Instruções e repetição

As instruções e repetições são componentes fundamentais da lógica de programação e são amplamente utilizadas em diversas linguagens, incluindo JavaScript. As instruções são ações que o programa executa, como atribuições, cálculos ou chamadas de função, enquanto as repetições permitem que um bloco de código seja executado várias vezes, de acordo com uma condição específica. Nesta aula, exploraremos como utilizar estruturas como `if-else`, `switch`, e os principais laços de repetição, como `for`, `while` e `do...while`, em JavaScript. O objetivo é compreender como aplicar essas ferramentas para tornar o código mais eficiente e dinâmico.

Definição de Instruções e Repetição

Instruções e estruturas de repetição formam a base lógica de qualquer programa de computador, e em JavaScript isso é especialmente verdadeiro. Uma instrução em programação é uma unidade básica de ação: ela define algo que o computador deve fazer. Pode ser uma declaração de variável, uma operação matemática, ou até a chamada de uma função. Tudo que o programa faz é composto de instruções.

Por exemplo, a instrução abaixo declara uma variável x e atribui o valor 10 a ela:

javascript

```
let x = 10;
```

Cada vez que escrevemos algo no código que realiza uma ação, estamos criando uma instrução. Um programa é, portanto, uma sequência de instruções.

Além de simplesmente declarar variáveis ou fazer operações únicas, muitas vezes precisamos executar uma ação várias vezes. É aí que entram as estruturas de repetição, ou laços. Laços permitem que um bloco de código seja repetido múltiplas vezes, até que uma condição específica seja atingida. Em JavaScript, temos três tipos principais de laços: `for`, `while`, e `do...while`.

Imagine que você deseja imprimir os números de 1 a 10 na tela. Em vez de escrever `console.log(1)`, `console.log(2)` até o número 10, podemos usar um laço `for` para simplificar:

javascript

```
for (let i = 1; i <= 10; i++) {

 console.log(i);}
```

Aqui, o laço `for` possui três partes essenciais: a inicialização (`let i = 1`), a condição (`i <= 10`), e o incremento (`i++`). Esse laço começa com `i` igual a 1, verifica se `i` é menor ou igual a 10, e incrementa `i` em 1 a cada iteração, até que a condição `i <= 10` não seja mais verdadeira.

Estruturas de repetição são essenciais para realizar operações repetitivas, economizando tempo e reduzindo erros. Em um fluxograma, podemos visualizar a repetição como um ciclo: começa em um ponto inicial, verifica uma condição e, enquanto ela for verdadeira, o código será executado. Caso contrário, o laço será interrompido.

Por exemplo, pense em um jogo que continua enquanto o jogador tiver vidas:

```
javascript

let vidas = 3;

while (vidas > 0) {

 console.log("Jogo em andamento. Vidas restantes: " + vidas);

 vidas–;  // Decrementa o número de vidas}
```

Neste código, o laço `while` continua rodando enquanto o número de vidas for maior que 0. Quando as vidas chegam a 0, o jogo termina.

Instruções Condicionais If-Else

Instruções condicionais são blocos de código que permitem que o programa "tome decisões". A estrutura mais comum é o if-else, que em português pode ser lido como "se... senão". Ele funciona avaliando uma condição: se a condição for verdadeira, um bloco de código é executado; se for falsa, outro bloco de código pode ser executado.

Por exemplo, imagine que você está escrevendo um código que decide se uma pessoa pode dirigir com base em sua idade:

```
javascript

let idade = 20;

if (idade >= 18) {
```

console.log(“Você pode dirigir.”);} else {console.log(“Você ainda não pode dirigir.”);}

Aqui, o programa verifica se a idade é maior ou igual a 18. Se for, a mensagem “Você pode dirigir” será exibida. Caso contrário, a mensagem “Você ainda não pode dirigir” será exibida.

Instruções condicionais são muito poderosas porque permitem que o programa siga caminhos diferentes dependendo dos dados ou das condições. Quando há mais de duas opções, podemos expandir a estrutura com `else if`, permitindo múltiplas verificações:

javascript

let clima = “nublado”;

if (clima === “ensolarado”) {console.log(“Dia perfeito para um passeio.”);} else if (clima === “chuvoso”) {console.log(“Não esqueça o guarda-chuva!”);} else {console.log(“O clima está imprevisível.”);}

Aqui, o programa avalia o valor da variável `clima`. Dependendo se está “ensolarado”, “chuvoso” ou outro valor, uma ação diferente será tomada. Essa flexibilidade torna as instruções condicionais fundamentais para criar programas dinâmicos e interativos.

Instruções Condicionais Switch

A estrutura switch é usada quando temos várias condições para testar e queremos uma forma mais organizada de fazer isso. Enquanto podemos usar múltiplos `if-else` para tratar essas condições, o `switch` torna o código

mais legível e fácil de manter. Ele verifica o valor de uma variável e executa diferentes blocos de código com base nesse valor.

Por exemplo, imagine um sistema que exibe o dia da semana baseado em um número:

javascript

```
let dia = 3;

switch (dia) {
  case 1:
    console.log("Domingo");
    break;
  case 2:
    console.log("Segunda-feira");
    break;
  case 3:
    console.log("Terça-feira");
    break;
  default:
    console.log("Dia inválido");}
```

Neste exemplo, o `switch` compara o valor da variável `dia` com os valores fornecidos em `case`. Se `dia` for igual a 3, o programa exibe "Terça-feira". Caso não corresponda a nenhum `case`, a opção `default` será executada, exibindo "Dia inválido".

O `switch` é especialmente útil quando temos muitas opções diferentes para testar e queremos evitar uma longa cadeia de `else if`.

Laço e Repetição

Laços de repetição são blocos de código que se repetem enquanto uma condição for verdadeira. Em JavaScript, os principais laços são o `for`, o `while` e o `do...while`.

O laço for é usado quando sabemos o número exato de vezes que queremos executar um bloco de código. Por exemplo, para imprimir os números de 1 a 5:

```
javascript

for (let i = 1; i <= 5; i++) {

 console.log(i);

}
```

O laço while é usado quando não sabemos ao certo quantas vezes algo precisa ser repetido. Ele continua executando enquanto a condição for verdadeira. Por exemplo:

```
javascript

let contador = 0;
```

```
while (contador < 5) {

  console.log(contador);

  contador++;}
```

Por fim, o laço do...while é semelhante ao `while`, mas ele garante que o código será executado ao menos uma vez, mesmo que a condição inicial seja falsa:

```
  javascript

let x = 6;

do {console.log(x);x++;} while (x < 5);
```

Neste caso, o valor de x é 6, e mesmo que a condição `x < 5` seja falsa desde o início, o código dentro do do será executado uma vez.

Blocos de Código e Rótulos

Blocos de código são usados para agrupar várias instruções dentro de chaves `{ }`. Cada vez que usamos um `if`, `for` ou função, estamos criando blocos de código. Eles ajudam a organizar o código e garantir que as instruções sejam executadas juntas. Um exemplo simples de um bloco de código é:

```
  javascript

if (true) {console.log("Isso será executado.");}
```

Já os rótulos são usados para identificar blocos de código de forma única. Eles permitem que você controle mais de perto o fluxo de execução, especialmente em laços de repetição complexos. Um exemplo é:

```
javascript

outerLoop: for (let i = 0; i < 3; i++) {

 for (let j = 0; j < 3; j++) {

  if (i === j) {

   break outerLoop;}

  console.log(`i: ${i}, j: ${j}`);}

}
```

Aqui, o rótulo `outerLoop` identifica o laço externo, permitindo que o `break` saia do laço correto, mesmo estando dentro de um laço aninhado.

Conteúdo Bônus

"Eloquent JavaScript" é um livro altamente respeitado disponível gratuitamente online. O livro cobre fundamentos do JavaScript, incluindo capítulos específicos sobre instruções e laços de repetição, com exercícios práticos e uma abordagem teórica sólida.

Referência Bibliográfica

DEITEL, P. J.; DEITEL, H. M. Ajax, rich internet applications e desenvolvimento web para programadores. Pearson: 2008

FELIX, R. (Org.). Programação orientada a objetos. Pearson: 2016

FERREIRA, R. D. Linguagem de programação. Contentus: 2020.

FLATSCHART, F.; BACHINI, C.; CUSIN, C. Open Web Platform. Brasport: 2013.

NEVES, M. C. B. de A. Sites de Alta Performance. Contentus: 2020

PAGE-JONES, M. Fundamentos do desenho orientado a objeto com UML. Pearson: 2001.

PUGA, S.; RISSETTI, G. Lógica de programação e estruturas de dados, com aplicações em Java. Pearson: 2016.

SEGURADO, V. S. (Org.). Projeto de interface com o usuário. Pearson: 2016.

Atividade Prática 4 – Instruções e repetição

Título da Prática: Entendendo conceitos básicos de instruções e repetição

Objetivos: Compreender e fixar a definição de instruções e repetição

Materiais, Métodos e Ferramentas: Computador com uma IDE instalada (recomendado: VS Code) e Javascript instalado via node/npm e o entendimento do contexto abaixo

Atividade Prática

As instruções ou comandos em JavaScript são a base de qualquer programa. Elas indicam ao interpretador o que o programa deve fazer, e

podem incluir uma série de ações, como a declaração de variáveis, chamadas de funções, e principalmente as estruturas de controle que guiam o fluxo do programa.

Entre as principais estruturas de controle em JavaScript, temos as instruções condicionais, como if e else, que permitem executar diferentes blocos de código dependendo das condições estabelecidas. A instrução if verifica uma condição e, se ela for verdadeira, executa um bloco de código. Caso contrário, o código dentro de um else (se presente) será executado.

Já o switch é uma alternativa ao uso de múltiplos if...else, permitindo verificar diversas possibilidades de uma forma mais limpa e organizada.

Além disso, temos as instruções de repetição, como for, while e do...while, que permitem repetir blocos de código várias vezes. O loop for é útil quando o número de repetições é conhecido previamente, enquanto o while é mais flexível e executa o código enquanto uma condição específica for verdadeira. O do...while é semelhante ao while, mas garante que o código seja executado ao menos uma vez antes de verificar a condição.

A seguir, serão apresentados exemplos de código que ilustram como cada uma dessas estruturas pode ser usada no contexto de um programa em JavaScript.

Exemplos de código

- Estrutura Condicional if e else:

let idade = 20;

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

} else {

console.log("Você é menor de idade.");

```
}
```

- Estrutura Condicional switch:

```
let cor = “azul”;

switch (cor) {

case “vermelho”:

                    console.log(“A cor é vermelha.”);

                    break;

case “azul”:

                    console.log(“A cor é azul.”);

                    break;

default:

                    console.log(“Cor desconhecida.”);

}
```

- Laço de Repetição for:

```
for (let i = 0; i < 5; i++) {

console.log("Número " + i);
```

```
}
```

- Laço de Repetição while:

```
let i = 0;

while (i < 5) {

console.log("Número " + i);

i++;

}
```

- Laço de Repetição do...while:

```
let i = 0;

do {

console.log("Número " + i);

i++;

} while (i < 5);
```

De acordo com o contexto e exemplos acima, responda as seguintes perguntas:

1. Qual a principal diferença entre a instrução if e a instrução else?

a. O if sempre executa um bloco de código, enquanto o else nunca executa nada.

b. O if executa um bloco de código quando a condição seja verdadeira, e o else executa quando a condição do if é falsa.

c. O if é usado para repetições, e o else para condições.

d. Não existe diferença entre if e else.

2. Quando é mais apropriado usar o laço de repetição for?

a. Quando o número de repetições não é conhecido.

b. Quando o número de repetições é determinado antes da execução do laço.

c. Quando se deseja executar o código pelo menos uma vez antes de verificar a condição.

d. Quando se deseja fazer a repetição sem nenhuma condição.

3. Qual é a principal vantagem do uso da instrução switch sobre múltiplos if...else?

a. O switch pode verificar apenas duas condições, enquanto o if...else pode verificar várias.

b. O switch torna o código mais limpo e organizado ao lidar com múltiplas opções.

c. O switch permite a execução do código sem condições, enquanto o if... else exige uma condição para cada bloco.

d. O switch pode ser usado apenas com números inteiros, enquanto o if... else pode ser usado com qualquer tipo de dado.

4. Qual a principal característica do laço do...while?

a. A condição é verificada antes da execução do bloco de código.

b. O código dentro do laço é executado pelo menos uma vez, independentemente da condição.

c. O laço nunca executa o código se a condição for falsa.

d. A condição do do...while não pode ser alterada durante a execução.

Gabarito Atividade Prática

1. b. O if executa um bloco de código quando a condição seja verdadeira, e o else executa quando a condição do if é falsa.

Comentário: O if executa o bloco de código quando a condição que ele verifica é verdadeira. Já o else só é executado quando a condição do if é falsa, fornecendo uma alternativa

2. b. Quando o número de repetições é determinado antes da execução do laço.

Comentário: O for é mais apropriado quando o número de iterações (repetições) é conhecido, como contar de 0 até 4, por exemplo.

3. b. O switch torna o código mais limpo e organizado ao lidar com múltiplas opções.

Comentário: O switch é ideal quando há muitas opções a serem verificadas, pois permite um código mais organizado e fácil de ler do que múltiplos if... else. Ele também facilita a manutenção do código.

4. b. O código dentro do laço é executado pelo menos uma vez, independentemente da condição.

Comentário: A principal característica do do...while é que ele executa o código pelo menos uma vez antes de verificar a condição. Mesmo que a condição seja falsa, o código será executado ao menos uma vez.

Ir para exercício