp);
                }
```

Demo

```
 constructor(){
console.log('Restaurant class just initiated new
instance');
//to call business class
super();
//to call the address class
this.address = new Address();
Restaurant.numberOfRetaurants++;
// this.Mobile='0788';
            }
```

Demo

```
const foodStation1 = new Restaurant();
//numberOfRestrant=1'
foodStation1.name = 'foodStation1';
foodStation1.slogan = 'Best Burger';
foodStation1.addEmployee('John', '30', 2, 300);
foodStation1.addEmployee('Ahmad', '20', 1, 300);
foodStation1.addEmployee('Ali', '35', 2, 300);
foodStation1.addEmployee('Omar', '33', 1, 300);
console.log(foodStation1);
console.log('number of restaurants',
Restaurant.numberOfRetaurants);
```

Demo

```
 const foodStation2 = new Restaurant();
//numberOfRestrant=1'
foodStation1.name = 'PizzaStation2';
foodStation2.slogan = 'Best Pizza';
foodStation2.addEmployee('John', '30', 2, 300);
foodStation2.addEmployee('Ahmad', '20', 1, 300);
foodStation2.addEmployee('Ali', '35', 2, 300);
```

Demo

```
foodStation2.addEmployee('Omar', '33', 1, 300);
foodStation2.addEmployee('Ahmad', '20', 1, 300);
foodStation2.addEmployee('Ali', '35', 2, 300);
foodStation2.addEmployee('Omar', '33', 1, 300);
console.log(foodStation2);
foodStation2.name = 'foodStation2’;
console.log(foodStation2);
```

Demo


```
console.log('number of restaurants',
Restaurant.numberOfRetaurants);
const foodStation3 = new Restaurant();
const foodStation4 = new Restaurant();
console.log('number of restaurants',
Restaurant.numberOfRetaurants);
```