



# Private Fields

Created	@Mar 08, 2020 3:47 PM
Tags	<span>ECMAScript</span> <span>Learning</span> <span>TypeScript</span> <span>程序员生涯</span>

[官网说明](#)

[概述](#)

[私有字段的用法](#)

[私有字段的尝试](#)

[私有字段和 private 修饰符的区别](#)

[私有字段的支持](#)

## 官网说明

[Handbook - TypeScript 3.8](#)

<https://www.typescriptlang.org/docs/handbook/release-notes/typescript-3-8.html#ecmascript-private-fields>

## 概述

对**私有字段**的支持是从 [TypeScript 3.8](#) 开始的。(私有字段目前在 EcmaScript 中尚处于 [stage-3](#) )

私有字段以 **#** 开始，如下：

```
class Person {
  #name: string

  constructor(name: string) {
    this.#name = name;
  }

  greet() {
    console.log(`Hello, my name is ${this.#name}!`);
  }
}
```

```

    }
}

let jeremy = new Person("Jeremy Bearimy");

jeremy.#name
// ~~~~~
// Property '#name' is not accessible outside class 'Person'
// because it has a private identifier.

```

## 私有字段的使用法

私有字段的使用规则：

- 私有字段以 `#` 开始。
- 每个私有字段的名字，在被包含的类中，都是唯一的。
- **TypeScript** 辅助功能修饰符(`public`、`private` 等)不能在私有字段上使用。
- 私有字段不能在所包含的类之外访问；即使是对于 **JavaScript** 使用者来说也是如此。

针对以上规则的第 2 条“每个私有字段的名字，在被包含的类中，都是唯一的。”，有以下示例对其做出解释：

- 常规属性声明容易在子类中被改写，私有字段不会被子类改写，每个私有字段，在所包含的类中，都是唯一的。

```

// 常规属性声明
class C {
    foo = 10;

    cHelper() {
        return this.foo;
    }
}

class D extends C {
    foo = 20;

    dHelper() {
        return this.foo;
    }
}

let instance = new D();
// 'this.foo' refers to the same property on each instance
console.log(instance.cHelper()); // prints '20'
console.log(instance.dHelper()); // prints '20'

```

```

// 私有字段声明
class C {
    #foo = 10;

    cHelper() {
        return this.#foo;
    }
}

class D extends C {
    #foo = 20;

    dHelper() {
        return this.#foo;
    }
}

let instance = new D();
// 'this.#foo' refers to a different field within ea
console.log(instance.cHelper()); // prints '10'
console.log(instance.dHelper()); // prints '20'

```

对于私有字段的使用，还需要注意的一点是：除非在某个对象的类型声明中声明了某个私有字段，否则，在未声明该私有字段的对象中无法使用该私有字段，会报错。

```

class Square {
  #sideLength: number;

  constructor(sideLength: number) {
    this.#sideLength = sideLength;
  }

  equals(other: any) {
    return this.#sideLength === other.#sideLength;
  }
}

const a = new Square(100);
const b = { sideLength: 100 };

// Boom!
// TypeError: attempted to get private field on non-instance
// This fails because 'b' is not an instance of 'Square'.
console.log(a.equals(b));

```

## 私有字段的尝试

在看过对**私有字段**的概述及用法介绍后，我实际尝试了一番，但却碰到了一些问题，具体如下：

- 示例代码：

```

// private-field.ts
class Person {
  #name: string;

  constructor(name: string) {
    this.#name = name;
  }

  greet() {
    console.log(`Hello, my name is ${this.#name}!`);
  }
}

const jeremy = new Person("Jeremy Bearimy");

jeremy.greet();

```

以上示例代码完全是从官网 Copy 下来的，但是在 IDE 中却显示出错！

这是怎么回事呢？

- 看一下报错信息：

```
// TS18028: Private identifiers are only available when targeting ECMAScript 2015 and higher.  
// Private 标识符仅在 ECMAScript 2015 及更高版本可用
```

原来如此！我当前的版本不支持私有字段！

但是不对啊，我的 TypeScript 版本就是 3.8 啊！

```
"typescript": "^3.8.3"
```

那是怎么回事呢？在 [stackoverflow](#) 的一个提问中我找到了原因：我的 `tsconfig.json` 配置错了！

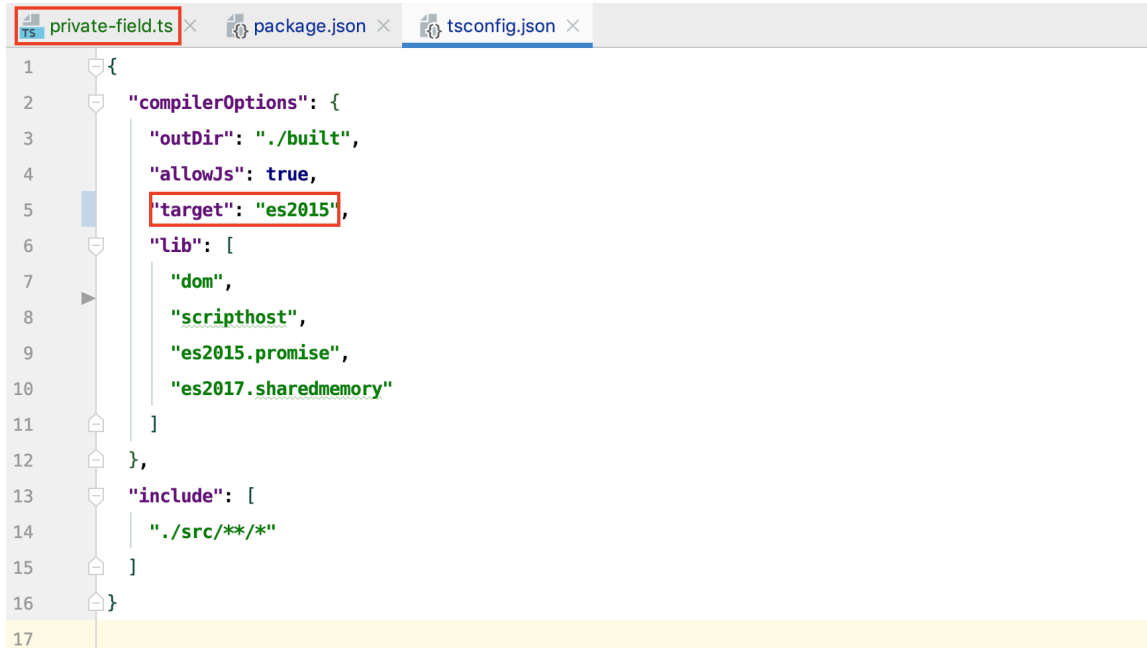


```
1 {
2   "compilerOptions": {
3     "outDir": "./built",
4     "allowJs": true,
5     "target": "es5",
6     "lib": [
7       "dom",
8       "es5",
9       "scripthost",
10      "es2015.promise",
11      "es2017.sharedmemory"
12    ]
13  },
14  "include": [
15    "./src/**/*.ts"
16  ]
17 }
```

我的编译目标版本错了，是 `es5`，而报错信息已经明确提示了，要求的版本是 **ES 2015 及以上**。

- 解决方案

找到了问题所在，那么，解决起来就很简单了！



```
1 {
2   "compilerOptions": {
3     "outDir": "./built",
4     "allowJs": true,
5     "target": "es2015",
6     "lib": [
7       "dom",
8       "scripthost",
9       "es2015.promise",
10      "es2017.sharedmemory"
11    ]
12  },
13  "include": [
14    "./src/**/*.ts"
15  ]
16 }
17
```

可以看到，在将编译目标版本改到 `es2015` 后，原本 `private-field.ts` 上的出错提示已经消失了。

- 运行效果

```
private-field.ts  package.json  tsconfig.json
1  class Person {
2      #name: string;
3
4      constructor(name: string) {
5          this.#name = name;
6      }
7
8      greet() {
9          console.log(`Hello, my name is ${this.#name}!`);
10     }
11 }
12

Person > greet()
Terminal: Local x +
ts-learning master ts-node src/private-field/private-field.ts
Hello, my name is Jeremy Bearimy!
```

- 私有字段的效果

示例已经顺利运行起来了，那么，到底这个**私有字段**有没有用呢？

在之前的示例代码中加上一行：

```
class Person {
    #name: string;

    constructor(name: string) {
        this.#name = name;
    }

    greet() {
        console.log(`Hello, my name is ${this.#name}!`);
    }
}

const jeremy = new Person("Jeremy Bearimy");
jeremy.greet();

console.log(jeremy.#name);
// ~~~~~
// TS18013: Property '#name' is not accessible outside class 'Person' because it has a private identifier.
```

报错了！

```
13  const jeremy = new Person( name: "Jeremy Bearimy");
14
15  jeremy.greet();
16  console.log(jeremy.#name);
17

TS18013: Property '#name' is not accessible outside class 'Person' because it has a private identifier.
Suppress with @ts-ignore  More actions...
```

果然，**私有字段**还是非常犀利的！😍😍😍

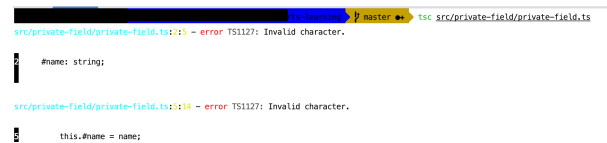
# 私有字段和 private 修饰符的区别

- TypeScript 中的 private 修饰符在编译为 JavaScript 后就完全被删掉了，相当于实际上是没有用的。
- 私有字段即便编译为 JavaScript 也会存在。

## 私有字段的支持

由于目前 ECMAScript 对**私有字段**的支持尚在 stage-3，虽然可以用 Babel 实现在 JavaScript 中使用**私有字段**，但是不使用插件的话，将 TypeScript 编译为 JavaScript 会报错。

当然，也有可能是可以编译成功的，但是我目前由于没有深入地研究，所以暂时还没有实验成功。



```
src/private-field/private-field.ts:1:15 - error TS1127: Invalid character.  
1 #name: string;  
src/private-field/private-field.ts:1:14 - error TS1127: Invalid character.  
1 this.#name = name;
```