```
//* 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 func
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 número de resultados.

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