

# Strings e Arrays

Labenu\_



# O que vamos ver hoje?

- Strings
- Protótipo de Strings
- Arrays
- Protótipo de Arrays



# Strings

Labenu\_



# Declaração de Strings

- Como vimos anteriormente, Strings são os tipos referentes à **textos**
- Temos 3 maneiras de escrever uma string:
  - Aspas Duplas: "Olá Mundo"
  - Aspas Simples: 'Olá Mundo'
  - Crase (Template String ou Template Literals):  
`Olá Mundo`



# Concatenação de Strings

- Também podemos juntar várias strings para formar uma nova
- Chamamos esse processo de **concatenação** e utilizamos o sinal de **+** para fazê-lo

```
const nome = "Chijo"
```

```
const idade = 27
```

```
const frase = "Meu nome é " + nome + " e tenho " + idade + " anos"
```

É necessário colocar o espaço para separar palavras



# Template Strings

- Não há diferença entre usar aspas simples ou duplas!
- A única diferente é a **Template String**, pois ela nos permite colocar variáveis javascript no meio da string

```
const nome = "Chijo"
```

```
const idade = 27
```

```
const frase = `Meu nome é ${nome} e tenho ${idade} anos`
```

```
// Meu nome é Chijo e tenho 27 anos
```

Para sinalizar que é uma  
variável, usamos **\$** e **{}**





# Exercício 1

Crie um programa que peça ao usuário para inserir o seu nome e sua cor favorita e imprima a mensagem: *"A cor favorita de FULANO é COR"*

Faça o exercício duas vezes, utilizando template strings e concatenação



# Protótipo de Strings

Labenu\_





# Protótipo de Strings

- O javascript nos fornece algumas informações (**propriedades**) e ações (**métodos**) que podemos realizar sobre uma string
- Falaremos de algumas delas na aula, mas se quiser conhecer mais, você pode visitar [esse link](#)



# Propriedade length

- A propriedade **length** nos diz qual é o **tamanho** de uma string, incluindo espaços

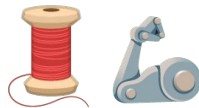
```
const nome = "Caio Teixeira"
```

```
console.log(nome.length) // 13
```

Vamos ver na prática! 



# Método toLowerCase()



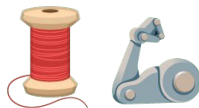
- o método **toLowerCase()** transforma todas as letras da sua string em minúsculas

```
const frase = "OieEeEee!"  
const fraseMinuscula = frase.toLowerCase()  
// fraseMinuscula = oieeeeeee!
```

Vamos ver na prática!



# Método `toUpperCase()`



- o método **`toUpperCase()`** transforma todas as letras da sua string em maiúsculas

```
const frase = "OieEeEee!"  
const fraseMaiuscula = frase.toUpperCase()  
// fraseMaiuscula = OIEEEEEEE!
```

Vamos ver na prática!



# Método trim()

- O método **trim** retira os espaços que existem antes e depois da sua string
- Útil em formulários como por exemplo de login!

```
const email = "  amanda@gmail.com  "
```

```
console.log(email.trim())  
// "amanda@gmail.com"
```

Vamos ver na prática! 



# Método `includes(caracteres)`

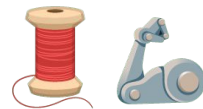
- O método **`includes(caracteres)`** determina se um conjunto de caracteres pode ser encontrado dentro de outra string, retornando **`true`** ou **`false`**

```
const frase = "Hoje comi cenoura"  
frase.includes("cenoura") // true  
frase.includes("batata") // false
```

Vamos ver na prática! 



# Método `replaceAll(chars1, chars2)`



- O método **`replaceAll(chars1, chars2)`** troca todas as ocorrências de um conjunto de caracteres (`chars1`) por alguma outra coisa (`chars2`)

```
const frase = "Hoje comi cenoura, adoro cenoura"  
const novaFrase = frase.replaceAll("cenoura", "batata")  
// novaFrase = Hoje comi batata, adoro batata
```

Vamos ver na prática!





## Exercício 2

Peça para o usuário escrever uma frase e imprima no console a frase alterada, com:

- Todas as letras maiúsculas
- Na língua do i (substituindo todas as vogais por i)

Retorne também o tamanho da frase.



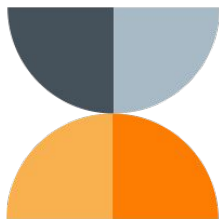


# Pausa para relaxar 🤪

10 min

- **Protótipo de Strings**

- length
- toLowerCase()
- toUpperCase()
- trim()
- includes(caracteres)
- replaceAll(chars1, chars2)



# Arrays

Labenu\_



# O que são arrays?

- Arrays nada mais são do que **listas de elementos**
  - **Ex:** lista de compras, lista de alunos, lista de números da loteria, lista telefônica...
- No javascript, usamos colchetes para agrupar os itens de uma lista:

```
const listaDeCompras = ["batata", "alface", "queijo"]
```

```
const listaDeNumerosMega = [2, 13, 26, 35, 41, 60]
```



# O que são arrays?

- Podemos colocar elementos de **qualquer tipo** que vimos até agora dentro de um array!
  - Números, strings e booleanos
- Também podemos ter elementos de tipos diferentes dentro de um mesmo array

```
const meuArray = ["bananinha", 15, true]
```



# Acessando um elemento

- Em um array, acessamos os elementos através da posição deles na lista!
- Funciona como se fosse uma lista numerada:

## Lista de Compras

1. Batata
2. Alface
3. Queijo



Qual é o **item na posição 2**?

Resposta: Alface



# Acessando um elemento

- Mas no caso dos arrays, a numeração não começa no 1, **mas sim no 0!**
- Para acessar um item, colocamos a sua posição (**índice**) entre colchetes após o nome do array

## Lista de Compras

0. Batata
1. Alface
2. Queijo

```
const listaDeCompras = ["batata", "alface", "queijo"]  
const segundoItem = listaDeCompras[1] // "alface"
```





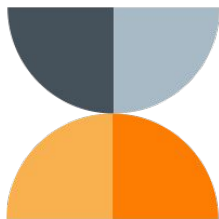
## Exercício 3

- Crie um array com pelo menos 5 raças de cachorro
- Peça para o usuário inserir um número de 1 a 5
- Imprima no console a raça correspondente à posição escolhida pelo usuário



# Pausa para relaxar 🤪

5 min



- **Arrays** são listas que podem conter elementos de qualquer tipo (strings, números, etc)
- Para acessar um elemento de um array, utilizamos a sua posição (ou **índice**)





# Protótipo de Arrays

Labenu\_



# Protótipo de Arrays

- O javascript nos fornece algumas informações (**propriedades**) e ações (**métodos**) que podemos realizar sobre uma lista (array)
- Falaremos de algumas delas na aula, mas se quiser conhecer mais, você pode visitar [esse link](#)



# Propriedade length

- A propriedade **length** nos diz qual é a **quantidade de itens** de um array

```
const pokemons = ["bulbasaur", "squirtle", "charmander"]  
console.log(pokemons.length) // 3
```

Vamos ver na prática! 



# Método **push(elemento)**

- O método **push(elemento)** adiciona um ou mais elementos ao final de um array

```
const numeros = [1, 2, 3]
```

```
numeros.push(4)
```

```
console.log(numeros) // [1, 2, 3, 4]
```

```
numeros.push(5, 6, 7)
```

```
console.log(numeros) // [1, 2, 3, 4, 5, 6, 7]
```

Vamos ver na prática! 



# Método pop()

- O método **pop()** remove o último elemento de um array

```
const meusPeixes = ["palhaço", "mandarim", "esturjão"]
```

```
meusPeixes.pop()
```

```
console.log(meusPeixes) // ["palhaço", "mandarim"]
```

Vamos ver na prática! 



# Método `includes(elemento)`

- O método **`includes(elemento)`** determina se um array contém um determinado elemento, retornando **`true`** ou **`false`**

```
const seriesBoas = ["Breaking Bad", "Brooklyn Nine-nine"]
```

```
seriesBoas.includes("Breaking Bad") // true
```

```
seriesBoas.includes("Game of Thrones") // false
```

Vamos ver na prática! 



# Método splice(i, n)

- O método **splice(i, n)** remove **n** elementos à partir da posição **i** do array

```
const letras = ["A", "B", "C", "D", "E", "F", "G", "H"]  
// índices (i)  0    1    2    3    4    5    6    7
```

```
letras.splice(2, 1)  
// novas letras = ["A", "B", "D", "E", "F", "G", "H"]  
// índices (i)    0    1    2    3    4    5    6
```

```
letras.splice(3, 2) // novas letras = ["A", "B", "D", "G", "H"]
```

Vamos ver na prática! 





## Exercício 4

Para este exercício, comece criando um array com os valores: 1, 2, 3, 4, 5 e 6.

1. Determine o tamanho do array
2. Adicione o número 7
3. Remova os números 4 e 5
4. Determine o novo tamanho do array





# Resumo

Labenu\_



# Resumo

- Temos 3 maneiras de escrever uma string:
  - Aspas Duplas: "Olá Mundo"
  - Aspas Simples: 'Olá Mundo'
  - Crase (Template String): `Olá Mundo`
- Template Strings nos permitem colocar variáveis javascript no meio do texto ⇒ `Olá \${nome}`
- Também podemos concatenar strings ⇒ "Olá " + nome



# Resumo

- **Protótipo de Strings**
  - length
  - toLowerCase()
  - toUpperCase()
  - trim()
  - includes(caracteres)
  - replaceAll(chars1, chars2)



# Resumo

- **Arrays** são listas de elementos que podem ter qualquer tipo
- Agrupamos esses itens usando colchetes `[]`
- Acessamos um item pelo **índice** (ou seja, sua posição na lista)



# Resumo

- **Protótipo de Arrays**
  - length
  - push(elemento)
  - pop()
  - splice(i, n)



Dúvidas? 🧐

Labenu\_





Obrigado(a)!