```
//SUPER:
//Quando trabalhamos com Orientação a Objeto, geralmente desejamos que os atributos e métodos das nossas Classes sejam alterados de acordo com o objeto que os chama, para isso usamos a palavra reservada "this".
//Porém, irão existir momentos onde desejaremos que os nossos elementos referenciem especificamente aos atributos e métodos da Classe em vez do elemento chamador.
//Isso acontece principalmente quando desejamos usar um método que está dentro da Super classe mas que possuí o mesmo nome do método sobrescrito no objeto chamador, para evitar que ele chame o método mais próximo, temos que usar a palavra reservada "super", irá mostrar ao javascritp que desejamos chamar o método da super classe...


//USANDO SUPER PARA EVITAR QUE UM MÉTODO SOBRESCRITO SEJA CHAMADO:
    const carro = { //Perceba que temos um objeto carro que possuí atributo para...
    velMax: 200, //Velocidade máxima, com o padrão de 200Km/h...
    velAtual: 0, //Velocidade atual comb o padrão e 0...
    aceleraMais(delta) { //Uma função que pega o this atual dela e compara com a velocidade máxima para não deixar ultrapasar
        if (this.velAtual + delta <= this.velMax) {
            this.velAtual += delta
        } else {
            this.velAtual = this.velMax
        }
    },
    status() {//E uma função de status que mostra qual a velocidade em que o carro está e quanto ele pode atingir...
        return `${this.velAtual}Km/h de ${this.velMax}Km/h`
    }

}

//Criamos um objeto ferrari que irá herdar os atributos de métodos de carro, porém, irá sobrescrever o atributo "velMax" assim que a prototipação acontecer...
const ferrari = {
    modelo: 'F40',
    velMax: 324
}

//Temos também um objeto volvo que vai sobrescrever somente o método status do objeto carro quando a prototipação acontecer...
const volvo = {
```

```
    modelo: 'V40',
    status() {
        return `${this.modelo}: ${super.status()}`//Perceba 2 coisas aqui:
        //1º: lembre-se que o objetivo da criação desse método é sobrescrever o já existente no protótipo de carro, então, assim que a herança acontecer, o this irá referenciar ao próprio objeto volvo, que será o this da vez.
        //2º: estamos usando "super" para chamar o próprio status do protótipo, que é o método status do objeto carro. Sempre que usarmos super, vamos referenciar ao elemento pai em uma herança, e não ao objeto chamador. Se colocássemos "this" aqui teríamos um estouro de pilha, pois ele iria chamar ao próprio status do volvo infinitamente...
    }
}

Object.setPrototypeOf(ferrari, carro) //Aqui temos a prototipação de fato com a função "setPrototypeOf()", onde colocamos sempre o objeto que desejamos que ser o herdeiro, e depois o objeto que desejamos que tenha seus elementos herdados...
Object.setPrototypeOf(volvo, carro)

//Veja como a herança ocorreu corretamente entre os objetos e o objeto pai "carro"
ferrari.aceleraMais(300)
console.log('\n1)', ferrari.status())

volvo.aceleraMais(180)
console.log('\n2)', volvo.status()) //Veja como o método status causou uma sobrescrita sobre o status do objeto pai "carro"
```

RESULTADO NO CONSOLE...

```
[Running] node "c:\Users\Almoxarifado\Documents\javascript\arquivos_das_aulas\087-super.js"

1) 300Km/h de 324Km/h

2) V40: 180Km/h de 200Km/h

[Done] exited with code=0 in 0.112 seconds
```