//PRIVADO E PROTEGIDO:

//O javascript trabalho os atributos e métodos privados e protegidos de uma forma totalmente diferente das linguagens fortemente tipadas.

//Enquanto as linguagens fortemente tipadas forçam o desenvolvedor a usar os elementos privados ou protegidos somente dentro das classes mãe e filhas, no Javascript não existe essa proibição, mas existem convenções de como fazer isso, e quando um desenvolvedor vê essas convenções cabe a ele respeitar elas.

//OBSERVAÇÕES: Está sendo desenvolvido uma notação para realmente impossibilitar o uso de atributos privados fora da sua classe. Essa notação é o "#", usando o hashtag antes do atributo ou classe, alguns navegadores e inclusive o node já aceitam essa notação.

//EXEMPLO DE CLASSE PRIVADA:

class Pessoa {

    constructor(nome, sexo){

        this.\_nome = nome //A convenção é usar o underline na frente do nome, isso evita que digitemos o nome diretamente

        this.\_sexo = sexo //ignorando o underline, isso nos ajuda a lembrar que aquele atributo ou método é privado...

    }

    mostrarPessoaSemUnderlineEThis(){

        console.log(nome, sexo) //Veja que se tentarmos usar os atributos diretamente vamos falhar...

    }

    mostrarPessoaSemUnderline(){

        console.log(this.nome, this.sexo) //Se tentarmos usar sem o underline o resultado será undefined...

    }

    mostrarPessoa(){

        console.log(this.\_nome, this.\_sexo) //Quando usamos com a notação de privado finalmente conseguimos...

    }

    \_nuncaUsarForaDaMaeOuFilhas(){ //Esse método é privado ou protegido, só deve ser usado pela classe mãe ou pelas

        console.log('Nunca me use assim né filho!!!') //filhas dela, pode ser acessada por fora se usar o underline mas

    }                                                 //essa seria uma péssima prática...

}

let margarete = new Pessoa('Margarete', 'F')

margarete.mostrarPessoaSemUnderlineEThis()

/\*RESULTADO NO CONSOLE:

        console.log(nome, sexo)

                    ^

ReferenceError: nome is not defined

\*/

margarete.mostrarPessoaSemUnderline()

/\*RESULTADO NO CONSOLE:

undefined undefined

\*/

margarete.mostrarPessoa()

/\*RESULTADO NO CONSOLE:

Margarete F

\*/

margarete.\_nuncaUsarForaDaMaeOuFilhas() //Nunca use um método ou atributo privado fora de sua classe ou classes filhas...

/\*RESULTADO NO CONSOLE:

Nunca me use assim né filho!!!

\*/

//USANDO ATRIBUTOS E MÉTODOS DE FORMA PROTEGIDA:

//Só relembrando, atributos e métodos protegidos são aqueles que só podem ser acessados pela classe mãe e suas filhas...

class Carro { //Perceba que colocamos todos os atributos e métodos privados ou protegidos...

    constructor(modelo, ano){

        this.\_modelo = modelo

        this.\_ano = ano

    }

    \_partida(){

        console.log(`O carro ${this.\_modelo} está andando...`)

        return this.\_modelo

    }

}

class Sedan extends Carro { //Perceba que a classe "Sedan" utiliza os atributos e métodos da classe mãe de forma

    constructor(modelo, ano){    //protegida...

        super(modelo, ano)

    }

    partida(){ //Temos um próprio método partida para Sedan, perceba que ele utiliza o método protegido "partida" da sua

        console.log(`Esse ${super.\_partida()} é um carro no estilo Sedan.`) //classe mãe, executando esse método e também

    }                                                                       // retornando o valor dele...

}

let gol = new Sedan('Gol', '2009')

gol.partida()

/\*RESULTADO NO CONSOLE:

O carro Gol está andando...

Esse Gol é um carro no estilo Sedan.

\*/

//USANDO A NOTAÇÃO PRIVADA COM O HASHTAG:

class Funcionario {

    //Note que primeiro devemos inicializar os atributos privados dentro da classe..

    #nome

    #id

    constructor(nome, id){ //Usando o construtor ligamos os atributos privados ao construtor da classe usando o "this"

        this.#nome = nome

        this.#id = id

    }

    get nome(){return this.#nome} //Como são privados, ele vão precisar de métodos acessores get e set...

    get id(){return this.#id}

    mostraFuncionario(){

        console.log(`nome: ${this.#nome} | id: ${this.#id}`) //Perceba que podemos acessar os atributos privados

    }                                                           //diretamente usando this, mas isso não é muito legal...

    mostraFuncionarioPrivado(){ //Somos obrigados a acessar os métodos get e set...

        console.log(`nome: ${this.nome} | id: ${this.id}`)

    }

}

let leonardo = new Funcionario('Leonardo', '45612')

console.log(leonardo.nome) //Perceba que podemos acessar o método get até de fora...

/\*RESULTADO NO CONSOLE:

Leonardo

\*/

leonardo.mostraFuncionarioPrivado()

/\*RESULTADO NO CONSOLE:

nome: Leonardo | id: 45612

\*/

leonardo.mostraFuncionario() //Podemos acessar os atributos privados sem um get por através do this, mas isso não é uma

/\*RESULTADO NO CONSOLE:         //boa prática...

nome: Leonardo | id: 45612

\*/