📘 ESTUDO COMPLETO — LOOPS `for` EM JAVASCRIPT

Loops servem para **repetir um bloco de código** enquanto uma condição for verdadeira.
Em JavaScript, os principais são:

- `for` (tradicional)
- `for...in`
- `for...of`

Cada um existe para **um propósito diferente**.

1️⃣ FOR TRADICIONAL (`for`)

📌 O que é

O `for` tradicional é usado quando você:

- Precisa de **controle total** do loop
- Precisa do **índice**
- Quer definir **início, condição e incremento**

📐 Sintaxe

```
for (inicialização; condição; incremento) {
  // código a ser executado
}
```

🔍 Como funciona

Parte	Função
Inicialização	Executa **uma vez** antes do loop
Condição	Testada antes de cada repetição
Incremento	Executado após cada repetição

☑️ Exemplo básico

```
for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

🧠 Quando usar

- Quando precisa do **índice**
- Quando precisa **pular**, **voltar**, **quebrar** o fluxo
- Quando precisa de **lógica mais complexa**

⚠️ Observação sobre escopo

```
let x = 10;

for (let x = 0; x < 3; x++) {
  console.log(x);
}

console.log(x); // 10
```

➡️ `let` respeita escopo de bloco
➡️ `var` **não** respeita (evite usar)

2️⃣ ITERÁVEIS — CONCEITO FUNDAMENTAL

📌 O que é um iterável?

Um **iterável** é qualquer estrutura que pode ser percorrida elemento por elemento.

☑️ Exemplos de iteráveis

- Arrays (`[]`)
- Strings (`"texto"`)
- Maps
- Sets
- NodeLists

❌ Não são iteráveis

- Objetos literais `{}` (por padrão)

🔍 Por que isso importa?

Porque:

- `for...of` **só funciona com iteráveis**
- `for...in` funciona com **objetos**

3️⃣ FOR...OF

📌 O que é

Usado para **percorrer valores** de um iterável.

📐 Sintaxe

```
for (const valor of iteravel) {
  // código
}
```

☑️ Exemplo com array

```
const carros = ["BMW", "Volvo", "Ford"];

for (const carro of carros) {
  console.log(carro);
}
```

☑️ Exemplo com string

```
for (const letra of "JavaScript") {
  console.log(letra);
}
```

🧠 Quando usar

- Quando quer **os valores**
- Quando não precisa do índice
- Código mais **limpo e legível**

❌ Erro comum

```
const obj = { a: 1, b: 2 };

for (const x of obj) {
  console.log(x);
}
// ❌ ERRO: objeto não é iterável
```

4️⃣ FOR...IN

📌 O que é

Usado para **percorrer chaves (propriedades)** de um objeto.

📐 Sintaxe

```
for (const chave in objeto) {
  // código
}
```

☑️ Exemplo com objeto

```
const pessoa = {
  nome: "Ana",
  idade: 25,
  cidade: "SP"
};

for (const chave in pessoa) {
  console.log(chave, pessoa[chave]);
}
```

🔍 O que ele retorna

- Retorna as **chaves**
- Não retorna valores diretamente

⚠️ Uso com arrays (NÃO RECOMENDADO)

```
const nums = [10, 20, 30];

for (const i in nums) {
  console.log(i); // índice, não valor
}
```

➡️ Pode causar bugs
➡️ Use `for...of` para arrays

🧠 Quando usar

- Quando precisa percorrer **objetos**
- Quando precisa das **chaves**

---

5 COMPARAÇÃO GERAL (RESUMO PARA MEMORIZAR)

Loop	Percorre	Usa índice	Ideal para
`for`	Qualquer coisa	Sim	Controle total
`for...of`	Valores	Não	Arrays, strings
`for...in`	Chaves	Não	Objetos

---

6 QUAL USAR EM CADA SITUAÇÃO

🔹 Array

```
for (let i = 0; i < arr.length; i++) {}
// ou
for (const item of arr) {}
```

🔹 String

```
for (const letra of texto) {}
```

🔹 Objeto

```
for (const chave in objeto) {}
```

---

7 REGRA DE OURO 🧠🔥

Valores → `for...of`
Chaves → `for...in`
Controle total → `for` tradicional

---

8 CONCLUSÃO FINAL

- `for` → mais poderoso, mais verboso
- `for...of` → moderno, limpo, ideal para iteráveis
- `for...in` → específico para objetos
- **Iteráveis definem se `for...of` funciona**
- Escolher o loop certo evita bugs e deixa o código profissional