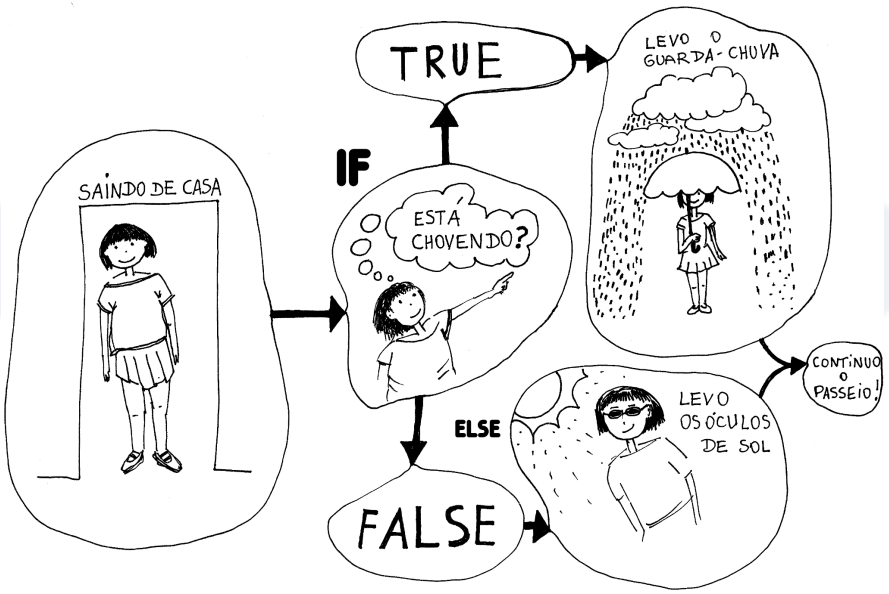
MATERIAL EXTRA

Estruturas condicionais

growdev

ESTRUTURAS CONDICIONAIS

Quando falamos em lógica de programação, as estruturas condicionais são recursos oferecidos pelas linguagens para que seja possível verificar uma condição e alterar o fluxo de execução do algoritmo. Assim, é possível definir uma ação específica para diferentes cenários e obter exatamente o resultado esperado durante o desenvolvimento de um site ou de uma aplicação.



(Disponível em Condicionais - artprog)

Para que servem as estruturas condicionais?

As estruturas condicionais servem para permitir que um programa tome decisões automáticas. Elas são usadas para:

- **Verificar condições**: Decidir se uma determinada ação deve ser executada, dependendo se uma condição é verdadeira ou falsa.
- **Controle do fluxo de execução**: Em vez de o código seguir uma sequência linear de instruções, ele pode "pular" ou escolher um bloco de código baseado na condição dada.
- **Lidar com diferentes cenários**: Muitos programas precisam lidar com diferentes entradas, como por exemplo, quando um usuário digita uma senha correta ou incorreta, ou quando uma variável atinge ou não um valor limite.

Por exemplo, imagine um formulário online que verifica se o campo de e-mail foi preenchido corretamente. O código poderia usar uma estrutura condicional para determinar se o formato do e-mail está correto antes de prosseguir com o envio dos dados.

Por que as Estruturas Condicionais são tão importantes em um algoritmo?

As estruturas condicionais são uma das partes mais fundamentais de qualquer algoritmo, por várias razões:

- **Tomada de Decisão**: Em qualquer algoritmo, a capacidade de tomar decisões com base em informações é essencial. Sem as estruturas condicionais, os programas seriam sequências rígidas e inflexíveis de comandos. As condições permitem que o algoritmo "pense" e decida qual caminho seguir.

- **Flexibilidade e Adaptação**: As estruturas condicionais permitem que o algoritmo se adapte às circunstâncias. Isso é crucial em cenários dinâmicos, como em sistemas de comércio eletrônico, onde o algoritmo precisa se ajustar com base na entrada do usuário, ou em jogos, onde o comportamento dos personagens muda conforme as ações do jogador.
- **Otimização e Eficiência**: Elas ajudam a otimizar o código, garantindo que certas operações sejam executadas apenas quando necessário. Isso torna o algoritmo mais eficiente e rápido, pois evita a execução de instruções desnecessárias.
- **Resolução de Problemas**: Quase todos os problemas resolvidos por meio de algoritmos envolvem algum tipo de decisão. Desde tarefas simples, como calcular o maior entre dois números, até a implementação de IA em jogos, as estruturas condicionais desempenham um papel central na resolução desses problemas.
- **Interatividade e Controle**: Em interfaces de usuário, as condicionais permitem responder às ações do usuário, como cliques, preenchimento de campos de formulários, ou navegação. A lógica condicional facilita a criação de sistemas interativos e responsivos.

No JavaScript, as estruturas condicionais mais comuns são o `if`, `else`, `else if`, `switch`, e o `operador ternário`. Cada uma dessas estruturas oferece maneiras diferentes de lidar com condições e fazer escolhas no código.

if

A estrutura `if` (se, em português) é a mais básica. Ela executa um bloco de código se a condição especificada for verdadeira.

Sintaxe:

```
if (condição) {
  // bloco de código a ser executado se a condição for verdadeira
}
```

Exemplo:

```
let idade = 18

if (idade >= 18) {
   console.log("Você é maior de idade.")
}
```

No exemplo acima, o programa verifica se a variável `idade` é maior ou igual a 18. Se for verdade, ele imprime a mensagem "Você é maior de idade".

else

O `else` é usado para fornecer um caminho alternativo caso a condição do `if` seja falsa. Ele define o que fazer quando a condição não for atendida.

Sintaxe:

```
if (condição) {
  // bloco de código a ser executado se a condição for verdadeira
} else {
  // bloco de código a ser executado se a condição for falsa
}
```

Exemplo:

```
let idade = 16

if (idade >= 18) {
  console.log("Você é maior de idade.")
} else {
  console.log("Você é menor de idade.")
}
```

Aqui, como a variável `idade` é 16, o programa vai para o bloco `else` e imprime "Você é menor de idade."

else if

Quando você tem mais de uma condição a ser verificada, pode usar o `else if`. Ele permite que você verifique múltiplas condições em sequência. O JavaScript avalia cada condição de cima para baixo até encontrar uma que seja verdadeira.

Sintaxe:

```
if (condição1) {
  // bloco de código a ser executado se a condição1 for verdadeira
} else if (condição2) {
  // bloco de código a ser executado se a condição2 for verdadeira
} else {
  // bloco de código a ser executado se nenhuma das
  // condições anteriores for verdadeira
}
```

Exemplo:

```
let nota = 75

if (nota >= 90) {
  console.log("Aprovado com excelência.")
} else if (nota >= 70) {
  console.log("Aprovado.")
} else {
  console.log("Reprovado.")
}
```

Nesse caso, como a `nota` é 75, a segunda condição (`nota >= 70`) é verdadeira, então o programa imprime "Aprovado."

Operador Ternário

O operador ternário é uma forma mais compacta de escrever um `if` simples, quando há apenas duas condições (verdadeira ou falsa). Ele é útil para expressar condições curtas de forma concisa.

Sintaxe:

```
condição ? valorSeVerdadeiro : valorSeFalso
```

Exemplo:

```
let idade = 18
let resultado = idade >= 18 ? "Maior de idade" : "Menor de idade"
console.log(resultado)
```

Esse código faz exatamente o mesmo que os exemplos anteriores, mas em uma única linha. Se a variável `idade` for maior ou igual a 18, ele retorna "Maior de idade"; caso contrário, "Menor de idade".

Switch

A estrutura `switch` é uma alternativa ao uso de múltiplos `else if` quando você precisa comparar o valor de uma variável com diferentes casos. Cada caso é testado, e quando um deles é verdadeiro, o bloco correspondente é executado.

Sintaxe:

```
switch (expressão) {
  case valor1:
    // bloco de código para valor1
    break
  case valor2:
    // bloco de código para valor2
    break
  default:
    // bloco de código se nenhum valor corresponder
}
```

Exemplo:

```
let cor = "vermelho"
switch (cor) {
  case "azul":
    console.log("A cor é azul.")
    break;
  case "vermelho":
    console.log("A cor é vermelho.")
    break;
  default:
    console.log("Cor desconhecida.")
}
```

Nesse exemplo, como a variável `cor` é igual a `"vermelho"`, o programa vai para o caso correspondente e imprime "A cor é vermelho."

Considerações Finais

- **Importância de `break` no `switch`**: O `break` impede que o código continue verificando outros casos após encontrar uma correspondência. Sem ele, o JavaScript executa todos os casos subsequentes, mesmo se eles não corresponderem.
- **Evitar muitas condições**: É uma boa prática evitar blocos muito complexos com múltiplos `if` e `else if`. Em vez disso, você pode reorganizar sua lógica ou usar outras técnicas, como mapeamento de valores, para tornar o código mais limpo.

As estruturas condicionais em JavaScript ajudam a moldar o fluxo do programa, permitindo que ele responda de maneiras diferentes dependendo dos dados e das situações encontradas. Saber usá-las bem é essencial para escrever código funcional e eficiente.

BONS ESTUDOS

growdev