# REDES DE COMUNICAÇÃO


11º - ANO

Professor: Rafael Henriques

E-mail: prof@rafaelhenriques.com

# JavaScript - Introdução

## Porquê aprender JavaScript

O JavaScript é uma das três linguagens que um programador web tem de aprender:

1. HTML que define o conteúdo das páginas WEB;
2. CSS para especificar o Layout das páginas WEB;
3. JavaScript para programar o comportamento das páginas WEB

O JavaScript não é usado apenas para elaborar páginas WEB. Muitos programas de computador cliente e de servidor usam JavaScript, exemplo disso é o Node.js. Muitos motores de base de dados usam o JavaScript como linguagem de Programação tal como o MongoDB e CouchDB

# JavaScript - Introdução

## Versões

| Ver | Official Name | Description |
|---|---|---|
| 1 | ECMAScript 1 (1997) | First Edition. |
| 2 | ECMAScript 2 (1998) | Editorial changes only. |
| 3 | ECMAScript 3 (1999) | Added Regular Expressions.<br>Added try/catch. |
| 5 | ECMAScript 5 (2009) | Added "strict mode".<br>Added JSON support.<br>Added String.trim().<br>Added Array.isArray().<br>Added Array Iteration Methods. |

# JavaScript - Introdução

## Versões

| Ver | Official Name | Description |
|---|---|---|
| … | … | … |
| 9 | ***ECMAScript** 2018 | Added rest / spread properties.<br>Added Asynchronous iteration.<br>Added Promise.finally().<br>Additions to RegExp. |

***ECMAScript **é o nome oficial da linguagem JavaScript****

**Nota:** O Java e o JavaScript são linguagens completamente diferentes, tanto no conceito e no desenho

# JavaScript - Introdução

## O que faz?

- **O JavaScript pode mudar o conteúdo de uma página**

Um dos muitos métodos do HTML do JavaScript é o **getElementById()**

Exemplo:

```
document.getElementById("demo").innerHTML = “Olá ao JavaScript";
```

# JavaScript - Introdução

## O que faz?

- **O JavaScript pode mudar o conteúdo de uma página**

```
document.getElementById("demo").innerHTML = “Olá ao JavaScript";
```

```
<!DOCTYPE html>
<html>
<body>
        <h2>O que o JavaScript pode fazer?</h2>
        <p id="demo">JavaScript pode mudar o conteúdo HTML.</p>
        <button type="button" onclick='document.getElementById("demo").innerHTML =
“Olá eu sou o JavaScript!“’>Clicar aqui!</button>

</body>
</html>
```

# JavaScript - Introdução

## O que faz?

- **O JavaScript pode mudar o conteúdo de uma página**

```
document.getElementById("demo").innerHTML = “Olá eu sou o JavaScript";
```

No exemplo dado usa o método **getElementById** para encontrar um elemento HTML com o id = “demo” e substitui o seu elemento de conteúdo (**innerHTML**) por “Olá eu sou o JavaScript";

# JavaScript - Introdução

## O que faz?

- **O JavaScript pode mudar o valor dos atributos**

```
document.getElementById('myImage').src='imagem2.jpg'
```

No exemplo dado usa o método **getElementById** para encontrar um elemento HTML com o id = "`myImage`" e substitui o seu elemento de fonte (**src**) pela imagem cujo o ficheiro tem o seguinte nome: `imagem2.jpg;`

# JavaScript - Introdução

## O que faz?

- **O JavaScript pode mudar o valor dos atributos**

```
<!DOCTYPE html>

<html>
<body>

      <h2>O que o JavaScript consegue fazer?</h2>
     <p>JavaScript consegue alterar o valor dos atributos.</p>
     <p>Neste caso altera o valor do atributo src (source) do elemento img.</p>

    <button onclick="document.getElementById('myImage').src='imagem1.jpg'  ">Médico</button>
    <img id="myImage" src= "imagem1.jpg"  style="width:100px">
    <button onclick="document.getElementById('myImage').src='imagem2.jpg' ">Segurança</button>

</body>
</html>
```

# JavaScript - Introdução

## O que faz?

- **O JavaScript pode mudar os Estilos CSS do HTML**

```
document.getElementById("demo").style.fontSize = "35px";
```

No exemplo dado usa o método **getElementById** para encontrar um elemento HTML com o id = “demo” e substitui o estilo tamanho da fonte (style.fontsize) para 35 px;

# JavaScript - Introdução

## O que faz?

- **O JavaScript pode mudar os Estilos CSS do HTML**

```
<!DOCTYPE html>
<html>
<body>

<h2>O que pode o JavaScript fazer?</h2>

<p id="demo">JavaScript  pode mudar o estilo de um elemento HTML.</p>

<button type="button" onclick="document.getElementById('demo').style.fontSize='35px'">Click Me!</button>


</body>
</html>
```

# JavaScript - Introdução

## O que faz?

- **O JavaScript pode mudar esconder e mostrar elementos**

```
document.getElementById("demo").style.display = “none";
```

No exemplo dado usa o método **getElementById** para esconder um elemento HTML com o id = “demo”;

```
document.getElementById("demo").style.display = “block";
```

No exemplo dado usa o método **getElementById** para mostrar um elemento HTML com o id = “demo”;

# JavaScript - Introdução

## O que faz?

- **O JavaScript pode mudar esconder e mostrar elementos**

```
<!DOCTYPE html>
<html>
<body>

<h2>What Can JavaScript Do?</h2>

<p id="demo">JavaScript can change the style of an HTML element.</p>

 <button onclick="document.getElementById('demo').style.display='none' ">Esconder</button>
 <button onclick="document.getElementById('demo').style.display='block' ">Mostrar</button>

</body>
</html>
```

# JavaScript - Introdução

## Onde escrever?

- **Como se pode escrever código JavaScript em numa página HTML**

- Com recurso à Tag <script>

```
<script>
document.getElementById("demo").innerHTML = “Olá  JavaScript";
</script>
```

Em exemplos antigos de JavaScript podem usar o atributo “type”:

<script type="text/javascript">

O atributo type já não é necessário por o JavaScript é a linguagem de scripting por defeito do html

# JavaScript - Introdução

## Onde escrever?

- **Como se pode escrever código JavaScript em numa página HTML**

- Nos eventos

```
<button onclick="document.getElementById('demo').style.display='block'
">Mostrar</button>
```

# JavaScript - Introdução

## Onde escrever?

- **Funções JavaScript e Eventos**

As Funções JavaScript são blocos de código em JavaScript que podem ser executados quando são chamados ou invocados por outros elementos de código.

Os eventos são reações a ações do browser ou dos utilizadores, exemplos desses eventos são:

- Quando uma página HTML acaba de carregar;
- Quando um campo do input altera de valores;
- Quando um botão é pressionado/clicado;

Quando um evento ocorre, nós programadores podemos querer alterar o comportamento de alguns elementos da página.

# JavaScript - Introdução

## Onde escrever?

- **Funções JavaScript e Eventos**

O JavaScript permite executar código quando um evento ocorre e manipular e aos atributos dos elementos HTML com código adicionado a esses mesmos atributos;

Exemplo:

Uma função pode ser chamada a ser executada quando o utilizador pressiona um botão.

```
<button onclick="aminhafuncao();">Função</button>
```

# JavaScript - Introdução

## Onde escrever?

- **JavaScript no <head> ou <body>**

Podemos colocar um numero indeterminado de scripts num documento HTML e podem ser colocados tanto na secção <head> de um página HTML como no <body> ou em ambos.

**Na <head>**

```
<!DOCTYPE html>
<html>
<head>
<script>
  function minhafuncao(){
                document.getElementById("demo").innerHTML = "ESTOU NA HEAD";
 }
</script>
<head>
<body>
…
```

# JavaScript - Introdução

## Onde escrever?

- **JavaScript no <head> ou <body>**

Podemos colocar um numero indeterminado de scripts num documento HTML e podem ser colocados tanto na secção <head> de um página HTML como no <body> ou em ambos.

**No <body>**

```
…
<body>
        <h2>JavaScript no Body</h2>
        <p id="demo">Onde Estou.</p>
        <button type="button" onclick="minhaFuncao()">Try it</button>
        <script>
                function minhaFuncao() {
                        document.getElementById("demo").innerHTML = "Eu estou no Body.";
                }
        </script>
</body>
…
```

# JavaScript - Introdução

## Onde escrever?

- **JavaScript no num ficheiro externo**

Scripts externas são práticas quando usamos o mesmo código em várias páginas.

Os ficheiros JavaScript têm a extensão **.js**

Para usar uma script externa, devemos colocar no atributo src (source) na tag <script> o nome do ficheiro, caso esteja em outra pasta devemos de indicar também o caminho.

```
<script src="minhaScript.js"></script>
```

**Ficheiro externo:** minhaScript.js

```
function myFunction() {
   document.getElementById("demo").innerHTML = “Estou num ficheiro externo ;)";
}
```

# JavaScript - Introdução

## Onde escrever?

- **JavaScript no num ficheiro externo**

**Vantagens:**

- Separa o código do HTML;
- Torna o HTML e o JavaScript mais fácil de ler e manter;
- Ficheiros JavaScript em Cache aumentam a velocidade de carregamento das páginas;
- Posso adicionar várias páginas externas a uma página usando várias tags <script>

```
<script src="minhaScript1.js"></script>
<script src="minhaScript2.js"></script>
<script src="minhaScript3.js"></script>
```

# JavaScript - Introdução

## Onde escrever?

- **JavaScript no num ficheiro externo**

**Vantagens:**

- Scripts externas podem ser referenciadas através de um URL completo, do próprio ou de outro site

```
<script src="https://www.w3schools.com/js/minhaScript1.js"></script>
```

- Ou através de um URL relativo para a corrente página Web

```
<script src=“pastajs/minhaScript1.js"></script>
```

# JavaScript - Introdução

## Onde escrever?

- **Alguns cuidados**

```
<!DOCTYPE html>
<html>
<head>
	<script>
		document.getElementById("demo").innerHTML = "ESTOU AQUI";
	</script>
<head>
<body>
…
```

O código JavaScript antes na <head> sem estar dentro de uma função, não provoca erros mas é inútil, pois não se consegue chamar para ser executado.

# JavaScript - Introdução

## Onde escrever?

- **Alguns cuidados**

```
…
	<body>
	<h2>JavaScript no Body</h2>

	<script>
		document.getElementById("demo").innerHTML = "ESTOU AQUI";
	</script>
	<p id="demo">não estou no sitio certo.</p>
…
```

O código JavaScript colocado antes dos elementos que pretende afetar também não provoca erros mas não funciona ou simplesmente não se comporta como o programador pretendia. No exemplo, após a página carregar, o programador pretendia que surgisse “ESTOU AQUI”, no entanto surge “não estou no sitio certo”, solução trocar a ordem dos elementos.

# JavaScript - Introdução

## Visualização (Display)

- **Os resultados do JavaScript podem ser mostrados de diferentes formas**

  - Escrito num elemento HTML, com o uso do **innerHTML;**
  - Escrito no output do HTML, com recurso ao **document.write();**
  - Escrito numa caixa de alerta, com uso do **window.alert();**
  - Escrito na consola do browser, com o uso **console.log();**

**innerHTML**

```
<script> document.getElementById("demo").innerHTML = 5 + 6; </script>
```

**innerHTML**

```
<script>  document.write(5 + 6); </script>
```

**Nota: usar document.write depois do documento HTML carregar, ele vai apagar todo o html existente no browser,**

# JavaScript - Introdução

## Visualização (Display)

- **Os resultados do JavaScript podem ser mostrados de diferentes formas**

**document.write() – exemplo**

```
<!DOCTYPE html>
<html>
<body>

        <h2>Exemplo</h2>
        <p>Ao carregar no botão, todo o html existente será apagado e surgirá o valor 11.</p>

        <button type="button" onclick="document.write(5 + 6)">vamos verificar?</button>
</body>
</html>
```

Deve de ser usado apenas em testes

# JavaScript - Introdução

## Visualização (Display)

- **Os resultados do JavaScript podem ser mostrados de diferentes formas**

**window.alert() - exemplo**

```
<!DOCTYPE html>
<html>
<body>

        <h2>Exemplo</h2>
        <p>Ao carregar no botão, surge uma mensagem de alerta.</p>

<button type="button" onclick="window.alert('Olá o JavaScript é engraçado!')">vamos verificar?</button>
</body>
</html>
```

Deve de ser usado para fazer chamadas de atenção ao utilizador.

# JavaScript - Introdução

## Visualização (Display)

- **Os resultados do JavaScript podem ser mostrados de diferentes formas**

**console.log() - exemplo**

```
<!DOCTYPE html>
<html>
<body>

        <script> console.log(5 + 6); </script>
</body>
</html>
```

Deve de ser usado para efetuar debugger na consola do Browser.
Clicar com o botão direito na página que queremos inspecionar e clicar na opção “Inspecionar” depois selecionar a aba “Console”

# JavaScript – Statements*

## Programas

- **Um programa de computador é uma lista de instruções para serem executadas por um computador.**
- **Numa Linguagem de programação as instruções de um programa são chamadas de statements/afirmações**
- **Um programa em JavaScript é uma lista de statements/afirmações programadas**

Exemplo

```
var x, y, z;     // Statement 1
x = 5;           // Statement 2
y = 6;           // Statement 3
z = x + y;       // Statement 4
```

***Statements – Em português Afirmação**

# JavaScript – Statements

## Composição

As Statements em JavaScript  são compostas por

- Valores;
- Operadores;
- Expressões;
- Keywords
- E comentários

**Exemplo:**

```
document.getElementById("demo").innerHTML = "Hello Dolly.";
```

# JavaScript – Statements

## Separação entre afirmações

O ponto e vírgula separa os statements, para isso basta adicionar  o caracter ponto e vírgula no fim da afirmação.

**Exemplo:**

```
var a, b, c; // Declara 3 variáveis
a = 5;       // Atribui o valor 5 à variável a
b = 6;       // Atribui o valor 6 à variável b
c = a + b;   // Atribui o resultado da soma de a com b à variável c
```

São permitidos vários staments na mesma linha desde que estejam separados por ponto e vírgula

```
a = 5; b = 6; c = a + b;
```

# JavaScript – Statements

## Espaços em branco; cumprimento e quebra de linhas

O JavaScript ignora os espaços em branco, as seguintes staments são equivalentes:

```
var person  =  “Manel";
var person=“Manel";
```

Quando as linhas de código são muito cumpridas pode-se colocar uma quebra de linha e o melhor sitio para a colocar é depois do operador:

```
document.getElementById("demo").innerHTML =
“Olá Manel!";
```

# JavaScript – Statements

## Blocos de código

As Statements podem ser agrupadas em **blocos de código** dentro de **chavetas**

```
function minhaFuncao() {
    document.getElementById("demo1").innerHTML = “Olá Manel!";
    document.getElementById("demo2").innerHTML = “Quem és tu?";
}
```

**O propósito dos blocos de código é serem executados juntos**

# JavaScript – Statements

## Palavras Chaves

As Statements iniciam frequentemente com uma palavra-chave para identificar uma ação JavaScript que tem de ser efetuada;

| Keyword / Palavra-Chave | Descrição |
|---|---|
| **do ... while** | Executa um bloco de Código e repete-o enquanto a condição for verdadeira |
| **for** | Marca um bloco de codigo para ser executado, enquanto houver uma condição verdadeira |
| **function** | Declara uma função |
| **if ... else** | Marca um bloco de Código para ser executado dependendo de uma condição |
| **var** | **Declarar uma variável** |

# JavaScript – Sintaxe

## Sintaxe

A sintaxe JavaScript é o conjunto de regras de como os programas em JavaScript são construídos.

```
var x, y;          // Como se declara variáveis
x = 5; y = 6;      // Como se atribui valores;
z = x + y;         // Como se processa valores
```

# JavaScript – Sintaxe

## Valores

A sintaxe JavaScript define dois tipos de valores:

**Valores Fixos;**
**Valores variáveis;**

Os **valores fixos** são chamados de **Literais**;
Os **valores vaiáveis** são chamados de **Variáveis**

# JavaScript – Sintaxe

## Valores - Literais

A regra mais importante para escrever valores fixos são:

**Os Números** são escritos com ou sem decimais;

**10.50**
**1001**

**As Strings** são escritas com aspas ou plicas;

**"João Manel"**

**‘João Manel'**

# JavaScript – Sintaxe

## Valores - Variáveis

Numa linguagem de programação as variáveis usadas para guardar valores que lhes são atribuídos.
O JavaScript usa a keyword para declarar variáveis;
O sinal de igual é usado para atribuir valores à variável.
No exemplo que se segue é declarada a varável X e depois é-lhe atribuído (dado) o valor 6

```
var x;

x = 6;
```

As variáveis são contentores para guardar os valores dos dados que vão sendo processados.

# JavaScript – Sintaxe

## Variáveis - Identificadores

As variáveis devem ser identificadas por nomes únicos que chamamos de identificadores.
Os identificadores de variáveis poderão ser nomes curtos como um carater (x ou y, por exemplo) ou nomes mais descritivos (idade; soma; valorTotal)

**Regras para construir nomes de variáveis** (indentificadores únicos), são:

1. **Os nomes podem conter, letras, dígitos, underscores e cifrão;**
2. **Os nomes devem de começar por uma letra;**
3. **Os nomes podem começar também com um $ ou _**
4. **Nomes são "case sensitive" isto é sensíveis às letras maiúsculas e minúsculas (y e Y são variáveis diferentes)**
5. **Palavras reservadas (tipo palavras-chaves) não podem ser usadas como nomes**

# JavaScript – Sintaxe

## Variáveis – Operador de atribuição

Em JavaScript o sinal de igual (=) é o operador de atribuição e não o operador de “igual a”

```
X = X + 5;
```

O operador “igual a”, em JavaScript é escrito da seguinte forma “==“

```
var X = 5;
var Y = 6;
var soma;
      if ( X == 5){
        soma = X + Y;
      }
```

# JavaScript – Sintaxe

## Variáveis – Declaração e Tipos

O JavaScript manipula muitos tipos de dados: números; strings; arrays; boleanos; objectos; etc.

Para declarar uma variável usa-se a palavra-chave var

```
var x, y;
var nome, ultimoNome;
```

# JavaScript – Sintaxe

## Variáveis – Tipos de Dados

O JavaScript manipula muitos tipos de dados: números; strings; arrays; boleanos; objectos; etc.

Em programação o conceito de tipo de dados é muito importante, para estarmos aptos a operar variáveis é importante conhecer como é que funcionam.

```
var idade = 16;                          // Number
var ultimoNome = "Johnson";                    // String
var x = {primeiroNome:"João", ultimoNome:“Silva“, idade: 25};    //Objeto
```

# JavaScript – Sintaxe

## Variáveis – Tipos de Dados

Sem a definição do tipo de dados o computador não resolvia a seguinte situação;

```
var x = 16 + "Volvo";
```

Faz sentido adicionar “Volvo” a dezasseis? Será que produz um erro ou obtemos um resultado?

Quando tentamos adicionar um numero a uma string o JavaScript vai tratar o número como uma string, é o mesmo que: var x = ”16” + "Volvo“

O resultado é: “16Volvo”

# JavaScript – Sintaxe

## Variáveis – Tipos de Dados

No entanto, o JavaScript interpreta o código da esquerda para a direita, logo diferentes sequencias obtemos  resultados diferentes, exemplo:

```
var x = 16 + 4 + "Volvo"; //resultado 20Volvo

var x = "Volvo“ + 16 + 4; //resultado Volvo164
```

# JavaScript – Sintaxe

## Variáveis – Tipos de Dados

O tipo de dados em JavaScript é dinâmico isto quer dizer que a mesma variável pode guardar diferentes tipos de dados:

```
var x;          // x é indefinida
x = 5;          //  Agora  x é um número
x = “Manel";      // Agora  x é uma String
```

# JavaScript – Sintaxe

## Variáveis – Tipos de Dados

Tipos de dados que podemos manipular em javaScript:

**Strings:**

```
var carNome1 = "Volvo XC60";  // Aspas
var carNome2 = 'Volvo XC60';  // Apóstrofo
```

Podemos usar Apóstrofos e Aspas dentro de uma string desde que não sejam do mesmo tipo das que envolvem a string.

```
var answer1 = "It's alright";
var answer2 = "He is called 'Johnny'";
var answer3 = 'He is called "Johnny"';
```

# JavaScript – Sintaxe

## Variáveis – Tipos de Dados

Tipos de dados que podemos manipular em javaScript:

**Números:**

```
var x1 = 34.00;     // Com casas decimals
var x2 = 34;        // Sem casas decimals
```

Grandes valores ou pequenos valores podem ser escritos com a notação exponencial:

```
var y = 123e5;      // 12300000
var z = 123e-5;     // 0.00123
```

# JavaScript – Sintaxe

## Variáveis – Tipos de Dados

Tipos de dados que podemos manipular em javaScript:

**Booleanos:**
So podem ter dois valores true ou false

```
var x = 5;
var y = 5;
var z = 6;
(x == y)       // Devolve true
(x == z)       // Devolve false
```

Os booleanos são usados para fazer testes condicionais

# JavaScript – Sintaxe

## Variáveis – Tipos de Dados

Tipos de dados que podemos manipular em javaScript:

**Arrays:**
Os arrays em JavaScript são inscritos entre parêntesis retos, os seus valores separados por vírgulas e os valores do tipo string têm de estar escritos entres aspas, o seguinte exemplo é declarada uma variável array com o nome “carros” para guardar as marcas.

```
var carros = ["Saab", "Volvo", "BMW"];
```

Nota: os arrays só podem ser de um tipo de dados;

# JavaScript – Sintaxe

## Variáveis – Tipos de Dados

Tipos de dados que podemos manipular em javaScript:

**Arrays:**
Os arrays em JavaScript são inscritos entre parêntesis retos, os seus valores separados por vírgulas e os valores do tipo string têm de estar escritos entres aspas, o seguinte exemplo é declarada uma variável array com o nome “carros” para guardar as marcas.

```
var carros = ["Saab", "Volvo", "BMW"];
```

Nota: os arrays só podem ser de um tipo de dados;

# JavaScript – Sintaxe

## Variáveis – Tipos de Dados

Tipos de dados que podemos manipular em javaScript:

**Objetos:**
Os objetos servem para definir as propriedades de “coisas” do mundo real. As propriedades são escritas aos pares **nome:valor** separadas por virgulas dentro de chavetas.
Exemplo: Para definir o objeto pessoa

Var pessoa =
**{primeiroNome:“João”, ultimoNome:“Silva”; idade: 58, corCabelo: “castanho”}**

# JavaScript – Sintaxe

## Variáveis – Tipos de Dados

Converter strings para números:

O método Global Number() converte strings que contêm números (tipo “3.14”) para (tipo 3.14), strings vazias converte para zero. Tudo o resto converte para NaN (Not a Number).

```
Number("3.14")    // devolve 3.14
Number(" ")       // devolve 0
Number("")        // devolve 0
Number("99 88")   // devolve NaN
```

# JavaScript – Sintaxe

## Variáveis – Tipos de Dados

Converter strings para números:

O método parseInt()

```
parseInt("10");          // devolve 10
parseInt("10.33");       // devolve 10
parseInt("10 20 30");    // devolve 10
parseInt("10 years");    // devolve 10
parseInt("years 10");    // devolve NaN
```

# JavaScript – Sintaxe

## Variáveis – Tipos de Dados

Converter strings para números:

O método parseFloat()

```
parseFloat("10");          // devolve 10
parseFloat("10.33");       // devolve 10.33
parseFloat("10 20 30");    // devolve 10
parseFloat("10 years");    // devolve 10
parseFloat("years 10");    // devolve NaN
```

# JavaScript – Sintaxe

## Operadores aritméticos

| Operador | Descrição |
|---|---|
| + | Adição |
| - | Subtração |
| * | Multiplicação |
| ** | Exponenciação |
| / | Divisão |
| % | Modus (Resto da divisão inteira) |
| ++ | Incremento |
| -- | Decremento |

# JavaScript – Sintaxe

## Operadores de atribuição

| Operador | Exemplo | É o mesmo que: |
|---|---|---|
| = | x = y | x = y |
| += | x += y | x = x + y |
| -= | x -= y | x = x - y |
| *= | x *= y | x = x * y |
| /= | x /= y | x = x / y |
| %= | x %= y | x = x % y |

# JavaScript – Sintaxe

## Tipos de Dados e Operadores

**Atenção:**

**Devem de complementar o vosso estudo do tipo de variáveis  e suas conversões com a consulta do sítio web da W3schools.**

**https://www.w3schools.com/js/default.asp**

# JavaScript – Functions

## Sintaxe

**As funções JavaScript são definidas** com a palavra chave **function** seguido do nome e de parêntesis. O nome das funções podem conter letras, underscores e dólares as mesmas regras que as variáveis. Nos parêntesis podemos incluir parâmetros separados por vírgulas **(parametro1, parametro2, …)**
O código a ser executado é colocado dentro de um par de chavetas **{ }**

```
function nome(parametro1, parametro2, parametro3) {
  // Código a ser executado
}
```

# JavaScript – Functions

## Invocar uma função

**Se uma função for invocada num statments ela tem de retornar ao ponto onde foi chamada para continuar a executar o código que está a seguir a esse statments.**

```
 // A função retorna o produto de a e b
function  minhaFuncao(a, b) {
  return a * b;
}

// A função é chamada, o valor returnado será atribuido a x
var x = minhaFuncao(4, 3);

document.getElementById("demo").innerHTML = x.toString();
```

# JavaScript – Functions

## Invocar uma função

**Se uma função for invocada num statments ela tem de retornar ao ponto onde foi chamada para continuar a executar o código que está a seguir a esse statments.**

```
 // A função retorna o produto de a e b
function  minhaFuncao(a, b) {
  return a * b;
}

// A função é chamada, o valor returnado será atribuido a x
var x = minhaFuncao(4, 3);

document.getElementById("demo").innerHTML = x.toString();
```

# JavaScript – Functions

## Invocar uma função

**Se invocar uma função sem usar o parêntesis será retornado a definição da função.**

```
function toCelsius(fahrenheit) {
  return (5/9) * (fahrenheit-32);
}
document.getElementById("demo").innerHTML = toCelsius; //Forma errada

document.getElementById("demo").innerHTML = toCelsius(); //Retorna NaN

document.getElementById("demo").innerHTML = toCelsius(77); //Forma correta
```

# JavaScript – Functions

## Variáveis locais

As varáveis declaradas dentro das funções só podem ser acedidas dentro das mesmas.

```
// Código aqui não consegue usar a variável carNome

function minhaFuncao() {
  var carNome = "Volvo";
  // Código aqui pode usar a variavel carNome
}

// Código aqui não consegue usar a variável carNome
```

# JavaScript – Objetos

## Declarar objetos

Os objetos são representações das propriedades de objetos do mundo real

| Propriedade | Valor da propriedade |
|---|---|
| primeiroNome | João |
| primeiroNome | Chaves |
| idade | 50 |
| corOlhos | azuis |

**Declaração em javaScript**

```
var pessoa = {
  primeiroNome: “João",
  ultimoNome: “Chaves",
  idade: 50,
 corOlhos: “azuis"
};
```

# JavaScript – Objetos

## Aceder as objetos

Pode-se aceder às propriedades e aos respetivos valores usando um dos seguintes métodos:

```
var pessoa=
{primeiroNome:“João", ultimoNome:“Chaves", idade:50, corOlhos: “azuis"};
```

```
var n = pessoa.primeiroNome;

var x = pessoa[“primeiroNome”];
```

```
document.getElementById("demo").innerHTML = n.toString();

document.getElementById("demo1").innerHTML = x.toString();
```

# JavaScript – Objetos

## Os objetos também podem conter métodos

```
var pessoa = {
        primeiroNome:“João",
        ultimoNome:“Chaves",
        idade:50,
        corOlhos: “azuis“,
        nomeCompleto: function(){
                return this.primeiroNome + “ “ + this.ultimoNome;
        }
};
```

**Palavra Chave this**

Na definição da função, this refere-se à própria função. No exemplo em cima, this refere-se ao objeto pessoa que é o dono da função nomeCompleto, por outras palavras this.primeiroNome quer referir-se à propriedade primeiroNome deste objeto.

Nota: deve de ler mais sobre a palavra this em w3schools

# JavaScript – Objetos

## Os objetos também podem conter métodos

```
var pessoa = {
        primeiroNome:“João",
        ultimoNome:“Chaves",
        idade:50,
        corOlhos: “azuis“,
        nomeCompleto: function(){
                return this.primeiroNome + “ “ + this.ultimoNome;
        }
};
```

**Como aceder:**

```
document.getElementById("demo").innerHTML =  pessoa.nomeCompleto();
```

# JavaScript – Eventos

## Os eventos mais comuns em HTML

| Eventos | Descrição |
|---|---|
| onchange | O elemento HTML foi alterado |
| onclick | O utilizador clicou no elemento HTML |
| onmouseover | O utilizador passou com o rato sobre o elemento HTML |
| onmouseout | O utilizador moveu o rato para fora do elemento HTML |
| onkeydown | O utilizador pressionou uma Tecla do teclado |
| onload | O browser terminou de carregar a página HTML |

# JavaScript – Strings

## Métodos e propriedades de strings

**length**

Tamanho de uma string em número de caracteres (inclui espaços em branco)

```
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
var sln = txt.length;
```

**indexOf()**

Encontrar uma string dentro de outra string, devolve a posição da primeira ocorrência da string que procura

```
var str = “Por favor localiza onde a palavra ‘localiza’ ocorre!";
var pos = str.indexOf("localiza");
```

**lastIndexOf()**

Encontrar a última ocorrência de uma string

```
var str = “Por favor localiza onde a palavra ‘localiza’ ocorre!";
var pos = str. lastIndexOf("localiza");
```

# JavaScript – Strings

## Métodos e propriedades de strings

`search()`

Procura e devolve a posição de uma string dentro de outra string

```
var str = “Por favor localiza onde a palavra ‘localiza’ ocorre!";
var pos = str. search("localiza");
```

## Extrair valores de strings

`slice(posInicio, posFim)`

Extrai partes de strings dentro de outra string, com a indicação da posição de inicio e posição de fim

```
var str = "Apple, Banana, Kiwi";
var res = str.slice(7, 13); // retorna “Banana”
var res = str.slice(-12, -6); //retorna “Banana”
var res = str.slice(7); // retorna “Banana, Kiwi”
```

# JavaScript – Strings

## Métodos e propriedades de strings

substr(posInicio, posFim)

É similar a slilce()

```
var str = "Apple, Banana, Kiwi";
var res = str. substr(7, 13); // retorna “Banana”
var res = str. substr(-12, -6); //retorna “Banana”
var res = str. substr(7); // retorna “Banana, Kiwi”
```

raplace(posInicio, posFim)

Substitui palavras/caracteres/strings dentro de uma string. Devolve uma nova string com as palavras que substitui-o.

str = "Por favor visite a Microsoft!";
var n = str.raplace("Microsoft", "W3Schools"); //retorna “Por favor visite a W3Schools!";

Nota: ler mais sobre este método em W3schools

# JavaScript – Strings

## Outros métodos

| método | Descrição |
|---|---|
| trim() | Remove espaços em branco numa string |
| charAt(*posição*) | Retorna o caractere numa determinada posição |
| charCodeAt(*posição*) | Devolve o Código Ascii do caracter de uma determinada posição |
| split() | Transforma uma string num **array**<br>`var txt = "a,b,c,d,e";  // String`<br>`txt.split(",");  // Split on commas`<br>`txt.split(" ");  // Split on spaces`<br>`txt.split("\|");` |
| | |

# JavaScript – Arrays

## Manipulando arrays

**Criar arrays:**

```
var cars = ["Saab", "Volvo", "BMW"];
```

```
var cars = [
  "Saab",
  "Volvo",
  "BMW"
];
```

Espaços e linhas não são importantes. A declaração de um array pode-se estender por mais que uma linha.

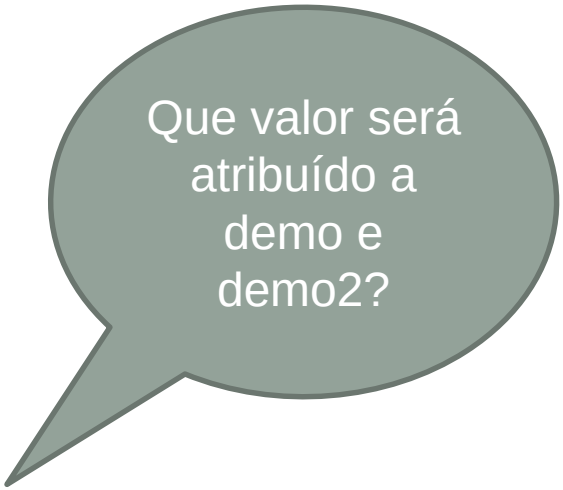
# JavaScript – Arrays

## Manipulando arrays

**Aceder aos valores de um array:**

```
<p id="demo"></p>
<p id="demo1"></p>
<p id="demo2"></p>

<script>
var cars = ["Saab", "Volvo", "BMW"]; //Declarar o array
var nome = cars[0]; //Acesso à posição zero do array
document.getElementById("demo").innerHTML = nome
document.getElementById("demo1").innerHTML = cars[1];
document.getElementById("demo2").innerHTML = cars;
</script>
```



# JavaScript – Arrays

## Métodos

| método | Descrição |
|---|---|
| **toString()** | **Converte um array numa string**<br>`var frutas = ["Banana", “Laranja", "Ananás", "Mango"];`<br>`document.getElementById("demo").innerHTML = frutas.toString();`<br>`// Resultado: Banana, Laranja, Ananás, Mango` |
| **join()** | **Similar ao anterior mas acrescenta um valor entre os valores do array**<br>`var frutas = ["Banana", “Laranja", "Ananás", "Mango"];`<br>`document.getElementById("demo").innerHTML = frutas.join(" * ");`<br>`// Resultado: Banana * Laranja * Ananás * Mango` |
| **pop()** | `var frutas = ["Banana", “Laranja", "Ananás", "Mango"];`<br>`frutas.pop();`<br>`// Remove o último elemento ("Mango") do array fruitas` |

# JavaScript – Arrays

## Métodos

| método | Descrição |
|---|---|
| **push()** | **Adiciona um novo elemento a um array e retorna o novo tamanho do array**<br>`var frutas = ["Banana", "Laranja", "Ananás", "Mango"];`<br>`var x = frutas.push("Kiwi"); //o valor de x é 5`<br>`// Resultado: ["Banana","Laranja","Ananás","Mango","kiwi"]` |
| **shift()** | **Remove o primeiro elemento do array e retorna o valor retirado**<br>`var frutas = ["Banana", "Laranja", "Ananás", "Mango"];`<br>`var x = frutas.shift(); //o valor de x é "Banana"`<br>`// Resultado: ["Laranja","Ananás","Mango"]` |
| **unshift()** | **Adiciona um novo elemento no inicio de um array**<br>`var frutas = ["Banana", "Laranja", "Ananás", "Mango"];`<br>`var x = frutas.unshift("Limão");  //o valor de x é 5`<br>`// Resultado: ["Limão","Banana","Laranja","Ananás","Mango"]` |

# JavaScript – Arrays

## Métodos

| método | Descrição |
|---|---|
| concat() | var arr1 = ["Cecília", “Luana"];<br>var arr2 = ["Emilia", "Tobias", "Lino"];<br>var arr3 = ["Rafael", "Mateus"];<br>var minhasCriancas = arr1.concat(arr2, arr3); |

Nota
Para conhecer outros métodos devem de consultar a seguinte página Web:

https://www.w3schools.com/js/js_array_methods.asp

# JavaScript – Arrays

## Métodos de ordenação

Ordenando um array com o método sort()

```
var frutas = ["Banana", “Laranja", "Ananás", "Mango"];
frutas.sort(); // Ananás, Banana, Laranja, Mango
```

O método reverse() ordena de trás para frente, para ordenar de forma descendente temos de:

```
var frutas = ["Banana", “Laranja", "Ananás", "Mango"];
frutas.sort(); //primeiro ordenar e depois
frutas.reverse() // Mango, Laranja, Banana, Ananás
```

# JavaScript – Arrays

## Métodos de ordenação

Ordenando um array com o números

**Problema:** os números são ordenados como strings, logo "25" é maior que "100", porque o "2" é maior que "1", por essa razão o método sort() produz um valor incorreto…

Para resolver o problema temos de providenciar uma função de comparação:

```
var pontuacao = [40, 100, 1, 5, 25, 10];
pontuacao.sort(function(a, b){return a - b});
```

# JavaScript – Arrays

## Métodos de ordenação

Ordenando um array com o números

Quando o método sort() compara dois valores ele envia dois valores para a função de comparação e ordena-os de acordo com o valor retornado (negativo, zero, positivo)

**Algoritimo:**

```
var x = comparar(a, b)
se x < 0 então a < b;
se x==0 então a==b;
se x > 0 então a > b;
```

```
var pontuacao = [40, 100, 1, 5, 25, 10];
pontuacao.sort(function(a, b){return a - b});
```

Assim, quando envia 40 para comparar com 100, o metodo sort() vai chamar a função de comparação function(40,100). A função vai calcular 40-100 e retorna -60 (um valor negativo) e assim o método sort vai ordenar 40 como menor que 100

# JavaScript – Arrays

## Métodos de ordenação

Ordenando um array com o números

`Ordenação Ascendente`

```
var pontuacao = [40, 100, 1, 5, 25, 10];
pontuacao.sort(function(a, b){return a - b});
```

`Ordenação descendente`

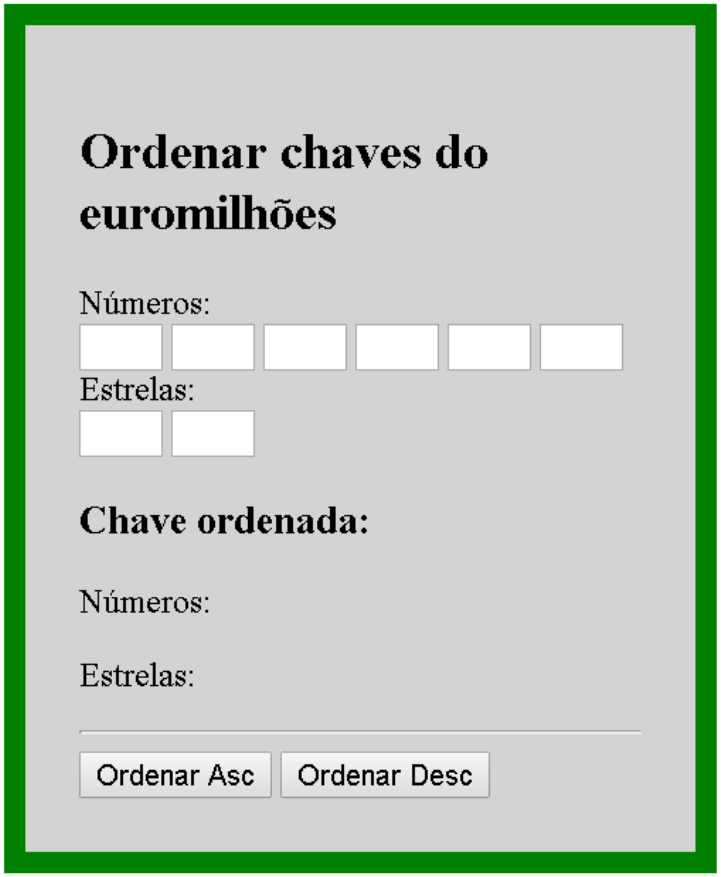
```
var pontuacao = [40, 100, 1, 5, 25, 10];
pontuacao.sort(function(a, b){return b - a});
```

# JavaScript – Arrays

## Métodos de ordenação

**Exercício**

Executar todas as operações necessárias afim de obter uma página web que mediante a introdução dos números da chave do Euromilhões ordene a mesma de forma ascendente e forma descendente.




Nota:
Attribute size
Este atribute especifica o tamanho do input em caracteres

```
<input type="text" name="firstname" value
="John" size="40">
```

# Uma solução possível

```
<script>
var chave = [0,0,0,0,0,0];
var estrelas = [0,0];
function lerValores(){
    chave[0]=parseInt(document.getElementById("n1").value);
    chave[1]=parseInt(document.getElementById("n2").value);
    chave[2]=parseInt(document.getElementById("n3").value);
    chave[3]=parseInt(document.getElementById("n4").value);
    chave[4]=parseInt(document.getElementById("n5").value);
    chave[5]=parseInt(document.getElementById("n6").value);
    estrelas[0]=parseInt(document.getElementById("e1").value);
    estrelas[1]=parseInt(document.getElementById("e2").value);
}
function ordenarAsc(){
  lerValores();
  chave.sort(function(a, b){return a - b});
  document.getElementById("numeros").innerHTML = chave;
  estrelas.sort(function(a, b){return a - b});
  document.getElementById("estrelas").innerHTML = estrelas;
}
function ordenarDesc(){
    lerValores();
    chave.sort(function(b, a){return a - b});
    document.getElementById("numeros").innerHTML = chave;
    estrelas.sort(function(b, a){return a - b});
    document.getElementById("estrelas").innerHTML = estrelas;
}
</script>
```

# JavaScript – Arrays

## Métodos de interação

Os métodos de interação de arrays operam em todos os itens de um array

### forEach()

O método forEach é um método recursivo (callback) que percorre todos os elementos de um array.

**Exemplo:**

```
var txt = "";
var num = [45, 4, 9, 16, 25];
num.forEach(mFuncao);

function  mFuncao(valor, index, array) {
  txt = txt + valor + "<br>";
}
```

# JavaScript – Boleanos

## Valores Boleanos

Muitas vezes em programação precisamos de valores que só podem conter um de dois valores, como por exemplo:

- SIM / NAO
- ON / OFF
- TRUE / FALSE

Para estes casos o javaScript possui um tipo de dados que são os **Boolean,** que só aceitam dois valores **true** or **false**

# JavaScript – Boleanos

## A função boleana

Pode-se usar a função boleana para determinar se uma dada expressão é verdadeira ou falsa.

```
Boolean(10 > 9)          // retorna true

Ou de outra forma mais fácil

(10 > 9)                 // também retorna true
10 > 9                   // também retorna true
```

# JavaScript – Operadores de comparação e Lógicos

Para um valor de x=5 então vamos obter os seguintes resultados de comparação

| Operador | Descrição | Comparando | Valor devolvido |
|---|---|---|---|
| == | Igual a | x == 8 | false |
| | | x == 5 | true |
| | | x == "5" | true |
| === | Valor igual e tipo de dado | x === 5 | true |
| | | x === "5" | false |
| != | Diferente | x != 8 | true |

# JavaScript – Operadores de comparação e Lógicos

Para um valor de x=5 então vamos obter os seguintes resultados de comparação

| Operador | Descrição | Comparando | Valor devolvido |
|---|---|---|---|
| !== | Valor diferente e tipo de dado diferente | x !== 5 | false |
| | | x !== "5" | true |
| | | x !== 8 | true |
| > | Maior que | x > 8 | false |
| < | Menor que | x < 8 | true |
| >= | Maior ou igual que | x >= 8 | false |
| <= | Menor ou igual que | x <= 8 | true |

# JavaScript – Condições

## if (condição)

Quando estamos a escrever código muitas vezes precisamos de diferentes ações para diferentes decisões… Com as expressões condicionais podemos garantir que uma determinada ação seja executada sempre que uma condição se verifique.

Para executar um bloco de código usa-se um statment if

```
if (idade < 18) {
  aviso = “Menor de idade.";
}
```

Nota: IF – ao usar-mos letras maiúsculas o JavaScript dá erro.

# JavaScript – Condições

## If(condição){…} else{…}

Se estivermos a analisar a variável idade para responder se é maior ou não de idade pode-se usar o else

```
if (idade < 18) {
  aviso = “Menor de idade.";
} else {
  aviso = “Maior de idade.";
}
```

# JavaScript – Condições

## If (condição){…} else if(condição){…}else{…}

Imagine que pretende criar um sistema de saudação num sitio web e mediante as horas que o utilizador acede à página se a hora for menor que as 12h então o sistema escreve “bom dia!”, se for menor que as 19h escreve “boa tarde!” e maior que 19h escreve “boa noite!”

```
if (hora < 12) {
  saudacao = “bom dia!";
} else if (hora < 19) {
  saudacao = “boa tarde";
} else {
  saudacao = “boa noite!";
}
```

# JavaScript – Condições

## switch

A expressão switch é usada para executar diferentes ações baseadas em diferentes decisões (quando determinada condição se verifica).
O método getDay devolve o dia da semana de uma data (0 = domingo; 1 = segunda; 2 = terça, …)

```
switch (new Date().getDay()) {
  case 0:
    day = “Domingo";
    break;
  case 1:
    day = “Segunda";
    break;

      ...
  case 6:
    day = "Sábado";
}
```

# JavaScript – Objeto Math

## Métodos do objeto Math

Este objeto ajuda-nos a efetuar operações com números

| método | Devolve |
|---|---|
| `Math.PI;` | `// returns 3.141592653589793` |
| `Math.round(4.7);`<br>`Math.round(4.4);` | `// returns 5`<br>`// returns 4` |
| `Math.pow(8, 2);` | `// retorna o valor de oito elevado a dois que é 64` |
| `Math.sqrt(64);` | `// calcula a raiz quadrada de 64 e retorna o resultado que é 8` |
| `Math.abs(-4.7);` | `// retorna o valor absoluto (positivo), 4.7` |

# JavaScript – Objeto Math

## Métodos do objeto Math

Este objeto ajuda-nos a efetuar operações com números

| método | Devolve |
|---|---|
| `Math.ceil(4.4);` | `// arredonda para cima devolve 5` |
| `Math.floor(4.7);` | `// arredonda para baixo devolve 4` |
| `Math.min(0, 150, 30, 20, -8, -200);` | `// retorna o menor valor do conjunto que é -200` |
| `Math.max(0, 150, 30, 20, -8, -200);` | `// retorna o maior valor do conjunto que é 150` |
| | |

# JavaScript – Objeto Math

## Math.random()

Retorna um valor entre zero (inclusive) e um

| método | Devolve |
|---|---|
| `Math.random();` | `// devolve um número aleatório entre 0 e 1` |
| **Como gerar números inteiros aleatórios em javaScript** | |
| `Math.floor(Math.random() * 10);` | `// devolve um número aleatório entre 0 e 9` |
| `Math.floor(Math.random() * 11);` | `// devolve um número aleatório entre 0 e 10` |
| `Math.floor(Math.random() * 100);` | `// devolve um número aleatório entre 0 e 99` |

# JavaScript – Objeto Math

## Math.random()

Uma boa ideia para o uso dos exemplos anteriores é criar uma função para todas as situações, por exemplo:

```
function getRndInteger(min, max) {
  return Math.floor(Math.random() * (max - min + 1) ) + min;
}
```

```
//obter o número do eurmilhões

        chave[0] = getRndInteger(1, 49);
```

# JavaScript – Loops

Os loops servem para executar o mesmo bloco de código um número de vezes, em vez de fazermos isto;

```
text += cars[0] + "<br>";
text += cars[1] + "<br>";
text += cars[2] + "<br>";
text += cars[3] + "<br>";
text += cars[4] + "<br>";
text += cars[5] + "<br>";
```

Podemos fazer assim:

```
var i;
for (i = 0; i < cars.length; i++) {
  text += cars[i] + "<br>";
}
```

# JavaScript – Lopps

```
for(statement 1; statement 2; statement 3){
//Bloco de código
}
```

Executa um bloco de código um número de vezes previamente conhecido…

```
for (i = 0; i < 5; i++) {
  text += “O número é: " + i + "<br>";
}
```

Normalmente só se inicia uma variável no entanto podemos inicializar mais do que uma variável

```
for (i = 0, len = cars.length, text = ""; i < len; i++) {
  text += cars[i] + "<br>";
}
```

# JavaScript – Lopps

## while(condição){ //bloco de código }

Executa um bloco de código enquanto a condição for verdadeira, o número de vezes que vai repetir o bloco de código não é conhecido.

```
var i, n;
i = getRndInteger(1, 12);
n = getRndInteger(1, 12);
//para garantir que o valor de n é diferente do valor de i
while (i == n) {
  n = getRndInteger(1, 12);
}
```

# JavaScript – Lopps

```
do{
//Bloco de código
}while(condição)
```

Executa primeiro bloco de código e só depois verifica se a condição continua verdadeira. Se sim continua a executar o bloco de código se sor falso termina a execução.

```
do {
  text += “O número é" + i;
  i++;
}
while (i < 10);
```

# JavaScript – Lopps

do{ … }while(condição)
versus
while(condição){…}

```
var text = "";
var i = 10;
do {
   text += “O número é" + i;
   i++;
}
while (i < 10);
```

**O resultado é:**

O número é: 10

```
var text = "";
var i = 10;
while (i < 10) {
  text += "<br>O número é " + i;
  i++;
}
```

**O resultado é:**

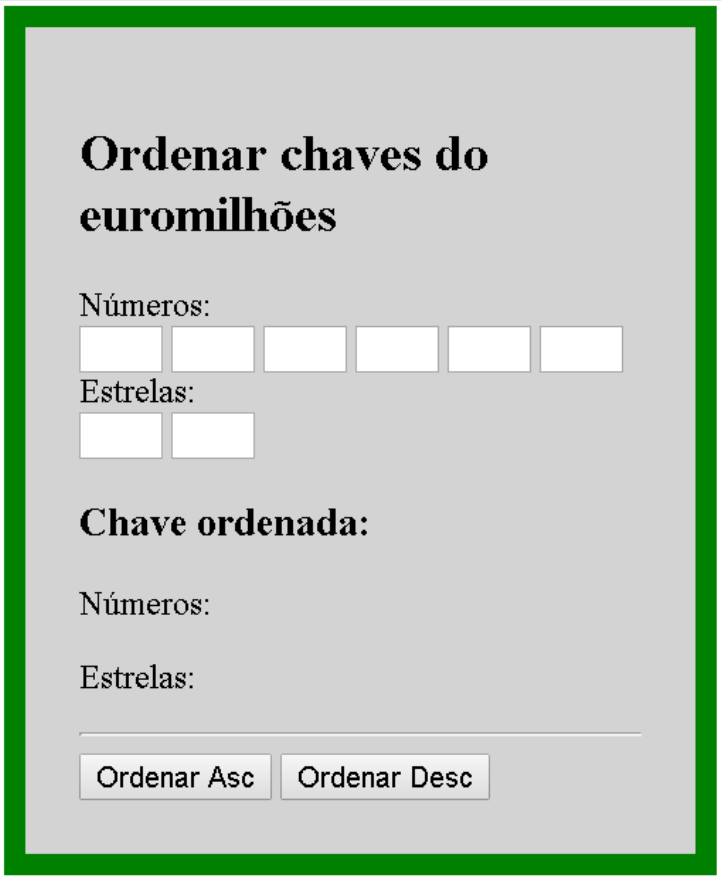
# JavaScript – Exercício

## Adicionar e procurar elementos num array

Alterar o exercício anterior *"Euromilhões"* de forma que sejam gerados números aleatórios.

Algoritmo

- Para todas as posições do array gerar um número aleatório;
- Encontrar a primeira estrela;
- Encontrar a segunda estrela e comparar à primeira;
- Escrever resultados



# JavaScript – Exercício

## Adicionar e procurar elementos num array

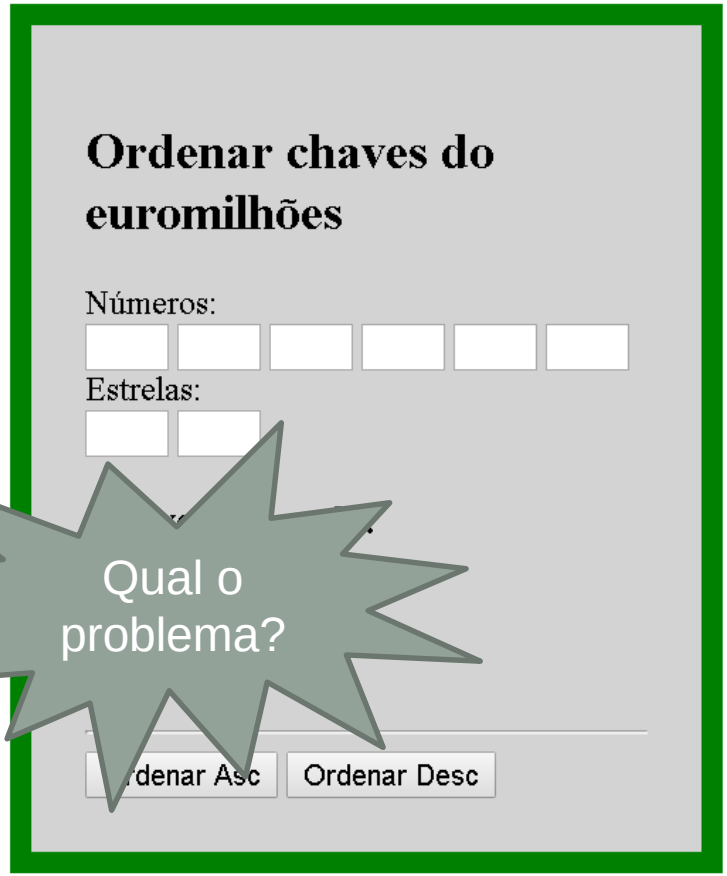
Algoritmo

- Para todas as posições do array gerar um número aleatório;

Para i = 0 até i < 6 fazer
    Chave[i] = obterNumAleatorio();
    i++;
Fim para

javaScript

```
for(i=0;i<6;i++){
    chave[i]=getRndInteger(1, 49);
}
```




Qual o problema?

# JavaScript – Exercício

## Adicionar e procurar elementos num array

Algoritmo

- Problema – podem aparecer números iguais;

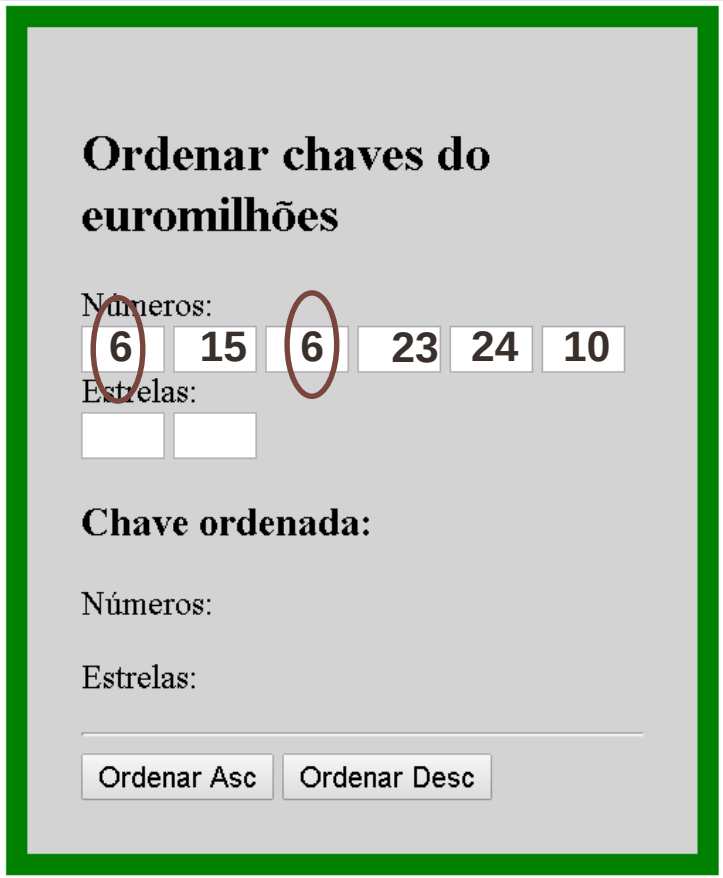
Para i = 0 até i < 6 fazer

Chave[i] = obterNumAleatorio();

Qual a solução?

i++;

Fim para



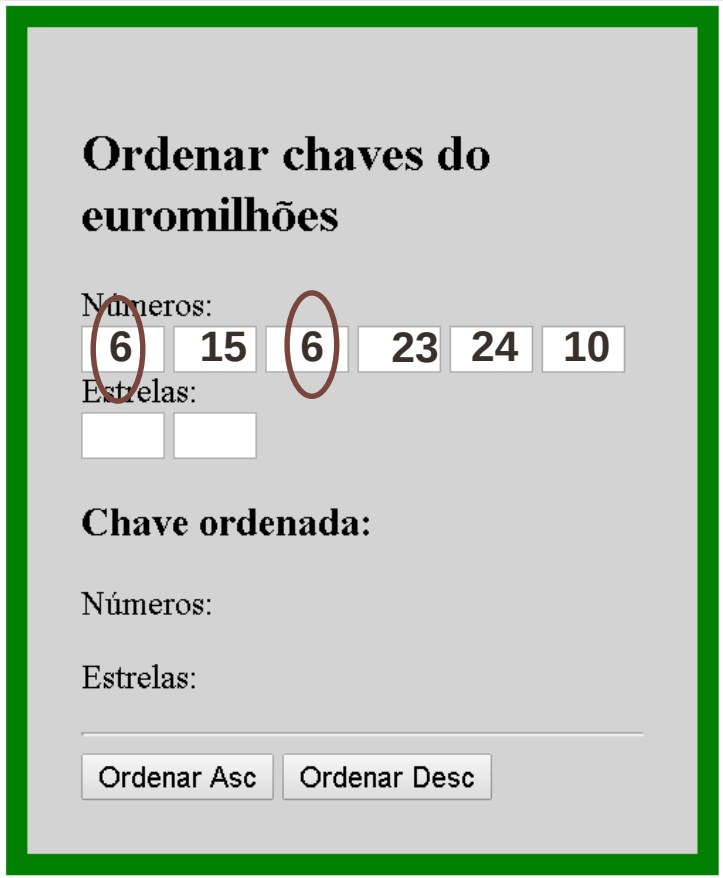
# JavaScript – Exercício

## Adicionar e procurar elementos num array

Algoritmo

- Solução – Comparar o número sorteado com todos os outros já existentes na chave, exceto o primeiro;

```
Para i =  0 até i < 6 fazer
    Chave[i] = obterNumAleatorio();
    Se i > 0 então
        Para id =  0 até id < i fazer
            Se Chave[id] == Chave[i] então
                Chave[i] = obterNumAleatorio();
                id =  -1;
            Fim Se
            id++;
        Fim para
    Fim Se
    i++;
Fim para
```



# JavaScript – Exercício

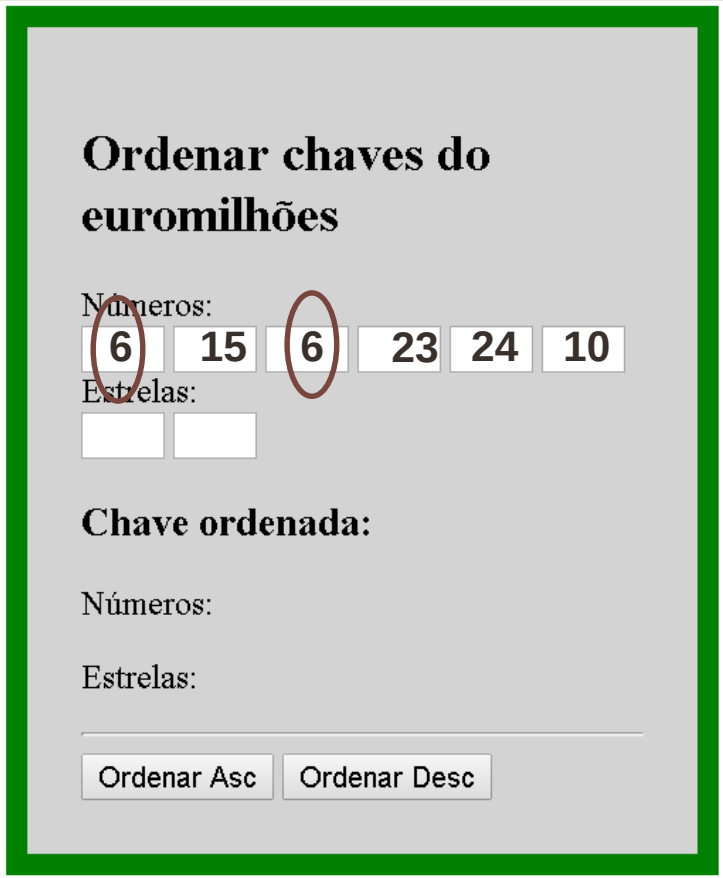
## Adicionar e procurar elementos num array

Algoritmo

- Solução – Comparar o número sorteado com todos os outros já existentes na chave, exceto o primeiro;

javaScript

```
...
var i, id;
for (i=0; i<6; i++) {
    chave[ i ] = getRndInteger(1, 49);
     if (i>0) {
         for(id = 0; id<i ; id++)
             if ( chave[ i ] == chave[ id ] ) {
                 chave[ i ] = getRndInteger(1, 49);
                 id = -1;
               }
          }
      }
}
...
```



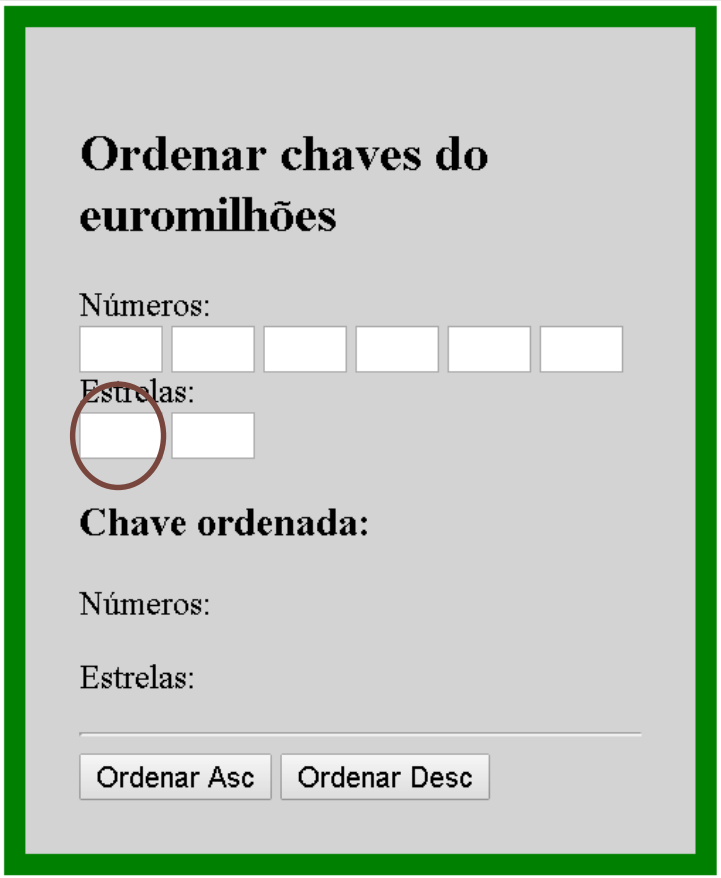
# JavaScript – Exercício

## Adicionar e procurar elementos num array

Alterar o exercício anterior *"Euromilhões"* de forma que sejam gerados números aleatórios.

Algoritmo

- ~~Para todas as posições do array gerar um número aleatório;~~
- **Encontrar a primeira estrela;**
- Encontrar a segunda estrela e comparar à primeira;
- Escrever resultados



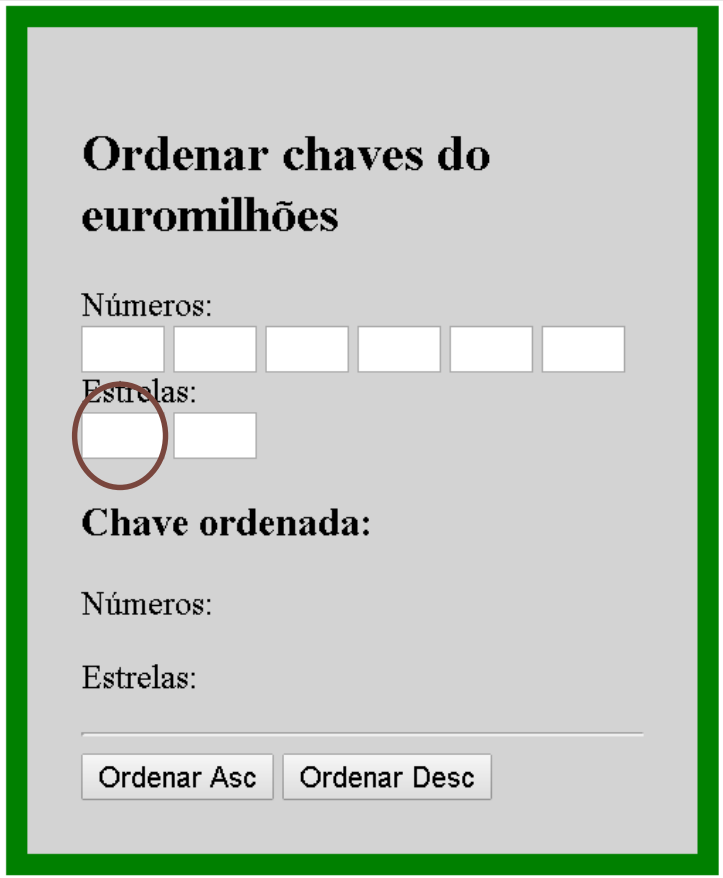
# JavaScript – Exercício

## Adicionar e procurar elementos num array

Alterar o exercício anterior *"Euromilhões"* de forma que sejam gerados números aleatórios.

Algoritmo

- ~~Para todas as posições do array gerar um número aleatório;~~
- **Encontrar a primeira estrela;**
- **Encontrar a segunda estrela e comparar à primeira;**
- Escrever resultados



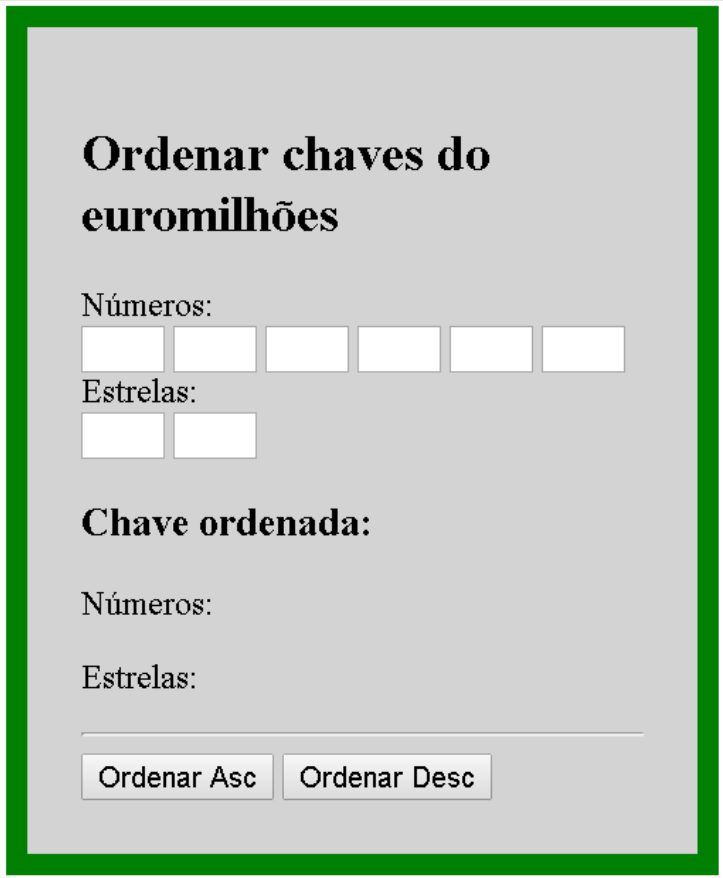
# JavaScript – Exercício

## Adicionar e procurar elementos num array

Algoritmo

- Encontrar a primeira estrela

```
...
var i, id;
for (i=0; i<6; i++) {
   chave[ i ] = getRndInteger(1, 49);
    if (i>0) {
        for(id = 0; id<i ; id++)
           if ( chave[ i ] == chave[ id ] ) {
                    chave[ i ] = getRndInteger(1, 49);
                    id = -1;
             }
          }
     }
}
estrelas[ 0 ] = getRndInteger(1, 12);
```



# JavaScript – Exercício

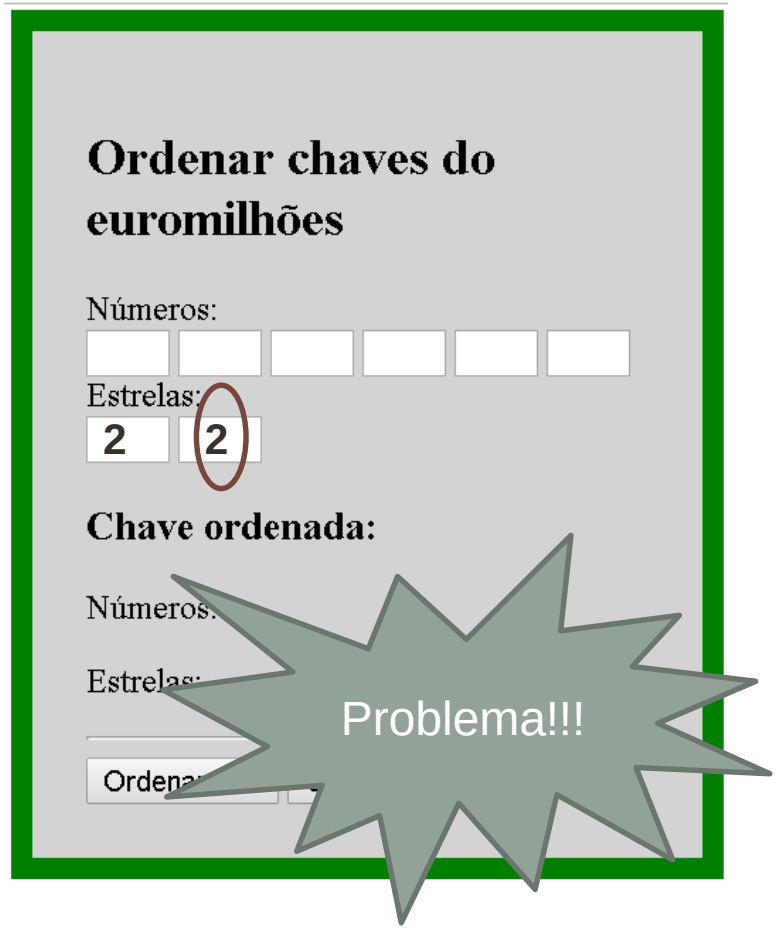
## Adicionar e procurar elementos num array

Algoritmo

- Encontrar a segunda estrela

```
...
var i, id;
for (i=0; i<6; i++) {
   chave[ i ] = getRndInteger(1, 49);
    if (i>0) {
        for(id = 0; id<i ; id++)
           if ( chave[ i ] == chave[ id ] ) {
                    chave[ i ] = getRndInteger(1, 49);
                    id = -1;
             }
          }
     }
}
```

**estrelas[ 0 ] = getRndInteger(1, 12);**
**estrelas[ 1 ] = getRndInteger(1, 12);**



# JavaScript – Exercício

## Adicionar e procurar elementos num array

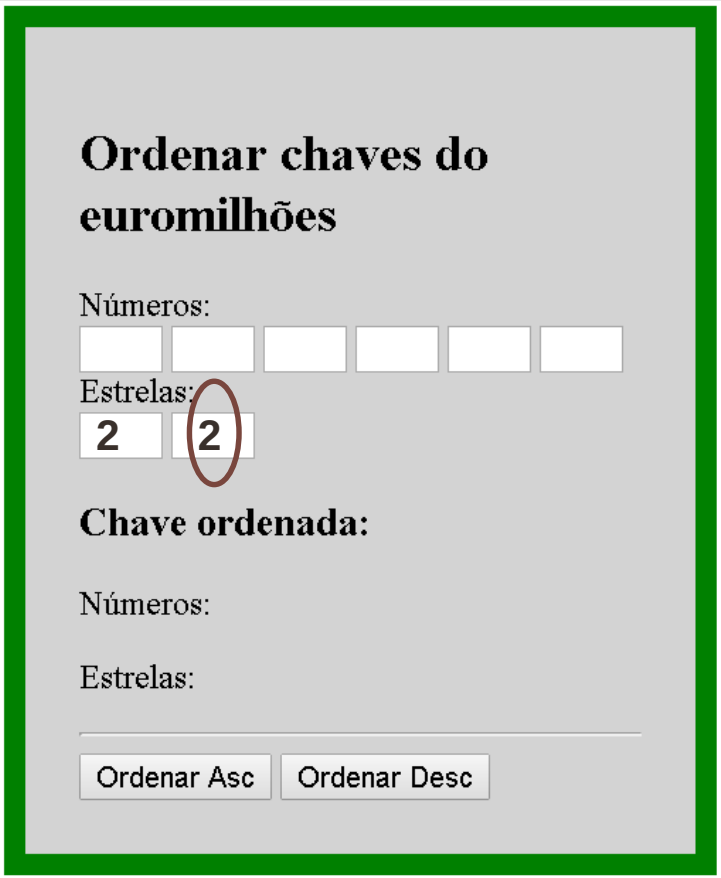
Algoritmo

- Comparar a segunda estrela com a primeira e se forem iguais gerar nova estrela

```
...
estrelas[ 0 ] = getRndInteger(1, 12);
estrelas[ 1 ] = getRndInteger(1, 12);

while (estrelas[0] == estrelas[ 1 ]) {

    estrelas[ 1 ] = getRndInteger(1, 12);
}
```



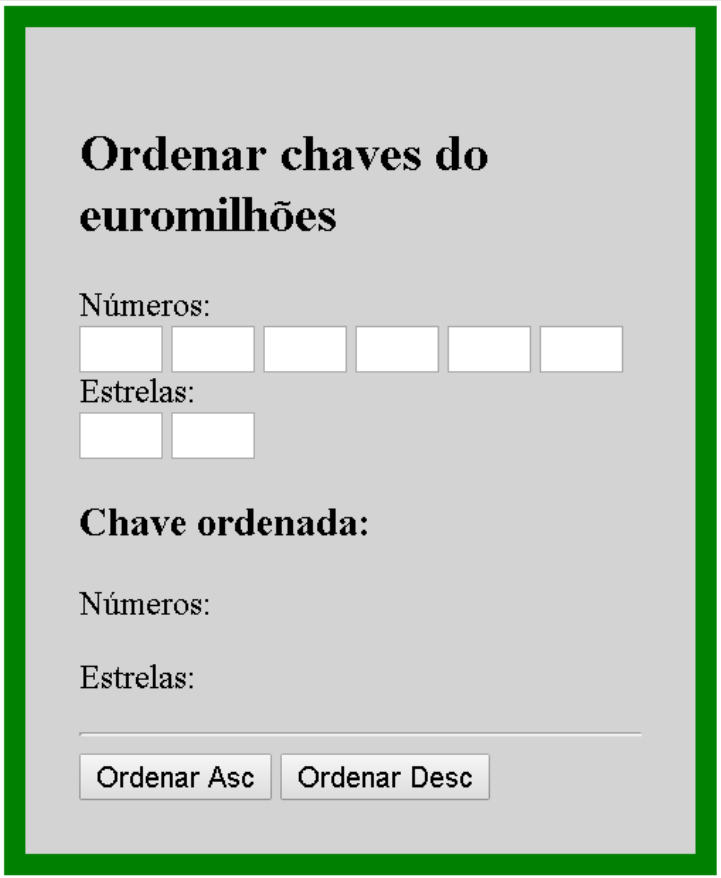
# JavaScript – Exercício

## Adicionar e procurar elementos num array

Alterar o exercício anterior *"Euromilhões"* de forma que sejam gerados números aleatórios.

Algoritmo

- ~~Para todas as posições do array gerar um número aleatório;~~
- ~~Encontrar a primeira estrela;~~
- ~~Encontrar a segunda estrela e comparar à primeira;~~
- Escrever resultados



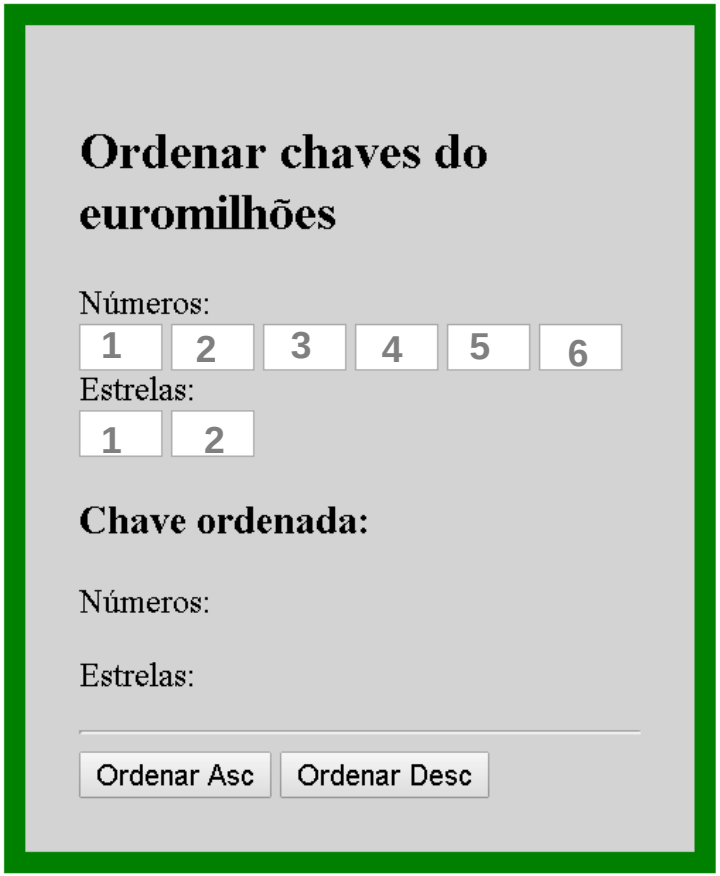
# JavaScript – Exercício

## Adicionar e procurar elementos num array

Algoritmo

- Escrever o resultado

```
...
estrelas[ 0 ] = getRndInteger(1, 12);
estrelas[ 1 ] = getRndInteger(1, 12);
while (estrelas[0] == estrelas[ 1 ]) {
        estrelas[ 1 ] = getRndInteger(1, 12);
}
document.getElementById("n1").value = chave[0];
document.getElementById("n2").value = chave[1];
document.getElementById("n3").value = chave[2];
document.getElementById("n4").value = chave[3];
document.getElementById("n5").value = chave[4];
document.getElementById("n6").value = chave[5];
document.getElementById("e1").value = estrelas[0];
document.getElementById("e2").value = estrelas[1];
```



# JavaScript – Exercício

```
function gerarChaves(){
    var i, id;
    for(i=0;i<6;i++){
        chave[i]=getRndInteger(1, 49);
        if(i>0){
            for(id=0;id<i;id++){
                if(chave[i]==chave[id]){
                    chave[i]=getRndInteger(1, 49);
                    id = -1;
                }
            }
        }
    }
    estrelas[0]=getRndInteger(1, 12);
    estrelas[1]=getRndInteger(1, 12);
    while(estrelas[0]==estrelas[1]){
        estrelas[1]=getRndInteger(1, 12);
    }
    document.getElementById("n1").value = chave[0];
    document.getElementById("n2").value = chave[1];
    document.getElementById("n3").value = chave[2];
    document.getElementById("n4").value = chave[3];
    document.getElementById("n5").value = chave[4];
    document.getElementById("n6").value = chave[5];
    document.getElementById("e1").value = estrelas[0];
    document.getElementById("e2").value = estrelas[1];
}
```