iscte
INSTITUTO
UNIVERSITÁRIO
DE LISBOA

emprego digital

Módulo 4: Introdução à programação em javascript

Aula 06
Listas e Ciclos

UPskill

Listas

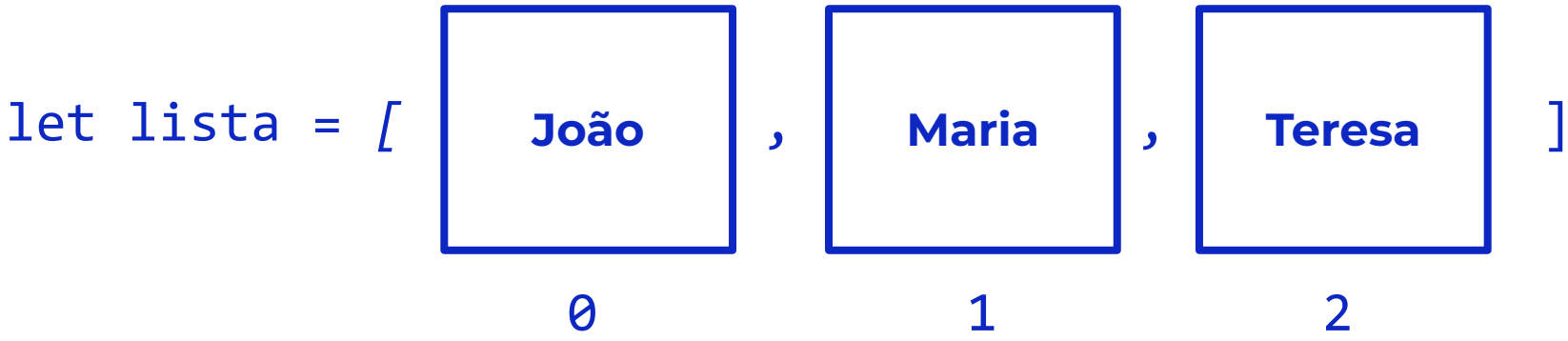
- As listas (Array) permitem-nos guardar uma quantidade indeterminada de informações **em série**.
- Podemos adicionar elementos e aceder aos mesmos através da posição em que eles estão.

```
let lista = ["João", "Maria", "Teresa"];
```

(é o mesmo que)

```
let lista = [
    "João",
    "Maria",
    "Teresa"
];
```

Listas - Acesso por índice



- Quando queremos referenciar um elemento de uma lista devemos utilizar o índice, que é 0 no primeiro elemento.
- Assim, para aceder ao último elemento desta lista de **3 elementos**, temos de aceder ao **índice 2**

Listas - Acesso por índice

```
// Para criar a lista/array
let lista = ["João", "Maria", "Teresa"];

// Para aceder ao primeiro elemento (índice 0)
console.log(lista[0]);
>> "João"
```

Listas - Length

```
// Para criar a lista/array
let lista = ["João", "Maria", "Teresa"];

// Para obter a dimensão da lista
console.log(lista.length);
>> 3
```

Listas - Utilização

```
// Para criar a lista/array
let lista = ["João", "Maria", "Teresa"];


// Para aceder ao último elemento
console.log(lista[lista.length - 1]);
>> lista[3 - 1]
>> lista[2]
>> "Teresa"
```

Listas - Adicionar elemento

```
let lista = ["João", "Maria", "Teresa"];

// Para adicionar um elemento
lista.push("André");

>> ["João", "Maria", "Teresa", "André"];
```

Listas - Encontrar elemento

```
let lista = ["João", "Maria", "Teresa"];

// Para procurar o índice de um elemento
console.log(lista.indexOf("Maria"));
>> 1
```

Listas - Utilização

```
let lista = ["João", "Maria", "Teresa"];

// Para obter os dois primeiros elementos
console.log(lista.slice(0, 2));
>> ["João", "Maria"]
```

Listas - Utilização

```
let lista = ["João", "Maria", "Teresa", "André"];

// Para obter todos os elementos depois do primeiro
console.log(lista.slice(1));
>> ["Maria", "Teresa", "André"]
```

Exercício 2 - Listas

- Com uma lista de nomes e uma lista de sobrenomes, criar uma função que devolva um **novo nome** aleatório.

  Pista: O método **`Math.random()`** devolve um valor aleatório de 0 a 1, nunca chegando a sair 1.

```
function nomeAleatorio(nomes, sobrenomes) {
    ...
}
let nomes = ["João", "Maria", "Teresa"];
let sobrenomes = ["Silva", "Santos", "Carvalho"];

nomeAleatorio(nomes, sobrenomes);

>> "Teresa Silva"
```

Ciclos

- Como vimos anteriormente, as listas podem ter dimensões variadas. Implementar um código flexível que consiga aceder aos itens de uma lista requer a utilização dos ciclos.
- Para a lista `["João", "Maria", "Teresa"]`:

```
console.log(lista[0]);
>> "João"
console.log(lista[1]);
>> "Maria"
console.log(lista[2]);
>> "Teresa"
```

inadequado, específico

```
for (let i = 0; i < lista.length; i++) {
   console.log(lista[i]);
}
>> "João"
>> "Maria"
>> "Teresa"
```

adaptado para listas de qualquer tamanho

Ciclo While

- O ciclo **`while`** permite-nos correr o mesmo bloco de código várias vezes, enquanto uma determinada condição se mantiver.
- Neste exemplo o código dentro do bloco while será corrido **enquanto** o valor na variável “contador” for superior a zero.
- O valor do contador está a ser reduzido cada vez que o código corre, por isso garantimos que o ciclo **termina**.

```
let contador = 5;
while (contador > 0) {
    console.log(contador);
    contador = contador - 1;
}

>> 5
>> 4
>> 3
>> 2
>> 1
```

Ciclo While

- Em alternativa podemos indicar ao ciclo uma **condição que é sempre verdadeira** (“true”, neste caso) e parar o ciclo com uma condição interna e através da instrução “**break**”.

```
let contador = 5;
while (true) {
    console.log(contador);
    contador = contador - 1;
    if (contador === 0)
        break;
}

>> 5
>> 4
>> 3
>> 2
>> 1
```

Ciclo For

- O ciclo `for` possibilita-nos a utilização de uma sintaxe alternativa ao **while**, mais organizada para iterações em listas, por exemplo.

```
for(instrução_1; instrução_2; instrução_3)
```

- instrução_1: executada uma vez antes do bloco de código;
- instrução_2: condição a verificar para executar o bloco de código;
- instrução_3: executada sempre depois do bloco de código

```
for (let contador = 5; contador > 0; contador--) {
   console.log(contador);
}
>> 5        >> 4
>> 3        >> 2
>> 1
```

Ciclo For

- O ciclo for é muito útil para iterar nos elementos de uma lista

```
for (let i = 0; i < lista.length; i++) {
   console.log(lista[i]);
}
>> "João"
>> "Maria"
>> "Teresa"
```

```
for (let i = lista.length - 1; i >= 0; i--) {
   console.log(lista[i]);
}
>> "Teresa"
>> "Maria"
>> "João"
```

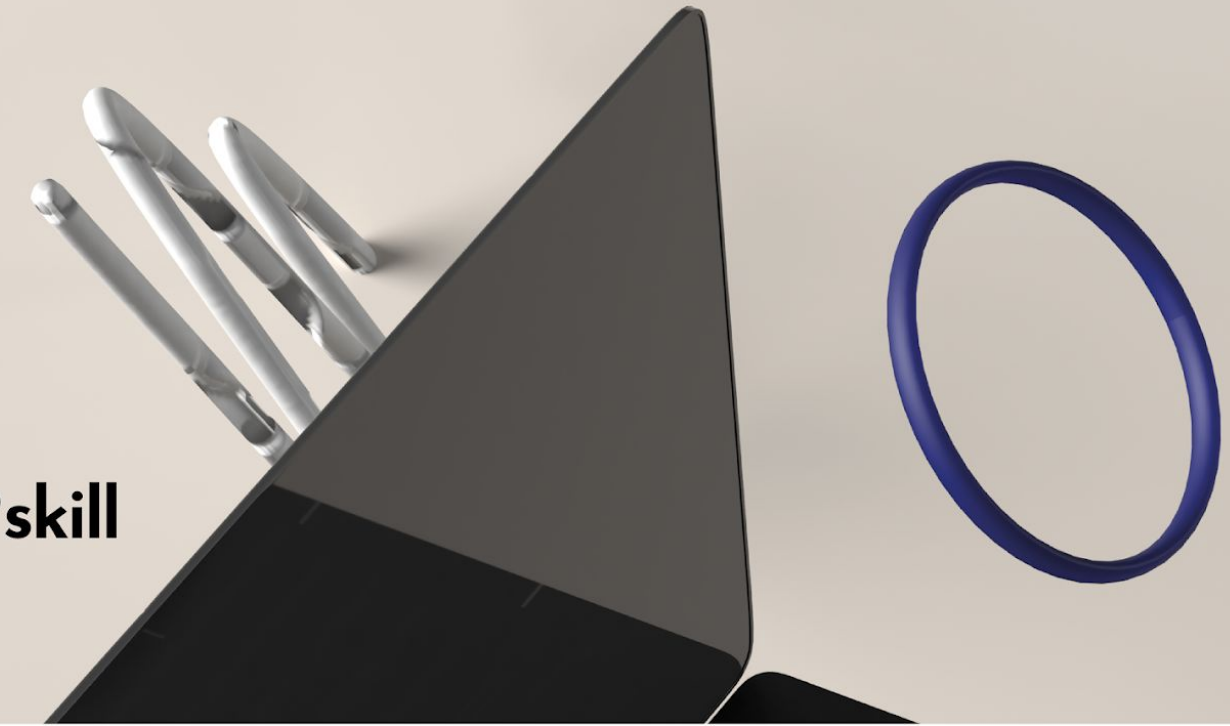
Qual a diferença?

Exercício 1

- Imprimir na consola todos os números pares de 1 a 100, usando as duas estruturas de ciclos (**`for`** e **`while`**).

Bónus: E para imprimir apenas os números primos?

O futuro profissional começa aqui



iscte INSTITUTO UNIVERSITÁRIO DE LISBOA

emprego digital

UPskill