can we use function name in function expression?

Yes, you can use a name in a function expression in JavaScript.

This is often done for clarity, particularly for debugging purposes or for recursion.

When you name a function expression, the name is local only to the function's body. This means it doesn't leak into the outer scope but can be used inside the function itself for recursion or for easier identification in stack traces during debugging.

Example of Named Function Expression

```
const factorial = function fact(n) {
    if (n <= 1) {
        return 1;
    } else {
        return n * fact(n - 1);
    }
};
console.log(factorial(5)); // Outputs: 120
console.log(typeof fact);   // Outputs: undefined
```

In this example,

fact is the name of the function expression assigned to the variable factorial. This name fact is available only within the function itself, allowing it to call itself recursively.

If you try to access fact outside the function, it won't be recognized, as shown by the typeof fact line which outputs undefined.

Benefits of Naming Function Expressions

Debugging: Named function expressions are easier to identify in stack traces. When a function throws an error, its name appears in the stack trace, making it easier to locate the source of the error.

Recursion: As shown in the example above, naming the function within the expression allows it to refer to itself, which is necessary for recursive functions.

Clarity: Naming a function can make code more readable by providing a clear indication of what the function is supposed to do, just like any variable name can help describe the content of the variable.

Thus, using names in function expressions can enhance the functionality and readability of your code, even though these names don't affect the surrounding scope.