Maiúsculas e minúsculas FAZEM diferença

Tente escolher nomes COERENTES para as variáveis

Evite ficar utilizando letras e números para nomear variáveis

Javascript não diferencia os números, sejam eles inteiros, reais, fracionado etc.

String = são palavras, frases

Boolean = true/false

typeof informa qual o tipo de dado constante na variável

Manipulação de dados

Convert String to number (3 ways):

Number.parseInt(n)

Number.parseFloat(n)

Number(n)

Convert number to string (2 ways):

String()

n.toString()

Formatando Strings

var s = ‘JavaScript’

‘Eu estou aprendendo s’ // nao faz interpolação

‘Eu estou aprendendo ’ + s // usa concatenação

`Eu estou aprendendo ${s}` // usa template string

s.length // quantos caracteres a string tem

s.toUpperCase() // tudo para ‘MAIÚSCULAS’

s.toLowerCase() // tudo para ‘MINÚSCULAS’

Template Strings

Nao usa mais aspas simples, mas sim ``(crase)

Forma antiga e formatação de strings:

'O aluno ' + nome + ' com ' + idade + ' anos tirou a nota ' + nota

'O aluno Marcos com 34 anos tirou a nota 8'

Forma nova:

`O aluno ${nome} com ${idade} anos tirou a nota ${nota}`

'O aluno Marcos com 34 anos tirou a nota 8'

Formatting numbers

Var n1 = 1543.5

n1.toFixed(2)

'1545.50'

n1.toFixed(2).replace('.', ',')

'1545,50'

n1.toLocaleString('pt-BR', {style: 'currency', currency: 'BRL'})

'R$ 1.545,50'

Operadores

Aritméticos: (Binários, necessitam de dois operandos)

**+** : soma 2+2 : 4

**-** : subtração 5-2 : 3

**\*** : multiplicação 5\*2 : 10

**/** : divisão 5/2 : 2.5

**%** : resto de divisão inteira 5%2 : 1 (5/2, o 1 é o resto da divisão de 5 por 2)

**\*\*** : potenciação 5\*\*2 : 25

Ordem de Precedência:

1 - ()

2 - \*\*

3 - \* / % (se tiver mais de um a ordem é da esquerda para direita)

4 - + -

Atribuição Simples:

Exemplo:

Var a = 5 + 3 : 8

Var b = a % 5 : 3

Var c = 5 \* b \*\* 2 : 45

Var d = 10 – a / 2 : 6

Var e = 6 \* 2 / d : 2

Var f = b % e + 4 / e : 3

Auto atribuição:

Var n = 3

n = n+4 : 7

n = n-5 : 2

n = n\*4 : 8

n = n/2 : 4

n = n\*\*2 : 16

n = n%5 : 1

A variável n foi assumindo outros valores com o passar de novos comandos, assim, a variável n começou com o valor de 3 e terminou com o valor de 1.

Simplificando: n **+=** 4 n -= 5 n \*= 4 n /= 2 n \*\*= 2 n %= 5

Utilizando o operador com o = (+=, -=, etc.), a linguagem Javascript entende que você está referenciando a própria variável e mais uma outra operação

Incremento:

Var x =5

x = x + 1 : 6

x = x - 1 : 5

x += 1 : 6

x -= 1 : 5

O operador de incremento é uma simplificação da simplificação, ou seja, o x += 1 pode ser substituído por x++ e o x -= 1 pode ser substituído por x--

x++(simplificado)

x-- (simplificado)

O incremento pode ser antes (pré-incremento) ++n ou depois (pós-incremento) n++

Operadores relacionais: (resultado SEMPRE booleano, true or false)

> maior – 5 > 2 : true

< menor – 7 < 4 : false

>= maior OU igual – 8 >= 8 : true

<= menor OU igual – 9 <= 7 : false

== igual – 5 == 5 : true

!= diferente – 4 != 4 : false

=== idêntico/igualdade restrita (mesmo valor e tipo) – 5 === ‘5’ : false, 5 === 5 : true

!== desigual restrito (totalmente diferentes em valor e tipo)

Operadores lógicos:

! negação – só tem um operando, operador unário, uma expressão depois do operador ex. !true or !false

&& conjunção (E) uma coisa E outra, ou seja, depende das duas condições para satisfazer. Operador binário.

|  |  |  |  |
| --- | --- | --- | --- |
| Condição 1 | Operador | Condição 2 | Resultado |
| true | && | true | true |
| true | false | false |
| false | true | false |
| false | false | false |

|| disjunção (OU) uma coisa OU outra. Operador binário

|  |  |  |  |
| --- | --- | --- | --- |
| Condição 1 | Operador | Condição 2 | Resultado |
| true | || | true | true |
| true | false | true |
| false | true | true |
| false | false | false |

Ordem de precedência:

1º - ! (Não)

2º - && (E)

3º - || (OU)

Operador ternário:

? : - Ele tem três partes, ou seja, três operandos, um teste lógico, o que acontecer quando o teste lógico for verdadeiro ou quando ele for falso

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Teste lógico | ? | true | : | false |

Ordem de precedência dos operadores:

|  |  |  |
| --- | --- | --- |
| Ordem | Operadores | Ordem dentro dos operadores |
| 1º | Aritméticos | 1º ( )  2º \*\*  3º \* / %  4º + - |
| 2º | Relacionais | Não possuem ordem de precedência, a leitura é da ESQUERDA para DIREITA |
| 3º | Lógicos | 1º !  2º &&  3º || |
| 4º | Ternário | Único, não tem ordem |

Árvore DOM (Document Object Model)

Selecionando elementos para navegar na árvore DOM

Por Marca (tagname) – getElementsByTagName() – pode selecionar mais de um objeto de uma vez utilizando o [], colocando um número que começa em 0, sendo o primeiro elemento que aparecer e vai indo, 1, 2, 3...

Por ID – getElementById ()

Por Nome – getElementsByName() – também pode selecionar mais de um objeto utilizando o []

Por Classe – getElementsByClassName ()

Por Seletor – querySelector () - querySelectorAll () Esse é o mais novo

ID é representado por #

Class é representado por .

DOM Events

Function é um conjunto de códigos, linhas que vão ser executadas só quando o evento ocorrer.

Um bloco em Javascript é delimitado por { }.

Function ação ( ) {

}

Detecção de erros em JS

O JS não mostra o erro diretamente no VS code, por isso tem que clicar com o botão direito no html aberto no navegador e ir em inspecionar, no canto superior direito ele mostra eventuais erros

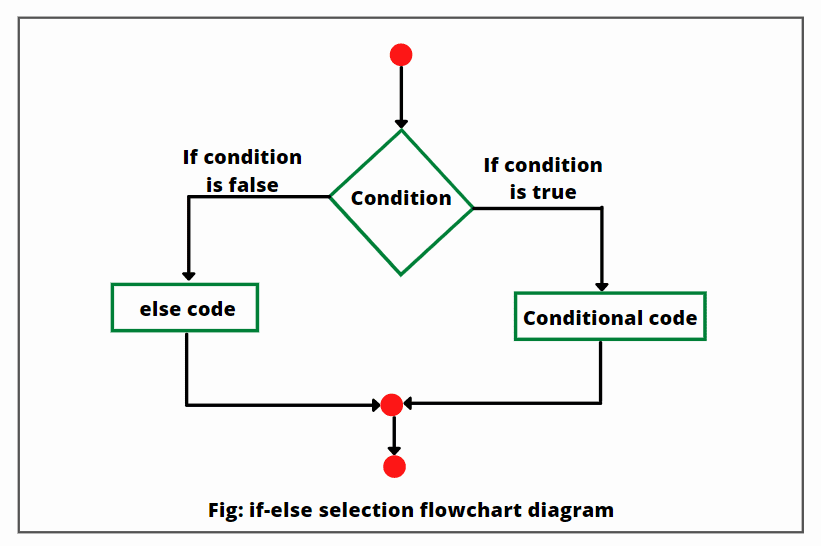
Condições em JS

If () {

} else {

}

É um losango no fluxograma que irá ter duas possibilidades, dois caminhos. Se acontecer determinada coisa, vai seguir o caminho da direita, se não esta coisa ou outra ocorrer outra coisa diferente, segue o caminho da esquerda.



Tipos de condições:

- Condição simples:

**if (condição) {**

**true**

**}**

- Condição composta:

**if (condição) {**

**true**

**} else {**

**}**

- Condições aninhadas: São condições dentro de condições.

**If (cond1) {**

**bloco 1**

**} else {**

**If (cond2) {**

**bloco 2**

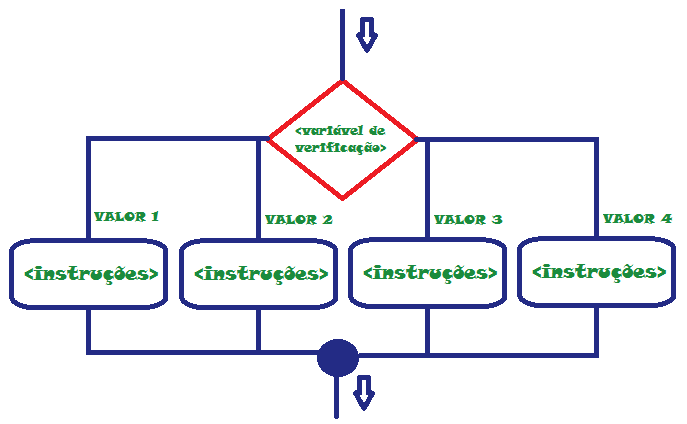
**} else {**

**bloco 3**

**}**

**}**

- Condição Múltipla:



Expressão:

Switch (expressão) {

case valor 1:

bloco 1

**break**

case valor 2:

bloco 2

**break**

case valor 3:

bloco 3

**break**

default:

bloco 4

**break**

}

SEMPRE UTILIZAR O **BREAK** AO FINAL DE CADA BLOCO.

Só funciona com números inteiros e strings.

Estruturas de repetição ou laços

1) while { }

2) do while {}

3) for (inicialização; teste lógico; incremento) – Ele tem três elementos para serem preenchidos nos parenteses.

Variáveis compostas, array ou vetores

Nada mais é que uma variável que tem vários elementos e cada elemento é composto por seu valor e por uma chave de identificação ou key

**O PRIMEIRO ÍNDICE É SEMPRE 0 E NAO 1**

**let num = [5,8,4]**

Aqui foi criado um vetor com três elementos, sendo que o primeiro elemento recebeu o valor 5 e a key 0, o segundo elemento recebeu o valor 8 e a key 1 e o terceiro elemento recebeu o valor 4 e a key 2. Caso queira adicionar mais um elemento neste array, basta digitar:

**num[3] = 6**

Assim, o JS vai entender que, como não há o elemento com a key 3 no array, ele adiciona este elemento que agora terá o valor 6 e a key 3. Se quiser adicionar um elemento na última posição do array, sem se importar exatamente qual é ela, basta digitar:

**num.push(7)**

Desta forma, no nosso array acima, este elemento teria o valor 7 e receberia a key 4, sendo colocado na última posição.

Para saber o comprimento do array (quantidade de elementos que ele possui) basta usar o código:

**num.length**

No nosso exemplo acima retornaria o valor 5, que indica que o array possui 5 elementos.

num

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 8 | 4 | 6 | 7 |
| 0 | 1 | 2 | 3 | 4 |

**num.sort()**

Este comando organiza o array em ordem crescente, pegando todos os elementos e organizando-os em ordem crescente de acordo com seu valor. Assim, no nosso exemplo o array ficaria da seguinte forma:

num

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 5 | 6 | 7 | 8 |
| 0 | 1 | 2 | 3 | 4 |

**console.log(num[0])** – Esse código é feito para exibir determinado valor do array.

Uma maneira melhor de exibir todos de uma vez e de forma personalizada é usando o for:

for (let pos = 0; pos < num.length; pos++) {

console.log(`A posição ${pos} tem o valor ${num[pos]}`);

}

Também há uma forma ainda mais simplificada que é:

for (let pos in num) {

console.log(`A posição ${pos} tem o valor de ${num[pos]}`)

}

**num.indexOf(7)**

Esse comando retorna a key na qual está presente o valor colocado entre parenteses, no nosso caso ele retornaria o valor 3, já que o 7 está na key de número 3. Caso coloque um valor que não existe no array, ele retornará o valor -1, que indica que o JS pesquisou no array mas não encontrou nenhuma key com o valor informado.

let pos = num.indexOf(8)

if (pos == -1) {

console.log(`O valor não foi encontrado!`)

} else {

console.log(`O valor está na posição ${pos}`)

}

Funções

São ações executadas assim que são chamadas ou em decorrência de algum evento.

Uma função **pode** receber parâmetros e retornar um resultado. Uma função em JS só pode ter um retorno.

function parimpar (n) {

if (n%2 == 0) {

return 'Par!'

} else {

return 'Ímpar!'

}

}

console.log(parimpar(8))

Parâmetros opcionais são valores pré-definidos que podem ser incluídos nos parâmetros, por exemplo:

function soma (n1=0, n2=0) {

return n1 + n2

}

console.log(soma(7,7))

No caso acima o parâmetro é (n1, n2), mas, opcionalmente, adicionamos o valor padrão para eles de 0, assim, caso a chamada inclua só um valor, não retornará o erro NaN (Not a number), mas sim executará a função considerando o valor informado somado de 0.

É possível também colocar uma função dentro de uma variável:

let v = function (x) {

return x\*2

}

console.log(v(5))

Função recursiva

É o caso de uma função que chama outra função, ou seja, podemos reescrever a função acima para descobrir o fatorial de um número com recursividade, pois sabemos que, matematicamente, o fatorial de um número é n! = n\*(n-1)!:

function fatorial(n) {

if (n == 1) {

return 1

} else {

return n \* fatorial(n-1)

}

}

console.log(fatorial(3))

Introdução a OBJETOS

É semelhante ao array, pois ele vai criar uma variável grande com diversos atributos (elementos), mas, diferentemente do array, a key dos valores também pode ser definida. Como exemplo, podemos citar a eventual criação de um cadastro de usuário, na qual queremos guardar o nome, o sexo, o peso e se ele engordou, vejamos:

cliente

|  |  |  |  |
| --- | --- | --- | --- |
| José | M | 85.4 | function engordar(p) { } |
| nome | sexo | peso | engordar() |

Diferentemente do array, o objeto pode conter em seu atributo uma function.