Matemática

Javascript

Ruan Gustavo de Oliveira - 13/09/2024

JavaScript

Até o momento, aprendemos que a linguagem de programação já nos proporciona algumas funcionalidades prontas, que nos ajuda a programar e criar sistemas, sites e apps de forma bem mais rápidas.

Vimos funcionalidades com:

- console.log(value): Função que imprime valores recebidos no console do navegador.
    - Ex: console.log(“Hello World! I am N” + 32, “A noob on computer programming”) // Output: “Hello World! I am N32 A noob on computer programming”
- “a”.charCodeAt(0): Função que recupera o endereço na tabela ASCII do character correspondente da String.
    - Ex: “abc”.charCodeAt(1) // Output: 98 Representa a Letra B na Tabela ASCII
- String.fromCharCode(97): Função responsável por retornar o character correspondente pelo código de endereço passado.
    - Ex: String.fromCharCode(97) // Output: “a"
    - Ex 2: String.fromCharCode(79, 108, 97, 32, 77, 117, 110, 100, 111, 33) // Output: "Ola Mundo!"

JavaScript

Também vimos algumas estruturas de algoritmos para executar algumas ações, como estruturas de loops (laços de repetições) e tomadas de decisões (verificações de estados e condições).

Laços de Repetições (FOR)

Em programação podemos utilizar algumas estruturas que executam ações repetidas vezes até que uma condição se satisfaça para que o algoritmo consiga sair desse laço e executar o restante do código.

Por exemplo, em nossas vidas utilizamos algumas desses laços dee repetição e executar ações repetidas até que uma condição nos tire desse laço e nos leve a outras ações (as vezes essas ações são outros laços de repetição 😁)

Por exemplo: “Siga caminhando em frente até passar 3 quadras, depois vire a direita e caminhe mais 4 quadras para chegar ao seu objetivo.”

JavaScript

Na página anterior, vimos um exemplo de execução de laço de repetição em nossas vidas, como nós podemos representar essas instruções de forma algorítmica?

1. Inicio
2. Passo 1: Comece a caminhar em frente.
3. Passo 2: Enquanto o numero de quadras passado é inferior a 3, continue caminhando.
4. Passo 3: Após pass 3 quadras, vire a direita.
5. Passo 4: Continue caminhando em frente.
6. Passo 5: Enquanto numero de quadras passado após virar a direita for menor que 4, continue caminhando.
7. Passo 6: Após passar 4 quadras, você chegou no seu objetivo.
8. Fim

O laço de repetição FOR, é uma das variadas formas de se fazer laços de repetição em programação, para que possamos utilizado, nós precisamos dizer a ele quais as condições ele deve seguir até seu objetivo, segue o exemplo a seguir

JavaScript

O laço de repetição FOR (PARA), é uma das variadas formas de se fazer laços de repetição em programação, e é ele que vamos utilizar nos nossos próximos exercícios, para que possamos utiliza-lo, nós precisamos dizer a ele quais as condições ele deve seguir até seu objetivo, segue o exemplo a seguir.

```
PARA X = 1; enquanto X < 100; some X++ {
        imprimir("Valor de X:", X)
}
```

No exemplo acima vimos a estrutura de como dizemos ao laço FOR, nós falamos para ele quais condições ele deve seguir repetindo até que a condição de parada seja atingida e ele possa parar de repetir as ações. Vamos lá:

Para X = 1; // X agora vale 1
E Enquanto X menor que 100; // Enquanto o valor de X for menor que 100, ele deve repetir a ação.
Some o valor de X mais 1; // Sem que o laço voltar ao inicio, somará o valor de X mais 1.
Enquanto o laço se repete até X satisfazer a condição de parada, o laço deve imprimir o valor de X.

O exemplo é simples e mostra como é a estrutura de um FOR, mas como ele executa isso e nos mostra o resultado?

JavaScript

PARA X = 1**; enquanto** X < 6**; some** X++ {
    imprimir(“Valor de X:”, X)
}

Exemplo do algoritmo rodando:

1. Iteração:

1. **`x = 1`, condição `1 < 6` é verdadeira.**
2. **Imprime: "Valor de X: 1”**
3. **`x` é somado com 1, e x agora é 2.**

2. Iteração:

1. **`x = 2`, condição `2 < 6` é verdadeira.**
2. **Imprime: "Valor de X: 2”**
3. **`x` é somado com 1, e x agora é 3.**

3. Iteração:

1. **`x = 3`, condição `3 < 6` é verdadeira.**
2. **Imprime: "Valor de X: 3”**
3. **`x` é somado com 1, e x agora é 4.**

4. Iteração:

1. **`x = 4`, condição `4 < 6` é verdadeira.**
2. **Imprime: "Valor de X: 4”**
3. **`x` é somado com 1, e x agora é 5.**

5. Iteração:

1. **`x = 5`, condição `5 < 6` é verdadeira.**
2. **Imprime: "Valor de X: 5”**
3. **`x` é somado com 1, e x agora é 6.**

6. **Após a 5º iteração, X se tornou 6. A condição X < 6 se tornou falsa, pq agora X igual a 6, então a condição de parada do loop foi atingida. Algoritmo saí do Loop.**

Resultado do algoritmo:

Output: “Valor de X: 1"
Output: “Valor de X: 2"
Output: “Valor de X: 3”
Output: “Valor de X: 4"
Output: “Valor de X: 5”

JavaScript

Segue o exemplo de um laço de repetição em Javascript.

```
let target = 100;

for (let i = 0; i < 100; i++) {
    console.log("Number:", i);
}
```

Copie o Exemplo ao lado, e cole no console do seu navegador, execute para ver o comportamento.

Tomadas de decisões (IF ELSE)

Em programação podemos utilizar algumas estruturas que nos ajudam em tomadas de decisão, essas construção de tomada de decisão permite ao programa executar diferentes ações com base em uma condição. Ajuda o algoritmo a decidir qual caminho tomar dependendo se uma determinada condição é verdadeira ou falsa.

Nós já utilizamos estruturas de tomadas de decisão “IF ELSE” em exercícios anteriores.

Com base no que já aprendemos sobre essa estrutura em exercícios anteriores, nós conseguimos tomar decisões para definir se uma letra é maiúscula ou minúscula, checamos se um valor está dentro de um intervalo de valores (ranges, Ex: 2 está entre 1 e 4?), e se uma condição é verdadeira ou falsa (Boolean).

Tomadas de decisões (IF ELSE)

Em nossa vida, sempre utilizamos das tomadas de decisões para realizar ações condizentes, segue um exemplo: “Hoje irei almoçar no restaurante, se tiver churrasco, comerei carne bovina, se não, comerei frango.”

1. Inicio
2. Passo 1: Verificar se tem churrasco.
3. Passo 2: Tem churrasco? Servir carne bovina.
4. Passo 3: Não tem churrasco? Servir frango.
5. Fim

As tomadas de decisões estão em vários momentos do nosso dia a dia, e na programação não é diferente, sempre teremos que falar para a maquina que decisões devem ser tomadas, e ela terá que decidir qual caminho tomar. Mas e como falamos isso para o computador, e como ele vai saber qual o caminho correto a decidir?

Para isso usamos o IF ELSE, onde podemos explicar forma algorítmica como a decisão deve ser tomada, e quais valores e condições se deve levar em consideração no momento da decisão.

Tomadas de decisões (IF ELSE)

Segue o exemplo a seguir de uma tomada de decisão utilizando IF ELSE (SE SENÃO)

```
SE X < 10 {
    imprimir(“O Valor de X é Menor que 10”)
} SENÃO {
    imprimir(“O Valor de X é Maior ou igual a 10”)
}
```

No exemplo acima vimos a estrutura de como dizemos ao computador como ele deve tomar uma decisão. Foi dado uma condição e dois caminhos. Vamos lá:

SE X < 10; // X é menor que 10
Imprima “O Valor de X é Menor que 10” // Se X é menor que 10, Ex: X = 2, o código deve imprimir a String.
SENÃO; // X é maior ou igual a 10;
Imprima “O Valor de X é Maior ou igual a 10” // Se X é maior ou igual a 10, Ex: X = 14, o código deve imprimir a Outra String.

O exemplo é simples e mostra como é a estrutura de um IF ELSE, mas como ele executa isso e nos mostra o resultado?

JavaScript

SE X < 10 {
imprimir(“O valor de X é menor que 10”)
} SENÃO {
imprimir(“O valor de X é maior ou igual que 10”)
}

Exemplo do algoritmo rodando:

1. X = 32

2. Verificação:

1. **`x < 10`, condição X < 10 é falsa, X é 32.**
2. **Pula para SENÃO**
3. **Imprime:** "O valor de X é maior ou igual que 10”

Resultado do algoritmo:

Output: “O valor de X é maior ou igual que 10"

JavaScript

Segue o exemplo de uma tomada de decisão em Javascript.

```
let x = 10;
if (x < 10) {
    console.log("O valor de X é menor que 10");
} else {
    console.log("O valor de X é maior ou igual a 10");
}
```

Copie o Exemplo ao lado, e cole no console do seu navegador, execute para ver o comportamento.

Tomadas de decisões (IF ELSE)

A forma de tomadas de decisões podem ser aplicadas de inúmeras formas, irei mostrar algumas delas utilizando linguagem de programação Javascript.

Nós podemos realizar múltiplas tomadas de decisões, podemos utilizar múltiplas condições, e elaborar tomadas de decisões muito mais complexas. Para isso nós podemos utilizar de alguns operadores lógicos que nos ajudam nessas tomadas de decisões.

- Operador E (&&): Representado por DOIS E's comerciais (&&), o operador && é o operador que nos ajuda que duas ou mais condições sejam satisfeitas para que a tomada de decisão esteja correta.
- Operador OU (||): Representado por DUAS PIPES (||), o operador || nos ajuda a para quando precisamos que pelo menos uma das condições necessárias estejam satisfeitas para tomadas de decisão.

Tomadas de decisões (IF ELSE)

Operador && (E)

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

Copie o Exemplo ao lado, e cole no console do seu navegador, execute para ver o comportamento.

No exemplo, foi colocado duas condições, e as duas devem ser verdadeiras para que satisfaça a tomada de decisão. Caso uma delas seja falsa, outra pessoa terá que dirigir.

Tomadas de decisões (IF ELSE)

Operador || (OU)

```
let temperatura = 5;
let chovendo = false;
if (temperatura <= 10 || chovendo === true) {
    console.log("Hoje é um ótimo dia para ficar em casa jogando. xD")
} else {
    console.log(“Vixi! Terei que trabalhar. :(”)
}
```

Copie o Exemplo ao lado, e cole no console do seu navegador, execute para ver o comportamento.

No exemplo, foi colocado duas condições, e uma das duas devem ser verdadeiras para que satisfaça a tomada de decisão. Caso as duas sejam falsas, terá que trabalhar.

Tomadas de decisões (IF ELSE)

Além dos operadores lógicos, é possível criar muitos caminhos que podem tomados nas tomadas de decisões.

- IF and ELSE: A tomada de decisão mais simples, onde existem dois caminhos, e uma delas será tomada de acordo com a condição proposta.
- IF, ELSE IF, ELSE (SE, ENTÃO SE, ENTÃO): Dessa forma podemos realizar várias tomadas de decisão, decidindo por múltiplos caminhos a serem tomados.
  - Ex: Se tem café, tomarei café com leite, então se tem Nescau, tomarei leite com Nescau, se não, vai Toddy mesmo.

Copie o Exemplo ao lado, e cole no console do seu navegador, execute para ver o comportamento.

```
let temperature = 25;
if (temperature > 30) {
    console.log("Hoje estâˆšá muito calor! Terei que ir trabalhar para aproveitar o ar Condicionado! :/");
} else if (temperature >= 20 && temperature <= 30) {
    console.log("Hoje ta gostosin, dia de jogar xD.");
} else {
    console.log("Hoje estâˆšá muito frio. Terei que ficar em casa e jogar. :/");
}
```

Informações Adicionais

Comparadores

Quando se trata de tomadas de decisões, nós utilizamos operadores de comparação para determinar a condição, sendo eles:

$$z$$

- > (Maior que): Define se um valor é maior que outro valor. Ex: 10 > 8 // 10 é maior que 8?
- < (Menor que): Definee se um valor é menor que outro valor. Ex: 6 < 11 // 6 é menor que 11?
- == (é igual) (em JS é: ===): Define se um valor é igual a outro. Ex (JS): 10 === 10. Ex (Outras linguagens): 10 == 10. // 10 é igual a 10?
- != (diferente) (em JS é: !==): Define se um valor é diferente de outro. Ex (JS): “A" !== “B”, Ex (Outras linguagens): “A" != “B" // A é igual a B?
- >= (Maior ou igual): Define se um valor é maior ou igual a outro. Ex: 10 >= 8?
- <= (Menor ou igual): Define se um valor é menor ou igual a outro. Ex: 7 <= 9?

Informações Adicionais

Comparadores

A diferença de “===" e “==“, “!=“ e “!==“, é pertencente a algumas linguagens de programação, como o Javascript, são utilizado para validar se os dois valores são estritamente iguais, por exemplo:

“A" === “A”, nesse exemplo, ele compara se os dois valores são do Tipo **Strings**, e se os dois temo mesmo valor, que no caso o valor é **A.**

O mesmo vale para o operador de diferença “!==“, que válida se os dois valores tem tipos e valores diferentes. Ex:

10 !== “A”, ne exemplo, os valores contém tipos e valores diferentes, um sendo **numérico** com valor **10**, e outra do tipo **String** com valor **A**

Operadores Matemáticos

Matemática

Operator	Operation	Example	Explanation
`+`	Soma	`a + b`	Soma dois valores
`-`	Subtração	`a - b`	Subtrai um valor de outro
`*`	Multiplicação	`a * b`	Multiplica um valor por outro
`/`	Divisão	`a / b`	Divide um valor por outro
`%`	Módulo	`a % b`	Pega o resto da divisão
`**`	Exponencial	`a ** b`	Realiza o calculo exponencial
`++`	Incrementa	`a++`	Acrescenta `mais` 1
`--`	Decrementa	`a--`	Diminui em 1
`-`	Negação Unária	`-a`	Nega os valores
`+`	Adição Unária	`+a`	Converte `valores` para numeros

Exercicios

Javascript

Aproveitando tudo que aprendemos até o momento, vamos realizar algumas atividades em Javascript, lembrando, que para resolve-los vc terá que utilizar sua criatividade com os conhecimentos contido nesse documento.

1. Faça um algoritmo que imprima todos os valores PAR contidos no intervalo de 0 a 100.
2. Faça um algoritmo que imprima todos os valores Impares contidos no intervalo de 0 a 100.
3. Faça um algoritmo que imprima o exponencial de um valor, **SEM UTILIZAR OS OPERADORES MATEMÁTICOS "**"**, utilize eles para validar se a saída de resultado do seu algoritmo está correta.
    1. Ex: 2 elevado a 2, é igual a 4
    2. Ex: 3 elevado a 3, é igual a 27
4. Faça um algoritmo que imprima todos os números primos em intervalo de valores.
    1. Ex de Entrada: 0 a 100 ou 32 a 80, ou 200 a 873.
    2. Ex Entrada e Saida: Entrada 10 a 50, Saida: 11 13 17 19 23 29 31 37 41 43 47

Exercicios

Javascript

5. Faça um algoritmo que resolva a seguinte equação b^2 - 4ac, onde o valor de B, A e C serão passados como entrada

    1. Ex Entrada e Saida: A = 1; B = -4; C = 2; Saida = 8

6. Faça um algoritmo que imprima a sequencia a baixo: F(n)=F(n–1)+F(n–2)

    1. Sequencia: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144

7. Faça um algoritmo que calcule o fatorial de um valor. n! = n * (n - 1) * (n - 2) * ... * 1)

    1. Ex: n = 5, o fatorial é: 5 * 4 * 3 * 2 * 1 = 120

Exercicios

Javascript

8. Faça um algoritmo que some os valores entre dois intervalos, e depois mostre o resultado da soma mais 1 é um valor primo.

    1. Dica: Use a lib Math.sqrt(valor).

    2. Ex Entrada: 1 a 7 = 28;

    3. Ex Saída:
        resultado da soma: 28
        resultado soma mais 1: 29
        O valor é Primo? R: sim

    4. Ex Entrada: 1 a 6 = 21;

    5. Ex Saída:
        resultado da soma: 21
        resultado soma mais 1: 22
        O valor é Primo? R: não

Exercicios

Javascript

9. Critical, Fatal e Puff: Escreva um algoritmo que imprima os números de 1 a 100. Mas para múltiplos de 3, imprima “Critical (valor do numero multiplo)” em vez do número, e para múltiplos de 5, imprima “Fatal (valor do numero multiplo)". Para números que são múltiplos de 3 e 5, imprima “Fatal Critical (valor do numero multiplo)”. Para números que não múltiplos de 3 e 5, imprima “Puff".

    1. Ex Saída: “Critical 3”, “Fatal 5”, “Critical Fatal 15”, "Puff"

10. Escreva um algoritmo que calcule a soma de todos os números pares e todos os números ímpares separadamente entre dois intervalos de valores.

    1. Ex entrada: 0 a 10;

    2. Ex Saida
       Par: 30
       Impar: 25

Exercicios

Javascript

11. Faça um algoritmo que some quantos multiplos de 3, multiplos 5 e multiplos de 3 e 5 existem entre o intervalo de dois valores.

    1. Ex Entrada: 0 a 10;

    2. Ex Saida:
       Quantidade Multiplos de 3: 3
       Quantidade Multiplos de 5: 2
       Quantidade Multiplos de 3 e 5: 1

12. Média de números: Escreva um algoritmo para encontrar a média de todos os números de 1 a n.

    1. Ex Extrada: 0 a 125;

    2. Ex Saida: Média: 63

Exercicios

Javascript

13. Escreva um algoritmo que imprima a pirâmide numérica estilo escada entre 1 e um valor n de linhas (altura da piramide).
    1. Ex entrada n: n = 10
    2. Ex Saida:

```
01
02 02
03 03 03
04 04 04 04
05 05 05 05 05
06 06 06 06 06 06
07 07 07 07 07 07 07
08 08 08 08 08 08 08 08
09 09 09 09 09 09 09 09 09
10 10 10 10 10 10 10 10 10 10
```

Exercicios

Javascript

14. Escreva um algoritmo para verificar se um determinado número n é um número perfeito. Um número é perfeito se a soma dos seus divisores (exceto ele mesmo) é igual ao próprio número.
    1. Ex entrada e Saida: n = 6; 6 é um numero perfeito.