



TRANSFORMANDO DO VISUALG PARA JAVASCRIPT

LÓGICA DE PROGRAMAÇÃO E ALGORITMOS

CST em Desenvolvimento de Software Multiplataforma

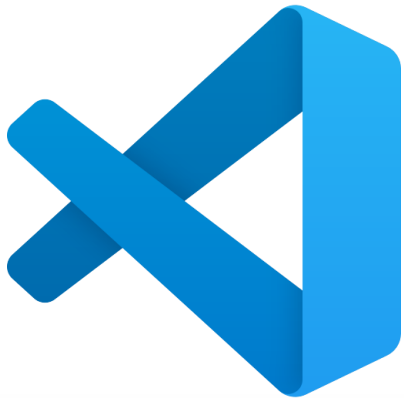


PROF. Me. TIAGO A. SILVA



AMBIENTE DE DESENVOLVIMENTO

- Para transformar o pseudocódigo do Visualg para JavaScript iremos precisar das ferramentas: a linguagem de programação **JavaScript**, IDE de desenvolvimento **Visual Studio Code** e o **NodeJS** como interpretador local.



VARIÁVEIS, ENTRADA E SAÍDA

VARIÁVEIS, ENTRADA E SAÍDA

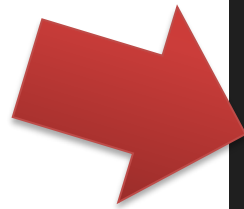
```
1 Algoritmo "Visualg_Para_Js"
2
3 Var
4     nome : caracter
5     idade: inteiro
6     salario : real
7 Inicio
8     Escreval("Qual o nome?")
9     Leia(nome)
10    Escreval("Qual idade?")
11    Leia(idade)
12    Escreval("Qual salário?")
13    Leia(salario)
14    Escreva("Olá ", nome, " vc tem ")
15    Escreva(idade, " e ganha R$ ", salario)
16 Fimalgoritmo
```

ENTRADA DE DADOS NO CONSOLE COM NODE E JS

```
1  // Para usar o equivalente ao Leia, você precisa instalar o pacote readline-sync.
2  // Para isso, abra o terminal e digite:
3  // npm install readline-sync
4  // Quando for executar, use o terminal e digite:
5  // node leia.js
6  // "leia.js" é o nome do arquivo que você salvou o código.
7  const read = require('readline-sync');
8  let nome = read.question("Qual seu nome?");
9  console.log(nome);
10
11 let n1 = read.question("Digite um numero: ");
12 let n2 = read.question("Digite outro numero: ");
13 let resultado = parseInt(n1) + parseInt(n2);
14 console.log(nome + "o resultado deu: " + resultado);
15 // Tudo que o usuário digita vem como string,
16 // por isso precisamos converter para número usando parseInt
```

VARIÁVEIS E SAÍDAS EM VISUALG E JAVASCRIPT

```
1 Algoritmo "Visualg_Para_Js"  
2  
3 Var  
4     nome : caracter  
5     idade : inteiro  
6 Inicio  
7  
8     nome <- "João"  
9     Escreval("O nome é ", nome)  
10  
11 Fimalgoritmo
```



```
1 // Declarando uma variável  
2 let nome = "João"; // Variável que pode mudar  
3 const idade = 42; // Constante que não muda  
4  
5 // Saída no Terminal: O nome é: João  
6 console.log("O nome é: ", nome);
```

CARACTERÍSTICAS DE VARIÁVEIS EM JAVASCRIPT

Característica	var	let	const
Escopo	Função ou global	Bloco	Bloco
Hoisting	Sim, inicializada como undefined	Sim, mas não acessível antes da declaração (temporal dead zone)	Sim, mas não acessível antes da declaração (temporal dead zone)
Redeclaração	Permitida	Não permitida	Não permitida
Atualização de Valor	Permitida	Permitida	Não permitida (exceto mutação de objetos e arrays)
Uso Preferencial	Evitar	Usar quando o valor pode mudar	Usar para valores que não mudam

OPERADORES

COMPARAÇÃO ENTRE OPERADORES

CATEGORIA	OPERADOR NO VISUALG	OPERADOR NO JAVASCRIPT	DESCRIÇÃO
Atribuição	< -	=	Atribui um valor a uma variável.
Igualdade	=	== ou ===	Verifica se os valores são iguais. === também compara os tipos no JavaScript.
Diferença	< >	!= ou !==	Verifica se os valores são diferentes. !== também compara os tipos no JavaScript.
Maior que	>	>	Verifica se um valor é maior que outro.
Menor que	<	<	Verifica se um valor é menor que outro.
Maior ou igual	>=	>=	Verifica se um valor é maior ou igual a outro.
Menor ou igual	<=	www.tiago.blog.br <=	Verifica se um valor é menor ou igual a outro.

COMPARAÇÃO ENTRE OPERADORES

CATEGORIA	OPERADOR NO VISUALG	OPERADOR NO JAVASCRIPT	DESCRIÇÃO
Adição	+	+	Soma dois valores ou concatena strings no JavaScript.
Subtração	-	-	Subtrai um valor de outro.
Multiplicação	*	*	Multiplica dois valores.
Divisão	/	/	Divide um valor pelo outro.
Módulo (resto)	%	%	Retorna o resto da divisão inteira.
Exponenciação	<i>Não disponível</i>	**	Eleva um número a uma potência no JavaScript.

COMPARAÇÃO ENTRE OPERADORES

CATEGORIA	OPERADOR NO VISUALG	OPERADOR NO JAVASCRIPT	DESCRIÇÃO
E lógico	E	&&	Retorna verdadeiro se ambas as condições forem verdadeiras.
OU lógico	OU		Retorna verdadeiro se uma das condições for verdadeira.
Negação lógica	NAO	!	Inverte o valor lógico.

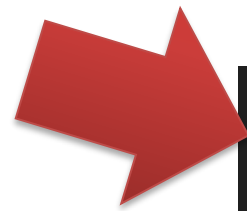
COMPARAÇÃO ENTRE OPERADORES

CATEGORIA	OPERADOR NO VISUALG	OPERADOR NO JAVASCRIPT	DESCRIÇÃO
Concatenação de strings	+	+	Concatena duas strings.
Incremento	<i>Não disponível</i>	++	Incrementa o valor da variável em 1 no JavaScript.
Decremento	<i>Não disponível</i>	--	Decrementa o valor da variável em 1 no JavaScript.
Atribuição com operação	<i>Não disponível</i>	+=, -=, *=, /=, %=	Realiza a operação e atribui o resultado à variável.

ESTRUTURAS DE CONTROLE

ESTRUTURA CONDICIONAL SIMPLES: SE

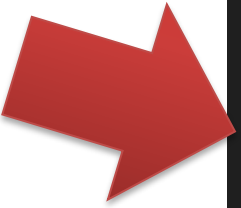
```
1 Algoritmo "Visualg_Para_Js"  
2  
3 Var  
4   idade : inteiro  
5 Inicio  
6  
7   Se (idade = 25) Então  
8     Escreval("Condição verdadeira")  
9   FimSe  
10  
11 Fimalgoritmo
```



```
1 let idade;  
2  
3 if(idade == 25) {  
4   console.log("Condição verdadeira");  
5 }
```

ESTRUTURA CONDICIONAL COMPOSTA: SE SENÃO

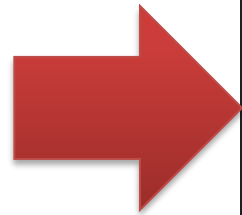
```
1 Algoritmo "Visualg_Para_Js"  
2  
3 Var  
4   idade : inteiro  
5 Inicio  
6  
7   Se (idade = 25) Então  
8     Escreval("Condição verdadeira")  
9   Senão  
10    Escreval("Condição falsa")  
11  FimSe  
12  
13 Fimalgoritmo
```



```
1 let idade;  
2  
3 if(idade == 25) {  
4   console.log("Condição verdadeira");  
5 } else {  
6   console.log("Condição falsa");  
7 }
```

CLASSIFICAÇÃO COM ESCOLHA CASO

```
1 Algoritmo "Visualg_Para_Js"
2
3 Var
4     estado : caracter
5 Inicio
6     estado <- "SP"
7     Escolha(status)
8     Caso "SP"
9         Escreval("São Paulo")
10    Caso "MG"
11        Escreval("Minas Gerais")
12    Caso "CE"
13        Escreval("Ceará")
14    Outrocaso
15        Escreval("Estado Inválido")
16    FimEscolha
17 Fimalgoritmo
```




```
1 let estado = "SP";
2
3 switch (estado) {
4     case "SP":
5         console.log("São Paulo");
6         break;
7     case "MG":
8         console.log("Minas Gerais");
9         break;
10    case "CE":
11        console.log("Ceará");
12        break;
13    default:
14        console.log("Estado inválido");
15        break;
16 }
```


ESTRUTURAS DE REPETIÇÃO

REPETIÇÃO PRÉ DETERMINADA: PARA

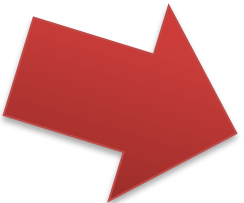
```
1 Algoritmo "Visualg_Para_Js"  
2  
3 Var  
4   i : inteiro  
5 Inicio  
6  
7   Para i de 1 ate 10 faça  
8     Escreval("Valor de i: ", i)  
9   FimPara  
10  
11 Fimalgoritmo
```



```
1  for(let i = 0; i < 10; i++) {  
2    |    console.log("O valor de i é: " + i);  
3  }
```

REPETIÇÃO INDETERMINADA: ENQUANTO

```
1 Algoritmo "Visualg_Para_Js"
2
3 Var
4   i : inteiro
5 Inicio
6
7   Enquanto (i < 10) Faça
8     Escreval("Valor de i: ", i)
9     i <- i + 1
10  FimEnquanto
11
12 Fimalgoritmo
```



```
1 let i = 0;
2 while(i < 10) {
3   console.log(i);
4   i = i + 1; // i++
5 }
```

REPETIÇÃO INDETERMINADA: REPITA

```
1 Algoritmo "Visualg_Para_Js"  
2  
3 Var  
4     parar : caracter  
5 Inicio  
6     Repita  
7         Escreval("Repetiu.")  
8         Escreval("Parar de repetir? S/n")  
9         Leia(parar)  
10    Até(parar = "S") ou (parar = "s")  
11    Escreval("Saiu do repita")  
12 Fimalgoritmo
```

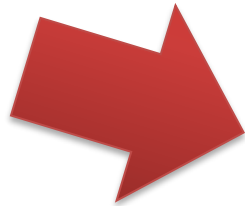
REPETIÇÃO INDETERMINADA: REPITA

```
1  ✓ // Equivalente do REPITA em JavaScript
2    // While equivale, neste caso, ao ATÉ
3    const read = require('readline-sync');
4    let parar = "n";
5  ✓ do {
6      console.log("Repetiu");
7      parar = read.question("Parar de repetir? S/n");
8      console.log(parar);
9  } while (parar !== "S" && parar !== "s");
10 // Note a diferença do operado, agora com AND (&&)
```

VETORES E MATRIZES

VETOR COM PARA

```
1 Algoritmo "Visualg_Para_Js"
2
3 Var
4   nomes : vetor[1..3] de caracter
5   i : inteiro
6
7 Inicio
8
9   nomes[1] <- "João"
10  nomes[2] <- "Maria"
11  nomes[3] <- "José"
12
13  Para i de 1 até 3 Faça
14    Escreval(nomes[i])
15  FimPara
16
17 Fimalgoritmo
```



```
1 // Declaração do vetor
2 let nomes = [];
3 nomes[0] = "João";
4 nomes[1] = "Maria";
5 nomes[2] = "José";
6 // Acessando cada posição do vetor
7 for (let i = 0; i < 3; i++) {
8   console.log(nomes[i]);
9 }
```

MATRIZES NO VISUALG

```
1 Algoritmo "Visualg_Para_Js"
2
3 Var
4     nomes: vetor[1..3, 1..2] de caracter
5     i, j : inteiro
6 Inicio
7
8     Para i de 1 até 3 Faça
9         Para j de 1 até 2 Faça
10             Escreval("Linha", i, " Coluna ", j)
11             Leia(nomes[i, j])
12         FimPara
13     FimPara
14 Fimalgoritmo
```


MATRIZES NO JAVASCRIPT

```
1  const read = require('readline-sync');
2  // Declaração da matriz
3  let nomes = [[]];
4
5  // Primeiro, acessa as linhas
6  for (let i = 0; i < 3; i++) {
7      // Depois, acessa as colunas
8      for (let j = 0; j < 2; j++) {
9          nomes = read.question("Linha " + i + " Coluna " + j + ": ");
10     }
11 }
```

FUNÇÕES E PROCEDIMENTOS

FUNÇÕES

```
1 Algoritmo "Visualg_Para_Js"
2
3 Funcao soma(a, b: real): real
4 Var
5     res : real
6 Inicio
7     res <- a + b
8     Retorne res
9 FimFuncao
10
11 Var
12
13 Inicio
14     Escreval( soma(2, 2) )
15 Fimalgoritmo
```

```
1 function somar(a, b)
2 {
3     let res = a + b;
4     return res;
5 }
6
7 console.log( somar(2, 3) ); // 5
```

PROCEDIMENTOS

1 Algoritmo "Visualg_Para_Js"

2

3 **Procedimento** saudacao(nome: caracter)

4 Var

5 msg : caracter

6 Inicio

7 msg <- "Olá " + nome

8 **Escreval**(msg)

9 **FimProcedimento**

10

11 Var

12

13 Inicio

14 saudacao("João")

15 Fimalgoritmo

```
1 function saudacao(nome)
2 {
3     let msg = "Olá " + nome;
4     console.log(msg);
5 }
6 saudacao("João"); // Olá Fulano
```

OBRIGADO!

- Encontre este **material on-line** em:
 - www.tiago.blog.br
 - Plataforma Teams
- Em caso de **dúvidas**, entre em contato:
 - **Prof. Tiago:** tiago.silva238@fatec.sp.gov.br

