Define static properties

2 minutes

The properties and methods of the classes defined so far are **instance properties**, meaning that they are instantiated and called on each instance of the class object. There is another type of property called a **static property**. Static properties and methods are shared by all instances of a class.

To make a property static, use the `static` keyword before a property or method name.

For example, you can add a new `static` property to the `Car` class called `numberOfCars` that stores the number of times that the `Car` class is instantiated and set its initial value to 0. Then, in the constructor, increment the count by one.

TypeScript

```
class Car {
    // Properties
    private static numberOfCars: number = 0;  // New static property
    private _make: string;
    private _color: string;
    private _doors: number;

    // Constructor
    constructor(make: string, color: string, doors = 4) {
        this._make = make;
        this._color = color;
        this._doors = doors;
        Car.numberOfCars++; // Increments the value of the static property
    }
    // ...
}
```

Notice that you use the syntax `className.propertyName` instead of `this.` when accessing the static property.

You can also define static methods. You can call the `getNumberOfCars` method to return the value of `numberOfCars`.

TypeScript

```
public static getNumberOfCars(): number {
    return Car.numberOfCars;
}
```

Instantiate the `Car` class as usual and then use the syntax `Car.getNumberOfCars()` to return the number of instances.

TypeScript

```
// Instantiate the Car object with all parameters
let myCar1 = new Car('Cool Car Company', 'blue', 2);
// Instantiates the Car object with all parameters
let myCar2 = new Car('Galaxy Motors', 'blue', 2);
// Returns 2
console.log(Car.getNumberOfCars());
```

Next unit: Extend a class using inheritance

Continue >

How are we doing? ☆ ☆ ☆ ☆ ☆