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

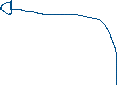
var nome = 'João' // Faz a ELEVAÇÃO o hosting

* 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

* Aula 44 tratamento de erros try, catch, throw

try{

    console.log(naoExisto);

} catch(err) {

    console.log('naoExisto não existe');

}

Try = tente fazer alguma coisa, caso de erro e não consiga, cairá em outro bloco no catch, esse bloco catch recebe o erro, pode se dado o nome que quiser.

Ex: catch(erro), catch(e) ou até catch(err)

O bloco do catch e oque será executado caso ocorra algum erro.

if(typeof x !== 'number' || typeof y !== 'number') {

        throw('x e y precisam ser números.');

    }

Nesse caso com o uso do throw estamos tratando o erro, mostrando como ele deve ser exibido, nesse caso ele mostrará oque foi pedido mais não mostrará onde ocorreu o erro.

function soma(x, y) {

    if(typeof x !== 'number' || typeof y !== 'number') {

        throw new Error('x e y precisam ser números.');

    }

Para resolver basta junto ao throw adicionar o new Error, que ajudará a mostrar onde está ocorrendo o erro

function soma(x, y) {

    if(typeof x !== 'number' || typeof y !== 'number') {

        throw new Error('x e y precisam ser números.');

    }

    return x + y;

}

try {

    console.log(soma(1, 2));

    console.log(soma('1', 2));

} catch(error) {

    console.log('Alguma coisa mais amigável pro usuário');

}

* Aula 45 tratamento e lançamento de erros try, catch finally

try {

    // É executada quando não há erros

} catch (e) {

    // É executada quando há erros

} finally {

    // Sempre é executada

}

function retornaHora(data) {

    if (!(data instanceof Date)) {

        console.log('Não é');

    }

}

retornaHora(new Date())

verificando se o valor que foi passado e uma instancia de Date

* Aula 46 setInterval e setTimeout

function mostraHora(){

    let data = new Date();

    return data.toLocaleTimeString('pt-br', {

        hour12: false

    })

}

function funcaoDoInterval() {

    console.log(mostraHora());

}

setInterval(funcaoDoInterval, 1000);

função setInterval será executado de 1 em 1 segundo(1000 milissegundos), a cada 1 segundo ele executará a função funcaoDoIntervalo.

function mostraHora(){

    let data = new Date();

    return data.toLocaleTimeString('pt-br', {

        hour12: false

    })

}

setInterval(function () {

    console.log(mostraHora());

},1000)

Ou faça a função anônima direto, o código irá ficar rodando até ser cancelado a execução, com ctrl + alt + m

Também e possível colocar toda essa função dentro de uma variável

function mostraHora(){

    let data = new Date();

    return data.toLocaleTimeString('pt-br', {

        hour12: false

    })

}

const timer = setInterval(function () {

    console.log(mostraHora());

},1000);

setTimeout (function () {

    clearInterval(timer)

}, 10000);

Com o setTimeout, e usado para parar a execução do setInterval, chame o clearInterval e passe o nome do intervalo criando. clearInterval(timer). Feito isso coloque qual o tempo que será feita a parada.

Esse setTimeout só será executada uma vez.

setTimeout (function() {

    console.log('Olá mundo!!!');

},5000)

Também podemos utilizar para mostrar uma mensagem de acordo com o passar do tempo, nesse caso será mostrada a mensagem quando se passar 5 segundos(5000 milissegundos)

* Aula 47 exercício

document.addEventListener('click', function(e) {

const el = e.target;

adicionando uma observador que em cada ação do clique irá retornar um evento, esse evento que trás o target, que serve para mostrar em que está sendo clicado.

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

let segundos = 0;

let timer;

function criaHoraDosSegundos(segundos) {

const data = new Date(segundos \* 1000);

return data.toLocaleTimeString('pt-BR', {

     hour12: false,

     timeZone: 'UTC'

});

}

function iniciaRelogio() {

timer = setInterval(function() {

    segundos++;

    relogio.innerHTML = criaHoraDosSegundos(segundos);

}, 1000);

}

document.addEventListener('click', function(e) {

const el = e.target;

if (el.classList.contains('zerar')) {

    clearInterval(timer);

    relogio.innerHTML = '00:00:00';

    relogio.classList.remove('pausado');

    segundos = 0;

}

if (el.classList.contains('iniciar')) {

    relogio.classList.remove('pausado');

    clearInterval(timer);

    iniciaRelogio();

}

if (el.classList.contains('pausar')) {

    clearInterval(timer);

    relogio.classList.add('pausado');

}

});

* Aula 48 exercício lista de tarefas

const inputTarefa = document.querySelector('.input-tarefa');

const btnTarefa = document.querySelector('.btn-tarefa');

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

function criaLi(){

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

    return li; //li desta função, diferente do outro

}

function criaTarefa(textoInput){

    const li = criaLi();  //li desta função, diferente o outro

    li.innerText = textoInput;

    tarefas.appendChild(li);

}

btnTarefa.addEventListener('click', function(){

    if (!inputTarefa.value) return; //se for vazio nada acontece

    criaTarefa(inputTarefa.value); //joga o texto pra outra função

});

Nesse código, há um input que se enviado vazio retorna nada, caso haja um valor, esse valor será enviado para a função criaTarefa, que assim que criada cria uma li(elemento de lista), e em seguida inseri o valor passado dentro dessa li criada, que em seguida, inseri essa li criada como sendo filha da const tarefas.

inputTarefa.addEventListener('keypress', function(e){

    console.log('Uma tecla foi pressionada');

});

Evento para ficar checando se uma tecla foi pressionada.

Keypress -> quando uma tecla foi pressionada

Keyup -> quando pressiona e solta a tecla

Keydown -> quando pressiona a tecla e ainda está pressionada

inputTarefa.addEventListener('keypress', function(e){

    console.log(e);

});

Nesse modelo será mostrada o evento que consta nas teclas.

key: "d" -> tecla pressionada

keyCode: 100 -> código chave da tecla pressionada

inputTarefa.addEventListener('keypress', function(e){

    if(e.keyCode === 13){

        console.log('tecla ENTER pressionada');

    }

});

Pegando se a tecla ENTER foi pressionada, pelo seu keyCode

function limpaInput(){

    inputTarefa.value = '';

    inputTarefa.focus(); //para deixar o focus do ponteiro marcado nela

}

Essa função cada vez que chamada limpa o conteúdo de inputTarefa, e coloca um o focus do ponteiro de escrita direto nesse input.

function criaBotaoApagar(li){

    li.innerText += ' ';

    const botaoApagar = document.createElement('button');

    botaoApagar.innerText('Apagar');

    li.appendChild(botaoApagar);

}

Essa função cria um botão em cada li criada, primeiro ela adiciona um espaço vazio na li, depois ele cria um botão, em seguida inseri um texto nesse botão, depois adiciona esse botão sendo filho de li.

// botaoApagar.classList.add('apagar');

    botaoApagar.setAttribute('class', 'apagar');

    botaoApagar.setAttribute('title', 'Apagar está tarefa');

aqui vemos dois tipos para inserir classes em nossos atributos.

No primeiro ele adiciona uma classList(‘apagar’) direto, nesse modelo só e possível criar o atributo de classe, já no segundo modelo,

Ele está chamando na primeira aspa o tipo de atributo, podendo ser href, class, type, title entre outros, e na segunda o seu valor.

document.addEventListener('click', function(e){

    const el = e.target;

    if (el.classList.contains('apagar')){

        console.log('achei');

    }

})

Função que fica verificando sé, o valor que foi clicado tem em seus atributos a classe apagar

document.addEventListener('click', function(e){

    const el = e.target;

    if (el.classList.contains('apagar')){

        el.parentElement.remove();

    }

})

Agora temos uma ação de remover.

Nesse caso ao clicar ele encontre uma classe com o nome de apagar, o elemento pai dessa class, será apagado. Removendo assim o pai dele, logo ele também será removido

Agora vamos entrar na parte de salvar essa tarefas direto no navegador com o uso de JSON.

function salvarTarefas() {

    const liTarefas = tarefas.querySelectorAll('li');

    console.log(liTarefas);

}

Com essa função selecionamos todas as li presentes em tarefas.

function salvarTarefas() {

    const liTarefas = tarefas.querySelectorAll('li');

    const listaDeTarefas = [];

    for (let tarefa of liTarefas){

        console.log(tarefa.innerText);

    }

}

Agora nós queremos somente o texto das tarefas.

Usamos então o for of, para fazer a interação.

Para cada li dentro de liTarefas, nós iremos jogá-la dentro de tarefa, e depois estamos pegando somente o texto dentro de tarefas.

function salvarTarefas() {

    const liTarefas = tarefas.querySelectorAll('li');

    const listaDeTarefas = [];

    for (let tarefa of liTarefas){

        let tarefaTexto = tarefa.innerText;

        tarefaTexto = tarefaTexto.replace('Apagar', '').trim();

        listaDeTarefas.push(tarefaTexto);

    }

}

Aqui estamos alternado o valor de Apagar de cada li para vazio e removendo os espaços vazios ao final e começo, em seguida estamos inserindo esses novos valores dentro da variável lista de tarefas.

Convertendo agora para JSON

function salvarTarefas() {

    const liTarefas = tarefas.querySelectorAll('li');

    const listaDeTarefas = [];

    for (let tarefa of liTarefas){

        let tarefaTexto = tarefa.innerText;

        tarefaTexto = tarefaTexto.replace('Apagar', '').trim();

        listaDeTarefas.push(tarefaTexto);

    }

    const tarefasJSON = JSON.stringify(listaDeTarefas);

    console.log(tarefasJSON);

}

Estamos agora pegando essa Array de listaDeTarefas que foi criada e transformando com um arquivo de texto JSON, com o uso do JSON.stringify

const tarefasJSON = JSON.stringify(listaDeTarefas);

localStorage.setItem('tarefas', tarefasJSON);

agora estamos salvando esse arquivo JSON no localStorage do navegador.

localStorage.setItem(‘como será chamado esse arquivo’, qual arquivo vai ser)

ficaram salva no navegador em Application -> local Storage -> site em questão

ele funciona como uma mine base de dados.

Agora vamos criar uma função para quando o site for carregado ele carregar essa informações que estão no localStorage

function adicionaTarefasSalvas(){

    const tarefas = localStorage.getItem('tarefas');

    const listaDeTarefas = JSON.parse(tarefas);

    for (let tarefa of listaDeTarefas) {

        criaTarefa(tarefa);

    }

}

adicionaTarefasSalvas();

aqui estamos pegando esse arquivo json que criamos, através do nome que colocamos quando foi pra criar e jogando em uma variável.

Em seguida estamos pegando essa variável e fazendo o parse e jogando em outra variável.

Depois usamos o for of, para cada item dentro de listaDeTarefas ele jogar em tarefa, e chamando a função que cria tarefa, passando cada valor que estava no json.

Lembre de chamar essa função para poder usar ela

JSON.stringinf() = converte um elemento javascript para um formato JSON

JSON.parse() = converte está tarefa de volta a um array

localStorage.getItem() = pega algo ao local storage

localStorage.setItem() = envia algo ao local storage