JavaScript curso introdutório

Variaveis

Tipos numeriocos

Funcoes

Estruturas condicionais

Console.log(“”) = imprimir

Alert(“”) = alertar uma mensagem na página

// = comentário

/\*

Comentário entre linhas

\*/

Variáveis e tipos de dados:

Três formas de variáveis:

Var: Declara uma variável podendo atribuir um valor junto na sua criação.

var umNumero;

var umNumero = 10;

let: Declara uma variável com escopo de bloco:

let outroNumero = 5;

const: Declara uma variável apenas de leitura. Após atribuir um valor a uma const, não será possível altera-lo:

const ola = “Olá Mundo!”

Importante:

* Apesar de ainda ser muito utilizado, o var não é mais recomendado pelas boas praticas de programação.
* Case-Sensitive: O JavaScript diferencia entre letras maiúsculas e minúsculas nas variáveis.
* Letras: Evitar cedilha e acentuação quando for declarar as variáveis.
* Numeros: Somente é possível utilizar números na declaração se o primeiro digito da variável for uma letra. Let carro2000 ou w3c
* Caracteres válidos: Underline ’\_’ e cifrão “$”: São permitidos em qualquer posição e são mais usados nas variáveis de bibliotecas e frameworks. Let \_variavel = 99; ou let$www = true;

O javaScrpit possui os seguintes tipos de dados:

* Number
* String
* Boolean
* Array
* Null e Undefined
* Symbol\*
* Object
* Function

Saber este tipo de informação é muito importante, pois ajuda a identificar erros.

Estruturas que recebem parâmetros e se informarmos o tipo errado, o programa poderá não funcionar como deveria.

Tipos de dados:

Apesar das variáveis das linguagens dinâmicas não precisarem ter seus tipos de dados identificados, isso não quer dizer que eles não existam.

Number:

São todos os números reais

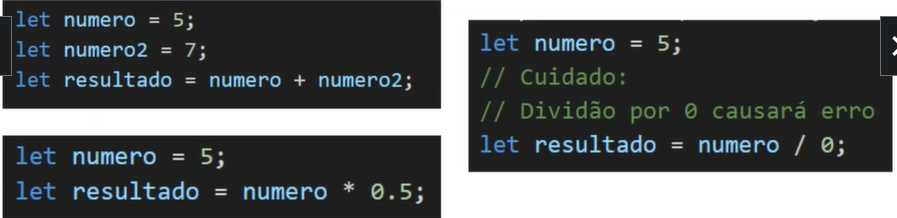
Let numeroInteiro = 100;

Let numeroDecimal = 5.38;

Let numeroNegativo = -200.99;

Operadores:

Podemos utilizar +, -, \* e / para executar as operações básicas com números

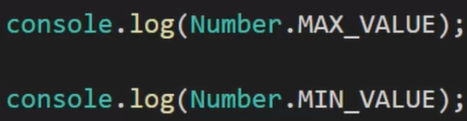


Importante:

Estes valores numéricos não são infinitos. Mesmo assim, o JavaScript trabalha com pronto flutuante de dupla precisão (double) que permite valores muito altos.

Máximo e Mínimo:

Para saber quais são estes valores máximos e mínimos:



Strings:

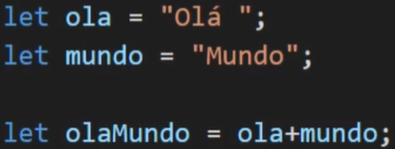
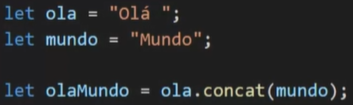
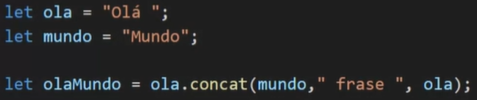
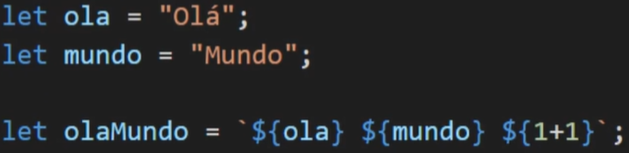
Valores de texto, normalmente chamados de “cadeia de caracteres”:

Let olaMundo = “Olá Mundo”;

Podemos utilizar aspas simples ou aspas duplas para encapsular a string

Caracteres Especiais – Para adicionar caracteres especiais (‘, “, &, quebra de linha e etc.) Utilzaremos a barra invertida (\).

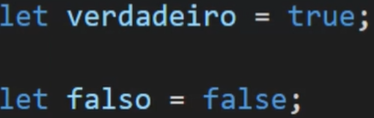
Concatenação de Strings:

1. Utilizar o sinal de + 
2. Utilizar o String.concat  Este método permite concatenar diversas strings, basta separar por virgula 
3. Template de Strings: é uma nova forma de concatenação utilizando o ${expressão} e a crase ao redor da string. 

Booleanos: São tipos de dados que possuem apenas dois valores:

TRUE: Verdadeiro

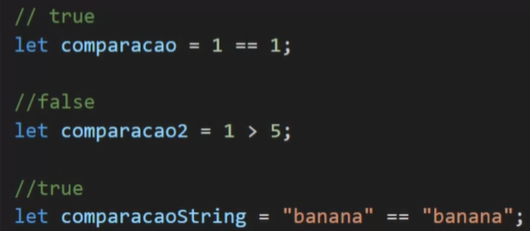
FALSE: Falso



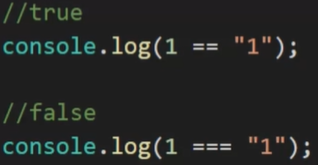
Observações: Os valores true e false não são escritos entre aspas.

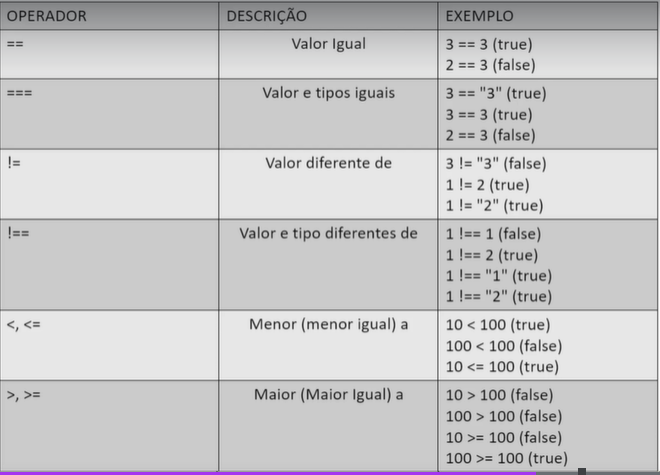
O JavaScript entende certos valores dos outros tipos de dados como true ou false.

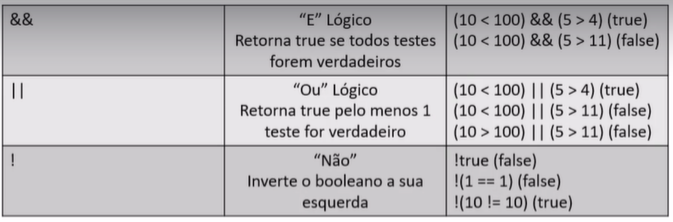
Comparações: Uma das formas mais comuns de se obter valores booleanos é através de comparações.



Comparações: Se você testar 1 == “1”, obterá true. Para fazer comparações que levam em consideração o valor e o tipo do dado, utilize ===





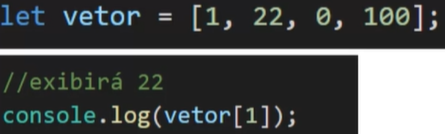


Array (vetor): É uma lista ou coleção de dados que pode ser acessada por índice. Para criar um vetor vazio basta criar uma variável e eatribuir [ ] a ela.

Let vetor = [1, 22, 0, 100 ];

Let vetor = [1, “Olá Mundo”, true, [1,2,3] ];

Para acessar os valores: Os valores podem ser acessados pelo seu índice.

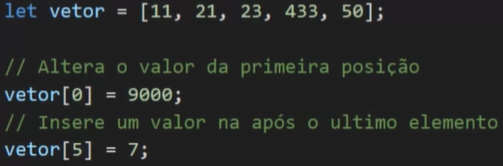


A indexação em javascript começa em 0.

Alterando e atribuindo valores pelo índice: Como o índice, você pode:

a) Alterar um valor existente

b) Inserir um novo valor em uma posição especifica.



Null: É um tipo de dado especial, ele representa a falta de valor de qualquer outro tipo de dado.

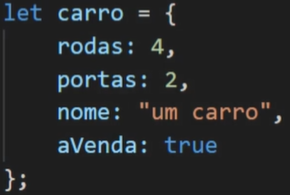
Undefined: Este tipo de dado aparece quando criamos uma variável e tentamos acessar seu valor antes de ter algo nela.

Undefined !== Null: undefined e null são diferentes.



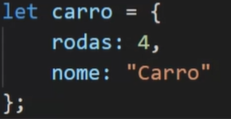
Reumidamente, isto occore pois null ainda é um valor e undefined é quando o JavaScript não sabe qual o tipo de dado.

Objeto: É um tipo de dado composto pelos outros tipos. Com ele, podemos organizar informações relacionadas em uma variável.

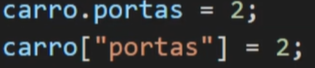


Criação: Um objeto vazio é bem simples de criar: 

No caso de um objeto com propriedades (variáveis), fazemos assim:

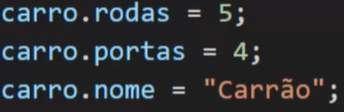
 importante separar as propriedades por virgula

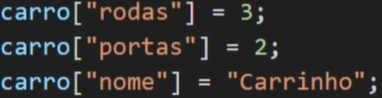
Caso você já tenha criado o objeto e queira adicionar um novo, você pode fazer de duas formas:



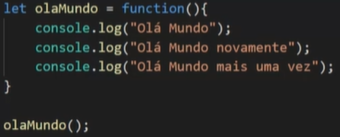
Se for utilizar a 2º opção precisa ter uma string dentro dos [ ].

A alteração de dos funciona da mesma forma.

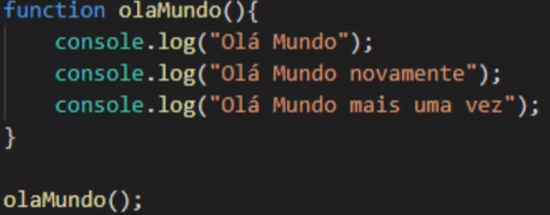




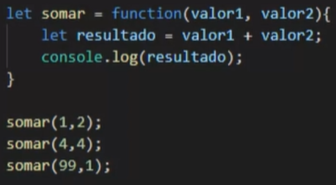
Funções: São utilizadas para criarmos uma sequência de operações para serem executadas



Outra forma de criar uma função é chamando o function com o nome dela.

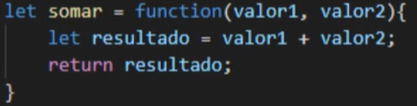


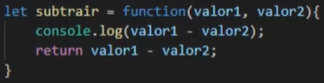
Também é possível passar valores para a função acessar.



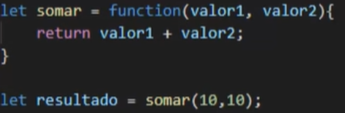
E também a outro recurso das funções, o Return.

O return para a função e devolve um valor





A utilizade do return é:



Uma função com return devolve um valor que podemos guardar em uma variável.

Condicionais: If, Else if, Else, Switch case

if(x == 1){

/Executa código1

} else{

/Executa código2

}

if(x == 1){

/Executa código1

} else if(1 < 3) {

/Executa código2

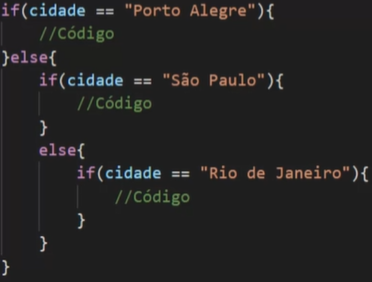
} else if(1 > 3) {

/Executa código3

}else{

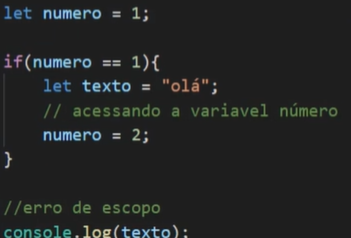
}



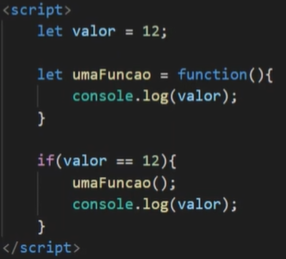


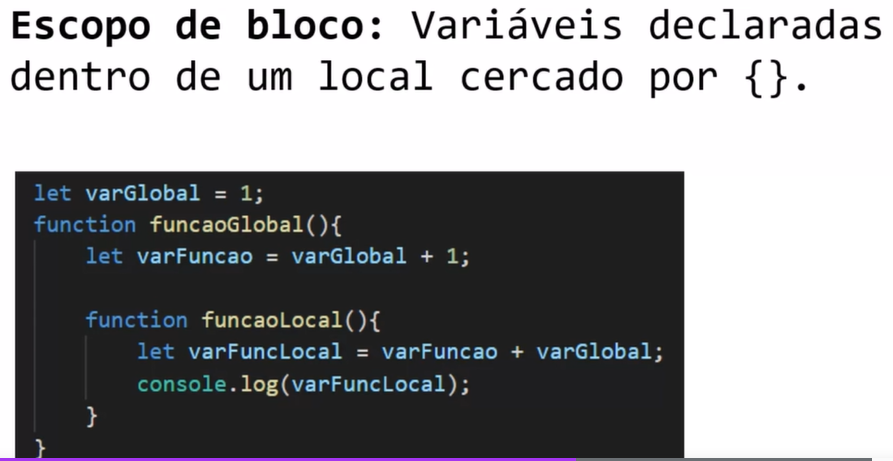
Escopo: O bloco de escopo é definido p

ela região ao qual variáveis e outros dados são visíveis.



Escopo global: Uma variável declarada fora de uma função, torna-se global e todos os scripts e funções em uma página da Web podem acessá-la.

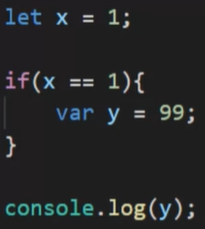




Essas variáveis não podem ser acessadas fora de seu escopo.

Var: Só possui escopo global e de bloco para funções.

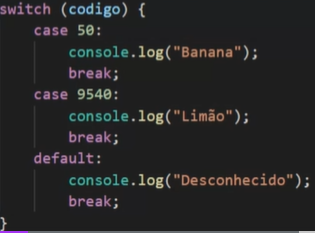
Ou seja o var se estiver dentro de um bloco ele consegue ser acessado de maneira global.



´

Na maioria dos casos usamos let por conta de saber que aquela variável vai estar vindo de um lugar aonde sabemos, ou seja facilita o controle de acesso a essas variável.

Switch: O switch serve para executar um código de acordo com diversos possíveis resultados.



Switch é uma boa alternativa quando temos uma quantidade grande de if/elseif caso existam muitas comparações de um valor.

Para cada possibilidade, é necessário ter um “case”

Break serve para encerrar a leitura do switch caso a condição seja atendida.

Default é o “else”, executará se nenhuma das condições for verdadeira ou corresponder ao valor

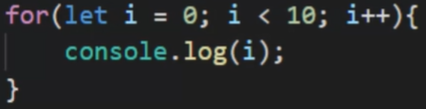
O switch compara tanto o dado quanto o valor ===, precisam ser iguais



Login com 2 objetos diferentes, 1 autenticado e 1 com erro

Loops: laços de repetição

FOR: O “for” é um comando que executa X vezes uma ação seguindo a forma como foi definido para ele faça, desde que a condição seja verdadeira;

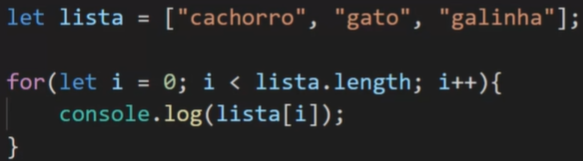


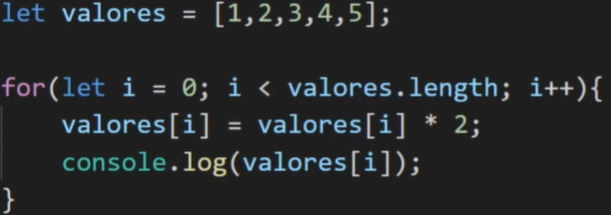
1º A primeira instrução é a criação da variável de controle let i = 0

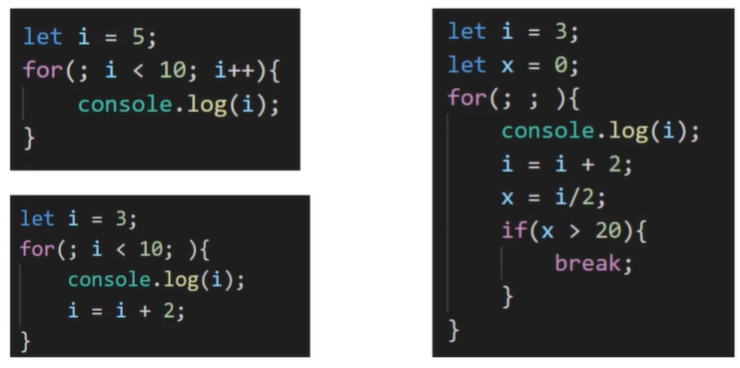
2º Configuração da condição i < 10

3º Configuração do método do loop, ou seja se vai incrementar i++, decrementar, colocar valores ao cubo etc.

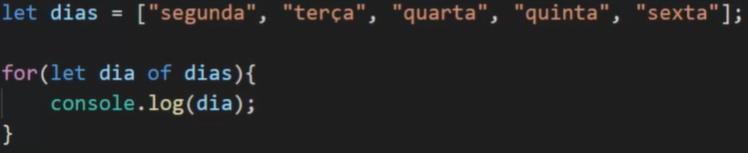
Utilidade: Além de executar uma mesma instrução varias vezes. Ele é perfeito para ler vetores...





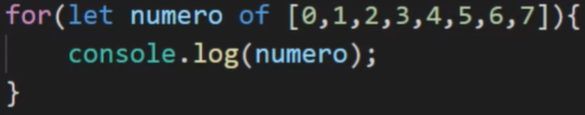


For OF: este loop é especifico para iterar entre os elementos de uma lista.

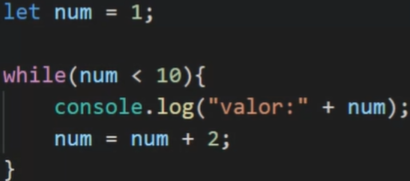


Percorre item por item, dependendo do tamanho da lista

For of Pode ser lido “Para cada item da lista”



While: este loop funciona de forma bem simples, ENQUANTO (WHILE), a condição for true, executa o código.



Do while: Faça Enquanto. Similiar o while, executa um código e depois testa se executará novamente.

A repetição é finalizada quando o teste é Falso(F), ou seja, enquanto o teste for verdadeiro as instruções serão executadas.

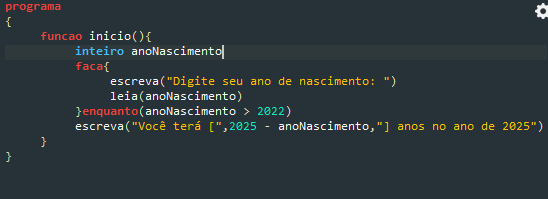
Os códigos sempre serão executados pelo menos uma vez, pois o teste(condição fica no final da repetição).

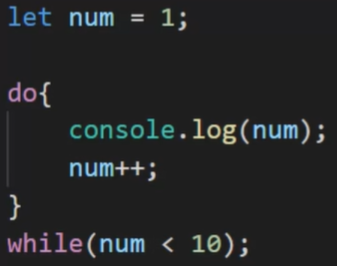
Faca{

// código que será repetido

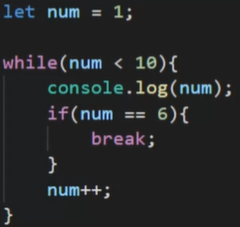
}enquanto(condição)

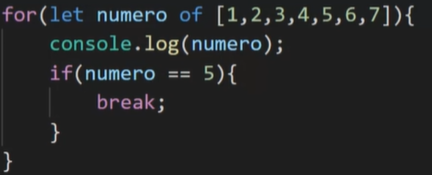
É bastastante utilizado em validação de entrada / E também exibição de menu



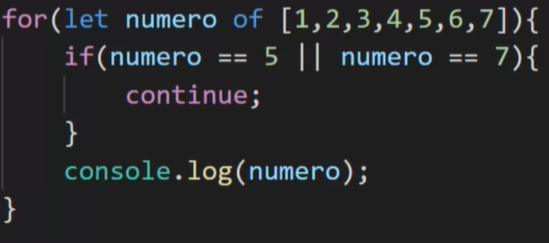


Break: É um comando que podemos utilizar quando queremos parar/encerrar o loop na hora.





Continue: Este comando é possível iniciar a próxima repetição do loop.



Nesse caso quando ele executa o primeiro for e chega em continue, ele retorna para o for novamente sem passar pelo console até terminar toda a repetição for.