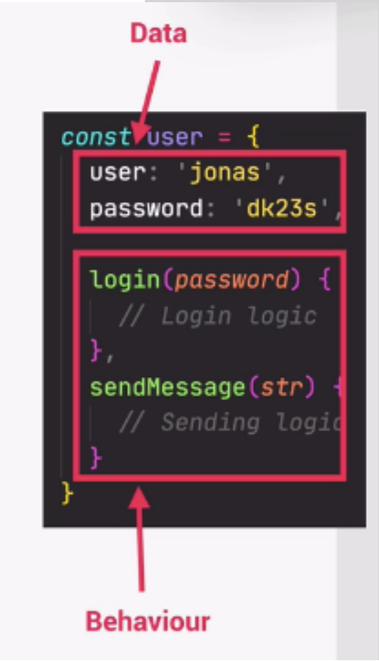
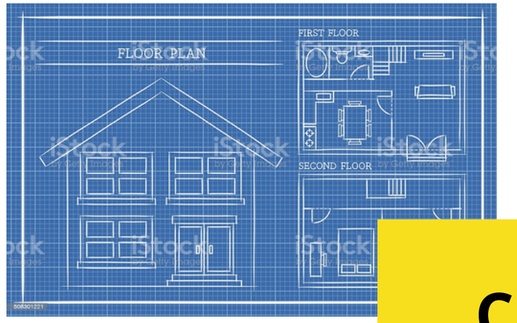
O QUE É PROGRAMAÇÃO ORIENTADA A OBJETOS? (OOP)



- Programação orientada a objetos (OOP) é um paradigma de programação baseado no conceito de objetos
- Nós usamos objetos para **modelar** (descrever) funcionalidades do mundo real ou funcionalidades abstratas.
- Objetos podem conter dados (propriedades) e código (métodos). Usando objetos, nós podemos empacotar **dados e seu comportamento correspondente** em um bloco
- Em OOP, objetos são blocos de código **independentes**.
- Objetos são **blocos de construção** de aplicações, e **interagem** entre si;
- OOP foi desenvolvido com o objetivo de **organizar** o código, tornar ele **mais flexível e mais fácil de manter**.

{LET'S CODE}

CLASSES E INSTÂNCIAS (OOP TRADICIONAL)



CLASSE

```
User {
  user
  password
  email

  login(password) {
    // Login logic
  }
  sendMessage(str) {
    // Sending logic
  }
}
```

new User('jonas')

new User('steven')

Instância

```
{
  user = 'jonas'
  password = 'dk23s'
  email = 'hello@jonas.io'

  login(password) {
    // Login logic
  }
  sendMessage(str) {
    // Sending logic
  }
}
```

Instância

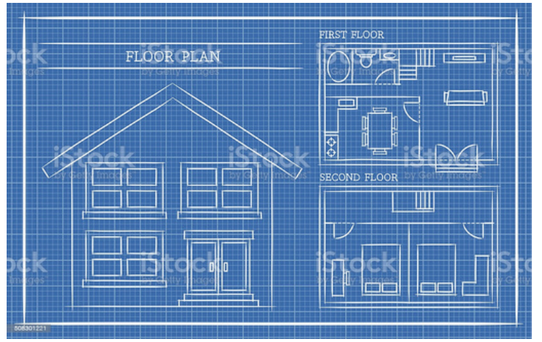
```
{
  user = 'steven'
  password = '5p8dz32dd'
  email = 'steven@tes.co'

  login(password) {
    // Login logic
  }
  sendMessage(str) {
    // Sending logic
  }
}
```

{LET'S CODE}

OS 4 PRINCÍPIOS FUNDAMENTAIS

"Como nós fazemos o design de uma classe? Como nós modelamos dados reais em classes?" 🤔

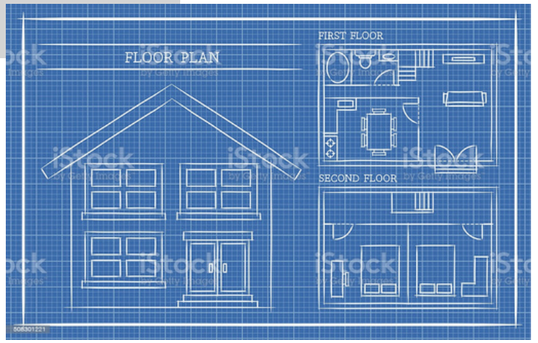


{LET'S CODE}

OS 4 PRINCÍPIOS FUNDAMENTAIS

- **Abstração**
- **Encapsulamento**
- **Herança**
- **Polimorfismo**

"Como nós fazemos o design de uma classe? Como nós modelamos dados reais em classes?" 🤔



PRINCÍPIO 1: ABSTRAÇÃO

Abstração

Encapsulamento

Herança

Polimorfismo

```
Phone {
  charge
  volume
  voltage
  temperature

  homeBtn() {}
  volumeBtn() {}
  screen() {}
  verifyVolt() {}
  verifyTemp() {}
  vibrate() {}
  soundSpeaker() {}
  soundEar() {}
  frontCamOn() {}
  frontCamOff() {}
  rearCamOn() {}
  rearCamOff() {}
}
```

Abstração: Ignorar ou esconder detalhes que **não importam**, nos permitindo ter uma perspectiva geral do que estamos implementando, ao invés de mexer com detalhes que não importam para nossa implementação

{LET'S CODE}

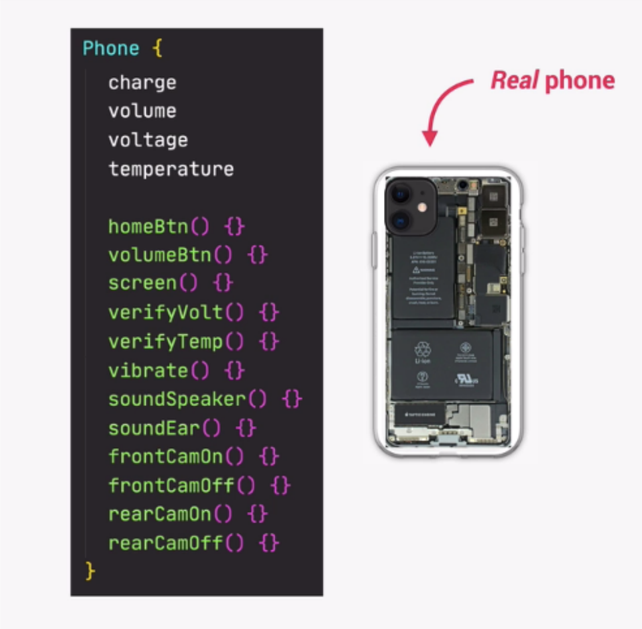
PRINCÍPIO 1: ABSTRAÇÃO

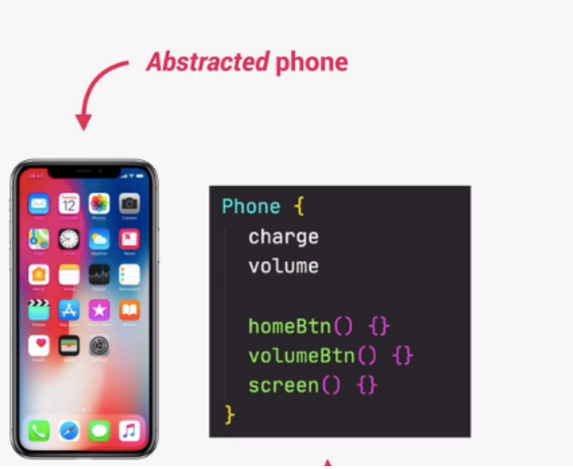
Abstração

Encapsulamento

Herança

Polimorfismo





Abstração: Ignorar ou esconder detalhes que **não importam**, nos permitindo ter uma perspectiva geral do que estamos implementando, ao invés de mexer com detalhes que não importam para nossa implementação

{LET'S CODE}

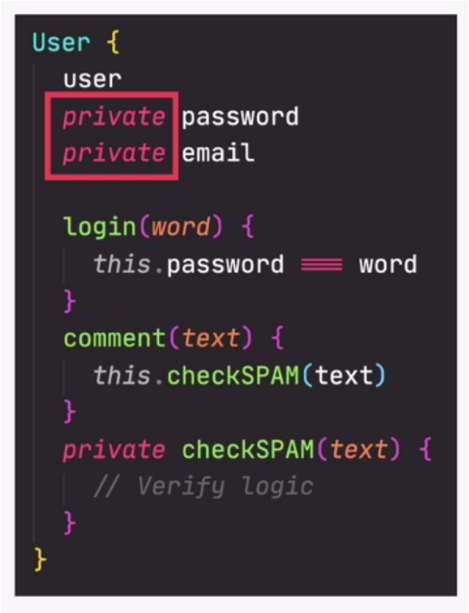
PRINCÍPIO 2: ENCAPSULAMENTO

Abstração

Encapsulamento

Herança

Polimorfismo



Encapsulamento: Manter propriedades e métodos privados dentro de uma classe de forma que eles **não sejam acessíveis fora da classe**.

{LET'S CODE}

PRINCÍPIO 3: HERANÇA

- Abstração
- Encapsulamento
- Herança
- Polimorfismo

```
User {
  user
  password
  email

  login(password) {
    // Login logic
  }
  sendMessage(str) {
    // Sending logic
  }
}
```

```
Admin {
  user
  password
  email
  permissions

  login(password) {
    // Login logic
  }
  sendMessage(str) {
    // Sending logic
  }
  deleteUser(user) {
    // Deleting logic
  }
}
```

{LET'S CODE}

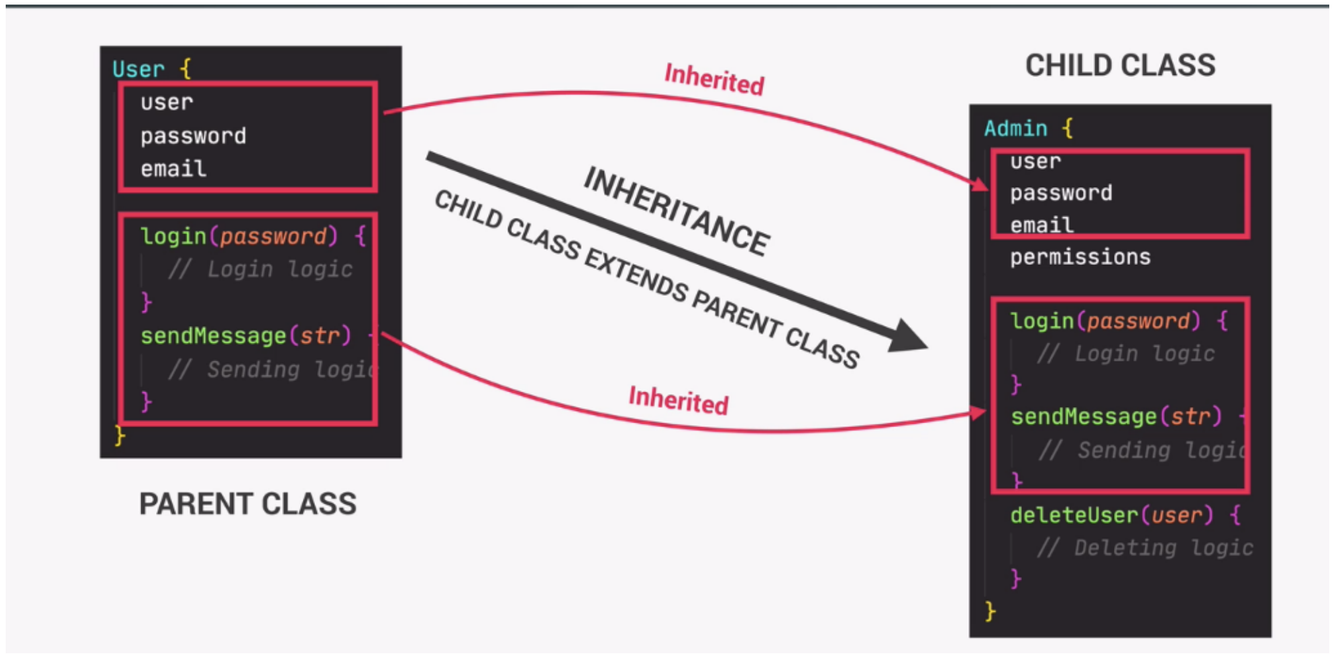
PRINCÍPIO 3: HERANÇA

Abstração

Encapsulamento

Herança

Polimorfismo



Herança: Manter todas as propriedades e métodos de uma certa classe acessíveis a uma classe filha, formando uma relação hierarquica entre classes. Isso nos permite usar lógicas comuns e modelar relações reais

{LET'S CODE}

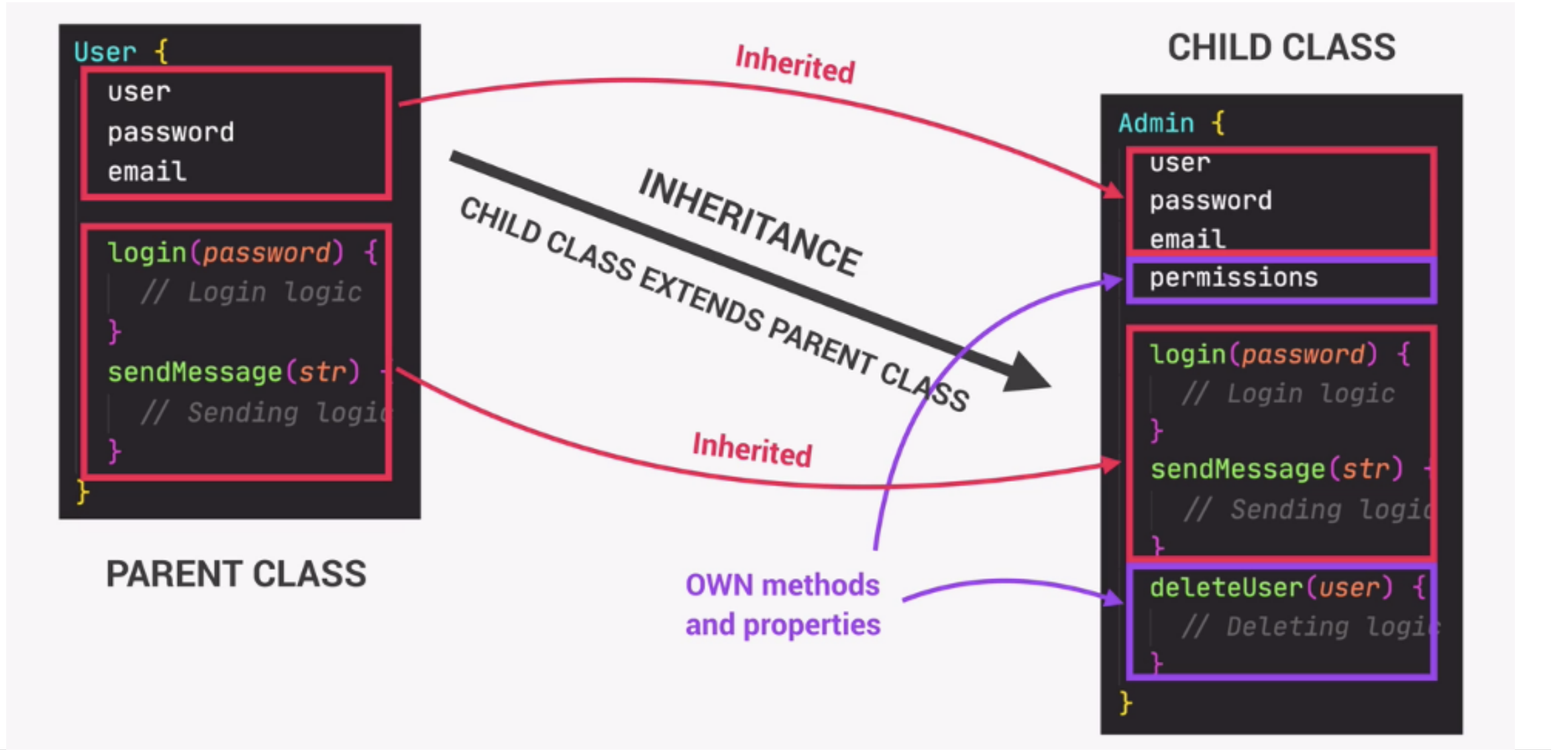
PRINCÍPIO 3: HERANÇA

Abstração

Encapsulamento

Herança

Polimorfismo

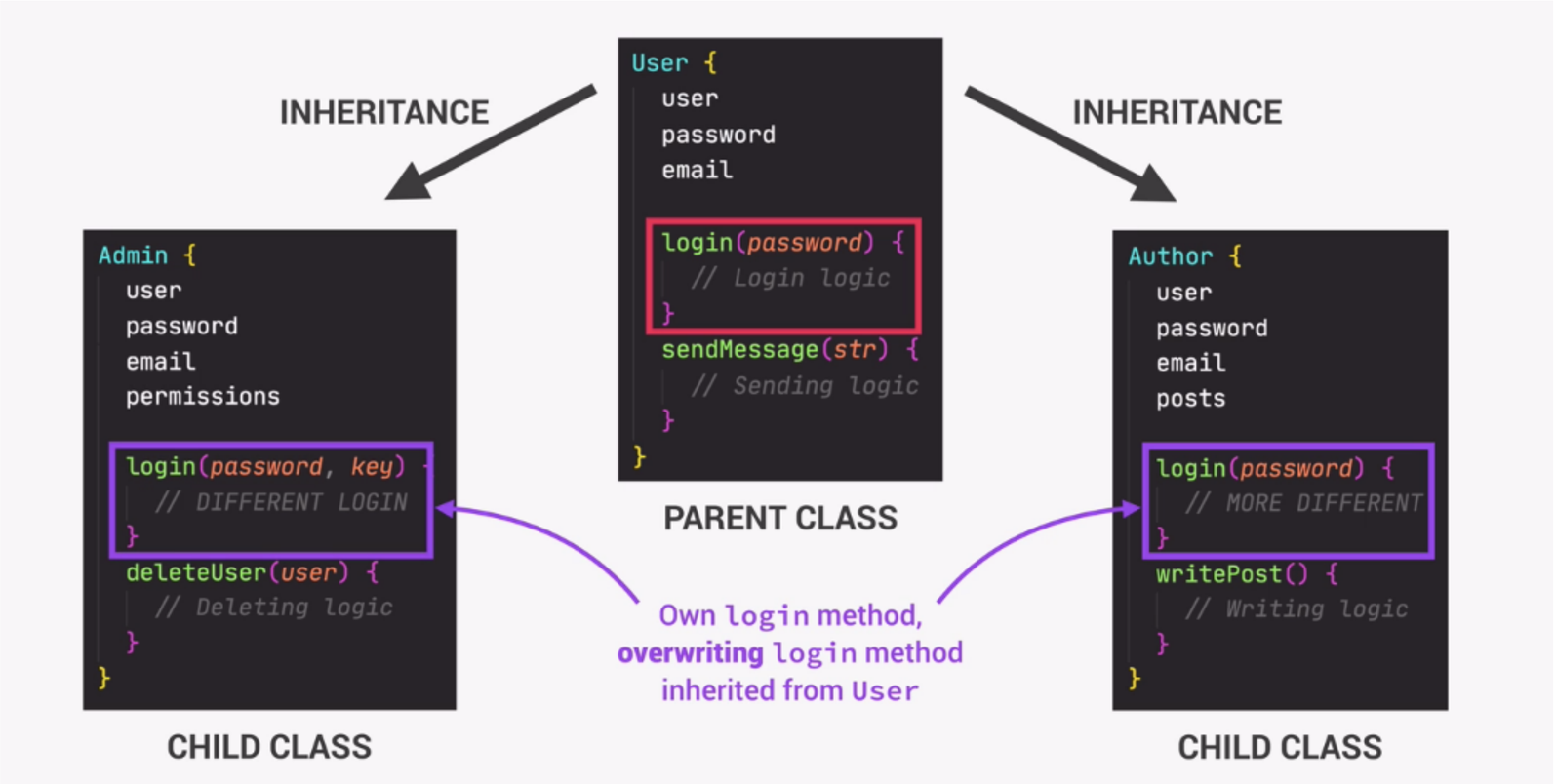


Herança: Manter todas as propriedades e métodos de uma certa classe acessíveis a uma classe filha, formando uma relação hierarquica entre classes. Isso nos permite usar lógicas comuns e modelar relações reais

{LET'S CODE}

PRINCÍPIO 4: POLIMORFISMO

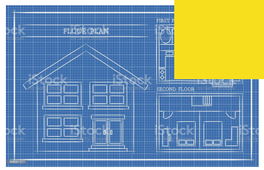
- **Abstração**
- **Encapsulamento**
- **Herança**
- **Polimorfismo**



Polimorfismo: Uma classe filha pode sobrescrever um método que foi herdado de uma classe pai.

OOP EM JAVASCRIPT: PROTOTYPES

"OOP CLÁSSICO": CLASSES



CLASSE

↓

INSTÂNCIA

Objetos (instâncias) são **instanciados** de uma classe.

"OOP EM JAVASCRIPT": PROTOTYPES

PROTOTYPE

↑

OBJETO

Objetos são **conectados** ao objeto prototype

herança prototípica: O protótipo contém métodos que são **acessíveis a todos os objetos conectados com aquele protótipo**.

O comportamento é **delegado** ao objeto prototype conectado