

Статичні члени класу

Статичні поля. Опис

Статичні поля :

- це поля, які належать класу, а не окремому об'єкту
- існують в одному екземплярі
- описуються з використанням службового слова static
- для звертання до них необхідно використовувати назву класу

Загальна форма	Приклад
<pre>===== опис статичних полів ===== class <u>Назва класу</u> { //--Опис полів класу (належать класу а не об'єктам)--- ---(існують тільки в одному екземплярі) --- static <u>назва_статичного_поля_1</u> = поч. значення1 static <u>назва_статичного_поля_2</u> = поч. значення2 constructor(<u>форм.параметри</u>){ //-- Опис полів (індивідуальні дані об'єктів)--- ---(у кожного об'єкта свої окремі екземпляри)---- this.властивість1 = значення1 this.властивість2 = значення2 . . . } }</pre>	<pre>class <u>Person</u> { //--Опис полів класу (належать класу а не об'єктам)--- -- Максимальний вік, що вважається молодим --- -- це величина для будь-якої людини --- static <u>maxYoungLimit</u> = 45 constructor(age) { //--Опис полів (вік у кожного свій)--- this.<u>Age</u> = age } }</pre>

Статичні поля. Звертання до статичного поля поза межами класу

- це поля, які належать класу, а не окремому об'єкту
- існують в одному екземплярі
- описуються з використанням службового слова *static*
- для звертання до них необхідно використовувати назву класу

Загальна форма

```
===== опис статичних полів =====  
class Назва класу {  
  
    //---Опис полів класу (належать класу а не об'єктам)---  
    //---(існують тільки в одному екземплярі) ---  
  
    static назва_статичного_поля_1 = поч._значення1  
    static назва_статичного_поля_2 = поч._значення2  
  
    . . . . .  
  
}
```

//---- створення об'єкта ----

об'єкт = new **Назва класу** (... параметри ...)

//---- звертання до статичного поля ----

Назва класу . назва_статичного_поля = значення

змінна = **Назва класу** . назва_статичного_поля

Приклад

```
class Person {  
  
    //---Опис полів класу (належать класу а не об'єктам)---  
  
    static maxYoungLimit = 35  
  
    constructor(name, age) {  
        //---Опис полів (вік у кожного свій)---  
        this.Name = name  
        this.Age = age  
    }  
    . . . . .  
}
```

//---- створення об'єкта ----

let p1 = new Person('Ivan', 43)

//---- звертання до статичного поля ----

Person . maxYoungLimit = 45 //Зміна значення

let s = **Person** . maxYoungLimit //Зчитування значення

Статичні поля. Звертання до статичного поля всередині нестатичних методів класу

- це поля, які належать класу, а не окремому об'єкту
- існують в одному екземплярі
- описуються з використанням службового слова static
- для звертання до них необхідно використовувати назву класу

<div>Загальна форма</div> <div>===== опис статичних полів ===== class Назва класу { //---Опис полів класу (належать класу а не об'єктам)--- static назва_статичного_поля_1 = поч._значення1 static назва_статичного_поля_2 = поч._значення2 //-- Опис методів -- функція-метод_1 (форм.парам.) { . . . Назва класу . назва_статичного_поля_1 . . . } }</div> <div>//---- створення об'єкта ---- об'єкт = new Назва класу (... параметри ...) //---- звертання до статичного поля ---- Назва класу . назва_статичного_поля = значення</div>	<div>Приклад</div> <div>class Person { //---Опис полів класу (належать класу а не об'єктам)--- static maxYoungLimit = 45 constructor(age) { //---Опис полів (вік у кожного свій)--- this.Age = age } isUserYoung() { return this.Age <= Person.maxYoungLimit } }</div> <div>//---- створення об'єкта ---- let p1 = new Person('Ivan', 43) let p2 = new Person('Olga', 36) document.write(Person . maxYoungLimit) document.write(p1.isUserYoung()) document.write(p2.isUserYoung()) //---- звертання до статичного поля ---- Person . maxYoungLimit = 45 document.write(Person . maxYoungLimit) document.write(p1.isUserYoung()) document.write(p2.isUserYoung())</div>
---	---

Статичні методи. Звертання до статичного поля всередині нестатичних методів класу

Загальна форма	Приклад
<pre>class Назва класу { //---Опис полів класу (належать класу а не об'єктам)--- static назва_статичного_поля_1 = поч._значення1 static назва_статичного_поля_2 = поч._значення2 //--- Опис нестатичних методів --- функція-метод_1 (форм.парам.) { Назва класу . назва_статичного_поля_1 } //--- Опис статичних методів --- static статична-функція-метод_1 (форм.парам.) { Назва класу . назва_статичного_поля_1 Або this . назва_статичного_поля_1 } }</pre>	<pre>class Person { //---Опис полів класу (належать класу а не об'єктам)--- static maxYoungLimit = 45 static compareCount = 0 constructor(age) { //---Опис полів (вік у кожного свій)--- this.Age = age } isUserYoung() { return this.Age <= Person.maxYoungLimit } static arePersonsSameYears(person1, person2) { Person.compareCount ++ //або this . compareCount ++ return person1.Age === person2.Age } }</pre>
<pre>//---- створення об'єкта ---- об'єкт = new Назва класу (... параметри ...) //---- звертання до статичного метода ---- Назва класу . назва_статичного_метода (параметри)</pre>	<pre>//---- створення об'єкта ---- const p1 = new Person('Ivan', 22) const p2 = new Person('Olga', 28) console.log(Person.arePersonsSameYears(p1, p2))</pre>

- описуються з використанням службового слова static
- для звертання ззовні або з нестатичних методів потрібно використовувати назву класу
- у цих методах this – це сам клас
- для звертання з статичних методів до інших статичних методів чи полів можна також використовувати *this*
- не можуть звертатись до нестатичних полів об'єкту через this !!!**

Статичні методи. Звертання до статичного поля всередині нестатичних методів класу

Загальна форма	Приклад
<pre>class Назва класу { //---Опис полів класу (належать класу а не об'єктам)--- static назва_статичного_поля_1 = поч._значення1 static назва_статичного_поля_2 = поч._значення2 //-- Опис нестатичних методів -- функція-метод_1 (форм.парам.) { . . . Назва класу . назва_статичного_поля_1 . . . } //-- Опис статичних методів -- static статична-функція-метод_1 (форм.парам.) { . . . Назва класу . назва_статичного_поля_1 Або this . назва_статичного_поля_1 } }</pre>	<pre>class Person { //---Опис полів класу (належать класу а не об'єктам)--- static maxYoungLimit = 45 static compareCount = 0 constructor(age) { //---Опис полів (вік у кожного свій)--- this.Age = age } isUserYoung() { return this.Age <= Person.maxYoungLimit } static arePersonsSameYears(person1, person2) { Person.compareCount ++ //або this . compareCount ++ return person1.Age === person2.Age } }</pre>
<pre>//---- створення об'єкта ---- об'єкт = new Назва класу (... параметри ...) //---- звертання до статичного метода ---- Назва класу . назва_статичного_метода (параметри)</pre>	<pre>//---- створення об'єкта ---- const p1 = new Person('Ivan', 22) const p2 = new Person('Olga', 28) console.log(Person.arePersonsSameYears(p1, p2))</pre>

- описуються з використанням службового слова static
- для звертання ззовні або з нестатичних методів потрібно використовувати назву класу
- у цих методах this – це сам клас
- для звертання з статичних методів до інших статичних методів чи полів можна також використовувати *this*
- не можуть звертатись до нестатичних полів об'єкту через this !!!**

Задача. Реалізувати конвертер валют. Курси валют та методи перетворення повинні бути статичними

Задача. Розробити "Рекламний агент". Випадковим чином у заданому діапазоні задається інтервал, і через згенеровану кількість секунд виводиться деяка реклама. Після цього знову генерується випадковий інтервал. У процесі роботи для усіх раніше створених об'єктів - рекламних агентів мінімальне/максимальне значення інтевалу можуть змінюватись.

Задача. Статистика методів.

Дано клас Масив, який зберігає масив і має методи для знаходження суми, добутку, максимального. Користувач може створити довільну кількість об'єктів даного класу. Підрахувати загальну кількість викликів кожного із методів (незалежно від об'єкта)

Статичні приватні поля, методи (геттери/сеттери). Властивості

- приватні поля, геттери, сеттери описуються з використанням службового слова *static*
- для звертання ззовні або з нестатичних властивостей потрібно використовувати назву класу
- у цих методах *this* – це сам клас
- для звертання з статичних методів до інших статичних методів чи полів можна також використовувати *this*

Загальна форма	Приклад
<pre>class Назва класу { //---1) Опис приватного статичного поля --- static # назва_статичного_поля = поч._значення1 static get метод_зчитування_геттер () { return this.# назва_статичного_поля } static set метод_запису_сеттер (newVal) { this.# назва_статичного_поля = newVal } }</pre>	<pre>class Person { //---Опис полів класу (належать класу а не об'єктам)--- static #maxYoungLimit = 45 static get maxYoungLimit() { return this.#maxYoungLimit } static set maxYoungLimit(newVal) { if (newVal < 16) throw new Error('The value is incorrect') this.#maxYoungLimit = newVal } }</pre>
<pre>//---- створення об'єкта ---- об'єкт = new Назва класу (... параметри ...) //---- звертання до статичного метода ---- Назва класу . назва_статичної_властивості = значення змінна = Назва класу . назва_статичної_властивості</pre>	<pre>//---- створення об'єкта ---- const p1 = new Person('Ivan', 22) const p2 = new Person('Olga', 28) Person. maxYoungLimit = 35 let s= Person. maxYoungLimit console.log(Person. maxYoungLimit)</pre>

Статичні поля. Singleton

- гарантує, що клас матиме тільки один екземпляр
- забезпечує глобальну точку доступу до цього екземпляра
- через низку притаманних недоліків деякі розробники вважають його антипатерном

Загальна форма	Приклад
<pre>===== опис статичних полів ===== class Назва класу { ///---Опис полів класу (належать класу а не об'єктам)--- ///---(існують тільки в одному екземлярі) --- static назва_статичного_поля constructor(форм.параметри){ ///---Якщо значення статичного поля визначене--- ///---то повертаємо вже раніше створений об'єкт --- if(Назва класу . назва_статичного_поля) return Назва класу . назва_статичного_поля ///---інаше створюємо новий об'єкт--- this.властивість1 = значення1 this.властивість2 = значення2 . . . Назва класу . назва_статичного_поля = this } }</pre>	<pre>class Director { //поле для зберігання першого створеного об'єкту static directorRef constructor(name, department) { if (Director.directorRef) return Director.directorRef this.name = name this.department = department Director.directorRef = this } toString() { return `\${this.name} - \${this.department}` } }</pre>
	<pre>const d1 = new Director('Ivan', 'depratment 1') console.log(d1.name) //Ivan const d2 = new Director('Olga', 'depratment 1') console.log(d2.name) //Ivan console.log(d1 === d2) //true</pre>

Задача. Черговий. Дано список студентів одного курсу (ПІБ, курс). Розробити менеджер чергових, який дозволяє випадковим чином обирати і запам'ятовувати обраного чергового студента (один раз обрали і не змінюємо)

Статичні члени класів у TypeScript

- Статичні властивості та методи належать безпосередньо класу, а не його екземплярам.
- Для доступу до них використовується ім'я класу, а не об'єкта.
- Використовується ключове слово `static`.

Загальна форма

```
class ClassName {  
    static staticProperty: Type;  
  
    static staticMethod(): ReturnType {  
        // код  
    }  
  
    instanceProperty: Type;  
  
    constructor(instanceProperty: Type) {  
        this.instanceProperty = instanceProperty;  
    }  
}
```

Приклад

```
class Calculator {  
    static PI: number = 3.14159; // статична властивість  
  
    static square(x: number): number { // статичний метод  
        return x * x;  
    }  
  
    multiply(a: number, b: number): number { // звичайний метод  
        return a * b;  
    }  
}
```

```
// Виклик статичних членів  
console.log(Calculator.PI);           // 3.14159  
console.log(Calculator.square(5));    // 25  
  
// Створюємо екземпляр  
const calc = new Calculator();  
console.log(calc.multiply(3, 4));      // 12
```

Статичні блоки

- Статичний блок - це блок коду всередині класу, який виконується один раз при завантаженні класу.
- Використовується для ініціалізації статичних властивостей, особливо складних або залежних від обчислень.
- Підтримується в TypeScript (через трансляцію в JS ES2022+).

Загальна форма

```
class ClassName {  
  static staticProperty1: Type;  
  static staticProperty2: Type;  
  
  static {  
    // код, що виконується один раз при завантаженні класу  
    ClassName.staticProperty1 = ...;  
    ClassName.staticProperty2 = ...;  
  }  
}
```

Приклад

```
class Config {  
  static settings: { [key: string]: string };  
  static version: number;  
  
  // Статичний блок для ініціалізації  
  static {  
    console.log("Ініціалізація класу Config...");  
    Config.settings = { theme: "dark", language: "en" };  
    Config.version = 1;  
  }  
}  
  
console.log(Config.settings.theme); // dark  
console.log(Config.version);        // 1
```


Приклад із кількома статичними блоками

Блоки виконуються в порядку оголошення

```
class Example {  
  static a: number;  
  static b: number;  
  
  static {  
    Example.a = 10;  
    console.log("Перший статичний блок виконано");  
  }  
  
  static {  
    Example.b = Example.a * 2;  
    console.log("Другий статичний блок виконано");  
  }  
}  
  
console.log(Example.a); // 10  
console.log(Example.b); // 20
```