Condicionais e Repetições em JavaScript

Aprendendo if, else, else if, for e while

by:Gregory Klaus

Quando usar if?

Quando precisamos tomar decisões no código, ou seja, executar algo somente se uma condição for verdadeira. Exemplo da vida real: Se você tiver nota maior que 7, você passou.

Como fazer?

Usamos o if para a verificação. Podemos adicionar else para o "senão", e else if para outras possibilidades.

Exemplo:

```
let nota = 8;

if (nota >= 7) {
  console.log("Aprovado");
}
```

Explicação:

O if representa o "se". Entre parênteses, temos a condição que queremos testar (nota >= 7).
Se for verdadeira, o bloco dentro das chaves será executado.
Se não for, o código ignora esse trecho.

Quando usar else?

Quando temos duas possibilidades: uma ação se a condição for verdadeira, outra se for falsa.

Como fazer?

Depois do if, adicionamos else com outro bloco de código para o caso contrário.

Exemplo:

```
let idade = 16;

if (idade >= 18) {
  console.log("Você pode dirigir");
} else {
  console.log("Você ainda não pode dirigir");
}
```

Explicação:

Se a idade for 18 ou mais, mostramos que a pessoa pode dirigir.
Se não for, a execução cai no else, mostrando a outra mensagem.
Usamos if (condição) para o "se" e else para o "senão".

Usando else if para múltiplas condições

Quando usar?

Quando queremos verificar mais de uma condição, de forma sequencial.

Quando usar?

Podemos colocar quantos else if quisermos entre if e else.

Exemplo:

```
let nota = 6;

if (nota >= 9) {
  console.log("Excelente");
} else if (nota >= 7) {
  console.log("Bom");
} else {
  console.log("Precisa melhorar");
}
```

Explicação:

O programa testa as condições na ordem. Se nota >= 9 for falsa, ele tenta nota >= 7.

Se nenhuma for verdadeira, vai para o else. Assim, conseguimos tratar várias faixas.

Operadores de comparação

Quando usar?

Quando precisamos comparar valores dentro do if para tomar decisões.

Quando usar?

Use operadores como ==, >, <, >=, ===, etc.

Exemplo:

Operador Significado Exemplo

== Igual (valor) a == b
=== Igual (valor e tipo) a === b
!= Diferente a != b
> Maiorque a > b
< Menor que a < b
>= Maior ou igual a >= b
<= Menor ou igual a <= b

Explicação:

Esses símbolos são usados nas condições dos if. Exemplo: idade >= 18 verifica se a idade é maior ou igual a 18.

Cuidado: = não é ==

Quando usar?

Sempre que for comparar valores, evite usar apenas =.

Quando usar?

Use == para comparar valores e === para comparar valor e tipo.

Exemplo:

```
let a = 10;

if (a == 10) {
  console.log("a é igual a 10");
}
```

Explicação:

= serve para atribuir valores, não comparar.

A linha let a = 10; está guardando o valor 10 na variável.

Já a == 10 está testando se o valor de a é igual a 10.

Estrutura de repetição: while

Quando usar?

Quando queremos repetir uma ação sem saber exatamente quantas vezes, apenas enquanto uma condição for verdadeira.

Como fazer?

```
while (condição) {
  // código a repetir
}
```

Exemplo:

```
let contador = 0;

while (contador < 3) {
  console.log("Contando: " + contador);
  contador++;
}
```

Explicação:

O while executa o bloco enquanto a condição for verdadeira.
Aqui, ele imprime até o contador chegar a 3.
Se esquecer o contador++, vira um loop infinito!

Estrutura de repetição: for

Quando usar?

Quando sabemos quantas vezes queremos repetir algo.

Como fazer?

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

Exemplo:

```
for (let i = 0; i < 3; i++) {
  console.log("Repetição: " + i);
}
```

Explicação:

O for tem 3 partes: começa em 0, vai até 2 (i < 3), e soma 1 a cada volta.
Ele imprime três vezes: com i = 0, 1 e 2.

Diferença entre for e while

Quando usar cada um?

	USE FOR	USE WHIL
Número fixo de repetições	✅	❌
Repetição até uma condição	⚠️	✅
Contar ou percorrer listas	✅	⚠️ Possíve

Como fazer?

Se souber o número de repetições, use for.
Se for algo incerto (ex: esperar uma senha correta), use while.

Como decidir?

Se souber o número de repetições, use for.
Se for algo incerto (ex: esperar uma senha correta), use while.

Exemplo:

```
// for
for (let i = 1; i <= 5; i++) {
  console.log(i);
}

// while
let senha = "";
while (senha !== "1234") {
  senha = prompt("Digite a senha:");
}
```

Estrutura de repetição: for

Quando usar?

Quando você quer verificar algo dentro de um laço, por exemplo, se um número é par ou ímpar.

Como fazer?

Usar if dentro do for normalmente.

Exemplo:

```
for (let i = 1; i <= 5; i++) {
  if (i % 2 === 0) {
    console.log(i + " é par");
  } else {
    console.log(i + " é ímpar");
  }
}
```

Explicação:

O laço percorre de 1 a 5. A cada número, o if testa se é par (% 2 === 0) e imprime a mensagem certa.

Erros comuns

Quando acontece?

Quando esquecemos detalhes da sintaxe ou lógica dos laços e condições.

Como evitar?

- Revise se está comparando com ==, não =.
- Teste seu código aos poucos.
- Cuidado com loops infinitos.

Exemplo de erro comum:

```
let i = 0;
while (i < 3) {
  console.log(i);
  // esqueci o i++
}
```

Explicação:

Sem o i++, o valor de i nunca muda. Isso trava o navegador!

Aplicação prática: Classificação por idade

Quando usar?

Para classificar pessoas conforme faixas etárias com if, else if e else.

Como fazer?

Compare a idade com os limites.

Exemplo:

```
let idade = 15;

if (idade < 12) {
  console.log("Criança");
} else if (idade < 18) {
  console.log("Adolescente");
} else {
  console.log("Adulto");
}
```

Explicação:

A idade é testada em cada condição. A primeira que for verdadeira será executada.

Desafio: notas e idade

Quando usar?

Quando quiser juntar condições e laços em um exercício mais completo.

Como fazer?

Use for para as notas, if para verificar idade e resultado.

Exemplo:

1. Peça nome e idade
2. Peça 3 notas (com for)
3. Calcule a média
4. Diga se é maior de idade
5. Diga se foi aprovado (média ≥ 7)

Por que isso é importante?

Quando usar?

Sempre que quiser que seu programa reaja a dados ou repita tarefas.

Como ajuda?

- Torna seu código inteligente
- Deixa o programa dinâmico e funcional
- Prepara para resolver problemas reais

Exemplo:

Aplicativos que mostram mensagens, calculam resultados, validam formulários — todos usam essas estruturas.