**MÓDULO 1**

**AULA 4: Criando o seu primeiro Script**

ECMAscript é um tipo de padronização internacional para a linguagem de Javascript

Geralmente a tag <script> do JS é colocada antes do fechamento da tag </body>

Não é obrigatório colocar ; ao final dos comandos, diferentemente do CSS

O primeiro comando:

<script>

**window.alert**('Minha primeira mensagem!')

**window.confirm**('Está gostando de JS?')

**window.prompt**('Qual é seu nome?')

</script>

**Window** 🡪 Faz a mensagem aparecer assim o navegador carrega

**Confirm** 🡪 Aparece a tela com a mensagem e 2 com “ok” e “cancelar”

**Pompt** 🡪 O usuário consegue digitar algo na caixinha que abre.

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

**MÓDULO 2 – COMANDOS BÁSICOS**

**AULA 5: Variáveis e Tipos Primitivos**

Para fazer comentários no código use: **/\* SUA MENSAGEM AQUI \*/**

Vaga a1 = carro1

O sinal **=** significa **RECEBE**

A1 = **null** (nula, sem nada)

As “vagas” do computador (terreno do estacionamento) são chamadas de **MEMÓRIA**, e ela possui várias Vagas (**VARIÁVEIS, VAR** **OU LET**) que irão receber dados.

Toda variável precisa de um nome que irá receber esses dados , esse nome é chamado de **IDENTIFICADOR. (Ex: n1, n2, s1, s2)**

**ATRIBUIÇÃO:** É quando uma variável recebe um dado.

Assim como tem vagas de tamanhos diferentes, existem variáveis de tamanhos diferentes para caber dados de tamanhos diferentes.

Ex:

var s1 = “JavaScript”

var s2 = ‘Curso em Vídeo’

var s3 = `Guanabara`

Para letras 🡪 Pode ser aspas duplas, aspa simples ou crase.

Número não precisa.

**NOMECLATURA DE IDENTIFICADORES:**

**Podem começar com letra, $ ou \_ 🡪 Geralmente é letra**

**É possível usar letras ou números**

**É possível usar acentos e símbolos**

**Não podem começar com números**

**Não podem conter espaços.**

**Não podem ser palavras reservadas (function, alert, etc..)**

**Dicas:**

Maiúsculas e minúsculas fazem diferença

Tente escolher nomes coerentes para as variáveis

Ao usar uma nova atribuição para uma variável que já tinha um dado, ela perde o dado antigo e fica com o novo.

Para abrir o Node pelo VSC:

No VS Code 🡪 Menu Superior 🡪 Terminal 🡪 Novo terminal 🡪 Digita node e irá abrir o NodeJS

**TIPOS DE VARIÁVEIS PRIMITIVAS:**

**Number (infinity, NaN)** 🡪 Números inteiros ou não

**Strings** 🡪 Cadeia de caracteres ou conjuntos de números OU números entre “ “

**Boolean** 🡪 True ou False

**Null**

**Undefined**

**Object (Array)**

**Function**

No node digite **typeof** **termo** que você deseja saber e ele lhe dirá qual a classificação essa variável receberia

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

**AULA 6: Tratamento de dados**

Aula 06 - Ex002, Ex003 e Ex004

Para salvar o nome que a pessoa digitar quando executar o script prompt você precisa atribui-lo a uma variável, EX:

**var nome = window.prompt('Qual é seu nome?')**

Quando for chamar a variável em outro comando 🡪 Ela não deve estar dentro de “” Ex:

window.alert('É um grande prazer te conhecer,'+ **nome** + ‘!’)

**+** 🡪 Concatenação ou adição

(number + number) para adição

(string + string) para concatenação

**As caixas de texto pegam o que a pessoa digitar, mesmo que seja número, e transforma em string. Por isso que:**

var n1 = window.prompt ('Digite um número: ') 🡪 n1 é string

var n2 = window.prompt ('Digite outro número: ') 🡪 n2 é string

var s = n1 + n2

window.alert ('A soma dos valores é: ' + s)

O s irá mostrar a n1 e n2 e não a soma entre os 2.

Por isso nós devemos transformar n1 e n2 (strings) em variáveis do tipo number.

**CONVERSÕES DE STRINGS PARA NUMBER:**

Para converter para número inteiro 🡪Number.**parseInt(**n**)**

Para converter para número fracionado 🡪 Number.**parseFloat(**n**)**

Para converter para número e o próprio JS escolha 🡪 **Number(n)**

**Então, PS: NOTE AS 2 ABERTURAS DE PARÊNTESES.**

var n1 = **Number.parseFloat**(window.prompt ('Digite um número: '))

var n2 = **Number.parseFloat**(window.prompt ('Digite outro número: '))

**CONVERSÕES DE NUMBER PARA STRING:**

**String(**n**)** OU n**.toString()**

**TEMPLATE STRING:**

**Var** **s** = ‘JavaScript’

‘Eu estou aprendendo s’ 🡪 Não faz interpolação

‘Eu estou aprendendo ‘ **+** **s** 🡪 Usa concatenação

`Eu estou aprendendo **${s}**` 🡪 Template String. 🡪 Tem que ser **crase**

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

**FORMATANDO STRING:** 🡪 Ex004

**Var** s = ‘JavaScript’

s**.length** 🡪 Diz quantos caracteres a string tem

s**.toUpperCase()** 🡪 Joga tudo para ‘MAIÚSCULAS’

s**.toLowerCase()** 🡪 Joga tudo para ‘minúsculas’

Não esqueça de abrir e fechar os parênteses

var nome = window.prompt('Qual é o seu nome? ')

**document.write**(`Olá, ${nome}! seu nome tem ${nome.length} letras.<br>`)

**document.write**(`Seu nome em maiúsculas é ${nome.toUpperCase()} e em minúsculas é ${nome.toLowerCase()}`)

**document.write** 🡪 Escreve algo no corpo do html

**FORMATANDO NÚMEROS: 🡪 Ex004**

**Var** n1 = 1543.5

n1**.toFixed(**2**)** 🡪 Coloca 2 casas decimais após o ponto.

n1**.toFixed(**2**).replace(**‘.’, ‘,’**) 🡪** Troca o ponto por vírgula

n1**.toLocaleString(**‘pt-BR’, **{**style: ‘currency’, currency: ‘BRL’**})** 🡪 Transforma uma variável de número em formato monetário ‘R$ n1’ . Para dólar troque o BRL por USD

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

**AULA 7: Operadores, parte 1**

**Tipo de operadores:**

**Aritméticos 🡪 + - \* / % \*\* (potência)**

5 \*\* 2 é o mesmo que 5² = 25

O % é o resto da divisão inteira, usando 5 % 2 o resultado é 1, pois 5|\_2

-4|2

**Resto da divisão =** 1

**Atribuição**

**Relacionais**

**Lógicos**

**Ternário**

**Serão ensinados na próxima aula**

**ORDEM DE PRECEDÊNCIA ENTRE OS OPERADORES ARITMÉTICOS – PRIORIDADES**

1º - ()

2º - \*\*

3º - Multiplicação, divisão e o resto da div % (prioridade da esquerda para direita)

4º - Adição e subtração

Exemplificando:

var a = 5 + 3 (O valor 8 ficará guardado em a)

var b = a % 5 (O valor 3 ficará guardado em b)

var c = 5 \* b \*\* 2 (O valor 45 ficará guardado em c)

var d = 10 – a / 2 (O valor 6 ficará guardado em d)

var e = 6 \* 2 / d (O valor 2 ficará guardado em e)

var f = b % e + 4 / e (O valor 3 ficará guardado em e)

**Auto-atribuições:**

Var n = 3

N = n + 4 🡪 Agora n armazena o 7 (houve a troca de atribuição), isso pode ser substituído por:

**N** = **n** + 4 🡪 N **+=** 4. Tem que ser apenas 1 variável, veja os exemplos:

N = n % 5 🡪 n **%=** 5 N = n \*\* 2 🡪 n **\*\*=** 2 N = n / 2 🡪 n **/=** 2 N = n \* 4 🡪 n **\*=** 4

**Incremento**

Var x = 5

X = x+1 ou x+=1 pode ser substituído por 🡪 **x++**,agora x vale 6

X = x-1 ou x-=1 pode ser substituído por 🡪 **x--**,agora x vale 4 (considerando x = 5)

O + e o – também podem vir antes do nome da variável.

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

**AULA 8: Operadores, parte 2**

**Operadores relacionais: > < >= <= == !=** (diferente)

O valor da operação relacional será um valor booleano (true ou false)

5 > 2 🡪 True

8 >= 8 🡪 True

4 != 4 🡪 False

5 == 5 🡪 True

Operador relacional de identidade: === testa a identidade (mesmo valor e mesmo tipo)

5 == 5 🡪 True

5 == ‘5’ (string) 🡪 **True**

5 **===** ‘5’ (string) 🡪 **False**, tem o mesmo valor, mas não o mesmo tipo

5 **!==** ‘5’ (string) 🡪 **True**

**===** testa a identidade (mesmo valor e mesmo tipo da variável)

**Operadores Lógicos:**

**! (negação)** 🡪 Depois dele vem um valor booleano de True ou False

! true 🡪 False

! false 🡪 True

**&&(conjunção, “e”)** 🡪 Antes e depois dele vem um valor booleano de True ou False

True **&&** True 🡪 **True**

True **&&** False 🡪 **False**

False **&&** True 🡪 **False**

False **&&** False 🡪 **True**

**|| (disjunção, “ou”)** 🡪 Antes e depois dele vem um valor booleano de True ou False

**Basta vir um resultado True que dará True || Os 2 valores devem ser False para dar False**

True **||** True 🡪 **True**

True **||** False 🡪 **True**

False **||** True 🡪 **True**

False **||** False 🡪 **False**

Ex:

Var a = 5

Var b = 8

A > b && b % 2 == 0

Fase && True 🡪 False

**ORDEM DE PRIORIDADE:**

**1º Aritméticos**

**1.1 - ( )**

**1.2 - \*\***

**1.3 - Multiplicação, divisão e o resto da div % (prioridade da esquerda para direita)**

**1.4 - Adição e subtração**

**2º Relacionais**

**3º Lógicos**

**3.1 – Não (!)**

**3.2 - &&**

**3.3 - ||**

**4º Ternário**

Exemplos:

Idade **>=** **15 &&** idade **<=** **17** 🡪 A idade está entre 15 e 17?

Estado **==** **’RJ’ ||** estado **==** **‘SP’** 🡪 O estado é RJ ou SP?

Salário **>** **1500 &&** sexo **!=** **‘M’** 🡪 O salário é acima de 1.500 e não é homem?

**Operadores Ternário: 3 partes**

**?** e **:** dentro de uma mesma expressão

Teste Lógico **?** true (oq vai acontecer se for vdd) **:** false (oq vai acontecer se for mentira)

Ex:

Média >= 7 **?** “Aprovado” **:** “Reprovado”

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

**MÓDULO 3 – Entendendo o DOM**

**AULA 9: Introdução ao DOM (Document Object Model)**

Ex005 e Ex006

**DOM:** Conjunto de objetos dentro do seu navegador que vai dar acesso aos componentes internos do website, ele não funciona dentro do NodeJS, ele está presente quando estou rodando JavaScript dentro do navegador.

**ÁRVORE DOM (começa da raiz)**

window (tudo de JS está aqui dentro)

Os 3 componentes na frente do window (existem bem mais) estão abaixo dele

Window {location 🡪 Qual é a localização do site (url, pagina anterior, etc...)

document 🡪 Documento atual

history 🡪 Vai guardando de onde você veio, para onde você vai

document 🡪 html (parent: pai / mãe) 🡪 head (child) 🡪 meta, title, etc..

html (parent: pai / mãe) 🡪 body (child) 🡪 h1, p, p, div, etc...

No ex005: O segundo <p> (parent: pai / mãe) 🡪 Strong

Quem tá em baixo é child e quem tá acima é parent.

**No comando você começa com quem tá na raiz e vai até o componente que você quer, colocando ponto . a cada transição de parent para child.**

**SELECIONANDO OS ELEMENTOS DA ÁRVORE:**

**Por Marca (tag) 🡪** **getElementsByTagName(‘tag’)** **[ ]**– Pode ser mais de uma tag

No exercício ele queria pegar a primeira <p> então no final faça: [0], se fosse o segundo seria [1].

O comando abaixo: window.document**.write**(p1**.innerHTML**) 🡪 Copiou o 1º <p>

O [ ] será usado quando o nome do componente é no PLURAL

**Por ID 🡪 getElementById(‘nome do id’)**

**Por Nome (name) 🡪 getElementsByName(‘nome que você deu’) [ ]**

**Por Classe 🡪 getElementsByClassName(‘nome da classe que você deu’) [ ]**

**Por Seletor (CSS**) **🡪 querySelector() e querySelectorAll()**

**querySelector() 🡪** No exemplo 🡪 var d = window.document.querySelector(**'div#msg'**)

**querySelectorAll()**

**inner.Text 🡪** Pega apenas o texto

**inner.HTML 🡪** Pega o texto e toda as suas tags filhas.

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

**AULA 10: Eventos DOM**

Ex006

Evento é tudo aquilo que possa acontecer com aquele DOM que você selecionou.

Tipos de Eventos:

mouseenter 🡪 O mouse entrou dentro do elemento

mouseout 🡪 O mouse saiu de dentro do elemento

mouseup 🡪 Final do movimento de clique no elemento

click 🡪 Movimento do clique inteiro sobre o elemento

Lista: <https://developer.mozilla.org/pt-BR/docs/Web/Events#:~:text=Eventos%20Dom%20(Dom%20Events)%20s%C3%A3o,adicionais%20sobre%20o%20que%20aconteceu>.

Funções: Funções que serão executadas quando o evento ocorrer

No exemplo ele fez na <div id=”area” **onclick=”clicar()”**

**function** nome da ação(aqui podem ser inseridos alguns parâmetros) **{**

Bloco

**}**

O evento pode ser configurado na parte HTML ou diretamente no script

**A definição das variáveis devem vir fora dos comandos function.**

**Adicionando eventos pelo JS:**

Atribua uma variável ao DOM que você deseja manipular, no caso a variável a

a**.addEventListener**(‘nome técnico do evento’, nome que você quer) 🡪 Veja o VSC

**function** nome dado após a virgula() {

a.variações = ‘ ’

}

**Caso tenha algum erro na digitação do código 🡪 Abra o Dev Tools que ele irá lhe informar a linha onde o erro está após o :26**

O html:26 🡪 Significa que o erro está na linha 26 OU acima.

**Ex006 - Calculadora**

Ele atribuiu 1 variável para cada input da calculadora (tn1 e tn2)

Ele precisou **transformar as 2 variáveis do input em número** (lembre-se que tudo que entra como caixa de texto fica como STRING) e atribuir a uma nova variável:

Var n1 = Number(tn1.value)

Var n2 = Number(tn2.value)

Agora atribua a soma a uma nova variável s e mostrou ela em uma <div id=”res”>

Var s = n1 + n2

Res.innerHTML = ` A soma entre ${n1} e ${n2} é igual a <strong> **${s}** </strong>`

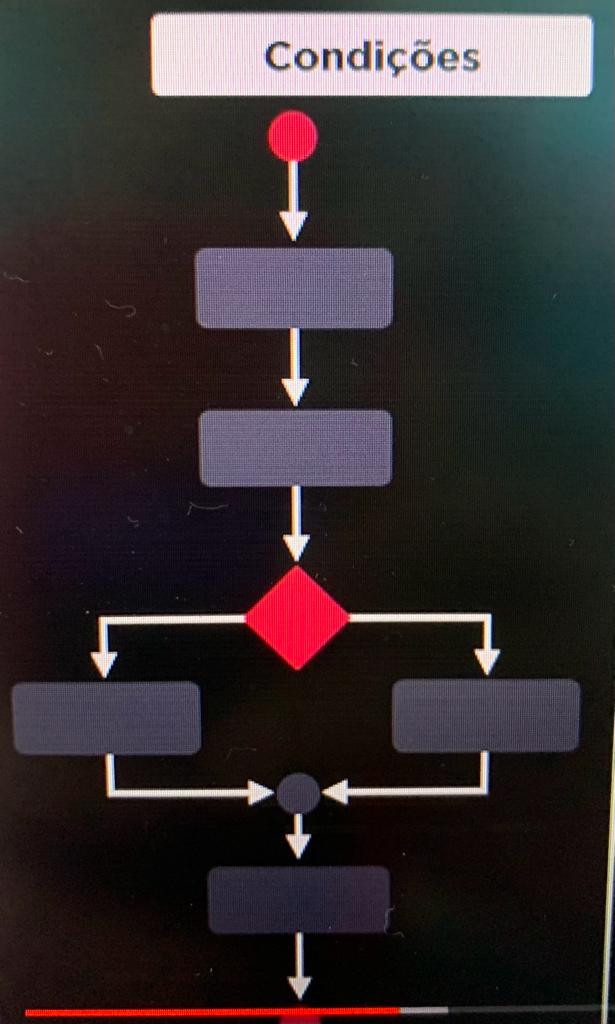
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

**MÓDULO 4 – CONDIÇÕES EM JAVASCRIPT**

**AULA 11: Condições – parte 1**

false

true

Ex008.js, ex009.js e ex010.html

If = Se (condição da esquerda)

Else = Senão (condição da direita)

Losangulo: **if (**condição**)** **{**

Bloco caso if seja true

**} else {**

Bloco caso if seja false

**}**

**Tipos de condição:**

Condição simples é quando só existe o bloco do if true 🡪 ex009

Condição composta é quando tem if e else 🡪 ex010

Para escrever na tela com arquivos.JS use:

**Console.log(‘**escreva a mensagem aqui**’)**

No ex010.html ele usou **+=** no res.innerHTML pq se fosse só = iria apagar a mensagem anterior devido a nova atribuição.

Para abrir a extensão NodeExec, vá no arquivo**.JS** e aperte **F8**

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

**AULA 12: Condições – parte 2**

Ex011.js , ex012.js e ex013.js

**ANINHANDO CONDIÇÕES:** É quando existe uma condição dentro de outra condição

**USE PARA TESTAR MAIS DE 1 CONDIÇÃO**

**if (condição 1)** **{**

bloco 1

**} else if (condição 2) {**

bloco 2

**} else {**

bloco 3

**}**

No exemplo:

var idade = 25

console.log(`Você tem ${idade} anos.`)

**if** (idade < 16) **{**

console.log('Não vota')

**}** **else if** (idade < 18 || idade > 65) **{**

console.log('Voto opcional')

**}** **else** **{**

console.log('Voto obrigatório')

**}**

Ex012

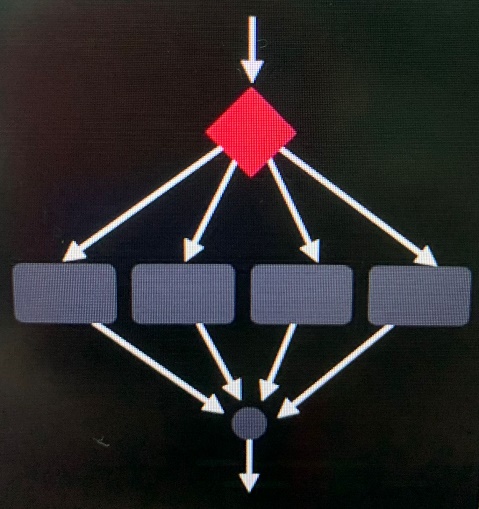
Para pegar a hora atual do sistema:

Var agora = **new Date()**

Var hora = **agora.getHours()**

Como é o no NodeJS, ele vai pegar a hora do servidor.

**CONDIÇÕES MÚLTIPLAS**

Serve para valores fixos e só funciona com **números inteiros**

Comando:

**Switch (**expressão**) {**

**case** valor 1**:**

bloco

**break**

**case** valor 2**:**

bloco

**break**

**case** valor 3**:**

bloco

**break**

**default:**

bloco

**break**

**}**

Default é caso nenhum dos cases seja compatível

Ex013.js 🡪 Muito bom.

Var agora = **new Date()**

Var diaSem = **agora.getDay()**

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

**AULA 12: Exercícios JavaScript (Parte 1 e 2)**

Pasta: Aula12ex 🡪 ex014 🡪 modelo.html

Ele cria o arquivo css e <script: src”script.js”></script> 🡪 Assim todo script fica em um arquivo separado.

<body onload="carregar()">

Para mudar a foto dinamicamente**: img.src = '**manha.jpg**'**

Para mudar a cor do bg dinamicamente: **document.body.style.background = '**#b9846f**'**

**AULA 12: Exercícios JavaScript (Parte 3)**

Pasta: Aula12ex 🡪 ex015 🡪 modelo.html

Os 2 inputs do tipo radio devem ter o mesmo name=’’” (para você poder desmarcar) e cada um teve sua label.

var data = new Date()

var ano = data**.getFullYear()** 🡪 pega os 4 dígitos do ano.

Criando tags dinamicamente e adicionando atributos a elas:

**var img = document.createElement('img') 🡪** Criou a tag <img>

**img.setAttribute('id', 'foto') 🡪** Deu o ID de ‘foto’ para a tag

No final:

res.style.textAlign = 'center'

res.innerHTML = `Detectamos ${gênero} com idade ${idade} anos`

**res.appendChild(img) 🡪** Para aparecer a <img> criada dinamicamente.

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

**MÓDULO 5 – REPETIÇÕES EM JAVASCRIPT**

**AULA 13: Repetições – parte 1**

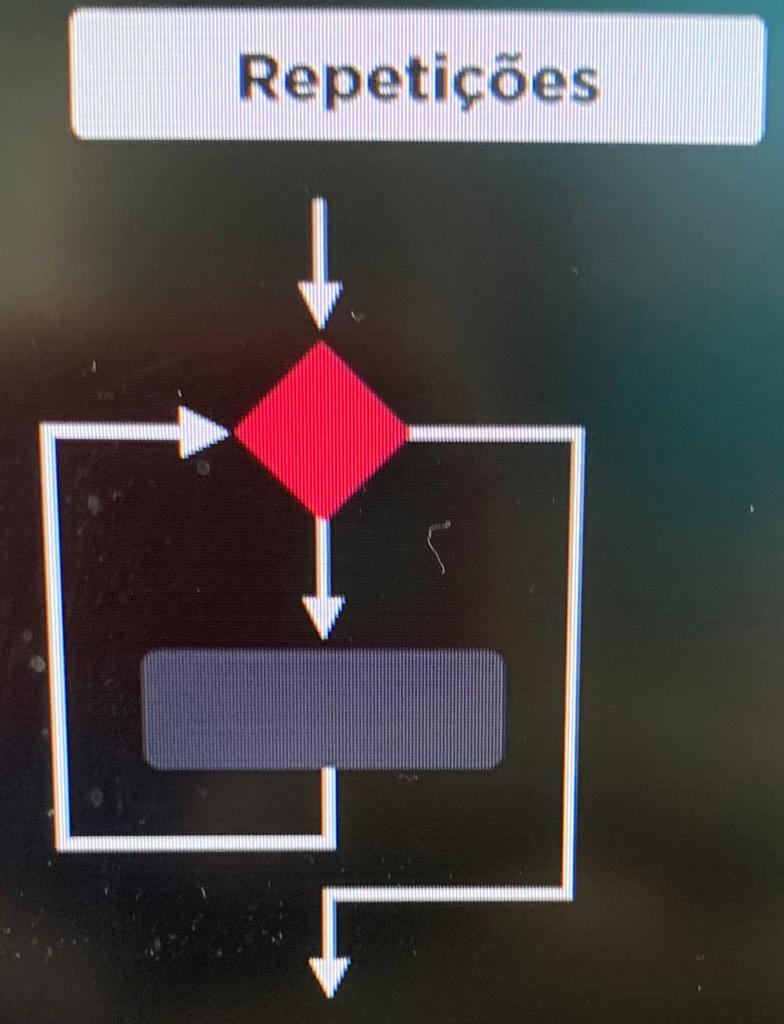
Pasta: aula13 🡪 ambiente.js

Sempre que a condição for verdadeira, o bloco é executado e volta ao losango (enquanto ele for positivo), se for negativo o laço é quebrado e segue o fluxo.

While (condição) {

1º testa a condição

2º executa o bloco



Codando:

**While (**condição**) {**

**comando**

**}**

Ex:

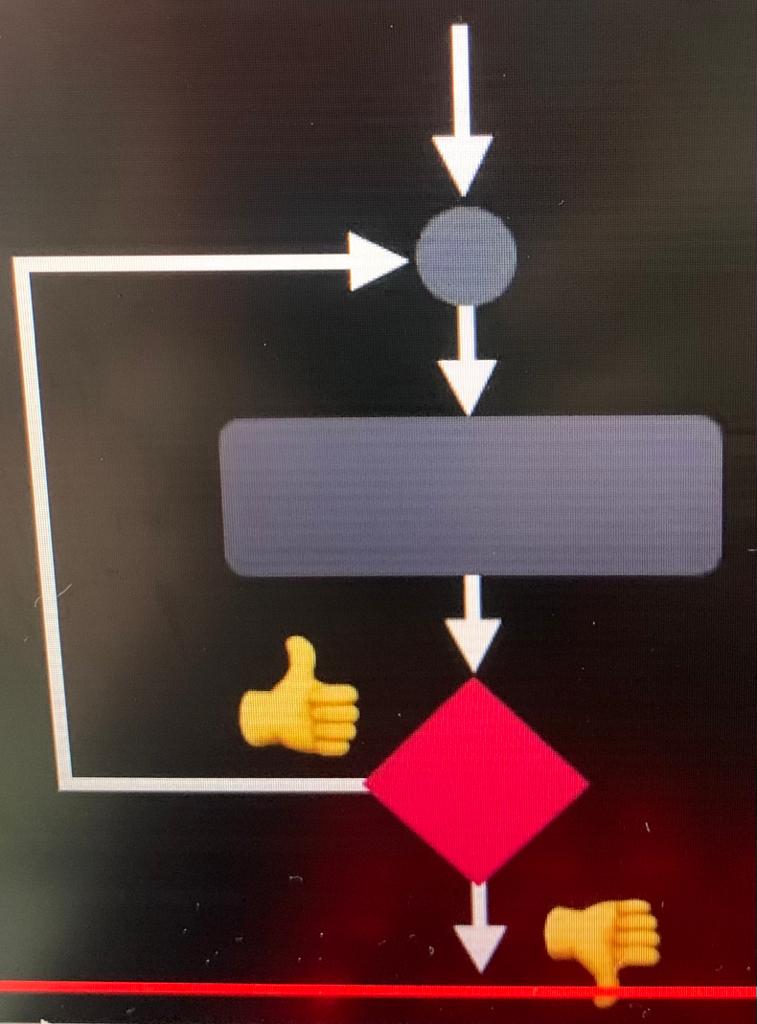
**Function** comerPizza() **{**

**While(**temFatia()**)** **{**

comerFatia()

**}**

**}**

****

**Do** **{**

comando

**}** **while** **(**condição**)**

1º Executa o bloco

2º Testa a condição

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

**AULA 14: Repetições – parte 2**

Pasta: aula14 🡪.js

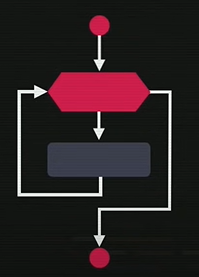
Estrutura de repetição com variável de controle

Passos:

Inicialização

Teste Lógico

Incremento (caso teste lógico for TRUE)



Codando:

**For (**inicio **;** teste **;** incremento**)** **{**

comando

**}**

Pegando o exemplo da aula anterior:

Var c = 1

White (c <= 5) {

Bloco

C++

}

Com o for seria:

Na primeira expressão, você deve definir uma variável

For (var c = 1; c <= 5; c++) {

console.log(c)

}

**EXERCÍCIOS DE REPETIÇÃO**

Aula14ex 🡪 ex016.html e ex017.html

**Para adicionar emojis no innerHTML faça = `\u{código do emoji depois do +}`**

Ao criar um elemento com var x = document.createElement(‘nome da tag’), você precisa inseri-la em algum local com id e q seja uma variável (Y, por ex), ai faça:

y.appendChild(‘x’)

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

**MÓDULO 6 – AVANÇANDO NOS ESTUDOS**

**AULA 15: Variáveis Compostas (Arrays)**

Pasta: aula15 🡪 ambiente.js

Variáveis simples: Só conseguem armazenar um valor por vez.

**Variáveis compostas: São capazes de armazenar vários elementos (conteúdo/valor + índice/chave) em uma mesma estrutura.**

Representação:

**Var a = [**valor 1, valor 2 , valor 3 , etc... **]**

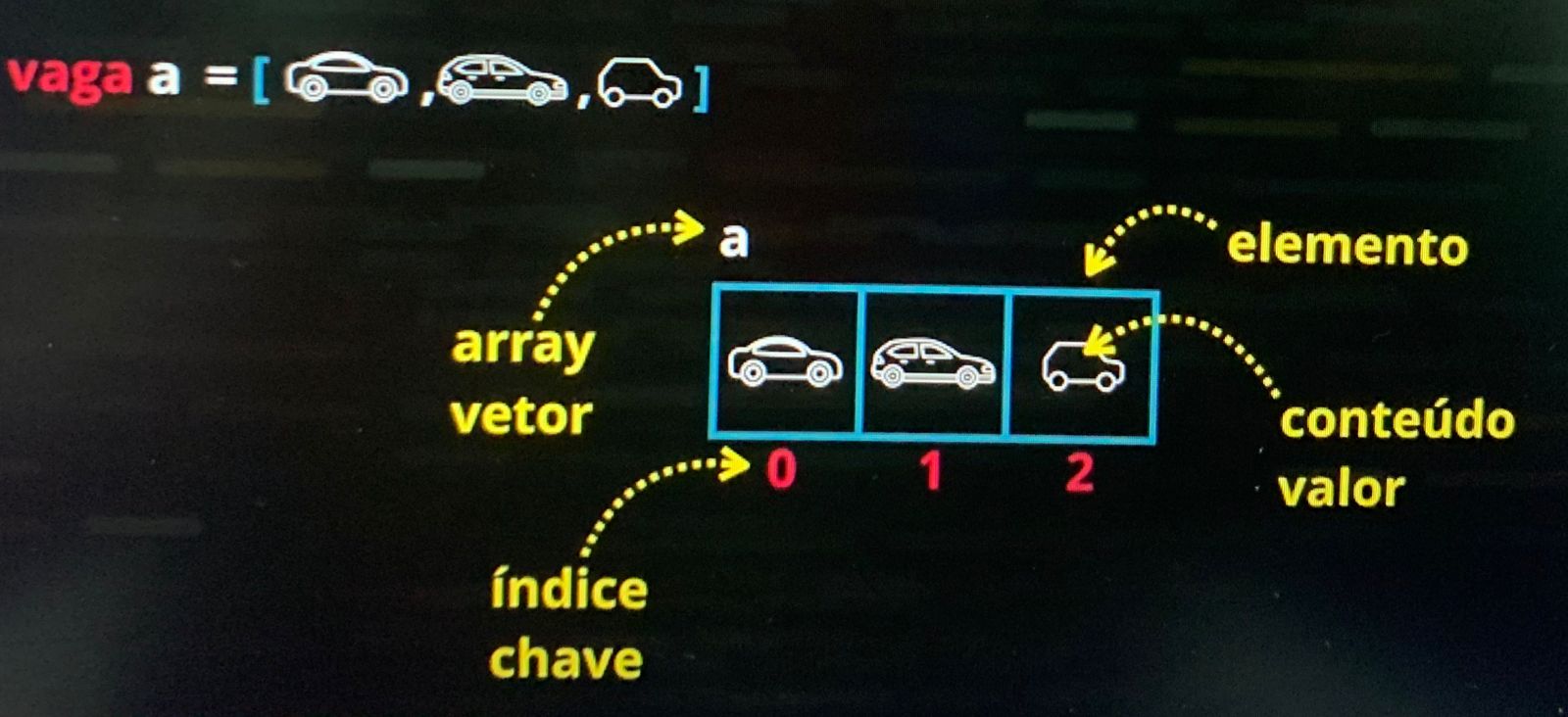
Var a = variável composta ou **Array (vetor)**

Cada valor recebe um índice q vai de 0 a 10

Na programação sempre começa com 0 invés do 1

Valor 1 🡪 a0

Valor 2 🡪 a1

Valor 3 🡪 a2

Para adicionar um valor a um novo elemento do array:

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

**5** é **0**; **8** é **1**; **4** é **2**

Adicionando uma nova vaga na posição 3:

Num**[3]** = **8**

Outra forma de adicionar considerando que o passo anterior foi realizado:

Num**.push(7)** 🡪 A posição **4** do array recebe o valor 7

Para saber o tamanho do array use: (num é o nome da variável do exemplo)

Num**.length** 🡪 Não há parêntese após 🡪 Vai retornar o **valor 5** (0, 1, 2, 3, 4)

Num**.sort()** 🡪 Colocará todos os valores em **ordem crescente** nos índices

**MOSTRANDO OS NÚMEROS DO VETOR COM UMA ESTRUTURA DE REPETIÇÃO**

Var num (vetor) = [4, 5, 6, 7, 8] 0, 1, 2, 3, 4

A variável pos definirá a posição

**For** (var **pos** = 0; **pos <** num**.length**; **pos**++) {

Console.log(num.**[pos]**)

}

Novo arquivo para o teste acima 🡪 vetornatela.js

Outra maneira, e mais simples, de fazer o código acima é:

**For (var** **pos** **in** num**)** **{** 🡪 Lê-se: Para cada posição na variável num (array)

**Console.log(**num**[pos])**

**}**

**Buscando valores dentro de um array:**

Var num (vetor) = [4, 5, 6, **7**, 8] 0, 1, 2, **3**, 4

Num**.indexOf(7)** 🡪 Vai procurar onde está o valor 7 e retornará o valor 3, se retornar o valor -1 significa que não tem.

Essa parte acima está no ambiente.js