ES6 中的关键字 super 该如何理解？

super 关键字**只能够以指定的形式出现在以下地点**，否则代码会出现 SyntaxError，无法通过编译：

- class 语句块内的 constructor 函数的定义内，例如：

```
class B extends A {
    constructor(){
        super()
    }}
```

- **建立 class 时，当且仅当“使用了 extends 关键字并指定了 constructor 函数”，super 关键字必须以 super([arg1[, arg2... argN]])的形式使用一次。**此时 super([arg1[, arg2... argN]])相当于创建了一个对象，且以该对象为 context 调用 extends 关键字指示的函数（以 new 的形式），随后这个对象成为 constructor 函数的 context。因此 **super([arg1[, arg2... argN]])必须出现在 constructor 函数内的第一个 this 关键字之前，**否则会报“this is not defined”的 ReferenceError。**亦可以 super . IdentifierName 的形式出现**（这就与是否使用了 extends 关键字无关了）。**super.IdentifierName 作为 getter 使用时，表示函数 B 的 prototype 属性对象的[[prototype]]；super.IdentifierName 作为 setter 使用时，super 表示 this；**

- class 语句块内的 static 函数的定义内，例如：

```
class B extends A {
    static foo {
        super.test1() // 相当于 A.test1.call(this)
        console.log(super.test2 === A.test2)
    }}
```

- **必须以 super . IdentifierName 的形式出现。super.IdentifierName 作为 getter 时，super 表示该函数 B 的[[prototype]]**（本例中 B 的[[prototype]]即 A）。若通过 super.IdentifierName()来调用函数，则此函数的相当于通过.call(this)调用。**super.IdentifierName 作为 setter 使用时，super 表示 this。**

- class 语句块内的一般方法（非 static 或 constructor 函数）的定义内，例如：

```
class B extends A {
    foo() {
        super.foo()
    }}
```

- **必须以 super . IdentifierName 的形式出现。super.IdentifierName 作为 getter 时，super 表示该函数 B 的 prototype 属性对象的[[prototype]]**。同样，若通过 super.IdentifierName()来调用函数，则此函数的相当于通过.call(this)调用；**super.IdentifierName 作为 setter 使用时，super 表示 this。**

- 在对象字面量的方法的定义内，例如：

```
let a = {
    foo() {
        super.test()
    }}
let b = {
    test() {
        console.log("I'm b.")
    }}
Object.setPrototypeOf(a, b);a.foo(); // "I'm b."
```

-

必须以 super . IdentifierName 的形式出现，super.IdentifierName 作为 getter 时，super 表示该对象的[[prototype]]。同样，若通过 super.IdentifierName()来调用函数，则此函数的相当于通过.call(this)调用。**super.IdentifierName 作为 setter 使用时，super 表示 this。**