In **TypeScript**, you *can* invoke a static method using **`this`** inside another **static** method — but **`this`** in that context refers to the **constructor function** (the class itself), not an instance.

This is useful when working with **inheritance**, because **`this`** will refer to the *current subclass*, enabling polymorphic static calls.

Example: Calling a static method with `this`

```
class Base {
    static greet(): void {
        console.log("Hello from Base");
    }

    static callGreet(): void {
        // 'this' refers to the class (constructor), not an instance
        this.greet();
    }
}

class Derived extends Base {
    static greet(): void {
        console.log("Hello from Derived");
    }
}

// Call from Base
Base.callGreet();     // Output: Hello from Base

// Call from Derived
Derived.callGreet();  // Output: Hello from Derived
```

Key Points:

1. **`this` in a static method** → refers to the class (constructor), not an object instance.
2. This allows **polymorphic static behavior** — subclasses can override static methods, and **`this`** will call the overridden version.

3. You can also call static methods directly via **`ClassName.methodName()`**.

Example with Parameters and Return Values

```
class MathUtil {
    static square(n: number): number {
        return n * n;
    }

    static logSquare(n: number): void {
        console.log(`Square of ${n} is ${this.square(n)}`);
    }
}

MathUtil.logSquare(5); // Output: Square of 5 is 25
```

✅ **Best Practice**:

Use **`this`** in static methods when you want **inheritance-friendly** calls.

Use **`ClassName.methodName()`** when you want to explicitly reference a specific class.

If you want, I can also show you **how `this` behaves differently in static vs. instance methods** in TypeScript so you can avoid common pitfalls.

Do you want me to prepare that?