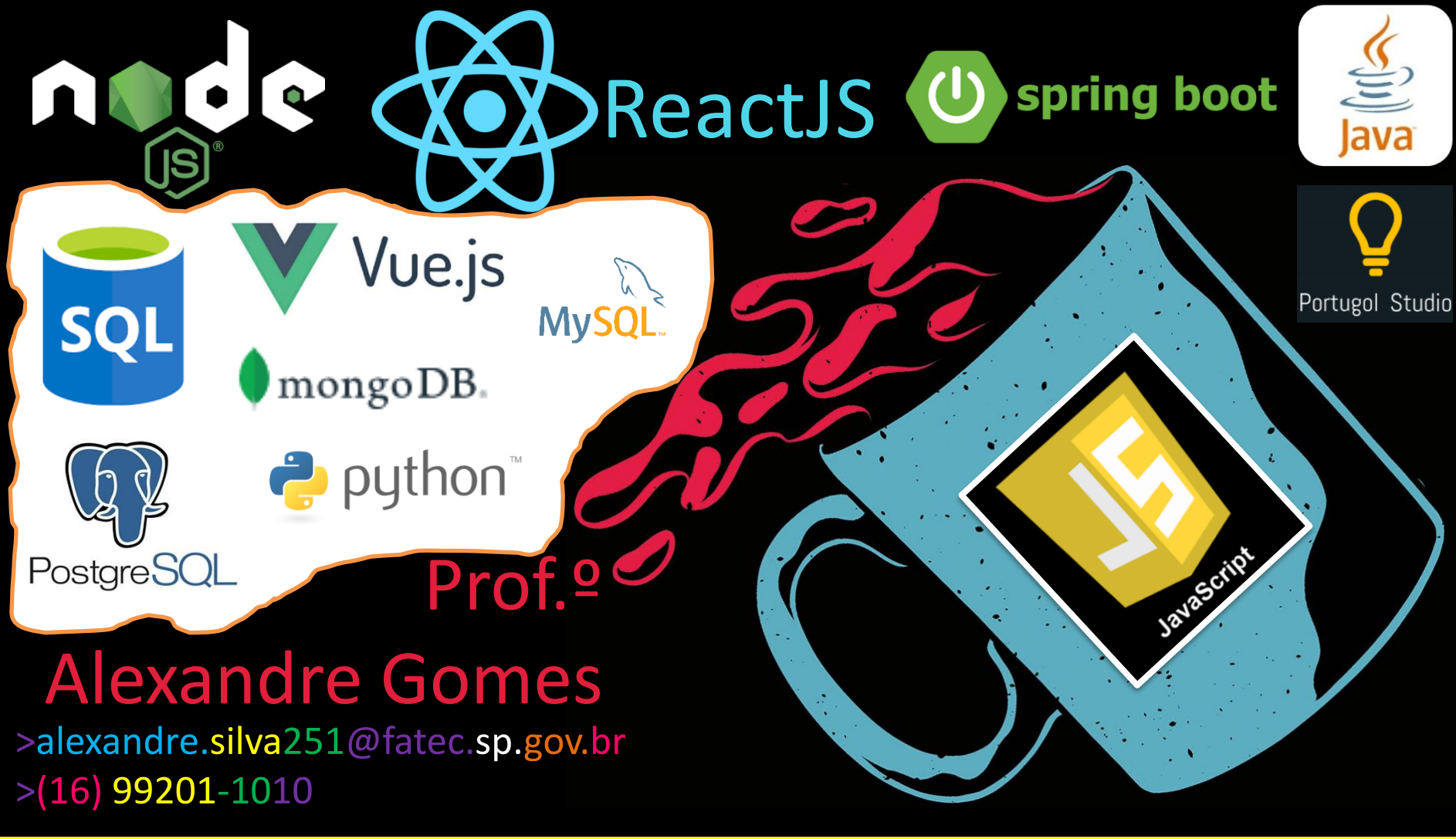
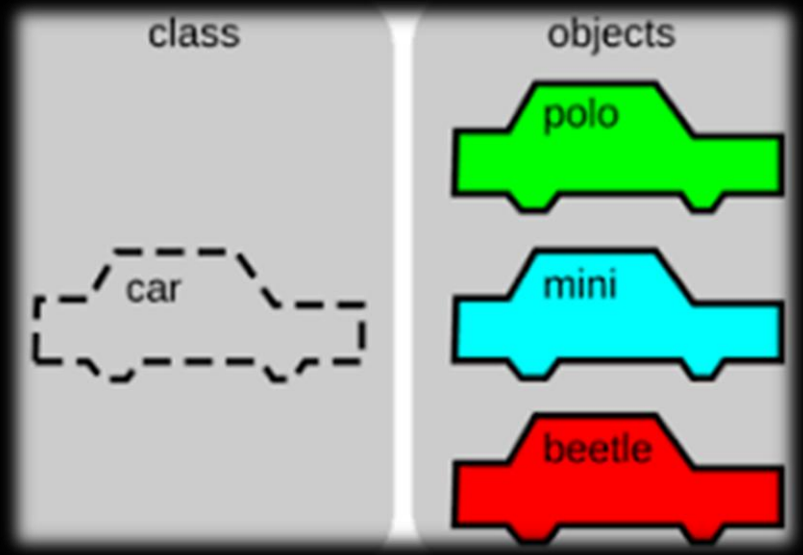

node
JS
ReactJS
spring boot
Java
Portugol Studio
SQL
Vue.js
MySQL
mongoDB.
PostgreSQL
python
JavaScript
Prof.º
Alexandre Gomes
>alexandre.silva251@fatec.sp.gov.br
>(16) 99201-1010

"3º ADS - ED"

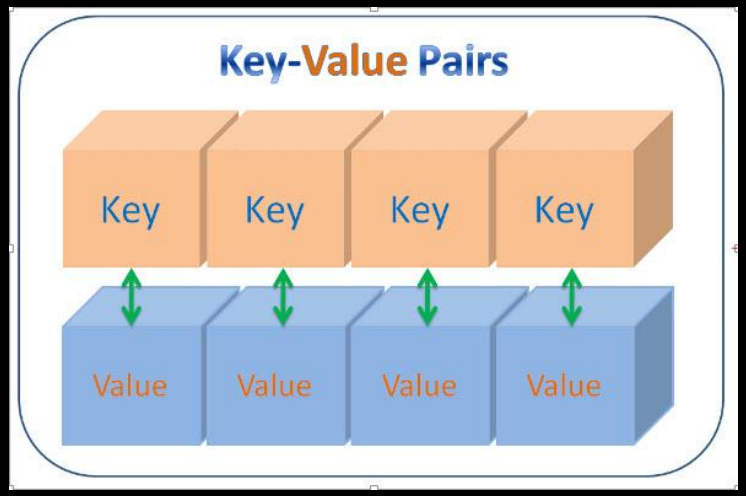
Objetos e Classes



Objetos em Javascript

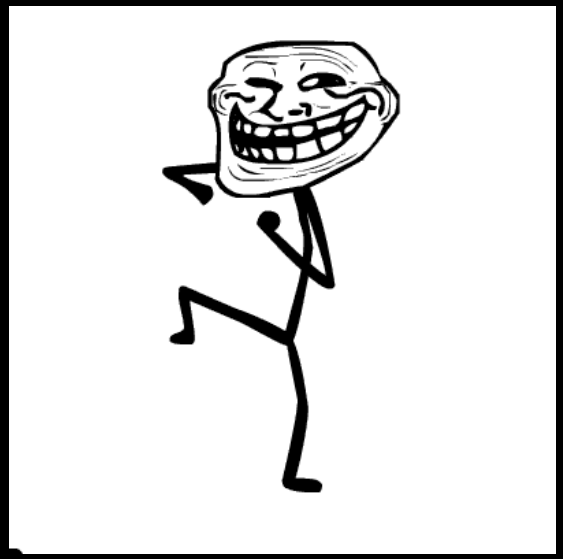
Em Javascript um objeto é uma coleção de propriedades, sendo cada propriedade definida como uma sintaxe de par chave : valor

A chave pode ser uma string e o valor pode ser qualquer informação.



Para criar um objeto, use a sintaxe literal de objeto { }.
Por exemplo, o exemplo a seguir cria um objeto vazio:

```
let empty = {};
```



Para criar um objeto com propriedades, use a sintaxe chave : valor. Por exemplo, para criar um objeto person com as propriedades firstName e lastName, faça da seguinte forma:

```
let person = {
    firstName: 'John',
    lastName: 'Doe'
};
```

PROPRIEDADE (1)

PROPRIEDADE (2)

CHAVE

VALOR

O objeto person possui duas propriedades:

- A primeira propriedade, identificada pela chave firstName e que possui como valor a string 'John'.
- A segunda propriedade, identificada pela chave lastName e que possui como valor a string 'Doe'.

Acessando as propriedades:

Para acessar uma propriedade de um objeto, use uma das duas notações (sintaxes):

- Notação de ponto
- Notação de array

A notação de ponto (.)

O exemplo mostra como usar a notação de ponto para acessar uma propriedade de um objeto:

```
objectName.propertyName
```

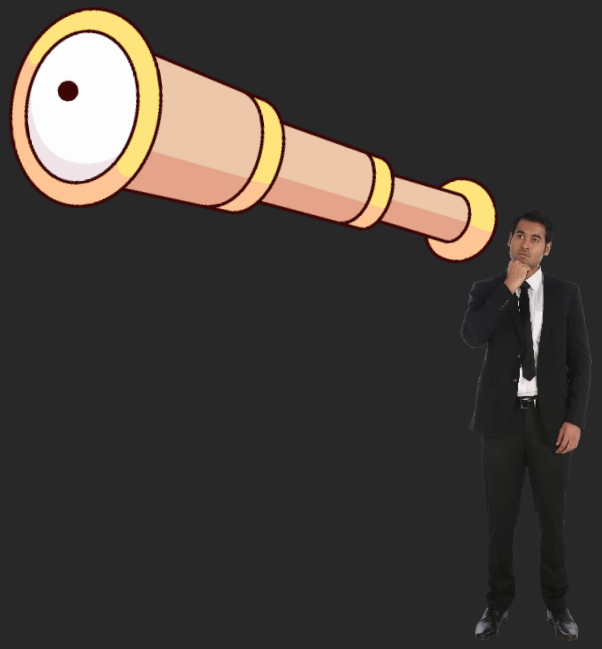
Por exemplo, para acessar a propriedade firstName do objeto person, use a seguinte expressão:

```
person.firstName
```

O trecho abaixo cria um objeto de person e mostra o nome e o sobrenome no console:

```
let person = {
    firstName: 'John',
    lastName: 'Doe'
};


console.log(person.firstName);
console.log(person.lastName);
```



A notação do tipo Array [ ]

O exemplo mostra como acessar o valor da propriedade de um objeto usando colchetes, como na notação de Array:

```
objectName['propertyName'];
```

O trecho abaixo cria um objeto de person e mostra o nome e o sobrenome no console:

```
let person = {
    firstName: 'John',
    lastName: 'Doe'
};


console.log(person['firstName']);
console.log(person['lastName']);
```

Quando um nome de propriedade contém espaços, você precisa colocá-lo entre aspas, por exemplo:

```
let address = {
    'building no': 3960,
    street: 'North 1st street',
    state: 'CA',
    country: 'USA'
};
```

Para acessar o **'building no'** você deve usar obrigatoriamente a notação de Array (**usar aspas e colchetes**):

```
address['building no'];
```

Se você usar a notação de **ponto**, será gerado um erro:

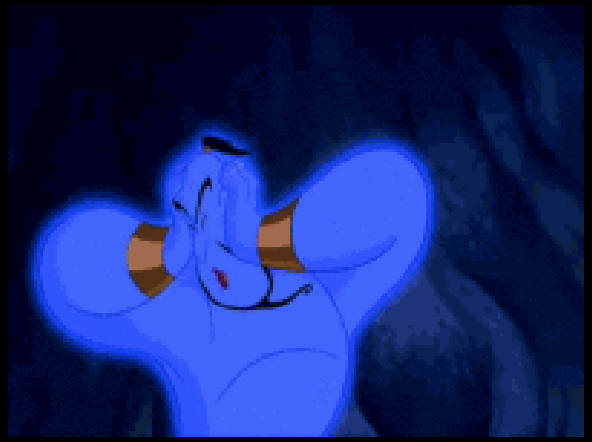
```
SyntaxError: Unexpected string
```

Tentar ler uma **propriedade** que não existe resultará em um **undefined**. Por exemplo:

```
console.log(address.district);
```

Resultado:

```
undefined
```



Alterar o valor de uma propriedade:

Para alterar o valor de uma propriedade, use o operador de atribuição. Por exemplo:

```
let person = {
    firstName: 'John',
    lastName: 'Doe'
};

person.firstName = 'Jane';

console.log(person);
```

```
{ firstName: 'Jane', lastName: 'Doe' }
```

Adicionar uma nova propriedade em um objeto:

Ao contrário de objetos em outras linguagens de programação, como Java ou c#, você pode adicionar uma propriedade a um objeto depois de criá-lo.

A declaração a seguir, adiciona a propriedade **age** ao objeto **person** e atribui **25** a ela:

```
person.age = 25;
```

Excluir uma propriedade de um objeto:

Para excluir uma propriedade de um objeto, use o operador delete

O exemplo a seguir remove a propriedade age do objeto person:

```
delete person.age;
```

Verificar se a propriedade existe:

Para verificar se uma **propriedade** existe em um **objeto**, use o operador **in**:

```
propertyName in objectName
```

O exemplo a seguir cria um objeto **employee** e usa o operador **in** para verificar se as propriedades **ssn** e **employeeId** existem no objeto:

```
let employee = {
    firstName: 'Peter',
    lastName: 'Doe',
    employeeId: 1
};

console.log('ssn' in employee);
console.log('employeeId' in employee);
```

Resultado:

```
false
true
```

Iterar sobre o objeto usando loop for...in:

Para iterar sobre todas as propriedades de um objeto sem saber os nomes das propriedades, use o loop for...in:

```
for(let key in object) {

    // ...

};
```

Por exemplo, a instrução a seguir cria um objeto website e itera sobre suas propriedades usando o loop for...in:

```
let website = {
    title: 'JavaScript Tutorial',
    url: 'https://www.javascripttutorial.net',
    tags: ['es6', 'javascript', 'node.js']
};

for (const key in website) {
    console.log(website[key]);
}
```

Resultado:

```
JavaScript Tutorial
https://www.javascripttutorial.net
[ 'es6', 'javascript', 'node.js' ]
```

Métodos

Objetos possuem ações. As ações são representadas por funções. O seguinte trecho adiciona a ação greet ao objeto person:

```
let person = {
    firstName: 'John',
    lastName: 'Doe'
};

person.greet = function () {
    console.log('Hello, World!');
}
person.greet();
```

Resultado:

```
Hello, World!
```

No exemplo a seguir, adicionamos uma expressão de **função** para criar a **ação** e atribuímos à propriedade **greet** do objeto **person**. Então, chamamos a função através da propriedade **greet** com o comando **greet()**. Quando uma função é uma propriedade de um **objeto**, isso é chamado de **método**. Além de usar uma expressão de **função**, você pode definir uma **função** e adicioná-la ao **objeto**, dessa forma:

```
let person = {
    firstName: 'John',
    lastName: 'Doe'
};

function greet() {
    console.log('Hello, World!');
}

person.greet = greet;

person.greet();
```

Método Abreviado

Você pode definir métodos usando a sintaxe literal do objeto:

```
let person = {
    firstName: 'John',
    lastName: 'Doe',
    greet: function () {
        console.log('Hello, World!');
    }
};
```

Método Abreviado

A partir da versão ES6, você pode tornar o código ainda mais curto:

```
let person = {
    firstName: 'John',
    lastName: 'Doe',
    greet() {
        console.log('Hello, World!');
    }
};


person.greet();
```

O valor "this“

Normalmente os **métodos** precisam acessar os dados armazenados no **objeto**.

Por exemplo, você pode desenvolver um **método** que retorne o **nome completo** do objeto **person**, concatenando **firstName** e **lastName**.

Dentro do **método**, o valor **this** faz referência ao **objeto** que contém o **método** para que você possa acessar uma **propriedade do objeto** usando a notação de **ponto**.

```
this.propertyName
```

O exemplo a seguir usa o valor this no método getFullName():

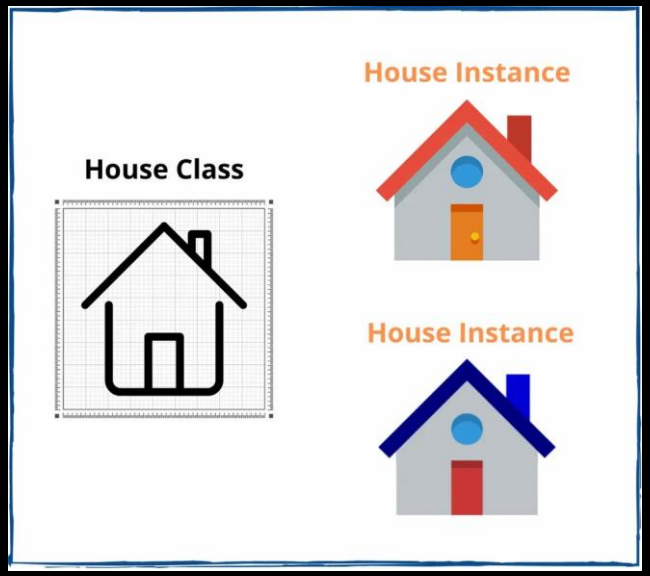
```
let person = {
    firstName: 'John',
    lastName: 'Doe',
    greet: function () {
        console.log('Hello, World!');
    },
    getFullName: function () {
        return this.firstName + ' ' + this.lastName;
    }
};



console.log(person.getFullName());
```

Classes em Javascript

Classes em Javascript proveem uma maneira mais simples e clara para criar objetos, lidar com herança e outras facilidades próprias de Programação Orientada a Objetos.



Robô

Vamos pensar na entidade robô.

Caso você tenha apenas imaginado um robô abstrato, sem forma física, que é uma máquina capaz de tomar decisões por si mesma, então você agiu corretamente. Se você imaginou um robô físico, como um robô na cor azul que anda sobre esteira, com dois braços e uma antena, então você não pensou no conceito de robô, mas sim em uma manifestação física possível de robô.

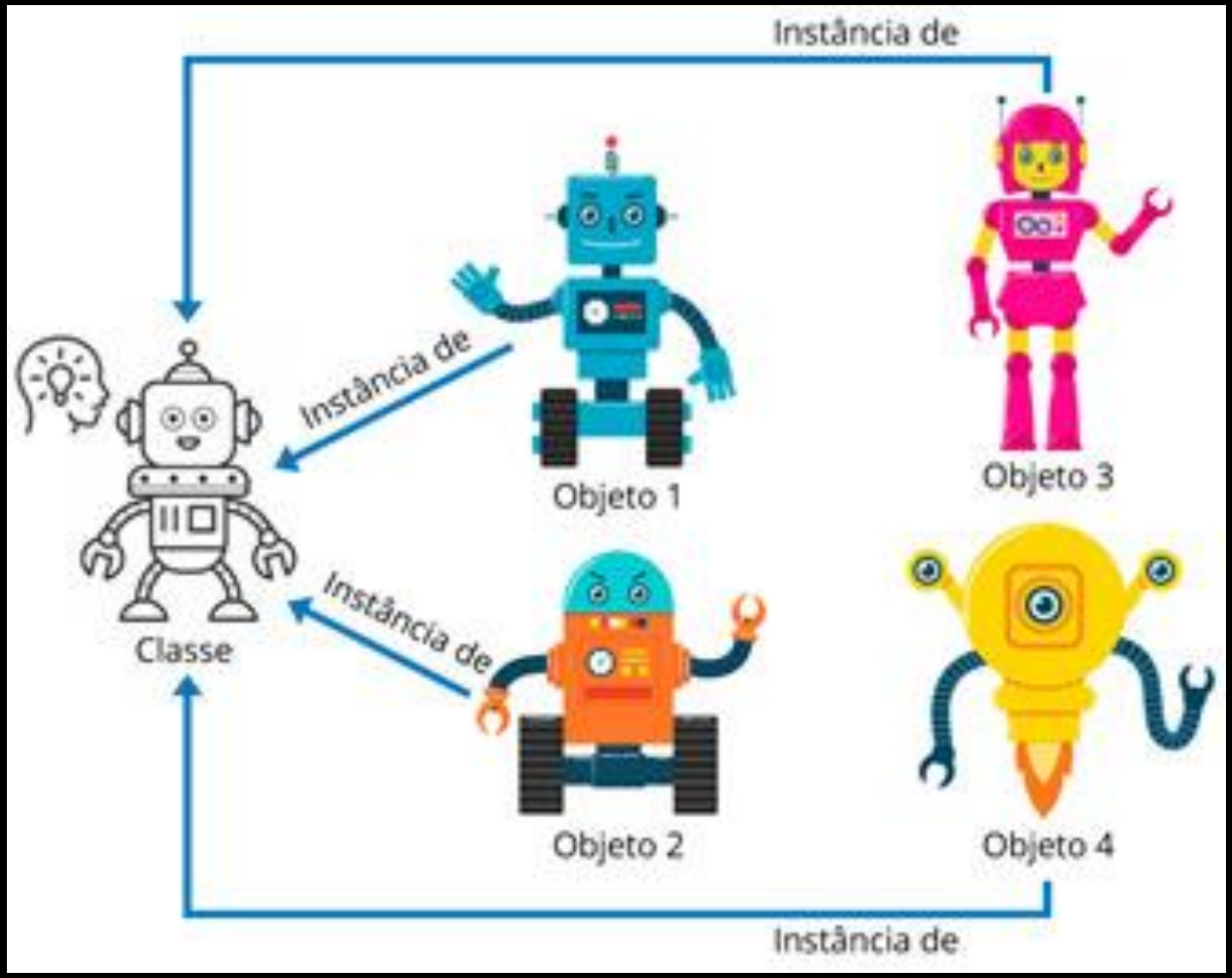
Classes:

A ideia de robô é **abstrata**, assim como são as **classes**.

Uma boa forma de compreender uma **classe** é pensar nela como o **projeto** ou a **modelagem** de algo.

Caso tenha imaginado algo **concreto**, então você pensou em um **objeto**, que é uma manifestação possível da **ideia de robô**, semelhantemente à ideia de **objeto** na programação.

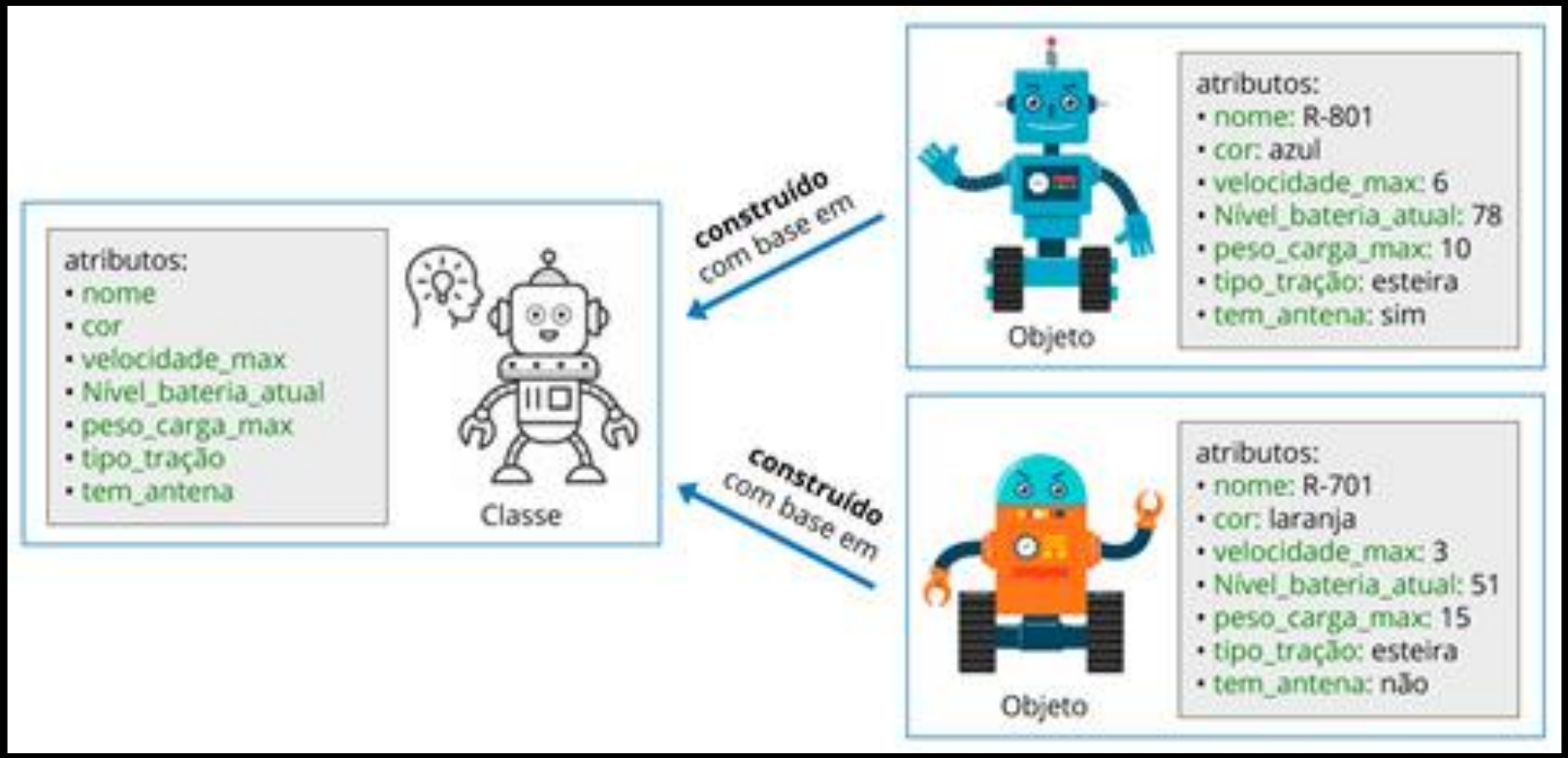
Classe e suas instâncias:



Atributos:

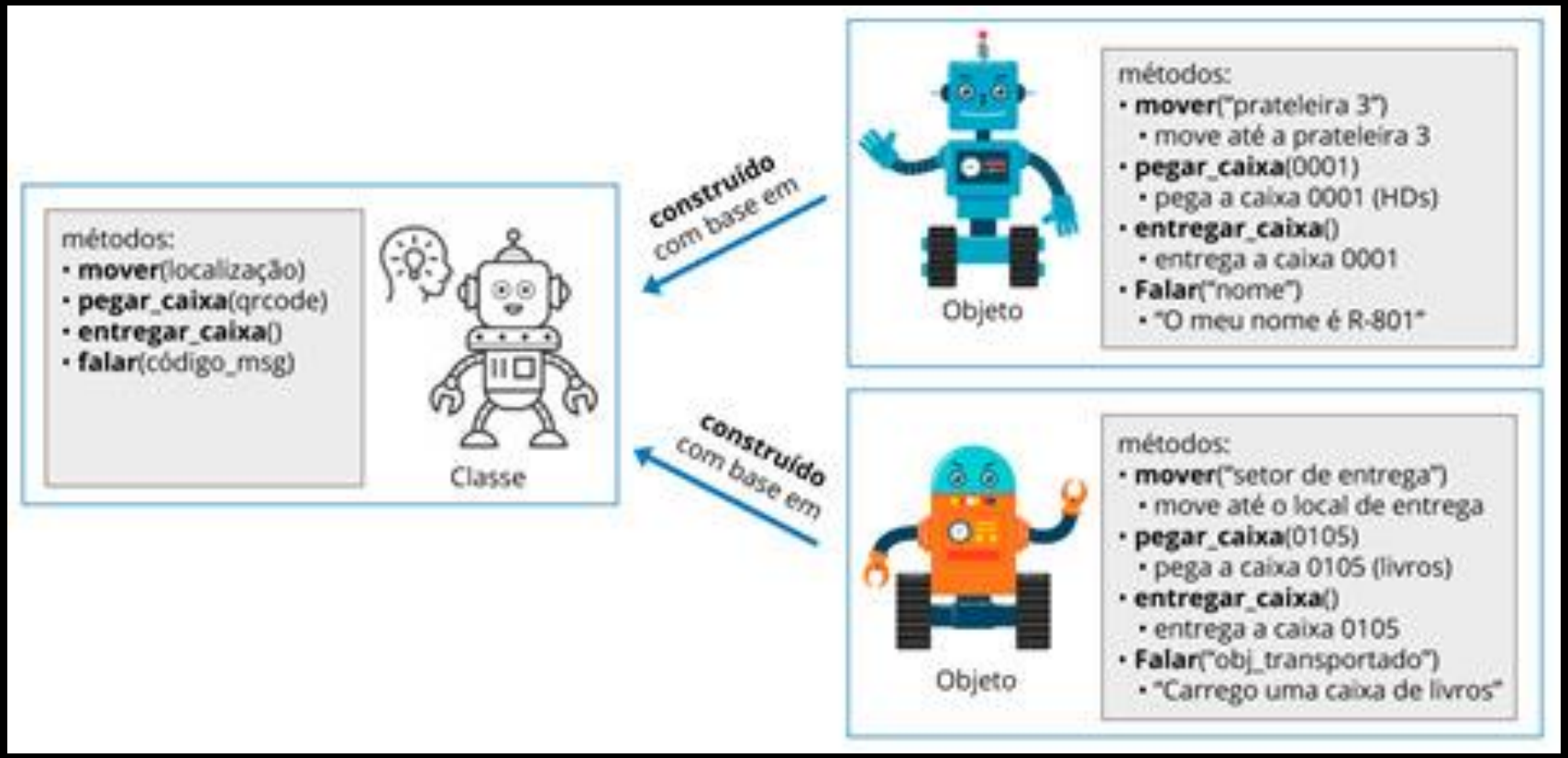
Os principais elementos que compõem uma **classe** são os **atributos** e os **métodos**.

Um **atributo** é um elemento que representa as características da **classe**.



Método:

Um outro elemento importante de uma classe são os métodos. Um método dá ao objeto da classe a capacidade de executar algum tipo de ação, comportamento ou processamento. Um robô é capaz de executar uma série de ações.



Classes, objetos, métodos:

Classe Carro (marca, modelo, cor, combustível)

Objeto carro A (Fiat, punto, branco, flex)
Objeto carro B (Ford, Ka, vermelho, gasolina)

Métodos: ligar, acelerar, frear

Classe Animal (tamanho, peso, raça)

Métodos: comer(), dormir(), caçar()

Objeto felino extends Animal
Método caçar()

Referências

https://www.javascript.com/
https://www.javascripttutorial.net/
https://developer.mozilla.org/pt-BR/docs/Web/JavaScript
https://www.devmedia.com.br/classes-no-javascript/23866

Bora codar VsCode