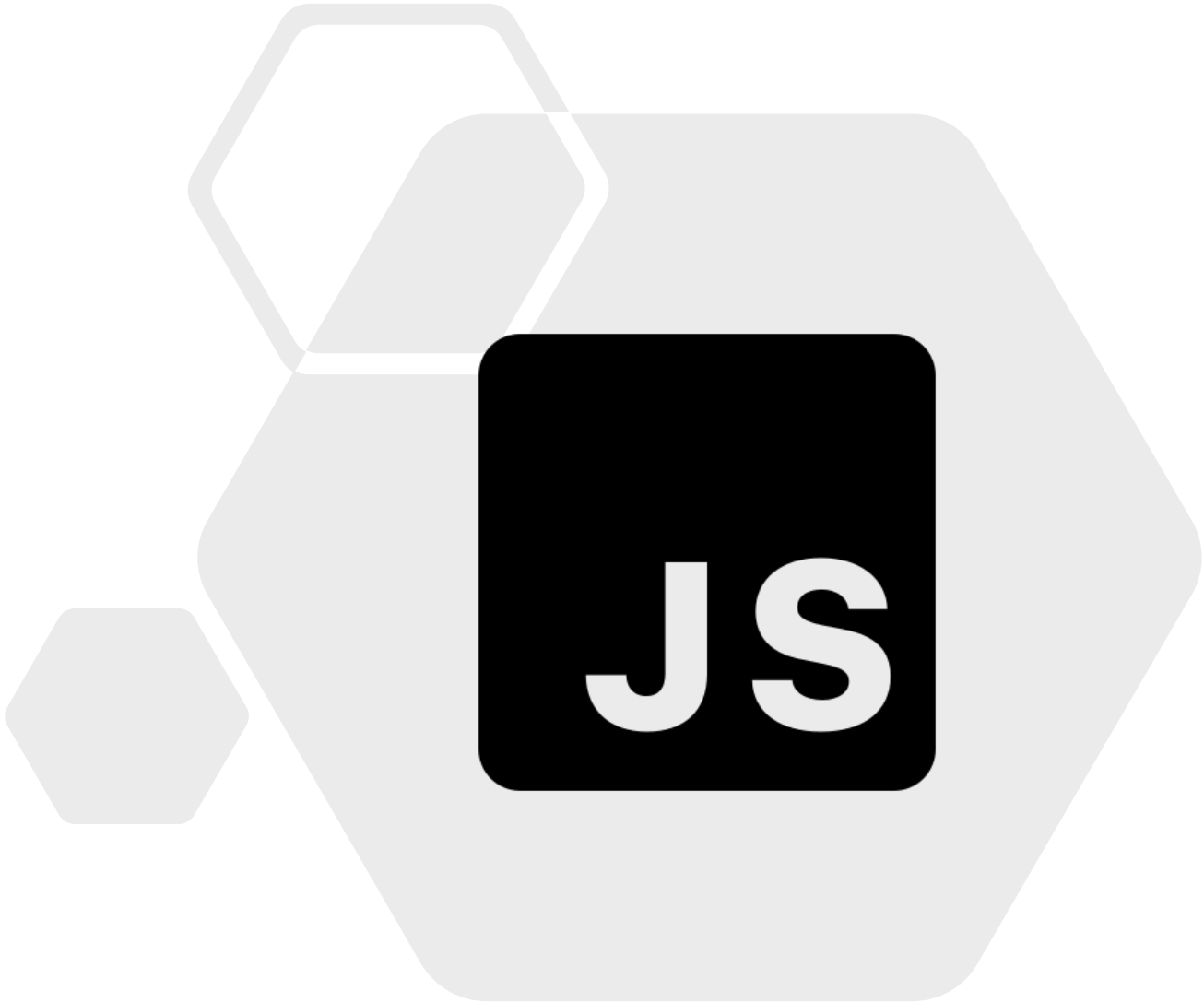
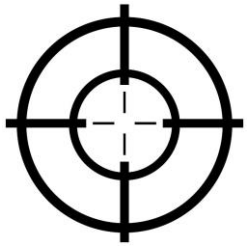
Classes e herança



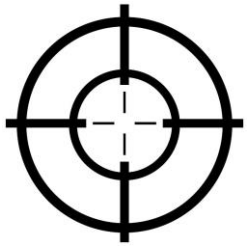
O que são classes?



ECMAScript 2015, também conhecido como ES6, introduziu o conceito classes. As classes são modelos para objetos.

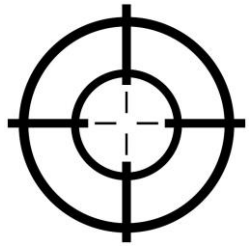
Como analogia: assim como um engenheiro desenha a planta baixa de um imóvel, a classe é a planta baixa do objeto.

Por que usar classes?



A sintaxe de classe é uma forma de escrever classes reutilizáveis, usando uma sintaxe mais fácil e limpa, que é mais semelhante às classes em C++ ou Java.

Como definir uma classe?

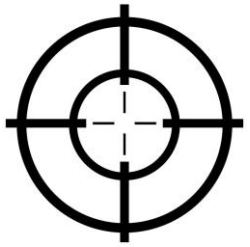


```
class Empresa{
    constructor(nome,razaoSocial){
        this.nome = nome
        this.razaoSocial = razaoSocial
    }
}

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

console.log(empresa.nome)
```

Para definir uma classe usa-se uma palavra reservada, a “class”. A classe é como se fosse uma função especial.

Os objetos são criados a partir de um método chamado de construtor.

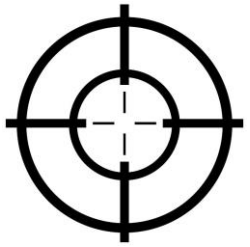
Método construtor...



O método construtor é um método especial para criar e inicializar um objeto. Só pode haver um método especial com o nome “construtor” em uma classe.

Se o método construtor não for definido, o JavaScript adicionará um método construtor vazio, ou seja, um que crie os objetos com suas propriedades “em branco”.

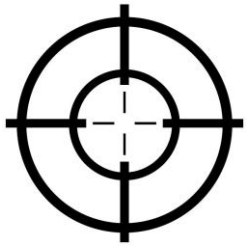
Outra de definir a classe



```
let Empresa = class {
    constructor(nome, razaoSocial) {
        this.nome = nome
        this.razaoSocial = razaoSocial
    }
}

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

console.log(empresa.nome)
```

Outra forma de se definir uma classe é através de uma “class expression”.

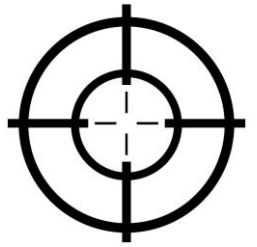
Corpo da classe



O corpo de uma classe é a parte que está entre chaves {}. É aqui que você define os membros da classe, como métodos, atributos ou construtor.

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

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

console.log(empresa.nome)
```

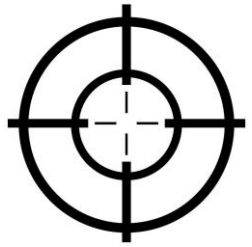
Métodos



Métodos são qualquer função dentro do corpo da classe.
Geralmente, coloca-se métodos para representarem coisas que os objetos devem fazer.

```
let Empresa = class {
    constructor(nome, razaoSocial) {
        this.nome = nome
        this.razaoSocial = razaoSocial
    }

    descricao() {
        return 'Nome: ' + this.nome + ' ' + this.razaoSocial
    }
}

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

console.log(empresa.descricao())
```

Métodos de acesso



A definição de métodos set e get permanece a mesma do padrão ES5.

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

    get obterNome() {
        return this.nome
    }
}

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

console.log(empresa.nome)
```

Class fields

Quando se usa classes, geralmente, se usa o jargão da programação orientada à objetos. Por isso, propriedades são chamadas de campos (class fields).

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

Ao declarar os campos antecipadamente, as definições de classe se tornam mais auto documentáveis.

Começando entender a orientação a objetos

Herança é um princípio de orientação a objetos, que permite que classes compartilhem atributos e métodos, através de “heranças”.

```
class PessoaGeral {
    constructor(nome) {
        this.nome = nome;
    }

    get obterNome() {
        return this.nome;
    }
}
```

```
class PessoaJuridica extends PessoaGeral {
    constructor(nome, razaoSocial) {
        super(nome)
        this.razaoSocial = razaoSocial
    }
}
```

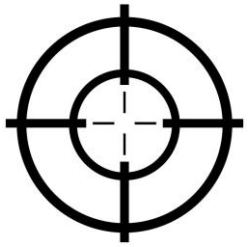
Começando entender a orientação a objetos

Se houver um construtor presente na superclasse, o construtor da subclasse precisa usá-lo também. Por isso o super() antes de usar “this”.

```
class PessoaGeral {
    constructor(nome) {
        this.nome = nome;
    }

    get obterNome() {
        return this.nome;
    }
}
```

```
class PessoaJuridica extends PessoaGeral {
    constructor(nome, razaoSocial) {
        super(nome)
        this.razaoSocial = razaoSocial
    }
}
```

Qual a vantagem da herança?



As principais vantagens da herança são a capacidade de reutilização e a legibilidade do código. Quando a classe filha herda as propriedades e a funcionalidade da classe pai, não precisamos escrever o mesmo código novamente na classe filha. Isso torna mais fácil reutilizar o código, nos faz escrever menos código e o código se torna muito mais legível.

JavaScript