In JavaScript, **arrow functions** have a few distinct characteristics compared to regular functions, especially when it comes to their behavior with `this` and `super`.

Key Point: Arrow Functions Do Not Have Their Own `super` or `this`

- **Arrow functions** do not have their own `this` context or `super` keyword. Instead, they inherit `this` and `super` from their surrounding context, meaning the object or class where the arrow function is defined.
- Therefore, you **cannot use `super` directly inside an arrow function** because `super` refers to the parent class, and arrow functions do not create their own `super` context.

Example Breakdown

javascript

```
class Parent {
  greet() {
    console.log('Hello from Parent');
  }
}

class Child extends Parent {
  greet() {
    const arrowGreet = () => {
      super.greet(); // Error: Cannot use 'super' in an arrow function
    };
    arrowGreet();
  }
}

const child = new Child();
child.greet();
```

Why Does This Code Fail?

1. In the `Child` class, we define a method `greet`.
2. Inside the `greet` method, we create an arrow function `arrowGreet`.
3. We attempt to call `super.greet()` inside the arrow function.
4. This throws an **error** because arrow functions don't have their own `super` context. Instead, they use the context from their parent scope (the surrounding function or class).

In this case, the arrow function is trying to access `super`, but it can't because `super` is tied to the `Child` class, not the arrow function.

Correct Approach

To avoid this issue, use a **regular function** instead of an arrow function. Regular functions have their own `this` and `super` context, which allows them to correctly refer to the parent class.

Corrected Code:

javascript

```
class Child extends Parent {
  greet() {
    function regularGreet() {
      super.greet(); // This works because regular functions have their own
`super` context.
    }
    regularGreet.call(this); // Pass the correct `this` context
  }
}

const child = new Child();
child.greet(); // Output: "Hello from Parent"
```

Summary of Notes:

- Arrow functions **inherit** `this` and `super` from their parent scope and do not create their own context for `super`.
- **Super** in JavaScript refers to the parent class and is only valid in regular methods or functions, not in arrow functions.
- Use **regular functions** if you need to access `super` in a class method.