SUMÁRIO


```
//* Aula4 do comando console
//#obs ele cita como aula 3 no vídeo, na udemy está 4

//| diferenças ao apresentar uma string (texto)
//#nota a aspas duplas pode conter aspas simples e o aspas simples pode envolver aspas duplas
//#nota o com crase pode conter ambas as aspas e é usado para template string

console.log('Jean "Meira"');
console.log("Jean 'Meira'");
console.log(`Jean Meira`);

//| números
//#nota todos são tipo number, não muda de inteiro(int) para ponto flutuante(float)

console.log(15.85);
console.log(35);

//| Múltiplos valores

console.log(35, 15.85, 'Jean Meira de novo');

//* Aula 8 - comentários
//#nota# comentários são ignorados ao se executar (pela engine/motor)

// Isto é um comentário
console.log('Hello world'); // aqui temos outro comentário
```

```
//console.log('código não executado')

//#nota# códigos com (//) serão considerados comentários e não são executados
/*  isto é
    é um comentário
    que permite mais de uma linha,
    formando um comentário por bloco
    #importante# não deve se esquecer de fechar
*/

//* Aula 09 - navegador vs Node (HTML+JavaScript)

console.log('Este trecho será exibido no console do navegador, usando um arquivo .js')

//#nota# é possível criar uma pasta com todos scrips .js que serão usados pelo arquivo
//#nota# assim é possível devinir um lugar com blocos convenientes de arquivos .js

//* HTML A PARTIR DAQUI
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <title>Minha primeira página HTML</title>

    <script>
        // um comentário
        console.log('olá mundo');

/*
#nota# não é uma boa prática colocar código javascrit nessa parte do    código,pois ao colocar nesse local gerará atraso no carregamento da página, devido ao fato de que ao encontrar um código o navegador tentará executar os scrips.
*/
    </script>
</head>

<body>

    <script src='index.js'>
        // console.log('Este trecho será exibido no console do navegador')

/* código JS pode ser colocada dentro da tag scrip, mas é melhor
```

```
prática separar os arquivos, usando assim o apontamento do caminho (src)
para o arquivo .js, da forma a seguir
            <script src='index.js'>
*/
        // Um melhor prática é adicionar os scripts no final do body
    </script>
</body>
</html>

//* Aula 10 - variáveis let e var
//#obs# aula 6 citada no vídeo

//#nota# variáveis podem ser atribuídas como let e var
//#nota# podendo ou não ser inicializadas com valor, se não receber valor
será considerado undefined
//#obs# le- se como variável do tipo let recebe o valor do tipo string
'João'.

let nome = 'João';          //? já inicializado na declaração

let nome2;                  //? declaração
nome2 = 'qualquer valor';   //? atribuição de um valor

console.log(nome, 'nasceu em 1984');
console.log('Em 2000', nome, 'conheceu Maria');
console.log(nome, 'caso-se com Maria em 2012');
console.log('Maria teve 1 filho com', nome, 'em 2015');
console.log('O filho de', nome, 'se chama Eduardo');

//#nota# var é mais antigo, o var permite redeclaração enquanto o let não,
 ambos pode serem reatribuidos
//#nota# recebendo assim outros valores
//#aviso#let não podem ser declaradas mais de uma vez enquanto var pode
var nome3 = 'Jean';
var nome3 = 'Guilherme';

console.log(nome3);

//#obs# A variável permite salvar valores que serão usados em múltiplas pa
rtes do código, além de trabalhar
//#obs# informações no decorrer do código e poder salvar informações vinda
s do usuário ou fontes externas.

//#importante# o uso de variáveis ou partes dinâmicas de código podem mais
 facilmente ser alteradas,
//#importante# o código exemplificado acima pode mudar o nome citados em t
odas as frases somente alterando
```

```
//#importante# a linha de declaração da variável nome, o que não seria pos
sível se fosse manualmente
//#importante# escrito em cada comando console.log();


//#nota# Não podemos criar variáveis com palavras reservadas ex -> let if
-- let let, e assim por diante;
//#nota# É recomendado que variáveis tenham nomes significativos ex -> let
 n = 'João';
//#nota# n é muito vago, podendo ser qualquer coisa
    //#obs# É bom que a variável tenha valor semântico.
//#nota# Não começar o nome de uma variável com um número;
//#nota# Variáveis em geral começam com letras minúsculas, (existem exceçõ
es);
//#nota# Não podem conter  espaços ou -, ex -> let nome cliente; let nome-
completo;
    //#obs# para variáveis com nomes múltiplos pode-se usar o padrão
camelCase ex -> let nomeCLiente.
//#nota# Variáveis são case-
sensitive, ou seja, nas variáveis letras maiúsculas e minúsculas diferem
//#nota# no resultado final;

//#importante#  NÃO UTILIZE VAR, UTILIZE LET

//* Aula 11 - Const

//* As mesmas regras de variáveis valem para constantes
//#nota# Não podemos criar constantes com palavras reservadas ex -> let if
 -- let let, e assim por diante;
//#nota# É recomendado que constantes tenham nomes significativos ex -> le
t n = 'João';
//#nota# n é muito vago, podendo ser qualquer coisa
    //#obs# É bom que a constante tenha valor semântico.
//#nota# Não começar o nome de uma constante com um número;
//#nota# constantes em geral começam com letras minúsculas, (existem exceç
ões);
//#nota# Não podem conter  espaços ou -, ex -> let nome cliente; let nome-
completo;
    //#obs# para constantes com nomes múltiplos pode-
se usar o padrão camelCase ex -> let nomeCLiente.
//#nota# constantes são case-
sensitive, ou seja, nas variáveis letras maiúsculas e minúsculas diferem
//#nota# no resultado final;
//#importante# Constantes não podem ser redeclaradas e nem tem novos valor
es atribuídos as mesmas.
    //#obs# não podem modificar seu valor
//#importante# devem ser inicializadas e declaradas ao mesmo tempo
```

```
const nome = 'João';

console.log(nome);

//#nota# É possível usar uma constante ou variável na declaração de outra.

const primeiroNumero = 5;
const primeiroNumeroStrg = '5';
const segundoNumero = 10;
const resultado = primeiroNumero * segundoNumero;
const resultadoDuplicado = 2* resultado;

console.log(resultado);                                    //? resposta es
perada -> 50
console.log(resultadoDuplicado);                           //? resposta es
perada -> 100

//* Verificando o tipo de uma const ou let com a função typeof();

console.log(primeiroNumero + segundoNumero);                //? resposta e
sperada -> 15
console.log(typeof(primeiroNumero + segundoNumero));        //? resposta e
sperada -> number
console.log(primeiroNumeroStrg + segundoNumero);            //? resposta e
sperada -> 510
console.log(typeof(primeiroNumeroStrg + segundoNumero));    //? resposta e
sperada -> string

//#nota# Ao receber uma string e um number o código concatena os valores e
m vez de somar,
//#nota# aprensentando a string 5 e o número 10 escritos em sequência.
//#nota# Porém, dessa forma o resultado passa a ser interpletado como uma
string.

//* Aula 15 - primeira diferença entre var e let
//#obs# no vídeo citado como aula 9

//? Var Aceita redeclaração

var nome ='Luiz';   //? Declaração
var nome = 'Otávio' //? Re-declaração

console.log(nome);  //? resultado esperado -> Otávio

//#aviso# Adendum não faça o demonstrado a seguir
nome1='Luiz';
```

```
//* Aula 16 - Tipos de dados primitivos

//| strings

const nome = 'Luiz';
const nome1 ="Luiz";
const nome2 = `Luiz`;

//| numbers

const num1 = 10;
const num2 = 10.5;

//| undfined
let nomeAluno               //? undefined -> não aponta para nenhum local
na memória

//| null
let sobernomeAluno = null   //? null -> não aponta para nenhum local na me
mória

//#nota# undefined != de null
//#nota# null é a indicação que foi escolhido que a variável não terá um v
alor, não aponta para nenhuma memória
//#nota# undefined é uma variável que não recebeu um valor.

//? boolean
const boolean = true;
const boolean2 = false;

//#nota# boolean assume valor true ou false, representando verdadeiro ou f
also, 1 ou 0;
//#obs# boolean tem peso lógico, ajuda em decisões do código. Muda o fluxo
 da aplicação, usando desvios condicionais.

//* Aula 17 - Operadores aritméticos, de atribuição e incremento.

//| Aritméticos
//_ adição          -> +
//_ subtração       -> -
//_ divisão         -> /
//_ multiplicação   -> *
//_ potenciaçã      -> **
//_ resto da divisão-> %

//#nota# a precedêndia das operações são realizadas conforme a matemática
```

```
//_ exemplo
const num1 =5;
const num2 =2;
const num3 =10;

console.log(num1 + num2 * num3);    //? resultado esperado -> 25

/*
    ! Ordem de precedência segue conforme indicado abaixo
    _ 1º ** (potenciação)
    _ 2º * (multiplicação) , / (divisão) e % (resto da divisão ou módulo d
a divisão)
    _ 3º + (adição) e - (subtração)
*/

//#obs# é possível alterar a ordem de precedência com o uso parênteses ()
//_ exemplo

console.log((num1 + num2) * num3);    //? resultado esperado -> 70

//| Incremento e decremento
//_  ++ soma um no valor (incrementa)
//_  -- subtraí um no valor (decrementa)

//_ exemplo (funcionam para incremento e decremento)

let contador =1;
contador++;
console.log(contador);

console.log(contador++)            //? realiza a ação e depois incrementa
 (pós incremento)
    //#obs# evite usar o modelo acima, com pós incremento junto com a funç
ão que irá usar de imediato, pois pode causar bugs.

console.log(++contador)            //? incrementa e depois realiza a ação
 (pré incremento)

//| Operadores de atribuição
//_ incremento de mais de uma unidade

const passo =2;
let contador1 =0;

contador1 = contador1 + passo;
console.log(contador1);
```

```
//_ De modo simplificado
//#obs# é possível usar com incremento, decremento, multiplicação, divição
 e potenciação

contador1 += passo;         //? o mesmo que digitar -> contador1 = contado
r1 + passo;
console.log(contador1);

/*
    ! cuidado com usar contas ou atribuições com variáveis, ao se usar com
 variáveis com tipagem diferentes de números
    ! pode se obter resultados adversos e inesperados.
    ! podendo até obter resultados NaN -> not a number
*/

//#importante# Quando for possível converta as variáveis para o tipo desej
ado para garantir o funcionamento

//_ Exemplo
const numTest = parseInt('5');

console.log(typeof(numTest));       //? resultado esperado -> number

//_ parseInt()      -> converte para inteiro, sem números após a vírgula
//_ partseFloat()   -> converte para float, números com valores após a vír
gula
//_ Nuber()         -> converte para número, sem distinção

//* Aula 18 - Alert, confirm e Prompt (Navegador)

//| Alert
alert('Mensagem');

//#nota# alert é um método do objeto window
//#obs# alert é um atalho para window.alert();
//_ o retorno é undefined, ou seja, não retorna valor algum.

//| Confirm
window.confirm('Deseja realmente apagar?');

//| prompt
//? sempre vai te retornar uma string
window.prompt('Digite o seu nome.');

//| é possível capturar o retorno da função
```

```
const confirma = confirm('Realmente deseja apagar?');
console.log('confirma tem valor:', confirma);

let num1 = prompt('Digite um número');
alert(`Você digitou: ${num1}`);
console.log(`Você digitou: ${num1}`);

//| A partir daqui é HTML
<!DOCTYPE html>
<html lang="pt-BR">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Aula 18</title>
</head>
<body>
    <script src="./script.js"></script>
</body>
</html>

//* Aula 21 - Mais sobre Strings


let umaString = "Um \"texto\""; //? a barra invertida é um caractere de es
cape.
//_ dessa forma é possível inserir aspas duplas em uma string declarada co
m aspas duplas

console.log(umaString);

umaString = "Um \\texto";        //? usando duas barras é possível fazer u
ma barra aparecer no resultado final

console.log(umaString);

//#nota# uma string é indexável. Cada caractere tem um índice. Começando e
m 0 (zero);
umaString = "Um texto";
console.log("A quinta letra da váriável umaString é: "+umaString[4]);
console.log(umaString.charAt(4));   //? usando o método charAt(); para rea
lizar o mesmo.
console.log(umaString.charCodeAt(4));   //? retorna o código da tabela ASC
II.

//| função concat concatena os textos, como o + ou uso de templateString
console.log(umaString.concat(' em', ' um', ' lindo dia'));
console.log(umaString + ' em' + ' um'+ ' lindo dia');
```

```
console.log(`${umaString} em um lindo dia`);

//| Pesquisando índice que começa uma palavra
console.log(umaString.indexOf('texto'));
//#nota# retorna o índice se achar e -1 se não achar.

//#obs# é possível enviar dois parâmetros, então o segundo será de onde a
pesquisa começa
console.log(umaString.indexOf('um', 3));    //? retorno -
1 pois não há um após o índice 3.

console.log(umaString.lastIndexOf('o'));    //? começa a procura do final
para o começo.
//#obs# pode dar diferença se mandar o parâmetos de início.

console.log(umaString.search(/x/));         //? similar ao indexOf, mas ac
eita expressões regulares.
//#obs# e em realidade pode ter funcionamentos um pouco mais amplos.

//| Replace
console.log(umaString.replace('Um', 'outra')); //? substitui uma palavra p
or outra

umaString = 'O rato roeu a roupa do rei de roma';
console.log(umaString.replace(/r/, '#'));     //? substitui somente o 1º r
console.log(umaString.replace(/r/g, '#'));    //? substitui todos os r

//| Checando o tamanho

console.log(umaString.length);
//           012345
umaString = 'O rato';
console.log(umaString.length);      //? conta como 0 a 5, ou seja 6

//| dividindo uma string

umaString = 'O rato roeu a roupa do rei de roma.';

console.log(umaString.slice(2, 6));
//#nota# Indica a posição de início e final, sendo que o final não é
contado.

console.log(umaString.slice(2));
//#nota# Se não receber o segundo parametro, conta como do início indicado
 até o final.

console.log(umaString.length-3);        //? resultado esperado -> 32
```

```
console.log(umaString.slice(-3));
console.log(umaString.slice(32));
/*
    _ Usar número negativo é o mesmo que adotar o início como o tamanho to
tal do texto
    _ menos o valor indicado. Sendo assim, no exemplo acima é printado ape
nas os 3
    _ últimos caracteres. E -3 é o mesmo que indicar o início de 32.
    _ Também é possível indicar começo e final com números negativos.
*/

//? também é possível usando outro comando.
console.log(umaString.slice(-5, -1));
console.log(umaString.substring(umaString.length - 5, umaString.length -
1));

//? Baseado em algum caractere.

console.log(umaString.split(' '));
console.log(umaString.split('r'));

console.log(umaString.split(' ', 2));       //? é possível limitar o númer
o de resultados.

//| representar em maiúsculo ou minúsculo
console.log(umaString.toUpperCase());
console.log(umaString.toLocaleLowerCase());

//* Aula 23 - Mais sobre números

let num1 =1;    // numbe
let num2 =2.5;  // number

console.log(num1 + num2);
console.log(num1.toString()+ ' e ' + num2.toString());
console.log(typeof num1);

//_ É realizado a contenação, mas o tipo do número permanece inalterável.

//? para converter -> num1 = num1.toString();

num1 = 15;
console.log(num1.toString(2));  //? representa o número em base binária.

num1 = 10.872158137674;
console.log(num1.toFixed(2));   //? limita o número de casas após a vírgul
a que será mostrado
```

```
//#obs# o número é arredondado.
//#aviso# recomenda-se fazer todos os
cálculos e apenas usar esse artifício ao exibir o resultado.

console.log(Number.isInteger(num1));    //? o número é um tipo inteiro ?
//_ Retorna verdadeiro ou falso.

let temp = num1 * 'olá';
console.log(temp, Number.isNaN(temp));  //? a conta é inválida, retornando
 um NaN;

//! padrão IEEE 754-2008 -> para imprecisão

num1 = 0.7;
num2 =0.1;

console.log(num1+num2);     //? resultado -> 0.79999...

//? o mesmo que num1 + num2;
num1 += num2;  //? 0.8
num1 += num2;  //? 0.9
num1 += num2;  //? 1.0

console.log(num1);  //? mas da o resultado 0.9999...

num1 = num1.toFixed(2); //? fixa apenas duas casas;
console.log(num1);      //? o resultado da 1.00, parecendo correto.
console.log(Number.isInteger(num1), Number.isInteger(1.00));    //? aqui n
ota-se a diferença.

//| Para corrigir o problema devemos...
num1 = 0.7;
num2 =0.1;

num1 += num2;  //? 0.8
num1 += num2;  //? 0.9
num1 += num2;  //? 1.0

num1 = Number(num1.toFixed(2));     //? fixar as casas garantindo que será
 um número

console.log(num1);
console.log(Number.isInteger(num1));    //? só será inteiro se as casas ap
ós o ponto forem .00

//| outra forma seria trabalhar com contas.
//_ não somando números com vírgulas
```

```
num1 = 0.7;
num2 =0.1;

num1 = ((num1*100)+(num2*100))/100;
console.log(num1);  //? 0.8

num1 = ((num1*100)+(num2*100))/100;
num1 = ((num1*100)+(num2*100))/100;

console.log(num1);
console.log(Number.isInteger(num1));

//* Aula 24 - Objeto Math

/*
    _ Math é um objeto embutido que tem propriedades
    _ e métodos para constantes e funções matemáticas.
    _ Não é um objeto de função.
    _ Todas as propriedades e métodos de Math são estáticas.
*/

let num1 = 9.54578;

//| arredondamento
//_ Para baixo
let num2 = Math.floor(num1); //? arredondando para baixo.
console.log(num2);  //? resultado esperado -> 9;

//_ Para cima
num2 = Math.ceil(num1); //? arredonda pra cima.
console.log(num2);  //? resultado esperado ->  10;

//_ Para o que estiver mais próximo
num2 = Math.round(num1); //? arredonda para o mais próximo
console.log(num2);  //? resultado esperado -> 10

num1 = 9.44578;
num2 = Math.round(num1);
console.log(num2);  //? resultado esperado -> 9

num1 = 9.50;
num2 = Math.round(num1);
console.log(num2);  //? resultado esperado -> 10
//#obs# Se o número estiver na metade será considerado para cima

//| O maior e menor número de uma sequência
```

```
console.log(Math.max(1,2,3,4,5,-10,-
50,1500,9,8,7,6));  //? resultado esperado -> 1500
console.log(Math.min(1,2,3,4,5,-10,-
50,1500,9,8,7,6));  //? resultado esperado -> -50

//| gerando um número aleatório entre 0 e 1, sem incluir o 1;
console.log(Math.random());

//_ é possível trabalhar para gerar outros resultados
const aleatorio = Math.round(Math.random() * (10-5) + 5);
console.log(aleatorio); //? aleatório entre 5 e 10.

//| o Math tem constantes
console.log(Math.PI);   //? o pi = 3.14158...
console.log(Math.E)     //? o número de Euler = 2,71828...

//| Potenciação
console.log(Math.pow(2, 10));   //? 2 elevado a 10
//#obs# pode-se dizer que é o mesmo que
console.log(2 ** 10);       //? 2 elevado a 10

//| Raiz
num1 = 9;
console.log(num1 ** (1/2));
console.log(Math.sqrt(num1));

//! Cuidado com divisões por 0, pois no Js isso é possível.
//? ex:
console.log(100 / 0); //? o número retornará infinity

//* Aula 26 - Arrays (básico)

const alunos = ['Jean', 'Maria', 'João'];
console.log(alunos)
//_ tentar organizar com um tipo só de dados dentro, mas não é regra
//_ o JavaScript permite a adição, mas não é boa prática de programação

//_ Arrays são indexados por elementos, começando no valor 0.
console.log(alunos[0])

//| é possível editar ou adicionar
alunos[1] = 'Roberta';
alunos[3] = 'Luiza';
console.log(alunos);

// _ Se o Array tem um elemento na posição ele altera, se não ele adiciona
.
```

```
//| Se quiser adicionar no último é possível usar conforme a seguir.
console.log(alunos.length)
alunos[alunos.length] = 'Fábio';
alunos[alunos.length] = 'Luana';
alunos.push('Otávio');
console.log(alunos);

//| Se quiser adicionar no começo...

alunos.unshift('Luiza');
console.log(alunos);

//| Removendo
alunos.pop();   //? remove o último
//_ é possível salvar o elemento removido
const removido = alunos.pop();
console.log(alunos, removido);

alunos.shift(); //? remove do começo
console.log(alunos);

delete alunos[2];   //? remove o índice específico
console.log(alunos, alunos[2])
//_ O delete não muda os índices, o que ocorre com pop e shift. O local ap
agado fica como undefined para o caso do delete;

alunos[2] = 'Carlos';
//| Dividindo um Array
console.log(alunos.slice(0, 3));
console.log(alunos.slice(0, -1));

//_ Arrays retornar como objetos, pois são como obejtos indexados
console.log(typeof alunos);
//? é possível checar se é um array
console.log(alunos instanceof Array);

//#Aviso# geralmente é melhor adicionar elementos no final do array, para
que não se tenha que mudar todas as possições.
//#Aviso# isso torna a performance do programa ruim.

//* Aula 28 - Funções (básico)

//#nota# funções executam ações, podendo ou não retornar algo

function saudacao(){
    console.log('Bom dia');
```

```
};

saudacao();

//| funções que recebem parâmetros

function saudacao1(nome){
    console.log(`Bom dia ${nome}`);
};

saudacao1('Jean');

//#nota# função são reutilizáveis
saudacao1('Fulano');

const variavel = saudacao1('Fulano');
console.log(variavel);
//#obs# toda função por padrão retorna undefied, e por isso é salvo undefi
ned na variavel.

//| Funções com retorno

function saudacao2(nome){
    console.log(`Bom dia ${nome}`);
    return 123456
};

const retorno = saudacao2('Jean');
console.log(retorno);

//#aviso# Esperasse que o retorno da função seja algo semantico com o nome
, e não que seja algo bem diferente.

function saudacao3(nome){
    return `Bom dia ${nome} pelo retorno`
};

const retorno2 = saudacao3('Jean');
console.log(retorno2);

//| funções com mais uso e que permitem repricar o seu uso.

function soma(x,y){
    const resultado = x +y;
    return resultado;
}
```

```
console.log(soma(2,2));
console.log(soma(3,1));
console.log(soma(5,10));

// console.log(soma(2,2), resultado); //#nota# daria erro pois o resultado
 faz parte do escopo local da função.

const resultado = soma(2,2);    //? é possível declarar uma const resultad
o, pois a que está dentro da função está isolada.

//! Quando a função chegar em um return sairá da função e aplica o retorno
, tudo após um return será ignorado.

function soma1(x,y){
    const resultado = x +y;
    return resultado;
    console.log('olá mundo');
}

console.log(soma1(5,10));

//| Valores faltando

console.log(soma()); //? retorna NaN
//#obs# será NaN mesmo se faltar só um

//_ É possível tratar atribuindo um valor inicial para os argumentos da fu
nção
function soma2(x=1,y=1){
    const resultado = x +y;
    return resultado;
}

console.log(soma2());
console.log(soma2(5,10));

//#obs# caso não receba valores, serão usados os pré-
definidos, caso receba será adotado o recebido.

//| Outros modos de criar funções

//_ Funções anônimas -> em variáveis.
//#obs# precisa do ponto e vírgula obrigatoriamente no final.

const raiz = function(n){
    return (n ** 0.5);
};
```

```
console.log(raiz(9));
console.log(raiz(16));
console.log(raiz(25));

//_ Arrow function

const raizArrow = (n) => {
    return ( n ** 0.5);
};

console.log('Arrow', raizArrow(9));

//_ Se tiver apenas uma linha de retorno é possível simplificar ainda mais
.

const raizArrow1 = (n) =>  n ** 0.5;
console.log('Arrow resumida',raizArrow1(9));

//#Aviso# funções são basicamente iguais, mesmo com as declarações diferen
tes.
//#Aviso# Quando entrar em this terão diferenças que serão expostas
//#importante# Não é uma boa prática criar funções que executem diversas a
ções.

//* Aula 29 - Objetos (básico)

//#Nota# é possível alteral os valores dos elementos de objetos e arrays.
//#Nota# mas não é possível reatribuir o valor ou mudar o tipo.
//#Nota# ao mudar elementos internos não há alterações pra onde é apontato
 na memória.

//| declaração

const pessoa1 = {
    nome: 'Luiz',
    sobrenome: 'Miranda',
    idade: 25
};

console.log(pessoa1.nome);
console.log(pessoa1.sobrenome);
//#Nota# da pra criar atributos que são como variáveis, mas estão internos
 ao objeto.
//#Nota# os atribuitos são separados por uma , no final de cada linha
//#Nota# e num objeto utiliza par chave e valor para cada atributo, que sã
o separados por :
```

```
//| criar por funções -> function factory
// fábrica de objetos.

function criaPessoa(nome, sobrenome, idade){    //? Isso são parâmetros de
 uma função
    return{ nome, sobrenome, idade};
    /*  É o mesmo que citar como nome: nome e assim por diante,
        Pois quando a chave e o valor possuem o mesmo nome é possível
        omitir o : valor
    */
}

const pessoa2 = criaPessoa('Jean','Meira','23');    //? Aqui são argumento
s que serão passados para os parâmetros.
console.log(pessoa2);

//#Nota# argumentos são os valores que são passados para o parãmetro.

//| Criar método no objeto

const pessoa3 = {
    nome: 'Jean',
    sobrenome: 'Meira',
    idade: 23,
    fala(){
        console.log(`${this.nome} ${this.sobrenome} está falando oi...
        A minha idade é ${this.idade}`)
    },
    incrementaIdade(){
        this.idade++;
    }
}

pessoa3.fala();
pessoa3.incrementaIdade();
pessoa3.fala();

//#Nota# this referencia o contexto da função, mas será dado mais detalhes
 a frente.

//* Aula 30 - Valores primitivos e por referência

/*
    _ Primitivos (imutáveis) - string, nuber, boolean, undefined, null
    _ também existem os bigint e symbol
    _ dado é o valor, a variável é somento uma caixa que contém o valor.
```

```
*/

let nome = 'Luiz';
nome = 'Otávio';
console.log(nome);
//#Obs# apesar de trocar o que está escrito, não se altera o dado / valor
primitivo

let a = 'A';
let b = a; // é feito uma cópia
console.log(a, b);

a ='Outra coisa';
console.log(a, b);
//_ b não se altera por ser uma cópia de a, então alterando a o b não é af
etado.

/*
    _ Por Referência (mutável) - array, object , function
    ! são passados por referência
    _ O que na verdade quando são atribuidos a outras variáveis,
    _ significa que vão apontar para a mesma referência, o mesmo local
    _ na memória.
*/

let c = [1, 2, 3];
let d=c;    // d vai apontar para o mesmo lugar na memória
console.log(c,d);

c.push(4);
console.log(c,d);

//_ b é afetado, pois o locar na memória é o mesmo, então mudando por a ou
 por b, afeta ambos
//? exemplo
d.pop();
console.log(c,d);

//| para cópiar o valor e mudar o local apontado na memória

let e = [1, 2, 3];
let f= [...e];
e.push(4);
console.log(e,f);
```

```
//* Aula 33 - Operadores de comparação

/*
    _ >     maior que
    _ >=    maior que ou igual a
    _ <     menor que
    _ <=    menor que ou igual a
    _ ==    igualdade   (checa valor)
    _ ===   igualdade estrita (checa valor e tipo)
    _ !=    diferente   (checa valor)
    _ !==   diferente estrito   (checa valor e tipo)

*/

console.log(10>5);
//? É possível salvar o valor em variáveis e constantes
const comp = 10 > 5;
console.log(comp);

//_ A funcionalidade é igual as usadas em matemática. Os estritos tem um c
onceito a mais.

//| Com variáveis

let num1 = 10;
let num2 = 11;

console.log(num1 <= num2);

//| Comparação e comparação estrita

num1 = 10;
num2 = 10;
console.log(num1 == num2);
num2 = '10';
console.log(num1 == num2);
console.log(num1 === num2);

/*
    _ A comparação normal somente compara o valor contido, enquanto
    _ a comparação estrita compara o valor e o tipo da variável ou
    _ dado em questão. Isso é válido para a igualdade e desigualdade.
    ! O uso do modo estrito é altamente recomendado para evitar
    ! Comportamentos indesejados na execução do código.
    ! EVITE O USO DA IGUALDADE E DESIGUALDADE COMUNS
*/
```

```
//* Aula 34 - Operadores Lógicos


/*
    _ Operadores lógicos
    _ && -> and -> e
    _ || -> or -> ou
    _ ! -> not -> não
*/

//| && (and)
//_ Para ser verdadeiro todos as expressões precisam ser verdadeira
console.log(true && true);
console.log(true && true && true && true);

//_ É possível salvar o valor em uma variável ou constante
const expressaoAnd = (true && true && true && true);
console.log(expressaoAnd);

//_ Se uma for falsa o resultado já é falso
console.log(true && false);
console.log(true && true && false && true);

//| || (or)
//_ Para ser verdadeiro pelo menos uma das expressões precisam ser verdade
ira
console.log(true || true);
console.log(true || false);

//_ É possível salvar o valor em uma variável ou constante
const expressaoOr = (true || true);
console.log(expressaoOr);

//_ Se uma for falsa se todas as expressões forem falsas
console.log(false || false);
console.log(false || false || false || false);
```

//| ! (not)

```
//_ Sai o oposto do que entrou
console.log("negação de true:", !true);
console.log("negação de dupla de true:", !!true);

//* Aula 35 - Avaliação de curto-circuito (short-circuit)

/*
    _ && -> false && true -> false : retorna "o valor" ao achar uma falsa
```

```
    | FALSY (VALORES QUE PODEM SER AVALIADOS COMO FALÇO)
    _ false
    _ 0
    _ '' "" `` (string vazias)
    _ null / undefined
    _ NaN

    #nota# Qualquer valor diferente dos presentes no FALSY avalia verdadei
ro
*/

//? exemplo
/*
    _ Ao se deparar com um FALSY o valor do mesmo é retornado, se em uma
    _ comparação não tiver nenhum FALSY a mesma irá retornar o último valo
r lido.
    _ Isso possibilita fazer uma redução nos circuitos para alguns casos,
tornando o
    _ código mais limpo e performático.
*/
console.log("Jean" && 0);   //? retorna 0;

/*
    _ A seguir tem um caso de curto circuito, onde o código é omitido em p
artes
    _ Porque devido as propriedades de falsy o código é perfeitamente váli
do.
    _ Será testado a condição e ao perceber se vaiExecutar é verdadeiro ou
 falso
    _ já retornará o resultado.
*/
function falaOi (){
    return 'Oi';
}

const vaiExecutar = false;

console.log(vaiExecutar && vaiExecutar);

/*
    _ O || (or) tem comportamento ao contrário
    _ || -> false && true -> true : retorna "o valor" ao achar uma true

    | FALSY (VALORES QUE PODEM SER AVALIADOS COMO FALÇO)
    _ false
    _ 0
    _ '' "" `` (string vazias)
```

```
    _ null / undefined
    _ NaN

    #nota# Qualquer valor diferente dos presentes no FALSY  avalia verdade
iro
*/

//* Aula 36 - if, else if e else (1)

const hora = 12;

if (hora >= 0 && hora < 12){
    console.log('Bom dia');
} else if(hora >= 12 && hora < 18){
    console.log('Boa tarde');
} else if (hora >= 18 && hora <=23){
    console.log('Boa noite');
}else {
    console.log('Olá');
}

/*
    _ If é o primeiro teste de condição, que ser verdadeiro, executa o cód
igo contido nas chaves.
    _ if pode ser usado sozinho.
    _ Se a condição for falsa pode ter outras condições diversas com o els
e if, em qualquer quantidade.
    _ O else if precisa ver seguido de um if, não podendo ser usado sozinh
o.
    _ Se todas as condições forém falsas será executado o conteúdo do else
.
    _ O else vem no final e só pode ser usado um.
*/
```