

# Dia 1

**Missão: Pedir o nome e a idade do usuário e escrever essas informações de uma forma amigável.**

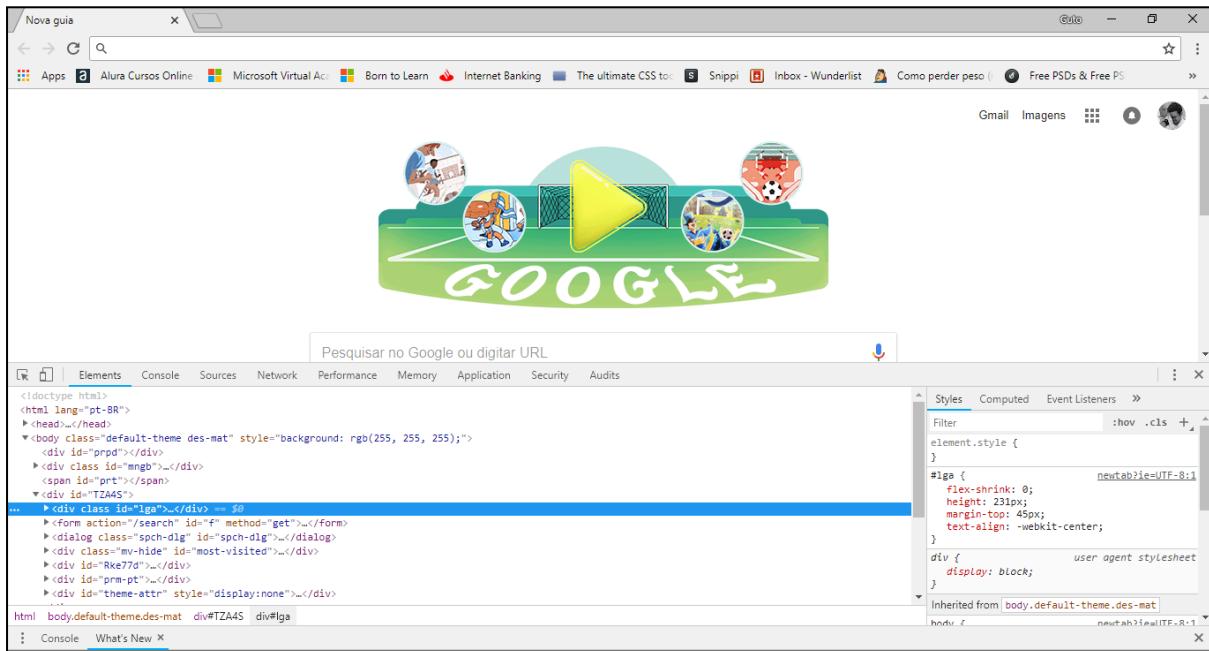
## Objetivo:

- Conhecer o console
- Trabalhar comandos de entrada e saída de dados do JavaScript

**Ao som de: Mumford & Sons - Hopeless Wanderer**

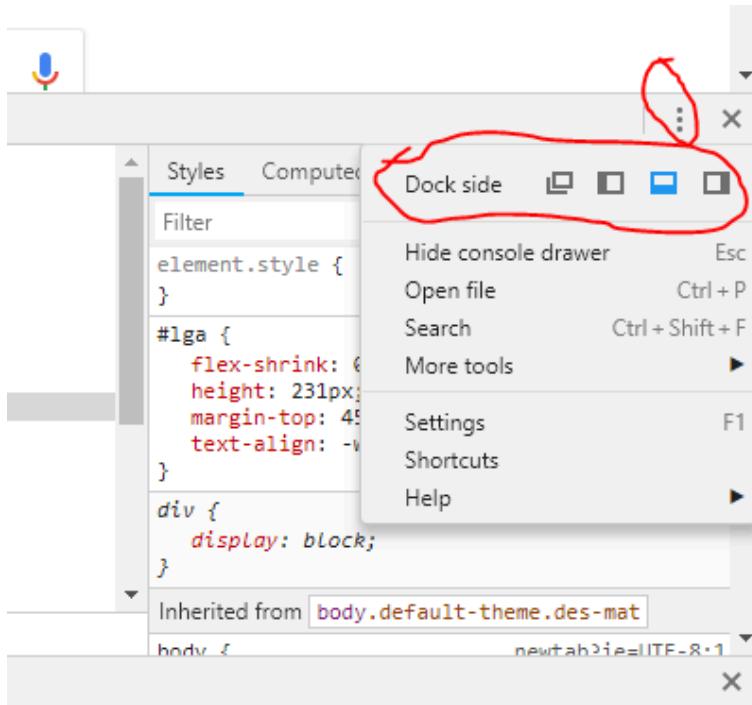
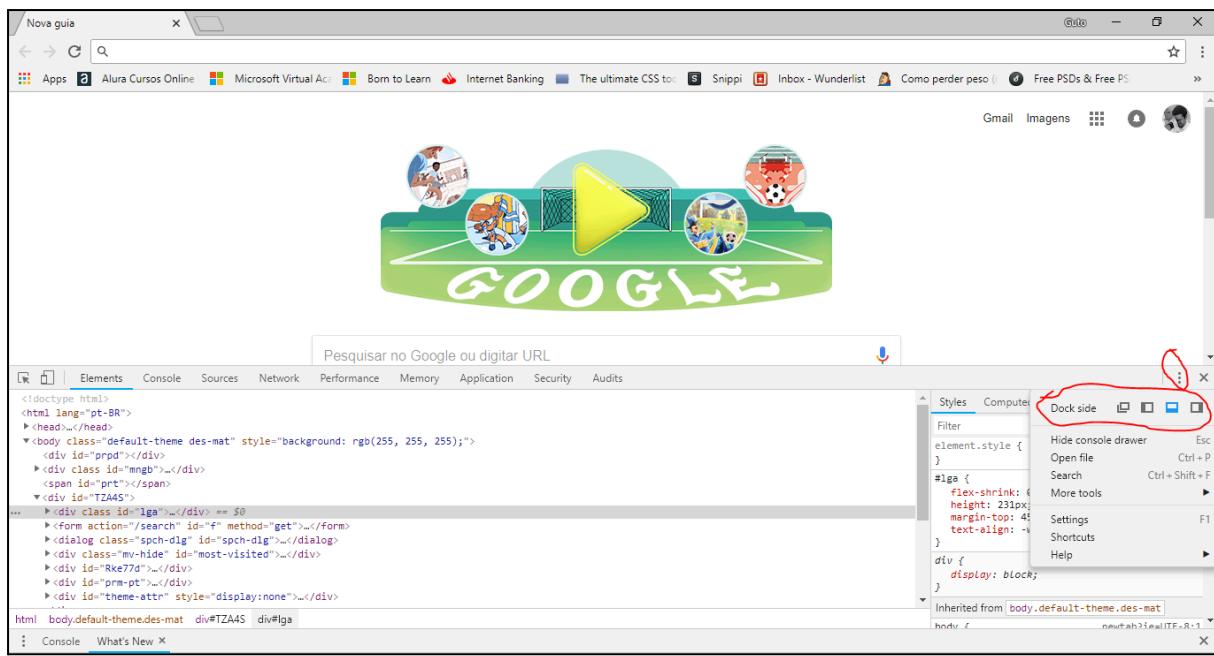
## Descrição:

Abra o navegador **Google Chrome**, com o navegador aberto, aperte a tecla **F12** (do teclado). Irá aparecer algo parecido com a imagem a seguir:

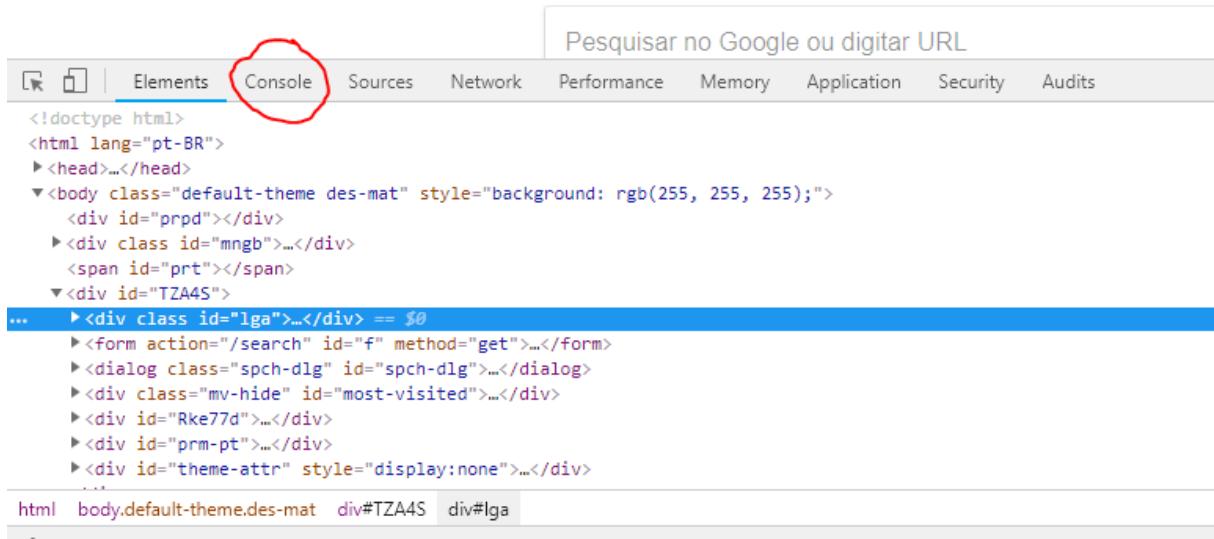


Esse painel é chamado de **Web Dev Tools**.

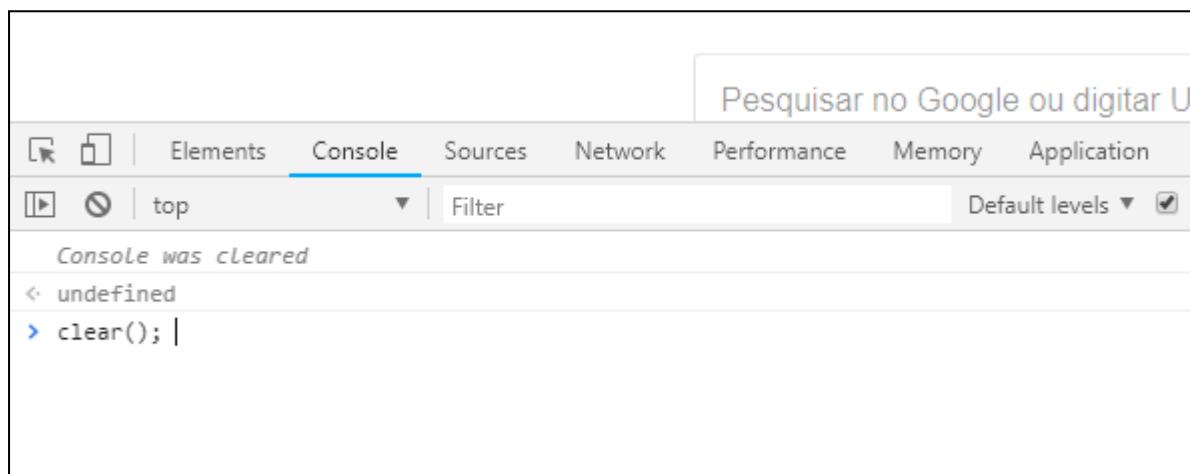
**Observação:** Caso a Web Dev Tools não esteja do lado de baixo e você queira deixar ela assim você pode clicar nos **3 pontos** no fim dela e escolher como ela vai ficar:



Click na aba **Console** do Web Dev Tools:



No console digite o comando `clear();` depois pressione enter para executar o comando.  
O comando `clear();` irá limpar o console:

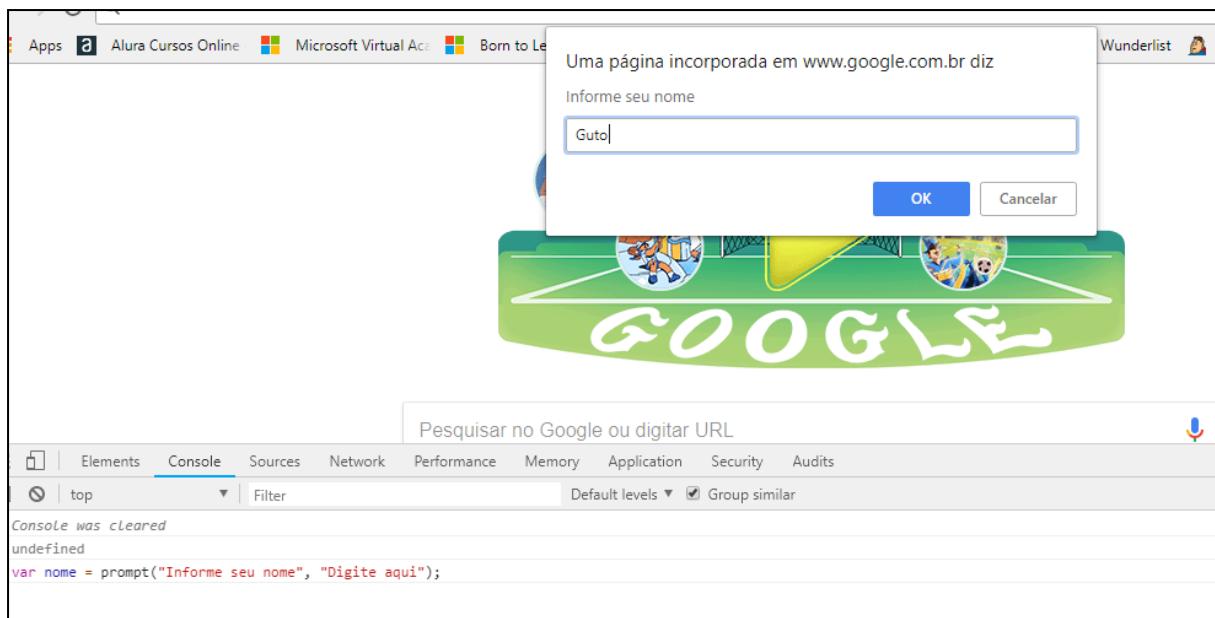


Sempre que quiser limpar o console você pode executar o comando `clear`.

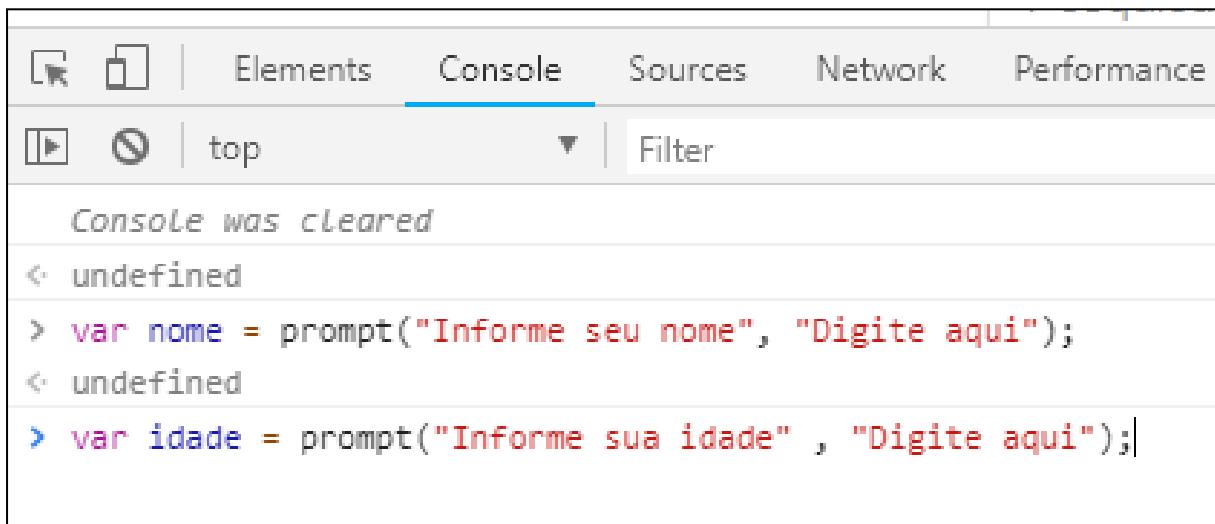
## Comandos de entrada de dados

Digite os comandos abaixo no console e pressione enter

```
var nome = prompt("Informe seu nome" , " Digite aqui ");
```



O comando `prompt` produz essa caixa para que o usuário possa entrar com informações. No caso, foi criado uma variável chamada `nome` e o valor informado pelo usuário será armazenado nesta variável.  
Continue com o código



## Exibindo os dados

Após ter pedido o usuário as informações nome e idade, chegou o momento de exibir essas informações. Digite os seguintes códigos:

```
console.log("Seu nome é");
console.log(nome);
```

O comando `console.log()` exibe no console do navegador o que estiver entre aspas (" ") ou o conteúdo da variável, no caso a variável `nome`. Ele está exibindo em duas linhas, caso você queira exibir tudo em uma linha só pode executar o seguinte comando.

```
console.log("Seu nome é " + nome);
```

E que tal melhorar a mensagem para completar o desafio:

```
console.log("Seja bem vindo " + nome + "!!! Você tem " + idade + " anos.");
```

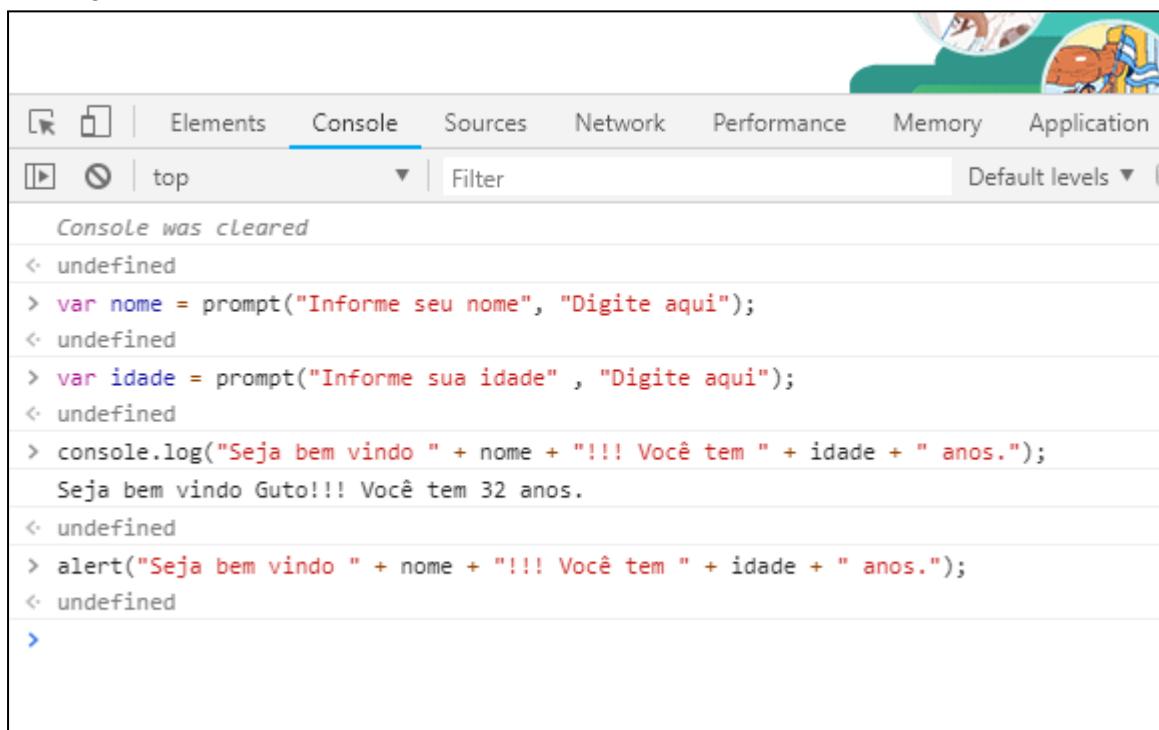
Pronto, o desafio está completo, mas como experimento, troque o comando `console.log()` pelo comando `alert()`.

```
alert("Seja bem vindo " + nome + "!!! Você tem " + idade + " anos.");
```



O comando `alert()` produz uma caixa de mensagem similar a do comando `prompt()`, com a diferença de que a do `alert` é apenas para saída de dados.

O código inteiro ficou assim:



Faça experimentos em cima desse exemplo, incremente ele pedindo outras informações

como data de nascimento ou endereço e mude suas mensagens e formas de exibição.

# Dia 2

## Missão: Realizar contas no console.

### Objetivo:

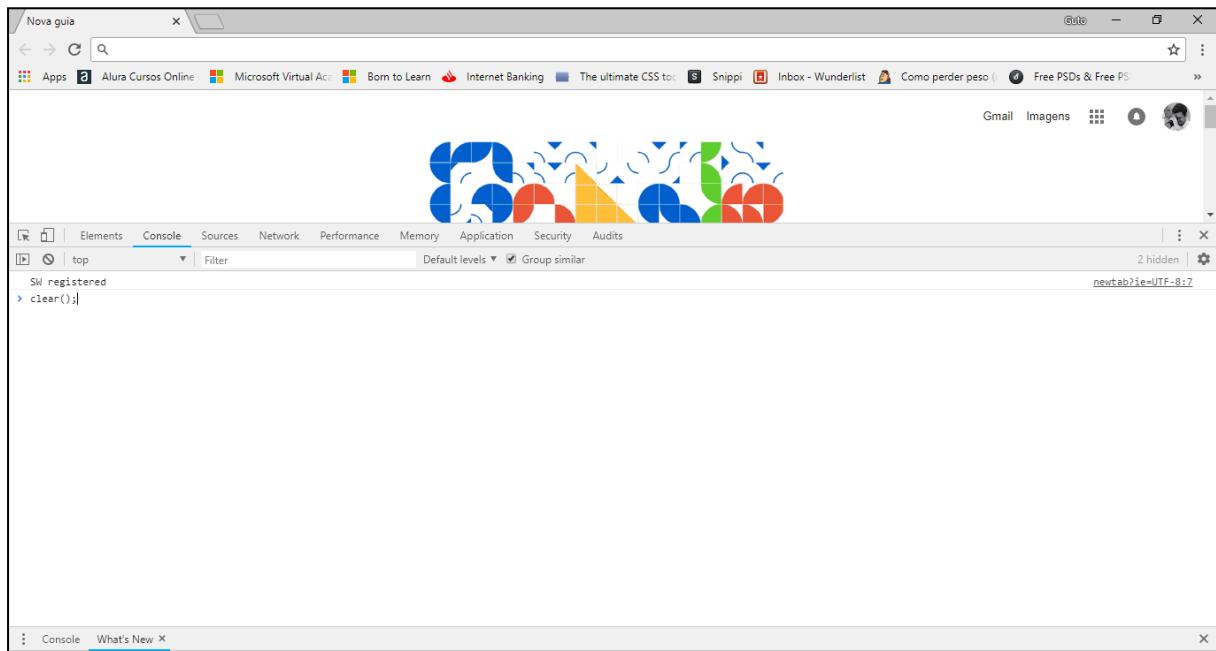
- Trabalhar com operadores matemáticos em Javascript

**Ao som de:** Imagine Dragons - It's Time

### Descrição:

Caso ainda não tenha feito a missão do dia 1, volte e realize ela.

Abra o console do Google Chrome, e já execute o comando para limpar o console:



### Operadores aritméticos

O console permite que você realize contas, execute os comandos abaixo e veja os resultados.

A screenshot of a browser's developer tools console tab. The console shows the following history of operations:

```
Console was cleared
< undefined
> 10+3
< 13
> 10/2
< 5
> 3*3
< 9
> 20-5
< 15
> |
```

Esses são os operadores aritméticos mais comuns:

- + → adição
- → subtração
- / → Divisão
- \* → Multiplicação

Além desses operadores temos mais alguns como:

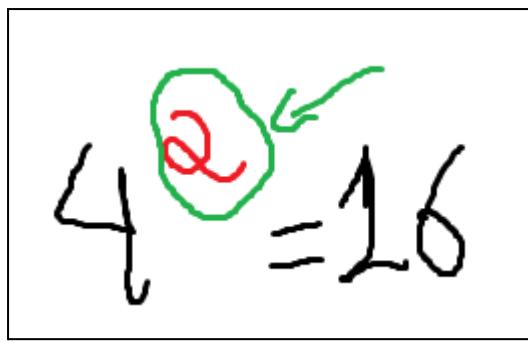
% → módulo (pega o resto da divisão e não o resultado)

A handwritten long division problem:

$$\begin{array}{r} 4 \longdiv{3} \\ - 3 \quad 1,333\dots \\ \hline 1 \end{array}$$

The quotient is 0. The remainder is circled in red at the bottom left.

\*\* → exponenciação (um número elevado a outro)



Execute os comandos abaixo como exemplo:

```
Elements Console So  
top | Fil  
  
Console was cleared  
< undefined  
> 4%3  
< 1  
> 4**2  
< 16  
> |
```

## Operadores relacionais

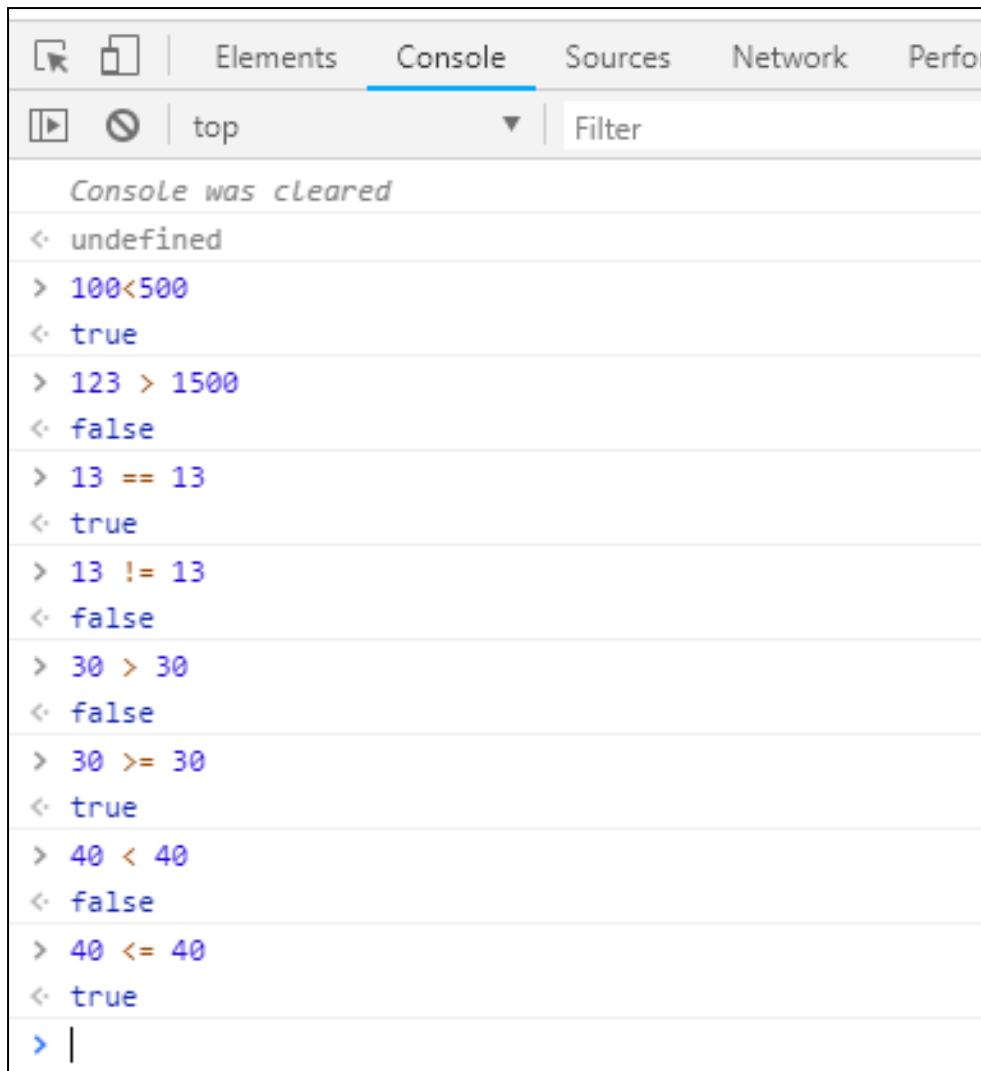
Operações utilizando os operadores relacionais retornam sempre true ou false, execute os comandos abaixo para ver um exemplo:

```
Elements Co  
top |  
  
Console was cleared  
< undefined  
> 10>5  
< true  
> 10<5  
< false  
> |
```

Os operadores relacionais mais utilizados são:

- $\text{==}$  → verifica a igualdade entre dois valores
- $\text{!=}$  → verifica se dois valores são diferentes
- $>$  → verifica se um valor é maior que outro
- $<$  → verifica se um valor é menor que outro
- $\geq$  → verifica se um valor é maior ou igual a outro
- $\leq$  → verifica se um valor é menor ou igual a outro

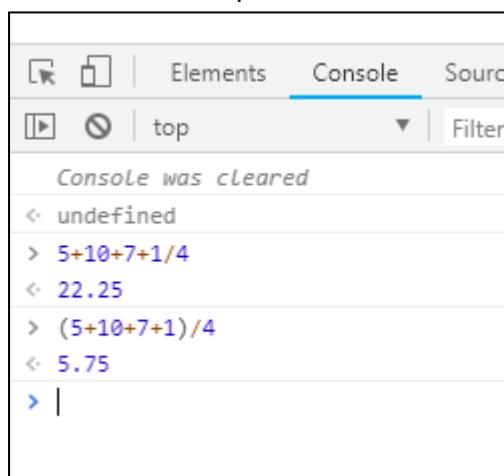
Digite os exemplos abaixo para ver os operadores funcionando:



The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The console output area displays the following sequence of operations and results:

```
Console was cleared
< undefined
> 100<500
< true
> 123 > 1500
< false
> 13 == 13
< true
> 13 != 13
< false
> 30 > 30
< false
> 30 >= 30
< true
> 40 < 40
< false
> 40 <= 40
< true
> |
```

Agora que você sabe que tem uma calculadora no seu navegador crie suas contas e exemplos para praticar, crie contas mais complexas para ver o comportamento e resultado das contas. Exemplo:



The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The console output area displays the following calculation:

```
Console was cleared
< undefined
> 5+10+7+1/4
< 22.25
> (5+10+7+1)/4
< 5.75
> |
```

Lembre-se que dentro da matemática existe a precedência de operadores (click [aqui](#) e [aqui](#) para ler mais sobre isso) e o parênteses quebra essa precedência.

Mais uma missão concluída, para aprender mais e ver quais outros operadores existem  
consulte a [documentação da Mozilla](#) que ajuda muito.

# Dia 3

**Missão: Calcular a média de venda à partir dos valores vendidos em cada um dos meses do 1º Trim de 2024 de um vendedor**

## Objetivo:

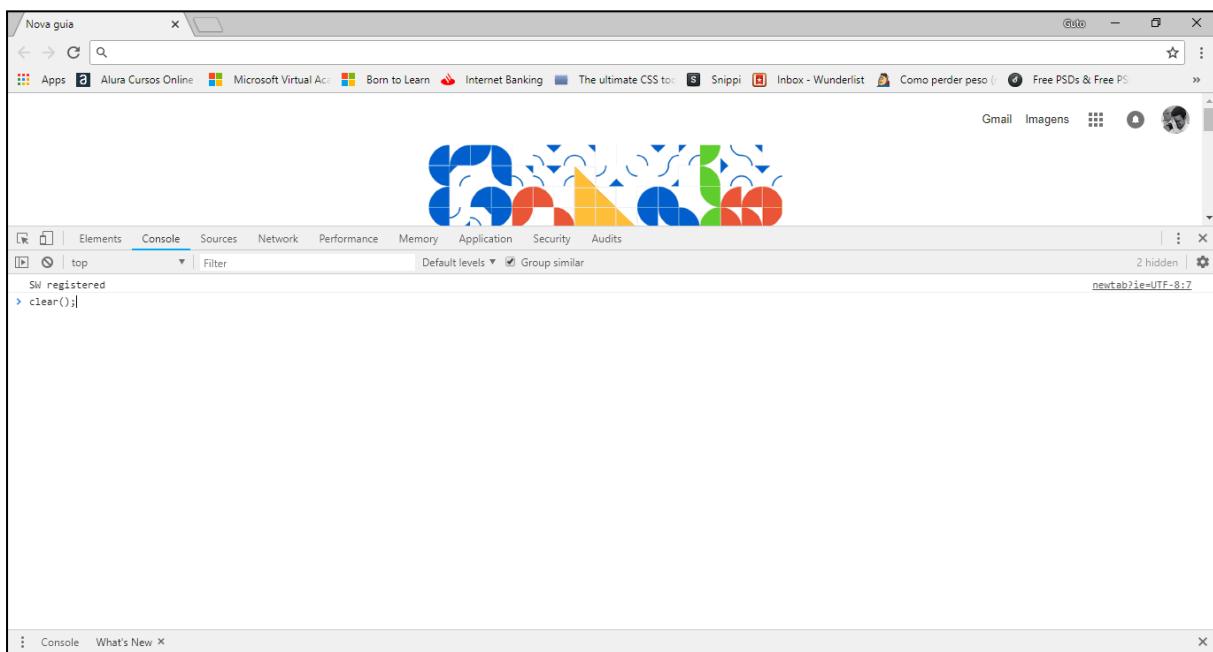
- Declarar variáveis
- Conversão de valores

**Ao som de: The Verve - Bitter Sweet Symphony**

## Descrição:

**Caso ainda não tenha feito a missão do dia 2, volte e realize ela.**

Abra o console do Google Chrome, e já execute o comando para limpar o console:

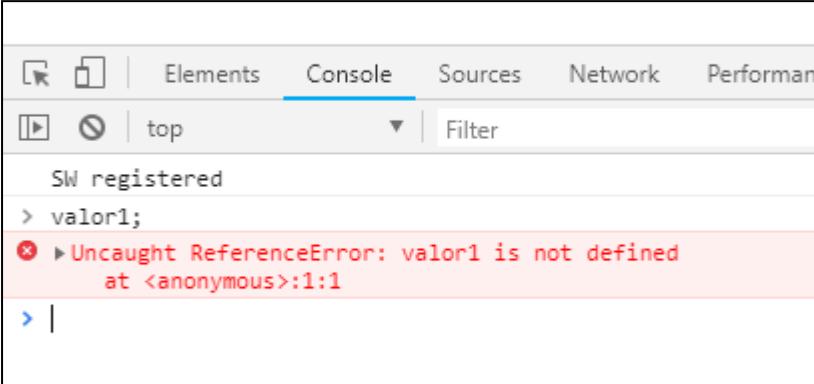


## Variáveis

No Javascript a declaração de variáveis não é obrigatória, porém muito recomendada. Digite o código abaixo para entender:

```
valor1;
```

Ao digitar `valor1;` será exibido um erro dizendo que *não está definido*.



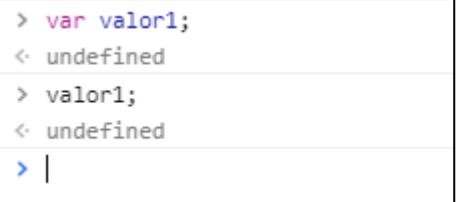
The screenshot shows a browser's developer tools console tab labeled "Console". The console output is as follows:

```
SW registered
> valor1;
✖ Uncaught ReferenceError: valor1 is not defined
    at <anonymous>:1:1
> |
```

Esse erro acontece porque não declaramos essa variável. Execute o comando abaixo para declarar a variável `valor1`:

```
var valor1;
```

E agora digite novamente o nome da variável no console:  
`valor1;`



The screenshot shows a browser's developer tools console tab labeled "Console". The console output is as follows:

```
> var valor1;
< undefined
> valor1;
< undefined
> |
```

Dessa vez não existiu o erro, porém foi informado a mensagem `undefined` que significa que essa variável está indefinida, sem um valor. Vamos atribuir um valor à essa variável:

```
valor1 = 50;
```

E agora vamos digitar o nome da variável novamente no console:

```
valor1;
```

```
> valor1 = 50;
< 50
> valor1;
< 50
>
```

Observe que agora, após de você digitar apenas o nome da variável foi exibido o valor 50. O operador de **atribuição** é o **= (igual)**, quando você utiliza o igual ele pega o que tiver do lado direito e atribui ao que estiver ao lado esquerdo do operador (Ex: `valor1 = 50`).

Voltando à declaração de variáveis, podemos declarar uma variável apenas a iniciando com um valor. Digite:

```
valor2 = 100;
valor2;
```

Ou mesmo se lembrar da primeira missão onde temos a seguinte linha:

```
var nome = prompt("Informe seu nome" , " Digite aqui ");
```

Nessa linha estamos declarando a variável e já inserindo um valor dentro dela, o que também é possível. **Afim de boas práticas recomenda-se sempre utilizar o var para definir suas variáveis, juntas ou separadas.** Exemplo:

```
var valor1;
var valor2;
```

OU

```
var valor1, valor2;
```

OU

```
var valor1 = 50;
var valor2 = 100;
```

## Missão

Declare as variáveis que serão utilizadas:

```
var vendedor, jan, fev, mar, media;
```

Peça as informações e armazene nas variáveis:

```
vendedor = prompt("Informe o nome do vendedor", "Digite aqui");
jan = prompt("Informe o valor das vendas de janeiro", "Digite aqui");
fev = prompt("Informe o valor das vendas de fevereiro", "Digite aqui");
mar = prompt("Informe o valor das vendas de março", "Digite aqui");
```

Calcule a média, lembrando que o cálculo da média é  $(jan + fev + mar)/3$ :

```
media = ( parseFloat(jan) + parseFloat(fev) + parseFloat(mar) )/3
```

Quando você entra com dados utilizando o comando `prompt` esses valores são armazenados como `string(texto)`, o comando `parseFloat()` converte os valores para ponto flutuante (número decimal);

Exiba as informações com uma mensagem amigável:

```
console.log("A média de vendas do " + vendedor + " no 1º trim foi " +
media);
alert("A média de vendas do " + vendedor + " no 1º trim foi " + media);
```

O código inteiro fica assim:

The screenshot shows a browser's developer tools console interface. At the top, there are buttons for play/pause, stop, and refresh, followed by a dropdown menu set to 'top' and a 'Filter' input field. To the right are 'Default levels' and a checked checkbox for 'Group similar'. The main area displays the following interaction:

```
Console was cleared
< undefined
> var vendedor, jan, fev, mar, media;
< undefined
> vendedor = prompt("Informe o nome do vendedor", "Digite aqui");
< "Guto"
> jan = prompt("Informe o valor das vendas de janeiro", "Digite aqui");
< "10000"
> fev = prompt("Informe o valor das vendas de fevereiro", "Digite aqui");
< "7500"
> mar = prompt("Informe o valor das vendas de março", "Digite aqui");
< "8000"
> media = ( parseFloat(jan) + parseFloat(fev) + parseFloat(mar) )/3
< 8500
> console.log("A média de vendas do " + vendedor + " no 1º trim foi " + media);
A média de vendas do Guto no 1º trim foi 8500
< undefined
> alert("A média de vendas do " + vendedor + " no 1º trim foi " + media);
< undefined
> |
```

Bem com isso mude o exemplo, exiba o total de vendas do primeiro trimestre, peça informações referentes ao segundo trimestre também, crie seus exemplos.

# Dia 4

## Missão: Validar se um candidato pode ou não iniciar o processo de CNH

### Objetivo:

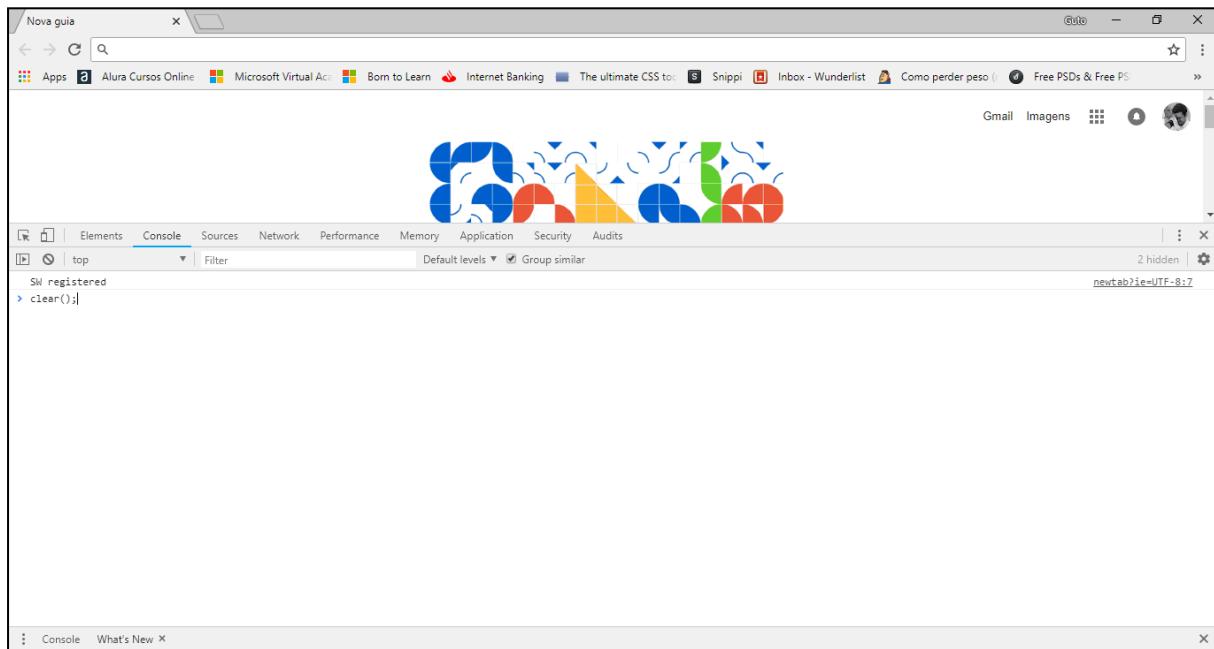
- Estrutura de decisão if (se)

**Ao som de:** Dwayne Johnson - You're Welcome

### Descrição:

Caso ainda não tenha feito a missão do dia 3, volte e realize ela.

Abra o console do Google Chrome, e já execute o comando para limpar o console:



### Estrutura de decisão

No Javascript temos algumas estruturas de decisão, nesse momento vamos conhecer o **if** (se). Digite os códigos abaixo para testar:

The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The console output is as follows:

```
Console was cleared
< undefined
> var numero = 7;
< undefined
> if(numero >= 10){
    alert("Este número é maior ou igual à 10");
}else{
    alert("Este número é menor do que 10");
}
< undefined
>
```

**Observação: para digitar como está no if utilize shift + enter para mudar de linha e o comando não ser executado.**

O comando if tem a seguinte estrutura:

```
if ( condição ) {
    instruções se a condição for verdadeiro
}else{
    instruções se a condição for falsa
}
```

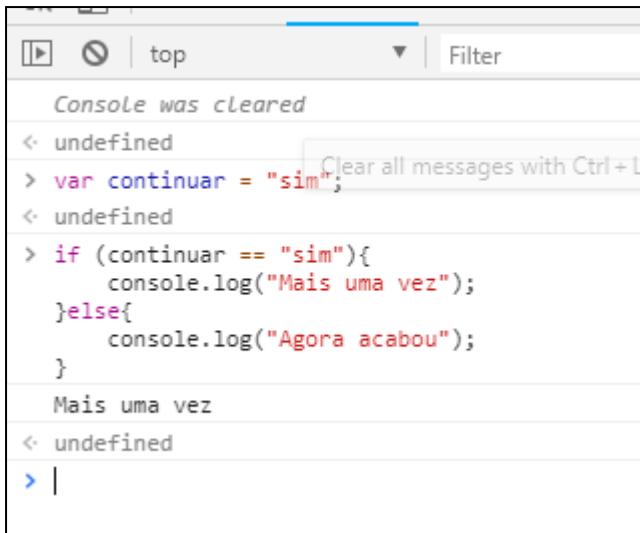
No código de exemplo como 7 é menor do que 10 será executado os comandos contidos no bloco else.

Digite esse outro exemplo:

The screenshot shows the Chrome DevTools interface with the 'Console' tab selected. The console output is as follows:

```
Console was cleared
< undefined
> var continuar = "sim";
< undefined
> if (continuar == "sim"){
    console.log("Mais uma vez");
}else{
    console.log("Agora acabou");
}
```

Nesse exemplo será exibida a mensagem “Mais uma vez” pois será executado os comandos no bloco verdadeiro do if pois a variável continuar possui um valor igual à “sim”.



```
Console was cleared
< undefined
> var continuar = "sim"; Clear all messages with Ctrl + L
< undefined
> if (continuar == "sim"){
    console.log("Mais uma vez");
}else{
    console.log("Agora acabou");
}
Mais uma vez
< undefined
> |
```

## Missão

Vamos primeiro pedir que o usuário digite o nome do candidato e a idade do candidato:

```
var nome = prompt("Informe o nome do candidato" , "Digite aqui");
var idade = prompt("Informe a idade do candidato" , "Digite aqui");
```

Agora vamos verificar se o candidato está apto a iniciar o processo e informar o usuário disso:

```
if( parseInt(idade) >= 18 ){
    alert("O candidato " + nome + " está apto à iniciar o processo.");
}else{
    alert("O candidato " + nome + " não está apto à iniciar o
processo.");
}
```

Observação: o comando `parseInt()`; é utilizado para converter o conteúdo da variável para inteiro. Para relembrar mais sobre o assunto revisite a missão 3.

O código inteiro fica da seguinte forma:

```
Console was cleared
< undefined
> var nome = prompt("Informe o nome do candidato" , "Digite aqui");
< undefined
> var idade = prompt("Informe a idade do candidato" , "Digite aqui");
< undefined
> if( parseInt(idade) >=18 ){
    alert("O candidato " + nome + " está apto à iniciar o processo.");
} else{
    alert("O candidato " + nome + " não está apto à iniciar o processo.");
}
< undefined
> |
```

Missão concluída, agora crie seus exemplos como por exemplo perguntar para o usuário se ele é casado ou solteiro e personalizar a mensagem de acordo com a resposta.

# **Dia 5**

**Missão: Verificar se é dia, tarde ou noite e mostrar uma mensagem personalizada.**

**Objetivo:**

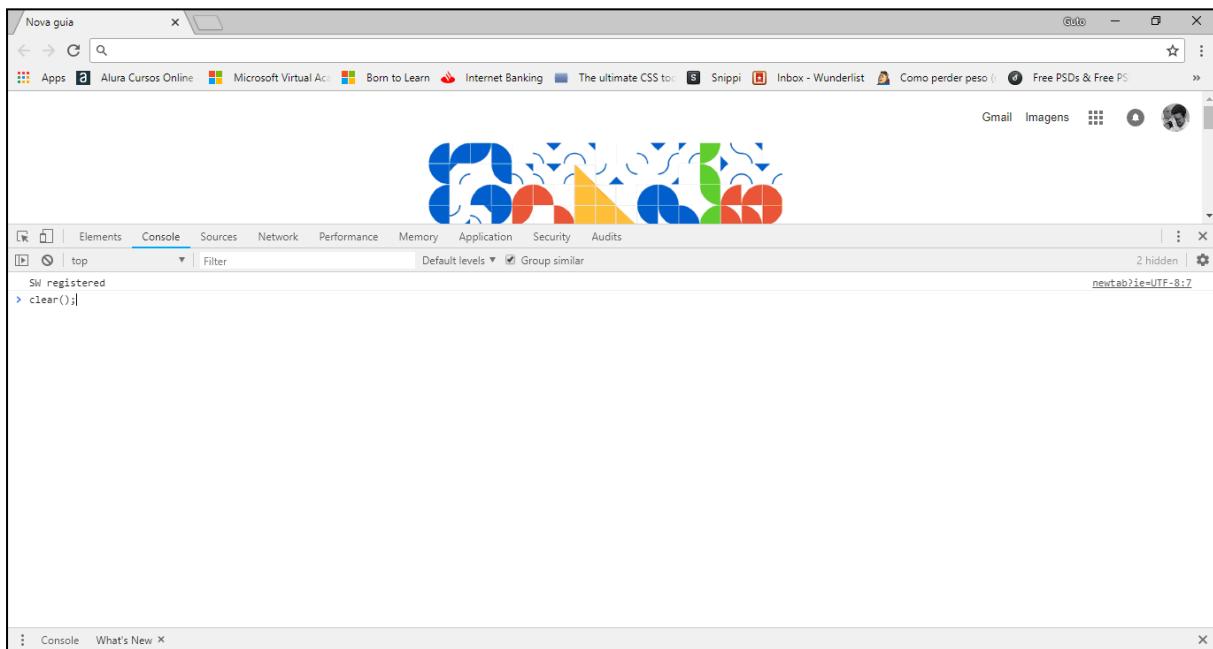
- Estrutura de decisão if (se)
- Objeto Date

**Ao som de: Keira Knightley - A Step You Can't Take Back**

**Descrição:**

**Caso ainda não tenha feito a missão do dia 4, volte e realize ela.**

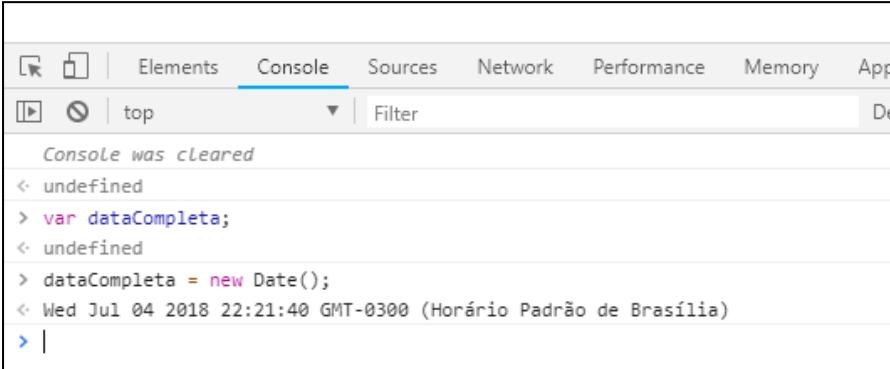
Abra o console do **Google Chrome**, e já execute o comando para limpar o console:



**Objeto Date**

Antes de tudo, precisamos declarar uma variável e descobrir a hora. Para isso iremos criar um objeto date. Digite o código abaixo:

```
var dataCompleta;  
dataCompleta = new Date;
```



A screenshot of a browser's developer tools console tab. The console shows the following interaction:

```
Console was cleared  
< undefined  
> var dataCompleta;  
< undefined  
> dataCompleta = new Date();  
< Wed Jul 04 2018 22:21:40 GMT-0300 (Horário Padrão de Brasília)  
> |
```

Na variável dataCompleta foi criado uma instância do objeto chamado Date. Através do objeto Date podemos pegar todas as informações da data completa como dia, mês, ano, hora, minuto, segundo. Digite os exemplos abaixo para ver o funcionamento:

```
dataCompleta.getDay(); // dia da semana em número  
dataCompleta.getDate(); // dia do mês  
dataCompleta.getMonth();  
dataCompleta.getFullYear();  
dataCompleta.getHours();  
dataCompleta.getMinutes();  
dataCompleta.getSeconds();
```

De posse dessas informações podemos solucionar nosso problema.

## Missão

Iremos personalizar a mensagem que iremos mostrar ao usuário, para isso iremos considerar que até antes do meio dia será “bom dia”, que até antes das 18 será “boa tarde” e antes da 00:00 será “boa noite”.

Criar uma variável para armazenar a data completa e uma variável para armazenar apenas à hora atual:

```
var dataCompleta;  
var horaAtual;
```

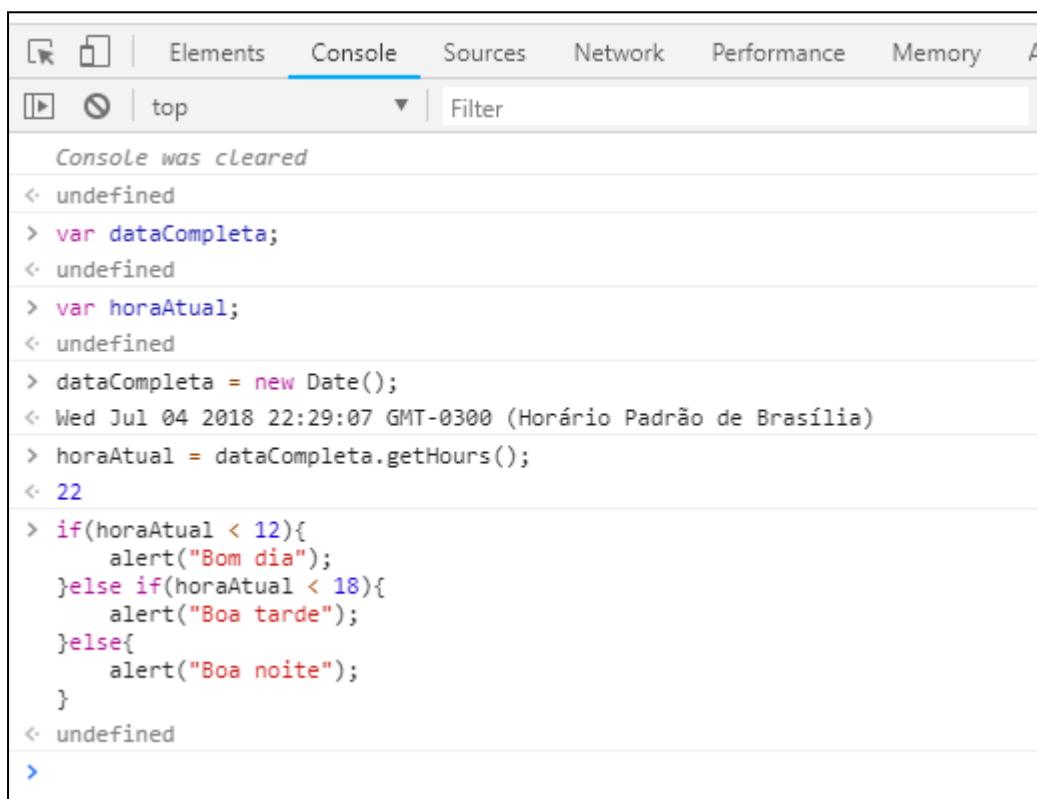
Agora vamos criar a instância do objeto Date e pegar a hora atual:

```
dataCompleta = new Date();
horaAtual = dataCompleta.getHours();
```

Neste momento precisamos apenas fazer a verificação para emitir a mensagem:

```
if(horaAtual < 12){
    alert("Bom dia");
} else if(horaAtual < 18){
    alert("Boa tarde");
} else{
    alert("Boa noite");
}
```

O código completo fica assim:



```
Console was cleared
< undefined
> var dataCompleta;
< undefined
> var horaAtual;
< undefined
> dataCompleta = new Date();
< Wed Jul 04 2018 22:29:07 GMT-0300 (Horário Padrão de Brasília)
> horaAtual = dataCompleta.getHours();
< 22
> if(horaAtual < 12){
    alert("Bom dia");
} else if(horaAtual < 18){
    alert("Boa tarde");
} else{
    alert("Boa noite");
}
< undefined
>
```

Crie seu exemplo agora, como por exemplo verificando o dia e dizendo se está na primeira quinzena do mês, se é o dia do pagamento (dia 5), etc.

# **Dia 6**

**Missão: Calcular as horas extras de um funcionário a partir das horas semanais trabalhadas. Considerar a carga horária do funcionário de 44 horas por semana.**

## **Objetivo:**

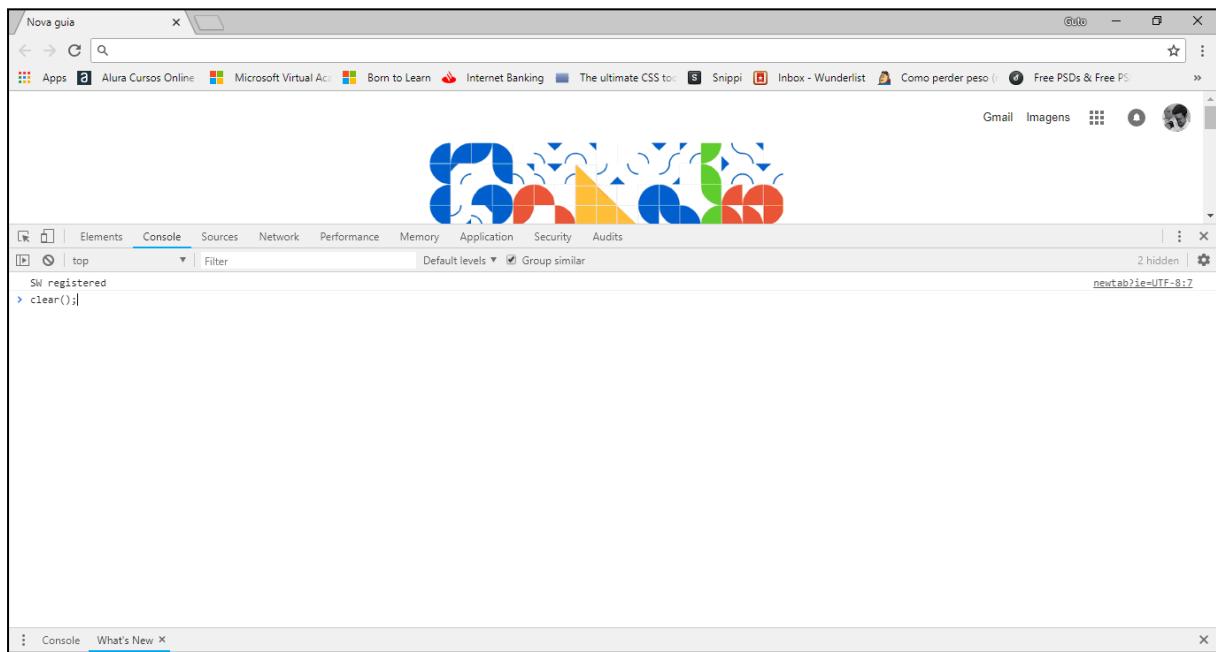
- Estrutura de decisão if (se)

**Ao som de: Natiruts - Dentro da Música II**

## **Descrição:**

**Caso ainda não tenha feito a missão do dia 5, volte e realize ela.**

Abra o console do **Google Chrome**, e já execute o comando para limpar o console:



## **Missão**

Iremos calcular em quantas horas o funcionário excedeu de sua carga horária semanal, as horas excedidas são as horas extras. Exemplo: O funcionário Pedro tem uma carga horária semanal de 44 horas, mas ele trabalhou 50 horas nesta semana, logo, Pedro fez 6 horas extras.

Para deixar mais completo nosso sistema, vamos verificar também se o funcionário está “devendo” horas. Por exemplo, se Pedro tem uma carga horária de 44 horas mas essa semana trabalhou 40, ele está com 4 horas negativas.

Vamos criar as variáveis para armazenar a carga horária semanal do funcionário, o saldo de horas, e, uma para armazenar as horas semanais trabalhadas. Essa última informação deve ser informado pelo usuário. Digite:

```
var cargaHorariaSemanal = 44;  
var saldoHoras = 0;  
var horasTrabalhadas = prompt("Informe o total de horas trabalhadas do  
funcionário nessa semana:", "Digite aqui");
```

Agora, vamos calcular o saldo de horas. Digite:

```
saldoHoras = horasTrabalhadas - cargaHorariaSemanal;
```

Agora, em cima do saldo de horas vamos fazer as verificações. Se o saldo de horas for positivo ( $>0$ ), significa que existem horas extras. Se for negativo ( $<0$ ), significa que o usuário trabalhou menos do que sua carga horária, e está em débito. Se nada disso acontecer significa que as horas trabalhadas e a carga horária semanal são iguais, por isso não existe nem horas extras nem horas em débito. Digite:

```
if(saldoHoras > 0){  
    alert("O funcionário fez " + saldoHoras + " hora(s) extra(s) nesta  
semana.");  
}else if(saldoHoras < 0){  
    alert("O funcionário está devendo " + (saldoHoras * -1) + " hora(s)  
nesta semana.");  
}else{  
    alert("O funcionário não possui hora extra essa semana");  
}
```

O código completo fica da seguinte forma:

The screenshot shows the 'Console' tab of a browser's developer tools. The console output is as follows:

```
Console was cleared
< undefined
> var cargaHorariaSemanal = 44;
< undefined
> var saldoHoras = 0;
< undefined
> var horasTrabalhadas = prompt("Informe o total de horas trabalhadas do funcionário nessa semana:","Digite aqui");
< undefined
> saldoHoras = horasTrabalhadas - cargaHorariaSemanal;
< 6
> if(saldoHoras > 0){
    alert("O funcionário fez " + saldoHoras + " hora(s) extra(s) nesta semana.");
} else if(saldoHoras < 0){
    alert("O funcionário está devendo " + (saldoHoras * -1) + " hora(s) nesta semana.");
} else{
    alert("O funcionário não possui hora extra essa semana");
}
< undefined
>
```

Observe o trecho destacado abaixo:

```
alert("O funcionário está devendo " + (saldoHoras * -1) + " hora(s) nesta semana.");
```

Porque foi utilizado a expressão `(saldoHoras * -1)` ?

Isso foi puramente estético, como nesse ponto o `saldoHoras` é negativo (pois o funcionário está devendo) eu multiplico por -1 para mostrar o `saldoHoras` sem o sinal de “-”. Veja como seria uma mensagem sem a expressão:

**O funcionário está devendo -10 hora(s) nesta semana.**

Veja como a mensagem fica estranha, você falar que ele está devendo menos dez. E agora depois de multiplicar por -1:

**O funcionário está devendo 10 hora(s) nesta semana.**

Bem, agora crie seus exemplos, faça por exemplo ao invés de utilizar a carga horária semanal utilize a carga horária mensal do funcionário, solicite o nome do funcionário, etc.

# Dia 7

## Missão: Exibir dia da semana por extenso.

### Objetivo:

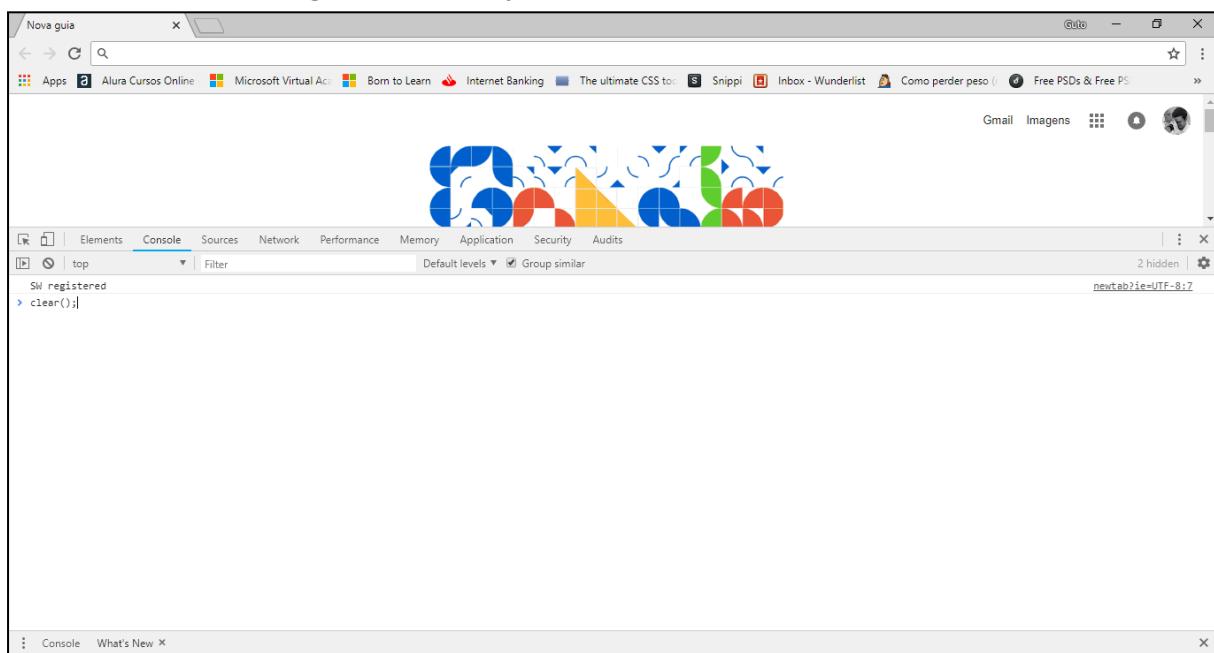
- Estrutura de decisão if (se)
- Estrutura de decisão switch (caso)

**Ao som de:** Vespas Mandarinas - Não Sei O Que Fazer Comigo

### Descrição:

**Caso ainda não tenha feito a missão do dia 6, volte e realize ela.**

Abra o console do **Google Chrome**, e já execute o comando para limpar o console:



### Estrutura de decisão switch

Nas missões anteriores utilizamos a estrutura de decisão if(se), porém ela não é a única estrutura de decisão existente. Vamos relembrar a estrutura de decisão if, digite:

```

var numero = prompt("Digite um número entre 1 e 3","Digite aqui");
if(numero == 1){
    console.log("Número 1");
}else if(numero == 2){
    console.log("Número 2");
}else if(numero == 3){
    console.log("Número 3");
}else{
    console.log("Opção inválida");
}

```

Para resolver a mesma situação poderíamos utilizar o comando switch. Em suma, a estrutura de controlo switch é um if para operar sobre a mesma variável ou expressão de entrada. Digite:

```

var numero = prompt("Digite um número entre 1 e 3","Digite aqui");

switch (numero) {
    case "1":
        console.log("Número 1");
        break;
    case "2":
        console.log("Número 2");
        break;
    case "3":
        console.log("Número 3");
        break;
    default:
        console.log("Opção inválida");
        break;
}

```

A instrução switch é similar a uma série de instruções IF sobre a mesma expressão. Em muitas ocasiões, você pode querer comparar a mesma variável (ou expressão) com muitos valores diferentes, executando uma peça diferente do código dependendo de qual valor ele se encaixar. Este é exatamente o que a instrução switch faz. Ao invés do bloco else para o resultado falso ele possui o bloco default que cumpre o mesmo objetivo.

## Missão

Nossa missão é verificar qual dia é hoje(da semana) e escrever por extenso. Exemplo: Hoje é segunda-feira.

Primeiro vamos criar variáveis para para armazenar a o objeto date, o dia da semana em número e o dia da semana por extenso. Digite:

```
var data = new Date();
var diaDaSemana = data.getDay();
var diaDaSemanaExtenso;
```

Agora vamos fazer a verificação para preencher a variável do dia da semana por extenso. Digite:

```
switch(diaDaSemana){
    case 1:
        diaDaSemanaExtenso = "domingo";
        break;
    case 2:
        diaDaSemanaExtenso = "segunda-feira";
        break;
    case 3:
        diaDaSemanaExtenso = "terça-feira";
        break;
    case 4:
        diaDaSemanaExtenso = "quarta-feira";
        break;
    case 5:
        diaDaSemanaExtenso = "quinta-feira";
        break;
    case 6:
        diaDaSemanaExtenso = "sexta-feira";
        break;
    default:
        diaDaSemanaExtenso = "segunda-feira";
        break;
}
```

Agora é personalizar a mensagem para o usuário. Digite:

```
console.log("Hoje é " + diaDaSemanaExtenso + ".");
```

Segue o código inteiro:

```
Console was cleared
< undefined
> var data = new Date();
< undefined
> var diaDaSemana = data.getDay();
< undefined
> var diaDaSemanaExtenso;
< undefined
> switch(diaDaSemana){
    case 1:
        diaDaSemanaExtenso = "domingo";
        break;
    case 2:
        diaDaSemanaExtenso = "segunda-feira";
        break;
    case 3:
        diaDaSemanaExtenso = "terça-feira";
        break;
    case 4:
        diaDaSemanaExtenso = "quarta-feira";
        break;
    case 5:
        diaDaSemanaExtenso = "quinta-feira";
        break;
    case 6:
        diaDaSemanaExtenso = "sexta-feira";
        break;
    default:
        diaDaSemanaExtenso = "segunda-feira";
        break;
}
< "quinta-feira"
> console.log("Hoje é " + diaDaSemanaExtenso + ".");
Hoje é quinta-feira.
< undefined
> |
```

Crie um outro exemplo agora, como por exemplo pegando o número do mês com getMonth(); e escrevendo em extenso. Exemplo: mês 1 = janeiro.

# Dia 8

## Missão: Efetuar uma das 4 operações matemáticas básicas.

### Objetivo:

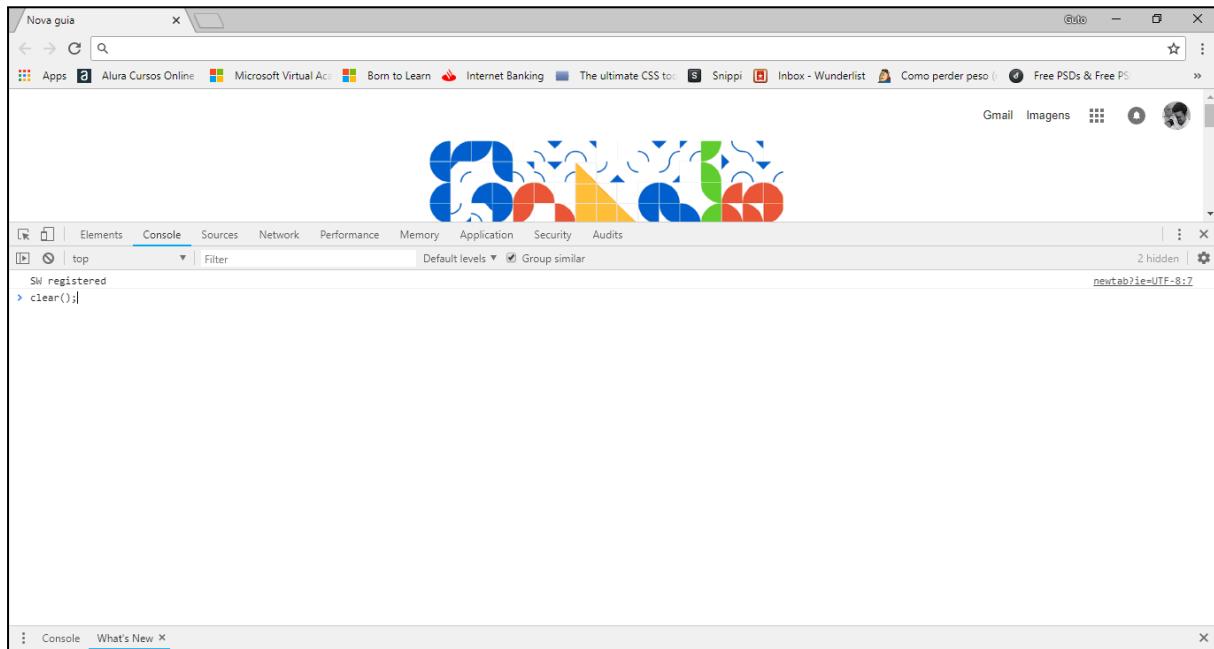
- Estrutura de decisão switch (caso)

**Ao som de:** Charlie Brown Jr - Céu Azul

### Descrição:

Caso ainda não tenha feito a missão do dia 7, volte e realize ela.

Abra o console do Google Chrome, e já execute o comando para limpar o console:



### Missão

Vamos efetuar um cálculo à partir de dois valores informados pelo usuário e de acordo com a operação escolhida pelo usuário. Vamos pedir os valores e a operação. Digite:

```
var numero1 = prompt("Informe o primeiro número","Digite aqui");
var numero2 = prompt("Informe o segundo número","Digite aqui");
var operacao = prompt("Informe o operador: \nAdição (+) \nSubtração (-)");
```

```
\nMultiplicação (*) \nDivisão(\\),"Digite aqui");
var resultado;
```

Agora vamos verificar qual operação foi escolhida e executar a operação. Digite:

```
switch(operacao){
    case "+":
        resultado = parseFloat(numero1) + parseFloat(numero2);
        break;
    case "-":
        resultado = parseFloat(numero1) - parseFloat(numero2);
        break;
    case "*":
        resultado = parseFloat(numero1) * parseFloat(numero2);
        break;
    case "/":
        resultado = parseFloat(numero1) / parseFloat(numero2);
        break;
    default:
        alert("Operação inválida");
        break;
}
```

Por final, vamos apresentar o resultado. Digite:

```
console.log("O resultado é " + resultado);
```

O código todo fica assim:

```
> var numero1 = prompt("Informe o primeiro número","Digite aqui");
< undefined
> var numero2 = prompt("Informe o segundo número","Digite aqui");
< undefined
> var operacao = prompt("Informe o operador: \nAdição (+) \nSubtração (-) \nMultiplicação (*)\nDivisão(\\)","Digite aqui");
< undefined
> var resultado;
< undefined
> switch(operacao){
    case "+":
        resultado = parseFloat(numero1) + parseFloat(numero2);
        break;
    case "-":
        resultado = parseFloat(numero1) - parseFloat(numero2);
        break;
    case "*":
        resultado = parseFloat(numero1) * parseFloat(numero2);
        break;
    case "/":
        resultado = parseFloat(numero1) / parseFloat(numero2);
        break;
    default:
        alert("Operação inválida");
        break;
}
< 12
> console.log("O resultado é " + resultado);
O resultado é 12
< undefined
>
```

VM76:1

Tente adicionar mais operações, como exponenciação.

# Dia 9

## Missão: Verificando confirmação para o usuário.

### Objetivo:

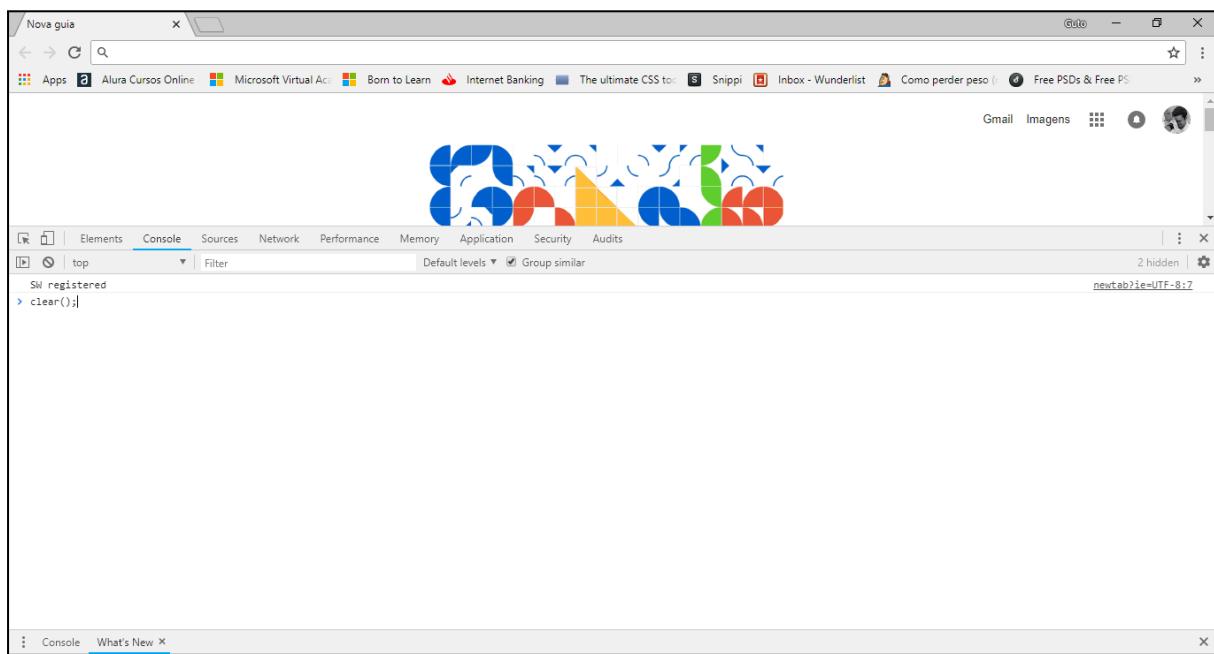
- Estrutura de decisão switch (caso)

**Ao som de:** Elton John - Rocket Man

### Descrição:

Caso ainda não tenha feito a missão do dia 8, volte e realize ela.

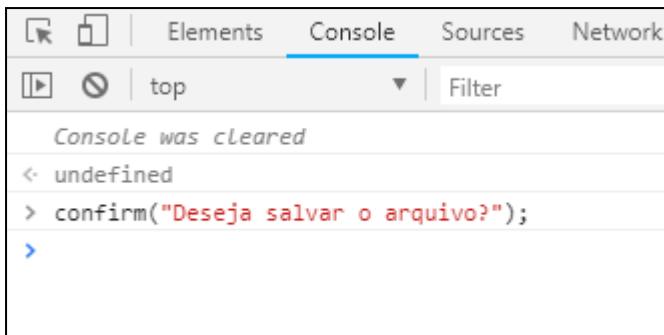
Abra o console do Google Chrome, e já execute o comando para limpar o console:



### Confirm

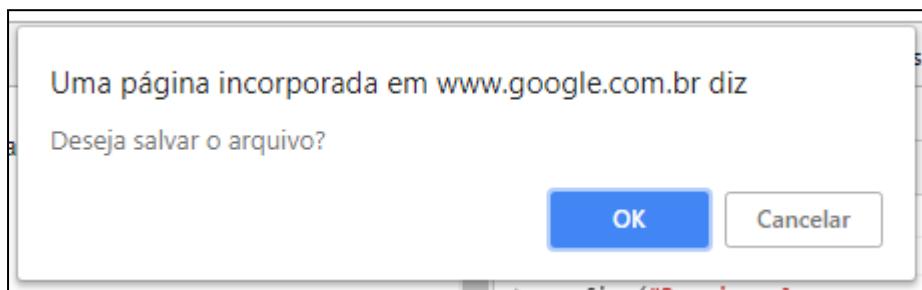
O comando `confirm` cria uma janela parecida com o `prompt`, porém nesta janela existem apenas dois botões: OK e Cancelar. Digite:

```
confirm("Deseja salvar o arquivo?");
```



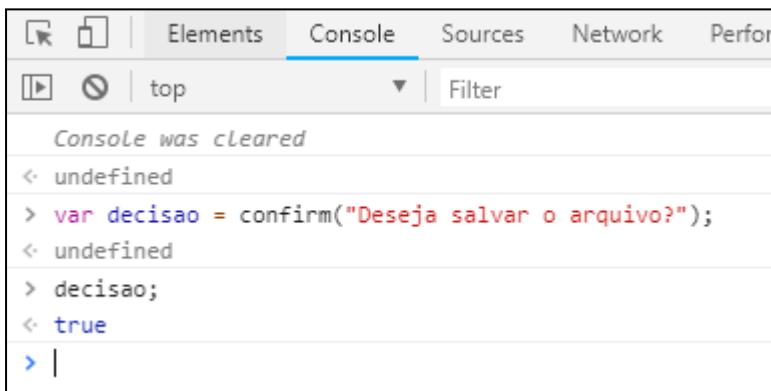
```
Elements Console Sources Network
▶ top Filter
Console was cleared
< undefined
> confirm("Deseja salvar o arquivo?");
>
```

A janela produzia será igual à esta:



Se você clicar no botão “OK” o retorno será **true**, se clicar no botão “Cancelar” o retorno será **false**. Podemos armazenar esse retorno em uma variável. Digite:

```
var decisao = confirm("Deseja salvar o arquivo?");
decisao;
```



```
Elements Console Sources Network Performance
▶ top Filter
Console was cleared
< undefined
> var decisao = confirm("Deseja salvar o arquivo?");
< undefined
> decisao;
< true
> |
```

## Missão

Vamos fazer uma pergunta para o usuário utilizando o comando `confirm` e de acordo com o botão que ele pressionar vamos mostrar uma mensagem.

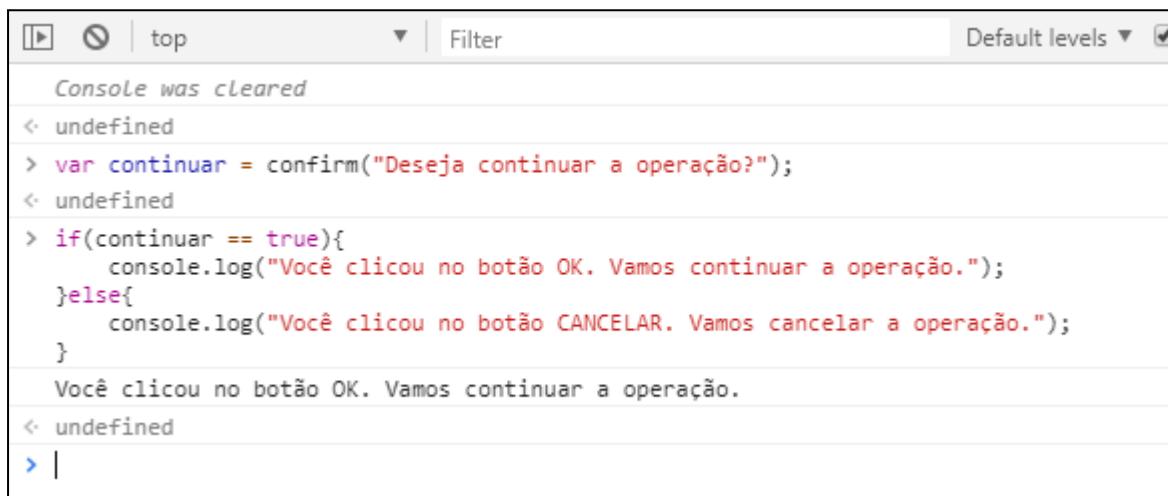
Primeiro, vamos criar a variável que vai armazenar o resultado do comando `confirm` e já escrever este comando também. Digite:

```
var continuar = confirm("Deseja continuar a operação?");
```

Agora, vamos verificar qual foi o botão pressionado utilizando a estrutura de decisão `if` e de acordo com o botão exibir a mensagem. Digite:

```
if(continuar == true){
    console.log("Você clicou no botão OK. Vamos continuar a
operação.");
} else{
    console.log("Você clicou no botão CANCELAR. Vamos cancelar a
operação.");
}
```

Veja como ficou o código inteiro:



The screenshot shows a browser's developer tools console interface. At the top, there are buttons for play/pause, stop, and refresh, followed by a dropdown menu set to "top". To the right of the dropdown are "Filter" and "Default levels" buttons. The main area of the console displays the following text:

```
Console was cleared
< undefined
> var continuar = confirm("Deseja continuar a operação?");
< undefined
> if(continuar == true){
    console.log("Você clicou no botão OK. Vamos continuar a operação.");
} else{
    console.log("Você clicou no botão CANCELAR. Vamos cancelar a operação.");
}
Você clicou no botão OK. Vamos continuar a operação.
< undefined
> |
```

Pense em mais situações onde você pode utilizar o `confirm` e crie seus exemplos.

# **Dia 10**

## **Missão: Melhorando nossa produtividade.**

### **Objetivo:**

- Instalar uma IDE

**Ao som de:** Capital Inicial - Vai e Vem ft. Seu Jorge

### **Descrição:**

**Caso ainda não tenha feito a missão do dia 9, volte e realize ela.**

Tudo o que fizemos até agora nas outras missões fizemos no console do navegador o Google Chrome, o que é muito bom pois para começar a programar foi bem simples. Uma das piores coisas em praticar no console é que ele não salva nada, então ao fechar o navegador, ou mesmo atualizar a janela, aquilo que você digitou era perdido. Nesse momento é hora de melhorar nosso ambiente de trabalho e instalar uma IDE para podermos salvar nossos códigos.

IDE (Integrated Development Environment) é um Ambiente de Desenvolvimento Integrado, ou seja, um editor de códigos com alguns recursos que melhoram a nossa produtividade. Existem ide's de vários portes, desde aquelas que são basicamente editores de códigos mais simples (Notepad++) até aqueles que possuem tantos recursos que talvez você nem utilize todos eles (Visual Studio).

Dentre essas que são mais simples, talvez as mais famosas sejam:

[NotePad++](#)

```
<NotepadPlus>
  <InternalCommands />
  <Macros>
    <Macro name="Trim Trailing and save" Ctrl="no" Alt="yes" Shift="yes" Key="83">
      <Action type="2" message="0" wParam="42024" lParam="0" sParam="" />
      <Action type="2" message="0" wParam="41006" lParam="0" sParam="" />
    </Macro>
  </Macros>
  <UserDefinedCommands>
    <Command name="Launch in Firefox" Ctrl="yes" Alt="yes" Shift="yes" Key="88">firefox &quot;$ (FULL_CURRENT_
    <Command name="Launch in IE" Ctrl="yes" Alt="yes" Shift="yes" Key="73">iexplore &quot;$ (FULL_CURRENT_PAT
    <Command name="Launch in Chrome" Ctrl="yes" Alt="yes" Shift="yes" Key="82">chrome &quot;$ (FULL_CURRENT_P_
    <Command name="Launch in Safari" Ctrl="yes" Alt="yes" Shift="yes" Key="70">safari &quot;$ (FULL_CURRENT_P_
    <Command name="Get php help" Ctrl="no" Alt="yes" Shift="no" Key="112">http://www.php.net/%20$ (CURRENT_WO
    <Command name="Google Search" Ctrl="no" Alt="yes" Shift="no" Key="113">http://www.google.com/search?q=$ (
    <Command name="Wikipedia Search" Ctrl="no" Alt="yes" Shift="no" Key="114">http://en.wikipedia.org/wiki/S
    <Command name="Open file" Ctrl="no" Alt="yes" Shift="no" Key="116">$ (NPP_DIRECTORY)\notepad++.exe $ (CURR
    <Command name="Open in another instance" Ctrl="no" Alt="yes" Shift="no" Key="117">$ (NPP_DIRECTORY)\notep
    <Command name="Open containing folder" Ctrl="no" Alt="no" Shift="no" Key="0">explorer $ (CURRENT_DIRECTOR
    <Command name="Open current dir cmd" Ctrl="no" Alt="no" Shift="no" Key="0">cmd /K cd /d $ (CURRENT_DIRECT
    <Command name="Send via Outlook" Ctrl="yes" Alt="yes" Shift="yes" Key="79">outlook /a &quot;$ (FULL_CURRE
  </UserDefinedCommands>
  <PluginCommands />
  <ScintillaKeys />
</NotepadPlus>
```

eXtensible Markup Language file      length : 2111    lines : 26      Ln:9 Col:22 Sel:0      Dos\Windows      ANSI      INS

## Visual Studio Code

```
<template>
  <div id="app">
    
    <HelloWorld msg="Welcome to Your Vue.js App"/>
  </div>
</template>

<script>
import HelloWorld from './components/HelloWorld.vue'

export default {
  name: 'app',
  components: {
    HelloWorld
  }
}
</script>

<style>
#app {
  font-family: 'Avenir', Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
  margin-top: 60px;
}
</style>
```

## [Atom](#)

The screenshot shows the Atom code editor interface. On the left, there's a sidebar titled "Project" displaying a file tree for a "real-time" directory. The "real-time-package.js" file is selected in the tree. The main area is a code editor tab titled "real-time-package.js" containing the following JavaScript code:

```
1  const {CompositeDisposable} = require('atom')
2  const {allowUnsafeNewFunction} = require('loophole')
3
4  let Client
5  allowUnsafeNewFunction(() => { Client =
6
7    const BufferBinding = require('./buffer-binding')
8    const EditorBinding = require('./editor-binding')
9
10   module.exports =
11     class RealTimePackage {
12       constructor (options) {
13         cons|
14
15       }
16
17     }
18
19   }
20
21   Client = RealTimePackage
22
23   if (allowUnsafeNewFunction) {
24     allowUnsafeNewFunction(() => {
25       Client = RealTimePackage
26     })
27   }
28
29   module.exports = Client
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
```

At the bottom right of the code editor, there's a status bar with "JavaScript" and a green "A" icon.

## [Sublime Text](#)

The screenshot shows the Sublime Text code editor. On the left, there's a sidebar titled "FOLDERS" showing a file tree for a "base64" directory. The "base64.cc" file is open in the main editor area. The code is C++ and implements a base64\_encode function. The code editor has dark-themed syntax highlighting. A vertical toolbar on the right side of the window contains various icons for file operations like save, close, and search.

```
34 void base64_encode(const uint8_t * data, size_t length, char * dst,
35                     base64_charset variant)
36 {
37   const char * charset = (variant == base64_charset::URL_SAFE)
38     ? URL_SAFE_CHARSET
39     : STANDARD_CHARSET;
40
41   size_t src_idx = 0;
42   size_t dst_idx = 0;
43   for (; (src_idx + 2) < length; src_idx += 3, dst_idx += 4)
44   {
45     uint8_t s0 = data[src_idx];
46     uint8_t s1 = data[src_idx + 1];
47     uint8_t s2 = data[src_idx + 2];
48
49     dst[dst_idx + 0] = charset[((s0 & 0xfc) >> 2)];
50     dst[dst_idx + 1] = charset[((s0 & 0x03) << 4) | ((s1 & 0xf0) >> 4)];
51     dst[dst_idx + 2] = charset[((s1 & 0x0f) << 2) | (s2 & 0xc0) >> 6];
52     dst[dst_idx + 3] = charset[(s2 & 0x3f)];
53   }
54
55   if (src_idx < length)
56   {
57     uint8_t s0 = data[src_idx];
58     uint8_t s1 = (src_idx + 1 < length) ? data[src_idx + 1] : 0;
59
60     dst[dst_idx++] = charset[((s0 & 0xfc) >> 2)];
61     dst[dst_idx++] = charset[((s0 & 0x03) << 4) | ((s1 & 0xf0) >> 4)];
62     if (src_idx + 1 < length)
63       dst[dst_idx++] = charset[((s1 & 0x0f) << 2)];
64
65     dst[dst_idx] = '\0';
66   }
67 }
```

## [Brackets](#)

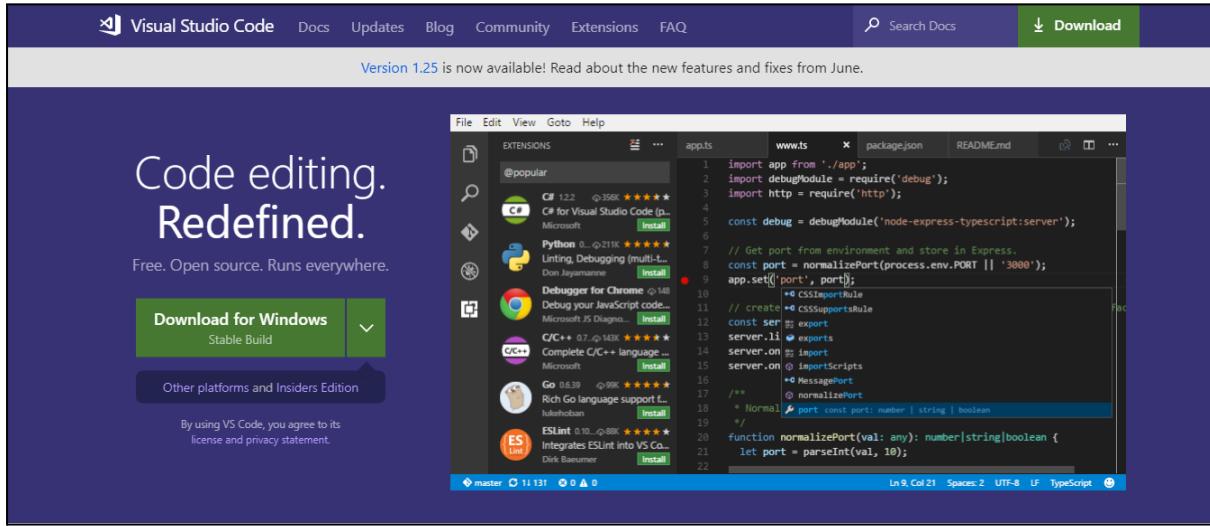
The screenshot shows the Brackets code editor interface. On the left, the 'Working Files' sidebar lists files like index.html, .gitignore, 404.html, blog.html, CNAME, contribute.html, and various CSS and JS files. The main editor area displays two files: index.html and brackets.io.css. The index.html file contains HTML code for a hero section, including a download button. The brackets.io.css file contains CSS for the hero-cta-button, setting it to a sans-serif font and center-aligning the text.

Basicamente, todos esses editores são muito parecidos entre si, a grande mudança entre eles são exatamente os plugins e extensões disponíveis para cada um. Plugins e extensões são as funções adicionais desenvolvidas para esses editores, essas funções podem ser desenvolvidas por qualquer pessoa da comunidade e não apenas funcionários das empresas.

## Missão

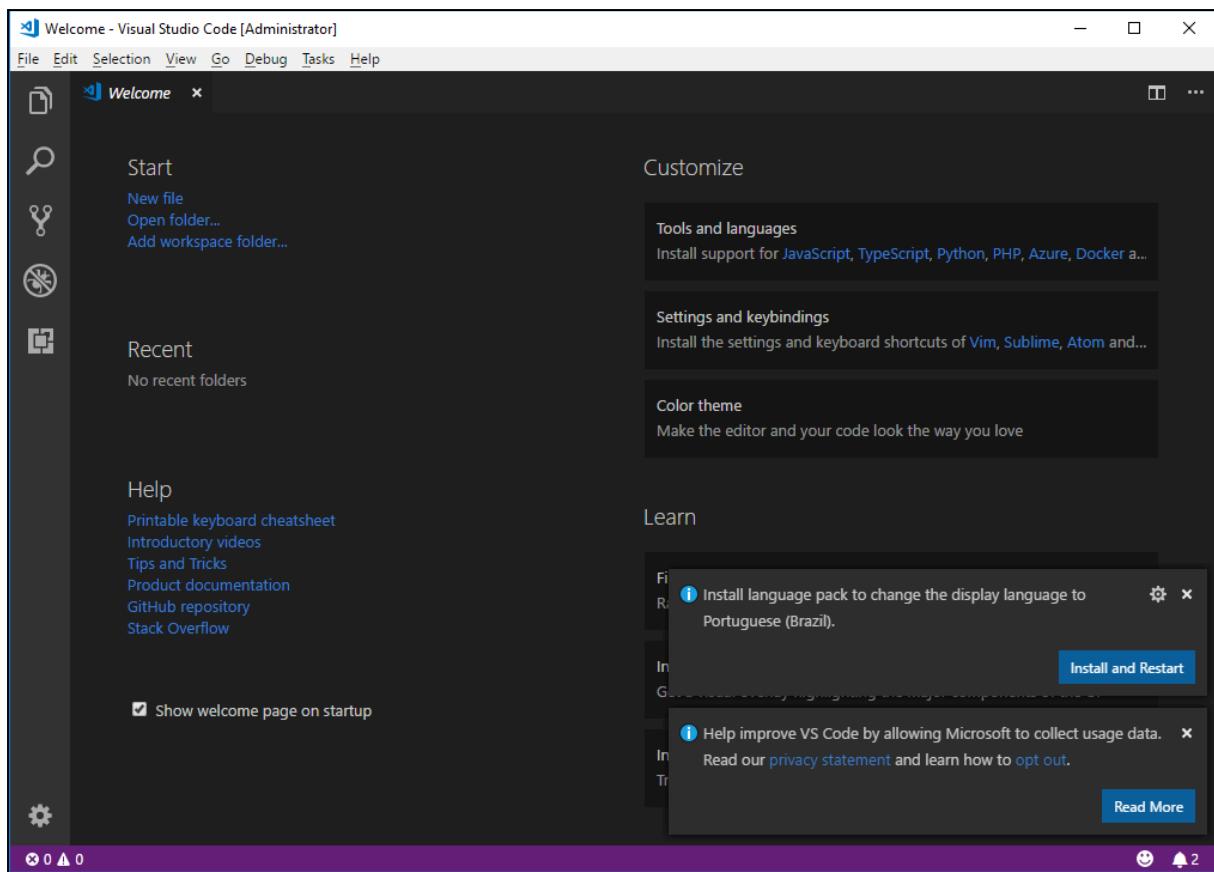
Durante nossas missões irei utilizar o Visual Studio Code como IDE padrão para realizar as atividades a partir desse momento.

Acesse o site do [Visual Studio Code](#) e faça download.



Após o download, instale o VS Code (o software não está disponível em português, pode instalar em inglês mesmo).

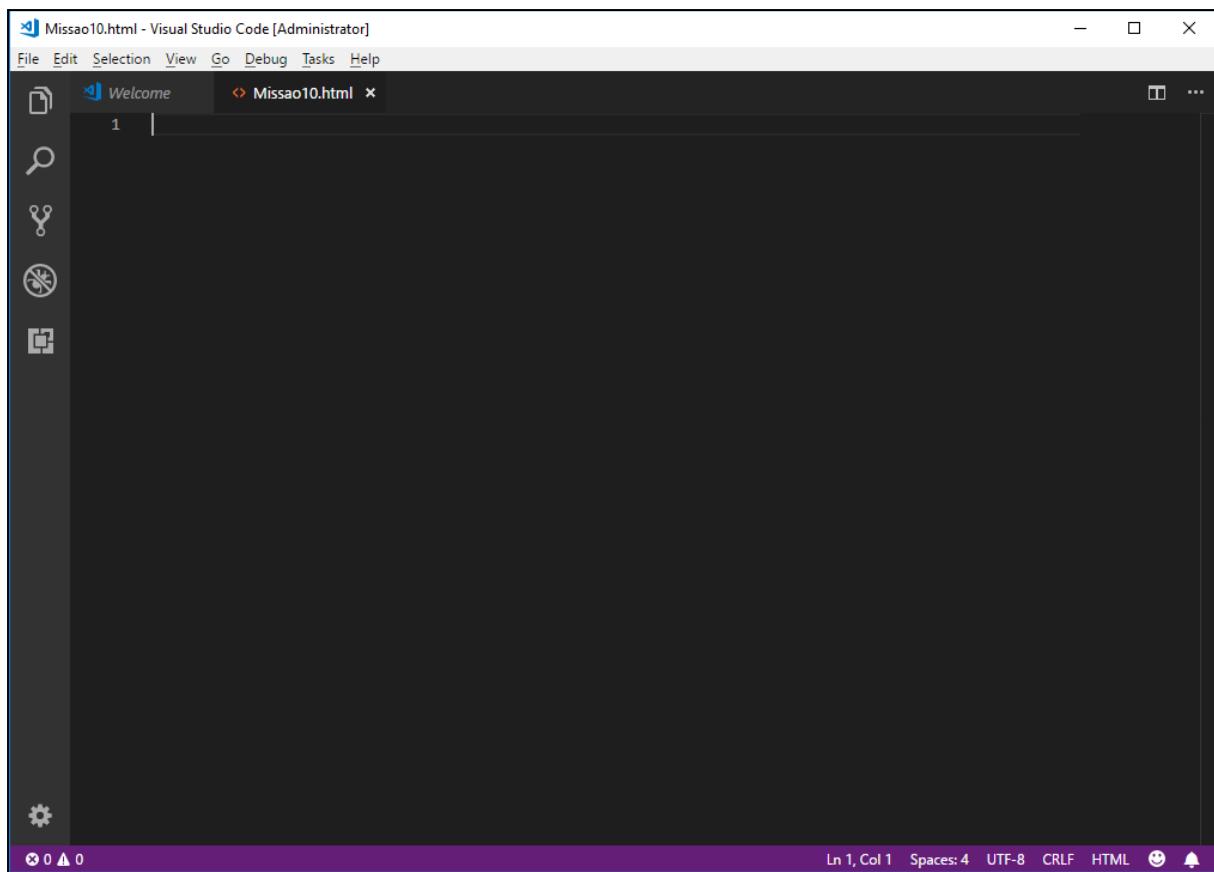
Terminada a instalação, abra o VS Code.



Ao abrir o VS Code você já verá as janelas para instalar o pacote para português (Brazil) disponível. Como estamos iniciando nosso estudo em programação, se quiser, pode instalar a tradução, eu não recomendo, vou utilizar em inglês.

Pressione **CTRL+N** (ou click em **File → New File**) para criar um novo arquivo, observe que ele criou uma nova aba chamada Untitled-1.

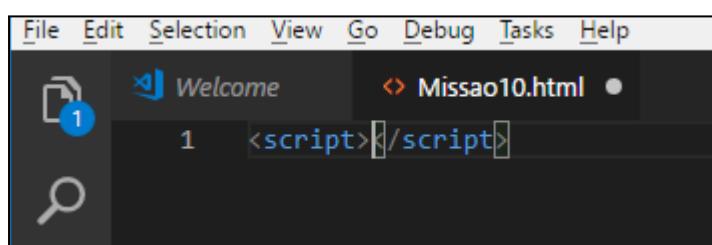
Pressione **CTRL+S** para salvar o arquivo, coloque o nome de “**Missao10.html**” e click em salvar.



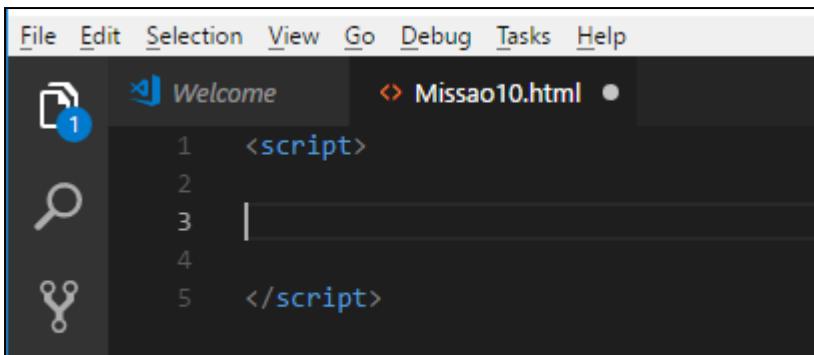
Digite:

```
<script>
```

Perceba que ao digitar `<script>` o editor colocar automaticamente `</script>`.



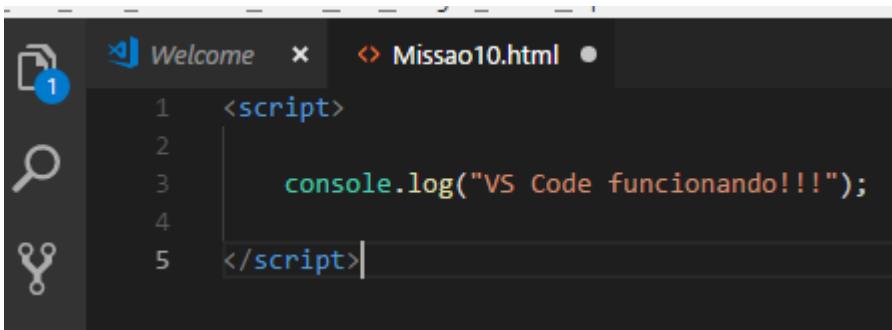
Todo nosso código JavaScript deverá ficar entre as tag's `<script></script>`. Click entre as tag's script e insira algumas linhas em branco:



A screenshot of the Visual Studio Code interface. The menu bar at the top includes File, Edit, Selection, View, Go, Debug, Tasks, and Help. Below the menu is a toolbar with icons for file operations, search, and settings. The main workspace shows a file named "Missao10.html" with the following content:

```
1 <script>
2
3 |
4
5 </script>
```

Digite a linha de comando como na imagem abaixo.

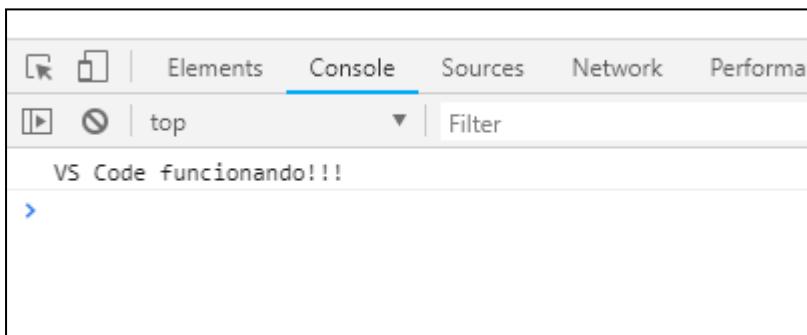


A screenshot of the Visual Studio Code interface, similar to the previous one but with additional code in the script tag:

```
1 <script>
2
3     console.log("VS Code funcionando!!!");
4
5 </script>
```

Perceba que na frente do nome do arquivo existe uma bolinha, essa bolinha significa que existem coisas que não foram salvas nesse arquivo. Pressione CTRL+S para salvar o arquivo e veja que essa bolinha vai sumir.

Depois de salvar o arquivo minimize o VS code, e abra o arquivo **Missao10.html** no Google Chrome (click com o botão direito no arquivo e escolha a opção **Abrir com → Google Chrome**), abra o console do navegador e veja nossa mensagem.



A screenshot of the Google Chrome DevTools Console tab. The tab bar includes Elements, Console, Sources, Network, and Performance. The console output shows the message:

```
VS Code funcionando!!!
>
```

Agora sua vez de praticar, escolha algumas (ou todas) das missões anteriores e crie arquivos separados no VS Code (Missao1.html, Missao2.html, Missao3.html, etc) e realize-as utilizando o VS Code.

# Dia 11

## Missão: Exibindo uma mensagem várias vezes.

### Objetivo:

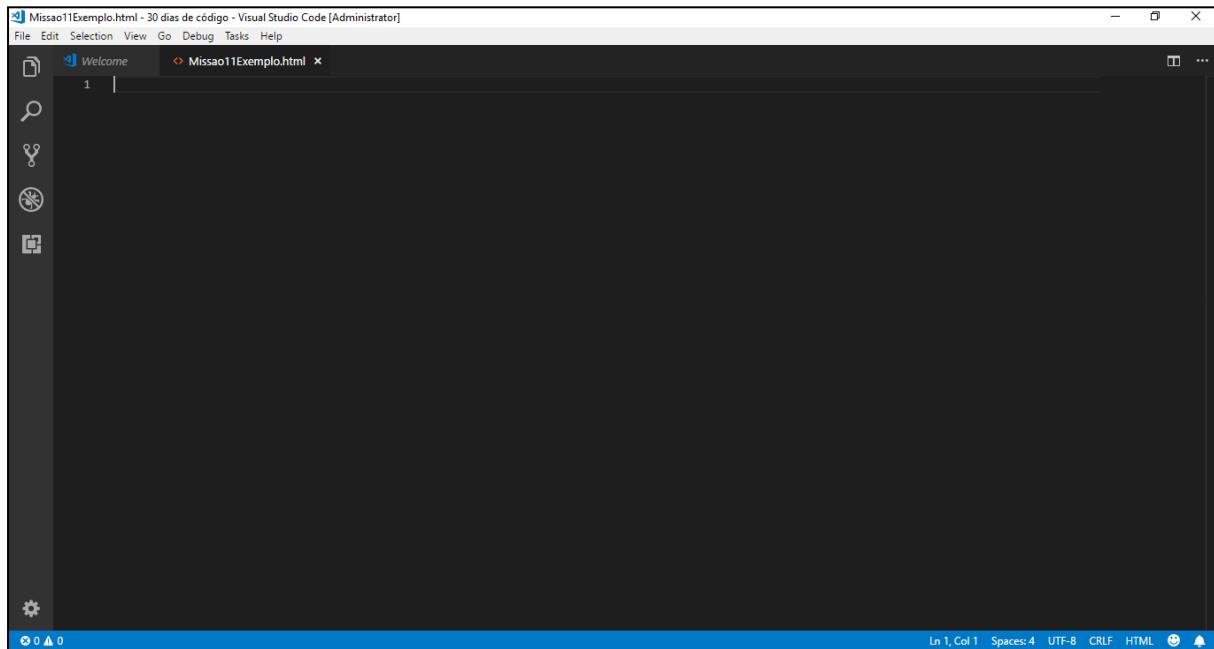
- Laço de repetição (while)

**Ao som de:** Alok, Bruno Martini feat. Zeeba - Hear Me Now

### Descrição:

Caso ainda não tenha feito a missão do dia 10, volte e realize ela.

Abra o VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao11Exemplo.html



O laço de repetição permite que você repita uma série de instruções até a condição para o laço se tornar falsa. Veja o exemplo a seguir, digite no VS Code dentro da tag script:

```
<script>
var contador = 1;
while( contador <= 5 ){
```

```

        console.log("Bom dia");
        contador = contador + 1;
    }
</script>

```

The screenshot shows the Visual Studio Code interface with a dark theme. On the left is a sidebar with icons for file operations like new file, save, and close. The main area displays the following code:

```

1 <script>
2     var contador = 1;
3     while( contador <= 5 ){
4         console.log("Bom dia");
5         contador = contador + 1;
6     }
7 </script>
8

```

Salve o arquivo e abra o arquivo no Google Chrome, abra o console e veja o resultado:

The screenshot shows a browser's developer tools console window. It displays two lines of text: "5 Bom dia" and "< 6".

Vamos entender linha à linha o que aconteceu:

Primeiro temos `var contador = 1;` nessa linha declaramos uma variável que vai servir para controlar quantas vezes o laço de repetição foi executado. A cada vez que o laço executar, na última linha dentro do laço existe o comando `contador = contador + 1;` que acrescenta mais um à essa variável de controle. Essa variável começa o programa valendo 1, mas depois de executar o laço uma vez ela vale 2, e depois 3 e assim sucessivamente.

Depois temos `while( contador <= 5 ){` aqui é a condição que deve ser satisfeita para que os comandos dentro do laço sejam executados. O contador começa valendo 1, como 1 é menor ou igual à 5 ele entra no laço e realiza um loop, a cada loop aumenta 1 no valor da variável contador, depois de 5 loops o contador tem o valor de 6, o que faz com que a condição seja falsa e o laço de repetição não seja mais executado.

Esse é o laço de repetição chamado enquanto...faça, ou seja enquanto a condição for verdadeira faça o que estiver dentro do laço.

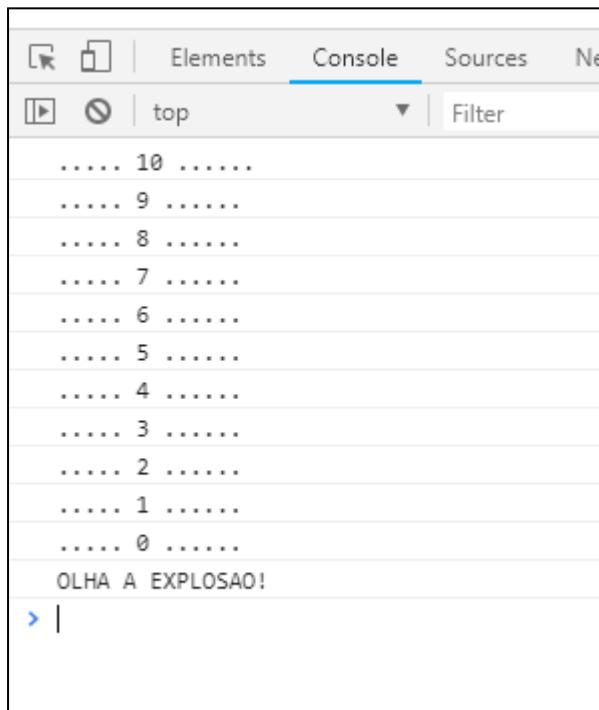
Vamos criar uma outra situação, tipo eu quero uma contagem regressiva, começando em 10 e depois que chegar a zero apareça a mensagem “OLHA A EXPLOSÃO!”. Volte ao VS Code e altere o código para esse:

```
<script>
    var contador = 10;
    while(contador >= 0){
        console.log("..... " + contador + " .....");
        contador = contador - 1;
    }
    console.log("OLHA A EXPLOSAO!");
</script>
```

A screenshot of a code editor window titled "Welcome" with a tab labeled "Missao11Exemplo.html". The code editor displays a script block with line numbers 1 through 9. The code itself is:

```
1  <script>
2      var contador = 10;
3      while(contador >= 0){
4          console.log("..... " + contador + " .....");
5          contador = contador - 1;
6      }
7      console.log("OLHA A EXPLOSAO!");
8  </script>
9
```

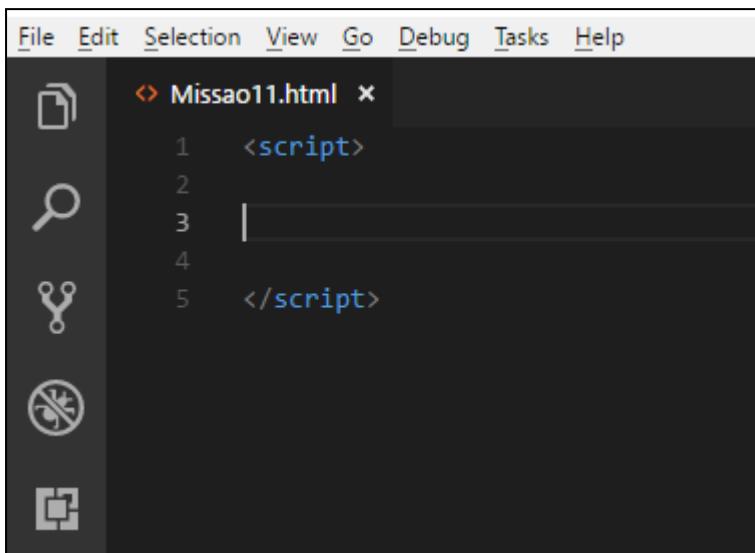
Salve seu arquivo e abra ele no Google Chrome novamente e veja o resultado:

A screenshot of the Google Chrome DevTools Console tab. The tab bar shows "Console" is active. The console output shows the following:

```
..... 10 .....
..... 9 .....
..... 8 .....
..... 7 .....
..... 6 .....
..... 5 .....
..... 4 .....
..... 3 .....
..... 2 .....
..... 1 .....
..... 0 .....
OLHA A EXPLOSAO!
```

## Missão

Abra o VS Code, aperte CTRL+N para criar um arquivo novo, aperte CTRL+S e salve o arquivo com o nome de Missao11.html



A screenshot of the Visual Studio Code interface. The menu bar at the top includes File, Edit, Selection, View, Go, Debug, Tasks, and Help. Below the menu is a dark-themed editor window. On the left side of the editor is a vertical sidebar with five icons: a document, a magnifying glass, a gear, a circular arrow, and a square. The editor window title is "Missao11.html \*". The code area contains the following lines:

```
1 <script>
2
3 |
4
5 </script>
```

A missão consiste em repetir uma mesma frase uma certa quantidade de vezes. Nós já fizemos isso no exemplo sobre laço de repetição, para essa missão vamos perguntar para o usuário quantas vezes ele quer repetir a mensagem.

Vamos criar uma variável para ser nosso contador e perguntar quantas vezes ele deseja repetir a mensagem. Digite:

```
var contador = 1;
var numeroRepeticao = prompt("Informe o numero de vezes que a mensagem deve aparecer", "Digite aqui");
```

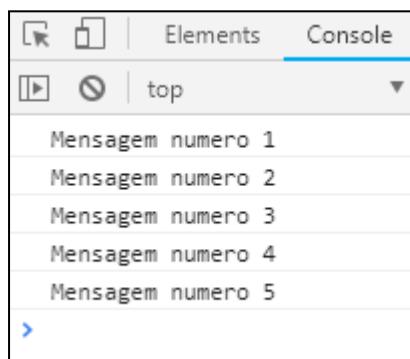
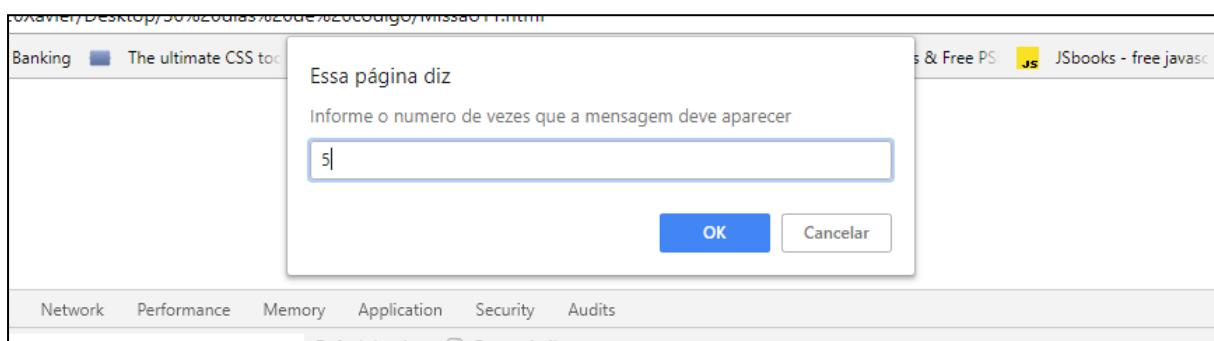
Vamos criar o laço de repetição para repetir a mensagem. Digite:

```
while(contador <= numeroRepeticao){
    console.log("Mensagem numero " + contador);
    contador = contador + 1;
}
```

Veja como ficou o código completo:

```
Missao11.html x
1 <script>
2
3 var contador = 1;
4 var numeroRepeticao = prompt("Informe o numero de vezes que a mensagem deve aparecer", "Digite aqui");
5
6 while(contador <= numeroRepeticao){
7     console.log("Mensagem numero " + contador);
8     contador = contador + 1;
9 }
10
11 </script>
```

Salve seu arquivo e abra no Google Chrome para ver o resultado:



Tente criar um exemplo de contagem regressiva onde o usuário informa qual o maior número para ir diminuindo.

# Dia 12

## Missão: Calcular a média de N alunos.

### Objetivo:

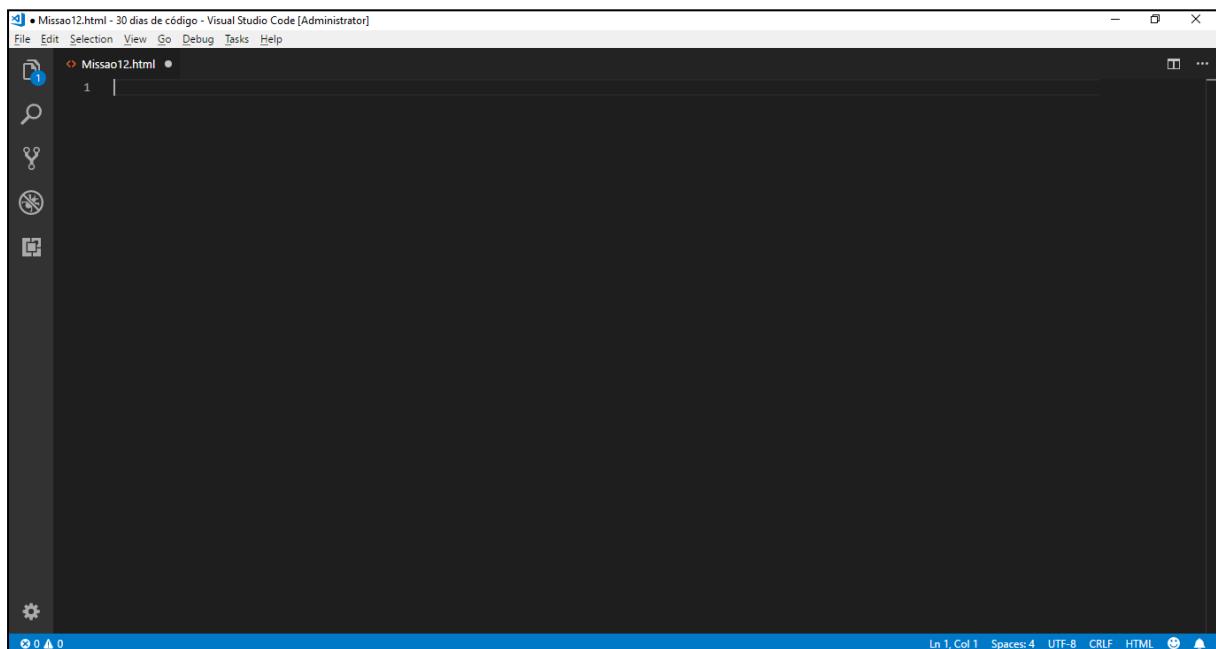
- Laço de repetição (while)

**Ao som de:** James Bay - If You Ever Want To Be In Love

### Descrição:

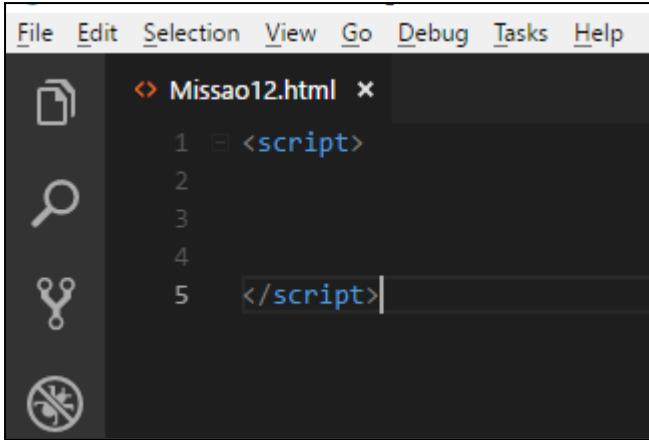
Caso ainda não tenha feito a missão do dia 11, volte e realize ela.

Abra o VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao12.html



E já digite:

```
<script>  
</script>
```



A screenshot of a code editor window titled "Missao12.html". The menu bar includes File, Edit, Selection, View, Go, Debug, Tasks, and Help. The left sidebar has icons for file, search, and refresh. The main area shows the following code:

```
1 <script>
2
3
4
5 </script>
```

## Missão

Vamos fazer um programa para calcular a média de um aluno. Ao apresentar o resultado, o programa deve perguntar se gostaria de calcular a média de outro aluno, caso a resposta seja SIM, o programa deve executar novamente.

Declare as variáveis que irão armazenar o nome, as notas e a média do aluno. Digite:

```
var nome, nota1, nota2, nota3, nota4, media;
```

Agora, declare a variável que vai armazenar a resposta do usuário que será um true ou false. Nós vamos inicializar ela com true para entrar no laço de repetição pelo menos a primeira vez. Digite:

```
var resposta = true;
```

Crie o laço de repetição, onde dentro do laço será pedido o nome do aluno, cada uma das notas, a média será calculada, será exibido o nome do aluno e sua respectiva nota e ao final será exibido uma janela com os botões OK ou CANCELAR para saber se deve ser executado o laço novamente. Digite:

```
while(resposta==true){
    nome = prompt("Informe o nome do aluno(a)","Digite aqui");
    nota1 = prompt("Informe a nota do 1º bimestre","Digite aqui");
    nota2 = prompt("Informe a nota do 2º bimestre","Digite aqui");
    nota3 = prompt("Informe a nota do 3º bimestre","Digite aqui");
```

```

nota4 = prompt("Informe a nota do 4º bimestre","Digite aqui");
media = (parseFloat(nota1) + parseFloat(nota2) + parseFloat(nota3)
+ parseFloat(nota4))/4;
alert("O(A) estudante " + nome + " obteve a media " + media);
resposta = confirm("Calcular a media de outro aluno?");
}

```

O código inteiro fica da seguinte forma:

```

1 <script>
2     var nome, nota1, nota2, nota3, nota4, media;
3     var resposta = true;
4
5     while(resposta==true){
6         nome = prompt("Informe o nome do aluno(a)","Digite aqui");
7         nota1 = prompt("Informe a nota do 1º bimestre","Digite aqui");
8         nota2 = prompt("Informe a nota do 2º bimestre","Digite aqui");
9         nota3 = prompt("Informe a nota do 3º bimestre","Digite aqui");
10        nota4 = prompt("Informe a nota do 4º bimestre","Digite aqui");
11        media = (parseFloat(nota1) + parseFloat(nota2) + parseFloat(nota3) + parseFloat(nota4))/4;
12        alert("O(A) estudante " + nome + " obteve a media " + media);
13        resposta = confirm("Calcular a media de outro aluno?"); 
14    }
15 </script>
16

```

Salve o arquivo e abra no Google Chrome para ver o resultado.

# Dia 13

## Missão: Calcular a média de N números.

### Objetivo:

- Laço de repetição (while)

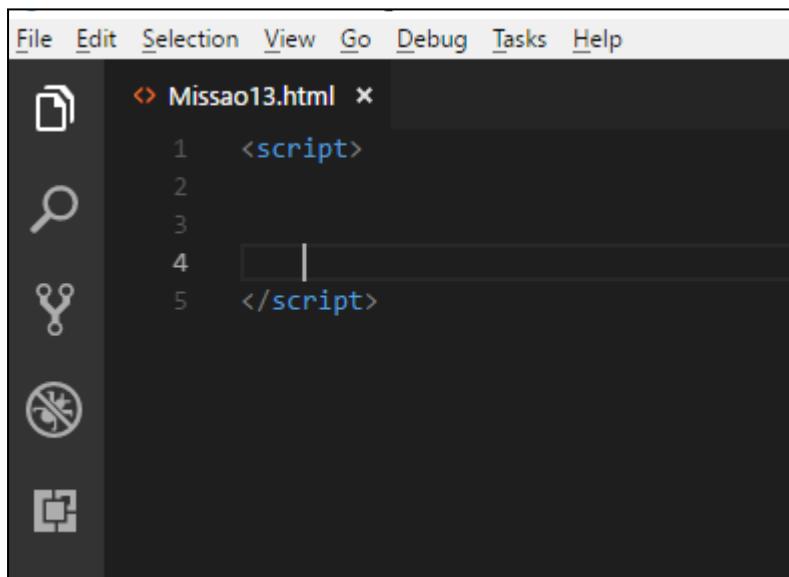
**Ao som de:** David Gilmour - Wish you were here

### Descrição:

Caso ainda não tenha feito a missão do dia 12, volte e realize ela.

Abra o VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao13.html e já digite:

```
<script>  
</script>
```



The screenshot shows the Visual Studio Code interface. The menu bar at the top includes File, Edit, Selection, View, Go, Debug, Tasks, and Help. Below the menu is a toolbar with icons for file operations like open, save, and search. The main workspace shows a file named "Missao13.html" with the following content:

```
File Edit Selection View Go Debug Tasks Help  
Missao13.html ×  
1 <script>  
2  
3  
4 |  
5 </script>
```

## Missão

Nosso programa vai calcular a média de quantos números o usuário quiser digitar. Relembrando o cálculo da média é a SOMA DE TODOS OS NÚMEROS DIVIDIDO PELA QUANTIDADE DE NÚMEROS. Por exemplo, se eu tenho 3 números, a média é a soma desses 3 números dividido por 3.

No nosso programa depois de digitar o número será perguntado se o usuário deseja digitar outro número, se ele disser que SIM, aparece a opção para inserir um novo número, se disser que não será exibido a média

Primeiro, vamos declarar as variáveis que irão armazenar a quantidade de números que o usuário digitou, a variável que vai armazenar se o usuário irá inserir um novo número, a variável que irá armazenar a média, a variável que vai armazenar a soma total dos números digitados e por último a variável que vai armazenar o número atual digitado. Digite:

```
var quantidadeNumeros = 0;
var novoNumero = "S";
var media = 0;
var somaNumeros = 0;
var numero = 0;
```

Agora vamos criar o laço de repetição, onde dentro do laço terão as instruções perguntando o número novo, somando esse novo número ao total já existente, aumentando em 1 a quantidade de números digitados e perguntando se o usuário deseja digitar mais um número. Digite:

```
while(novoNumero == "S"){
    numero = prompt("Informe um numero","Digite aqui!");
    somaNumeros = parseFloat(somaNumeros) + parseFloat(numero);
    quantidadeNumeros = quantidadeNumeros + 1;
    novoNumero = prompt("Deseja informar mais um numero? \n[S]im\n[N]ao");
}
```

Depois de terminado o laço vamos calcular a média e exibir o resultado. Digite:

```
media = somaNumeros / quantidadeNumeros;

console.log("A media dos numeros informados é " + media);
```

O código inteiro fica assim:

```

<script>
    var quantidadeNumeros = 0;
    var novoNumero = "S";
    var media = 0;
    var somaNumeros = 0;
    var numero = 0;

    while(novoNumero == "S"){
        numero = prompt("Informe um numero","Digite aqui!");
        somaNumeros = parseFloat(somaNumeros) + parseFloat(numero);
        quantidadeNumeros = quantidadeNumeros + 1;
        novoNumero = prompt("Deseja informar mais um numero?
\n[S]im\n[N]ao");
    }

    media = somaNumeros / quantidadeNumeros;

    console.log("A media dos numeros informados é " + media);

</script>

```

The screenshot shows a code editor window with the following details:

- File Menu:** File, Edit, Selection, View, Go, Debug, Tasks, Help.
- Toolbar:** Includes icons for file operations (New, Open, Save, Find, Copy, Paste, Select All, Undo, Redo), a search icon, and a refresh/circular arrow icon.
- Code Area:**
  - File name: Missao13.html
  - Content:

```

1 <script>
2     var quantidadeNumeros = 0;
3     var novoNumero = "S";
4     var media = 0;
5     var somaNumeros = 0;
6     var numero = 0;
7
8     while(novoNumero == "S"){
9         numero = prompt("Informe um numero","Digite aqui!");
10        somaNumeros = parseFloat(somaNumeros) + parseFloat(numero);
11        quantidadeNumeros = quantidadeNumeros + 1;
12        novoNumero = prompt("Deseja informar mais um numero? \n[S]im\n[N]ao");
13    }
14
15    media = somaNumeros / quantidadeNumeros;
16
17    console.log("A media dos numeros informados é " + media);
18
19 </script>
```

Salve o arquivo e abra no Google Chrome para ver o resultado, não esqueça de abrir o console.

# Dia 14

## Missão: Criar um contador.

### Objetivo:

- Laço de repetição (while)
- Operadores de incremento e decremento

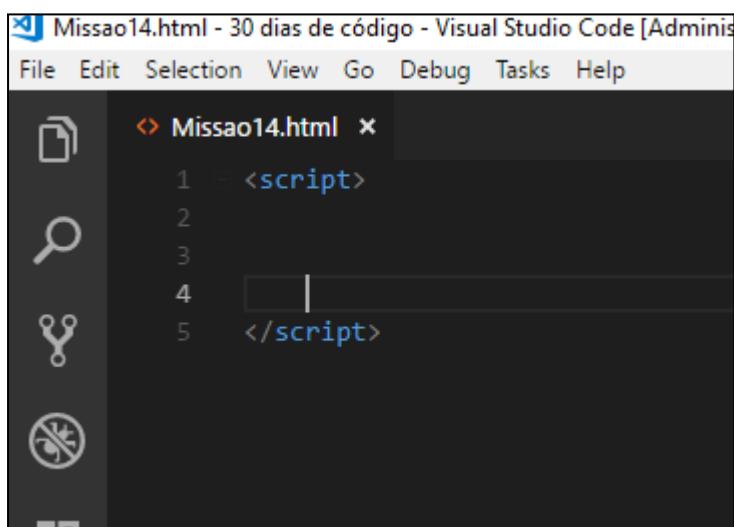
**Ao som de:** Paul McCartney - Helter Skelter

### Descrição:

Caso ainda não tenha feito a missão do dia 13, volte e realize ela.

Abra o VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao14.html e já digite:

```
<script>  
</script>
```



The screenshot shows a dark-themed instance of Visual Studio Code. The title bar reads "Missao14.html - 30 dias de código - Visual Studio Code [Admin]". The menu bar includes File, Edit, Selection, View, Go, Debug, Tasks, and Help. On the left is a sidebar with icons for file, search, and other tools. The main editor area contains the following code:

```
1 <script>  
2  
3  
4  
5 </script>
```

## Missão

Vamos criar um contador que vai de 0 à 10 aumentando de um em um, e depois volta de 10 à 0 diminuindo de um em um.

Até agora quando queríamos aumentar de um em um em nossos exemplos utilizamos algo como:

```
contador = contador + 1;
```

E quando queríamos diminuir usamos:

```
contador = contador - 1;
```

Porém existem operadores específicos para incrementar (+1) e decrementar (-1) valores, são os operadores `++` e `--` e são eles que utilizaremos nessa missão.

Primeiro vamos declarar as duas variáveis que vão servir como contadores, uma que deverá aumentar e outra que deverá diminuir. Digite:

```
var contador = 0;
var contador2 = 10;
```

Agora crie o laço de repetição que deve mostrar de 0 à 10, aumentando de um por um por causa da instrução `contador++`. Digite:

```
alert("Vamos aumentar");
while(contador <= 10){
    console.log(contador);
    contador++;
}
```

Agora crie o laço de repetição que deve mostrar de 10 à 0, diminuindo de um por um por causa da instrução `contador2--`. Digite:

```
alert("Vamos diminuir");
while(contador2 >= 0){
    console.log(contador2);
    contador2--;
}
```

O código completo fica assim:

```
<script>
    var contador = 0;
    var contador2 = 10;

    alert("Vamos aumentar");

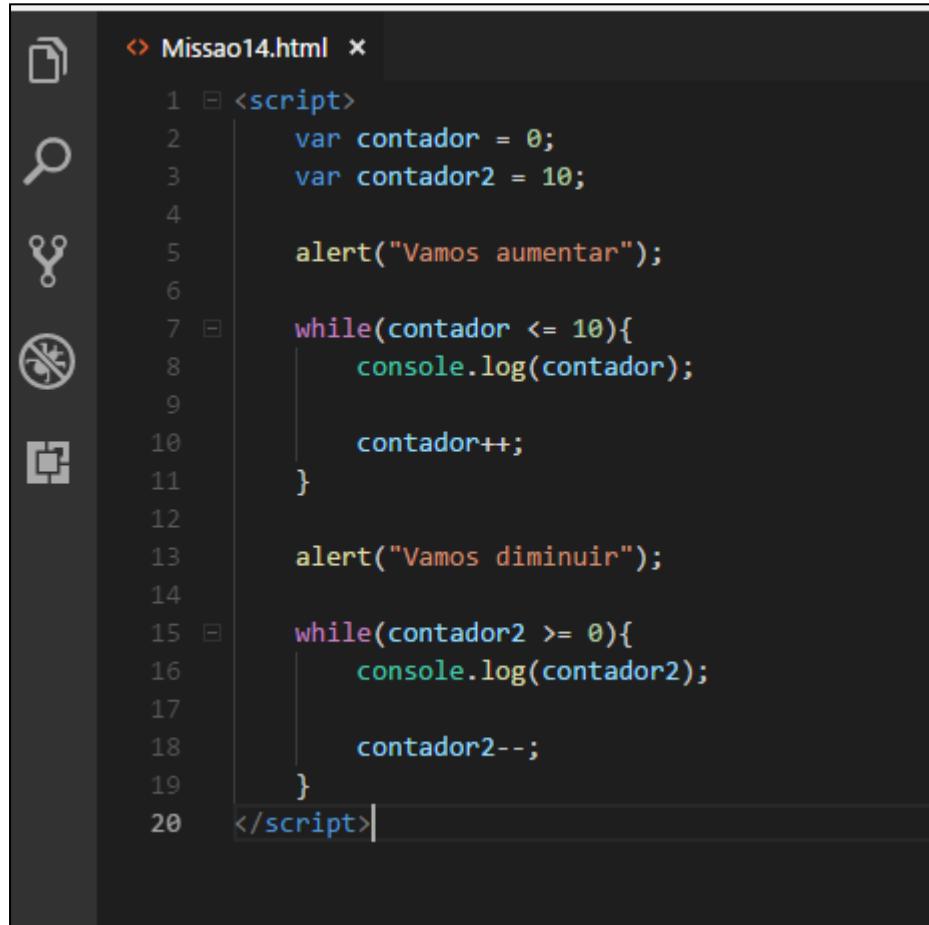
    while(contador <= 10){
        console.log(contador);

        contador++;
    }

    alert("Vamos diminuir");

    while(contador2 >= 0){
        console.log(contador2);

        contador2--;
    }
</script>
```



The screenshot shows a code editor window with a dark theme. On the left is a vertical toolbar with icons for file operations, search, and other functions. The main area displays the script code from the previous snippet. The code is numbered from 1 to 20 on the left side. The syntax highlighting is color-coded: blue for keywords like `<script>`, `var`, `while`, and `console`; green for strings like `"Vamos aumentar"` and `"Vamos diminuir"`; and orange for numbers like `0`, `10`, and `10` in the loop conditions.

```
1 <script>
2     var contador = 0;
3     var contador2 = 10;
4
5     alert("Vamos aumentar");
6
7     while(contador <= 10){
8         console.log(contador);
9
10        contador++;
11    }
12
13    alert("Vamos diminuir");
14
15    while(contador2 >= 0){
16        console.log(contador2);
17
18        contador2--;
19    }
20 </script>
```

Salve o arquivo e abra no Google Chrome para ver o resultado, não esqueça de abrir o console.

# Dia 15

## Missão: Armazenando lista de carros de um estacionamento

### Objetivo:

- Utilizando Array
- Laço de repetição(while)

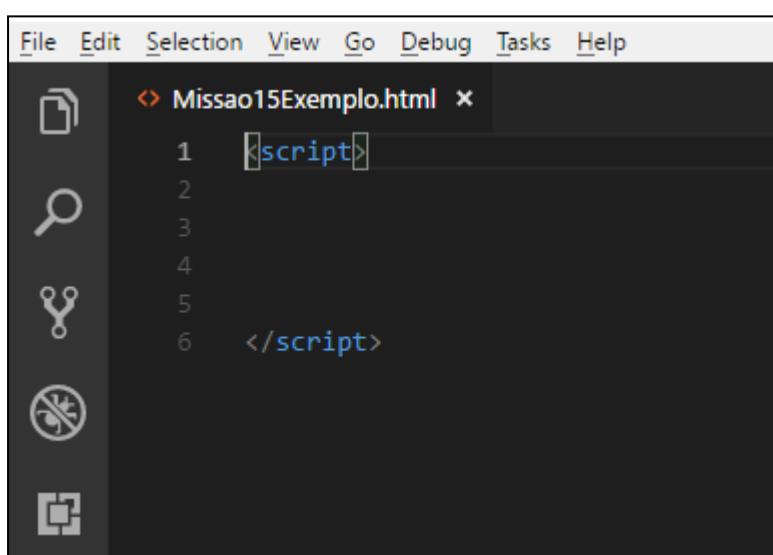
**Ao som de:** Bob Dylan Hurricane

### Descrição:

Caso ainda não tenha feito a missão do dia 14 volte e realize ela.

Abra o VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao15Exemplo.html e já digite:

```
<script>  
  
</script>
```



A screenshot of the Visual Studio Code interface. The menu bar at the top includes File, Edit, Selection, View, Go, Debug, Tasks, and Help. On the left is a sidebar with icons for file operations like new file, save, find, and refresh. The main editor area shows a single line of code: <script>. This line is highlighted with a light blue background and has a small yellow bracket icon to its left, indicating it's selected. The line number 1 is also visible to the left of the code.

### Array

Array é um tipo de variável(não confundir com tipo de dados) utilizado para armazenar múltiplos valores. A grande característica do array é que ele pode ter subdivisões endereçadas(indice) e em cada uma dessas subdivisões armazenar um valor diferente. Por exemplo, vamos imaginar que você precise guardar o nome de 5 alunos, normalmente, com o que aprendemos até aqui você faria da seguinte forma, digite:

```
var aluno1 = "Maria";
var aluno2 = "Rose";
var aluno3 = "Claudio";
var aluno4 = "Renata";
var aluno5 = "Fernando";
```

```
2  var aluno1 = "Maria";
3  var aluno2 = "Rose";
4  var aluno3 = "Claudio";
5  var aluno4 = "Renata";
6  var aluno5 = "Fernando";
7
```

E para exibir o nome de cada aluno você faria algo parecido com isso, digite:

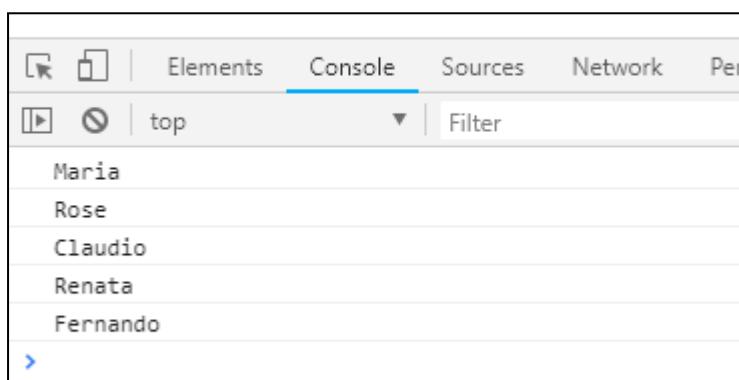
```
console.log(aluno1);
console.log(aluno2);
console.log(aluno3);
console.log(aluno4);
console.log(aluno5);
```

```
8  console.log(aluno1);
9  console.log(aluno2);
10 console.log(aluno3);
11 console.log(aluno4);
12 console.log(aluno5);
```

O código completo até aqui:

```
1 ④ <script>
2      var aluno1 = "Maria";
3      var aluno2 = "Rose";
4      var aluno3 = "Claudio";
5      var aluno4 = "Renata";
6      var aluno5 = "Fernando";
7
8      console.log(aluno1);
9      console.log(aluno2);
10     console.log(aluno3);
11     console.log(aluno4);
12     console.log(aluno5);
13
14     </script>
```

Salve seu arquivo e abra no Google Chrome para visualizar o resultado que estará no console:



Dessa forma conseguimos produzir o resultado esperado, que era armazenar o nome dos alunos e exibi-los posteriormente. Mas será que essa é a melhor maneira? Até aqui é a maneira possível, portanto, a melhor maneira dentro do que aprendemos.

## Array na prática

Um pouco mais acima está escrito que “array é um tipo de variável utilizado para armazenar múltiplos valores” vamos ver o funcionamento de array na prática para resolver nosso problema.

Para criarmos um array a sintaxe é:

```
var nome_array = [ valor1 , valor2 , valor3 , valor 4 , ...];
```

Vamos substituir as 5 primeiras linhas do código que criamos, onde foram declaradas as variáveis, por uma linha apenas. Digite:

```
var alunos = ["Maria" , "Rose" , "Claudio" , "Renata" , "Fernando"];
```

```
1 <script>
2 var alunos = ["Maria" , "Rose" , "Claudio" , "Renata" , "Fernando"];
3 |
```

As variáveis aluno1, aluno2, aluno3, aluno4 e aluno5 foram substituídas pela variável alunos. A variável alunos é um array que contém todos os nomes dos alunos, ou seja contém um conjunto de valores.

O array alunos contém 5 valores, logo falamos que ela contém 5 posições. Para acessar cada uma das posições utilizamos um índice. Esse índice inicia em 0. Observe a tabela abaixo:

0	1	2	3	4
Maria	Rose	Claudio	Renata	Fernando

Se eu quiser trocar o nome da Renata por Renato eu preciso acessar a posição 3 do array alunos e substituir o valor. Digite:

```
alunos[3] = "Renato";
```

```
3
4 alunos[3] = "Renato";
5 |
```

Para mostrar cada nome disponível no array alunos basta eu utilizar o comando console.log utilizando cada uma das posições do array alunos. Digite:

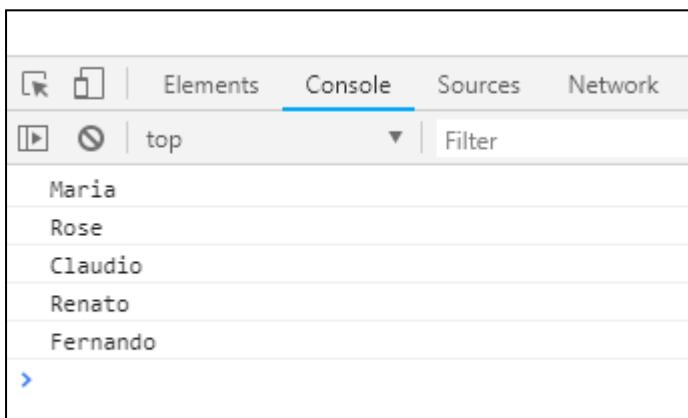
```
console.log(alunos[0]);
console.log(alunos[1]);
console.log(alunos[2]);
console.log(alunos[3]);
console.log(alunos[4]);
```

```
6  console.log(alunos[0]);
7  console.log(alunos[1]);
8  console.log(alunos[2]);
9  console.log(alunos[3]);
10 console.log(alunos[4]);
```

O código inteiro até o momento está assim:

```
1  <script>
2  var alunos = ["Maria" , "Rose" , "Claudio" , "Renata" , "Fernando"];
3
4  alunos[3] = "Renato";
5
6  console.log(alunos[0]);
7  console.log(alunos[1]);
8  console.log(alunos[2]);
9  console.log(alunos[3]);
10 console.log(alunos[4]);
11 </script>
```

Salve e abra no navegador para ver o resultado no console:



Veja que nosso código já está menor do que o anterior, e também mais performático, pois ao invés de utilizarmos 5 variáveis na memória estamos utilizando apenas uma.

Podemos utilizar o que já aprendemos sobre laço de repetição e melhorar também a exibição dos nomes dos alunos. Digite:

```
var contador = 0;
while(contador<=4){
    console.log(alunos[contador]);
    contador++;
}
```

Veja o código completo:

```
1  ↳ <script>
2    var alunos = ["Maria" , "Rose" , "Claudio" , "Renata" , "Fernando"];
3
4    alunos[3] = "Renato";
5
6    var contador = 0;
7    ↳ while(contador<=4){
8        console.log(alunos[contador]);
9        contador++;
10   }
11  </script>
```

Salve e abra no navegador para ver o resultado no console.

Podemos dar uma melhor organizada no nosso código e comparar com a primeira versão.  
versão atual:

```
1  ↳ <script>
2    var alunos = ["Maria" , "Rose" , "Claudio" , "Renata" , "Fernando"];
3    var contador = 0;
4    ↳
5    ↳ while(contador<=4){
6        console.log(alunos[contador]);
7        contador++;
8    }
9    </script>
```

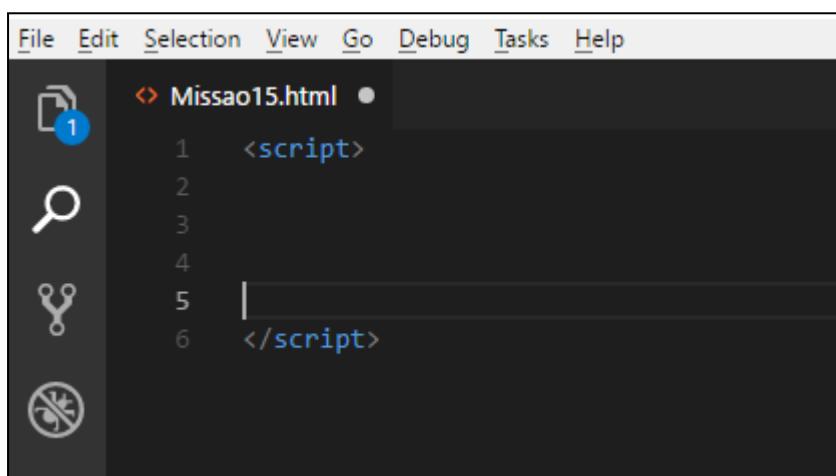
Primeira versão:

```
1  ↳ <script>
2    var aluno1 = "Maria";
3    var aluno2 = "Rose";
4    var aluno3 = "Claudio";
5    var aluno4 = "Renata";
6    var aluno5 = "Fernando";
7
8    console.log(aluno1);
9    console.log(aluno2);
10   console.log(aluno3);
11   console.log(aluno4);
12   console.log(aluno5);
13
14  </script>
```

## Missão

Crie um arquivo novo no VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao15.html e já digite:

```
<script>  
    </script>
```



Nossa missão consiste em armazenar a lista de carros de um estacionamento. Primeiro vamos declarar a variável responsável por armazenar a lista de carros e o contador. Digite:

```
var carros = ["Carro1" , "Carro2" , "Carro3" , "Carro4" , "Carro5"];  
var contador = 0;
```

Agora crie o laço de repetição responsável por popular o array carros com os modelos dos carros. Digite:

```
while(contador<=4){
```

```
    carros[contador] = prompt("Informe o modelo do carro nº " + contador ,  
"Digite aqui");  
    contador++;  
}
```

Agora vamos zerar novamente a variável contador para que ela possa ser utilizada para criar um outro loop. Digite:

```
contador = 0;
```

Por fim, vamos criar o laço de repetição responsável por exibir os carros no console. Digite:

```
while(contador <= 4){  
    console.log("Modelo nº " + contador + ": " + carros[contador]);  
    contador++;  
}
```

O código completo fica assim:

```
1  <script>  
2  var carros = ["Carro1" , "Carro2" , "Carro3" , "Carro4" , "Carro5"];  
3  var contador = 0;  
4  
5  while(contador<=4){  
6      carros[contador] = prompt("Informe o modelo do carro nº " + contador , "Digite aqui");  
7      contador++;  
8  }  
9  
10 contador = 0;  
11  
12 while(contador <= 4){  
13     console.log("Modelo nº " + contador + ": " + carros[contador]);  
14     contador++;  
15 }  
16 </script>
```

Salve o arquivo e abra no Google Chrome para ver o resultado, não esqueça de abrir o console.

# Dia 16

## Missão: Adicionando e removendo elementos da lista de carros

### Objetivo:

- Utilizando Array
- Métodos do Array

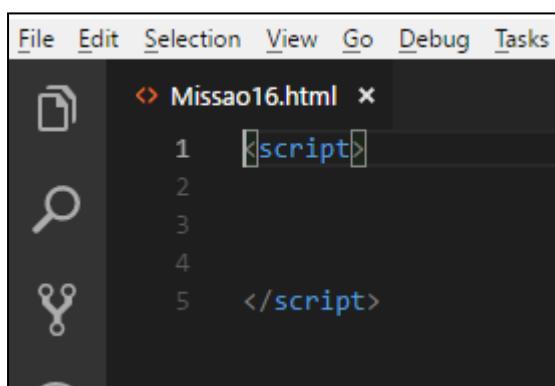
**Ao som de:** Chuck Berry - You Never Can Tell

### Descrição:

Caso ainda não tenha feito a missão do dia 15, volte e realize ela.

Abra o VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao16.html e já digite:

```
<script>  
  
</script>
```



A screenshot of the Visual Studio Code interface. The menu bar at the top includes File, Edit, Selection, View, Go, Debug, and Tasks. On the left is a sidebar with icons for file operations like New, Open, Save, Find, and Run. The main editor area shows a single line of code: <script>. This line is highlighted with a red rectangle, indicating it is the current selection or the focus of the mission. The line number 1 is also visible to the left of the code.

### Missão

Vamos criar uma lista de carros e manipular os elementos delas, removendo carros existentes e adicionando novos.

Primeiro vamos criar o array com a lista de carros. Digite:

```
var carros = ["Uno" , "Gol" , "Celta" , "Civic" , "Palio" , "Siena"];
```

Nesse momento criamos um array com o nome de carros e armazenamos todos os valores à frente dele. Se fôssemos representar o array carros graficamente seria algo parecido com:

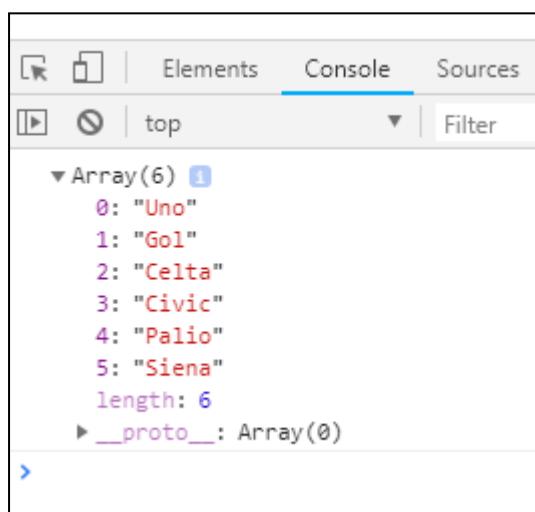
0	1	2	3	4	5
Uno	Gol	Celta	Civic	Palio	Siena

Temos então um array com 6 posições que possui índices de 0 à 5.

Uma forma que podemos fazer para visualizar o conteúdo deste array é utilizar o comando `console.dir`. Digite:

```
console.dir(carros);
```

Salve o arquivo e abra no Google Chrome para ver o resultado, não esqueça de abrir o console para ver o resultado que será algo como:



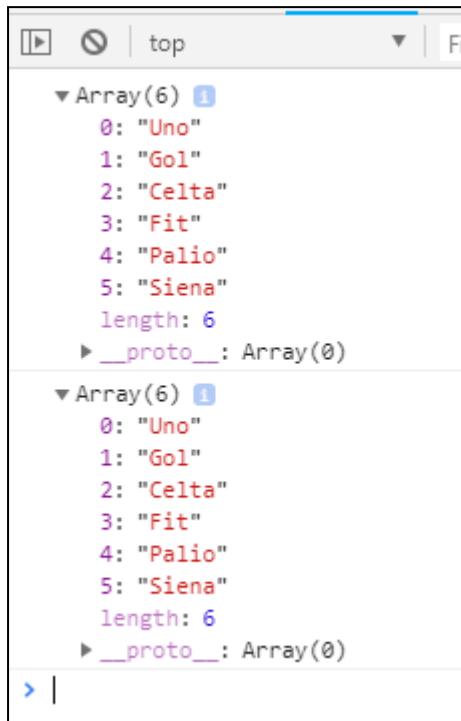
Vamos alterar o “Civic” para “Fit”. Digite:

```
carros[3] = "Fit";
```

E já utilize o `console.dir` novamente para ver o resultado. Digite:

```
console.dir(carros);
```

Salve o arquivo e abra no Google Chrome para ver o resultado, não esqueça de abrir o console para ver o resultado que será algo como:



Observe que da primeira exibição para a segunda o valor da posição 3 foi alterado. O código inteiro até o momento é:

```
<script>
var carros = ["Uno" , "Gol" , "Celta" , "Civic" , "Palio" , "Siena" ];
console.dir(carros);

carros[3] = "Fit";
console.dir(carros);

</script>
```

```
④ Missao16.html ●
1  ↗ <script>
2    var carros = ["Uno" , "Gol" , "Celta" , "Civic" , "Palio" , "Siena" ];
3    console.dir(carros);
4
5    carros[3] = "Fit";
6    |console.dir(carros);
7
8  </script>
```

Vamos remover o último valor do array carros. Digite:

```
carros.pop();
```

Veja como está o código completo:

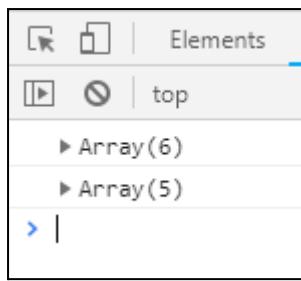
```
<script>
var carros = ["Uno" , "Gol" , "Celta" , "Civic" , "Palio" , "Siena" ];
console.dir(carros);

carros[3] = "Fit";
carros.pop();

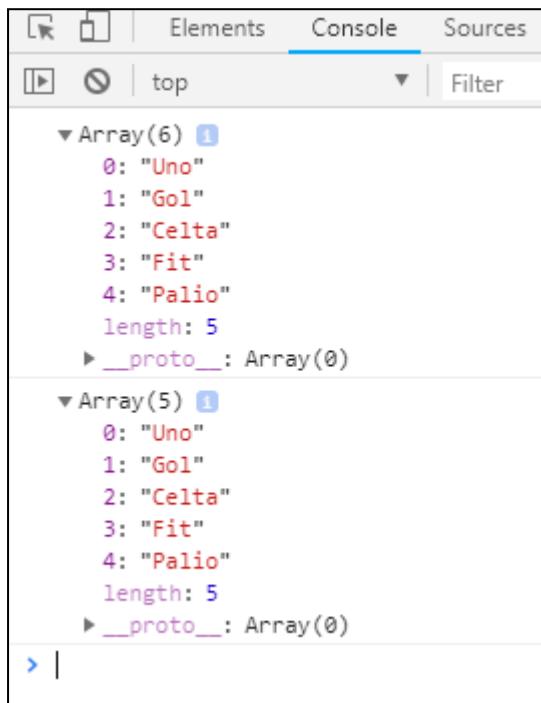
console.dir(carros);
</script>
```

```
④ Missao16.html ●
1  <script>
2    var carros = ["Uno" , "Gol" , "Celta" , "Civic" , "Palio" , "Siena" ];
3    console.dir(carros);
4
5    carros[3] = "Fit";
6    carros.pop();
7
8    console.dir(carros);
9  </script>
```

O método pop do array remove o último elemento do array. Salve e abra seu arquivo no navegador para ver algo parecido com isso:



Observe que agora a primeira exibição do array possui 6 posições e a segunda 5, pois entre as exibições foi aplicado o método pop. Click nas setinhas para expandir e ver o conteúdo dos arrays:



Agora vamos inserir mais dois valores no array carros, vamos inserir “Agile” e “Sandero”. Logo depois da linha do comando “pop” digite:

```
carros.push("Agile");
carros.push("Sandero");
```

Veja o código completo:

```
<script>
var carros = ["Uno" , "Gol" , "Celta" , "Civic" , "Palio" , "Siena" ];
console.dir(carros);

carros[3] = "Fit";
carros.pop();
```

```
carros.push("Agile");
carros.push("Sandero");

console.dir(carros);
</script>
```

```
1  <script>
2    var carros = ["Uno" , "Gol" , "Celta" , "Civic" , "Palio" , "Siena" ];
3    console.dir(carros);
4
5    carros[3] = "Fit";
6    carros.pop();
7    carros.push("Agile");
8    carros.push("Sandero");
9
10   console.dir(carros);
11  </script>
```

Salve e abra seu arquivo no navegador para ver algo parecido com isso:



O método push insere um valor ao final do array. Veja que a primeira exibição do array possui 6 posições, e a segunda 7, entre a primeira e a segunda foi aplicado o método pop,

que removeu uma posição fazendo nosso array ficar com 5 posições e depois aplicado duas vezes o método push, fazendo o array ficar com 7 posições.

# Dia 17

## Missão: Era uma vez o JavaScript

### Objetivo:

- Conhecendo mais sobre o JavaScript
- Métodos do Array

**Ao som de:** Rend Collective - Build Your Kingdom Here

### Descrição:

**Caso ainda não tenha feito a missão do dia 16, volte e realize ela.**

Abra o VS Code, e não aperte nada, a missão de hoje é apenas conhecer um pouco mais sobre JavaScript e suas possibilidades e aplicações.

### Missão

**Primeiro, JavaScript é DIFERENTE de Java.**

Você pode estar se perguntando: "por que se comenta tanto sobre JavaScript hoje em dia? Por que essa linguagem ganhou tanta popularidade, e por que eu, desenvolvedor, tenho que aprendê-la?"

Para iniciarmos nossa conversa sobre JavaScript, vale avisar que ela é a **linguagem de programação nativa da Web**. Qualquer navegador que você esteja utilizando, incluindo o Google Chrome, com certeza dá suporte ao JS, afinal ele é utilizado para dar dinamismo a nossas páginas. Por exemplo, quando clicamos na ferramenta de busca do em algum site, o fato de que seja detectado o evento de clique no ícone da lupa e, em seguida, ocorra uma pequena animação para exibir a caixa de busca, é realizável com JS.

Talvez você tenha conhecimento maior sobre HTML e CSS e garanta que seja possível fazer o mesmo usando o transform, já que também podemos criar algumas animações apenas com CSS puro. Porém, certas ações como algumas validações específicas nos formulários, como aquelas realizadas nos campos de "nome" e "e-mail", não são possibilidades unicamente com HTML. Esse tipo de dinamismo e interatividade que temos atualmente nos formulários e nas páginas Web vieram com o JavaScript.

Graças ao JS, aumentamos a usabilidade do usuário nas páginas web. Um exemplo é quando clicamos em alguma categoria no menu de algum site, ele filtra os conteúdos. Isso só é possível com JavaScript.

É uma linguagem bastante poderosa que permite que o usuário consiga **interagir** com a página. Isto já seria um excelente motivo para despertar seu interesse pela linguagem, mas além de ser dominante nos cenários dos navegadores, o JavaScript recentemente também se expandiu para o lado do servidor com o lançamento do **Node.js**, um ambiente que permite criarmos um servidor Web completo utilizando JavaScript. Desta forma, a linguagem ganhou grande popularidade e caiu no gosto dos desenvolvedores, tornando-se parte do coletivo imaginário de qualquer programador moderno.

O Javascript não "ganhou terreno" apenas no servidor, sendo possível programarmos um Arduíno com a biblioteca **Johnny-Five**, criarmos um aplicativo para Desktop utilizando o framework **Electron**, e até mesmo encontrarmos um banco de dados como o **MongoDB**, que utiliza a sintaxe do JavaScript em suas *queries*.

O JavaScript está presente em diversas áreas do desenvolvimento, ampliando as habilidades de desenvolvedor para os mais diversos ramos da tecnologia, e é claro, controlando e manipulando qualquer página Web!

Para finalizar, assista o vídeo abaixo até o minuto 10:00, até esse minuto é explicado a origem do JavaScript.

<https://www.youtube.com/watch?v=jXPU0uAzmYY>

# **Dia 18**

## **Missão: Dicionário de animais**

### **Objetivo:**

- Array
- Laço de repetição
- Estrutura de decisão

**Ao som de:** Taylor Swift - Delicate

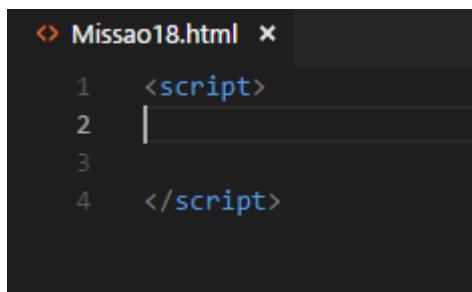
### **Descrição:**

**Caso ainda não tenha feito a missão do dia 17, volte e realize ela.**

Abra o VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao18.html e já digite:

```
<script>
```

```
</script>
```



```
Missao18.html ✘
1  <script>
2  |
3
4  </script>
```

### **Missão**

Vamos criar um programa onde o usuário escolhe um animal e são exibidas algumas informações sobre esse animal.

Vamos declarar as variáveis que vão controlar a exibição, armazenar a opção do animal a ser exibido e o array que vai armazenar as informações dos animais. Digite:

```
var continuar = "S";
var animalExibir;
var animais = new Array(4);
```

A linha `var animais = new Array(4);` cria um array chamado animais com 4 posições. Vamos agora colocar informações no array animais. Digite:

```
animais[0] = "Cuidado, a picada de uma radioativa pode te transformar em
super-herói";
animais[1] = "Escorpião é o oitavo signo astrológico do zodíaco";
animais[2] = "Cobra é uma denominação genérica, utilizada frequentemente na
língua portuguesa como sinônimo para serpente."
animais[3] = "Na linguagem vulgar, chama-se lagarta ao primeiro estágio
larval dos insetos da ordem dos Lepidoptera.";
```

Vamos criar o laço de exibição e a estrutura de decisão que vai verificar a opção escolhida para exibir as informações dos animais. Digite:

```
while(continuar == "S"){
    console.clear();

    animalExibir = prompt("Escolha uma opção abaixo:\n[0] Aranha \n[1]
Escorpião \n[2] Cobra \n[3] Lagarto", "Digite aqui");

    switch(animalExibir){
        case "0":
            console.log("Aranha");
            console.log(animais[animalExibir]);
            break;

        case "1":
            console.log("Escorpião");
            console.log(animais[animalExibir]);
            break;

        case "2":
            console.log("Cobra");
            console.log(animais[animalExibir]);
            break;

        case "3":
            console.log("Lagarto");
            console.log(animais[animalExibir]);
```

```

        break;

    default:
        console.log("Opção invalida");
        break;
    }

    continuar = prompt("Deseja ver outro animal? [S]im / [N]ão", "Digite
aqui");
}

```

E por fim, vamos limpar o console exibir uma mensagem para finalizar o programa. Digite:

```

console.clear();
console.log("Obrigado!");

```

O código inteiro fica assim:

```

<script>
var continuar = "S";
var animalExibir;
var animais = new Array(4);

animais[0] = "Cuidado, a picada de uma radioativa pode te transformar em
super-herói";
animais[1] = "Escorpião é o oitavo signo astrológico do zodíaco";
animais[2] = "Cobra é uma denominação genérica, utilizada frequentemente na
língua portuguesa como sinônimo para serpente."
animais[3] = "Na linguagem vulgar, chama-se lagarta ao primeiro estágio
larval dos insetos da ordem dos Lepidoptera.";

while(continuar == "S"){
    console.clear();

    animalExibir = prompt("Escolha uma opção abaixo:\n[0] Aranha \n[1]
Escorpião \n[2] Cobra \n[3] Lagarta", "Digite aqui");

    switch(animalExibir){
        case "0":
            console.log("Aranha");
            console.log(animais[animalExibir]);
            break;

        case "1":
            console.log("Escorpião");
            console.log(animais[animalExibir]);
    }
}

```

```

        break;

    case "2":
        console.log("Cobra");
        console.log(animais[animalExibir]);
        break;

    case "3":
        console.log("Lagarto");
        console.log(animais[animalExibir]);
        break;

    default:
        console.log("Opção invalida");
        break;
    }

    continuar = prompt("Deseja ver outro animal? [S]im / [N]ão", "Digite aqui");
}

console.clear();
console.log("Obrigado!");

</script>

```

```

1 <script>
2 var continuar = "S";
3 var animalExibir;
4 var animais = new Array(4);
5
6 animais[0] = "Cuidado, a picada de uma radioativa pode te transformar em super-herói";
7 animais[1] = "Escorpião é o oitavo signo astrológico do zodíaco";
8 animais[2] = "Cobra é uma denominação genérica, utilizada frequentemente na língua portuguesa como sinônimo para serpente.";
9 animais[3] = "Na linguagem vulgar, chama-se lagarta ao primeiro estágio larval dos insetos da ordem dos Lepidoptera.";
10
11 while(continuar == "S"){
12     console.clear();
13
14     animalExibir = prompt("Escolha uma opção abaixo:\n[0] Aranha \n[1] Escorpião \n[2] Cobra \n[3] Lagarta", "Digite aqui");
15
16     switch(animalExibir){
17         case "0":
18             console.log("Aranha");
19             console.log(animais[animalExibir]);
20             break;
21
22         case "1":
23             console.log("Escorpião");
24             console.log(animais[animalExibir]);
25             break;
26
27         case "2":
28             console.log("cobra");
29             console.log(animais[animalExibir]);
30             break;
31
32         case "3":
33             console.log("Lagarto");
34             console.log(animais[animalExibir]);
35             break;
36
37         default:
38             console.log("Opção invalida");
39             break;
40     }
41
42     continuar = prompt("Deseja ver outro animal? [S]im / [N]ão", "Digite aqui");
43 }
44
45 console.clear();
46 console.log("Obrigado!");
47
48 </script>

```

Salve o arquivo e abra no Google Chrome para ver o resultado, não esqueça de abrir o console.

# **Dia 19**

## **Missão: Dicionário de animais**

### **Objetivo:**

- Array
- Laço de repetição

**Ao som de:** Justin Timberlake - CAN'T STOP THE FEELING

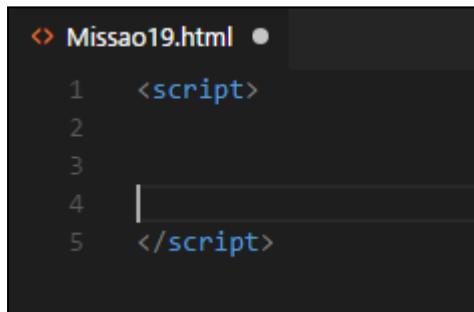
### **Descrição:**

**Caso ainda não tenha feito a missão do dia 18, volte e realize ela.**

Abra o VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao19.html e já digite:

```
<script>
```

```
</script>
```



```
↳ Missao19.html •
1  <script>
2
3
4  |
5  </script>
```

### **Missão**

Vamos criar um programa para o controle de cursos, onde o usuário deve informar o nome do curso, o código do curso e o conteúdo programático para o curso. No caso do conteúdo programático, o usuário pode informar quantas vezes quiser.

Vamos declarar as variáveis que vão armazenar o nome do curso, o código do curso, a variável de controle para informar um novo conteúdo, uma variável para armazenar temporariamente o conteúdo, o array de conteúdos e a variável de contador. Digite:

```
var nomeCurso = prompt("Informe o nome do curso" , "Digite aqui");
var codigoCurso = prompt("Informe o código do curso" , "Digite aqui");
var novoConteudo = "S";
var conteudoTemp = "";
var conteudoCurso = [];
var contador = 0;
```

Vamos criar o laço de repetição que vai pedir as informações de conteúdo. Digite:

```
while(novoConteudo == "S"){
    conteudoTemp = prompt("Informe o conteúdo do curso" , "Digite aqui");
    conteudoCurso.push(conteudoTemp);
    novoConteudo = prompt("Deseja cadastrar um novo conteúdo para o curso?[S]im / [N]ão" , "Digite aqui");
}
```

No laço acima, o conteúdo é armazenado em uma variável e depois inserido no array através da linha `conteudoCurso.push(conteudoTemp);`

Agora crie as instruções para exibir as informações do curso e o laço que vai percorrer o array exibindo o conteúdo do curso. Digite:

```
console.clear();
console.log("Código: " + codigoCurso);
console.log("Curso: " + nomeCurso);
console.log("Conteúdo: ");
while(contador < conteudoCurso.length){
    console.log(conteudoCurso[contador]);
    contador++;
}
```

O código completo fica assim:

```
<script>
var nomeCurso = prompt("Informe o nome do curso" , "Digite aqui");
var codigoCurso = prompt("Informe o código do curso" , "Digite aqui");
var novoConteudo = "S";
var conteudoTemp = "";
var conteudoCurso = [];
var contador = 0;
```

```

while(novoConteudo == "S"){
    conteudoTemp = prompt("Informe o conteúdo do curso" , "Digite aqui");
    conteudoCurso.push(conteudoTemp);
    novoConteudo = prompt("Deseja cadastrar um novo conteúdo para o
curso?[S]im / [N]ão" , "Digite aqui");
}

console.clear();
console.log("Código: " + codigoCurso);
console.log("Curso: " + nomeCurso);
console.log("Conteúdo: ");
while(contador < conteudoCurso.length){
    console.log(conteudoCurso[contador]);
    contador++;
}
</script>

```

```

1 <script>
2 var nomeCurso = prompt("Informe o nome do curso" , "Digite aqui");
3 var codigoCurso = prompt("Informe o código do curso" , "Digite aqui");
4 var conteudoCurso = [];
5 var novoConteudo = "S";
6 var conteudoTemp = "";
7 var contador = 0;
8 while(novoConteudo == "S"){
9     conteudoTemp = prompt("Informe o conteúdo do curso" , "Digite aqui");
10    conteudoCurso.push(conteudoTemp);
11    novoConteudo = prompt("Deseja cadastrar um novo conteúdo para o curso?[S]im / [N]ão" , "Digite aqui");
12 }
13
14 console.clear();
15 console.log("Código: " + codigoCurso);
16 console.log("Curso: " + nomeCurso);
17 console.log("Conteúdo: ");
18 while(contador < conteudoCurso.length){
19     console.log(conteudoCurso[contador]);
20     contador++;
21 }
22 </script>

```

Salve o arquivo e abra no Google Chrome para ver o resultado, não esqueça de abrir o console.

# Dia 20

## Missão: Exibir os alunos de uma lista

### Objetivo:

- Laço de repetição(for)

**Ao som de:** Charlie Puth - We Don't Talk Anymore (feat. Selena Gomez)

### Descrição:

Caso ainda não tenha feito a missão do dia 19, volte e realize ela.

Abra o VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao20Exemplo.html e já digite:

```
<script>
```

```
</script>
```

The screenshot shows a dark-themed code editor window titled "Missao20Exemplo.html". It contains a single line of code: "<script>". The cursor is positioned at the end of the line, indicated by a vertical bar. The code editor has a minimalist interface with a black background and light-colored text.

### Mais um laço de repetição

Assim como temos algumas estruturas de decisão (vimos o if e o switch) temos também diversos laços de repetição:

- for
- for...in
- while

- do..while

Qualquer problema que você resolve com um laço é possível resolver com outro, claro que pode ser que tenha uma solução “mais fácil” dependendo do problema e do laço utilizado, mas em geral você pode utilizar o que você conseguir para resolver o problema. Nós já estudamos o laço while e agora vamos ver o laço for.

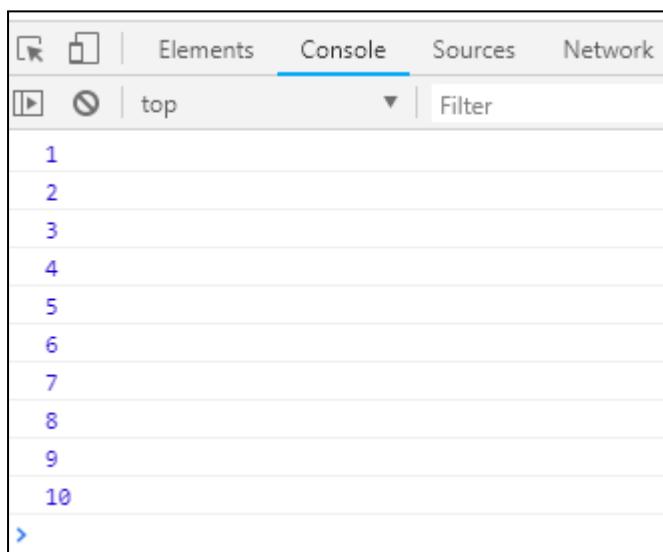
## Laço de repetição FOR

Vamos criar um exemplo para ver o laço `for`. Digite:

```
for(contador = 1 ; contador <= 10 ; contador++){
    console.log(contador);
}
```

```
1  <script>
2
3  for(contador = 1 ; contador <= 10 ; contador++){
4      console.log(contador);
5
6
7  </script>
```

Salve o arquivo e abra no Google Chrome, abra o console e veja o resultado:



O laço `for` que criamos irá executar 10 vezes e em cada uma das vezes vai exibir o conteúdo da variável `contador`. Vamos entender o laço `for`:

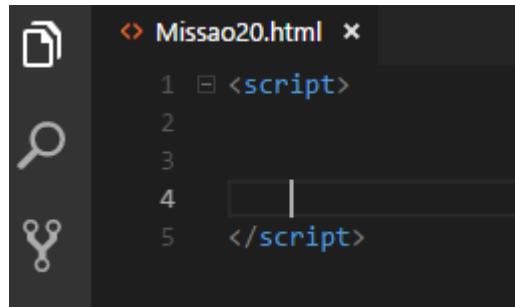
```
for( declararação_de_variável ; condição ; incremento ){  
}
```

**declaração de variável** → contador = 1  
**condição** → contador <= 10  
**incremento** → contador++

## Missão

Crie um arquivo novo no VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao20.html e já digite:

```
<script>  
  
</script>
```



Nossa missão será criar um array com o nome dos alunos de um curso e exibir o nome de cada um dos alunos no console.

Fizemos algo muito parecido em um exemplo na missão 15. Veja:

```
1 <script>  
2   var alunos = ["Maria" , "Rose" , "Claudio" , "Renata" , "Fernando"];  
3   var contador = 0;  
4   |  
5   while(contador<=4){  
6     |   console.log(alunos[contador]);  
7     |   contador++;  
8   }  
9   </script>
```

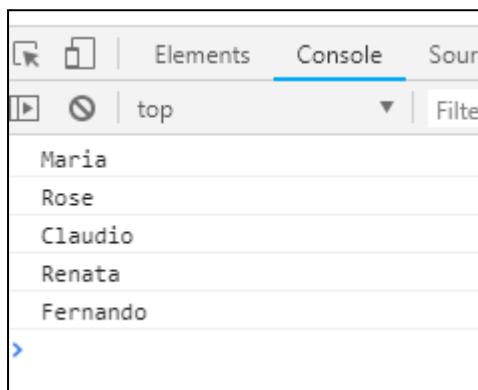
Iremos fazer a mesma coisa, mas utilizando o laço **for** para evidenciar as diferenças.

Digite:

```
var alunos = ["Maria" , "Rose" , "Claudio" , "Renata" , "Fernando"];
for(contador = 0; contador < 5 ; contador++){
    console.log(alunos[contador]);
}
```

Primeiro temos a variável **contador**. A condição diz que o laço vai se repetir enquanto **contador for menor que 5**, pois **5 é a quantidade de posições do array alunos**. E a **cada repetição do laço** será executado à instrução **contador++** que fará com que o valor de contador seja aumentado de um por um.

Salve, abra no Google Chrome e visualize o resultado no console:



Veja o código inteiro:

```
1 ① <script>
2  var alunos = ["Maria" , "Rose" , "Claudio" , "Renata" , "Fernando"];
3 ② for(contador = 0; contador < 5 ; contador++){
4      |   console.log(alunos[contador]);
5  }
6  </script>
```

Compare agora com o exemplo da missão 15 e veja como esse código ficou mais enxuto, porém, para quem está iniciando, talvez um pouco mais complexo:

```
1 <script>
2   var alunos = ["Maria" , "Rose" , "Claudio" , "Renata" , "Fernando"];
3   var contador = 0;
4   |
5   while(contador<=4){
6     console.log(alunos[contador]);
7     contador++;
8   }
9 </script>
```

Um bom exercício é revisitar as missões anteriores e tentar utilizar o laço **for** para resolver as missões que envolverem laço de repetição.

# Dia 21

## Missão: Criando uma agenda de contatos

### Objetivo:

- Laço de repetição(for)

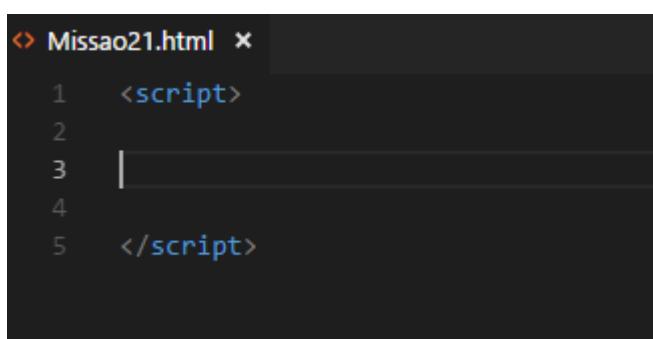
**Ao som de:** The Weeknd, Kendrick Lamar - Pray For Me

### Descrição:

**Caso ainda não tenha feito a missão do dia 20, volte e realize ela.**

Abra o VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao21.html e já digite:

```
<script>  
</script>
```



The screenshot shows a dark-themed code editor window titled "Missao21.html". It contains five lines of code, each preceded by a line number (1, 2, 3, 4, 5). The code consists of a single opening script tag on line 1 and a closing script tag on line 5, with a cursor positioned between them on line 3.

```
1  <script>  
2  
3  
4  
5  </script>
```

### Missão

A missão consiste em criar uma agenda de contatos que contenham os seguintes dados: nome, telefone e cidade. Para cada parte de informação teremos um array diferente, então nosso código terá os arrays: nomes, telefones e cidades. Observe os exemplos:

## Nomes

0	1	2
João	Maria	Joana

## Telefones

0	1	2
99111-1111	99222-2222	99333-3333

## Cidades

0	1	2
Americana	Piracicaba	Campinas

A ligação entre as informações será através do índice, por exemplo: o contato 0 é o João, com o telefone 99111-1111 da cidade Americana.

Vamos começar o código declarando as variáveis e arrays que vamos utilizar. Digite:

```
var nomes = [];
var telefones = [];
var cidades = [];
var novoContato = "S";
var contador = 0;
```

Agora, vamos criar o laço que pega as informações e popula os arrays. Digite:

```
while(novoContato == "S"){
    nomes[contador] = prompt("Informe o nome","");
    telefones[contador] = prompt("Informe o telefone","");
    cidades[contador] = prompt("Informe a cidade","");
    novoContato = prompt("Cadastrar novo contato?[S]im / [N]ao","");
    contador++;
}
```

Por fim, crie o laço que exibe as informações cadastradas. Digite:

```

for(i=0;i < nomes.length ; i++){
    console.log("Nome: " + nomes[i]);
    console.log("Telefone: " + telefones[i]);
    console.log("Cidade: " + cidades[i]);
    console.log("-----");
}

```

Veja o código inteiro:

```

<script>
var nomes = [];
var telefones = [];
var cidades = [];
var novoContato = "S";
var contador = 0;

while(novoContato == "S"){
    nomes[contador] = prompt("Informe o nome","");
    telefones[contador] = prompt("Informe o telefone","");
    cidades[contador] = prompt("Informe a cidade","");
    novoContato = prompt("Cadastrar novo contato?[S]im / [N]ao","");
    contador++;
}

for(i=0;i < nomes.length ; i++){
    console.log("Nome: " + nomes[i]);
    console.log("Telefone: " + telefones[i]);
    console.log("Cidade: " + cidades[i]);
    console.log("-----");
}
</script>

```

```
1 <script>
2 var nomes = [];
3 var telefones = [];
4 var cidades = [];
5 var novoContato = "S";
6 var contador = 0;
7
8 while(novoContato == "S"){
9     nomes[contador] = prompt("Informe o nome","");
10    telefones[contador] = prompt("Informe o telefone","");
11    cidades[contador] = prompt("Informe a cidade","");
12    novoContato = prompt("Cadastrar novo contato?[S]im / [N]ao","");
13    contador++;
14 }
15
16 for(i=0;i < nomes.length ; i++){
17     console.log("Nome: " + nomes[i]);
18     console.log("Telefone: " + telefones[i]);
19     console.log("Cidade: " + cidades[i]);
20     console.log("-----");
21 }
22 </script>
```

Salve, abra no Google Chrome, preencha as informações e visualize no console as informações.

# **Dia 22**

## **Missão: Escrevendo na tela**

### **Objetivo:**

- Laço de repetição(for)

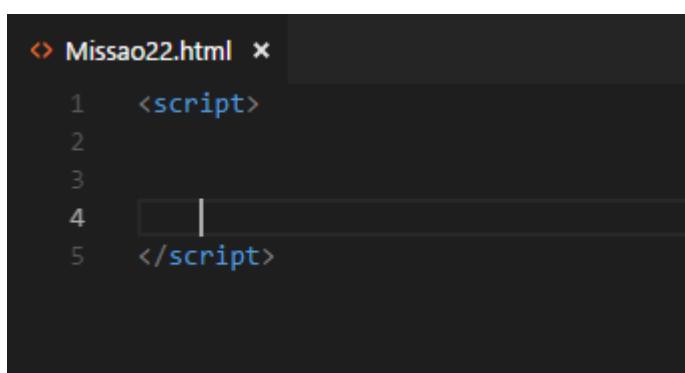
**Ao som de:** Drake - In My Feelings

### **Descrição:**

**Caso ainda não tenha feito a missão do dia 21, volte e realize ela.**

Abra o VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao22.html e já digite:

```
<script>  
</script>
```



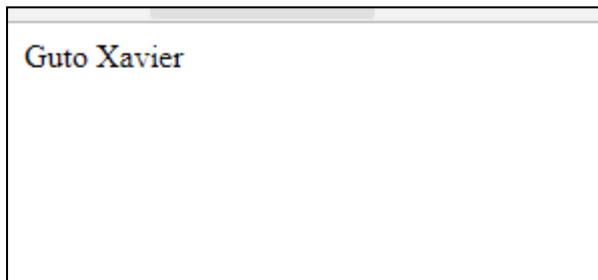
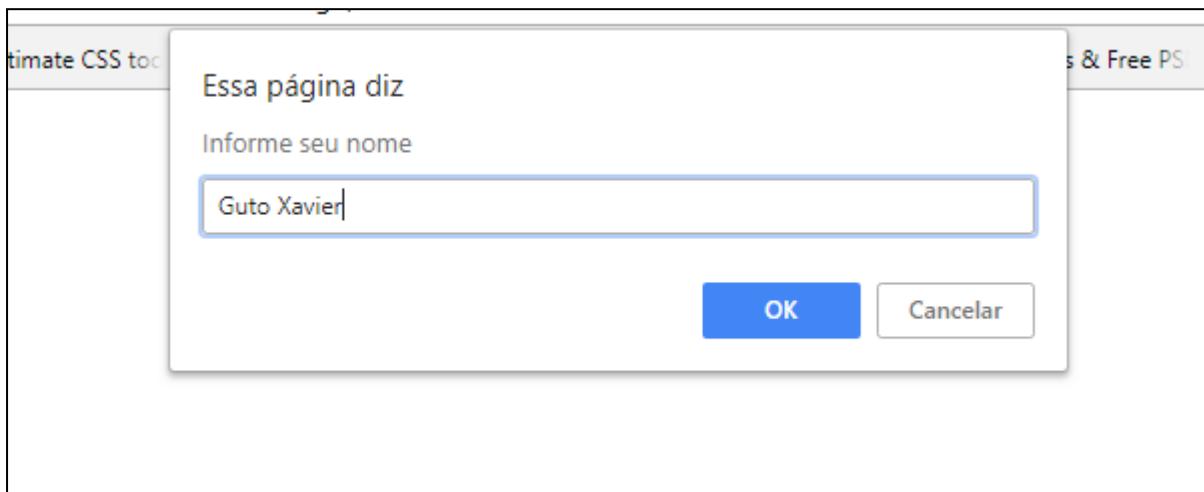
A screenshot of the VS Code editor showing a single file named "Missao22.html". The code consists of two lines: "<script>" on line 1 and "</script>" on line 5. The cursor is positioned between the closing brace of the first line and the opening brace of the second line. The background of the code editor is dark.

### **Missão**

Todas as saídas de dados que trabalhamos até agora foram no console ou utilizando alert. Agora vamos pegar informações do usuário e escrever dentro de sua página html. Para fazer isso vamos utilizar o comando document.write(). Digite:

```
var nome = prompt("Informe seu nome","Digite aqui");
document.write(nome);
```

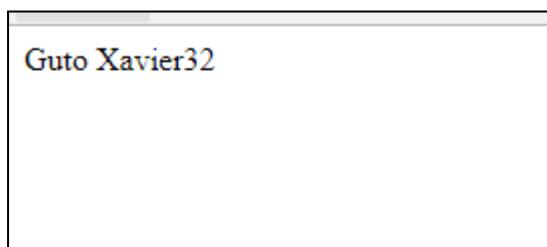
Salve, abra no Google Chrome, preencha as informações e visualize o seguinte resultado:



Agora vamos pedir a idade do usuário. Digite:

```
var idade = prompt("Informe sua idade","Digite aqui");
document.write(idade);
```

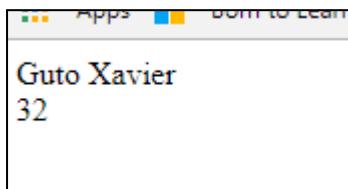
Salve e veja o resultado:



Veja que o nome e a idade ficaram “colados”, vamos colocar uma informação em cada linha. Para isso insira a linha a seguir antes da linha `document.write(idade);`. Digite:

```
document.write("<br>");
```

Salve e veja o resultado:



Veja o código inteiro:

```
<script>
var nome = prompt("Informe seu nome","Digite aqui");
document.write(nome);

var idade = prompt("Informe sua idade","Digite aqui");
document.write("<br>");
document.write(idade);
</script>
```

```
1 <script>
2 var nome = prompt("Informe seu nome","Digite aqui");
3 document.write(nome);
4
5 var idade = prompt("Informe sua idade","Digite aqui");
6 document.write("<br>");
7 document.write(idade);
8 </script>|
```

E podemos incrementar um pouco mais nosso código:

```
1 <script>
2 var nome = prompt("Informe seu nome","Digite aqui");
3 document.write("Nome: " + nome);
4
5 var idade = prompt("Informe sua idade","Digite aqui");
6 document.write("<br>");
7 document.write("Idade: " + idade);
8 </script>
```



# Dia 23

## Missão: Refinando nosso cálculo de média

### Objetivo:

- Laço de repetição(for)

**Ao som de:** Shinedown - Simple Man

### Descrição:

**Caso ainda não tenha feito a missão do dia 22, volte e realize ela.**

Abra o VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao23.html e já digite:

```
<script>  
</script>
```

A screenshot of the VS Code interface showing a single line of code in a dark-themed editor. The file is named 'Missao23.html'. The code consists of a single line starting with the opening tag '<script>'.

### Missão

Vamos pedir o nome do aluno e 4 notas para calcular a média, porém vamos verificar se cada uma das notas digitadas é realmente um número, caso não seja iremos pedir que o

usuário informe a nota novamente. Também iremos utilizar um comando para limitar a média à uma casa decimal(e não mais ficar apresentando resultados tipo 7.899999999999999). Vamos declarar as variáveis que irão armazenar a nota, a média e nosso contador, e depois disso já vamos pedir o nome do aluno. Digite:

```
var media = 0, nota = 0, contador = 0;
var nome = prompt("Informe o nome do aluno","Digite aqui!!!");
```

Nada de novo até aqui, agora criamos o laço de repetição que “prenderá” o usuário até que ele digite 4 notas válidas(números). Digite:

```
while(contador < 4){
    nota = prompt("Informe a nota do aluno","Digite aqui");
    if(isNaN(nota)){
        alert("Voce nao digitou um numero valido");
    }else{
        media = parseFloat(media) + parseFloat(nota);
        contador++;
    }
}
```

A novidade aqui é a função `isNaN(nota)`, essa função verifica se o conteúdo dentro dos parênteses **Não É um Número(isNaN = is Not a Number)**, caso não seja um número ele executa as instruções da parte `true` do `if`, ou seja exibe a mensagem “*Voce nao digitou um numero valido*”. Se for um número, são executadas as instruções da parte `false` do `if`, ou seja, o novo valor é somado à variável media e o contador é incrementado em 1. Vamos agora fazer o cálculo da média e apresentar as informações na página. Digite:

```
media = media / 4;

document.write("A média de " + nome);
document.write(" é " + media.toFixed(1));
```

Nessa parte do código a novidade é o `toFixed(1)`, que faz com que o conteúdo da variável media seja apresentado utilizando apenas 1 casa decimal.

Veja o código inteiro:

```
<script>

var media = 0, nota = 0, contador = 0;
var nome = prompt("Informe o nome do aluno","Digite aqui!!!");

while(contador < 4){
```

```

nota = prompt("Informe a nota do aluno","Digite aqui");
if(isNaN(nota)){
    alert("Voce nao digitou um numero valido");
}else{
    media = parseFloat(media) + parseFloat(nota);
    contador++;
}
}

media = media / 4;

document.write("A média de " + nome);
document.write(" é " + media.toFixed(1));
</script>

```

```

1 <script>
2
3 var media = 0, nota = 0, contador = 0;
4 var nome = prompt("Informe o nome do aluno","Digite aqui!!!");
5
6 while(contador < 4){
7     nota = prompt("Informe a nota do aluno","Digite aqui");
8     if(isNaN(nota)){
9         alert("Voce nao digitou um numero valido");
10    }else{
11        media = parseFloat(media) + parseFloat(nota);
12        contador++;
13    }
14 }
15
16 media = media / 4;
17
18 document.write("A média de " + nome);
19 document.write(" é " + media.toFixed(1));
20 </script>

```

Salve, abra no Google Chrome, preencha as informações e visualize o resultado.

# **Dia 24**

## **Missão: Tornando a cálculo da média reutilizável**

### **Objetivo:**

- Criando funções personalizadas

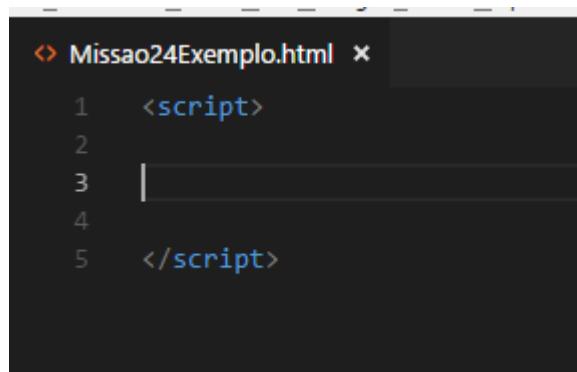
**Ao som de:** Vance Joy - 'Riptide'

### **Descrição:**

**Caso ainda não tenha feito a missão do dia 23, volte e realize ela.**

Abra o VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao24Exemplo.html e já digite:

```
<script>  
|</script>
```



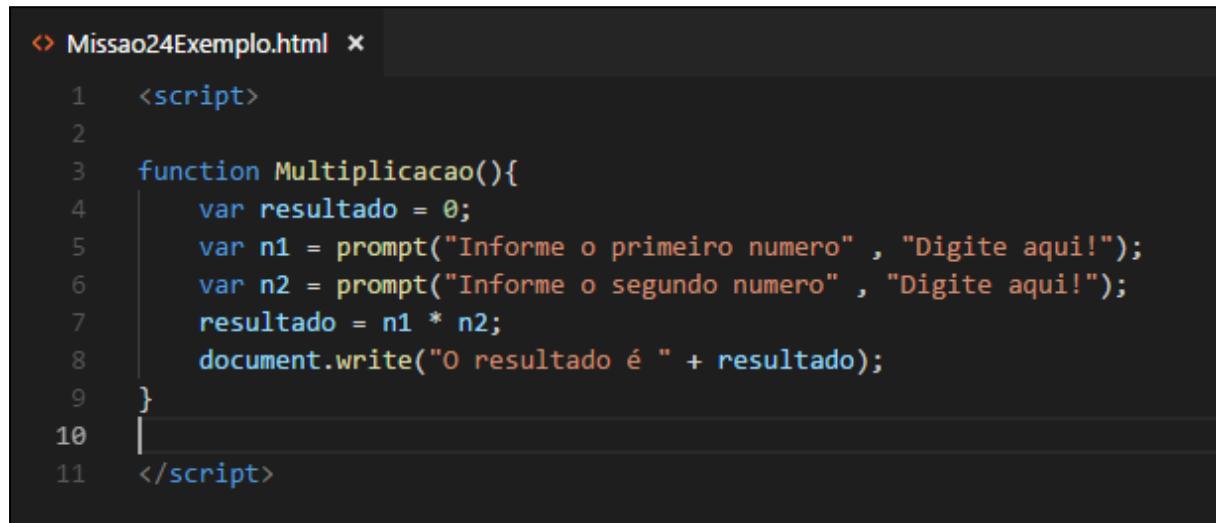
```
Missao24Exemplo.html  
1  <script>  
2  |  
3  |  
4  |  
5  </script>
```

Nós já fizemos algumas missões onde o objetivo era criar o cálculo da média, porém o problema que temos é de esses códigos não serem reutilizáveis, e toda vez que queremos calcular uma média diferente precisamos recarregar a página. Isso ainda acontece pois não trabalhamos com funções!

Funções, de um jeito bem simples, é quando você pega um conjunto de instruções e dá um nome, de modo que toda vez que você chame por esse nome sejam executadas aquelas instruções.

Vamos criar um exemplo: precisamos efetuar um cálculo de multiplicação entre dois números. Digite:

```
function Multiplicacao(){
    var resultado = 0;
    var n1 = prompt("Informe o primeiro numero" , "Digite aqui!");
    var n2 = prompt("Informe o segundo numero" , "Digite aqui!");
    resultado = n1 * n2;
    document.write("O resultado é " + resultado);
}
```



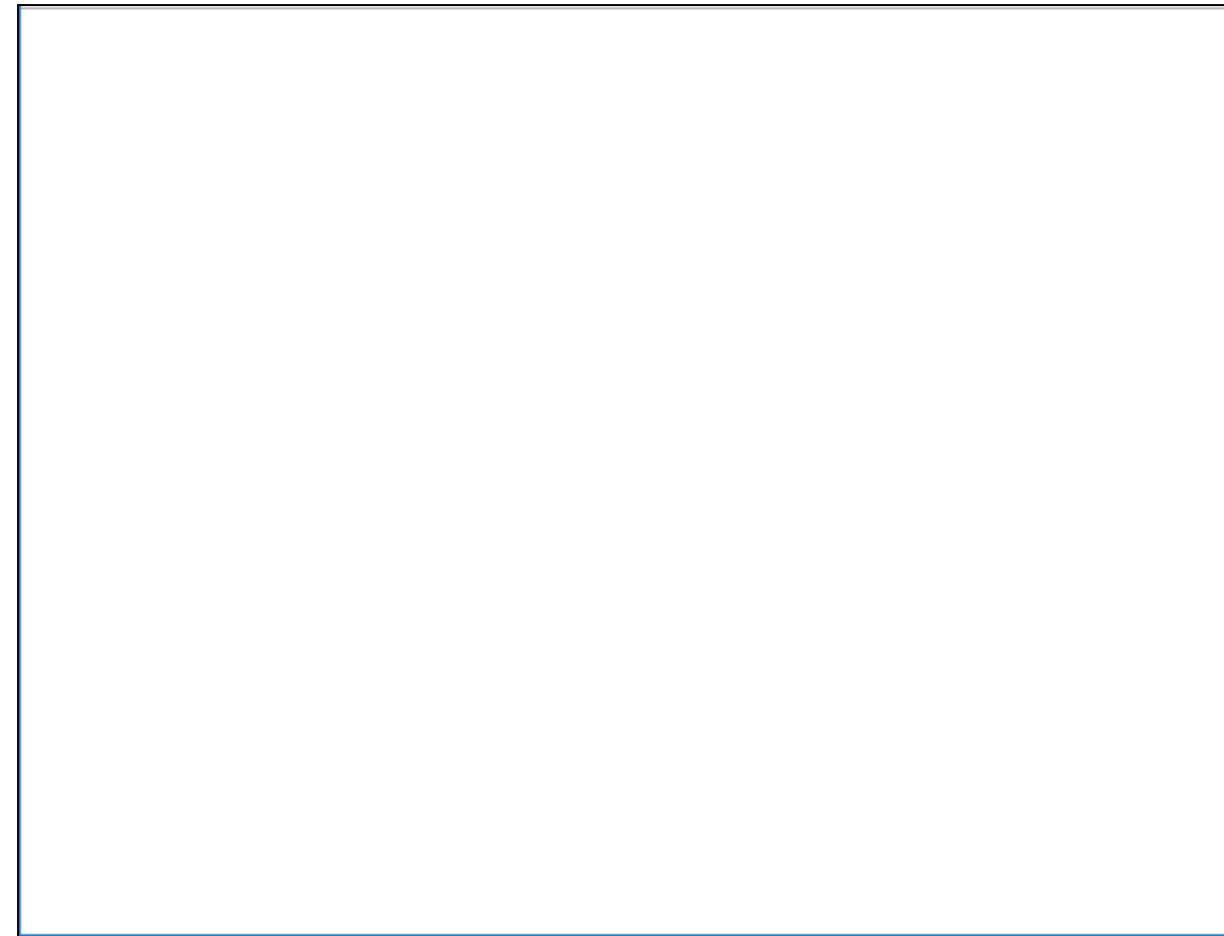
The screenshot shows a code editor window with the title 'Missao24Exemplo.html'. The code is a JavaScript function named 'Multiplicacao' that prompts the user for two numbers, multiplies them, and then writes the result back to the page. The code is numbered from 1 to 11.

```
1 <script>
2
3 function Multiplicacao(){
4     var resultado = 0;
5     var n1 = prompt("Informe o primeiro numero" , "Digite aqui!");
6     var n2 = prompt("Informe o segundo numero" , "Digite aqui!");
7     resultado = n1 * n2;
8     document.write("O resultado é " + resultado);
9 }
10
11 </script>
```

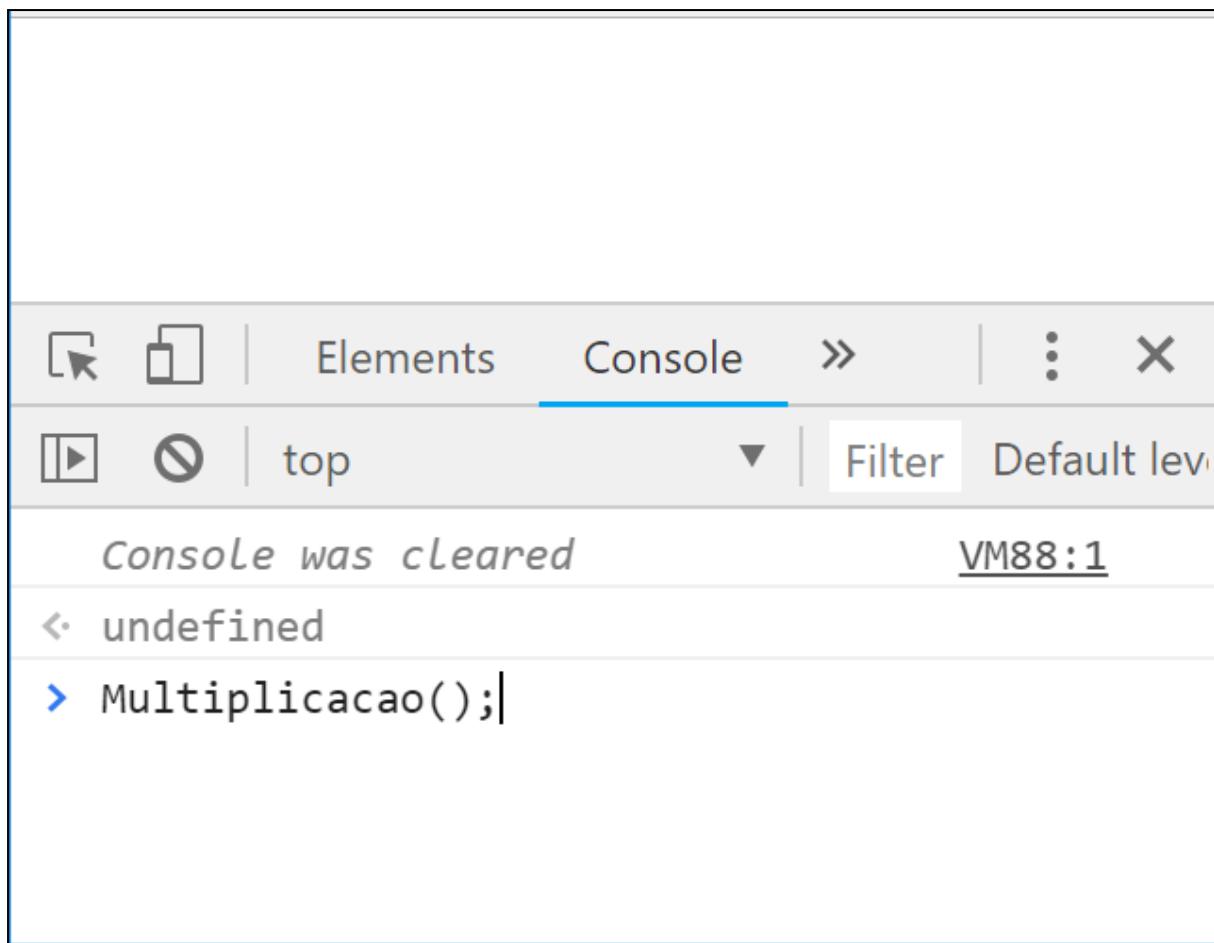
O que temos de novo é a function. A sintaxe para se criar uma função é a seguinte:

```
function NomeDaFunção(){
    instruções;
    instruções;
    instruções;
}
```

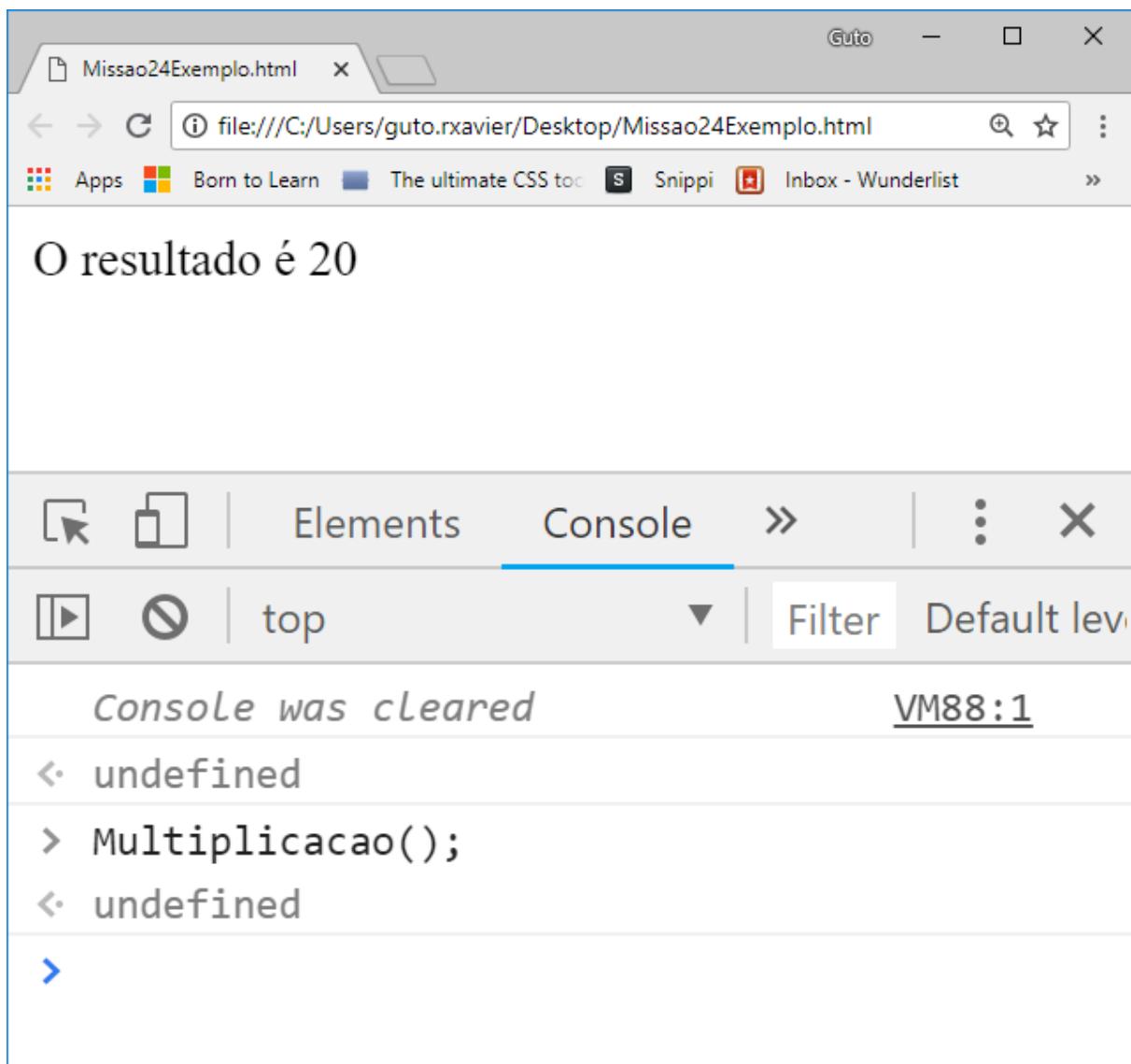
As regras para o nome da função são basicamente os mesmos de variáveis. Salve seu arquivo e abra no Google Chrome, e veja que nada vai acontecer:



Em todos nossos outros exemplos, quando você abria o arquivo o código já era executado automaticamente, dessa vez não foi pois todo o código está dentro de uma função, e para esse código ser executado você deve chamar essa função. Abra o console e digite o nome da função e pressione enter:

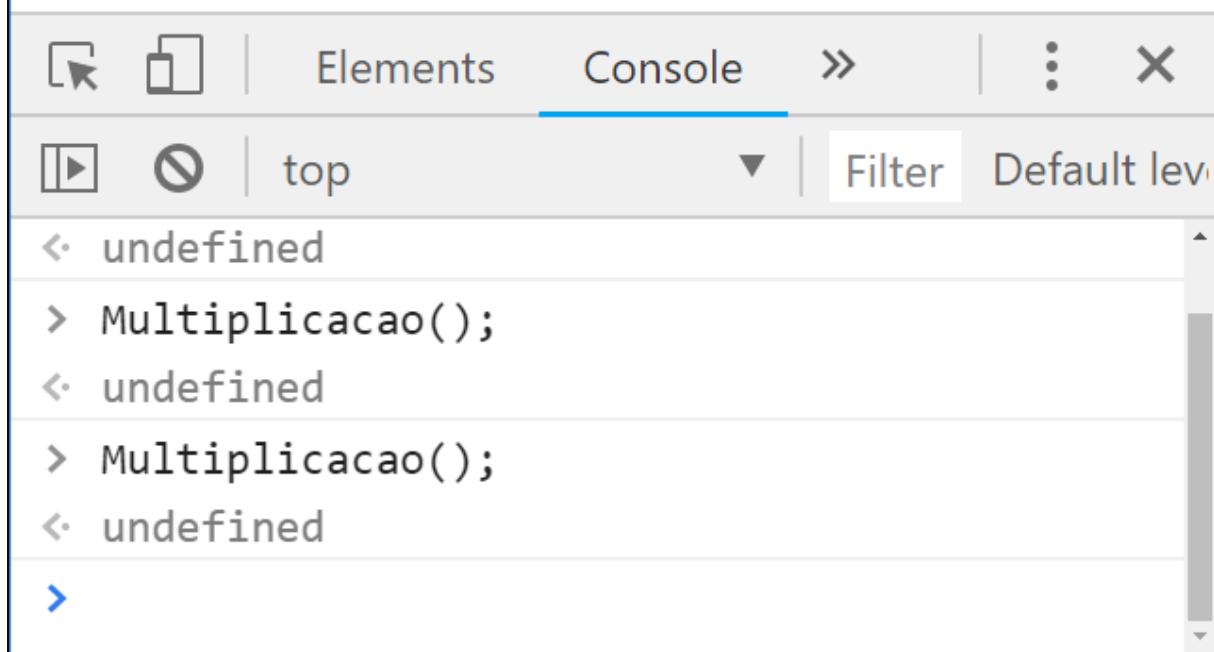


Veja que agora sim seu código foi executado, os dois números foram pedidos e o resultado apareceu na tela:



O legal é que agora, sempre que precisar fazer o cálculo é só “chamar” a função.

O resultado é 20O resultado é 60



## Missão

Abra o VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao24.html e já digite:

```
<script>
```

```
</script>
```

A screenshot of a VS Code code editor window. The title bar says 'Missao24.html \*'. The editor shows five lines of code:

```
1 <script>
2
3
4
5 </script>
```

The cursor is positioned at the end of the fifth line.

Vamos pedir as notas e calcular a média, vamos colocar essa lógica dentro de uma função.  
Digite:

```
function Media(){
    var media = 0, nota = 0, contador = 0;
    var nome = prompt("Informe o nome do aluno","Digite aqui!!!");

    while(contador < 4){
        nota = prompt("Informe a nota do aluno","Digite aqui");
        if(isNaN(nota)){
            alert("Voce nao digitou um numero valido");
        }else{
            media = parseFloat(media) + parseFloat(nota);
            contador++;
        }
    }
    media = media / 4;
    document.write("Aluno(a): " + nome );
    document.write("<br>");
    document.write("Media....: " + media );
}
```

```
↳ Missao24.html ✘
```

```
1  <script>
2
3  function Media(){
4      var media = 0, nota = 0, contador = 0;
5      var nome = prompt("Informe o nome do aluno","Digite aqui!!!");
6
7      while(contador < 4){
8          nota = prompt("Informe a nota do aluno","Digite aqui");
9          if(isNaN(nota)){
10              alert("Voce nao digitou um numero valido");
11          }else{
12              media = parseFloat(media) + parseFloat(nota);
13              contador++;
14          }
15      }
16      media = media / 4;
17      document.write("Aluno(a).. " + nome );
18      document.write("<br>");
19      document.write("Media....: " + media );
20  }
21
22  </script>
```

Salve, abra no Google Chrome, abra o console, digite o nome da função ( Media(); ) , pressione enter, preencha as informações e visualize o resultado.

Aluno(a):: Guto

Media....: 7.75

The screenshot shows a browser's developer tools interface with the 'Console' tab selected. At the top, there are icons for selection, copy, and paste. Below the tabs, there are icons for play, stop, and refresh, followed by the text 'top'. The main area displays the following command and its result:

```
> Media();  
< undefined  
>
```

# **Dia 25**

## **Missão: Abrindo site de notícias**

### **Objetivo:**

- Criando funções personalizadas
- Utilizar funções diversas do JavaScript

**Ao som de:** Metallica - Nothing Else Matters

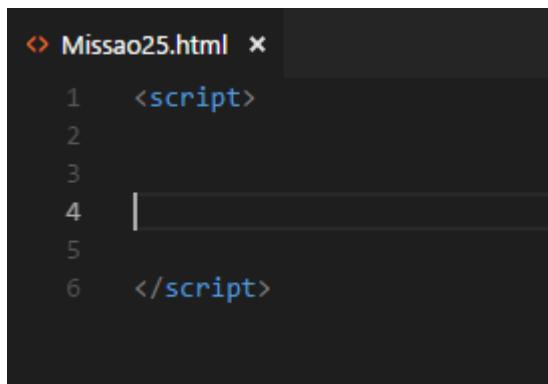
### **Descrição:**

**Caso ainda não tenha feito a missão do dia 24, volte e realize ela.**

Abra o VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao25.html e já digite:

```
<script>
```

```
</script>
```



```
1  <script>
2
3
4  |
5
6  </script>
```

### **Missão**

Vamos criar uma função que abra o site do UOL e uma que abra o site GLOBO. Digite:

```
<script>
function AbrirUOL(){
    alert("O site UOL será aberto");
    window.open("https://uol.com.br");
}

function AbrirGlobo(){
    alert("O site GLOBO será aberto");
    window.open("https://globo.com");
}
</script>
```

```
1  <script>
2  function AbrirUOL(){
3      alert("O site UOL será aberto");
4      window.open("https://uol.com.br");
5  }
6
7  function AbrirGlobo(){
8      alert("O site GLOBO será aberto");
9      window.open("https://globo.com");
10 }
11 </script>
12 |
```

Salve, abra no Google Chrome, abra o console, digite o nome da função ( AbrirUOL(); ou AbrirGlobo(); ), pressione enter, preencha as informações e visualize o resultado.

# Dia 27

## Missão: Abrindo qualquer site

### Objetivo:

- Criando funções personalizadas com parâmetros

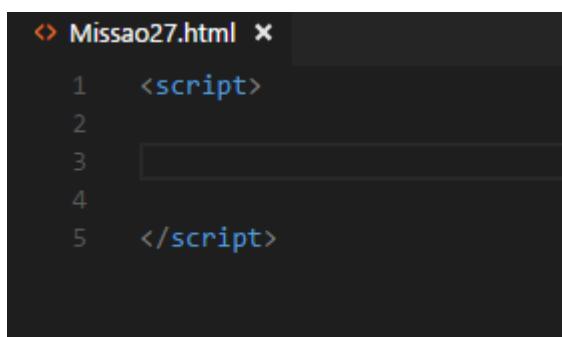
**Ao som de:** The Pigeon Detectives - I'm Not Gonna Take This

### Descrição:

Caso ainda não tenha feito a missão do dia 26, volte e realize ela.

Abra o VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao27.html e já digite:

```
<script>  
    </script>
```



```
1  <script>
```

### Missão

A resolução do nosso problema será muito parecido com o da missão 25, porém na missão 25 se quiséssemos abrir 10 sites diferentes teríamos que criar 10 funções diferentes, aqui vamos ter apenas uma função, porém essa função receberá um parâmetro. Digite:

```
function AbrirSite(nomeSite , enderecoSite){
```

```

        alert("O site " + nomeSite + " será aberto");
        window.open(enderecoSite);
    }

```



```

<script>
function AbrirSite(nomeSite , enderecoSite){
    alert("O site " + nomeSite + " será aberto");
    window.open(enderecoSite);
}
</script>

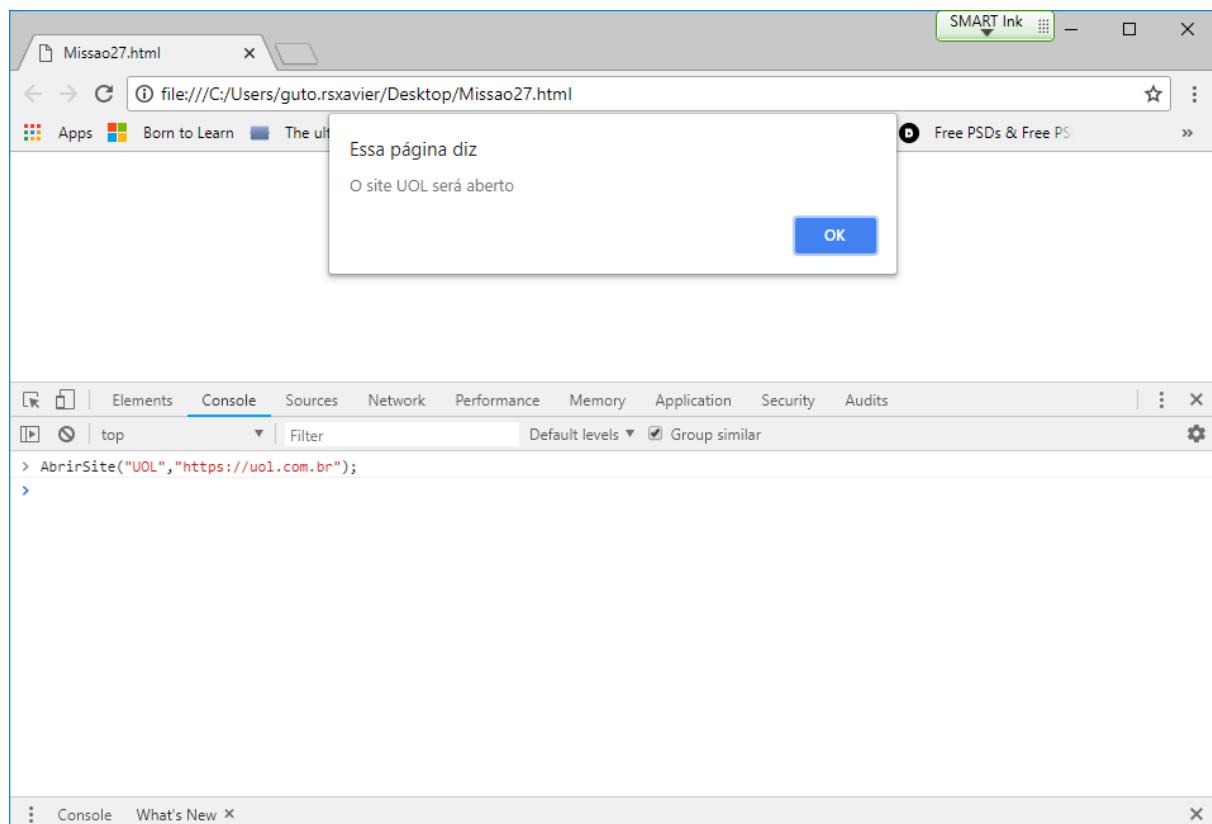
```

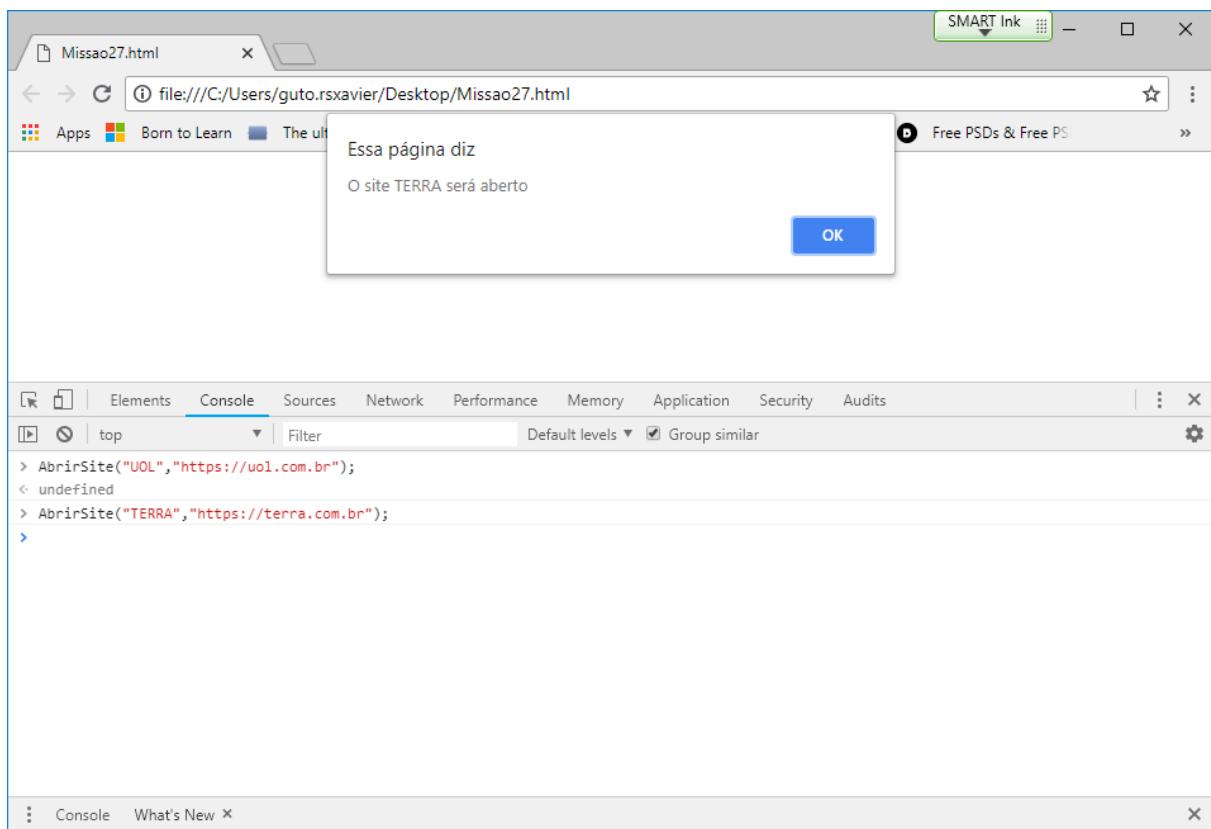
Salve, abra no Google Chrome, abra o console, digite o nome da função junto com os parâmetros:

```

AbrirSite("UOL","https://uol.com.br");
AbrirSite("TERRA","https://terra.com.br");

```





Quando você chama a função desse jeito

`AbrirSite("TERRA","https://terra.com.br");` o valor "TERRA" será passado para a variável nomeSite e o valor "`https://terra.com.br`" será passado para a variável enderecoSite. Utilizando parâmetros nossas funções ficam muito mais dinâmicas e úteis.

# **Dia 28**

## **Missão: Acréscimo e desconto**

### **Objetivo:**

- Criando funções personalizadas com parâmetros

**Ao som de:** Dimmi, Zeeba - Found U

### **Descrição:**

**Caso ainda não tenha feito a missão do dia 27, volte e realize ela.**

Abra o VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao28.html e já digite:

```
<script>  
</script>
```

## **Missão**

Vamos criar uma função que receba 3 parâmetros: o valor total, uma porcentagem e a operação. Se a operação for acréscimo a porcentagem deve ser acrescida ao valor total, se a operação for desconto a porcentagem deve ser descontada do valor total. Digite:

```
<script>  
function operacao(total, porcentagem, operacao){  
    var valor_porcentagem = porcentagem/100 * total;  
    var total_pagar = 0;  
    if(operacao == "acrescimo"){  
        //acréscimo
```

```

        total_pagar = total + valor_porcentagem;
    }else{
        //desconto
        total_pagar = total - valor_porcentagem;
    }

    console.log("Total a pagar: R$" + total_pagar.toFixed(2));
}
</script>

```

```

1 ▼ <script>
2 ▼ function operacao(total, porcentagem, operacao){
3     var valor_porcentagem = porcentagem/100 * total;
4     var total_pagar = 0;
5     if(operacao == "acrescimo"){
6         //acréscimo
7         total_pagar = total + valor_porcentagem;
8     }else{
9         //desconto
10        total_pagar = total - valor_porcentagem;
11    }
12
13    console.log("Total a pagar: R$" + total_pagar.toFixed(2));
14 }
15 </script>

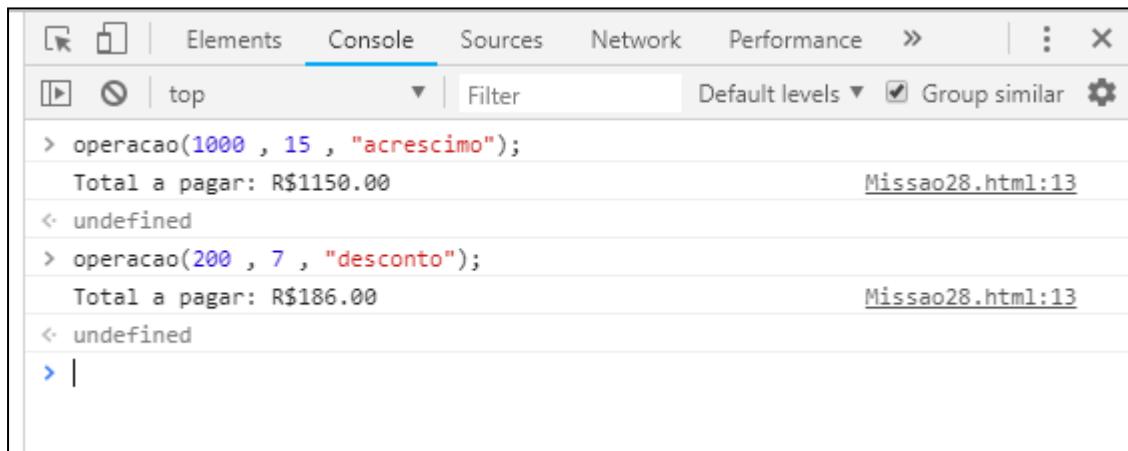
```

Salve, abra no Google Chrome, abra o console, digite o nome da função junto com os parâmetros:

```

operacao(1000 , 15 , "acrescimo");
operacao(200 , 7 , "desconto");

```



# Dia 29

## Missão: Classificação de filmes

### Objetivo:

- Criando funções personalizadas com parâmetros

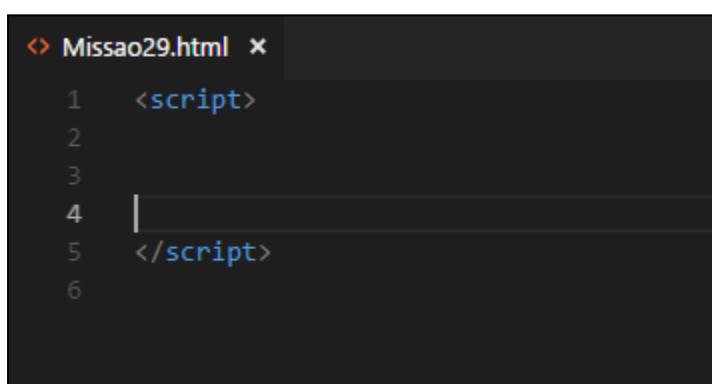
**Ao som de:** Cat Dealers & Evokings feat Magga - Gravity

### Descrição:

Caso ainda não tenha feito a missão do dia 28, volte e realize ela.

Abra o VS Code, aperte CTRL+S e salve o arquivo com o nome de Missao29.html e já digite:

```
<script>  
    </script>
```



```
↳ Missao29.html ✘  
1  <script>  
2  
3  
4  |  
5  </script>  
6
```

## Missão

Vamos criar um sistema de classificação de filmes onde mostramos o filme e o usuário da uma nota de 1 à 5. Depois exibimos o filme e sua classificação, mostrando a classificação com estrelas(asteriscos). Vamos precisar de dois arrays, um para armazenar os filmes e o outro para armazenar as classificações. Digite:

```
var classificacao = [];
var filmes = ["Conan" , "Harry Potter" , "Matrix" , "Crepusculo"];
```

Agora vamos criar a função Classificacao, essa função será composta por um laço que mostra filme à filme e pede a nota e armazena essa nota no array classificacao. Digite:

```
function Classificacao(){
    alert("Informe uma classificacao de 1 a 5 estrelas pra cada filme");
    for(contador = 0 ; contador < filmes.length ; contador++){
        classificacao[contador] = prompt("Informe a quantidade de estrelas
pro filmes " + filmes[contador]);
    }
}
```

Para finalizar vamos criar a função ExibeFilmes. Nessa função temos dois laços de repetição, um para exibir os filmes, e dentro do laço dos filmes temos um para exibir os asteriscos, ou seja, para cada filme será montada uma string com as estrelas(asteriscos) equivalentes à nota informada pelo usuário. Digite:

```
function ExibirClassificacoes(){
    var numEstrelas = "";
    for(contador = 0 ; contador < filmes.length ; contador++){

        numEstrelas = "";

        for(contadorEstrelas = 1; contadorEstrelas <= classificacao[contador]
; contadorEstrelas++){
            numEstrelas += "*";
        }

        console.log(filmes[contador] + " " + numEstrelas + " estrelas " );
    }
}
```

Veja o código inteiro:

```
<script>

var classificacao = [];
var filmes = ["Conan" , "Harry Potter" , "Matrix" , "Crepusculo"];

function Classificacao(){
    alert("Informe uma classificacao de 1 a 5 estrelas pra cada filme");
    for(contador = 0 ; contador < filmes.length ; contador++){
        classificacao[contador] = prompt("Informe a quantidade de estrelas
```

```

    pro filmes " + filmes[contador]);
}
}

function ExibirClassificacoes(){
    var numEstrelas = "";
    for(contador = 0 ; contador < filmes.length ; contador++){
        numEstrelas = "";

        for(contadorEstrelas = 1; contadorEstrelas <= classificacao[contador] ;
        contadorEstrelas++){
            numEstrelas += "*";
        }

        console.log(filmes[contador] + " " + numEstrelas + " estrelas " );
    }
}

</script>

```

```

1  <script>
2
3  var classificacao = [];
4  var filmes = ["Conan" , "Harry Potter" , "Matrix" , "Crepusculo"];
5
6  function Classificacao(){
7      alert("Informe uma classificacao de 1 a 5 estrelas pra cada filme");
8      for(contador = 0 ; contador < filmes.length ; contador++){
9          classificacao[contador] = prompt("Informe a quantidade de estrelas pro filmes " + filmes[contador]);
10     }
11 }
12
13 function ExibirClassificacoes(){
14     var numEstrelas = "";
15     for(contador = 0 ; contador < filmes.length ; contador++){
16
17         numEstrelas = "";

18         for(contadorEstrelas = 1; contadorEstrelas <= classificacao[contador] ; contadorEstrelas++){
19             numEstrelas += "*";
20         }

21         console.log(filmes[contador] + " " + numEstrelas + " estrelas " );
22     }
23 }
24
25 </script>
26
27
28 |

```

Salve, abra no Google Chrome, abra o console, digite o nome da função, preencha os dados, e execute a segunda função:

```

Classificacao();
ExibirClassificacoes();

```

The screenshot shows the 'Console' tab of a browser's developer tools. The tab bar includes 'Elements', 'Console' (which is active), and 'Sources'. Below the tabs, there are icons for back, forward, and search, followed by the text 'top' and a dropdown arrow. A 'Filter' input field is also present. The main area displays a stack trace:

```
> Classificacao();
< undefined
> ExibirClassificacoes();
Conan ***** estrelas
Harry Potter **** estrelas
Matrix *** estrelas
Crepusculo ** estrelas
< undefined
>
```

# **Dia 30**

## **Missão: CONCLUÍDA COM SUCESSO**

**Ao som de:** Swedish House Mafia - Don't You Worry Child ft. John Martin

### **Parabéns!!!**

Se você chegou até aqui foi por sua dedicação. Acredito que hoje saiba mais de JavaScript do que à 30 dias atrás.

A ideia dessa atividade nunca foi só JavaScript, por trás disso teve resiliência, disposição, perseverança, dedicação, entre outras coisas.

Abra os links abaixo e compreenda o significado desses 30 dias de código (que na verdade são 100):

- [100 dias de código](#)
- [Aprenda a programar com desafio 100 dias de Código](#)
- [O que eu aprendi escrevendo código por 100 dias](#)

Depois de 30 dias é hora de descansar um pouco, tire alguns dias de folga, se recompense (compre um chocolate pra você) agradeça as pessoas que te ajudaram e avise que elas te ajudaram a ser um profissional melhor.

Depois dessa pausa inicie um novo ciclo, pegue um outro assunto e se proponha a estudar 30 dias, criando exemplos e atividades referentes à esse assunto. Vá documentando sua evolução, de preferência de forma visível e física, como com uma agenda, adesivos ou post-it espalhados pelo seu ambiente de estudo.

Caso queira se aprofundar mais em JavaScript segue uma lista de sites com conteúdo gratuito e de qualidade:

- <https://www.freecodecamp.org>
- <https://www.codecademy.com>
- <https://rocketseat.com.br/curso-javascript-basico-do-zero>
- [Desvendando a linguagem JavaScript](#)
- [Curso de JavaScript \(Universidade XTI\)](#)