```
//SUPER:

//A função "super()" do javascript é uma função que utilizamos para herdar atributos e métodos de uma classe extendida para a classe atual.

//Veja como podemos usá-la:

//Temos uma classe aqui que define nome e sexo de uma pessoa e possuí uma função que pode retorná-los:
class Pessoa {
    constructor(nome, sexo){
        this.nome = nome
        this.sexo = sexo
    }

    setPessoa(){
        console.log(
        `
        ${this.nome}
        ${this.sexo}
        `
        )
    }
}

//E uma classe Profissão que se aproveita dos dados da classe Pessoa e ainda acrescenta a profissão da pessoa.
class Profissao extends Pessoa{
    constructor(nome, sexo, profissao){ //os parâmetros são os mesmos de Pessoa e ainda recebe mais 1 para a profissão...
        super(nome, sexo) //O super faz com que os atributos de Pessoa sejam herdados
        this.profissao = profissao
    }

    setFuncionario(){ //Veja que podemos referenciá-los normalmente...
        console.log(
        `
        ${this.nome}
        ${this.profissao}
```

```
        ${this.sexo}
        `)
    }
}

//Veja que Paulo pode usar os atributos e métodos de ambas as classes:
let paulo = new Profissao('Paulo', 'Masculino', 'Pedreiro')
paulo.setFuncionario()
/*RESULTADO NO CONSOLE:
Paulo
Pedreiro
Masculino
*/

paulo.setPessoa()
/*RESULTADO NO CONSOLE:
Paulo
Masculino
*/

console.log(paulo)
/*RESULTADO NO CONSOLE:
Profissao { nome: 'Paulo', sexo: 'Masculino', profissao: 'Pedreiro' }
*/




//USANDO SUPER PARA REFERENCIAR A UM MÉTODO DE UMA CLASSE MÃE:
class TesteInterno{ //Perceba que tudo o que há na classe mãe "TesteInterno" é um método chamado "teste"...
    teste(){
        console.log('Eu sou o teste interno!')
    }
}

class TesteExterno extends TesteInterno{ //Na classe "TesteExterno" que herda da classe "TesteInterno" temos também um
```

```
    teste(){                                //método "teste" mas que dentro dele chama diretamente o método "teste" da
        super.teste()                       //classe mãe. OBSERVAÇÃO: super() só pode ser chamado dentro de um médoto
        console.log('E eu sou o teste da classe Teste Externo!!!') //ou função.
    }
}

let testador = new TesteExterno()
testador.teste() //Perceba que quando chamamos o método "teste" ele executa dos 2 testes, da classe mãe e da filha...
/*RESULTADO NO CONSOLE:
Eu sou o teste interno!
E eu sou o teste da classe Teste Externo!!!
*/
```