Introdução aos Arrays: material complementar.

Hoje, vamos explorar um conceito fundamental chamado "arrays". Não se preocupe se você nunca mexeu com isso; estamos aqui para guiá-lo passo a passo.

O que é um Array?

Em programação, um array também chamado de **vetor** ou **arranjo** é como uma caixa que pode armazenar vários valores sob um único nome. Pense nisso como uma lista de compras, onde você pode ter vários itens organizados em um só lugar.

- Quando pensamos em uma **variável**, podemos pensar que ela é uma caixinha que pode armazenar um valor. *Você lembra disso? Caso não lembre, volte até os slides da Aula 01 para revisar variáveis (página 20).*

- Já o array é como uma grande caixa que possui dentro dele outras várias caixinhas. Em cada caixinha, ele pode armazenar um valor diferente (mas do mesmo tipo de todos os outros).

Como assim “uma caixa com caixas” 🤔? Do “mesmo tipo”?!

Exemplo:

Imagine que você está **fazendo uma lista para ir à feira.** Cada posição da lista é um item a ser comprado na feira e todos precisam fazer sentido!

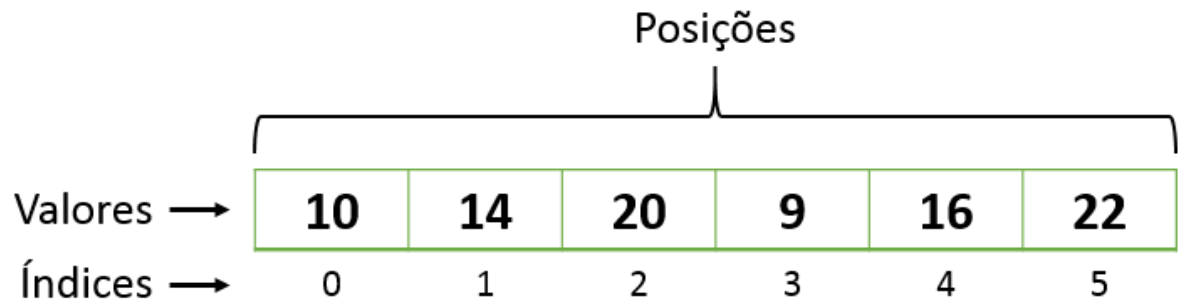
```
Lista da Feira:

- Banana
- Maçã
- Uva
- Mamão
- Fusca azul
```

Se analisarmos essa lista, tenho certeza que você concorda que o **“Fusca azul” não faz sentido estar nessa lista.** Certo?

Mas por qual motivo não faz sentido? Porque o fusca azul é de outro tipo! Ele não é do mesmo grupo do que os outros itens da lista.

Então, por isso, em um array conseguiremos apenas armazenar elementos de um mesmo tipo, que no nosso caso serão strings números (int / float), entre outros…

Estrutura de um Array



Em um array nós temos os valores e os índices.

Os **valores são o conteúdo do seu array**, ou seja, o que você quer guardar dentro deles. Os **índices indicam a posição** (que começa no 0). Nesse exemplo da foto, o valor 10 está na posição 0 e o 14 na posição 1.

Como Criar um Array

Vamos começar com um exemplo simples em JavaScript. Cada elemento é armazenado em uma posição específica e pode ser acessado usando seu índice.

Os colchetes: [ ] declaram um array.

As vírgulas: , separam os elementos de um array.

```
// Criando um array de números
let meuArray = [1, 2, 3, 4, 5];
```

```
// Acessando o primeiro elemento
let primeiroElemento = meuArray[0];
console.log(primeiroElemento); // Isso imprimirá 1
```

Para que Podemos Usar Arrays?

Arrays são incrivelmente versáteis e podem ser usados para diversas finalidades:

- **Armazenamento de Dados:** Podemos armazenar listas de informações, como números, strings, objetos e até mesmo outros arrays.
- **Ordenação e Filtragem:** Permitem organizar e manipular dados de maneira eficiente.
- **Iteração e Loops:** Facilitam a execução de operações em conjunto para cada elemento do array. **Por exemplo:** somar número+10 para cada número de um array.

Explicando:

1. Primeiro, apenas criamos nosso array. Ele contém 5 números e as posições dele vão de 0 até 4.

```
// Criando um array de números
let meuArray = [1, 2, 3, 4, 5];
```

2. Criamos um laço de repetição for.

Vamos entender o que queremos dizer com esse laço for.

```
// Iterando sobre cada número e somando 10
for (let posicao = 0; posicao < meuArray.length; posicao++) {
  meuArray[posicao] = meuArray[posicao] + 10;
}
```

- **`let posicao = 0;`**: Inicializamos uma variável **`posicao`** com 0, que representa o primeiro índice do array.
- **`posicao < meuArray.length;`**: A condição para continuar o loop é que **`posicao`** seja menor que o comprimento do array `meuArray`.

  * o *`.lenght`* encontra o tamanho do `meuArray`, neste caso é 5.

- `posicao++`: Após cada iteração, incrementamos `posicao`, ou seja, somamos +1 para avançar para o próximo índice.

Dentro do loop:

- **`meuArray[posicao] = meuArray[posicao] + 10;`**: Para cada elemento no índice atual `posicao`, adicionamos 10 ao valor existente. Estamos modificando diretamente o array original.

3. Por fim, exibimos o resultado no console, mostrando o array após a iteração e adição de 10 a cada elemento.

```
// Exibindo o resultado
console.log("Array após somar 10 a cada elemento:", meuArray);
```

A saída esperada será: [11, 12, 13, 14, 15]

Isso demonstra como o loop `for` é utilizado para iterar sobre os elementos de um array e como podemos modificar esses elementos durante o processo.

Remover e adicionar elementos a Arrays.

Em JavaScript, você pode adicionar e remover elementos de arrays de várias maneiras. Vamos explicar como realizar as operações mais básicas.

Adicionando Elementos:

`push()` - Adicionar ao Final:

O método **`push()`** adiciona um ou mais elementos ao final de um array.

```
let frutas = ["Maçã", "Banana", "Laranja"];
frutas.push("Morango", "Abacaxi");
// Agora, frutas é ["Maçã", "Banana", "Laranja", "Morango", "Abacaxi"]
```

Removendo Elementos:

`pop()` - Remover do Final:

O método `pop()` remove o último elemento de um array e retorna esse elemento.

```
let frutas = ["Maçã", "Banana", "Laranja"];
let ultimaFruta = frutas.pop();
// Agora, frutas é ["Maçã", "Banana"], e ultimaFruta é "Laranja"
```

Extra: *existem métodos diferentes em JS, caso queira pesquisar:*

- *`unshift()` - adicionar no início*
- *`splice()` - adicionar no meio*
- *`shift()` - remover do início*
- *`splice()` - remover no meio (os parâmetros são diferentes do splice para adicionar)*

Lista de exercícios (pratique!):

Ao entender o básico dos arrays, você estará preparado para explorar conceitos mais avançados na jornada da programação. Continue praticando e experimentando para aprimorar suas habilidades!

Nos procure para tirar dúvidas sobre a lista de exercícios.

1) Crie um array chamado numeros que contenha os números de 1 a 10.

2) Acesse o terceiro elemento do array numeros e armazene em uma variável chamada terceiroElemento.

3) Substitua o segundo elemento do array numeros pelo dobro do seu valor atual.

4) Adicione o número 6 ao final do array numeros.

5) Crie um novo array chamado frutas com pelo menos três nomes de frutas.

6) `let frutas = ["banana","maca",1.2]` o que tem de errado nesse array? Por que?

Extras:

7) Utilize um loop para exibir cada elemento do array numeros no console.

8) Calcule e exiba a soma de todos os elementos do array numeros.