

Resolução dos Exercícios

- 1) Primeiro, defini um vetor de 5 números inteiros, chamado “lista”. Depois, apenas crie um loop *for* para dar um *alert* de cada um dos itens separadamente.

```
<script>
  let lista = [1,2,3,4,5]

  for (item of lista) {
    alert(item)
  }
</script>
```

- 2) Crie um vetor com 10 números inteiros. Após isso, utilizei o comando *.reverse()*. Segundo o esse site:

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/reverse,

o método “*.reverse()*” serve para inverter a ordem dos elementos da lista, de modo que o último se torne o primeiro, o penúltimo o segundo e assim por diante. Ao fim, utilizei o comando *alert* para imprimir a lista invertida.

```
<script>
  let lista = [1,2,3,4,5,6,7,8,9,10]

  lista.reverse()

  alert(lista)
</script>
```

- 3) Inicialmente, criei um array com 4 números inteiros, que representam as notas. Juntamente a isso, criei uma variável para somar os valores através do método “*.reduce()*”. Através do site:

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Array/reduce,

pude entender que o método “`.reduce()`” executa uma função, de forma a reduzir todos os elementos de uma lista a um único valor segundo o algoritmo fornecido pelo desenvolvedor. No caso dessa atividade, informei os valores (a,b) para o programa, onde “a” representa o valor acumulado e “b” o valor atual da lista que está sendo processado. Assim, indiquei que eles deveriam ser somados ($\Rightarrow a + b$) a partir do valor 0, indicado após a vírgula.

```
let notas = [7,8,9,10];  
let soma = notas.reduce((a,b)=>a+b,0);
```

Assim, calculei a média dos valores dividindo a variável “soma” pelo comprimento da lista “notas”.

```
let media = soma / 4;
```

Ao final, apenas imprimi os valores por meio do “`alert`”. Perguntando para o ChatGPT, ele me ensinou como eu poderia inserir variáveis dentro de um alert, substituindo as aspas (") pelo crase (`) e utilizando `${ }` para inserir variáveis.

como posso inserir variáveis dentro de um alert através da interpolação de strings

< 2/2 >



Para inserir variáveis dentro de um `alert` utilizando a interpolação de strings em JavaScript, você pode usar template literals, que são strings delimitadas por crases (``) em vez de aspas. Com template literals, você pode inserir variáveis diretamente dentro da string usando a sintaxe `${variável}`.

Aqui está um exemplo:

javascript

Copiar código

```
let nome = "Felipe";  
let idade = 17;  
  
alert(`Olá, meu nome é ${nome} e eu tenho ${idade} anos.`);
```

Meu código:

```
alert(`Nota 1: ${notas[0]} | Nota 2: ${notas[1]} | Nota 3: ${notas[2]} | Nota 4: ${notas[3]} | Média: ${media.toFixed(2)}`)
```

- 4) De início, crie 4 variáveis. A primeira delas chamada “*lista*”, que seria responsável por armazenar os 10 caracteres a serem identificados pelo programa. Também criei o vetor “*vogais*” para informar as vogais do alfabeto. Além disso, criei a variável “*contagem*”, que seria responsável por contar a quantidade de consoantes identificadas pelo programa. Por último, criei o array “*consoantes*”, para guardar os valores diferentes de vogais.

```
let lista = ['a','b','c','d','e','f','g','h','i','j'];
let vogais = ['a','e','i','o','u'];
let contagem = 0;
let consoantes = [];
```

Assim, através do comando “*for...of*”, que segundo o site:

[https://developer.mozilla.org/pt-](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/for...of)

[BR/docs/Web/JavaScript/Reference/Statements/for...of](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/for...of),

percorre separadamente cada um dos objetos dentro de um array, utilizei a condicional “*if*” para verificar se o objeto do vetor “*lista*” estava incluso nas vogais por meio do “*.includes()*”. No site:

[https://developer.mozilla.org/pt-](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global%20Objects/Array/includes)

[BR/docs/Web/JavaScript/Reference/Global Objects/Array/includes](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global%20Objects/Array/includes),

é explicado que o método “*.includes()*” serve para verificar se um determinado array possui entre seus itens um determinado elemento, retornando *true* ou *false*. Dessa forma, voltando para o meu código, caso o item não esteja dentro da lista vogais, significa que é uma consoante. Assim, através do operador de incremento “*++*”, que adiciona em 1 o valor de uma variável, é feita a contagem de consoantes, enquanto o *consoantes.push(item)* adiciona a consoante ao array.



Em JavaScript, o operador `++` é um **incrementador** que aumenta o valor de uma variável numérica em 1.

```
for (item of lista) {  
    if (vogais.includes(item) == false) {  
        contagem++;  
        consoantes.push(item)  
    }  
}
```

Ao final, apenas utilizei o `alert` para imprimir os resultados por meio da interpolação de strings.

```
alert(`Quantidade de consoantes: ${contagem} | Consoantes: ${consoantes}`)
```

- 5) Primeiro, crie 3 listas: A primeira para armazenar 20 números inteiros, a segunda para armazenar os números pares e a terceira para números ímpares.

```
let lista = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20];  
let listaPar = [];  
let listaImpar = [];
```

Com isso, através do `for...of`, que irá percorrer cada um dos itens dentro do vetor `lista`, utilizei a condicional `if` para verificar se o número é par, através do operador lógico `%`, que representa módulo. Caso a divisão de um número por 2 tenha um resto zero, o número será par. Dessa forma, o número será adicionado a `listaPar` por meio do comando `listaPar.push(num)`. Caso a operação interior retorne um resultado igual a `false`, será ativado a condicional `else`, que irá adicionar o número a `listaImpar` por meio do código `listaImpar.push(num)`.

```
for (num of lista) {
  if (num % 2 == 0) {
    listaPar.push(num)
  } else {
    listaImpar.push(num)
  }
}
```

Por último, utilizei a interpolação de strings e o comando alert para imprimir os textos juntamente as variáveis utilizando “\${}”.

```
alert(`Lista: ${lista}`)
alert(`Números pares: ${listaPar}`);
alert(`Números ímpares: ${listaImpar}`);
```

- 6) Inicialmente, criei duas variáveis, sendo elas uma lista chamada “medias” para armazenar as médias e outra nomeada “maior7” para realizar a contagem de médias maiores ou equivalentes a 7.


```
let maior7 = 0;
let medias = [];
```

Assim, utilizei o loop “for”. Segundo o ChatGPT, ao invés de percorrer os valores ou índices de uma lista, ela irá percorrer uma quantidade determinada de valores. O primeiro elemento (i = 0) indica o valor inicial desse loop, que será 0. A segunda parte (i < 10) diz que o loop deve se repetir até que essa condição se torne falsa. Por último, o i++ terá a função de somar valor da variável “i”. Assim, a cada nova repetição do loop, o valor de “i” será acrescido em 1. Cada uma dessas especificações é separada por meio de “;”.



Essa linha de código é um loop `for` em JavaScript. O loop `for` executa um bloco de código repetidamente enquanto uma condição é verdadeira. Aqui está uma explicação detalhada:

javascript

 Copiar código

```
for (i = 0; i < 10; i++)
```

- `i = 0`: Inicializa a variável `i` com o valor 0. Isso é feito apenas uma vez, antes do loop começar.
- `i < 10`: Condição de continuação do loop. Enquanto `i` for menor que 10, o código dentro do loop será executado.
- `i++`: Incrementa o valor de `i` em 1 após cada iteração do loop.

Assim, utilizei dois loops `for`. O primeiro será repetido 10 vezes para coletar as notas dos 10 alunos. Após criar a variável `soma`, será iniciado outro loop `for`, que se repetirá 4 vezes, para assim poder coletar as 4 notas dos alunos por meio da variável `num` do tipo `float` e do comando `prompt`. A cada nova execução, os valores serão somados a variável `soma`, por meio da linha `"soma += num"`, que irá somar o valor atual de `soma` a `num`.

Após coletar as 4 notas, será calculada a média dos valores, dividindo a variável `soma` por 4. Segundo o site:

https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Number/toFixed,

o método `toFixed(2)` serve para informar ao programa que o número terá 2 casas decimais.

Após fazer esse cálculo, a media será adicionada a lista de médias por meio do código `"medias.push(media)"`.

```
for (i = 0; i < 10; i++) {  
  let soma = 0;  
  for (i2 = 0; i2 < 4; i2++) {  
    let num = parseFloat(prompt(`Informe a nota ${i2+1} do estudante ${i+1}: `));  
    soma += num;  
  }  
  media = (soma / 4).toFixed(2);  
  medias.push(media);  
}
```

Com isso, através do loop `"for...of"`, que irá percorrer cada elemento da lista de médias, o condicional `if` irá verificar se o número é maior ou igual a 7. Caso o programa retorne `true`, a variável `maior7` terá seu valor acrescentado em 1 por meio do código `"maior7++"` para servir como contador.

```
for (item of medias) {
    if (item >= 7) {
        maior7++;
    }
}
```

Por último, utilizei a interpolação de strings e o comando alert para imprimir os textos juntamente as variáveis utilizando “\${ }”.

```
alert(`Médias: ${medias}\nAlunos com nota igual ou maior que 7: ${maior7} estudantes`)
```

- 7) Primeiro, defini 3 variáveis. A primeira delas é uma lista nomeada “*numeros*” contendo 5 números inteiros. A segunda é a variável “*soma*” para armazenar a soma dos valores, e a terceira chamada “*multiplicacao*”, que será responsável por armazenar a multiplicação entre os elementos do array.

```
let numeros = [1,2,3,4,5];
let soma = 0;
let multiplicacao = 1;
```

Através do loop “*for...of*”, cada um dos números do vetor “*numeros*” será percorrido. Assim, ele será somado a variável soma, por meio do comando “*soma += numero*”, que irá acrescentar o valor de número ao valor atual da variável “*soma*”, enquanto a variável “*multiplicacao*” irá multiplicar o valor adjacente dentro dela pelo valor de “*numero*”. Assim, ao final do loop, a variável soma terá somado todos os valores do array “*numeros*”, enquanto “*multiplicacao*” terá multiplicado os valores de “*numeros*” da esquerda para a direita.

```
for (numero of numeros) {
    soma += numero;
    multiplicacao *= numero;
}
```

Por fim, utilizei o alert e a interpolação de strings para exibir os resultados do programa, inserindo variáveis junto a textos por meio dos caracteres “\${ }”.

```
alert(`Números: ${numeros} | Soma dos números: ${soma} | Multiplicação dos números: ${multiplicacao}`)
```

- 8) Primeiro, crie dois arrays. O primeiro chamado “idades”, que irá armazenar as idades fornecidas pelo usuário, e o segundo, nomeado “alturas”, que terá a função de guardar as alturas.

```
let idades = [];  
let alturas = [];
```

Assim, criei um loop for, passando o seguinte algoritmo para ele: $i = 0$: A variável i deve começar por 0; $i < 5$: Define que, enquanto essa condição for verdadeira, o programa deve ser executado, mas caso se torne falsa, deve encerrar o loop; $i++$: A cada execução do loop, o valor de i será acrescentado em 1.

Dessa forma, a variável “idade”, por meio de um *prompt*, irá perguntar ao usuário a idade. Após coletar esse dado, será perguntado a altura armazenado na variável “altura”. Por fim, o valor de “idade” será adicionado à lista “idades”, enquanto “altura” será adicionado a “alturas”. Esse processo se repetirá por 5 vezes.

```
for (i = 0; i < 5; i++) {  
  let idade = parseInt(prompt(`Informe a idade da pessoa ${i+1}: `))  
  let altura = parseFloat(prompt(`Informe a altura da pessoa ${i+1}: `))  
  
  idades.push(idade)  
  alturas.push(altura)  
}
```

Ao final, os *arrays* serão impressos por meio do alert. Contudo, conforme pede a atividade, os vetores devem estar na ordem reversa a que foram lidos pelo programa. Dessa maneira, utilizei o método “.reverse()” para inverter a ordem dos itens dentro das duas listas.

```
alert(`Idades: ${idades.reverse()}`)  
alert(`Alturas: ${alturas.reverse()}`)
```


- 9) Primeiro, criei o array “*numeros*” para armazenar o valor de 10 números inteiros, juntamente a variável soma, que irá armazenar a soma do quadrado dos números.

```
let numeros = [1,2,3,4,5,6,7,8,9,10];  
let soma = 0;
```

Utilizando o “for...of”, que irá percorrer separadamente cada um dos valores do array “*numeros*”, fiz as operações: Primeiro, o valor da variável “*numero*” será substituído pelo seu quadrado. Assim, ele será acrescentado ao valor da variável soma por meio do comando “*soma += numero*”, que ao final do loop terá somado o valor do quadrado de cada um dos elementos do array “*numeros*”.

```
for (let numero of numeros) {  
    numero = numero**2;  
    soma += numero;  
}
```

Por fim, utilizei o *alert* e a *interpolação de strings* por meio de crase (‘) para exibir a variável “*soma*”.

```
alert(`Soma do quadrado dos números: ${soma}`)
```

- 10) Primeiro, crie 3 arrays: O primeiro, chamado “*lista1*”, armazenando 10 valores, assim como a “*lista2*”. Por último, criei o vetor “*lista3*”, que será a concatenação das duas listas anteriores intercaladamente.

```
let lista1 = [1,3,5,7,9,11,13,15,17,19];  
let lista2 = [2,4,6,8,10,12,14,16,18,20];  
let lista3 = [];
```

Após isso, que um loop for para ler a posição dos termos da *lista1*, que possui a mesma quantidade da *lista2*. Assim, utilizei o “*lista3.push(lista1[i])*” para adicionar um elemento da “*lista1*” a “*lista3*”. Logo abaixo, “*lista3.push(lista2[i])*” acrescentava a “*lista3*” um item da “*lista2*”.

```
for (i in lista1) {
    lista3.push(lista1[i]);
    lista3.push(lista2[i]);
}
```

Ao final do loop, a “lista3” terá 20 itens, 10 de cada uma das outras listas, mas de forma intercalada. Assim, utilizei o “alert” e a *interpolação de strings* para imprimir cada uma das 3 listas separadamente.

```
alert(`Lista 1: ${lista1} | Lista 2: ${lista2} | Lista 3: ${lista3}`)
```

- 11) Para iniciar, criei 3 listas, assim como na atividade anterior, “lista1” e “lista2”, cada um com 10 elementos cada, juntamente a “lista3”, que será a concatenação de ambas os vetores. Além disso, também criei as variáveis “valor1” e “valor2”, que servirão como uma forma de contagem.

```
let lista1 = [1,3,5,7,9,11,13,15,17,19];
let lista2 = [2,4,6,8,10,12,14,16,18,20];
let lista3 = [];
let valor1 = 0;
let valor2 = 0;
```

Assim, utilizei o loop while para dar sequência. Segundo o site:

[https://developer.mozilla.org/pt-](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/while)

[BR/docs/Web/JavaScript/Reference/Statements/while](https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Statements/while),

o while é um laço que executa determinado código enquanto uma certa opção for satisfeita. No caso dessa atividade, enquanto “valor1” for menor que o comprimento da “lista1” ou enquanto “valor2” for menor que o comprimento da “lista2”, o loop será mantido. Caso alguma dessas condições não se satisfaça, o loop será encerrado.

Dessa forma, dentro do laço while, crie primeiramente um laço for, que se repetirá 3 vezes ou até que o “valor1” tenha um comprimento menor que “lista1”. O mesmo acontece para o segundo for, que trará as mesmas condições, porém para a “lista2”. Essas condições garantem que sejam adicionados apenas

os 10 valores presentes em cada lista, já que, sem essa funcionalidade de parar quando o valor de uma das variáveis atingisse 10, o programa iria adicionar 2 valores vazios a cada lista, já que os índices 10 e 11 não teriam nenhum conteúdo. Essa estrutura também garante que 3 valores de cada lista sejam adicionados intercaladamente a “lista3”.

```
while (valor1 < lista1.length || valor2 < lista2.length) {  
  for (i2 = 0; i2 < 3 || valor1 < lista1.length; i2++) {  
    lista3.push(lista1[valor1]);  
    valor1++;  
  }  
  for (i3 = 0; i3 < 3 || valor2 < lista2.length; i3++) {  
    lista3.push(lista2[valor2]);  
    valor2++;  
  }  
}
```

Por fim, utilizei o alert e a interpolação de strings para exibir as três listas.

```
alert(`Lista 1: ${lista1} | Lista 2: ${lista2} | Lista 3: ${lista3}`)
```

- 12) Para dar início a atividade, através do ChatGPT, gerei 30 valores de idades e alturas aleatórios apenas para fins de poupar tempo.

```
let idades = [14, 12, 13, 15, 14, 16, 12, 13, 14, 15,  
             16, 13, 12, 14, 13, 15, 14, 12, 13, 16,  
             15, 14, 13, 12, 14, 16, 15, 13, 12, 14];  
  
let alturas = [1.60, 1.55, 1.62, 1.70, 1.68, 1.65, 1.58, 1.72, 1.60, 1.66,  
              1.75, 1.63, 1.57, 1.69, 1.61, 1.64, 1.67, 1.59, 1.65, 1.70,  
              1.66, 1.68, 1.62, 1.54, 1.63, 1.71, 1.69, 1.64, 1.56, 1.67];
```

Assim, criei uma variável chamada “media”. Para calcular a média das alturas, utilizei o método .reduce() para realizar a soma de todos os valores da lista “alturas”. Como já explicado anteriormente, a função do método .reduce() é diminuir a quantidade de termos de uma lista para apenas um, seguindo uma função determinada pelo desenvolvedor. No meu caso, defini os valores “a” e “b” ((a,b)), sendo “a” o valor acumulado e “b” o valor atual a ser lido pelo programa. Desse modo, defini que o programa deveria retornar a + b (=> a+b), a partir do número 0 (,0).

Após realizar a soma de todos os valores, realizei a divisão pelo comprimento da lista de alturas (lista.length). Por fim, adicionei o “toFixed(2)” para arredondar a média para 2 casas decimais.

```
let media = (alturas.reduce((a,b) => a+b,0) / alturas.length).toFixed(2);
```

Criei a variável contagem, dando a ela um valor de zero.

```
let contagem = 0;
```

Também criei um loop “for..in” para ler os índices da lista idades. Dentro do for, a primeira condicional “if” tem a função de verificar se a idade do estudante no índice determinado pelo laço “for” é maior ou igual a 13, ele dará sequência ao programa. Assim, ao iniciar o próximo condicional if, o programa irá verificar se a altura desse estudante, determinado pelo índice do laço for, é inferior a média. Caso o programa retorne true, o valor de contagem será acrescido em 1, por meio do código “contagem += 1”, que soma o valor atual da variável (contagem) ao valor fornecido (1).

```
for (idade in idades) {  
  if (idades[idade] >= 13) {  
    if (alturas[idade] < media) {  
      contagem += 1;  
    }  
  }  
}
```

Para finalizar o programa, utilizei o alert e a interpolação de strings para imprimir o resultado da contagem de alunos com mais de 13 anos que possuem altura inferior e média.

```
alert(`Alunos com 13 anos ou menos com altura menor que a média: ${contagem}`);
```

- 13) Inicialmente, defini 3 variáveis. A primeira delas é um array chamado “temperaturas”. O segundo um vetor nomeado “temperaturasAcimaMedia”. Por fim, uma lista com o nome “meses”, que contém os nomes de todos os meses do ano.

```
let temperaturas = [];
let temperaturasAcimaMedia = [];
let meses = ['janeiro', 'fevereiro', 'março',
'abril', 'maio', 'junho', 'julho', 'agosto',
'setembro', 'outubro', 'novembro', 'dezembro']
```

Prosseguindo, criei um loop “for...of” para percorrer todos os valores “mes” dentro da lista “meses”. Assim, o programa perguntará ao usuário qual a temperatura média de cada mês do ano, adicionando a resposta ao vetor “temperaturas” por meio do “temperaturas.push(temperatura)”.

```
for (mes of meses) {
  let temperatura = parseFloat(prompt(`Informe a temperatura de ${mes}: `));
  temperaturas.push(temperatura)
}
```

Após o final do loop, o programa fará o cálculo da média anual de temperatura dentro da variável “media”. Primeiro, utilizei o método .reduce() para diminuir os valores dentro da lista “temperaturas” a um único valor, que segundo as regras fornecidas ((a,b)=>a+b,0), será a soma de todos os valores. Depois desse cálculo, o total será dividido pelo comprimento dessa mesma lista, obtendo assim a média. Para finalizar, utilizei o “toFixed(2)” para garantir que a média fosse convertida em um valor de apenas 2 casas decimais.

```
let media = (temperaturas.reduce((a,b) => a + b,0) / temperaturas.length).toFixed(2);
```

Também utilizei um loop “for...in” para ler os índices do array “temperaturas”. Com isso, a condicional if tem a função de verificar se a temperatura no índice fornecido pelo loop for é maior que a média. Caso o valor retornado seja verdadeiro, o programa irá adicionar ao vetor “temperaturasAcimaMedia” a temperatura média e o seu mês correspondendo.

```
for (i in temperaturas) {
  if (temperaturas[i] > media) {
    temperaturasAcimaMedia.push(`${meses[i]} - ${temperaturas[i]}°C`)
  }
}
```

Por fim, utilizei o alert e a interpolação de strings para imprimir a média anual de temperaturas, bem como os meses juntamente a sua temperatura onde a média foi maior que a média anual.

```
alert(`Média anual: ${media}°C, ${temperaturasAcimaMedia}`)
```

- 14) De início, criei duas variáveis. A primeira delas, chamada “contagemSim”, tem como função contar a quantidade de vezes em que foi dito “sim”. Além disso, o array “perguntas” armazena as 5 perguntas a serem feitas ao usuário.

```
let contagemSim = 0;
let perguntas = ["Telefonou para a vítima?", "Esteve no local do crime?",
                 "Mora perto da vítima?", "Devia para a vítima?",
                 "Já trabalhou com a vítima?" ];
```

Após isso, defini uma função para fazer as perguntas. Segundo o site:

<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Functions>,

funções são elementos fundamentais em JavaScript, sendo essenciais em tarefas que exigem a repetição de determinado procedimento, já que são formadas por um conjunto de instruções que serão repetidos toda vez que a função for chamada.

Voltando ao meu código, dentro da função “fazerPergunta”, informei a variável “texto”, que é um valor a ser substituído por outro, devendo ser inserido quando a função for chamada.

Dentro da função criada, primeiro criei a variável “resposta”, que irá coletar a resposta do usuário para uma pergunta a ser inserida posteriormente. O loop while, logo abaixo, tem exatamente essa função. O laço de repetição de repetirá infinitamente até que se satisfaça determinado condição. No caso do meu código, se a resposta, utilizando o método “.toLowerCase()", que transforma todos os caracteres de uma variável em minúsculo, for igual a “sim”, a variável “contagemSim” deve ter seu valor acrescido em 1 por meio do comando “contagemSim++”, bem como quebrar o loop infinito. Entretanto, caso a resposta seja “não” ou “nao”, o laço apenas será interrompido. Contudo, caso a resposta não seja igual a nenhum dos itens anteriores, a pergunta tornará a ser

feita e continuará sendo exibida infinitamente até que a resposta seja “sim” ou “não”/”não”, através dos atributos inseridos dentro da condicional “else”.

```
function fazerPergunta(texto) {  
  let resposta = prompt(texto)  
  
  while (true) {  
    if (resposta.toLowerCase() == 'sim') {  
      contagemSim++;  
      break  
    } else if (resposta.toLocaleLowerCase() == 'não' || resposta.toLocaleLowerCase() == 'nao') {  
      break  
    } else {  
      resposta = prompt(texto)  
    }  
  }  
}
```

Criada a função, utilizei um loop “for...of” para ler todos os valores dentro da lista “perguntas”. Assim, para cada item, a função “fazerPergunta()” será chamada, informando a variável “pergunta” para substituir o parâmetro “texto”. Assim, as 5 perguntas serão feitas conforme os comandos definidos na função.

```
for (pergunta of perguntas) {  
  fazerPergunta(pergunta)  
}
```

Ao final, criei uma condicional if, que irá fazer a verificação da “contagemSim”. Caso o valor seja igual a 2, o programa irá exibir “Suspeito”. Caso o valor seja maior ou igual a 3 ou menor ou igual a 4, será impresso “Cúmplice”. Entretanto, caso a variável seja igual a 5, o alert mostrará “Culpado”. Por fim, se nenhum desses requisitos se encaixar, o programa exibirá “Inocente”.

```
if (contagemSim == 2) {  
  alert('Suspeito')  
} else if (contagemSim >= 3 && contagemSim <= 4) {  
  alert('Cúmplice')  
} else if (contagemSim == 5) {  
  alert('Culpado')  
} else {  
  alert('Inocente')  
}
```

- 15) Inicialmente, criei 3 variáveis. A primeira delas, um array chamado “notas”, servirá para armazenar as notas informadas pelo usuário. A segunda,

“notasAcimaMedia”, irá armazenar a contagem de notas acima da média. Por fim, “notasAbaixoSete” contará a quantidade de notas abaixo da média 7.

```
let notas = [];  
let notasAcimaMedia = 0;  
let notasAbaixoSete = 0;
```

Após isso, defini um loop “while (true)”, que irá repetir os comandos dentro dele infinitamente até que seja dado um “break”. Assim, criei uma variável nomeada “nota”, que irá pedir para o usuário um dado float por meio de um prompt uma nota. Caso o valor informado seja “-1”, o loop será encerrado. Caso não, a nota será adicionada a lista notas por meio do comando “notas.push(nota)”.

```
while (true) {  
  let nota = parseFloat(prompt('Informe a nota: '))  
  if (nota == -1) {  
    break  
  } else {  
    notas.push(nota)  
  }  
}
```

Após o fim do loop, defini uma variável chamada “qntNotas”, que irá armazenar a quantidade de notas fornecidas pelo usuário, por meio do comando “notas.length”, que lê o comprimento da lista “notas”. Abaixo, criei a variável soma, que por meio do método “.reduce()”, realizei a soma de “notas”. Assim, informei os parâmetros “a” e “b” a função, estabelecendo que o programa deve retornar a + b (=> a + b), iniciando a soma pelo valor 0. No cálculo da média, estabeleci a variável “media”, que realiza a operação de dividir “soma” por “qntNotas”, juntamente ao método “.toFixed(2)” ao final que irá arredondar o valor para 2 casas decimais.

Finalizado esse processo, iniciei um loop “for...of”, que tem a função de percorrer cada uma das notas dentro do array “notas”. Através da condicional if, caso o valor seja maior que a média, a variável “notasAcimaMedia” terá seu valor acrescido em 1, através do comando “notasAcimaMedia++”. Também criei uma segunda condicional if, para que se a nota for menor que 7, a variável

“notasAbaixoSete” tenha seu valor acrescido em 1, por meio do comando “notasAbaixoSete++”.

```
let qntNotas = notas.length;
let soma = notas.reduce((a,b) => a + b,0);
let media = (soma / qntNotas).toFixed(2);
for (item of notas) {
  if (item > media) {
    notasAcimaMedia++
  }


  if (item < 7) {
    notasAbaixoSete++
  }
}
```

Depois disso, criei a variável notas invertidas, e dentro dela inseri um “\n” para que a linha fosse quebrada e o texto fosse inserido abaixo. Assim, criei um loop “for...of” para ler as notas de uma cópia da lista notas invertida. Segundo o ChatGPT, o operador spread (...), tem a função de criar a cópia de um outro array, sem afetar a lista original.

1. Criando uma Cópia de um Array:

Quando você usa o operador spread dentro de um array, como em [...lista], ele cria uma cópia superficial dos elementos do array lista.

javascript

 Copiar código

```
let lista = [1, 2, 3];
let copia = [...lista];

console.log(copia); // Saída: [1, 2, 3]
```

No exemplo acima, copia é uma nova array que contém os mesmos elementos de lista, mas são arrays diferentes na memória. Modificar copia não afeta lista.

Assim, o loop for tem a função de adicionar cada uma das notas da lista “notas” invertida na variável “notasInvertidas”, mas com o diferencial de que cada nota seria exibida uma abaixo da outra devido o “\n”, que executa a quebra de linha.

```
let notasInvertidas = "\n";
for (n of [...notas].reverse()) {
  notasInvertidas += n + '\n';
}
```

Por fim, utilizei uma sequência de alerts para exibir todos os requisitos requeridos pela atividade.

```
alert(`Quantidade de valores lidos: ${qntNotas}`)
alert(`Notas informadas: ${notas}`)
alert(`Notas informadas na ordem inversa em que foram informadas: ${notasInvertidas}`)
alert(`Quantidade de notas acima da média: ${notasAcimaMedia}`)
alert(`Quantidade de notas abaixo de 7: ${notasAbaixoSete}`)
alert(`Obrigado por utilizar meu programa! Tchau~`)
```

- 16) Primeiro, defini uma variável chamada “salários”, que é um array responsável por armazenar os salários calculados com base nas vendas dos vendedores.

```
let salarios = [];
```

Após isso, criei uma função contar(valora, valorb), que tem como objetivo contar a quantidade de salários que estão dentro de um determinado intervalo (valora a valorb). Dentro dessa função, iniciei uma lista vazia chamada “lista”, que será preenchida com os salários que atendem aos critérios do intervalo definido.

Depois, adicionei uma condicional if que verifica se o valor inicial do intervalo (valora) é menor que 1000. Se for, exibe um alerta informando a quantidade de funcionários que recebem um salário dentro do intervalo (valora a valorb). Caso contrário, o alerta informa a quantidade de funcionários que recebem mais do que valora.

```
function contar(valora, valorb) {
  let lista = [];
  for (let posicao in salarios) {
    if (salarios[posicao] >= valora && salarios[posicao] < valorb) {
      lista.push(salarios[posicao])
    }
  }
  if (valora < 1000) {
    alert(`Quantidade de funcionários que recebem de ${valora} a ${valorb}: ${lista.length}`)
  } else {
    alert(`Quantidade de funcionários que recebem mais de ${valora}: ${lista.length}`)
  }
}
```

Em seguida, criei um loop while(true) para coletar os valores de vendas dos vendedores. O loop continua solicitando entradas até que o usuário digite -1,

indicando o fim da entrada de dados. Cada valor de venda inserido é convertido para um salário utilizando a fórmula $(vendas * 0.09) + 200$, e o salário é então adicionado ao array `salarios`.

```
while (true) {  
  let salario = parseInt(prompt('Informa o valor de vendas do vendedor (Digite -1 para parar): '))  
  if (salario == -1) {  
    break  
  } else {  
    salario = (salario*0.09)+200;  
    salarios.push(salario)  
  }  
}
```

Após coletar os salários, utilizei um loop `for` que percorre os intervalos de 200 a 1000 com incrementos de 100. Para cada intervalo, a função `contar()` é chamada para contar os salários que caem dentro daquele intervalo. Quando o loop atinge o valor 1000, a função `contar()` é chamada novamente para contar todos os salários acima de 1000.

```
for (i = 200; i < 1001; i += 100) {  
  if (i < 1000) {  
    contar(i,i+99)  
  } else {  
    contar(i,Infinity)  
  }  
}
```

Por fim, criei uma funcionalidade para buscar um valor de vendas específico e verificar se o salário correspondente está na lista de salários. Solicitei ao usuário que inserisse a quantidade de vendas a ser buscada, converti esse valor para o salário correspondente e percorri o array `salarios` para verificar se o valor está presente. Se encontrado, exibe um alerta informando a posição na lista; caso contrário, exibe uma mensagem de valor não encontrado.

```
let buscar = (parseInt(prompt('Informe a quantidade de vendas a ser buscado na lista: '))*0.09)+200;  
let encontrado = false;  
  
for (let posicao in salarios) {  
  if (buscar == salarios[posicao]) {  
    alert(`O salário ${buscar.toFixed(2)} está na posição ${posicao} da lista`);  
    encontrado = true;  
    break;  
  }  
}  
  
if (!encontrado) {  
  alert('Valor não encontrado na lista');  
}
```

- 17) Para começar, criei uma variável chamada nome que armazena o nome do atleta fornecido pelo usuário por meio de um prompt. Em seguida, o código verifica se o nome não está vazio usando o método trim(), que remove espaços em branco antes e depois da string. Se o nome não for uma string vazia, o programa prossegue para coletar os dados dos saltos.

```
let nome = prompt('Informe o nome do atleta: ');
if (nome.trim() !== '') {
    let saltos = [];
```

Para coletar as distâncias dos saltos, criei um array chamado saltos. Utilize um loop for que itera 5 vezes, uma para cada salto. Dentro do loop, há um loop while (true) que continua pedindo ao usuário a distância de cada salto até que um valor numérico válido seja inserido. O valor obtido é transformado em número de ponto flutuante usando parseFloat e, após ser validado, é adicionado ao array saltos. Caso o valor não seja um número, o loop continua até que um valor válido seja fornecido.

```
    let saltos = [];
    for (i = 0; i < 5; i++) {
        while (true) {
            let salto = parseFloat(prompt(`Informe a distância do ${i+1}º salto: `)).toFixed(2);
            if (!isNaN(salto)) {
                saltos.push(parseFloat(salto));
                break;
            }
        }
    }
}
```

Após coletar todos os saltos, calculei a média das distâncias utilizando o método .reduce(), que soma todos os valores do array e divide pelo número total de saltos para obter a média. O resultado é arredondado para duas casas decimais com .toFixed(2).

```
let media = (saltos.reduce((a,b) => a + b, 0) / saltos.length).toFixed(2);
```

Por fim, o código exibe um alert que mostra o nome do atleta, as distâncias dos cinco saltos e a média desses saltos. Se o nome do atleta for uma string vazia, o programa exibe uma mensagem de erro, solicitando que o nome do atleta seja

informado para continuar. Isso garante que o usuário forneça todos os dados necessários antes de calcular e exibir as informações finais.

```
    alert('Atleta: ${nome}\nSaltos: ${saltos[0]} - ${saltos[1]} - ${saltos[2]} - ${saltos[3]} - ${saltos[4]}\nMédia dos saltos: ${media}')
  } else {
    alert('É necessário informar o nome do atleta para continuar');
  }
}
```

- 18) Para iniciar, o código cria duas variáveis, jogadores e qntVotos, que são arrays vazios. A variável jogadores armazenará os números dos jogadores que receberem votos, e qntVotos armazenará a quantidade de votos correspondentes a cada jogador. Em seguida, o código entra em um loop while (true) que continua a execução até que o usuário decida encerrar, indicando isso ao inserir o número 0.

```
let jogadores = [];
let qntVotos = [];

while (true) {
  let resposta = parseInt(prompt('Número do jogador (0 para sair): '));
  if (resposta == 0) {
    break
  }
}
```

Dentro do loop, o código solicita ao usuário que insira o número de um jogador por meio de um prompt. Esse número é convertido para um inteiro usando parseInt. Se o usuário inserir 0, o loop é interrompido com a instrução break. Se o número inserido estiver fora do intervalo de 1 a 23, o código exibe um alerta informando que o número deve estar dentro desse intervalo ou que 0 pode ser usado para sair. Se o número estiver dentro do intervalo válido, o código verifica se esse número já foi registrado na lista jogadores usando o método indexOf. Se o número não estiver na lista (ou seja, indexOf retorna -1), o número do jogador é adicionado ao array jogadores e o contador correspondente em qntVotos é iniciado com 1. Se o número já estiver na lista, o contador de votos para esse jogador é incrementado em 1.

```

    } else if (resposta < 1 || resposta > 23) {
        alert('Digite um número entre 1 e 23 ou 0 para sair!');
    } else {
        let indice = jogadores.indexOf(resposta);
        if (indice === -1) {
            jogadores.push(resposta);
            qntVotos.push(1);
        } else {
            qntVotos[indice]++;
        }
    }
}

```

Depois que o loop é encerrado, o código calcula o total de votos utilizando o método `.reduce()` no array `qntVotos`, que soma todos os valores e armazena o total na variável `somaVotos`. Em seguida, um alerta é exibido mostrando o total de votos computados.

```

let somaVotos = qntVotos.reduce((a,b) => a+b,0)

alert(`Votos computados: ${somaVotos}`)

```

O código então constrói uma string texto para exibir os dados dos jogadores e seus votos. Para encontrar o melhor jogador, o código inicializa variáveis para armazenar o maior número de votos, o número do melhor jogador e a porcentagem de votos que ele obteve. Um loop `for` é usado para iterar sobre cada jogador em `jogadores`. Para cada jogador, o número de votos é obtido a partir do índice correspondente em `qntVotos`. A string texto é atualizada com o número do jogador, o número de votos e a porcentagem de votos calculada. Se o número de votos do jogador atual for maior do que o maior número registrado até o momento, o código atualiza as variáveis relacionadas ao melhor jogador.

```

let texto = 'Jogador      Votos      Porcentagem';
let votosMelhorJogador = -Infinity;
let melhorJogador = '';
let porcentagemMelhorJogador = 0;

for (jogador of jogadores) {
    let votosJogador = qntVotos[jogadores.indexOf(jogador)];
    texto += '\n${jogador}      ${votosJogador}      ${((votosJogador/somaVotos)*100).toFixed(2)}%';
    if (votosJogador > votosMelhorJogador) {
        melhorJogador = jogador;
        votosMelhorJogador = votosJogador;
        porcentagemMelhorJogador = ((votosJogador/somaVotos)*100).toFixed(2);
    }
}

```

Finalmente, o código exibe dois alertas. O primeiro alerta mostra a lista completa de jogadores, seus votos e as porcentagens de votos. O segundo alerta informa qual jogador recebeu o maior número de votos, quantos votos foram recebidos e qual foi a porcentagem do total. Isso fornece um resumo claro e detalhado dos resultados da votação.

```
alert(texto)
alert(`O melhor jogador segundo voto popular foi o número ${melhorJogador},
com ${votosMelhorJogador} votos, correspondendo a ${porcentagemMelhorJogador}% do total!`)
```

- 19) No código fornecido, iniciei definindo duas variáveis: sistemas e votos. A variável sistemas é um array que armazena os nomes dos sistemas operacionais possíveis, enquanto votos é um array de seis elementos, todos inicializados com zero, que contabiliza o número de votos para cada sistema.

```
let sistemas = ['Windows Server', 'Unix', 'Linux', 'Netware', 'Mac OS', 'Outro'];
let votos = [0,0,0,0,0,0];
```

O loop while (true) é usado para coletar votos dos usuários de forma contínua até que o usuário insira 0 para encerrar a votação. Dentro do loop, um prompt solicita ao usuário que escolha um sistema operacional, representado por números de 1 a 6. O valor digitado é convertido para um número inteiro com parseInt. Se a resposta for 0, o loop é interrompido com break. Caso contrário, se a resposta estiver dentro do intervalo válido (1 a 6), o voto é incrementado no índice correspondente do array votos. Se a resposta não for válida, um alerta informa que o valor é inválido.

```
while (true) {
  let resposta = parseInt(prompt(`Qual o melhor Sistema Operacional para uso em servidores?\n
1 - Windows Server\n2 - Unix\n3 - Linux\n4 - Netware\n5 - Mac OS\n6 - Outro\n`));
  if (resposta === 0) {
    break
  } else if ([1,2,3,4,5,6].includes(resposta)) {
    votos[resposta-1]++;
  } else {
    alert(`Valor inválido`);
  }
}
```

Após a coleta dos votos, a variável somaVotos é calculada usando o método .reduce() para somar todos os votos no array. Esta soma é usada para calcular a porcentagem de votos para cada sistema.

```
let somaVotos = votos.reduce((a,b) => a+b,0);
```

A função `print` é responsável por gerar uma string formatada para exibir os votos e a porcentagem correspondente de cada sistema operacional. Ela recebe um item como argumento, encontra o índice desse item no array `sistemas`, calcula a porcentagem de votos com base no total de votos e retorna uma string formatada com o nome do sistema, o número de votos e a porcentagem.

```
function print(item) {
  let ind = sistemas.indexOf(item);
  let percent = ((votos[ind]/somaVotos)*100).toFixed(2);
  let text = `${item}          ${votos[ind]}          ${percent}%`;
  return text;
}
```

Para determinar qual sistema operacional recebeu o maior número de votos, o código utiliza um loop `for...of` que percorre o array `sistemas`. Durante a iteração, o índice do sistema atual é encontrado, e se o número de votos para esse sistema for maior do que o maior número de votos registrado até então (`sistemaMaisVotadoVotos`), o sistema é atualizado como o mais votado, e a porcentagem correspondente é calculada.

```
let sistemaMaisVotadoVotos = -Infinity;
let porcentagemSistemaMaisVotado = '';
let sistemaMaisVotado = '';
let indice = 0;
for (sistema of sistemas) {
  indice = sistemas.indexOf(sistema);
  if (sistemaMaisVotadoVotos < votos[indice]) {
    sistemaMaisVotadoVotos = votos[indice];
    sistemaMaisVotado = sistema;
    porcentagemSistemaMaisVotado = ((votos[indice]/somaVotos)*100).toFixed(2);
  }
}
```

Finalmente, um `alert` exibe um resumo dos resultados da votação, incluindo uma tabela formatada com os votos e porcentagens para cada sistema operacional e a informação sobre o sistema mais votado, incluindo o número de votos e a porcentagem total.

```
alert(`Sistema Operacional    Votos    %\n-----      ----    ---\n${print(sistemas[0])}\n${print(sistemas[1])}\n${print(sistemas[2])}\n${print(sistemas[3])}\n${print(sistemas[4])}\n${print(sistemas[5])}\n\nO Sistema Operacional mais votado foi o ${sistemaMaisVotado}, com ${sistemaMaisVotadoVotos} votos, correspondendo a ${porcentagemSistemaMaisVotado}% dos votos.`)
```


20) Neste código, começamos criando dois arrays vazios: salarios e abonos. O objetivo é armazenar os salários dos colaboradores e os respectivos abonos calculados.

```
let salarios = [];  
let abonos = [];
```

Utilizamos um loop while (true) para solicitar ao usuário que informe o salário de cada colaborador. Se o usuário inserir 0, o loop é interrompido com o comando break. Caso o valor inserido não seja um número (isNaN(pergunta)), um alerta é exibido solicitando um valor válido. Se o valor for numérico e diferente de zero, ele é adicionado ao array salarios.

```
while (true) {  
  let pergunta = parseInt(prompt('Informe o salário (0 para parar): '));  
  if (pergunta == 0) {  
    break  
  } else if (isNaN(pergunta)) {  
    alert('Informe um valor para continuar!');  
  } else {  
    salarios.push(pergunta);  
  }  
}
```

Depois de coletar todos os salários, o código inicializa três variáveis: maiorAbono, qntAbonosMinimos, e sequenciaSalarios. maiorAbono é iniciado com -Infinity para garantir que qualquer abono calculado será maior que esse valor inicial. qntAbonosMinimos é iniciado em 0 para contar quantos abonos mínimos foram aplicados. sequenciaSalarios é uma string que será usada para armazenar a sequência de salários e abonos formatados para exibição.

```
let maiorAbono = -Infinity;  
let qntAbonosMinimos = 0;  
let sequenciaSalarios = ``;
```

Se o abono calculado for maior do que o valor atual armazenado em maiorAbono, atualizamos maiorAbono com o novo valor. Também adicionamos

uma linha à string `sequenciaSalarios` formatando os salários e abonos para exibição.

```
for (indice in salarios) {
  let abono = salarios[indice] * 0.2;
  if (abono <= 100) {
    abono = 100;
    qntAbonosMinimos++;
  }
  abonos.push(abono);

  if (abono > maiorAbono) {
    maiorAbono = abono;
  }

  sequenciaSalarios += `R${salarios[indice].toFixed(2)} - R${abonos[indice].toFixed(2)}\n`;
}
```

Após o loop, o código exibe um alerta contendo informações detalhadas: uma lista dos salários e abonos, o número total de colaboradores processados, o total gasto com abonos (calculado usando o método `.reduce()`), a quantidade de colaboradores que receberam o valor mínimo de abono e o maior valor de abono encontrado.

```
alert(`Salário      - Abono\n${sequenciaSalarios}\nForam processados ${salarios.length} colaboradores\nTotal gasto com abonos: R${abonos.reduce((a,b)=>a+b,0).toFixed(2)}\nValor mínimo pago a ${qntAbonosMinimos} colaborador(es)\nMaior valor de abono: R${maiorAbono.toFixed(2)}`)
```

- 21) Nessa atividade, começamos criando dois arrays vazios, `modelosCarros` e `consumoCarros`, que irão armazenar, respectivamente, os modelos dos carros e o consumo de cada um deles. Também definimos duas constantes, `distanciaPercorrida` com o valor de 1000 (representando a distância a ser percorrida) e `precoGasolina` com o valor de 2.25 (o preço da gasolina por litro).

```
let modelosCarros = [];  
let consumoCarros = [];  
const distanciaPercorrida = 1000;  
const precoGasolina = 2.25;
```

O código utiliza um loop `while (true)` para permitir ao usuário inserir informações sobre vários carros até decidir parar. Dentro do loop, o programa solicita ao usuário o modelo do carro através de um prompt. Se o modelo fornecido estiver vazio, um alert é exibido solicitando que o nome do carro seja informado. Se o usuário inserir '0', o loop verifica se ao menos um modelo foi fornecido. Se não houver modelos registrados, um alert informa que é necessário informar ao

menos um modelo para continuar. Caso contrário, o loop é encerrado. Se o usuário fornecer um modelo diferente de '0', o código solicita o consumo do carro, converte a entrada para um número de ponto flutuante usando `parseFloat`, e então adiciona o modelo e o consumo aos arrays `modelosCarros` e `consumoCarros`, respectivamente.

```
while (true) {
  let userCarro = prompt('Informe o modelo do carro (0 para parar): ');
  if (userCarro.trim() === '') {
    alert('Informe o nome do carro')
  } else if (userCarro === '0'){
    if (modelosCarros.length == 0) {
      alert('É necessário informar ao menos um modelo para continuar')
    } else {
      break
    }
  } else {
    let userConsumo = parseFloat(prompt('Informe o consumo do carro: '))
    modelosCarros.push(userCarro);
    consumoCarros.push(userConsumo);
  }
}
```

Após a coleta dos dados, o código determina o carro mais econômico. Para isso, inicializa duas variáveis, `consumoCarroMaisEconomico` com `-Infinity` e `carroMaisEconomico` com uma string vazia. Utilizando um loop `for`, o código percorre todos os modelos de carro, encontrando o índice do carro atual nos arrays. Compara o consumo atual com o menor consumo registrado e, se for menor, atualiza `carroMaisEconomico` e `consumoCarroMaisEconomico` com o carro e o consumo atual, respectivamente.

```
let consumoCarroMaisEconomico = -Infinity;
let carroMaisEconomico = '';
for (carro of modelosCarros) {
  let indice = modelosCarros.indexOf(carro);
  if (consumoCarroMaisEconomico < consumoCarros[indice]) {
    carroMaisEconomico = carro;
    consumoCarroMaisEconomico = consumoCarros[indice];
  }
}
```

Em seguida, o código constrói uma mensagem detalhada para ser exibida ao usuário. Utilizando um loop `for`, ele percorre os modelos de carro, calcula a quantidade de gasolina necessária para percorrer a distância definida (dividindo

a distância pelo consumo do carro) e o custo associado (multiplicando a quantidade de gasolina pelo preço da gasolina). Adiciona essas informações à mensagem em formato de string, onde cada linha inclui o modelo do carro, o consumo, a quantidade de gasolina necessária e o custo total.

```
let mensagem = `Relatório Final\n`;
for (item of modelosCarros) {
  let indiceCarro = modelosCarros.indexOf(item);
  let consumoCarro = consumoCarros[indiceCarro];
  let qntGasolina = (distanciaPercorrida/consumoCarro).toFixed(2);
  let custo = (qntGasolina*precoGasolina).toFixed(2)
  mensagem += `${item} - ${consumoCarro.toFixed(2)} - ${qntGasolina} litros - R$${custo}\n`
}
```

Finalmente, o alert exibe a mensagem final com o relatório detalhado de cada carro e o carro mais econômico, incluindo o menor consumo.

```
alert(mensagem+`O menor consumo é do ${carroMaisEconomico}.`)
```

- 22) O código começa criando um array chamado listaContagem com quatro elementos, todos inicializados com zero. Este array será utilizado para armazenar a quantidade de vezes que cada tipo de defeito foi relatado. Além disso, define-se um array perguntas que contém descrições dos quatro tipos possíveis de defeitos que o usuário pode escolher.

```
let listaContagem = [0,0,0,0];
let perguntas = ['1 - Necessita de esfera','2 - Necessita de limpeza',
'3 - Necessita de troca do cabo ou conector','4 - Quebrado ou inutilizado']
```

O programa entra em um loop infinito com while (true), que só será interrompido quando o usuário decidir parar. Dentro desse loop, o código exibe um prompt que solicita ao usuário informar qual é o defeito do mouse, oferecendo quatro opções numeradas. O número informado pelo usuário é convertido para um inteiro com parseInt.

```
while (true) {
  let pergunta = parseInt(prompt(`Qual o defeito do mouse? (0 para parar)
\n\n${perguntas[0]}\n${perguntas[1]}\n${perguntas[2]}\n${perguntas[3]}\n`))
```

Se o número inserido pelo usuário for 0, o loop é encerrado com break. Caso contrário, o código verifica o valor da variável pergunta e atualiza o array listaContagem incrementando o índice correspondente ao defeito selecionado. Se o número fornecido não corresponder a nenhuma das opções válidas, o programa exibe um alert solicitando que o usuário digite um número válido.

```

    if (pergunta == 0) {
        break
    } else if (pergunta == 1) {
        listaContagem[0]++;
    } else if (pergunta == 2) {
        listaContagem[1]++;
    } else if (pergunta == 3) {
        listaContagem[2]++;
    } else if (pergunta == 4) {
        listaContagem[3]++;
    } else {
        alert('Digite um número de identificação válido para fazer a contagem!')
    }
}

```

Após sair do loop, o código calcula a quantidade total de mouses processados utilizando o método `.reduce()`, que soma todos os valores do array `listaContagem`. Em seguida, cria-se uma variável `mensagem` para formatar e armazenar as informações a serem exibidas. Um loop `for` percorre o array `listaContagem`, calculando o percentual de cada tipo de defeito em relação ao total de mouses. Este percentual é formatado com duas casas decimais usando `.toFixed(2)` e a informação é adicionada à variável `mensagem`.

```

let qntMouses = listaContagem.reduce((a,b)=>a+b,0);

let mensagem = ``;
for (i=0; i<listaContagem.length; i++) {
    let percentual = ((listaContagem[i]/qntMouses)*100).toFixed(2)
    mensagem += `${perguntas[i]}          ${listaContagem[i]}          ${percentual}%\n`
}

```

Finalmente, o código utiliza um `alert` para exibir a quantidade total de mouses e uma tabela com a situação de cada tipo de defeito, a quantidade registrada e o percentual correspondente. A tabela é formatada para apresentar claramente os dados coletados.

```

alert(`Quantidade de mouses: ${qntMouses}\n\nSituação
Quantidade      Percentual\n${mensagem}`)

```

- 23) No código fornecido, começamos criando duas variáveis, `usuarios` e `espacoUsuarios`, que são arrays destinados a armazenar os nomes dos usuários e o espaço em disco que cada um utiliza, respectivamente. O código então entra em um loop `while (true)` que permite ao usuário inserir informações continuamente até que decida parar.

```
let usuarios = [];  
let espacoUsuarios = [];
```

Dentro do loop, o código solicita ao usuário que informe o nome de um usuário por meio de um prompt. Se o usuário deixar o campo em branco, um alerta é exibido pedindo que insira um nome válido. Se o usuário digitar '0', o loop é interrompido com break, encerrando a coleta de dados. Caso contrário, o código solicita o espaço em disco utilizado por aquele usuário em bytes. Esse valor é convertido para um número inteiro com parseInt e, em seguida, tanto o nome do usuário quanto o espaço em disco são adicionados aos arrays usuarios e espacoUsuarios.

```
while (true) {  
  let perguntaUser = prompt('Informe o usuário (0 para parar): ');  
  if (perguntaUser.trim() == '') {  
    alert('Digite algo para continuar')  
  } else if (perguntaUser === '0') {  
    break  
  } else {  
    let perguntaEspaco = parseInt(prompt('Informe o espaço em disco utilizado por ${perguntaUser} em Bytes: '));  
    usuarios.push(perguntaUser);  
    espacoUsuarios.push(perguntaEspaco);  
  }  
}
```

Após a coleta de dados, o código calcula o espaço total ocupado por todos os usuários somando os valores no array espacoUsuarios usando o método .reduce(). O espaço médio ocupado é então calculado dividindo o espaço total pelo número de usuários, e o resultado é arredondado para duas casas decimais com .toFixed(2).

```
let espacoTotal = espacoUsuarios.reduce((a,b)=>a+b,0);  
let espacoMedio = (espacoTotal/usuarios.length).toFixed(2);
```

Para formatar a exibição dos resultados, o código define duas funções. A primeira, converterBytes, converte bytes em megabytes dividindo o valor por 1024 elevado à segunda potência e arredondando o resultado para duas casas decimais. A segunda função, calcularPorcentagem, calcula a porcentagem do espaço ocupado por um usuário em relação ao espaço total, multiplicando o valor em bytes por 100 e dividindo pelo espaço total, novamente arredondado para duas casas decimais.

```
function converterBytes(bytes) {
    return (bytes / 1024**2).toFixed(2);
}

function calcularPorcentagem(bytes){
    return ((bytes / espacoTotal)*100).toFixed(2);
}
```

Em seguida, o código constrói uma string mensagem que formata os dados dos usuários. Usando um loop for...of, ele itera sobre o array usuarios, obtendo o índice de cada usuário no array e construindo uma linha formatada que inclui o número do usuário, o nome, o espaço utilizado em megabytes e a porcentagem de uso.

```
let mensagem = ``;
for (user of usuarios) {
    let indice = usuarios.indexOf(user);
    mensagem += `${indice+1}    ${user}    ${converterBytes(espacoUsuarios[indice])} MB
    ${calcularPorcentagem(espacoUsuarios[indice])}%\n`;
}
```

Finalmente, a string mensagem é exibida em um alerta, junto com o título e o cabeçalho formatado, o espaço total e médio ocupado, ambos convertidos para megabytes com a função converterBytes. Isso fornece uma visão clara e formatada do uso do espaço em disco pelos usuários.

```
alert(`ACME Inc.          Uso do espaço em disco pelos usuários
\n-----\nNr.
Usuário    Espaço utilizado    % do uso\n\n${mensagem}\nEspaço total ocupado:
${converterBytes(espacoTotal)} MB\nEspaço médio ocupado: ${converterBytes(espacoMedio)} MB`)
```

- 24) Para começar, defini uma variável chamada resultados, que é um array vazio destinado a armazenar os resultados dos lançamentos dos dados. Em seguida, criei uma função chamada dado, que gera um número aleatório entre 1 e 6, simulando o lançamento de um dado. A função utiliza Math.random() para obter um número aleatório entre 0 e 1, multiplica esse número por 6, usa Math.floor() para arredondar o valor para baixo e adiciona 1 para garantir que o número esteja entre 1 e 6. O resultado é então adicionado ao array resultados com o método .push().

```

let resultados = [];

function dado() {
  let resultado = Math.floor(Math.random() * 6) + 1;
  resultados.push(resultado)
}

```

Depois, utilizei um loop for que itera 100 vezes, chamando a função dado em cada iteração para simular o lançamento do dado 100 vezes e armazenar todos os resultados no array resultados.

```

for (i = 0; i < 100; i++) {
  dado()
}

```

Para exibir a frequência de cada número lançado, criei uma variável texto e iniciei com o cabeçalho da tabela: 'Número Frequência'. Em seguida, usei outro loop for que itera de 0 a 5, representando cada possível resultado do dado (1 a 6). Dentro deste loop, iniciei uma variável contagem com 0 para contar quantas vezes cada número aparece no array resultados. Utilizei um loop for...of para percorrer cada item do array resultados e, se o item corresponder ao número atual do loop (i.e., i2 + 1), incrementa a contagem.

```

let texto = 'Número    Frequência'
for (i2 = 0; i2 < 6; i2++) {
  let contagem = 0;
  for (item of resultados) {
    if (item == i2+1) {
      contagem++
    }
  }
}

```

Após contar as ocorrências de cada número, adicionei uma linha ao texto com a frequência do número correspondente. A formatação é feita utilizando interpolação de strings para garantir que os números e as frequências apareçam corretamente alinhados.

```

}
texto += ` \n      ${i2+1}          ${contagem}`
}

```


Finalmente, exibi o conteúdo da variável “texto” em um alert, mostrando a tabela com a frequência de cada número obtido nos lançamentos dos dados.

```
alert(texto)
```