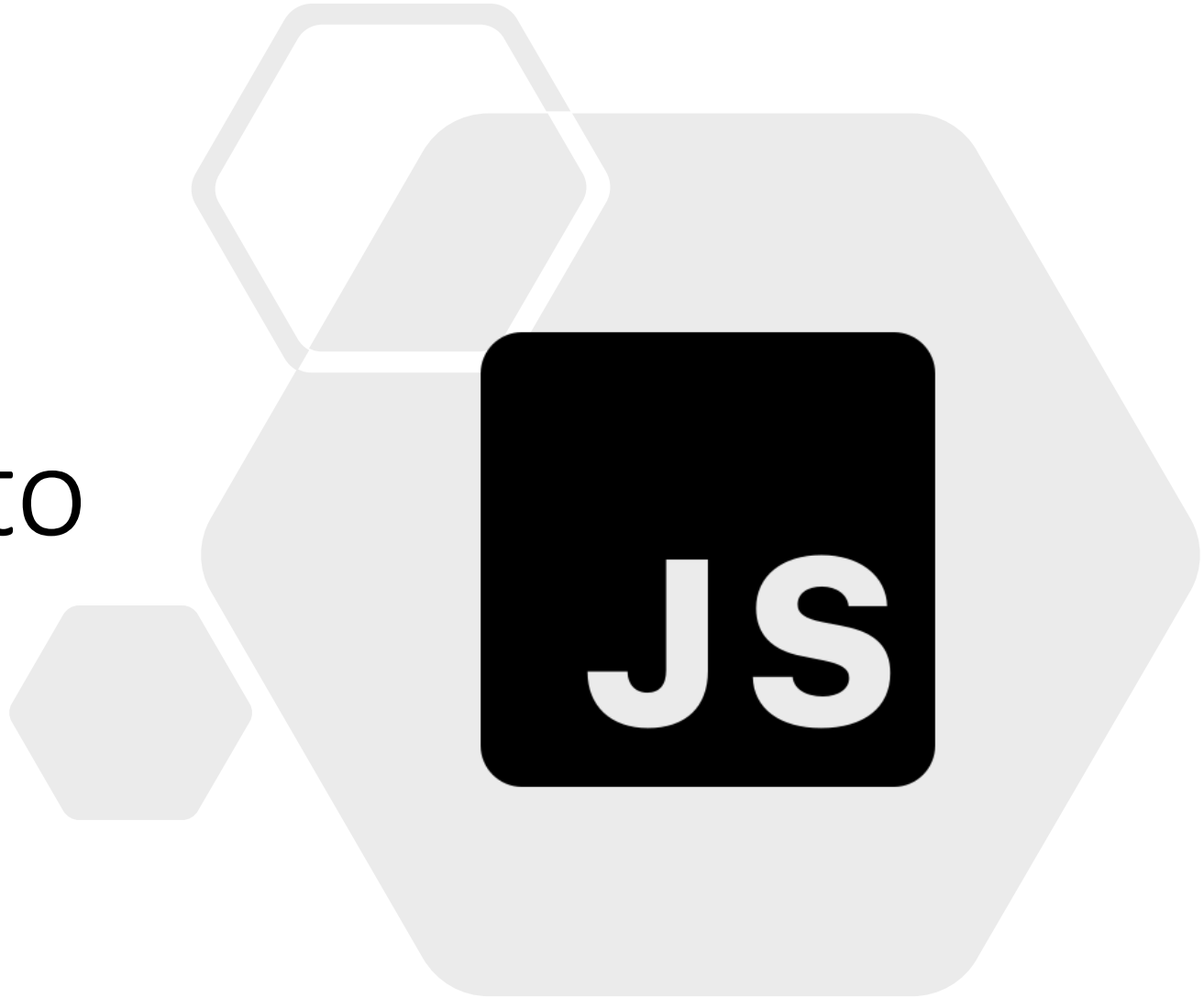
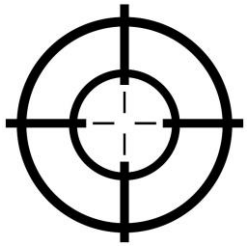
Classes e encapsulamento



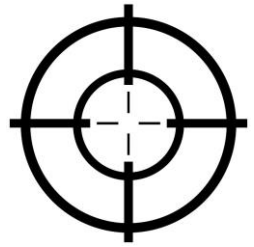
Campos públicos...



Os campos (propriedades) de uma classe são públicos por padrão. Isto significa que podem ser acessados e alterados.

```
class Empresa {
    constructor(razaoSocial, nomeFantasia, cnpj) {
        this.nomeFantasia = nomeFantasia
        this.razaoSocial = razaoSocial
        this.cnpj = cnpj
    }
}

let empresa = new Empresa('ABC LTDA', 'Mercado Online', '999999999')

empresa.cnpj = '888888888'

console.log('Qual o cnpj: ' + empresa.cnpj)
```

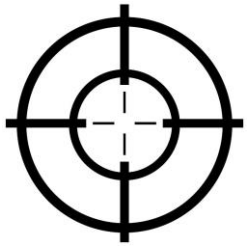
Privar (bloquear) as vezes é necessário...



```
class Empresa {
    #cnpj
    constructor(razaoSocial, nomeFantasia, cnpj) {
        this.nomeFantasia = nomeFantasia
        this.razaoSocial = razaoSocial
        this.#cnpj = cnpj
    }

    get pegarCnpj(){
        return this.#cnpj
    }
}

let empresa = new Empresa('ABC LTDA', 'Mercado Online', '999999999')

console.log('Qual o cnpj: ' + empresa.pegarCnpj)
```

As vezes, não se deseja alteração, direta, para um atributo. Neste caso é necessário privar o acesso com “#”.

Sobre atributos privados...



Os campos privados são declarados com # no nome (diz-se “hash names”), são identificados com o prefixo com #, por isso # é uma parte do próprio nome do atributo.

Os campos privados são acessíveis, diretamente, somente no construtor da classe e dentro da própria declaração da classe.

Privar o acesso a campos é uma medida de segurança, com objetivo de garantir a qualidade de um dado sensível. O nome disto é encapsulamento.

Qual a vantagem do encapsulamento?

O encapsulamento é um mecanismo de restrição do acesso direto a alguns componentes de um objeto, de forma que outros elementos não possam acessar os valores de todas as variáveis de um objeto específico.

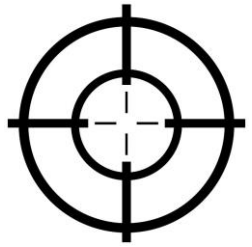
O encapsulamento pode ser usado para ocultar atributos e métodos associados a uma classe ou objeto instanciado.

Métodos também podem ser privados?

```
class Empresa {
    #cnpj
    constructor(razaoSocial, nomeFantasia, cnpj) {
        this.nomeFantasia = nomeFantasia
        this.razaoSocial = razaoSocial
        this.#cnpj = cnpj
    }

    #colocarMaiusculo(texto) {
        return texto.toUpperCase()
    }

    mostrarDetalhes() {
        return 'Nome da empresa: ' + this.#colocarMaiusculo(this.razaoSocial)
            + '\nNome fantasia: ' + this.#colocarMaiusculo(this.nomeFantasia)
    }

    get pegarCnpj() {
        return this.#cnpj
    }
}

let empresa = new Empresa('ABC LTDA', 'Mercado Online', '999999999')

console.log('Detalhes: \n' + empresa.mostrarDetalhes())
```

Encapsular significa acessar com restrição.

Sobre o encapsulamento no JavaScript...



Os atributos privados precisam ser declarados antes de serem usados. Geralmente isto é feito antes de declaração do método construtor.

Não é possível excluir um atributo privado do objeto. Esta operação pode ser aplicada apenas para atributos públicos.

Importante! Em várias linguagens é comum usar métodos get e set para manipular atributos privados.

Atributos ou métodos estáticos

Atributos e métodos estáticos são elementos especiais que não precisam de um objeto para serem usados.

```
class Empresa {
    static telefoneGeral
    constructor(razaoSocial, nomeFantasia, cnpj) {
        this.nomeFantasia = nomeFantasia
        this.razaoSocial = razaoSocial
        this.cnpj = cnpj
    }
}

Empresa.telefoneGeral = '(12) 999999999'

console.log('Telefone da empresa: \n' + Empresa.telefoneGeral)
```

Diz-se que estes elementos pertencem a classe e são usados diretamente a partir do nome dela.

JavaScript