INSTITUTO SUPERIOR POLITECNICO DE TECNOLOGIAS E CIENCIAS

2º Semestre

2023/24

Introdução ao JavaScript

JavaScript

- Linguagem utilizada para adicionar comportamento as páginas web.
- Não têm relação com a linguagem de programação java.
- Pode alterar os contéudos de uma página web no seu carregamento ou em resposta a alguma acção do utilizador.
- É uma linguagem client side ou seja é executado no cliente (web browser).

Adição no documento HTML

- **Internal:** o código JavaScript fica entre a tag <script> e </script>

```
<body>
    <script>
        alert('Hello World');
    </script>
</body>
```

- **External:** o código JavaScript fica num ficheiro externo com extensão **.js** e depois referencia-se esse ficheiro tal como um ficheiro **.css**

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

Comandos de Saída JavaScript
Output

- **alert():** escreve o contéudo numa caixa de alerta
- **document.write():** escreve o contéudo no documento HTML

  Nota: Se for utilizado após a página estar carregada remove todo o conteúdo HTML. Por isso deve ser utilizado apenas para efeito de testes.

- **innerHTML():** escreve o contéudo num elemento HTML
- **console.log():** escreve o contéudo na consola do browser

- **Sintaxe:** conjunto de regras que ditam como devemos escrever um programa em JavaScript

- Um **programa** é composto por um conjunto de instruções e estas podem ser:
  - Variáveis
  - Palavras Reservadas
  - Operadores
  - Comentários
  - Expressões

- Uma instrução termina sempre com um **;**

- JavaScript é **untyped** ou seja quando declaramos uma variável não é necessario especificar o seu tipo.

- Para declarar varíaveis usamos a keyword(palavra reservada) **var**

- O **var** permite no momento da execução do código determinar qual o tipo de dados da variável

```
<script>
        var x = 10;
         var disciplina = “Aplicações Web”;
</script>
```

- O nome das variáveis podem conter nomes mas não podem começar com um número.

- Exemplos válidos
  - x1
  - y2
  - gift4you
- Exemplos Inválidos
  - 4Teste
  - 2give
  - 1ForAll

- As variáveis não podem ter operadores matemáticos ou lógicos.

- Exemplo inválidos:
  - nome-completo
  - boxes+bags
  - preco*5

- As variáveis não podem conter espaços em branco
- O nome das variáveis não podem ser palavras resrvadas do JavaScript mas podem conter palavras reservadas.

- **Exemplo inválidos:**
  - function
  - char
  - class
  - for
  - var
- **Exemplo Válidos:**
  - theFunction
  - myVar
  - forLoop

- O nome das variáveis são case-sensitive. MyData, myData, mydata, MYDATA são variáveis diferentes.

Operadores aritméticos

- Binários

Operador	Sintaxe	Descrição
+	op1 + op2	Calcula a soma de op1 com op2
-	op1 - op2	Calcula a diferença entre op1 e op2
*	op1 * op2	Calcula o produto de op1 com op2
/	op1 / op2	Calcula a divisão de op1 por op2
%	op1 % op2	Calcula o resto da divisão de op1 por op2

- Incremento ou decremento

Operador	Sintaxe	Descrição
++	++op	Incrementa o valor de op uma unidade; avalia o valor de op;
++	op++	Avalia o valor de op; incrementa o valor de op uma unidade;
--	--op	Decrementa o valor de op uma unidade; avalia o valor de op
--	op--	Avalia o valor de op; decrementa o valor de op uma unidade;

- Operadores relacionais

Operador	Sintaxe	Devolve true se:
>	op1 > op2	op1 for maior que op2
>=	op1 >= op2	op1 for maior ou igual a op2
<	op1 < op2	op1 for menor que op2
<=	op1 <= op2	op1 for menor ou igual op2
==	op1 == op2	op1 for igual a op2
!=	op1 != op2	op1 for diferente de op2

Operadores de atribuição

- Operador de atribuição normal: =
- Operadores de atribuição compactos

Operador	Sintaxe	É equivalente a:
+=	op1 += op2	op1 = op1 + op2
-=	op1 -= op2	op1 = op1 - op2
*=	op1 *= op2	op1 = op1 * op2
/=	op1 /= op2	op1 = op1 / op2
%=	op1 %= op2	op1 = op1 % op2

- Operadores lógicos

Operador	Sintaxe	Devolve true se:
&&	op1 && op2	op1 e op2 são ambos true (AND)
||	op1 || op2	ambos ou um deles é true (OR)
!	!op1	op1 é false (NOT)

- O operador **+** quando utilizado com strings concatena os valores em causa
  - Exemplo

```
<script>

    var firstName = "Jose";
    var lastName = "Antunes";
    var fullName = firstName + " " + lastName;

</script>
```

  - Qual o resultado disto?
    - Jose Antunes

JavaScript – **Tipos de Dados**

- Numa linguagem de programação um tipo de dados é definido pela:
  - Gama de valores que pode representar e pelas
  - Operações que sobre eles se podem efectuar
- Cada linguagem de programação disponibiliza ao programador um conjunto de tipo de dados próprio.
- Uma variável em JavaScript pode guardar valores dos seguintes tipos de dados:
  - Number
  - String
  - boolean
  - Array
  - Object

JavaScript – Numbers

- Para o JavaScript o tipo para os números é sempre o float 64 bits ao contrário de outras linguagens.
- Os números são guardados como vírgula fluatuante precisão dupla
- Os números podem ser inscritos com ou sem casas decimais

```
var nota = 15;
Var nota = 15.7;
```

- Por definição o JavaScript apresenta os números na base 10 mas podemos utilizar o metodo **.toString()** para mudar isso.

```
var numero = 128;
numero.toString(base que pretendemos);
```

JavaScript – **Numbers**

- **NaN** – Not a Number : é uma palavra reservada do JavaScript que indica que um valor não é um number

  - Exemplo

```
var x = 100 / "Tacuara“

alert(x);  // imprime NaN  (Not a Number)

var z = 100 / "10";  // o z já será um number
```

- **isNaN():** é uma função/ método que permite verificar se um dado valor é um number ou não.

```
var x = 100 / "Tacuara";  // o x será um NaN  (Not a Number)

isNaN(x);  // devolve true uma vez que x não é número
```

JavaScript
Funções para o tipo Number

- As funções para o tipo de dados **Number** ajudam-nos a realizar operações sobre os números.

- Existem dois tipos de métodos(funções):

  - **Globais**: podem ser usados em qualquer tipo de dados

    - Number() : converte uma variável num número
    - parseFloat() : converte o argumento recebido para um float
    - parseInt() : converte o argumento recebido para um inteiro

  - **Específicos para um Number**: são utilizados para o tipo de dados Number

    - toString() : devolve um número como String
      - Existem outros métodos para além do toString()

JavaScript
Funções para o tipo Number

- **Number()**: pode ser utilizado para converter varáveis em numbers

```
<script>

  var existe = true;
  Number(existe);  // converte o true para o seu valor por isso devolve 1


  var naoExiste = false;
  Number(naoExiste); // converte o false para o seu número por isso devolve 0

  var valor  = "10";
  Number(valor);    // converte a string x para um number por isso devolve 10

  var valor2 = "10 20";
  Number(valor2);   // devolve NaN porque não consegue converter o valor2 para um number

</script>
```

JavaScript
Funções para o tipo Number

- **parseInt()**: converte uma string em number, aceita espaços em brancos mas apenas devolve o primeiro número

```
parseInt("10");          // devolve 10
parseInt("10.33");       // devolve 10
parseInt("10 20 30");    // devolve 10
parseInt("10 kwanzas");   // devove 10
parseInt("kwanzas 10");   // devolve um NaN porque não é possível converter a string kwanzas para number
```

JavaScript – String

- As Strings são utilizadas para armazenar e manipular texto.

```
var disciplina = “Aplicações Web";
```

- O texto pode estar entre aspas ou plicas

var disciplina = "Aplicações Web";

ou

var disciplina = ' Aplicações Web';

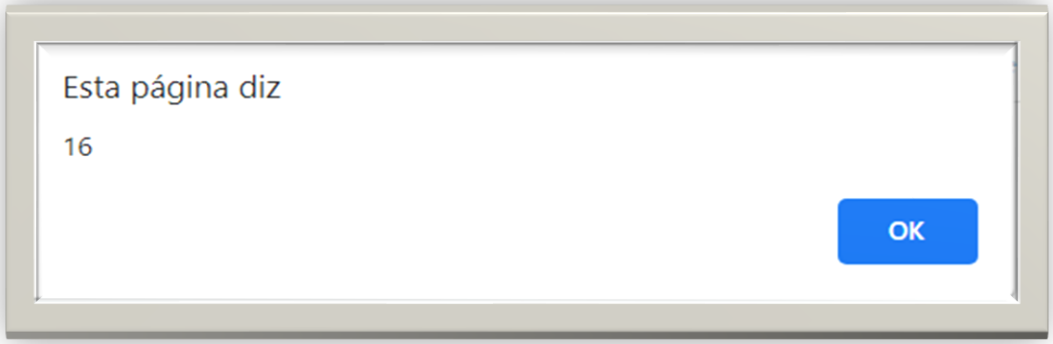
JavaScript – String Propriedades

- **String Length:** A propriedade length devolve-nos o tamanho de String

```
<script>
    var disciplina = "Aplicações Web";
    var tam = disciplina.length;

    alert(tam);
</script>
```

- **Qual o resultado disto?**



JavaScript – String
Funções

- As funções ajudam o programador a manipular as strings. O JavaScript têm muitas funções para esse efeito são algumas delas:

  - **charAt():** devolve-nos o caracter que se encontra na posição(índice) que passarmos como argumento.

```
<script>
        var frase = "Olá Mundo";
        var letra = frase.charAt(2);
</script>
```

- Qual o resultado disto?

  á

JavaScript – String
Funções

- **concat():** junta duas ou mais strings e devolve uma nova string

```
<script>
        var string1 = "A função concat";
        var string2 = "permite";

        alert(string1.concat(" ").concat(string2).concat(" juntar duas ou mais strings!!!!"));
</script>
```

- Qual o resultado disto?

  A função concat permite juntar duas ou mais strings!!!!

JavaScript – String
Funções

- **indexOf():** devolve a primeira posição ou índice em que ocorre um caracter de uma string

```
<script>
        var frase = "Vamos aprender JavaScript";

        var indice = frase.indexOf("JavaScript");
        alert(indice);

</script>
```

- Qual o resultado disto?

15

Nota: se o texto não for encontrado devolve **-1**

JavaScript – String Funções

- **replace():** substitui um valor numa string por outro valor

```
<script>
        var frase = "Vamos aprender JavaScript";

        var novafrase = frase.replace("Vamos", "Hoje vamos");
        alert(novafrase);

</script>
```

- Qual o resultado disto?

  Hoje vamos aprender JavaScript

- slice(start, end)
- substring(start, end)
- substr(start, lenght)

- slice(): o método **slice** copia até, mas não incluindo o elemento, indicado por fim.

```
<script>
        var string = "Vamos aprender JavaScript";

        var substring = string.slice(6, 14);
        alert(substring);

</script>
```

- Qual o resultado disto?

  aprender

- O **slice()** aceita posições negativas

Nota: Se *start* for negativo, é tratado como ***length + start***, onde *length* é o comprimento da string. Se *end* for negativo, é tratado como ***length + end*** onde *length* é o comprimento da string

- substring(): o método **substring é semelhante ao slice**

```
<script>
        var string = "Vamos aprender JavaScript";

        var substring = string.substring(6, 14);
        alert(substring);

</script>
```

- Qual o resultado disto?

  aprender

- O **substring() não** aceita índices negativos.

JavaScript – String
Funções Split

- split(): permite converter uma string num array, recebe como parâmetro o caracter pelo qual vai efetuar a separação

```
<script>
        var string = "AA,BB,CC,DD";

        var novaString = string.split(',');
        alert(novaString[1]);

</script>
```

- Qual o resultado disto?

BB

- toUpperCase(): permite converter uma string para maiúsculas

```
<script>
        var frase = "Tópicos Especiais I";

        var frase1 = frase.toUpperCase();
        alert(frase1);

</script>
```

- Qual o resultado disto?

TÓPICOS ESPECIAIS I

JavaScript – String
Funções para Extração

- toLowerCase(): permite converter uma string para minúsculas

```
<script>
        var frase = "Tópicos Especiais I";

        var frase1 = frase.toLowerCase();
        alert(frase1);

</script>
```

- Qual o resultado disto?

tópicos especiais i

JavaScript – **Boolean**

- O tipo Boolean representa os valores **true** e **false**

- É útil nos casos em que pretendemos que uma variável somente possa guardar valores como:

  - Sim / Não
  - Ligado / Desligado
  - True / False

- Um array pode ser visto como uma variável especial que é capaz de armazenar mas do que um valor de uma só vez.

Sintaxe

- var nome_array = [elemeto1, elemento2, …];

```
<script>

        var numeros = [10, 12, 14, 16,18, 20 ];
        var nomes = ["Daniel", "Mafalda", "José", "Maria","Oscar Cardozo"];

        for (i = 0; i < numeros.length ; i++)
        {
              alert(numeros[i]);
        }
</script>
```

JavaScript – Array
Propriedades

- **length**: devolve o número de elementos de um array

```
<script>

    var palavras = ["olá", "mundo", "tópicos", "especias", "ISPTEC", "Informática"];
    var palavrasLength =  palavras.length;

    alert( palavrasLength);

</script>
```

- Qual o resultado disto?

  6

JavaScript – Array
Métodos

- Existem métodos que facilitam a tarefa do programador quando se manipulam arrays. Abaixo podemos ver alguns deles:

  - pop()
  - push()
  - shift()
  - unshift()
  - splice()
  - sort()
  - reverse()

JavaScript – Array
Métodos

- **pop():** remove o último elemento de um array e devolve o elemento.

```
<script>
    var palavras = ["olá", "mundo", "tópicos", "especias", "ISPTEC", "Informática"];
    palavras.pop(); // Remove "Informática" do array palavras
</script>
```

- **push():** adiciona um novo elemento no final do array

```
<script>
    var palavras = ["olá", "mundo", "tópicos", "especias", "ISPTEC", "Informática"];
    palavras.push("Programação"); // Adiciona "Programação" ao array palavras
</script>
```

- **shift():** é semelhante ao **pop()** mas este remove o primeiro elemento do array em vez do último.

- **unshift():** é semelhante ao push() com a diferença de que este adiciona o novo elemento ao inicio do array.

  - **Nota:** Esse método não funciona muito no IE 8 e em versões anteriores

- **splice():** permite adicionar ou remover elementos de um array de acordo com os parâmetros passados.
  - Adicionar elemento

```
<script>
    var palavras = ["olá", "mundo", "tópicos", "especias", "ISPTEC", "Informática"];
    palavras.splice(2, 0, "JavaScript");

     for(i = 0; i < palavras.length; i++)
     {
      document.write(palavras[i]);
      document.write("<br/>");
     }
</script>
```

Elemento a ser adicionado

Posição onde vamos adicionar o novo elemento

Nº de elementos que pretendemos que sejam removidos

JavaScript – Array
Métodos

- Remover elemento

```
<script>
    var palavras = ["olá", "mundo", "tópicos", "especias", "ISPTEC", "Informática"];
    palavras.splice(3, 1);

     for(i = 0; i < palavras.length; i++)
     {
      document.write(palavras[i]);
      document.write("<br/>");
     }
</script>
```

Posição do elemento que vamos remover

Nº de elementos que pretendemos que sejam removidos

- Remove o elemento **“especiais”** do array

JavaScript – Array
Métodos – sort()

- **sort():** método que permite a ordenação de um array
  - Por definição ordena os elementos do array como strings ou seja **alfabeticamente** e por ordem **crescente**

```
<script>
    var palavras = ["Maria", "António", "Bruna", "Carla"];
    palavras.sort();   // O array passa a ser ["António", "Bruna", "Carla", "Maria"]
</script>
```

  - Para ordenar por ordem decrescente utilizamos o método **reverse()** que inverte um vetor;

```
<script>
    var palavras = ["Maria", "António", "Bruna", "Carla"];

    palavras.sort();   // O array passa a ser ["António", "Bruna", "Carla", "Maria"]

    palavras.reverse(); // O array passa a ser ["Maria", "Carla", "Bruna", "António"]
</script>
```

JavaScript – Array
Métodos – sort()

- E se tivermos um array de números o que acontece ao chamarmos **array.sort()**

```
<script>
    var myArray = [7, 40, 300];

    myArray.sort();

</script>
```

- Qual o resultado ?
  - ~~[300, 40, 7]~~

JavaScript – Array
Métodos – sort()

- Para fazer a ordenação de um vetor com números o método **sort()** têm de receber um parâmetro que vai ser **uma função de comparação**

  - **function (a, b){ return a-b } // compara "a" e "b" e devolve -1 , 0, ou 1**

    - **Menor que 0**: ordena de modo que o índice "a" seja menor que o de "b"
      - Ordem crescente
    - **Igual a 0**: "a" e "b" são considerados iguais por isso não é realizada a ordenação
    - **Maior que 0:** ordena de modo que o índice de "b" seja maior que o do "a"
      - Ordem decrescente

```
<script>
    var myArray = [25, 8, 7, 41];
    myArray.sort(function(a, b) { return a - b });
</script>
```

- Qual o resultado ?
  - [7, 8, 25, 41]

JavaScript – Função

- É um bloco de instruções ao qual é atribuído um nome, que pode ser invocado a partir desse.

- Pode receber parâmetros e devolver um valor.

- Com as funções podemos reutilizar código ou seja definir a função uma vez e utiliza-la quantas vezes quisermos

- Evitar ter-se código repetido.

- Sintaxe

```
<script>
    function  nome_da_funcao(param1, param2, ..., paramN){
            Implememementação
    }
</script>
```

```
function olaMundo(){

    alert("Ola Mundo");
}
```

- Para função devolver valores é necessário utilizar o **return**

```
<script>

    alert(soma(2, 4));

    function soma(num1, num2){

        return num1 + num2;
    }


</script>
```

- As funções são invocadas como resposta por exemplo:
  - Ocorrência de um evento tal como o clique de um botão

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title></title>JavaScript</title>
        <meta charset="utf-8">
    </head>
    <body>

    <p id="paragrafo">Eu sou um paragrafo por definição<p>
     <input type="submit" onclick="mudaTextoParagrafo()" value="Clica-me"/>

    <script>

        function mudaTextoParagrafo(){

            // Vamos afetar propriedade color do elemento com id = paragrafo
            document.getElementById("paragrafo").style.color = "red";
             // Vamos afetar propriedade font-size do elemento com id = paragrafo
            document.getElementById("paragrafo").style.fontSize = "25px";

            // Vamos mudar conteudo do elemento cujo id = paragrafo
            document.getElementById("paragrafo").innerHTML = "O meu texto mudou";
        }

       </script>

    </body>
</html>
```

- Funcionam tal com em C, JAVA, C#

If

```
if (cond){

      instrução ou bloco de Instruções;
}
```

If-else

```
if (cond){

      instrução ou bloco de Instruções;
}
else{
      instrução ou bloco de instruçoes;
}
```

```
<body>
    <script>
            if (1)
            {
                alert('A condição if é verdadeira');
            }
            else{
                 alert('Executou a instrução do else');
            }
     </script>
</body>
```

```
var corCarro = prompt('Qual a cor do carro que pretende');

switch (corCarro) {
  case 'vermelho':
    alert('Vermelho é uma cor espectacular!');
    break;
  case 'preto':
    alert('Preto é bom, mas têm de se lavar o carro constantemente!');
    break;
  case 'branco':
    alert('Branco está com desconto!');
    break;
  default:
   alert('A cor:' + corCarro + ' não existe');
    break;
};
```

- Sintaxe

```
while(cond)
    instrução
```

```
while(cond){
    instruções ou bloco de instruções;
}
```

```
<script>

    var x = 10;

    while(x  >  0) {
        x--;
        alert("O valor de x é " + x);
    }

</script>
```

- Sintaxe

```
do
  instrução;
while(cond);
```

```
do {
  instrução ou bloco de instruções;
}while(cond);
```

```
<script>

    var x = 10;

        do
        {
            x--;
            alert("O valor de x é: " + x);
        }
        while(x  >  0);

</script>
```

- Sintaxe

```
for
  instrução;
```

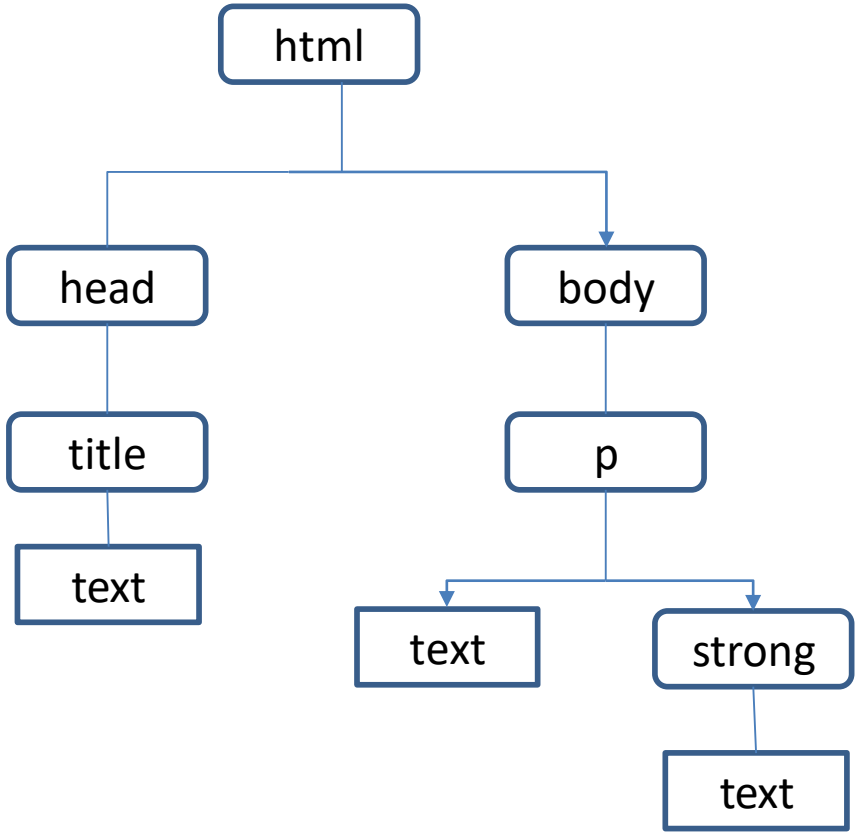
```
for(inicialização; condição; ação){
    instrunção ou bloco de instruções;
}
```

```
<script>

    for (var i = 1; i < 10; i++) {
        alert(i);
    }
</script>
```

- DOM (document object model): pode ser visto como a forma que os navegadores(browsers) visualizam o documento HTML.
- Após carregar o documento o navegador(browser) cria uma estrutura de árvore na memória que vai permitir manipula-lo a vontade.
- O JavaScript assim como outras linguagens, possui uma serie de funções que vão pemitir criar, remover, alterar os elementos da árvore.
  - getElementById()
  - getElementsByTagName()
  - getElementsByName()

```
<html>
    <head>
        <title>Manipulação do DOM</title>
    </head>
    <body>
        <p>
            Este paragráfo contém um exemplo de
            <strong>texto em negrito</strong>
        </p>
    <body>
</html>
```



Dúvidas