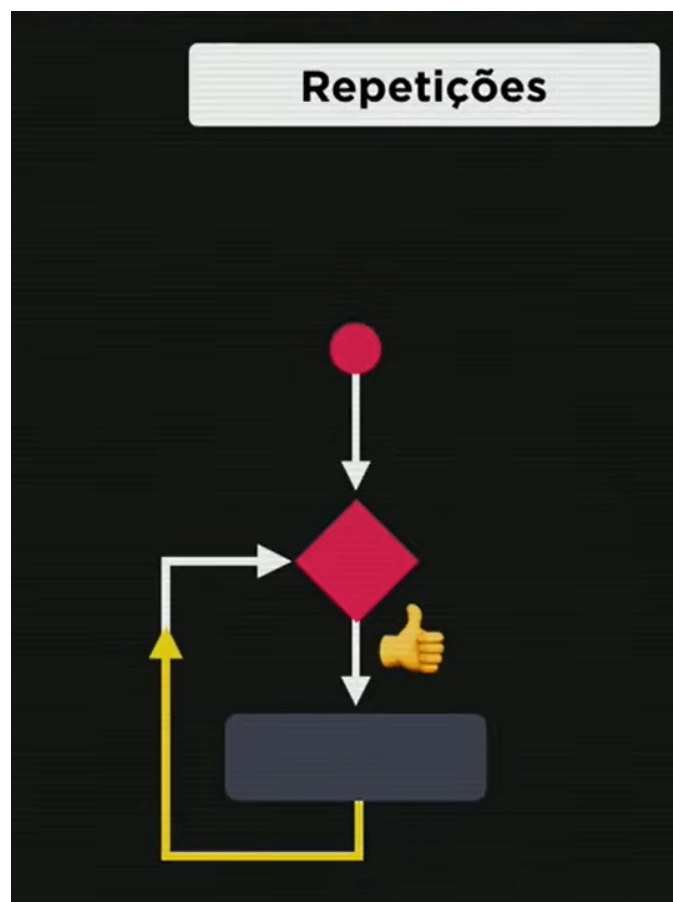


Módulo E: Repetições em JavaScript

▼ Aula 13 - Repetições (Parte 1)

O laço de repetição é outro tipo de estrutura de controle na programação. Testa condições também, mas, diferente do if-else, repete a execução de um bloco.



Estrutura de um laço de repetição. Fonte: Curso em Vídeo.

Enquanto a condição for verdadeira, em um loop, o bloco de código continuará a ser executado.

Usar laços de repetição reduzem o número de linhas de código. 😊

Estruturas de repetição

```

while (condição) {
    // código
} // se a condição for verdadeira, o laço continua a se
repetir. Se for falsa, para.

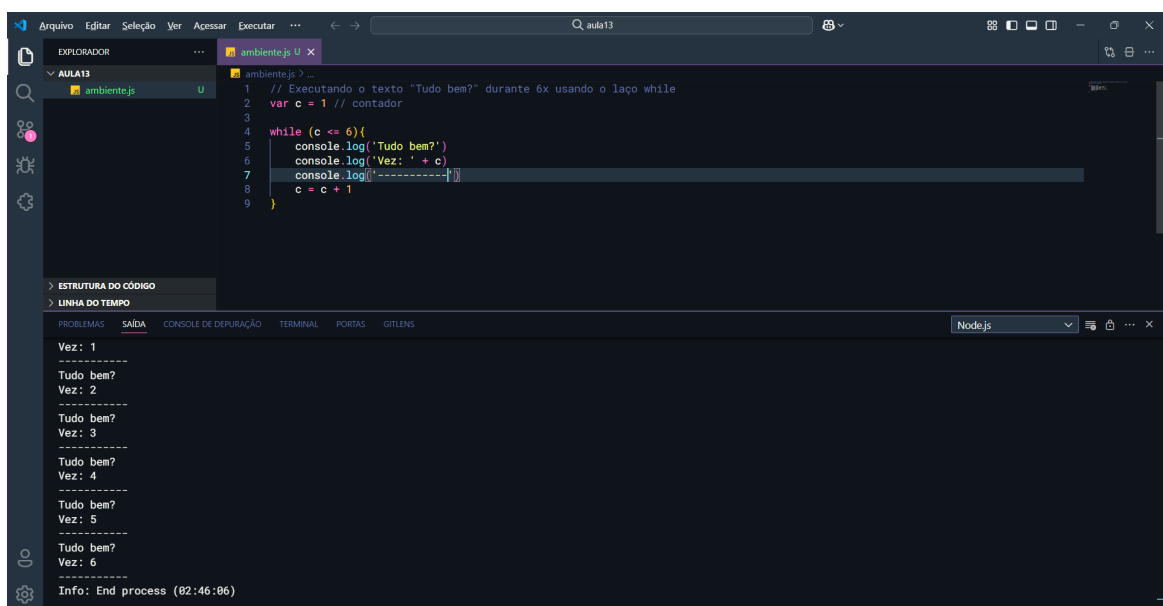
// Estrutura do while
do {
    // código
} while (condição) // no do while, a condição explicada
ao final do bloco do código

// Estrutura for
for (contador; condição; incremento){
    // código
} // no for, tudo é declaração na mesma linha

```

while = ENQUANTO

1o exercício e execução (usando `while`):



The screenshot shows a code editor with a file named 'aula13'. The code defines a variable 'c' and uses a 'while' loop to print 'Tudo bem?' six times, incrementing 'c' each time. The output in the console shows the text 'Tudo bem?' repeated six times, each preceded by 'Vez: ' and a number from 1 to 6. The console also shows 'Info: End process (02:46:06)'.

```

1 // Executando o texto "Tudo bem?" durante 6x usando o laço while
2 var c = 1 // contador
3
4 while (c <= 6){
5     console.log('Tudo bem?')
6     console.log('Vez: ' + c)
7     console.log('-----')
8     c = c + 1
9 }

```

Output in console:

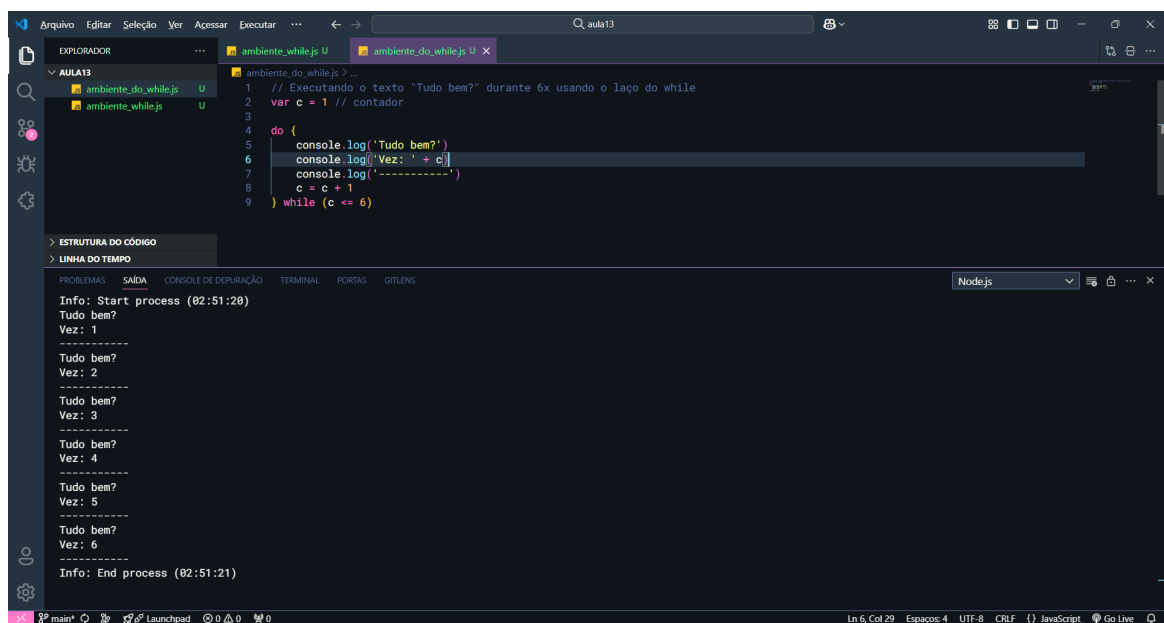
```

Vez: 1
Tudo bem?
Vez: 2
Tudo bem?
Vez: 3
Tudo bem?
Vez: 4
Tudo bem?
Vez: 5
Tudo bem?
Vez: 6
Tudo bem?
Info: End process (02:46:06)

```

Fonte: Compilação da autora.

2o exercício e execução (usando `do while`):



```
1 // Executando o texto "Tudo bem?" durante 6x usando o laço do while
2 var c = 1 // contador
3
4 do {
5     console.log("Tudo bem?")
6     console.log("Vez: " + c)
7     console.log("-----")
8     c = c + 1
9 } while (c <= 6)
```

Info: Start process (02:51:20)
Tudo bem?
Vez: 1

Tudo bem?
Vez: 2

Tudo bem?
Vez: 3

Tudo bem?
Vez: 4

Tudo bem?
Vez: 5

Tudo bem?
Vez: 6

Info: End process (02:51:21)

▼ Exercícios JavaScript (Parte 5)

Código HTML:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Modelo de Exercício</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <!-- Cabeçalho -->
    <header>
        <h1>Vamos contar...</h1>
    </header>

    <!-- Seção -->
    <section>
        <!-- div 1 -->
```

```

        <div id="dados">
            <label for="">Início:</label>
            <input type="number" name="inicio" id="txt
i">

            <br><br>

            <label for="">Fim:</label>
            <input type="number" name="fim" id="txtf">
            <br><br>

            <label for="">Passo:</label>
            <input type="number" name="passo" id="txtp">
            <p><input type="button" value="Contar" oncli
ck="contar()"></p> <!-- Quando o botão "Contar" for pres
sionado, a função contar() será chamada -->
        </div>

        <br>

        <!-- div 2 -->
        <div id="res">
            Preparando a contagem...
        </div>
    </section>

    <!-- Rodapé -->
    <footer>
        <p>&copy; GabriellaXavier</p> <!-- Símbolo de co
pyright-->
    </footer>
    <!-- Ligação com o código JS -->
    <script src="script.js"></script>
</body>
</html>

```

Código JS:

```

// Criando a função contar()
function contar() {
    let ini = document.getElementById('txti') // pega o
    valor do campo cujo id é 'txti' e armazena na variável
    'ini'
    let fim = document.getElementById('txtf') // pega o
    valor do campo cujo id é 'txtf' e armazena na variável
    'fim'
    let passo = document.getElementById('txtp') // pega
    o valor do campo cujo id é 'txtp' e armazena na variável
    'passo'
    let res = document.getElementById('res') // pega o q
    ue tá na div 'res' e atribuí à variável 'res'

    // Se o tamanho de ini, ou fi, ou passo for igual a
    0...
    if (ini.value.length == 0 || fim.value.length == 0 |
    | passo.value.length == 0) {
        res.innerHTML = 'Impossível contar!' // Apresent
        ar essa msg
    }
    else { // Senão...
        res.innerHTML = 'Contando: <br>'
        // Convertendo de texto para número:
        let i = Number(ini.value)
        let f = Number(fim.value)
        let p = Number(passo.value)

        if (p <= 0) { // o que for definido nesse if val
        erá nos outros. Se inicialmente valer = 0, logo passará
        a ser = 1 e assim continuará o fluxo
            window.alert('Passo inválido! Considerando P
            ASSO 1')
            p = 1
        }

        if (i < f) {
            // Contagem crescente

```

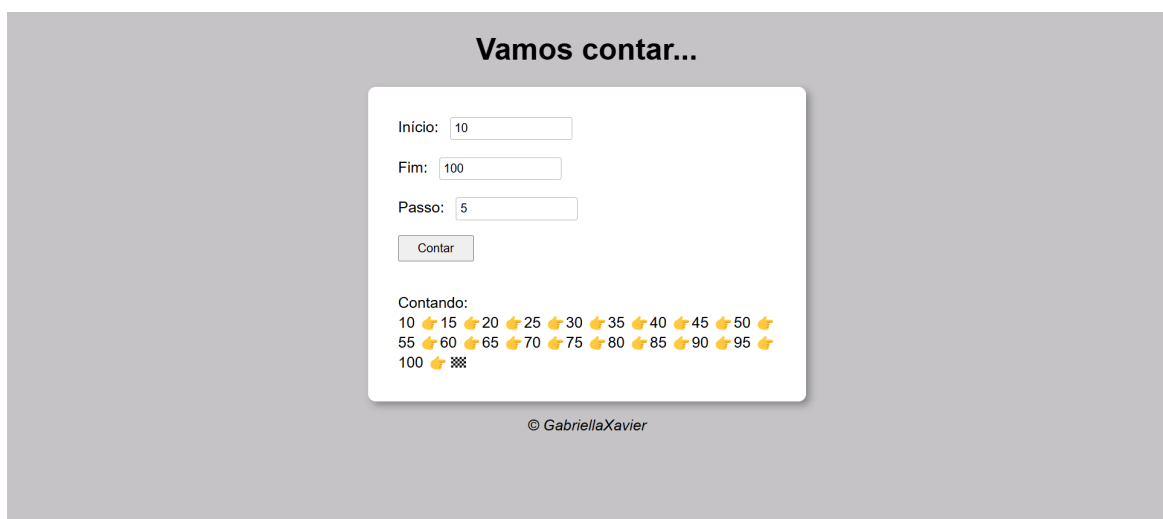
```

        for(let c = i; c <= f; c += p){ // c += p é
o mesmo que c = c + p
            res.innerHTML += `${c} \u{1F449}` // emo
ji do unicode: 🖐️
        }
    }
    else {
        // Contagem regressiva:
        for(let c = i; c >= f; c -= p){ // c -= p é
o mesmo que c = c - p
            res.innerHTML += `${c} \u{1F449}` // emo
ji do unicode: 🖐️
        }
    }

    res.innerHTML += `\u{1F3C1}` // emoji do unicod
e: 🏁
    }
}

```

Resultado:



Fonte: Compilação da autora.

▼ Exercícios JavaScript (Parte 6)

Criando uma tabuada com HTML e JavaScript

Código HTML:

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Tabuada</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <!-- Cabeçalho -->
  <header>
    <h1>Tabuada</h1>
  </header>
  <!-- Seção -->
  <section>
    <!-- div 1 -->

    <div id="inputs">
      <label for="">Número:</label>
      <input type="number" name="num" id="txtn">
      <input type="button" value="Gerar Tabuada" id="btnTab" onclick="tabuada()"> <!-- chama a função tabuada() quando o botão é clicado -->
      <input type="button" value="Limpar" id="btnLimpar" onclick="limpaRes()"> <!-- chama a função limpaRes() quando o botão é pressionado -->
    </div>

    <br>

    <!-- div 2 -->
    <div>
      <select name="tabuada" id="seltab" size="10">
```

```

        <option>Digite um número:</option>
    </select>

</div>
</section>
<!-- Rodapé -->
<footer>
    <p>&copy; GabriellaXavier</p> <!-- Símbolo de co
pyright-->
</footer>
<script src="script.js"></script>
</body>
</html>

```

Código JS:

```

function tabuada(){
    let num = document.getElementById('txtn') // pega o
    número informado pelo usuário e armazena-o na variável n
    um

    let tab = document.getElementById('seltab') // assoc
    iação com o select

    if (num.value.length == 0){
        // Se o campo estiver vazio, essa msg será exibi
        da para o usuário
        window.alert('Por favor, digite um número!')
    }
    else {
        // Se o campo não estiver vazio quando o botão f
        or clicado, então será feita uma conversão de dados
        let n = Number(num.value)

        let c = 1 // contador

        tab.innerHTML= '' // limpa o select antes da tab
        uada começar
    }
}

```



```

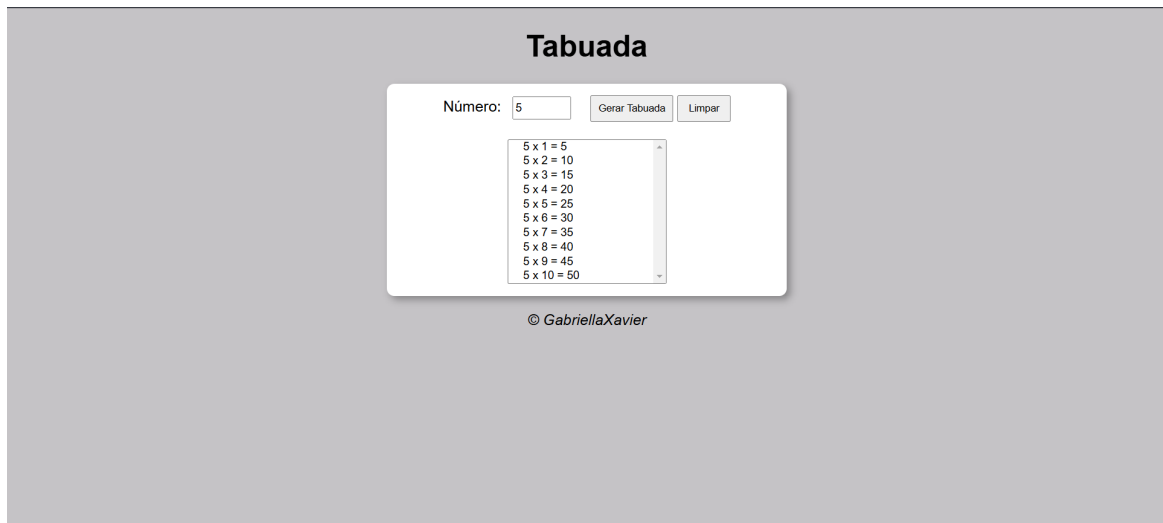
        // Laço para calcular a tabuada
        while (c <= 10){
            let item = document.createElement('option')
            // criando opções dentro do select
            item.text = `${n} x ${c} = ${n * c}` // cálculo e resultado da tabuada dentro do select
            item.value = `tab${c}` // para poder selecionar as opções exibidas da tabuada
            tab.appendChild(item) // adiciona o elemento
            c++ // incremento
        }
    }
}

function limpaRes(){ // função que limpa os resultados da tela
    let tab = document.getElementById('selctab') // referência ao select
    tab.innerHTML = '' // Limpa todas as opções do select

    let num = document.getElementById('txtn') // referência ao campo txtn, onde o usuário informa um valor
    num.value = '' // Limpa o campo de texto (input)
}

```

Resultado da execução:



Tabuada em JS. Fonte: Compilação da autora.

Outras informações sobre JS

▼ `const`

É um tipo de variável que, uma vez atribuído um valor a ela, não dá para alterar. É como se fosse fixo.

▼ `let`

- **Escopo de Bloco:** A variável existe apenas no bloco onde foi declarada (entre `{ }`).

```
if (true) {  
  let nome = "João";  
  console.log(nome); // "João"  
}  
console.log(nome); // Erro: nome is not defined
```

- **Reatribuição permitida:** Você pode atribuir novos valores à variável.

```
let idade = 25;
idade = 30;
console.log(idade); // 30
```

- **Não permite declarações duplicadas** no mesmo escopo.

```
let cidade = "São Paulo";
let cidade = "Rio de Janeiro"; // Erro: Identifier
'cidade' has already been declared
```

- **Temporal Dead Zone (TDZ):** A variável não pode ser acessada antes de ser declarada no código.

```
console.log(x); // Erro: Cannot access 'x' before ini
tialization
let x = 10;
```

▼ Diferenças `var` x `let`

A principal diferença entre `let` e `var` em JavaScript é o **escopo** e o **comportamento** de declaração.

1. Escopo:

- `let` tem **escopo de bloco**, ou seja, a variável declarada com `let` é acessível apenas dentro do bloco `{ }` onde foi definida (como em um `if`, `for` ou função).
- `var` tem **escopo de função**, o que significa que a variável declarada com `var` é acessível em toda a função, mesmo que declarada dentro de um bloco (como um `if`).

2. Declaração duplicada:

- `let` não permite que a mesma variável seja declarada mais de uma vez no mesmo escopo.

- `var` permite declarar a mesma variável várias vezes no mesmo escopo, o que pode causar confusão ou erros.

3. Hoisting:

- `let` é "hoisted" (elevação) para o topo do seu bloco, mas não pode ser acessado até a linha onde foi declarado, resultando na "**zona morta temporal**".
- `var` é "hoisted" para o topo da função ou escopo global e pode ser acessado antes da declaração, mas com valor `undefined`.

Resumo:

- `let` é mais seguro e previsível, com escopo de bloco e restrição a reatribuições no mesmo escopo.
- `var` é mais antigo, tem escopo de função e permite declarações duplicadas no mesmo escopo.