

JavaScript

ESTRUTURAS E CONTROLE DE FLUXO



Controle de fluxo

if...else

switch

Laços e iteração

for

while

do...while

break/continue

for..of

for..in

Controle de fluxo- **if...else**

Uma declaração condicional é um conjunto de comandos que são executados caso um condição especificada seja verdadeira.

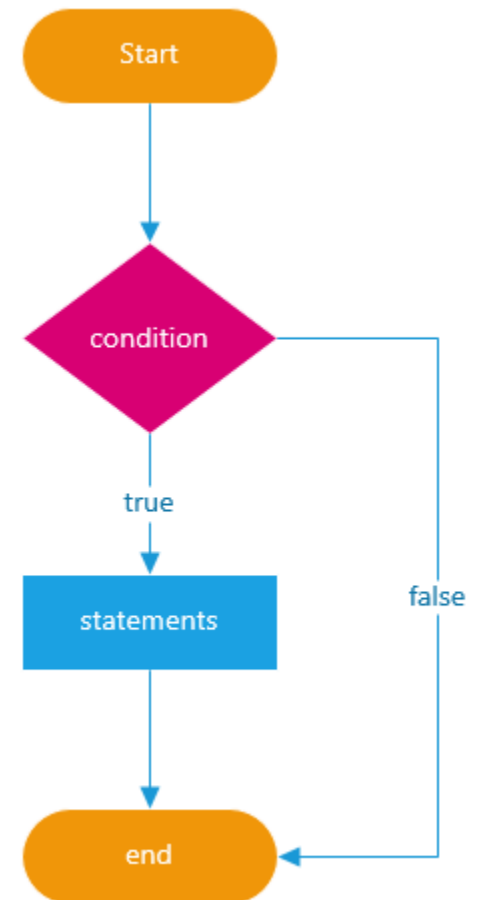
O JavaScript suporta duas declarações condicionais: **if...else** e **switch**.

Controle de fluxo- if...else

A instrução **if** executa **uma instrução** ou **bloco** de código se uma condição for satisfeita

```
if( condition ) {  
  //statements  
}
```

Boa prática: Sempre usar as chaves, mesmo se houver apenas uma instrução a ser executada



Controle de fluxo- if...else


condition pode ser qualquer expressão válida.

Em geral, a condição é avaliada como um valor booleano, **true** ou **false**.

```
if( condition ) {  
  //statements  
}
```

```
let year = prompt('Em que ano a  
especificação ECMAScript-2015 foi  
publicada?', '');
```

```
if (year == 2015) {  
  alert('Temos um Xeroque Rolmes aqui!' );  
}
```



```
//IF
```

```
let x = 25;
```

```
if( x > 10 ) {  
    console.log('x É MAIOR QUE 10');  
}
```

```
// undefined (false) pois não existe tratamento
```

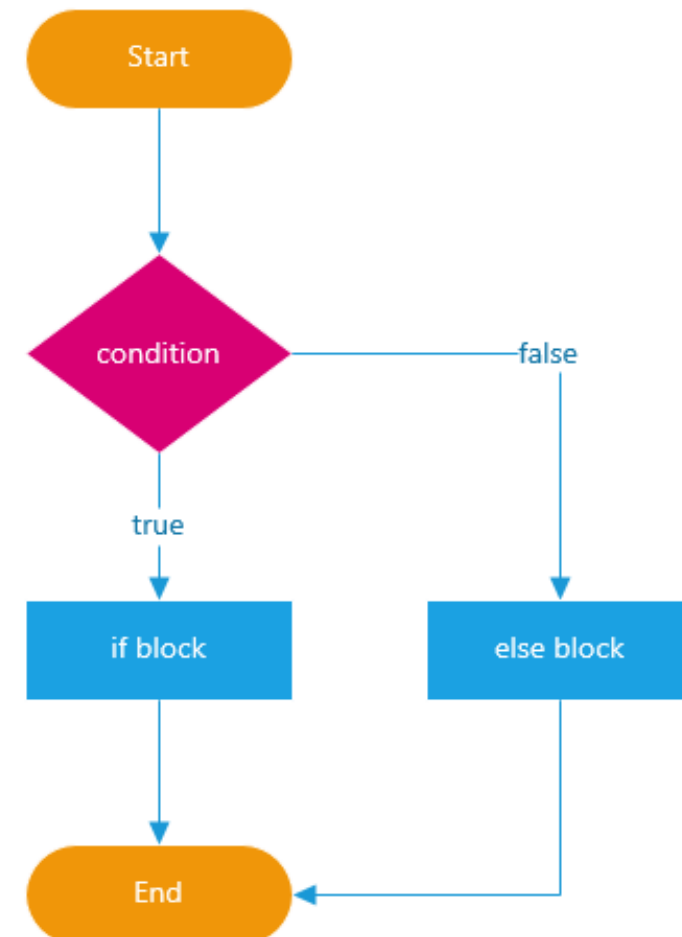
```
let x2 = 9;
```

```
if( x2 > 10 ) {  
    console.log('x2 É MAIOR QUE 10');  
}
```

Controle de fluxo- if...else

Para executar outra instrução quando o **condition** avaliar como **false**, use o **else**

```
If (condition) {  
    //statement  
}  
else {  
    //statement (when condition  
    evaluates to false)  
}
```



Controle de fluxo- if...else

Você também pode **encadear** as declarações com **else if** para obter várias condições testadas em sequência

```
if (condition_1) {  
    // statments  
}  
else if (condition_2) {  
    // statments }  
else {  
    // statments  
}
```




```
//ELSE
```

```
let x = 5;
```

```
if (x > 10) {  
    console.log('x É MAIOR QUE 10');  
}
```

```
else {  
    console.log('x É MENOR OU IGUAL A 10');  
}
```

```
//ELSE IF - Compara Se a for maior, menor ou igual a b
```

```
let a = 10, b = 20;
```

```
if (a > b) {  
    console.log('a é MAIOR que b');  
}
```

```
else if (a < b) {  
    console.log('a é MENOR que b');  
} else
```

```
{  
    console.log('a é IGUAL a b');  
}
```

Atente pelo modo da declaração da
VARIÁVEL B

Controle de fluxo : **operador condicional**

JavaScript fornece um operador condicional ou **operador ternário** que pode ser usado como uma abreviação da instrução **if else**

```
// TERNÁRIO ? :  
let nota = 0.3;
```

```
nota >= 0.5 ? "aprovado" : "reprovado"
```



```
// VEJA AS DIFERENÇAS
```

```
var animal = 'kitty';  
var result = (animal === 'kitty') ? 'fofo' : 'ainda bom';  
console.log(result)
```

```
var animal = 'kitty';  
var result = '';
```

```
if (animal === 'kitty') {  
    result = 'fofo';  
}  
else {  
    result = 'ainda bom';  
}  
console.log(result)
```

=== Valor e Tipo igual (idêntico)

Para **x= 10** temos que :

x === 8 -> retorna false

x === 10 -> retorna true

x === "10" -> retorna false



```
//Múltiplos
```

```
let age = prompt('idade?', 18);
```

```
let message = (age < 3) ? 'Hi, bebê!' :  
  (age < 18) ? 'Fala Xovem!' :  
  (age < 100) ? 'Saudações vovô!' :  
  'Que idade incomum!';
```

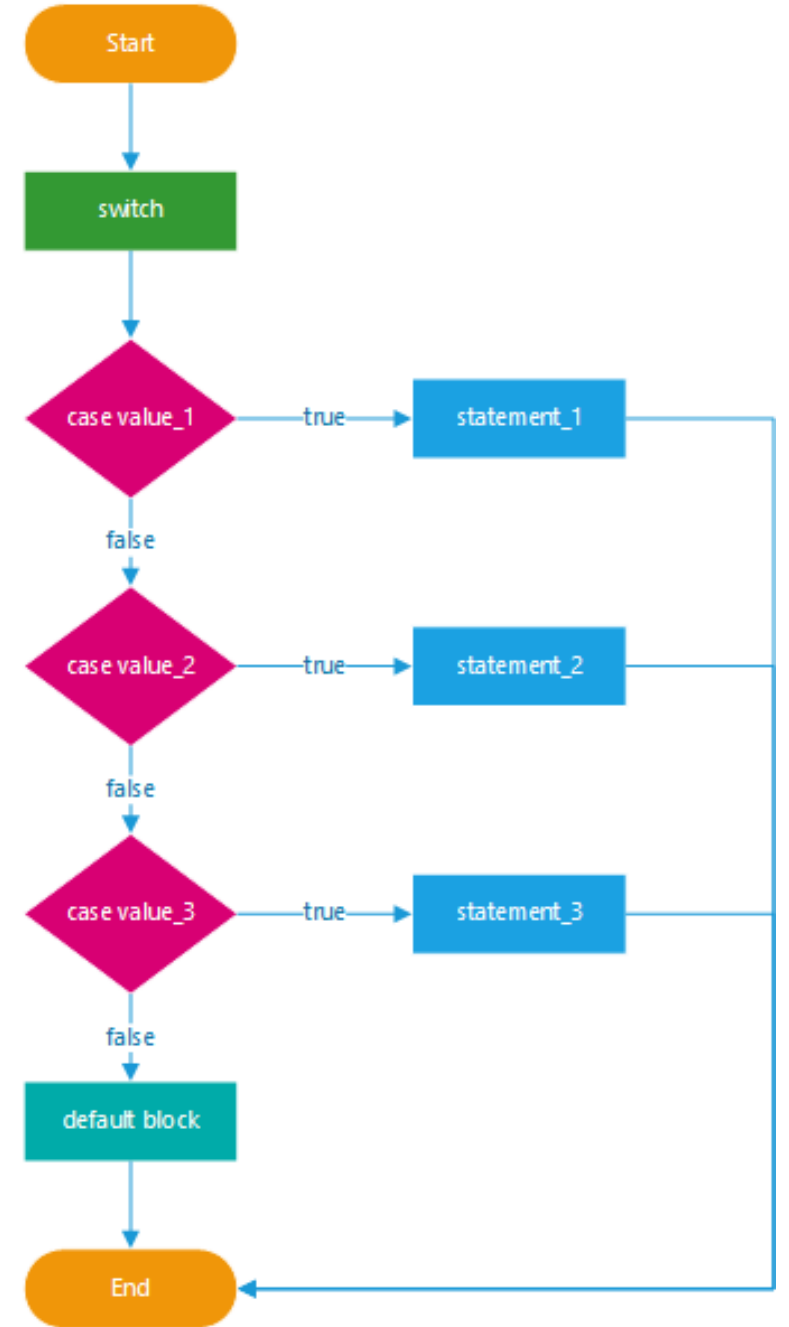
```
alert( message );
```

Controle de fluxo- **switch.. case**

Semelhante à instrução
if else, você usa o
switch..case para
controlar operações
condicionais complexas.

```
switch (expression) {  
  case value_1:  
    statement_1;  
    break;  
  case value_2:  
    statement_2;  
    break;  
  case value_3:  
    statement_3;  
    break;  
  default:  
    default_statement;  
}
```

A instrução **break** associada a cada cláusula case, garante que o programa **sairá do switch** assim que a declaração correspondente for executada



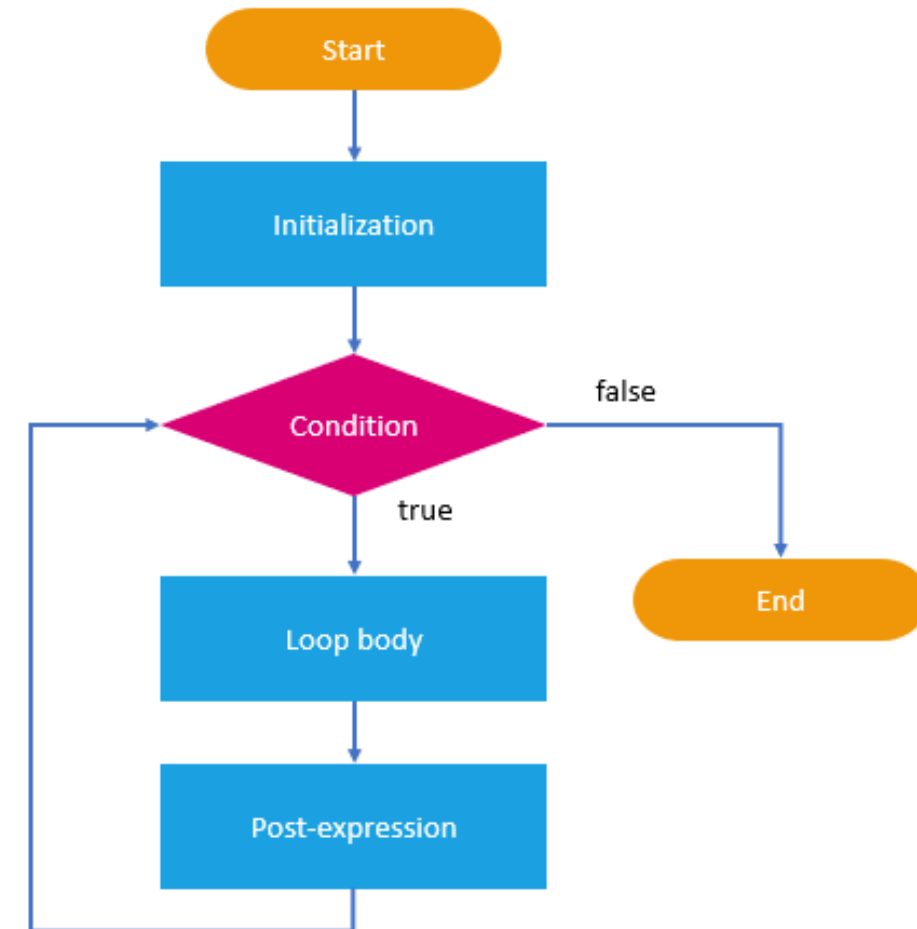
```
var tipofruta="Banana";

switch (tipofruta) {
  case "Laranja":
    console.log("O quilo da laranja está R$0,59.");
    break;
  case "Maçã":
    console.log("O quilo da maçã está R$0,32.");
    break;
  case "Banana":
    console.log("O quilo da banana está R$0,48.");
    break;
  case "Cereja":
    console.log("O quilo da cereja está R$3,00.");
    break;
  case "Manga":
    console.log("O quilo da manga está R$0,56.");
    break;
  case "Mamão":
    console.log("O quilo do mamão está R$2,23.");
    break;
  default:
    console.log("Desculpe, não temos " + tipofruta + ".");
}
console.log("Gostaria de mais alguma coisa?");
```

Laços e iteração **for**

A instrução de repetição **for** permite que você crie um loop com três expressões opcionais

```
for (begin; condition; step) {  
  // ... loop body ...  
}
```




```
for (begin; condition; step) {  
  // ... loop body ...  
}
```

FOR		
begin	<code>let i = 0</code>	Executa uma vez ao entrar no loop
condition	<code>i < 3</code>	Verificado antes de cada iteração do loop. Se for falso, o loop para.
body	<code>alert(i)</code>	Executa repetidamente enquanto a condição é verdadeira.
step	<code>i++</code>	Executa após o corpo em cada iteração.

```
for (var counter = 1; counter < 5; counter++) {  
  console.log('Dentro do loop:' + counter);  
}  
console.log('Fora do loop:' + counter);
```

```
//Acessar a variável counter após o loop causou a ReferenceError  
for (let counter = 1; counter < 5; counter++) {  
  console.log('Dentro do loop:' + counter);  
}  
console.log('Fora do loop:' + counter);
```

//loop sem inicialização-opcional

```
var j = 1;
for (; j < 10; j += 2) {
  console.log(j);
}
```

//loop sem condição - opcional
//Se omitido, precisará um break
//para encerrar o loop

```
for (let j = 1;; j += 2) {
  console.log(j);
  if (j > 10) {
    break; }
}
```

//loop sem nenhuma expressão - é opcional.
//Se omitido, precisará um break para
encerrar o loop //também modificar a variável
do contador

```
let j = 1;
for (;;) {
  if (j > 10)
    break;
  console.log(j);
  j += 2;
}
```

//loop sem o corpo do loop, você coloca um
ponto-e-vírgula após a instrução for

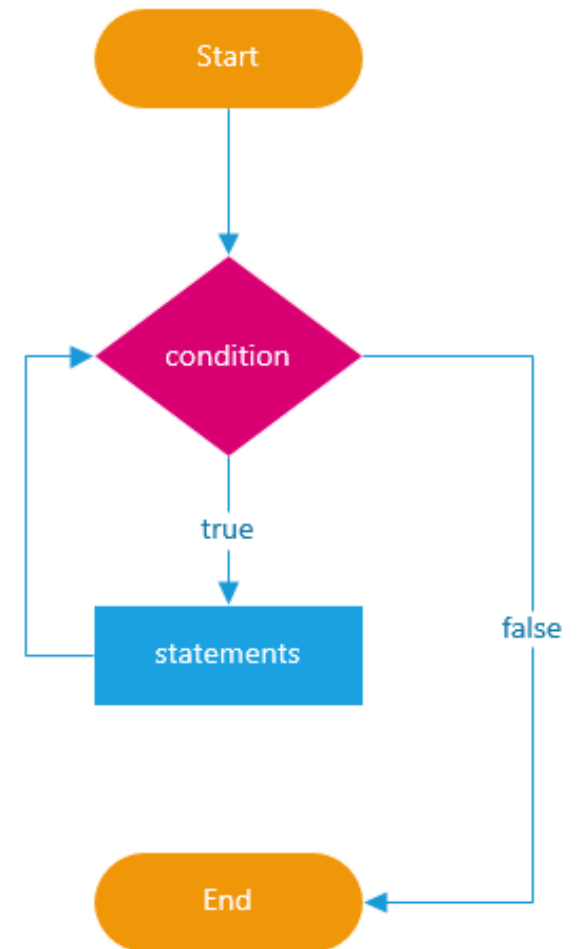
```
let sum = 0;
for (let i = 0; i <= 9; i++, sum += i);
console.log(sum);
```

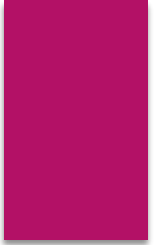
Laços e iteração **While**

A instrução **while** cria um loop que executa um bloco de código, até que a condição de teste seja avaliada como **true**.

```
while (expression) {  
    // statement  
}
```

A instrução **while** avalia a expressão **antes** de cada iteração do loop.





```
// while é conhecido como loop de pré-teste
// É possível que seja executado
```

```
let i = 0;
while (i < 3) {
    // mostra 0, 1 e 2
    alert( i ); i++;
}
```

```
// Cuidado! veja o alert e o console.log
// No console é mostrado o valor da
// variável ao final
```

```
let count = 1;
while (count < 10) {
    alert('contador é: ' + count);
    count +=2;
}
```

```
// DECREMENTANDO NO WHILE // Lembrado
que o console.log mostra // o valor
final da variavel
```

```
var i = 100;
while (i > 0) {
    console.log(i);
    i--; // equivalente a i=i-1
}
```

```
// EVITE LAÇOS INFINITOS // EXECUTE
ESSE CÓDIGO POR RISCO // E CONTA
PRÓPRIA!!!!
```

```
while (true) { console.log("Olá,
mundo");
}
```

label e break

- A declaração **label** identifica um laço e permite que este seja referenciado em outro lugar no seu programa
- Quando você utiliza **break** com um **label**, ele encerrará o **label** específico.

Pode-se utilizar o **break** nas instruções **while**, **do-while**, **for**, ou **switch** sem o **label**, o **break** encerrará imediatamente o laço



```
//LABEL
```

```
var x = 0;
```

```
var z = 0
```

```
labelCancelaLaco:
```

```
while (true) {
```

```
    console.log("Laço exterior: ----- " + x);
```

```
    x += 1;
```

```
    z = 1;
```

```
while (true) {
```

```
    console.log("Laço interior: " + z);
```

```
    z += 1;
```

```
    if (z === 10 && x === 10) {
```

```
        break labelCancelaLaco;
```

```
    } else if (z === 10) {
```

```
        break;
```

```
    }
```

```
    }
```

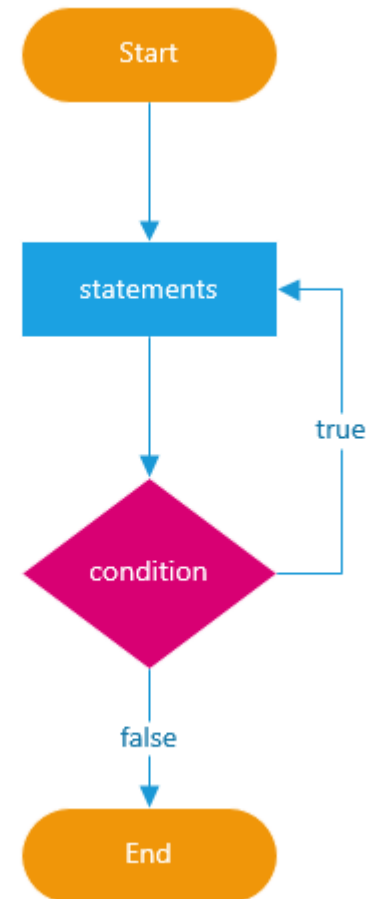
```
}
```

Laços e iteração **do...while**

A instrução **do..while** cria um loop que executa um bloco de código, até que a condição de teste seja avaliada como **false**.

```
do {  
    statement(s);  
} while(expression);
```

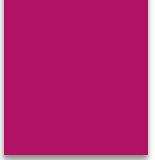
A instrução **do..while** avalia a expressão **depois** de cada iteração do loop.



Laços e iteração **do...while**

Importante: **do..while** costuma ser utilizado quando o loop precisa executar pelo menos uma vez

O exemplo mais típico de uso do **do..while** é obter uma entrada do usuário até que o valor fornecido seja esperado.



```
const MIN = 1;
const MAX = 12;

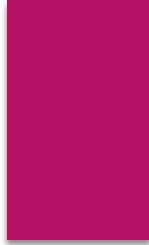
let secretNumber = Math.floor(Math.random() * (MAX - MIN + 1)) + MIN;
let guesses = 0; // armazena o número de palpites
let hint = ''; // para armazenar dica
let number = 0;

do { // obtém a entrada do usuário
  let input = prompt(`Por favor insira um número entre ${MIN} e ${MAX}` + hint);
  // pegue o inteiro
  number = parseInt(input); // aumentar o número de palpites
  guesses++; // verifique o número de entrada com o número secreto
  // dê uma dica se necessário
  if (number > secretNumber) {
    hint = ', e menor que ' + number;
  } else if (number < secretNumber) {
    hint = ', e maior que ' + number;
  } else if (number == secretNumber) {
    alert(`Bravo! você está correto depois ${guesses} palpite(s).`);
  }
} while (number != secretNumber);
```

label e continue

A declaração **continue** pula a iteração atual de um loop, imediatamente para a próxima instrução.

```
do{ if (condition) {  
    continue; // pula para expression  
}  
// mais instruções aqui  
// ...  
}  
while(expression); // continue pula para cá
```



```
// CONTINUE
let s = 'Esta é uma demonstração de declaração de continue em
JavaScript';
let counter = 0;
i=0;

do{
    i++;
    if (s.charAt(i) !== 's') {
        continue; // pula para expression
    }
    counter++;
} while (i < s.length); // continue pula para cá

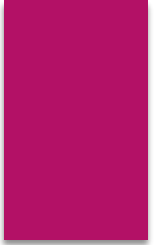
console.log('O número de S encontrados na string é ' + counter);
```

Laços e iteração **for..of**

A declaração **for...of** executa iterações a partir de uma variável específica, percorrendo **todas as propriedades** de um **objeto**

for...of “interage sobre valores das propriedades de um objeto, na ordem original de inserção”.

Na prática, isso significa que o laço enxerga os seus valores e não as propriedades



```
for (variavel of objeto) {  
  declaracoes  
}
```

```
const numeros = [1,2,3,4,5];
```

```
for(let numero of numeros) {  
  console.log(numero);  
}  
// resultado: 1, 2, 3, 4, 5
```

- laço for...of itera os valores das propriedades
- laço for...in itera as propriedades

Laços e iteração **for..in**

A declaração **for...in** executa iterações a partir de uma variável específica, percorrendo **todas as propriedades** de um **objeto**

for...in “interage sobre propriedades enumeradas de um objeto, na ordem original de inserção”.

Na prática, isso significa que o laço enxerga as propriedades e não os seus valores

```
for (variavel in objeto) {  
  declaracoes  
}
```

```
const numeros = [1,2,3,4,5];
```

```
for(let numero in numeros) {  
  console.log(numero);  
}  
// resultado: 0, 1, 2, 3, 4
```

- laço for...of itera os valores das propriedades
- laço for...in itera as propriedades

```
// VEJA A DIFERENÇA
// vamos iterar o objeto Casa
// que possui 3 propriedades:
// área, altura, andares
```

```
const Casa = {
  area: 1000,
  altura: 7,
  andares: 2
}

for(let prop in Casa) {
  console.log(prop);
}

// area
// altura
// andares
```

```
// MUDANDO DE IN PARA OF
// PARA MOSTRAR OS VALORES
```

```
const Casa = {
  area: 1000, altura: 7, andares: 2,
  [Symbol.iterator]: function* () { //for...of
    procura pela propriedade [Symbol.iterator] do
    objeto
    yield this.area;
    yield this.altura;
    yield this.andares;
  }
}

for(let prop of Casa) {
  console.log(prop);
}

// 1000, 7, 2
```