**Java script**

**<script src="script.js"></script>** para poder ter acesso do java no html, sempre é bom colocar ele no fim do </body>.

4 formas principais de mostrar para usuário final

**document.write("**algun texto") esta sendo executado junto com algo que já existe. Vc cria

**window.alert("**mensagem de exemplo") usando a janela do navegador

**alert**('opa, estou funcionado') para colocar uma mensagem

**console.log(**"a mensagem inserir  aqui aparecera no console.") mostrar mensagem no console.

**If** significa **se, Else** significa então, **depois, ||** significa **OU** dentro de uma condição

**var nome = "**frederico**";** essa variável vai ser enviada hosting em todo site, é uma String

**let nome = "**frederico**"**; essa é específico apenas para aquele console na área que vc vai usar

**const nome = "**frederico**"**; na deixa vc mudar por outra variável.

**var idade = 90**; number

**var salvo** = true; boolean verdadeiro ou falso, ligado ou desligado

**var salvo = false;** boolean verdadeiro ou falso, ligado ou desligado

**var cidade**; tipo indefinido.

**Typeof xxx;** no inspecionar, da o tipo de variável

1. **document.getElementById**("paragrafo**").innerHTML** = "Novo Comentario"; para trocar um texto do html por outro.
2. **var = paragrafo = document.getElementById("paragrafo");**

**.innerHTML = "Novo Comentario";** outra forma de mudar o html

1. **document.getElementsByClassName(‘lista’)** para selecionar por class, e pode ser repetido varias vezes.
2. **document.getElementsByTagName(‘.....’)** seleciona direto na tag
3. **document.getElementsByName(‘.......’)** seleciona pelo nome
4. **document.querySelector(‘......’)** para selecionar através do id ou classe usando . ou #
5. **document.querySelectorAll('.....')** //seleciona através da classe todos com o mesmo nome usando o .
6. **function alterar( ) { }** tudo que por aqui faz parte da função que preciso.
7. **Onclick**=”console.log(‘clicou’)” evento de clicar **Onmouseover**=”console .log(‘passou o mouse’)” **Onmouseout**=’console.log(‘tirou o mouse’)” **Onkeyup**  = evento de clique, quando o usuário solta a tecla **Onkeydow** = evento de clique onde enquanto apertado, a tecla esta sem parar
8. **Switch** para várias coisas, como case Temos dois tipos de **for() loop** e **for() lopp Array**

***for(let i = 0***;(criação das variáveis) i < 50;(Condição) i++) (Incrementção)

1. **White()** enquanto essa condição. Loop infinito, quase não usa
2. **Console.log(nome.length)** saber quantas caracteres o String tem
3. **nome.indexOf('lacerda');** pesquisar qual a posição que se encontra esse texto no let nome

if(nome.indexOf('silva') > -1) {resultado = 'achou!'} else { resultado + 'Não achou'}

1. ele pegue o caractere apartir do numero pedido.

Ex: let nome = 'frederico vidigal santos da silva' ; let resultado = nome.slice(0, 10); console.log(resultado );

15. **substring()** não pega negativos  **substr(posição inicial e quantos caracteres)**

16**. nome.replace('frederico', 'Nycollas');** usa para trocar um nome por outro. nome.toUppercase coloca tudo maiúsculo

17. **nome.toLowerCase()** coloca caixa baixa **nome.concat()**; quase inútil não se usa

18 **nome.trim()** ele tira os espaços. **nome.charAt(5);** saber a posição do string

**Métodos arrays**

1 **toString();** para juntar um array em lista. **join('-'**) ele pega o que tem dentro da aspa e coloca no arrey

2 **.indexOf('corante')** ele vai retornar a posição que se encontra, se der -1 é porque não encontrou.

3. **lista.pop();** remove o último **lista.shift()**; remove o primeiro da lista.

4. **lista.push('Prato');** para adicionar um item dentro do array **lista[0] = 'ovos'**; para trocar um item na posição informada.

5. **lista[lista.length] = 'ovos'**; ele descobre a posição e coloca.

**6. lista.splice(1, 1);** Para deletar um item do array usa se splice entre parêntese o numero do item e a quantidade

7.**let res = lista.concat(lista2)**; para juntar dois arrays em um so.

**8. lista.sort()**; para colocar a lista em ordem alfabética. **lista.reverse();**Para mudar a ordem é só depois adicionar

let lista = [45, 4, 9, 16, 25]; let lista2 = [];lista2 = **lista.map(function(item) {**

**return item \* 2**;}); let res = lista2; console.log(res); isso faz com que os números do array retorne vezes dois ou let lista = [45, 4, 9, 16, 25]; let lista2 = []; **for(let i in lista)** { **lista2.push(lista[i]** \* 2);}

let res = lista2; console.log(res); que é a mesma coisa .

**9.para filtrar** let lista = [45, 4, 9, 16, 25]; let lista2 = [];

lista2 = lista.filter(function(item){if(item <20) {return true;} else { return false }});

let res = lista2;console.log(res);

**10.lista.every()** para poder ver se todos é ou não **lista.some();** para ver se alguns são maiores que

lista2 = **lista.some**(function(item){return (item < 4)? true : false;}); forma resumida da condição

11.**Find ou findIndex** trás para mim o que estou procurando;

**let d = new Date(); console.log (d)** forma de puxar a hora.

13.**console.log (d.toDateString() )** data mais resumida console.log (d.toUTCString() ) data e hora sem fuso horário

14.**d.getFullYear();** exibir somente o ano **d.getMonth();** somente o ano lembrando que no java os meses começa de 0 a 11.

**15.d.getDate();** somente dia do mês **d.getDay();** somente o dia da semana, ela começa no domingo com 0 e vai ate 6 sabado

16.**d.getHours();** somente a hora.**d.getMinutes();** somente o minuto.**d.getSeconds();** somente os segundos

17.**d.getMilliseconds();** somente os mile segundos **d.getTime();** quantidade de segundos após 1970.

**Manipular o date**

18.**d.setFullYear(2000)** trocar o ano **d.setMonth(0)** trocar o mês **d.setDate(0)** trocar o dia.

19.**d.setDate(d.getDate() + 5);** para ver quantos dias a frente no caso 5

**na matemática usa se o Math. Alguma coisa.**

20.**math.Round(6,67)** ele arredonda o numero **Math.floor(6.67)** ele arredonda para baixo

21.**Math.Ceil(6.67)** ele arredonda para cima **Math.abs(-9.6598)** numero absoluto é ele positivo

22.**Math.min** diz quem é o menor de vários números **Math.max** diz quem é o maior de vários números

23.**Math.random()** número aleatório entre 0 e 1**Math.floor(Math.rondom() \* 100)** para pegar um numero aleatório de 0 a 100

**Para poder exibir a hora na pagina em tempo real**

function showtime() {let d = new Date(); let h = d.getHours(); let m = d.getMinutes();

let s = d.getSeconds(); let txt = h+':'+m+':'+s; document.querySelector('.demo').innerHTML = txt;}

**let timer = setInterval(showtime, 1000);**para por algo para rodar em determinado tempo uma só vez let timer;

**function rodar() { timer = setTimeout(function() { document.querySelector('.demo').innerHTML = 'Rodou!'; }, 2000 );}function parar(){**

**clearTimeout(timer)**

**}**

Para fazer um **template String** usa se `` virgula ao contrário e onde é os dados **${}** para simplificar a frase.

**Let { } = pessoa** para poder desconstruir objetos ou **function pegarNomeCompleto({nome, sobrenome}){Return `${nome} ${sobrenome}`;} console.log(pegarNomeCompleto(pessosal))**

**Arrow functions** let somar = (x, y) **=>** x + y; **let letrasNoNome = (nome) => nome.length;** quando tenho um paremetro só nao preciso usar os parentese na arrow function nao posso usar o this, somente emu ma função normal.

**Closest** ache o elemento mais próximo

**dom=** document object model todas as tag que tenho na pagina;

**function subirTela()** {**window.scrollTo({top:0,left:0,behavior: ‘smooth’});** para der subir o scroll suavemente

**Modificadores de acesso**

**private:** acesso apenas dentro da classe **protected:** acesso por classes no mesmo pacote e subclasses **public:** acesso de qualquer classe

**Modificadores de classes, variáveis ou métodos**

**abstract:** classe que não pode ser instanciada ou método que precisa ser implementado por uma subclasse não abstrata **class:** especifica uma classe **extends:** indica a superclasse que a subclasse está estendendo **final:** impossibilita que uma classe seja estendida, que um método seja sobrescrito ou que uma variável seja reinicializada **implements:** indica as interfaces que uma classe irá implementar **interface:** especifica uma interface **native:** indica que um método está escrito em uma linguagem dependente de plataforma, como o C

**new:** instancia um novo objeto, chamando seu construtor

**static:** faz um método ou variável pertencer à classe ao invés de às instâncias

**strictfp:** usado em frente a um método ou classe para indicar que os números de ponto flutuante seguirão as regras de ponto flutuante em todas as expressões

**synchronized:**indica que um método só pode ser acessado por uma thread de cada vez

**eywordse:** impede a serialização de campos

**volatile:**indica que uma variável pode ser alterada durante o uso de threads

**Controle de fluxo dentro de um bloco de código**

**break:** sai do bloco de eywor em que ele está

**case:** executa um bloco de código dependendo do teste do switch

**continue:**pula a execução do código que viria após essa linha e vai para a próxima passagem do loop

**default:**executa esse bloco de eywor caso nenhum dos teste de switch-case seja verdadeiro

**do:**executa um bloco de código uma vez, e então realiza um teste em conjunto com o while para determinar se o bloco deverá ser executado novamente

**else:** executa um bloco de código alternativo caso o teste if seja falso

**for:**usado para realizar um loop condicional de um bloco de código

**if:** usado para realizar um teste lógico de verdadeiro o falso

**instanceof:** determina se um objeto é uma instância de determinada classe, superclasse ou interface

**return:** retorna de um método sem executar qualquer código que venha depois desta linha (também pode retornar uma variável)

**switch:**indica a variável a ser comparada nas expressões case

**while:** executa um bloco de código repetidamente enquanto a condição for verdadeira

**Tratamento de erros**

**assert:** testa uma expressão condicional para verificar uma suposição do programador

**catch:** declara o bloco de código usado para tratar uma exceção

**finally:**bloco de código, após um try-catch, que é executado independentemente do fluxo de programa seguido ao lidar com uma exceção

**throw:**usado para passar uma exceção para o método que o chamou

**throws:** indica que um método pode passar uma exceção para o método que o chamou

**try:** bloco de código que tentará ser executado, mas que pode causar uma exceção

**Controle de pacotes**

**import:**importa pacotes ou classes para dentro do código **package:** especifica a que pacote todas as classes de um arquivo pertencem

**Primitivos**

**boolean:**um valor indicando verdadeiro ou falso **byte:** um inteiro de 8 bits (signed)

**char:** um caracter eyword (16-bit unsigned) **double:** um número de ponto flutuante de 64 bits (signed) **float:** um número de ponto flutuante de 32 bits (signed)

**int:** um inteiro de 32 bits (signed) **long:** um inteiro de 64 bits (signed)

**short:** um inteiro de 32 bits (signed)

**Variáveis de referência**

**super:** refere-se a superclasse imediata **this:** refere-se a instância atual do objeto

**Retorno de um método**

**void:** indica que o método não tem retorno

**Palavras reservadas não utilizadas**

**const:** Não utilize para declarar constantes; use public static final

**goto:** não implementada na linguagem Java por ser considerada prejudicial

**Literais reservados**

De acordo com a Java Language Specification, null, true e false são tecnicamente chamados de valores literais, e não eywords. Se você tentar criar algum identificador com estes valores, você também terá um erro de compilação.  
Read more: <http://www.linhadecodigo.com.br/artigo/83/as-52-palavras-reservadas-do-java.aspx#ixzz6rwucsKj5>

**Alguns comandos importantes**

**documentElement** Recupera o elemento raiz do documento HTML. **getElementById** Retorna um elemento que usa o atributo ID correspondente. **createElement** Cria um novo nó de elemento para a página. **createAttribute** Cria um novo nó de atributo para a página. **createTextNode** Cria um novo nó de texto para a página. **getElementsByTagName** Recupera uma lista de elementos com o nome em questão. **appendChild** Insere um novo elemento filho. **removeChild** Remove o elemento filho em questão. **parentNode** Retorna o nó pai de um dado nó.

**Quadro 5.1 Eventos de mouse Propriedade Ocorre quando...**

**onClick** O usuário clica em um elemento. **onDblClick** O usuário clica duas vezes em um elemento. **onMouseDown** O usuário pressiona o botão do mouse sobre um elemento. **onMouseMove** O ponteiro do mouse está em movimento sobre o elemento. **onMouseOver** O ponteiro do mouse está sobre o elemento. **onMouseOut** O usuário move o ponteiro do mouse para fora do elemento. **onMouseUp** O usuário solta o botão do mouse sobre um elemento.

**Quadro 5.2 Eventos de teclado Propriedade Ocorre quando...**

**onKeyDown** O usuário está pressionando uma tecla. **onKeyPress** O usuário pressiona uma tecla. **onKeyUp** O usuário solta a tecla (previamente pressionada)

**Quadro 5.3 Eventos de objetos ou frames Propriedade Ocorre quando...**

**onAbort** Há a interrupção no carregamento de uma imagem. **onError** Uma imagem não carrega adequadamente. **onLoad** Um objeto foi carregado. **onResize** Um documento visualizado é redimensionado. **onScroll** Um documento visualizado é rolado pela barra de rolagem do navegador. **onUnload** Uma página deixa de ser exibida.

**Quadro 5.4 Eventos de formulários Propriedade Ocorre quando...**

**onBlur** Um campo do formulário perde o foco. **onChange** O conteúdo de um campo do formulário é alterado. **onFocus** Um campo do formulário ganha foco. **onReset** Um formulário é restaurado, isto é, tem seus campos limpos. **onSelect** Um usuário seleciona algum texto dentro de um campo do formulário. **onSubmit** Um formulário é enviado

**métodos de manipulação de números**

n.**toString** transforma o número em String; n.**toFixed(2)** transformar um numero grande em um numero de duas casas decimais; **parseInt(n)** transforma a String em numero; **parseFloat(n)** transforma a String com decimal em número decimal;

**operador spread** (... mais tag anterior) **let numeros** = [1,2,3,4];

**let outros** = **[...numeros**,5,6,7,8];

console.log(outros) resultado outros = [1, 2, 3, 4, 5, 6, 7, 8]

**operador rest** function adicinar(**...**numeros) { console.log(numeros);}

adicinar(5,6,7,8,9,10) ex: function adicinar(nomes, ...novosNomes) { let novoConjunto = [ ...nomes,...novosNomes ]; return novoConjunto}let nomes = ["frederico", "Paulo"];let outros = adicinar(nomes, "Antonio", "maria", "jose");

console.log(outros);

**includes** você consegue ver se tem na lista. Ex: console.log(exemplo.includes(“o que procuramos”));

**repeat**  so colocarmos .repeat(quantidade de); **console.log( `${lista}`.repeat(5) );**

**keys** console.log(Object.keys(lista)); **values** console.log(Object.values(lista)); **entries** console.log(Object.entries(lista) );

**PARSE =** pega uma String e transforma em JSON (JSON.parse);

**STRING =** pega um JSON e transforma em String (JSON.stringify);

**SINCRONO = condigo estaem execução, inicia acaba e vai pro procimo**

let nome = 'frederico';

let sobrenome = 'vidigal';

let completo = nome+' '+sobrenome;

**ASSINCRONO= quando espera resposta continua execultando**

let nome = 'frederico';

let sobrenome = 'vidigal';

let temperatura = Maquininha.pegarTemperatura();//assincrona

let completo = nome+' '+sobrenome;

**CALLBACK**= Quando tiver o resultado, ele da a resposta.

Promisse = function pegarTemperatura() {

return new Promise(function(resolve, reject){

console.log("pegando temperatura...");

setTimeout(function(){ resolve('40 na sombra');

}, 2000); });

} function pegarTemperatura() {

    return **new** *Promise*(function(*resolve*, *reject*){

        console.log("pegando temperatura...");

        setTimeout(function(){

            resolve('40 na sombra');

        }, 2000);

    });

}

// usando a promisse

let temp = pegarTemperatura();

temp.then(function(*resultado*){

    console.log("temperatura: "+*resultado*)

});

temp.catch(function(*error*){

    console.log('Eita, deu errado!')

});

Modelo de requisição

fetch

<!DOCTYPE *html*>

<html *lang*="pt-br">

<head>

    <meta *charset*="UTF-8">

    <meta *http-equiv*="X-UA-Compatible" *content*="IE=edge">

    <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

    <link *rel*="stylesheet" *href*="style.css">

    <title>Página</title>

</head>

<body>

   <button *onclick*="loadPosts()">Carregar Post</button>

   <div *id*="posts"></div>

    <script *type*="text/javascript" *src*="script.js"></script>

</body>

</html>

//script.js

function loadPosts() {

    document.getElementById("posts").innerHTML= 'Carregando...';

    fetch('https://jsonplaceholder.typicode.com/comments')

        .then(function(*resultado*) {

            return *resultado*.json();

        })

        .then(function(*json*){

            document.getElementById("posts").innerHTML= *json*.length+' comentarios'

        })

        .catch(function(*error*){

            console.log("Deu problema!");

        });

}

Fetch 2

function loadPosts() {

    document.getElementById("posts").innerHTML= 'Carregando...';

    fetch('https://jsonplaceholder.typicode.com/posts')

        .then(function(*resultado*) {

            return *resultado*.json();

        })

        .then(function(*json*){

            montarBlog(*json*);

        })

        .catch(function(*error*){

            console.log("Deu problema!");

        });

}

function montarBlog(*lista*) {

    let html = ' ';

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

        html += '<h3>'+*lista*[i].title+'</h3>';

        html += *lista*[i].body+'<br/>';

        html += '<hr/>';

    }

    document.getElementById('posts').innerHTML = html;

}

Async e await

async function loadPosts() {

    document.getElementById("posts").innerHTML= 'Carregando...';

    let req = await fetch('https://jsonplaceholder.typicode.com/posts');

    let json = await req.json();

    montarBlog(json);

}

function montarBlog(*lista*) {

    let html = ' ';

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

        html += '<h3>'+*lista*[i].title+'</h3>';

        html += *lista*[i].body+'<br/>';

        html += '<hr/>';

    }

    document.getElementById('posts').innerHTML = html;

}

Fetch com post

<!DOCTYPE *html*>

<html *lang*="pt-br">

<head>

    <meta *charset*="UTF-8">

    <meta *http-equiv*="X-UA-Compatible" *content*="IE=edge">

    <meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

    <link *rel*="stylesheet" *href*="style.css">

    <title>Página</title>

</head>

<body>

   <button *onclick*="inserirPost()">Carregar Post</button>

       <script *type*="text/javascript" *src*="script.js"></script>

</body>

</html>

//script.js

async function inserirPost() {

    document.getElementById('posts').innerHTML = "Carregando...";

    let req = await fetch('https://jsonplaceholder.typicode.com/posts',{

        method: 'Post',

        body: JSON.stringify({

            title: 'Titulo de teste',

            body: 'Corpo de teste',

            UserId: 4

        }),

        headers: {

            'content-type': 'application/jason'

        }

    });

    let json = await req.json();

    console.log(json);

}

Get

async function inserirPost() {

    document.getElementById('posts').innerHTML = "Carregando...";

    let req = await fetch('https://jsonplaceholder.typicode.com/posts',{

        method: 'GET'

    });

    let json = await req.json();

    console.log(json);

}

Upload de arquivos com JS

async function enviar(){

    let arquivo = document.getElementById('arquivo').files[0];

    let body = **new** *FormData*();

    body.append('title', 'bla bla bla');

    body.append('arquivo' , arquivo);

    let req = await fetch('https://www.meusite.com.br/upload', {

        method: 'POST',

        body: body,

        headers: {

            'Content-type': 'multipart/form-data'

        }

    });

}

Thumbnails com JS

function mostrar(){

    let imagem = document.getElementById('imagem').files[0];

    let img = document.createElement('img');

    img.src = *URL*.createObjectURL(imagem);

    img.width = 600

    document.getElementById('area').appendChild(img);

}

Thumbnails com FileReader

function mostrar(){

    let reader = **new** *FileReader*();

    let imagem = document.getElementById('imagem').files[0]

    reader.onloadend = function() {

        let img = document.createElement('img');

        img.src = reader.result;

        img.width = 500;

        document.getElementById('area').appendChild(img);

    }

    reader.readAsDataURL(imagem);

}

Função calcular metro Quadrado

function calcularImovel(metragem, quartos){

    let metroQuadrado = 3.000

    let preco = 0

    switch(quartos) {

        case 1:

        default:

            preco = metragem \* metroQuadrado;

        break;

        case 2:

            preco = metragem \* (metroQuadrado \* 1.2);

        break;

        case 3:

            preco = metragem \* (metroQuadrado \* 1.5);

        break;

    }

    return preco;

}

let metragem = 123;

let quartos = 2;

let preco = calcularImovel(metragem, quartos);

console.log(`A CASA CUSTA R$ ${preco}`);

função para validar

function validar(usuario, senha) {

    if (usuario === 'pedro' && senha === '123') {

        return true;

    }else {

        return false;

    }

}

let usuario = 'pedro';

let senha = '1234';

let validacao = validar (usuario, senha);

if(validacao) {

    console.log('Acesso concedido.')

}else {

    console.log('Acesso NEGADO')

}