Seção 3: JavaScript - Lógica de programação

Extensões vs code:

Code runner = executa código no terminal

Material icon theme = coloca ícone em cada arquivo

Alt + shift + certinha = copia e já cola o código no vsc

* Aula 21 operadores de comparação

/\*

Operadores de comparação

> maior que

>= maior que ou igua a

< menor que

<= menor que ou igual a

== igualdade (valor) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

=== igualdade estrita (valor e tipo)

!= diferente (valor) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

!== diferente estrito (valor e tipo)

\*/

* Aula 22 operadores lógicos

/\*

&& -> AND -> E

|| -> OR  -> OU

!  -> NOT -> NÃO

\*/

const expressaoAnd = true && false //Só será verdadeiro se todas

forem verdadeira

const expressaoOr  = true || false //Aumenos um verdadeiro para ser  verdadeiro

const expressaoNot = !true         //valor contrário

* Aula 23 avaliação de curto-circuito(short-circuit)

/\*

&& -> false && true -> false "o valor mesmo"

|| -> true || false -> vai retornar o valor verdadeiro

FALSY

false

0

'' "" ``

null / underfined

NaN

\*/

console.log('Rodrigo' && true && 'Maria'); // vai retornar o ultimo valor no caso 'Maria'

console.log('Luiz' && undefined && 'Maria'); //ao encontrar um valor sendo como falso , a operação irá parar

function falaOi(){

    return 'Oi'

}

const vaiExecutar = false

//true         //false

vaiExecutar && falaOi()

console.log(vaiExecutar && falaOi()); // vai retornar false

console.log(0 || false || null || 'Luiz Otávio' || true); // vai retornar o primeiro valor verdadeiro que ele encontrou

* Aula 24 estrutura condicionais (if, else if e else)

/\*

Entre  0 - 11 = Bom dia

Entre 12 - 17 = Boa tarde

Entre 18 - 23 = Boa noite

\*/

const hora = 19

if (hora >= 0 && hora < 12) {

    console.log('Bom dia!');

} else if (hora >= 12 && hora <= 17) {

    console.log('Boa tarde!');

} else if (hora >= 18 && hora <= 23){

    console.log('Boa noite!');

} else {

    console.log('Hora invalida!!!');

}

* Aula 26 exercício

    const p = document.createElement('p');

    p.innerHTML = 'Qualquer coisa';

resultado.appendChild(p)

criando um elemento p de parágrafo, mais não chama ele no html

depois está inserindo no parágrafo um frase, e com appendchild esta criando um filho p para colocar em resultado

p.classList.add('paragrafo-resultado')

para adicionar uma classe ao parágrafo que foi criado

* Aula 27 operações ternarias

const pontuacaoUsuario = 999;

if (pontuacaoUsuario >= 1000) {

    console.log('Usuário VIP');

} else {

    console.log('Usuário normal');

}

Passa a ser:

const nivelUsuario = pontuacaoUsuario >= 1000 ? 'Usuario VIP' : 'Usu

ário normal';

//(condicao) ? 'valor para verdadeiro' : 'valor para falso'

* Aula 28 objeto date

const date = new Date();

sempre que chama a palavra new está chamando uma função construtora.

Funções construtoras sempre vai começar com a primeira letra maiúscula

// 01/01/1970 - Timestamp unix ou época unix - marco 0

//                 s   m    h  \*  milisegundos

const tresHoras = 60 \* 60 \* 3  \*  1000;

const umDia = 60 \* 60 \* 24 \* 1000

const date = new Date(0 + tresHoras); // marco 0, poorque temos um

recuo de -3 horas

console.log(date.toString());

pode passar assim:

//                    ano  mês dia  h  m   s   milessimosegundos

const date = new Date(2019, 3, 20, 15, 14, 27, 500);

console.log(date.toString());

mês = jan, fev, mar, abri, mai, ...

[0] [1] [2] [3] [4] [...]

Milésimos segundos vai até 999, quando chega no 1000 ele zera e corrige o segundos

const data = new Date('2019-12-20 20:19:30');

console.log(data.toString());

outra forma de passar uma data

const data = new Date('2019-12-20 20:19:30');

console.log('Dia',data.getDate());         //Dia

console.log('Mês',data.getMonth() + 1);    //Mês começa do 0

console.log('Ano',data.getFullYear());     //Ano

console.log('Hora',data.getHours());       //Hora

console.log('Min',data.getMinutes());      //Minutos

console.log('Seg',data.getSeconds());      //segundos

console.log('ms',data.getMilliseconds());  //Milisegundos

console.log('Dia da semana',data.getDay());//Dia da semana

console.log(data.toString());

console.log(Date.now());

obtendo do marco zero até a data que foi chamado em milésimos de segundos

function zeroAEsquerda(num) {

    return num >= 10 ? num : `0${num}`

}

function formatData(data) {

    const dia = zeroAEsquerda(data.getDate());

    const mes = zeroAEsquerda(data.getMonth() + 1);

    const ano =  zeroAEsquerda(data.getFullYear());

    const hora =  zeroAEsquerda(data.getHours());

    const min =  zeroAEsquerda(data.getMinutes());

    const seg =  zeroAEsquerda(data.getSeconds());

    return `${dia}/${mes}/${ano} ${hora}:${min}:${seg}`

}

const data = new Date();

const dataBrasil = formatData(data)

console.log(dataBrasil);

* Aula 29 switch/case

const data = new Date('1987-04-30 00:00:00');

const diaSemana = data.getDay();

let diaSemanaTexto;

if (diaSemana === 0 ) {

    diaSemanaTexto = 'domingo';

} else if (diaSemana === 1 ) {

    diaSemanaTexto = 'Segunda';

} else if (diaSemana === 2 ) {

    diaSemanaTexto = 'terça';

} else if (diaSemana === 3 ) {

    diaSemanaTexto = 'quarta';

} else if (diaSemana === 4 ) {

    diaSemanaTexto = 'quinta';

} else if (diaSemana === 5 ) {

    diaSemanaTexto = 'sexta';

} else if(diaSemana === 6 ) {

    diaSemanaTexto = 'sabado';

} else {

    diaSemanaTexto = 'Dia da semana INVÁLIDO'

}

console.log(diaSemanaTexto);

passaria a ser:

switch (diaSemana) {

    case 0:

        diaSemanaTexto = 'Domingo';

        break;

    case 1:

        diaSemanaTexto = 'Segunda';

        break;

    case 2:

        diaSemanaTexto = 'Terça';

        break;

    case 3:

        diaSemanaTexto = 'Quarta';

        break;

    case 4:

        diaSemanaTexto = 'Quinta';

        break;

    case 5:

        diaSemanaTexto = 'Sexta';

        break;

    case 6:

        diaSemanaTexto = 'Sábado';

        break;

    default:

        diaSemanaTexto = 'Valor Inválido';

}

Lembrando que em casa case e necessário o uso do break para cancelar a ação, se não o mesmo só irá parar quando chegar no default ou quando achar um break ou tenha uma palavra return quando dentro de uma função.

    case 3:

        diaSemanaTexto = 'Quarta';

        return diaSemanaTexto;

const data = new Date();

const h1 = document.querySelector('.texto');

h1.innerHTML = data.toLocaleString('pt-BR', {

    dateStyle:'full', timeStyle:'short'

});

Modelo rápido de pegar a data

* Aula 31 diferença var e let

let nome = 'Luiz';

var nome2 = 'Luiz';

let nome2 = 'Otávio'

//ERRO

var nome2 = 'Otávio'

// e possivel sobrepor uma declaração

const verdadeira = true;

// Let tem escopo de bloco { ...bloco}

// Var só tem escopo de função

let nome = 'Luiz';

var nome2 = 'Luiz';

if (verdadeira) {

    let nome = 'Otávio';

    console.log(nome, nome2);

}

Dentro do bloco o nome e Otavio e fora o nome e Luiz

let nome = 'Luiz';

var nome2 = 'Luiz';

if (verdadeira) {

    let nome = 'Otávio';

    //console.log(nome);

    if (verdadeira) {

        //let nome = 'Outra coisa'

        console.log(nome);

    }

}

Aqui como não achou ele vai procurar mais acima ate achar a variável.

let nome = 'Luiz';//criada

var nome2 = 'Luiz';

if (verdadeira) {

    let nome = 'Otávio'; //criada

    var nome2 = 'Rogerio' //redeclarando

    if (verdadeira) {

        let nome = 'Outra coisa';

        var nome2 = 'Ronaldo' //redeclarada

    }

}

Cada variável que contenha var ela e redeclarada, nesse caso nome2 começou como Luiz depois passou a ser Rogerio e finalizou sendo Ronaldo. As variáveis com são chamadas com let só são alteradas dentro de seus blocos

function falaOi () {

    var nome = 'Luiz'

}

falaOi();

console.log(nome); //ERRO pois as funções protegem as variáveis

porem o contrário e possível acontecer, a função consegue acessar o entorno porem o entorno não consegue acessar as variáveis da função

var nome = 'Luiz'

function falaOi () {

    console.log(nome);

}

falaOi();

console.log(nome);  //Undefined

console.log(sobrenome); //Gera ERRO

let sobrenome = 'Miranda' // Não acontece a elevação

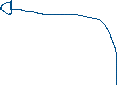
var nome = 'João' // Faz a ELEVAÇÃO o hosting

* Aula 32 atribuição via desestruturação arrays

let a = 'A';

let b = 'B';

let c = 'C';



const numeros = [b, c, a];



[a, b, c] = numeros;

console.log(a, b, c); //B C A

esta desestruturando e passando novos valores de acordo com suas ordem



//               0   1  2   3   4   5   6   7   8

const numeros = [100, 200, 300, 400, 500, 600, 700, 800, 900];

const [primeiroNumero, segundoNumero] = numeros;

console.log(primeiroNumero, segundoNumeros); //1 2

nesse modelo ele está criando duas variáveis const e seus valores serão pegado da Array de numeros de acordo com a ordem que foram colocados.

const numeros = [100, 200, 300, 400, 500, 600, 700, 800, 900];

const [primeiroNumero, segundoNumeros, ...resto] = numeros;

console.log(primeiroNumero, segundoNumeros, resto);

agora eu estou pegando o resto que não foram realocados e inserindo na const resto.

Pri seg resto

100 200 [300, 400, 500,600, 700, 800,900]

const numeros = [100, 200, 300, 400, 500, 600, 700, 800, 900];

const [um, , tres, , cinco, , sete] = numeros;

console.log(um, tres, cinco); //100 300 500

você pode no caso pular valores utilizando espaços vazios

//               0          1          2

//            0  1  2    0  1  2    0  1  2

const num = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];

const [, [ , , seis], ] = num;

console.log(seis); // 6

criando uma const seis, e pegando o valor que se encontra na mesma ordem da const num.

* Aula 33 atribuição via desestruturação (Objetos)

const pessoa = {

    nome: 'Luiz',

    sobrenome: 'Miranda',

    idade: 29,

    endereco: {

        rua: 'Av Brasil',

        numero: 320

    }

}

console.log(pessoa);

criando um objeto

const nome = pessoa.nome

console.log(nome); //Luiz

atribuição normal

//Atribuição via desestruturação

const { nome } = pessoa

console.log(nome); //Luiz

pega o atributo que esta dentro de pessoa que tenha o nome de nome, e já cria uma const nome com esse valor.

const { sobrenome, time = 'Não tem'} = pessoa

console.log(sobrenome, time); //Miranda Não tem

caso o valor setado não tenha no objeto, pode se passar um valor como padrão, nesse caso o objeto pessoa não tem o atributo time, logo o seu padrão será Não tem

const { nome: teste = 'Rodrigo' } = pessoa  // alterando o nome do

atributo

console.log(teste);

aqui passamos o nome da chave nome para teste

const { endereco: { rua, numero } } = pessoa;

console.log(rua, numero);

aqui estamos pegando o objeto endereco que está dentro do objeto pessoa.

Nesse caso não temos mais o atributo endereco e sim as const rua e numero.

const { endereco: { rua: r, numero } } = pessoa; //atribuindo

 a variavel rua em r

console.log(r, numero);

mudando o nome da variável rua para r

const { nome, ...resto } = pessoa;

console.log(nome, resto);

pegando só a variável nome e colocando o que sobrou em resto.

* Aula 34 for clássico estrutura de repetição

for (let i = 0; i <= 5; i++) {

    console.log(`Linha ${i}`);

}

for (let i = 0; i <= 10; i++) {

    const par = i % 2 === 0;

    console.log(i, par);

}

Vendo se e par ou impar

for (let i = 0; i <= 10; i++) {

    const imp\_par = i % 2 === 0 ? 'Par' : 'Ímpar'

    console.log(i, imp\_par);

}

Com uso do operador ternário

//                0       1       2       4

const frutas = ['Maça', 'Pera', 'Uva', 'Banana']

for (let i = 0; i < frutas.length; i++) {

    console.log(`A fruta é ${frutas[i]}`);

}

//                0       1       2       4          5        6

const frutas = ['Maça', 'Pera', 'Uva', 'Banana', 'Melancia', 'Kiwi']

for (let i = 0; i < frutas.length; i++) {

    console.log(`Índice ${i}, ${frutas[i]}`);

}

* Aula 35 exercício

const elementos = [

    {tag: 'p', texto: 'Frase 1'},

    {tag: 'div', texto: 'Frase 2'},

    {tag: 'footer', texto: 'Frase 3'},

    {tag: 'section', texto: 'Frase 4'}

];

const container = document.querySelector('.container');

// pegando a classe de container

const div = document.createElement('div');

// criando uma tag div

for (let i = 0; i < elementos.length; i++) {

// percorrendo o array de elementos

    let { tag, texto } = elementos[i];

// incrementando por decrementarão uma variável

    const criaTag = document.createElement(tag);

// criando uma tag e jogando em uma variável

    criaTag.innerHTML = texto

// inserindo um valor na variável que foi criada

    div.appendChild(criaTag);

// inserindo a variael que foi crianda com filha de outra variavel

}

container.appendChild(div)

// inserindo como filha todo o conteúdo contido na div

const h1 = document.querySelector('h1');

h1.style.backgroundColor = 'blue';

manipulando o style através do Javascript.

* Aula 36 for in estrutura de repetição

//For in ->lê os índices ou  chaves do objeto

const frutas = ['Pera', 'Maçã', 'Uva'];

for (let indice in frutas) {

    console.log(frutas[indice]);

}

const pessoa = {

    nome: 'Luiz',

    sobrenome: 'Otávio',

    idade: 30

};

for (let chaves in pessoa) {

    console.log(chaves);

}

Com objetos

console.log(pessoa.nome);    //Luiz

console.log(pessoa['nome']); //Luiz

como acessar uma propriedade de um objeto.

const chave = 'nome';

console.log(pessoa[chave]); //Luiz

outro método através de uma variável retornando o mesmo valor

* Aula 37 for of estrutura de repetição

//            012345678910

const nome = 'Luiz Otávio';

// for (let i= 0; i < nome.length; i++) {

//     console.log(nome[i]);

// }

for ( let i in nome) {

    console.log(i);

}

for (let valor of nome) {

    console.log(valor);

}

O uso do for in retorna o indice dos valores ex: 1,2,3,4,5,6,7,8,9,10

Com o uso do for of, invés dele retornar o indice ele irá retorna o valor.

Ex: l,u,i,z, ,o,t,a,v,i,o

const nomes = ['Luiz', 'Otávio', 'Henrique'];

nomes.forEach( function(valor, indice, array) {

    console.log(valor, indice);

})

Outro modelo de retornar valores e índices com o uso do forEach.

Com ele nos podemos pegar o valor o indice e o array completo

//For clássico - Geralmente com iteráveis ( array ou strings)

//For in - Retorna o índice ou chave (strings, arrys ou Objetos)

//For of - Retorna o valor em si (iteráveis, array ou string)

Para objetos use somente o for in.

* Aula 38 exercícios

const paragrafos = document.querySelector('.paragrafos');

paragrafos.querySelectorAll('p')

console.log(paragrafos.querySelectorAll('p'));

aqui estamos retornando uma NodeList,(se comporta como um array porém não e um).

estamos selecionando a classe .paragrafos em seguida selecionamos todos os p(paragrafos) desse primeira classe.

const estiloBody = getComputedStyle(document.body);

const backgroundBody = estiloBody.backgroundColor;

getComputedStyle(de onde quer pegar) -> pega o estilo css do elemento desejado, depois estamos pegando somente o background do elemento desejado e jogando dentro de uma variável.

* Aula 39 while e do while estrutura de repetição

let i = 0;

while (i <= 10) {

    console.log(i);

    i++;

}

function random(min, max) {

    const r = Math.random() \* (max - min) + min;

    return Math.floor(r);

}

const min = 1;

const max = 50;

let rand = random(min, max);

while ( rand !== 10) {

    rand = random(min, max);

    console.log(rand);

}

Nesse código o programa vai ficar executando até o numero for igual a 10

do {

    rand = random(min, max);

    console.log(rand);

} while(rand !== 10);

Como o uso do, do while, primeiro ele executa o código pra depois ele consultar se a condição e verdadeira

* Aula 40 break e continue

const numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9];

for (let numero of numeros) {

    if (numero === 2) {

        continue;

    }

    console.log(numero);

}

Nessa código quando a condição encontra o código continue, ele para de ler e vai para a próxima interação.

const numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9];

for (let numero of numeros) {

    if (numero === 7) {

        break;

    }

    console.log(numero);

}

console.log('Saindo');

no caso do break, assim que ele e encontrado ele para o laço de repetição.

Lembre de no caso de ter um incremento de controle em seu código, para que o código não quebre e importante fazer o incremento ou decremento de seu controle antes do continue, se não haverá um loop infinito

if (c === 2) {

        console.log('Pulando o indice 2');

        c++;

        continue;

    }

Se a variável de controle estivesse após o continue, ela não seria executada fazendo assim um loop infinito