

## > **Ficha Prática Nº1 (JavaScript – Introdução)**



JavaScript é uma linguagem de programação fortemente utilizada no desenvolvimento web, permitindo a criação de páginas interativas e dinâmicas, através da sua capacidade em manipular elementos HTML e CSS em tempo real. Embora o JavaScript se tenha tornado mais conhecido neste contexto, é atualmente usado em diversas outras áreas de desenvolvimento de software.

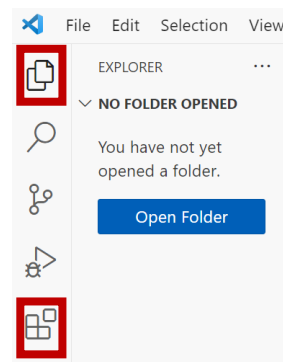
Esta ficha tem como objetivo praticar **conceitos de programação com recurso à linguagem JavaScript**.

Algumas considerações em relação às aulas práticas:

- > Todas as fichas práticas serão disponibilizadas no inforestudante. As resoluções das fichas **não são** para avaliação e, portanto, não é necessário efetuar qualquer entrega ao professor.
- > O editor selecionado para as aulas práticas é o *Visual Studio Code (VSCode)*, no entanto, os alunos podem optar por outro editor de preferência.
- > O *download* do VSCode pode ser efetuado em <https://code.visualstudio.com/>. Caso não pretendam efetuar instalação, podem usar a versão online disponível em <https://vscode.dev/>

## > **Preparação do ambiente**

- Instale o **Node.js** disponível em <https://nodejs.org/>. Este permite a execução de JavaScript, no lado do servidor, em vez de se usar obrigatoriamente um browser.
- Se recorrer ao VSCode, sugere-se a instalação  da extensão **“Code Runner”** para auxiliar a compilação e execução de blocos de código JS.
- Algumas considerações para o executar o JS no VSCode:
  - > F5 > Atalho Ctrl+Alt+N para executar ou F1>Run Code;
  - > Clicar no botão direito no Editor e depois “Run Code”
  - > Clicar no menu “Run” (F5 ou CTRL+F5)
  - > Ao instalar a extensão sugerida, pode executar o código existente no ficheiro (botão direito do rato em cima do ficheiro>”Run Code”  ou então ou selecionar apenas o bloco de código pretendido e com o botão direito do rato selecionar “Run Code”.
- Crie a pasta **ficha1** e abra essa mesma pasta no VSCode.
- Crie o ficheiro **ficha1.js** e grave-o nessa pasta e abra-o no editor.
- A primeira linha de código a especificar (no contexto das aulas práticas) deverá ser **'use strict';**



## Parte I – Introdução

**1>** Faça uma análise aos trechos de código apresentados em cada uma das alíneas e antecipe qual será o resultado, devendo ter em consideração que poderão existir erros de código. Por fim, copie o código para o ficheiro Ficha1.js, execute-o, e confirme o resultado apresentado no *output do Visual Studio Code*. Nota: Deverá sempre usar o código *'use strict'* no início do código durante as aulas práticas de LS. Retire as suas conclusões e esclareça as suas dúvidas.

**a.** Analise o seguinte código e confirme o resultado na consola do VSCode.

```
'use strict';
let a=3;
let b=6;
let c;
console.log("1 =",a+b);
console.log("2 =",a+"c");
console.log("3 =",a+"5");
console.log("4 =",a+"b");
console.log("5 =",a+c);
console.log("6 =",c);
console.log(`Variavel a*b = ${a*b} ( a=${a} e b=${b})`);
```

**b.** Faça o mesmo procedimento para o seguinte bloco de código.

```
'use strict';
let a=3;
let b=6;
let c;
c=a+b;
console.log("1 =",c);
c=a+"---"+b;
console.log("2 =",c);
c="6";
console.log("3 =",a+c);
console.log("4 =",b===c);
console.log("5 =",b==c);
console.log("6 =",b!=c);
console.log("7 =",b!=c);
console.log("8 =",a++);
console.log('9 = ',a);
a=4;
console.log('10 = ',++a);
console.log('11 = ',+a);
```

**c.** Existe algum erro no seguinte trecho de código? Se sim, qual?

```
const disciplina;
disciplina="Linguagens Script";
console.log(disciplina);
```

- d. O que é apresentado na consola? Identifica algum erro?

```
'use strict';
let uc='Linguagens';
uc+='Script';
console.log('Disciplina:'+uc+' - 2 semestre');
```

- e. Faça o mesmo procedimento para o seguinte bloco de código.

```
'use strict';
const nome1='Nuno'
const nome2='Ricardo'
const resultado = `Os nomes são ${nome1} e ${nome2}`
console.log(resultado);
console.log(resultado+' e Filipe');
```

## 2> Resolva os seguintes exercícios com recurso à linguagem JavaScript.

- a. Implemente o código para calcular o maior de **dois** números, devendo apresentar na consola a mensagem “O maior entre ??, ?? = ??”. Declare as variáveis no início da seguinte forma:

```
const num1=5;
const num2=10;
```

Teste o seu código com diferentes valores e se os números forem iguais a mensagem deverá ser antes : “Os números são iguais!”.

- b. Efetue alterações ao código implementado na alínea a), de forma a calcular o maior de três números, devendo apresentar na consola a mensagem “O maior entre ??, ?? e ?? = ??”. Declare as variáveis no início com os valores pretendidos.

Teste o seu código, com diferentes valores.

- c. Implemente o código para efetuar a soma de todos os números entre dois valores. Esses dois valores devem ser inicializados em duas variáveis – **min** e **max**. Por fim, o programa deverá apresentar na consola a soma obtida.

## 3> Considere o seguinte *array*:

```
const numeros = [5,10,-12,2,15,-5,-2,-3]
```

- a. Execute o seguinte código e verifique o resultado.

```
console.log(numeros.length);
```

- b. Implemente o código para obter o número maior existente no *array* e apresente o resultado na consola (recorra a uma estrutura de controlo de fluxo).
- c. Implemente o código necessário para obter a soma de todos os números positivos. O resultado deverá ser apresentado na consola.

## Parte II – Global / Local / Block Scope e Hoisting

- 4>** Analise e antecipe o resultado de cada um dos seguintes trechos de código. **Nota:** poderão existir erros que impeçam a execução completa do código. Confirme se o resultado obtido na consola é o esperado. Esclareça as suas dúvidas.

**Tome especial cuidado ao uso/declaração das variáveis e à sua localização, tendo em mente o funcionamento e características da linguagem JavaScript.**

- a.** Qual o resultado do seguinte trecho de código?

```
'use strict';
let n = 50
if (true) {
  let n = 2
  console.log(n)
}
console.log(n);
```

- b.** Qual o resultado do seguinte trecho de código?

```
'use strict';
let n = 50
if (true) {
  console.log(n);
  n = 2
  console.log(n)
}
console.log(n);
```

- c.** Qual o resultado do seguinte trecho de código?

```
'use strict';
let n = 50
if (true) {
  console.log(n);
  let n = 2
  console.log(n)
}
console.log(n);
```

- d.** Qual o output?

```
'use strict';
let str = 'Linguagens Script';
function fazQualquerCoisa() {
  console.log(str);
}
fazQualquerCoisa();
```

e. Qual o output?

```
'use strict';
let str = 'Linguagens';
function fazQualquerCoisa() {
    str = 'Script';
}
console.log(str);
fazQualquerCoisa();
console.log(str);
```

f. Veja a diferença com ou sem o recurso à instrução 'use strict';

```
'use strict';
function fazQualquerCoisa() {
    str = 'Script';
}
fazQualquerCoisa();
console.log(str);
```

g. Qual o output?

```
'use strict';
function fazQualquerCoisa() {
    let str = 'Script';
}
fazQualquerCoisa();
console.log(str);
```

h. Qual o output?

```
'use strict';
let str = 'Linguagens';
function fazQualquerCoisa() {
    let str2 = ' Script';
    console.log(str+str2);
}
fazQualquerCoisa();
console.log(str+str2);
```

i. Qual o output na consola?

```
var str = 'Linguagens';
function fazQualquerCoisa() {
    var str2 = ' Script';
    if (str==='Linguagens') {
        var dim='ok';
        console.log("->" +dim);
    }
    console.log(str+str2+"- "+dim);
}
fazQualquerCoisa();
console.log(str+str2);
```

**j.** Analise e verifique qual o objetivo do seguinte bloco de código.

```
'use strict';
let str = 'Linguagens';
function fazQualquerCoisa() {
    let str2 = ' Script';

    if (str.length > str2.length) {
        let dim="Primeira é maior!"
        console.log(dim);
    }
    else if (str.length === str2.length) {
        let dim="São iguais!"
        console.log(dim);
    }
    else {
        let dim="Segunda é maior!"
        console.log(dim);
    }

    console.log(str+str2+"-"+dim);
}
fazQualquerCoisa();
```

**k.** Qual o resultado do seguinte trecho de código?

```
function mensagem() {
    let nome='José';
    console.log(`Olá ${nome}`);
}
mensagem();
mensagem('Maria');
mensagem('Maria','Jose','Vieira');
```

**l.** Qual o resultado do seguinte trecho de código?

```
mensagem();
function mensagem() {
    let nome='José';
    console.log(`Olá ${nome}`);
}
```

**m.** Qual o resultado do seguinte trecho de código?

```
function mensagem(nome='!') {
    console.log(`Olá ${nome}`);
}
mensagem();
mensagem('Maria');
mensagem('Jose');
mensagem('Cristiana','Areias');
```

## Parte III – Resolução de exercícios com funções

5> Resolva os seguintes exercícios, recorrendo a função, sempre que considerar o mais correto.

a. Implemente a função **compara** que verifique se dois números são iguais. Deve retornar **true** em caso afirmativo e **false** caso contrário. Implemente a função com e sem recurso ao *operador ternário*.

b. Implemente a função **parOuImpar** que, recebendo como parâmetro um numero, escreva na consola se o número é par ou impar. Use o operador %

```
parOuImpar(5); //Deverá escrever 'O número é impar!'
parOuImpar(4); //Deverá escrever 'O número é par!'
```

c. Implemente a função **obtemQuadrado** que devolva o quadrado de um valor.

```
console.log(obtemQuadrado(2)) //Será apresentado 4 na consola
console.log(obtemQuadrado(9)) //81
console.log(obtemQuadrado(10)) //100
```

d. Implemente a função **areaRetangulo** que devolva a área e um retângulo. Se for passado apenas um valor, então, os dois valores devem ser considerados iguais. Resolva sem recorrer a qualquer if.

```
console.log(areaRetangulo(5,10)) // 50
console.log(areaRetangulo(10,20)) // 200
console.log(areaRetangulo(5)) // 25
```

e. Implemente a função **contaVogais** que receba por parâmetro uma *string* e devolva o número de vogais existentes nessa string. Pode recorrer aos métodos **charAt()**, **toLowerCase()**, métodos estes que permitem verificar o que se encontra numa determinada posição do carácter e converter tudo para minúsculas, respetivamente.

```
console.log(contaVogais("Ola")) //2
console.log(contaVogais("Linguagens Script")) //5
```

f. Considere o seguinte array:

```
const palavras=['angular','bootstrap','javascript','vue','svelte','react'];
```

- > Implemente a função **imprimeArray** que recebe por parâmetro um *array* de *strings* e imprima os elementos desse *array* na consola.
- > Implemente a função **insertBegin**, que recebendo por parâmetro um **array** e uma **palavra**, insira essa palavra no início do array. A versão a implementar não deve recorrer aos métodos *built-in* de inserção de elementos em arrays do JavaScript como o **push**. A função a implementar deverá ter o mesmo comportamento que **palavras.push('ember')**;

## Parte IV – Jogo do Galo

6> Implemente as seguintes alíneas em JavaScript de forma a simular o tradicional **jogo do galo**.

Para isso, considere as seguintes variáveis:

```
let jogadorAtual = "0";
let tabuleiro = [
    [ " ", " ", " " ],
    [ " ", " ", " " ],
    [ " ", " ", " " ],
];
```

a. Implemente a função **imprimeTabuleiro** cujo objectivo é imprimir o tabuleiro na consola. Deverá, como exemplo, apresentar o seguinte aspeto(neste caso sem qualquer jogada):

```

| |
| |
| |

```

b. Implemente a função **jogada** que recebe por parâmetro o valor da **linha** e **coluna**, e altere o tabuleiro, com o valor do jogador atual, nessa coordenada.

- > A jogada só poderá ser aceite se, na posição em questão, ainda não existir qualquer jogada;
- > Quando uma jogada é aceite, será necessário alterar o jogadorAtual;
- > Deverão ser apresentadas as mensagens “*Posição ocupada. Jogue novamente!*” e “*Próximo Jogador=...*” no instante correto.

c. Implemente a função **verificaVencedor**.

- > A função deverá retornar o código do vencedor ou 0 se não houver ainda um vencedor.
- > Altere a função **jogada** de forma a verificar o vencedor e apresente a devida mensagem caso haja um vencedor.

d. Implemente a função **verificaFimJogo**.

- > A função deverá retornar **true** em caso de fim de jogo, caso contrário, **false**.
- > Altere a função **jogada** de forma a verificar o fim de jogo caso não seja detetado nenhum vencedor após uma jogada. Caso seja fim de jogo, este deverá apresentar a mensagem “*Fim de Jogo! Jogo empatado!*”.

e. Implemente alterações necessárias de forma que, quando o jogo já tenha terminado, apresente a mensagem “*Jogada não efetuada! O jogo já terminou sendo o vencedor => ...*” na tentativa de nova jogada.

f. Teste o seu código, invocando as seguintes jogadas, o qual deverá terminar o jogo e apresentar o vencedor.



## &gt; Situação com Vencedor

```

jogada(1, 1);
jogada(0, 0);
jogada(0, 0);
jogada(0, 1);
jogada(2, 0);
jogada(2, 1);
jogada(0, 2);

```

	X	

Proximo Jogador -> 0

0		
	X	

Proximo Jogador -> X  
Posição ocupada.Jogue novamente!

0	X	
	X	

Proximo Jogador -> 0

0	X	
	X	
0		

Proximo Jogador -> X

0	X	
	X	
0	X	

Vencedor = X  
Jogada não efetuada! O jogo já terminou sendo o vencedor => X

## &gt; Situação com Empate

```

jogada(1, 1);
jogada(0, 0);
jogada(0, 0);
jogada(0, 1);
jogada(2, 0);
jogada(0, 2);
jogada(0, 1);
jogada(1, 2);
jogada(1, 0);
jogada(2, 1);
jogada(2, 2);

```

	X	

Proximo Jogador -> 0

0		
	X	

Proximo Jogador -> X  
Posição ocupada.Jogue novamente!

0	X	
	X	

Proximo Jogador -> 0

0	X	
	X	
0		

Proximo Jogador -> X

0	X	X
	X	
0		

Proximo Jogador -> 0  
Posição ocupada.Jogue novamente!

0	X	X
	X	O
0		

Proximo Jogador -> X

0	X	X
X	X	O
0		

Proximo Jogador -> 0

0	X	X
X	X	O
0	O	

Proximo Jogador -> X

0	X	X
X	X	O
0	O	X

Fim de Jogo! Jogo empatado!