

INICIANDO JAVASCRIPT



**QUE A FORÇA ESTEJA COM VOCÊ.
PROF. FERNANDO LUCAS**

INTRODUÇÃO AO JAVASCRIPT

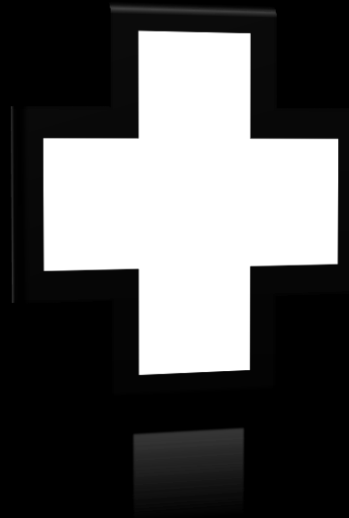
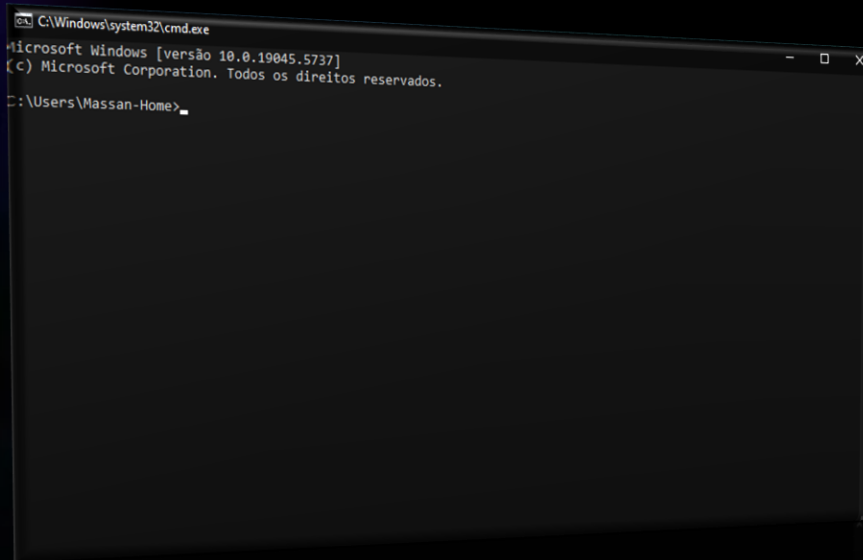
JAVASCRIPT É UMA LINGUAGEM DE PROGRAMAÇÃO USADA PARA ADICIONAR LÓGICA, COMPORTAMENTO E INTERATIVIDADE.

ELA É LEVE, INTERPRETADA, MULTIPARADIGMA E MUITO USADA TANTO NO NAVEGADOR QUANTO NO BACKEND (COM NODE.JS).



GLOSSÁRIO:
MULTIPARADIGMA
BACKEND

VAMOS UTILIZAR O CMD PARA CRIAR UMA NOVA PASTA E ABRIR A MESMA NO VSCODE.



COMANDOS –

DIR – LISTA OS ARQUIVOS

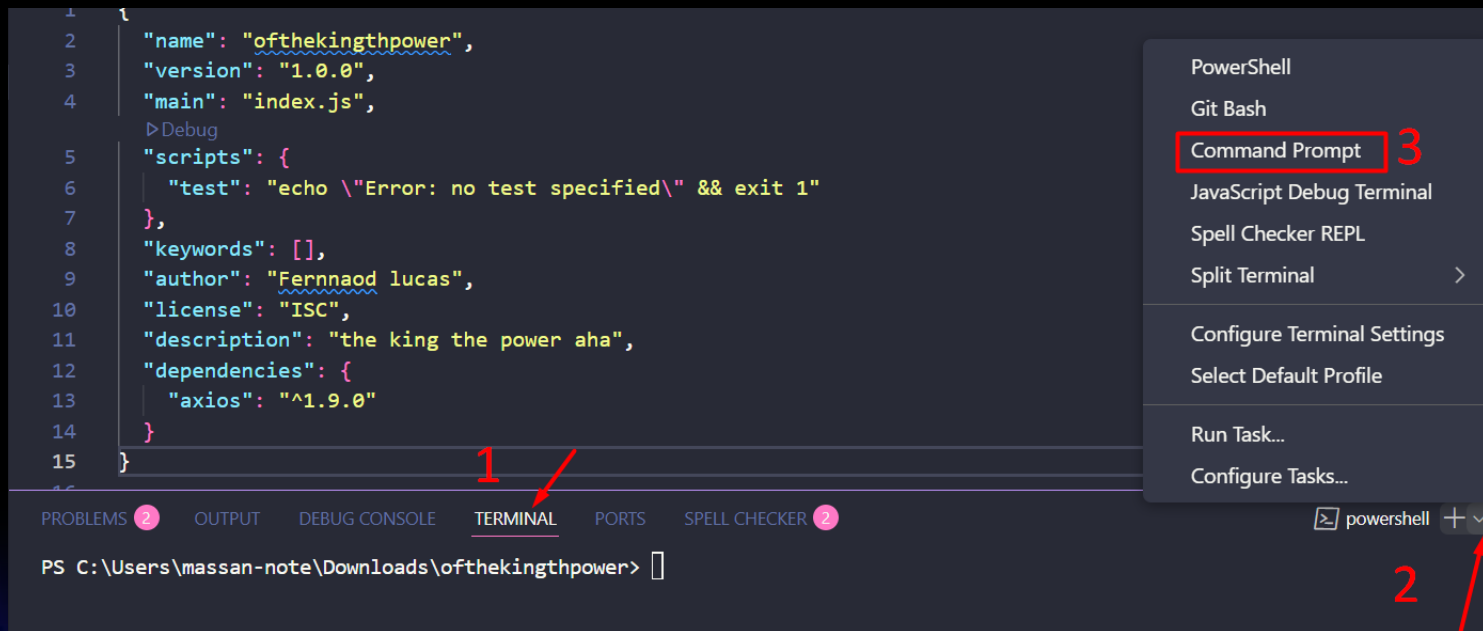
CD 'NOME DA PASTA' – NAVEGA NAS PASTAS

MKDIR – CRIA UMA PASTA NOVA

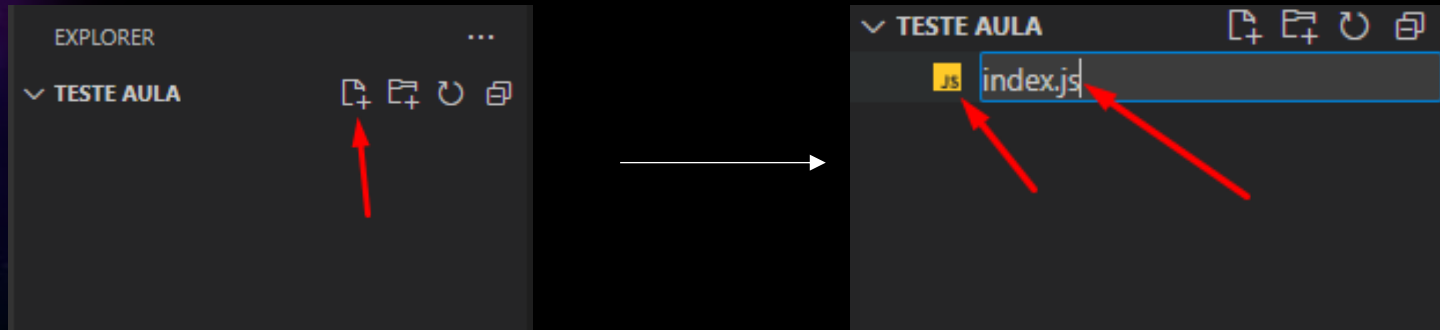
CODE . – ABRE O VSCODE NA PASTA QUE VOCÊ ESTÁ.

SEMPRE UTILIZAMOS O CMD PARA EXECUTAR NOSSOS PROJETOS. VAMOS CONTINUAR UTILIZANDO ELE MAS AGORA DENTRO DO VSCODE.

NO VSCODE APERTE **CTRL J PARA ABRIR O TERMINAL. CASO ELE VENHA COM O **POWERSHELL** VOCÊ DEVE ALTERAR PARA O **COMAND PROMPT (QUE É O CMD)**.**



CRIANDO ARQUIVOS EM JAVASCRIPT NO VSCODE.



CRIE UM NOVO ARQUIVO **.js.**

ESCOLHA UM NOME QUE POR PADRÃO É INDEX, E REPREARE QUE O QUE VOCÊ COLOCA DEPOIS DO '.' É QUE DEFINE O TIPO DO ARQUIVO.

PODERIA SER:

INDEX.HTML****

INDEX.CSS****

INDEX..JS****

INDEX.PKG****

ENTRE OUTRAS 1000 LINGUAGENS QUE EXISTEM.

CRIANDO UMA CONSTANTE E MOSTRANDO ELA NO CONSOLE.

```
JS teste.js > ...  
1  const menssage = "Hello World"  
2  console.log(menssage)
```

**COPIE NO SEU NOVO
ARQUIVO.JS**

**CTRL J
PARA ABRIR O CONSOLE**

**VERIFIQUE SE VOCÊ ESTA
NA PASTA CORRETA.**



**AGORA NO CONSOLE ESCREVA.
O NOME DO MOTOR (espaço) O NOME DO ARQUIVO QUE VOCÊ QUER
EXECUTAR.**

```
JS teste.js
```

VARIÁVEIS E TIPOS DE DADOS

**VARIÁVEIS SÃO ESPAÇOS NA MEMÓRIA PARA GUARDAR DADOS.
EXEMPLOS:**

LET

= VALOR QUE PODE MUDAR.

CONST

**= VALOR CONSTANTE (NÃO PODE SER ALTERADO DURANTE A
EXECUÇÃO DO CÓDIGO).**

VAR

= FORMA ANTIGA (EVITAR)

IMAGINE AS VARIÁVEIS COMO CAIXAS PARA GUARDAR DADOS!

JÁ OS TIPOS DE DADOS NÃO O QUE A VARIÁVEL GUARDA E SIM COMO SE DEFINE O QUE ELA ESTÁ GUARDANDO.

EXEMPLO:

```
let exemplo1 = "1"
```

DECLARANDO QUE UM DADO SERÁ GUARDADO E QUE ELE PODE SER ALTERADO.

DECLARANDO O NOME DA VARIÁVEL PARA QUE POSSAMOS USA-LA DEPOIS. (ESSE NOME VOCÊ QUE ESCOLHE)

O TIPO DO DADO QUE ESTÁ SENDO GUARDADO. NESSE CASO UMA STRING OU SEJA UM TEXTO.

```
let exemplo2 = 1
```

O TIPO DO DADO QUE ESTÁ SENDO GUARDADO. NESSE CASO UM INT OU SEJA UM VALOR INTEIRO.

E QUAL A DIFERENÇA?

EXEMPLO DE DIFERENÇA

EXERCICIO:

Crie um arquivo chamado `exec1.js`
copie o seguinte código nele:

```
let exemplo1 = "1"  
let exemplo2 = 1  
console.log(exemplo1+exemplo1)  
Agora execute o código!
```

**O NOME DO MOTOR (espaço) O NOME DO ARQUIVO QUE VOCÊ QUER EXECUTAR.
O QUE ACONTECEU?**

AGORA EXECUTE DESSA OUTRA FORMA:

```
console.log(exemplo2+exemplo2)
```

O QUE ACONTECEU?



**VOCÊ NOTOU QUE DE ACORDO COM O TIPO DO DADO QUE A VARIÁVEL GUARDA
O RESULTADO PODE SER ALTERADO.**

AGORA VAMOS FALAR DOS:

TIPOS PRINCIPAIS



TIPOS DE DADOS

STRING

= **TEXTO** – **EXEMPLO:** `let exemplo1 = "1"`

NUMBER

= **VALOR INTEIRO OU DECIMAL** – **EXEMPLO:** `let exemplo2 = 1`

BOOLEAN

= **VERDADEIRO OU FALSO** – **EXEMPLO** - `let exemplo3 = true`
`let exemplo4 = false`

NULL

= **NULO, VAZIO PROPOSTALMENTE** – **EXEMPLO** - `let exemplo5 = null`

UNDEFINED

= **NÃO DEFINIDO** – **EXEMPLO** - `let exemplo6 = undefined`

OBJECT

ARRAY

FUNCTION

- **VEREMOS POSTERIORMENTE**

TEMPLATE STRINGS

JÁ ENTENDEMOS, QUE OS DADOS QUE ESTÃO ENTRE ASPAS SÃO STRINGS.

EXEMPLO: ' ISTO É UMA FRASE E NÃO REPRESENTA UM COMANDO. '

QUER DIZER QUE NÃO PODEMOS CHAMAR COMANDOS OU VARIÁVEIS DENTRO DESSA STRING OU FRASE.

VAMOS IMAGINAR QUE TEMOS A SEGUINTE SITUAÇÃO.

TEMOS ESSA A SEGUINTE VARIÁVEL:

```
let notaDoAluno1 = 7.4
```

IMAGINA QUE QUEREMOS UTILIZAR O VALOR DESSA VARIÁVEL DENTRO DE UMA FRASE.

MAS ANTES VOCÊ CONSEGUE RESPONDER ESSAS DUAS QUESTÕES.

QUAL O VALOR QUE ESSA VARIÁVEL `notaDoAluno1` ESTA GUARDANDO?
QUAL O TIPO DELA NO MOMENTO ATUAL?

TEMPLATESTRING

**ISSO MESMO O VALOR QUE ELA GUARDA É 7.4
E O TIPO DELA É NUMBER.**

AGORA CONTINUEMOS A SEGUINTE SITUAÇÃO.

QUEREMOS ESCREVER O VALOR DESSA VARIÁVEL DIRETO NA FRASE.

```
let notaDoAluno1 = 7.4  
console.log("a nota do aluno foi notaDoAluno1")
```

EXECUTE O CÓDIGO VERIFIQUE O QUE ACONTECEU



TEMPLATESTRING

NOTE QUE FALHAMOS MISERAVELMENTE.

TENTE ASSIM AGORA:

TROQUE AS ASPAS ""

POR APÓSTROFOS ``

```
console.log(`a nota do aluno foi notaDoAluno1`)
```

EXECUTE O CÓDIGO E VEJA MAGICA

AGORA PRECISAMOS ENTENDER COMO FUNCIONA A MATEMÁTICA NO CÓDIGO.

AGORA VAMOS FALAR DE:

OPERADORES E EXPRESSÕES



OPERADORES

OPERADORES SÃO SÍMBOLOS QUE REALIZAM OPERAÇÕES COM VARIÁVEIS E VALORES.

◆ PRINCIPAIS TIPOS DE OPERADORES EM JAVASCRIPT:



1. Operadores Aritméticos

Operador	Significado
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Resto da divisão
**	Exponenciação

OPERADORES

EXEMPLO 1: QUESTÃO SIMPLES (VISUAL)

```
console.log(5 + 3); // 8  
console.log(10 % 3); // 1
```

OPERADORES

EXEMPLO 2: USO PRÁTICO COMUM

```
let preco = 20;  
let quantidade = 3;  
let total = preco * quantidade;  
console.log(`Total a pagar: R$`, ${total}`);
```

OPERADORES



EXERCÍCIO

**CRIE DUAS VARIÁVEIS: NOTA1 E NOTA2.
CALCULE A MÉDIA DAS NOTAS E MOSTRE NO CONSOLE:
"A MÉDIA DO ALUNO FOI: X"**

OPERADORES

OPERADORES DE ATRIBUIÇÃO

◆ OPERADORES DE ATRIBUIÇÃO EM JAVASCRIPT:

Operador	Exemplo	Equivalente
=	<code>x = 5</code>	atribui 5 a x
+=	<code>x += 3</code>	<code>x = x + 3</code>
-=	<code>x -= 2</code>	<code>x = x - 2</code>
*=	<code>x *= 4</code>	<code>x = x * 4</code>
*=	<code>x *= 4</code>	<code>x = x * 4</code>

OPERADORES

EXEMPLO 1: QUESTÃO SIMPLES (VISUAL)

```
let a = 10;  
a += 5;  
console.log(a);
```

OPERADORES

EXEMPLO 2: USO PRÁTICO COMUM

```
let saldo = 100;  
saldo -= 20;  
console.log("Saldo atual:", saldo);
```


OPERADORES



EXERCÍCIO

**CRIE UMA VARIÁVEL ECONOMIAS COM VALOR 0.
DEPOIS, ADICIONE 50 DUAS VEZES E SUBTRAIA 30.
MOSTRE O VALOR FINAL.**

OPERADORES

OPERADORES DE COMPARAÇÃO

◆ OPERADORES DE COMPARAÇÃO EM JAVASCRIPT:

Operador	Significado
==	Igual (com conversão)
===	Igual (valor e tipo)
!=	Diferente (com conversão)
!==	Diferente (valor e tipo)
>	Maior que
<	Menor que
>=	Maior ou igual
<=	Menor ou igual

<=

Menor ou igual

OPERADORES

EXEMPLO 1: QUESTÃO SIMPLES (VISUAL)

```
console.log(5 == "5");  
console.log(5 === "5");
```

OPERADORES

EXEMPLO 2: USO PRÁTICO COMUM

```
let idade = 18;  
console.log(idade >= 18);
```

OPERADORES

EXEMPLO 3: USO PRÁTICO EM QUESTÕES

```
let senhaCorreta = "1234";  
let senhaDigitada = "1234";  
console.log(senhaCorreta === senhaDigitada);
```

OPERADORES




EXERCÍCIO

**CRIE UMA VARIÁVEL TEMPERATURA COM UM VALOR.
DEPOIS VERIFIQUE SE A TEMPERATURA ESTÁ ACIMA DE 30 GRAUS.
MOSTRE ESTÁ QUENTE? TRUE/FALSE.**

CASO VOCÊ QUEIRA SABER O TIPO DE VARIÁVEL QUE VOCÊ CRIOU EXISTE UM COMANDO PARA ISSO -

```
console.log(typeof notaAluno01)
```



**MOSTRA
NO CONSOLE**

**MOSTRA
O TIPO**

**O QUE
VAI SER
TESTADO**

IMPRIMA O COMANDO `console.log(typeof` 🕒🕒🕒🕒)

7 – PODEMOS ESCREVER **COMENTÁRIOS**, QUE SÃO PALAVRAS QUE NÃO SERÃO INTERPRETADAS COMO CÓDIGOS OU COMANDOS, SERVE PRINCIPALMENTE PARA INFORMAR ALGO.

```
// Criar um programa que calcula a média  
// das notas entre os alunos e envia  
// mensagem do cálculo da média.
```



```
const aluno01 = "Mayk"  
const aluno02 = 'Diego'  
const notaAluno01 = 9.8
```

9 – CRIANDO UMA CALCULADORA DE MEDIA DOS ALUNOS.

**CRIE TRÊS VARIÁVEIS DO TIPO
NUMBER PARA GUARDAR AS
NOTAS.**

CRIE UMA VARIÁVEL PARA GUARDAR O CÁLCULO DE MEDIA.

MOSTRE TUDO ISSO NO CONSOLE !

CRIANDO UMA CALCULADORA DE MÉDIA DOS ALUNOS.

```
const notaAluno01 = 9.8  
const notaAluno02 = 10  
const notaAluno03 = 2
```

**CRIE TRÊS VARIÁVEIS DO TIPO
NUMBER PARA GUARDAR AS
NOTAS.**

```
const media = (notaAluno01 + notaAluno02 + notaAluno03) / 3
```

CRIE UMA VARIÁVEL PARA GUARDAR O CÁLCULO DE MEDIA.

```
console.log(media)
```

MOSTRE TUDO ISSO NO CONSOLE !

```
node index.js
```

AGORA VAMOS PRO UPGRADE

UTILIZE UMA ESTRUTURA CONDICIONAL PARA PARABENIZAR A TURMA COM UMA MENSAGEM NO CONSOLE **SE A MÉDIA FOR MAIOR QUE 5 E SE NÃO FOR JÁ SABE.**

DICA

```
if ( ) {  
    // faz alguma coisa  
} else {  
    // faz outra coisa  
}
```

```
IF (VARIABLE > 5){  
    CONSOLE.LOG ( ----- ) }  
  
else{  
    Console.log ( -----) }
```

DESAFIO

UTILIZE UMA TEMPLATE STRING PARA MOSTRAR A VARIÁVEL MEDIA DENTRO DA MENSAGEM.

AGORA VAMOS PRO UPGRADE

UTILIZE UMA ESTRUTURA CONDICIONAL PARA PARABENIZAR A TURMA COM UMA MENSAGEM NO CONSOLE SE A MÉDIA FOR MAIOR QUE 5 E SE NÃO FOR JÁ SABE.

DESAFIO

UTILIZE UMA TEMPLATE STRING PARA MOSTRAR A VARIÁVEL MEDIA DENTRO DA MENSAGEM.

```
if (media > 5) {  
    console.log(`A nota foi de ${media}. Parabéns`)  
} else {  
    console.log('A média é menor que 5')  
}
```

AGORA VAMOS PRO UPGRADE

DESAFIOS !

```
console.log( 5 > 4 ) // true
console.log( 5 < 4 ) // false
console.log( 5 >= 4 ) // true
console.log( 4 <= 4 ) // true
```

OPERADORES DE COMPARAÇÃO

>	Maior
<	Menor
>=	Maior igual a
<=	Menor igual a
==	Igual a
===	Igual e do mesmo tipo
!=	Diferente de
!==	Diferente, inclusive do tipo

```
console.log( 4 == "4" )
console.log( 4 === "4" )
console.log( 4 != "5" )
console.log( 4 !== "5" )
```

```
// DESAFIO 1
// dar bonificação de 1.000
// se o vendedor possuir mais que 100 pontos
```

```
// DESAFIO 2
// verificar se a pessoa é maior de 18 anos
// se sim, deixar entrar, se não, bloquear a entrada
// se a pessoa tiver 17 anos
// avisar para voltar quando fizer 18 anos
```

QUE A FORÇA ESTEJA COM VOCÊ.
PROF. FERNANDO LUCAS

OPERADORES LÓGICOS

OPERADORES DE LÓGICOS

&& "E" As duas condições devem ser verdadeiras para que a condição final seja verdadeira.
|| "OU" Uma das condições deve ser verdadeira para que a condição final seja verdadeira.
! "NÃO" Nega uma condição

```
console.log(5 == 5 && 6 == 6) // true  
console.log(5 == 5 && 6 != 6) // false
```

```
console.log(5 != 5 || 6 == 6) // true  
console.log(5 == 5 || 6 != 6) // true
```

```
console.log(!(5 > 6)) // true  
console.log(!(5 < 6)) // false
```

→ **MELHORE O SEU CÓDIGO**

↓

```
// DESAFIO 1  
// se sim, deixar entrar, se não, bloquear a entrada  
// se a pessoa tiver 17 anos  
// avisar para voltar quando fizer 18 anos
```


OBJETOS :3

IMAGINE, SEU CADASTRO NA ESCOLA É ENCONTRADO NO SISTEMA PELO SEU NOME.

**UMA ALUNA CHAMADA LAIS
NO SISTEMA DA ESCOLA TEM VÁRIAS INFORMAÇÕES COMO:**

ENDEREÇO

TELEFONE

NOTA

SALA

TURNO

**AGORA IMAGINE SE O SISTEMA TIVESSE QUE CRIAR UMA VARIÁVEL
PARA CADA INFORMAÇÃO DA LAIS.**

VAMOS CONVERTER ESSA SITUAÇÃO PRO JAVASCRIPT.

OBJETOS :3

ENDEREÇO

TELEFONE

NOTA

SALA

TURNO

**AGORA IMAGINE SE O SISTEMA TIVESSE QUE CRIAR UMA VARIÁVEL
PARA CADA INFORMAÇÃO DA LAIS.**

VAMOS CONVERTER ESSA SITUAÇÃO PRO JAVASCRIPT.

```
const nome = lais;  
const enderecoDaLais = "rua kennedy n4"  
const notaFinal = 9.8  
const salaDaLais = 5  
const turnoDaLais = "manha"
```

OBJETOS :3

```
const nome = lais;  
const enderecoDaLais = "rua kennedy n4"  
const notaFinal = 9.8  
const salaDaLais = 5  
const turnoDaLais = "manha"
```

BOM, PERCEBEMOS QUE LOTAMOS NOSSO PROGRAMA DE VARIÁVEIS E ISSO CERTAMENTE NÃO É BOM.

MAS SOLUCIONAMOS NOSSO PROBLEMA É CLARO.

E SE TIVESSE MAIS ALUNOS NA ESCOLA?

JÁ IMAGINOU QUANTAS VARIÁVEIS TERÍAMOS QUE CRIAR ?

```
const nome = lais;
```

```
const enderecoDaLais
```

```
const nome = lais;
```

```
const nome = lai
```

```
const enderecoDa
```

```
const notaFinal
```

```
const salaDaLais = 5
```

```
const turnoDaLais = "manha"
```

```
const nome = lais;
```

```
daLais = "rua kennedy n4"
```

```
= 9.8
```

```
lais;
```

```
coDaLais = "rua kennedy n4"
```

```
nal = 9.8
```

```
const salaDaLais = 5
```

```
const turnoDaLais = "manha"
```

```
"rua kennedy n4"
```

```
a"
```

OBJETOS :3

CERTAMENTE ISSO NÃO SERIA **GOOD** PRA GENTE.
PRA ISSO NOS USAMOS UM MÉTODO CHAMADO DE **OBJETO**.
E SE PUDÉSSEMOS GUARDAR TODOS OS DADOS DA LAIS DENTRO DE UMA VARIÁVEL SÓ? E MELHOR E SE PUDÉSSEMOS CHAMAR CADA INFORMAÇÃO SOMENTE CHAMANDO A VARIÁVEL LAIS.

ANTES NÓS PENSÁVAMOS EM VARIÁVEIS COMO UMA PEQUENA CAIXA.
AGORA VAMOS PENSAR EM OBJETOS COMO UMA PEQUENA CÔMODA CHEIA DE GAVETAS.

ESTRUTURA DO OBJETO EM JAVASCRIPT:

```
1  const lais = {  
2      |  endereco: "rua kennedy n4",  
3      |  notaFinal: 9.8,  
4      |  sala: 5,  
5      |  turno: "manha"  
6  }  
7  console.log (lais.notaFinal)
```

OBJETOS :3

AGORA VAI FICAR **GOOD, POSSO CHAMAR SOMENTE
A INFORMAÇÃO QUE EU QUERO.
CHAME NO CONSOLE SOMENTE A INFORMAÇÃO QUE VOCÊ QUER IMPRIMIR.**

```
1  const lais = {  
2      endereco: "rua kennedy n4",  
3      notaFinal: 9.8,  
4      sala: 5,  
5      turno: "manha"  
6  }  
7  console.log (lais.endereco)
```

```
1  const lais = {  
2      endereco: "rua kennedy n4",  
3      notaFinal: 9.8,  
4      sala: 5,  
5      turno: "manha"  
6  }  
7  console.log (lais.notaFinal)
```

**TUDO QUE ESTÁ DENTRO DO OBJETO CHAMAMOS DE
PROPRIEDADES.**

```
1  const lais = {  
2      endereco: "rua kennedy n4",  
3      notaFinal: 9.8,  
4      sala: 5,  
5      turno: "manha"  
6  }  
7  console.log (lais.endereco)
```

**USAMOS '{}' PARA INICIAR O OBJETO
USAMOS A ',' PARA SEPARAR AS PROPRIEDADES
USAMOS O '.' PARA ACESSAR AS PROPRIEDADES**

OBJETOS :3

NA SIMPLICIDADE, AGORA CRIE VÁRIOS OBJETOS ENVOLVENDO AS MESMAS PROPRIEDADES PARA TODOS.

```
1  const lais = {  
2    endereco: "rua kennedy n4",  
3    notaFinal: 9.8,  
4    sala: 5,  
5    turno: "manha"  
6  }
```

OBJETOS :3

UMA CURIOSIDADE, ESTAMOS USANDO O COMANDO CONSOLE.LOG ()

E ESSE DANADO É LITERALMENTE UM OBJETO.

CONSOLE.LOG ()

OBJETO JAVASCRIPT

**. PARA CHAMAR UMA
PROPRIEDADE.**

**A PROPRIEDADE
ESCOLHIDA.**

ARRAYS O/

**AINDA NO SENTIDO DE DIMINUIR NOSSO NÚMERO DE VARIÁVEIS
TEMOS OS**

ARRAYS

(VETORES)

**QUE SÃO UMA FORMA DE ARMAZENAR TODOS OS OBJETOS EM
UMA VARIÁVEL.**

DECLARANDO DESSA FORMA....

ARRAYS O/

VAMOS CRIAR UM PROGRAMA QUE GUARDA AS INFORMAÇÕES DOS ALUNOS DE UMA DETERMINADA ESCOLA.

```
const escola = []
```

→ **PRIMEIRO CRIAMOS O **ARRAY** E DAMOS UM NOME A ELE, SÃO OS **[]** QUE VÃO FAZER COM QUE NOSSA VARIÁVEL SE TORNE UM **ARRAY**.**

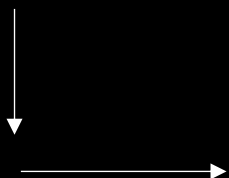
```
const escola = [{}, {}, {}, {}]
```

→ **DENTRO DE CADA **{}** HAVERÁ UM OBJETO COM SUAS PROPRIEDADES.**

ARRAYS O/

AGORA TEMOS QUE ENTENDER QUE CADA OBJETO GANHOU
UMA **POSIÇÃO** NO NOSSO **ARRAY**.
COMEÇANDO A CONTAR DO **NUMERO 0**.

```
const escola = [{}, {}, {}, {}]
```



```
const escola = [  
  {  
    nome: "lais",  
    endereco: "rua kennedy n4",  
    notaFinal: 9.8,  
    sala: 5,  
    turno: "manha"},  
  {  
    nome: "pedro teles",  
    endereco: "rua amanha tem copa",  
    notaFinal: 9.8,  
    sala: 5,  
    turno: "manha"},  
  {  
    nome: "savio",  
    endereco: "rua kenya west ",  
    notaFinal: 9.8,  
    sala: 5,  
    turno: "manha"},  
]
```

0

1

2

ARRAYS O/

AGORA COM O NOSSO COMANDO GOD DOS TESTES O
CONSOLE.LOG ().
VAMOS A PARTIR DA POSIÇÃO DO **ARRAY** IMPRIMIR AS
INFORMAÇÕES DO PEDRO.

```
const escola = [
```

```
{  
  nome:"lais",  
  endereco: "rua kennedy n4",  
  notaFinal: 9.8,  
  sala: 5,  
  turno: "manha"},
```

0

```
{  
  nome:"pedro teles",  
  endereco: "rua amanha tem copa",  
  notaFinal: 9.8,  
  sala: 5,  
  turno: "manha"},
```

1

```
{  
  nome:"savio",  
  endereco: "rua keny west ",  
  notaFinal: 9.8,  
  sala: 5,  
  turno: "manha"},
```

2

```
console.log(escola[1].endereco,escola[1].nome)
```

E ASSIM CONSEGUIMOS CHAMAR AS
INFORMAÇÕES QUE QUEREMOS.

ARRAYS O/

AGORA É SUA VEZ !!!!!

CRIE 2 ARRAYS CHAMADOS DE
ALUNOSDATURMAA
ALUNOSDATURMAB

2 – CADA POSIÇÃO UM OBJETO COM 3 PROPRIEDADES.

NOME

NOTA

ENDEREÇO

3 – IMPRIMA NA TELA TUDO QUE ESTIVER NA TERCEIRA
POSIÇÃO DO ARRAY.

FUNÇÕES 0/

PENSANDO EM REAPROVEITAR NOSSOS CÓDIGOS

FUNÇÃO

**ELA PERMITE QUE POSSAMOS REAPROVEITAR UM BLOCO DE
CÓDIGO VÁRIAS VEZES, SEM PRECISAR ESCREVE-LO NOVAMENTE.**

FUNÇÕES O/

ELA POSSUI DOIS MOMENTOS

1 — DECLARAR

2 - CHAMAR

FUNÇÕES O/

DECLARANDO

```
function soma (a,b){}
```

COMANDO

O NOME

OS PARÂMETROS
QUE SERÃO RECEBIDOS
DENTRO DA FUNÇÃO
E EXECUTADOS.

OS COMANDOS QUE
ELA DEVE EXECUTAR.

return

O RETORNO
DA FUNÇÃO.

FUNÇÕES O/

CHAMANDO

```
soma (5,6)
```

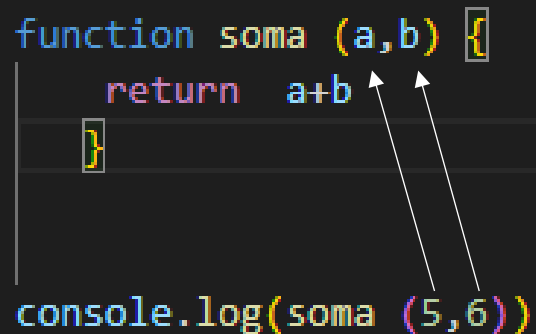
↓
NOME

↓
**OS PARÂMETROS
QUE SERÃO ENVIADOS PARA DENTRO DA FUNÇÃO.**

FUNÇÕES O/

EXEMPLO

```
function soma (a,b) {  
  return a+b  
}  
  
console.log(soma (5,6))
```

A diagram illustrating the process of passing arguments to a function. Two white arrows originate from the arguments '5' and '6' in the function call 'soma (5,6)' at the bottom. These arrows point upwards to the parameters 'a' and 'b' respectively in the function definition 'function soma (a,b) {' at the top. This visualizes how the values 5 and 6 are substituted for the variables a and b within the function's scope.

OS PARÂMETROS DA NOSSA FUNÇÃO SÃO A E B (a,b), QUANDO NÓS CHAMAMOS NOSSA FUNÇÃO PRECISAMOS ENTREGAR OS PARÂMETROS PARA SEREM SUBSTITUÍDOS DENTRO DELA, NESTE CASO É O (5,6).

TRY AI MAN !

EXEMPLO

```
function calculaMedia(alunos) {  
  return (alunos[0].nota + alunos[1].nota + alunos[2].nota) / 3  
}  
  
const media1 = calculaMedia(alunosDaTurmaA)  
const media2 = calculaMedia(alunosDaTurmaB)  
  
function enviaMensagem(media, turma) {  
  // Se a média for maior que 5, parabenizar a turma  
  if (media > 5) {  
    console.log(`A media da turma ${turma} foi de ${media}. Parabéns`)  
  } else {  
    console.log(`A média da turma ${turma} é menor que 5`)  
  }  
}  
  
enviaMensagem(media1, 'turmaA')  
enviaMensagem(media2, 'turmaB')
```

ESTRUTURA DE REPETIÇÃO O/

PENSANDO EM REPETIR COMANDOS

ESTRUTURA DE REPETIÇÃO

- FOR -

**ELA PERMITE QUE POSSAMOS REPETIR O MESMO CÓDIGO
VÁRIAS VEZES.**

ESTRUTURA DE REPETIÇÃO 0/

NOVA CONFIGURAÇÃO DE VARIÁVEL DESBLOQUEADA

LET

DIFERENTE DA CONFIGURAÇÃO **CONST**, O **LET** CONSEGUE ALTERAR SEU VALOR A QUALQUER MOMENTO DO PROGRAMA.

```
const qualquerNome = 5
```



```
let qualquerNome = 5
```

ESTRUTURA DE REPETIÇÃO O/

EXEMPLO

```
for (let i = 0 ; i < 9; i++){  
  console.log(i)  
}
```

O COMANDO
i++
É A MESMA
COISA DE i + 1.

O **FOR** FUNCIONA COMO UM CONTADOR, ENQUANTO A CONDIÇÃO FOR VERDADEIRA ELE VAI CONTINUAR RODANDO O CÓDIGO.
E SEMPRE QUE ELE EXECUTAR O CÓDIGO ELE VAI ALTERAR O VALOR DA VARIÁVEL DE CONTROLE, NO NOSSO CASO (i).

ENTÃO SEMPRE QUE O CÓDIGO É EXECUTADO É ADICIONADO +1 A VARIÁVEL **LET** (i).

TRY AI MAN !

ESTRUTURA DE REPETIÇÃO O/

EXEMPLO

COM ARRAYS

NO LUGAR DE ESCREVERMOS UMA QUANTIDADE ESPECIFICA PARA O CONTADOR, PODEMOS PEDIR QUE ELE CONTE AS POSIÇÕES DO ARRAY DE FORMA AUTOMÁTICA COM A PROPRIEDADE **LENGTH**.

```
for (let i = 0; i < alunos.length; i++) {  
  | console.log(i); |  
}
```

TRY AI MAN !

ESTRUTURA DE REPETIÇÃO 0/

EXEMPLO

COM ARRAYS

CONSTRUA UMA **FUNÇÃO COM UM "FOR"** QUE PERCORRE E CALCULA A MÉDIA DE TODOS OS ALUNOS DAS TURMAS QUE VOCÊ CRIOU.
DEPOIS IMPRIMA ESSES VALORES NO CONSOLE.
DESAFIO – TENTE UTILIZAR TEMPLATES STRINGS DENTRO DAS IMPRESSÕES.

ESTRUTURA DE REPETIÇÃO O/

```
function calculaMedia(alunos) {  
  let soma = 0;  
  for (let i = 0; i < alunos.length; i++) {  
    soma = soma + alunos[i].nota  
  }  
  
  const media = soma / alunos.length  
  return media  
}
```

```
const media1 = calculaMedia(alunosDaTurmaA)  
const media2 = calculaMedia(alunosDaTurmaB)
```

```
function enviaMensagem(media, turma) {  
  // Se a média for maior que 5, parabenizar a turma  
  if (media > 5) {  
    console.log(`A media da turma ${turma} foi de ${media}. Parabéns`)  
  } else {  
    console.log(`A média da turma ${turma} é menor que 5`)  
  }  
}
```

```
}
```

```
enviaMensagem(media1, 'turmaA')  
enviaMensagem(media2, 'turmaB')
```