

Documentação de Apoio

- Slides disponíveis na plataforma moodle da Unidade Curricular
- <https://developer.mozilla.org/en-US/docs/Web/JavaScript> - JavaScript

1 Exercícios JavaScript

1.1 Introdução

Deverá criar o primeiro programa que escreva “*Hello World*” numa consola tanto no runtime JavaScript NodeJS ou no runtime presente no seu browser de eleição.

Deverá para o efeito criar um ficheiro com o nome *main.js* onde deve colocar às instruções em JavaScript. Posteriormente de executar o ficheiro de criado no runtime NodeJS utilizando o comando:

- `node main.js`

Para correr o ficheiro *main.js* utilizando o runtime Javascript do browser, deve criar o ficheiro *index.html*, ver figura 1, no mesmo diretório e abrir o ficheiro criado com um browser moderno à sua escolha. O ficheiro *main.js* é importado e executado pela tag script dentro do ficheiro html.

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>PAW Aula 1</title>
</head>

<body>
  <script src="main.js"></script>
</body>
</html>

```

Figura 1 - Template para execução de JavaScript no Browser (*index.html*)

1.2 Operações em JavaScript

Para cada uma das expressões presentes na tabela 1, calcule mentalmente o resultado que será obtido na consola de JavaScript. Anote as suas repostas e justificações.

Tabela 1 - Expressões em JavaScript

Expressão	Resultado Esperado	Justificação
<code>1=="1"</code>		
<code>1===1</code>		
<code>"a"==97</code>		
<code>1 / 3</code>		
<code>typeof undeclaredVariable</code>		
<code>var x = 12</code> <code>typeof x</code>		

[] == []		
[1] == [1]		
var a = [1, 2] var b = a a == b		
var a = [1, 2] var b = a a === b		
1 != 5		
!(1 != 5)		
11 % 2		
11 / 2		

Num terminal, execute o comando node para abrir a consola interativa. Copie cada expressão representada nas linhas da tabela e compare os resultados obtidos com as suas respostas. Verifique as diferenças e encontre a justificação para cada uma delas.

1.3 Funções em JavaScript

1.3.1 Operações com funções

Verifique o excerto apresentado na figura 2. Identifica algum problema de programação presente no ficheiro? É esperado apresentar algum erro de programação na consola de JavaScript? Anote as suas observações.

```
function hello(a){
    console.log('Hello')
}
hello()
hello("world")
hello(true)
hello(2)

function demo(a,b,c){
    console.log(a)
}
demo(2);
demo(2,3);
demo(2,3,4);

function new_demo(a,b=3){
    console.log(a + " " + b)
}
new_demo(1)
new_demo(1,4)

function odd_demo(a,b){
    if (b==undefined){
        console.log(a)
```

```
    }else{  
        console.log(a + " " +b)  
    }  
}  
odd_demo(1)  
odd_demo("hello", 3)
```

Figura 2 - Excerto de código sobre funções em JavaScript

Copie o excerto na figura 2 para um ficheiro JavaScript e execute-o. Verificou algum erro na execução do ficheiro? Estava de acordo com as suas expectativas?

```
function soma(a,b){  
    return a+b  
}  
console.log(soma(1,2))  
console.log(soma(1))  
console.log(soma())  
  
function concat(a,b){  
    return a+b  
}  
console.log(concat("Hello ", "World"))  
console.log(concat(""))  
console.log(concat())  
  
function odd_demo2(a,b){  
    if (b==undefined){  
        console.log(a)  
    }else{  
        console.log(a + " " +b)  
    }  
}  
odd_demo2(1)  
odd_demo2("hello", 3)  
odd_demo2()  
  
function element(index){  
    var arr =[1,2,3]  
    return arr[index]  
}  
  
console.log(element(-1));  
  
function sample(c){  
    console.log(unknown)  
}
```

```
sample()
```

Figura 3 - Excerto de código sobre funções em JavaScript

Copie o excerto na figura 3 para um ficheiro JavaScript e execute-o. Quais as diferenças em relação ao excerto na figura 2. Corrija os erros detetados e volte a correr o script do excerto na figura 3. Corrija todas as expressões undefined e NaN. Repare que são erros não detetados pelo runtime JavaScript e que geram soluções estranhas.

1.3.2 Funções

Crie um ficheiro JavaScript à sua escolha, defina funções que resolvam cada um dos problemas enunciados. Teste as funções criadas com argumentos a sua escolha e imprima os resultados na consola.

Assim, utilizando criando um ficheiro JS, escreva funções que:

- Calcule a multiplicação dois números passados como argumento e retorne o valor;
- Devolva a maior string de um array de strings passado como argumento;
- Dada uma string substitua a primeira letra pela letra maiúscula equivalente e devolva a string. Se já estiver em maiúsculas deverá retornar a mesma string.
- Devolva o número que mais vezes apareça num array de inteiros passado como argumento e retorne o valor;
- Valide que uma string passada como argumento representa um email válido. A função deve retornar um booleano (*true/false*).
- Formate um número recebido como argumento para apresentar exatamente 9 dígitos e retorne o valor com formato string. Se o número tiver menos de 9 dígitos devem ser adicionados zeros à esquerda. Caso o número tenha mais de 9 dígitos deve lançar uma exceção.
- Calcular se o um número passado como argumento é um número primo. A função deve retornar um booleano (*true/false*).
- Converta um valor em cêntimos de euro nas respetivas moedas e retorne as moedas sobre a forma de array. Exemplo: 46 retorna [20,20,5,1], um conjunto de moedas que perfazem o valor pretendido.
- Crie uma função que verifique se uma palavra passada como argumento é um palíndromo e retorne um booleano (*true/false*).
- Crie uma função que retorne o número de dias do mês de uma data passada como parâmetro. (Nota: tenha em consideração o Objecto Date e o facto de existirem anos bissextos.)

1.4 Objetos

Criando ficheiros JavaScript à sua escolha, defina scripts que permitam dar resposta aos exercícios. Teste os resultados da execução do script com argumentos à sua escolha na consola.

1.4.1 Ciclos e objetos

Analise os excertos de código com diferentes ciclos sobre objetos em JavaScript nas figuras 4, 5 e 6. Qual o resultado esperado de cada execução?

```
// for each com objeto
const myObject = {a: 1, b: 2, c: 3};
for (const property in myObject) {
  console.log(property);
}

// for each com array
const myArray = [1,2,3];
for (const property in myArray) {
  console.log(property);
}
```

Figura 4 – Excerto 1 ciclo simples for .. in ...

```
//for each com objetos
const object = {a: 1, b: 2, c: 3};
for (const property in object) {
  console.log("Propriedade: "+property+" Objeto:"+object[property]);
}
//for each com array
const array = [1,2,3];
for (const property in array) {
  console.log(" "+property+" "+array[property+""]);
}
}
```

Figura 5 – Excerto 2 ciclo simples for .. in ...


```
// for of com arrays
//const object1 = {a: 1, b: 2, c: 3};
//for (const element of object1) {
//  console.log(element);
//}

// for of com arrays
const array1 = ['a', 'b', 'c'];
for (const element of array1) {
  console.log(element);
}
```

Figura 6 - Excerto 3 ciclo simples for ... of ...

Copie os excertos de código para um ficheiro e execute-o. Verifica os resultados esperados?
Quais os resultados inesperados nos exemplos fornecidos?

Porque razão o exemplo for .. of se encontra comentado? O que acontece se tentar executar as linhas comentadas?

 <div data-bbox="435 147 528 224"> ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO </div>	Programação em Ambiente Web Docentes – FAS, JRMR, NJR Ficha Prática 1
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------

1.4.2 Lista de contactos

Defina uma função para criar um objeto contacto a partir dos seus argumentos. Cada contacto deve conter pelo menos um nome e um número de telefone, sendo outros parâmetros como email, idade e alcunha opcionais. Na função criar objeto deve garantir que o nome começa por uma letra maiúscula e que o número de telefone tem exatamente 9 dígitos, caso contrário deve lançar uma exceção.

Adicione uma variável global que defina um array vazio para armazenar os contactos a criar. Adicione funções *add*, *remove* e *update* para adicionar, remover ou atualizar contatos na variável global.

Crie uma nova função que seja capaz de filtrar todos os contactos com idade superior a um número passado como parâmetro.

1.4.3 Batalha Naval

Crie um objeto em JavaScript para gerir um jogo de batalha naval. O objeto deve conter um título, um array representando o mapa do jogo, um contador de jogadas, um método para marcar a posição de um navio e um método para verificar que um tiro acerta num navio.

Teste o seu objeto definido um mapa, adicionando navios e simulando tiros que acertam e não acertam em navios.

1.4.4 Prototype

Atendendo ao exemplo fornecido na figura 7, para que serve a palavra prototype. Copie o código presente na figura e observe os resultados obtidos.

```
function Person(first, age) {
    this.firstName = first;
    this.age = age;
}

var person1 = new Person("Pedro", 12)
var person2 = new Person("Paulo", 15)

console.log(person1)
console.log(person2)

Person.prototype.nationality = "PT"

console.log(person1)
console.log(person2)
console.log(person1.nationality)
console.log(person2.nationality)

Person.prototype.evaluateAge = function(){
    if(this.age<=12){
        return "criança";
    } else if(this.age<=21){
        return "adolescente";
    } else {
        return "idoso";
    }
}
```

```
    }  
}  
  
console.log(person1.evaluateAge())  
console.log(person2.evaluateAge())
```

Figura 7 - Demonstração uso do prototype

Verifique que com a palavra `prototype` podemos injetar novos atributos e métodos em objetos criados a partir de um construtor. Com a versão do JavaScript ES6, temos mecanismos alternativos para usar mecanismos clássicos de POO em JavaScript.

1.5 Classes

Tirando partido de ES6, resolva os seguintes exercícios:

- Redefina os exercícios 1.4.2 e 1.4.3 usando classes em JavaScript da sintaxe de ES6.
- Use o conhecimento de POO e reescreva o exemplo fornecido na figura 7 com a criação de classes e herança para estender as propriedades de uma classe.

2 Cursos Online para Introdução a JavaScript

2.1 Realize o curso online disponível na plataforma: CodeAcademy: Introduction To JavaScript (<https://www.codecademy.com/learn/introduction-to-javascript>). Realize os seguintes tópicos:

- Introduction To JavaScript
- Variables
- Control Flow
- Functions
- Scope
- Arrays
- Loop

2.2 Realize o conjunto de exercícios disponíveis na plataforma developers da Mozilla (<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>) como forma de se familiarizar com programação JavaScript com os seguintes tópicos:

- Grammar and types
- Control flow and error handling
- Loops and iteration
- Functions
- Expressions and operators
- Numbers and dates
- Text formatting
- Indexed collections
- Working with objects
- Details of the object model

2.3 Realize o conjunto de exercícios disponíveis na plataforma w3schools (https://www.w3schools.com/js/exercise_js.asp?filename=exercise_js_variables1) como forma de se familiarizar com programação javascript com os seguintes tópicos:

- Variables
- Operators
- Data Types
- Functions
- Objects
- Events
- String
- Arrays
- Dates
- Math
- Comparisons
- Conditions
- Switch
- Loops