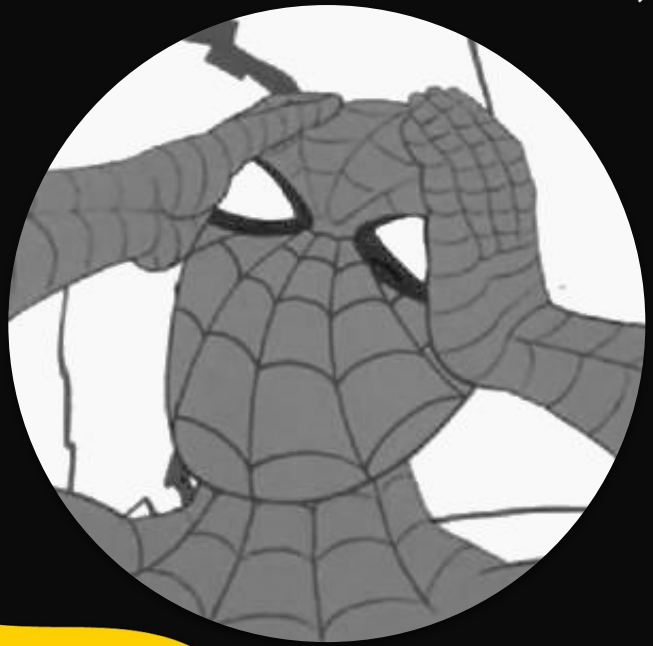
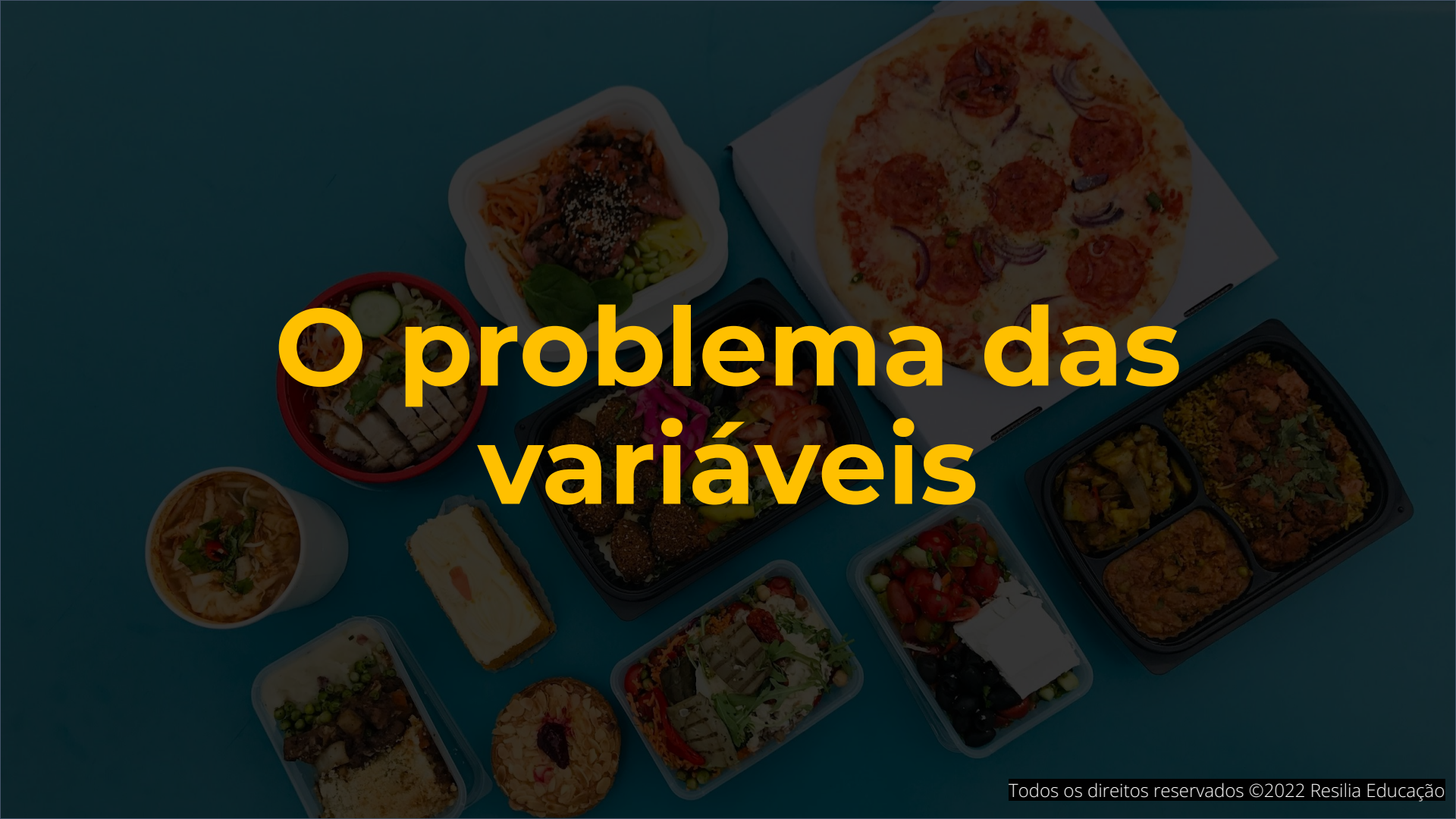
De novo não



Fonte: Meme Genarator

Módulo 1 - Aula 9 - HARD

RESILIA | Senac

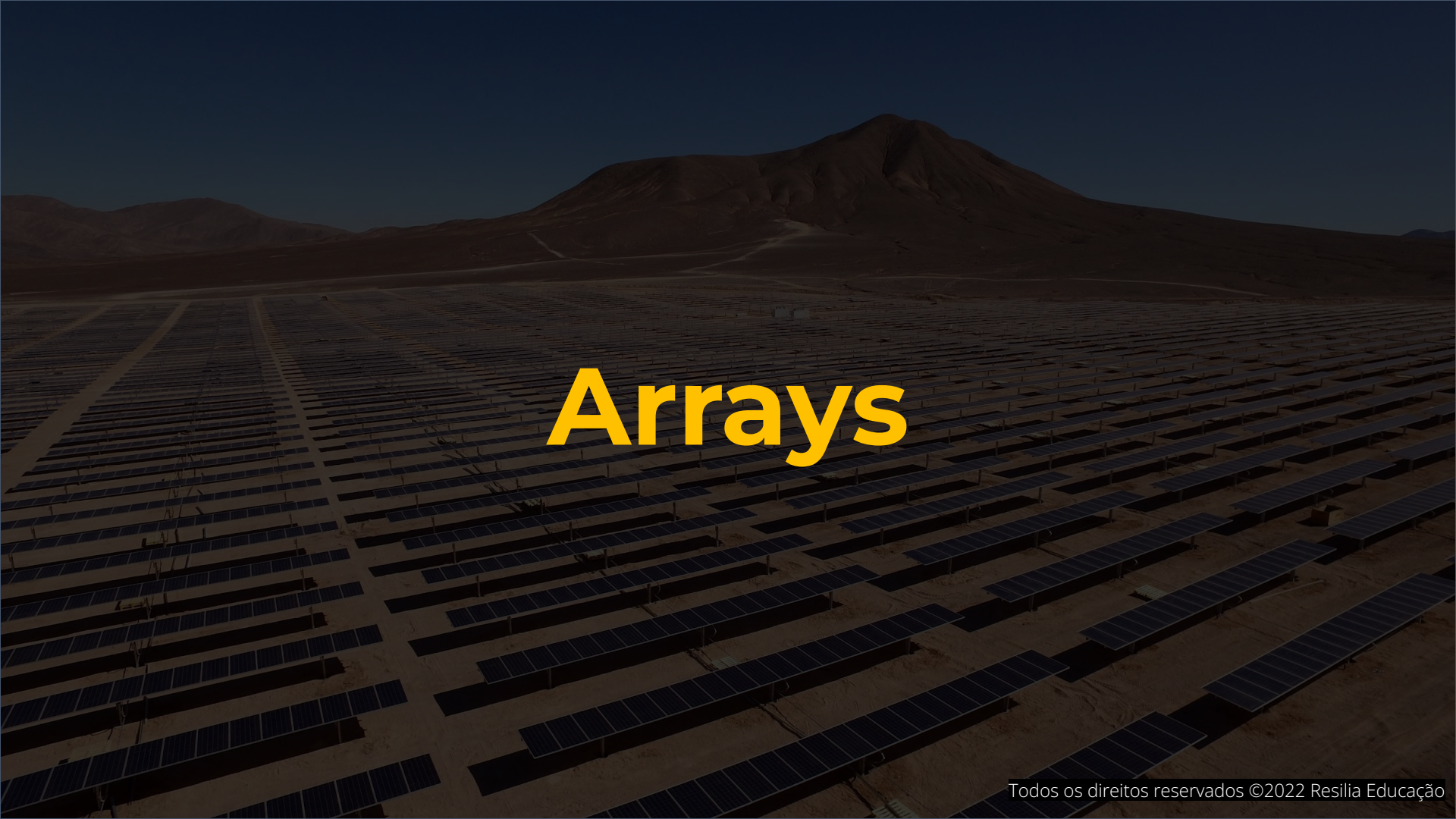


O problema das variáveis

O problema das variáveis

Variáveis tradicionais (numbers, strings) são uma **estrutura de dados** capaz de **guardar** apenas **um elemento**.
Em alguns cenários, faria mais sentido guardar um **conjunto de dados** em uma **única variável**.
Ex - Lista de aniversário:

```
var convid1 = 'Ivan';
var convid2 = 'Gabi';
var convid3 = 'Joka';
var convid4 = 'Tati';
```



Arrays

Arrays são uma **estrutura de dados**/tipo do JavaScript capaz de armazenar uma **coleção de dados** em uma única variável.

Ex:

```
var listaConvid = [
    'Jessica',
    'Edu',
    'Felipe',
    'Samantha'];
```

Arrays: declaração

Para **declarar** um **array** podemos utilizar **duas notações** diferentes.
São elas:

```
var array1 = [];
var array2 = new Array();
```

Array: inicialização

Podemos **adicionar valores** nos **arrays** no momento em que **declaramos** eles.
O **número de elementos** adicionados na inicialização pode **variar**.
Ex:

```
var listaConvid1 = ['Kelwin', 'Thaís', 'Paulinha'];
var listaConvid2 = new Array('Fábio', 'Dudu', 'Vic', 'Jansen');
```

Array: inserção

Podemos **inserir valores** no **final de arrays** utilizando a função **push**.
O **método push** pode receber um **número variável** de **elementos**.
Ex:

```
var listaConvid = [];
listaConvid.push('Amanda');
listaConvid.push('Nina', 'Bruno', 'Carol');
```

Arrays: remoção

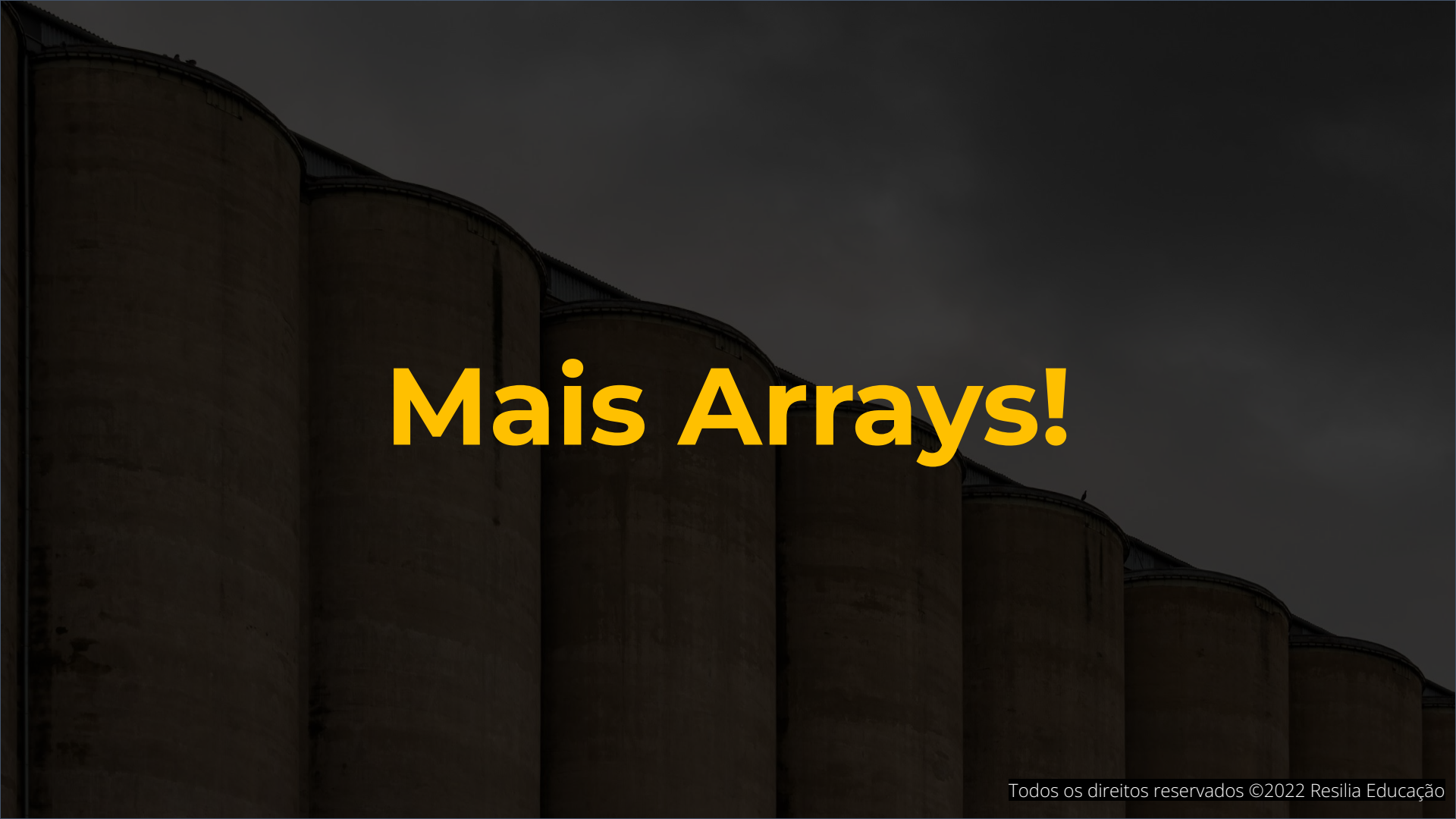
Podemos **remover o último elemento** de um **arrays** utilizando a função **pop**.
Ex:

```
var listaConvid = ['Vê', 'Thomaz', 'Cinthia', 'Luiz', 'Nic'];
var desconvidados = [listaConvid.pop(), listaConvid.pop()];
```

Arrays e loops

Como **arrays** armazenam **diversos elementos** eles naturalmente são **estruturas iteráveis**. Isso quer dizer que podemos utilizar **estruturas de repetição** para acessar todos os elementos de um array.
Ex:

```
var listaConvid = ['Vê', 'Thomaz', 'Cinthia', 'Luiz', 'Nic'];
var indice = 0;
while( indice < listaConvid.length ) {
    console.log(listaConvid[indice]);
    indice++;
}
```



Arrays

Arrays são uma **estrutura flexível**, capaz de armazenar um **conjunto de valores** em uma única variável.

Ex - Sorteios da Loteria:

```
//sorteio específico
var sorteio54454 = [12, 32, 45, 60, 42, 7];
//histório dos sorteios
var sorteios = [
    [32, 15, 27, 55, 12, 38],
    [25, 17, 13, 22, 06, 01],
    [16, 18, 15, 38, 23, 29]
];
```

Arrays: deleção de elemento

Para **remover elementos** em uma **posição específica**, podemos utilizar o método **splice()**. Ele recebe como **argumentos** a **posição de remoção** e a **quantidade de elementos** devem ser removidos.
Seu **retorno** é a **parte "cortada"** do array e ele **modifica o array original**.
Ex - lista da feira:

```
var feira = ['goiaba', 'maçã', 'couve', 'cenoura'];
feira = feira.splice(2,2);
```

Arrays: substituição

Podemos **alterar** um **elemento** de um **array**, a partir de sua **posição**, **atribuindo** à posição um **novo valor**.
Ex - Lista da feira:

```
var feira = ['goiaba', 'maçã', 'couve', 'cenoura'];
feira[0] = 'pastel';
```

Loops e repetição

Estruturas de repetição: while

O laço **while** é executado **executado** enquanto a **condição** fornecida for **verdadeira**.

Para **controlá-lo**, precisamos de algumas **estruturas auxiliares**.

Ex:

```
var nMastigadas = 0;
while(nMastigadas < 20) {
    console.log(`Mastigada de número ${nMastigadas} realizada`);
    nMastigadas++;
}
console.log('Finalmente engoliu!');
```

Estruturas de repetição: for

A palavra reservada **for** (para) **executa** o **bloco de código** associado a ela (entre chaves).

O **for** recebe "**argumentos**" de:

- Inicialização
- Condição
- Expressão final

Ex:

```
var feira = ['goiaba', 'maçã', 'couve', 'cenoura'];
for (var i=0; i < feira.length; i++) {
    console.log(`Fui à feira e comprei: ${feira[i]}`);
}
```

For: inicialização

A **inicialização** em um **laço for** é utilizada para **declarar e inicializar** uma **variável**.

Geralmente essa **variável** é um **contador** que será **utilizado** na **iteração** e na **condição**. **Após** a inicialização deve-se **utilizar ;** para separar/**finalizar** a **expressão**.

Ex:

```
var feira = ['goiaba', 'maçã', 'couve', 'cenoura'];
for (var i=0; i < feira.length; i++) {
    console.log(`Fui à feira e comprei: ${feira[i]}`);
}
```

For: condição

A **condição** em um **laço for** é utilizada para **validar** a próxima iteração/execução de código.

Geralmente ela é **utilizada comparando** a **variável** de controle/**contagem** com um limitador. **Após** a condição deve-se **utilizar ;** para separar/**finalizar** a **expressão**.

Ex:

```
var feira = ['goiaba', 'maçã', 'couve', 'cenoura'];
for (var i=0; i < feira.length; i++) {
    console.log(`Fui à feira e comprei: ${feira[i]}`);
}
```

For: expressão final

A **expressão final** em um **laço for** é utilizada para **executar** uma **instrução** ao **final** de uma **iteração**/execução de código.

Geralmente ela é **utilizada incrementando** a **variável** de controle/**contagem**.

Ex:

```
var feira = ['goiaba', 'maçã', 'couve', 'cenoura'];
for (var i=0; i < feira.length; i++) {
    console.log(`Fui à feira e comprei: ${feira[i]}`);
}
```

For e arrays

For loops e **arrays** são uma **combinação** direta e **poderosa**.

Com a **sintaxe** de declaração do **for** conseguimos **acessar** todos os **elementos** de um **array** com **poucas linhas de código**.

Laços internos

Em alguns cenários precisamos utilizar **laços de repetição internos**.

Exemplo:

Temos um array com listas de gastos anotados na semana.

Como chegamos a soma de gastos em cada semana?

```
var gastos = [
    [32.98, 17.3, 28.4, 55.2],
    [25.7, 17.13, 13.00]
]
```

Laços internos: possível solução

```
for (var i=0; i < gastos.length; i++) {
    var totalSemana=0;
    for(var j=0; j < gastos[i].length; j++) {
        totalSemana += gastos[i][j];
    }
    console.log(`Gasto na semana ${i}: ${totalSemana}`);
    totalSemana = 0;
}
```

Laços internos: solução melhorada

```
function somaArr(arr) {
    var total = 0;
    for (var i=0; i < arr.length; i++) {
        total += arr[i];
    }
    return total;
}

for (var i=0; i < gastos.length; i++) {
    console.log(`Gasto na semana ${i}: ${somaArr(gastos[i])}`);
}
```

Atividade: Linhas e triângulos

Utilizando loops for e arrays, faça o que é pedido a seguir:

ASCII art é a criação de peças artísticas utilizando símbolos ASCII para formar imagens.

Nessa atividade você dará o primeiro passo para a sua carreira de artista da programação.

Em uma nova pasta, utilizando arquivos HTML e JavaScript, desenvolva um programa que:

- Implemente a função linhaAsterisco. Essa função:
  - Recebe um número
  - Retorna uma string com a quantidade de asteriscos igual ao argumento passado
- Teste a função linhaAsterisco com diferentes argumentos utilizando a página HTML
- Implemente a função trianguloAsterisco. Essa função:
  - Recebe um número (n)
  - Retorna um triângulo de asteriscos com o mesmo número de linhas que o argumento passado (n), um asterisco na primeira linha e n asteriscos na última linha
- Teste a função trianguloAsterisco com diferentes argumentos utilizando a página HTML

Atividade: Linhas e triângulos - Exemplos

linhaAsterisco(1); → Retorna: '*'

linhaAsterisco(3); → Retorna: '***'

linhaAsterisco(5); → Retorna: '*****'

trianguloAsterisco(1); → Retorna: '*'

trianguloAsterisco(3); → Retorna:
```
'*
**
***'
```

trianguloAsterisco(5); → Retorna:
```
'*
**
***
****
*****'
```