

CTeSP PSI 2018/19

Introdução às Linguagens e Tecnologias Web

Javascript

Pedro Colarejo

pcolarejo@ua.pt

Escola Superior de Tecnologia e Gestão de Águeda

Universidade de Aveiro

2019/20



SUMÁRIO

IV. Javascript

- Introdução
 - Localização
 - *Output*
 - Sintaxe
 - Variáveis
 - Operadores
 - Tipos de dados
 - Conversão
 - Funções
 - *Scope*
 - Eventos
 - Objetos
 - Array, String, Date, Math
 - Instruções condicionais, instruções de repetição
 - Validação de dados em formulários
-

JAVASCRIPT

Javascript?

- Linguagem de programação que permite adicionar interatividade em páginas HTML
- Uma linguagem interpretada (sem necessidade de compilação)
- Pode ser utilizada livremente (não é necessário licenciamento)
- Javascript e Java são duas linguagens completamente distintas
- Permite trabalhar com objetos

- Obs: As páginas Web não são o único contexto onde o JavaScript é usado. Muitos programas de desktop e servidor usam JavaScript. O Node.js é o mais conhecido. Algumas bases de dados, como o MongoDB e o CouchDB, também usam JavaScript como linguagem de programação.

JAVASCRIPT

O que posso fazer com Javascript?

- Mudar conteúdos, propriedades, estilos, esconder, mostrar elementos HTML dinamicamente (interatividade)
- Monitorizar e responder a eventos do utilizador
- Validar dados de formulários antes de serem enviados para o servidor
- Criar e manipular cookies HTTP
- etc.

Alguns exemplos:

- https://www.w3schools.com/js/tryit.asp?filename=tryjs_intro_style
- https://www.w3schools.com/js/tryit.asp?filename=tryjs_intro_hide

JAVASCRIPT

- A construção das páginas deverá ser efetuada de forma a que um dado utilizador consiga visualizar o seu conteúdo mesmo que o Javascript seja desativado no browser
- Por exemplo, ao utilizar-se apenas Javascript para efetuar validações de dados de formulário do lado do cliente, deve ter-se em conta o facto de o Javascript poder ser manipulado/desativado, e dessa forma facilmente ultrapassadas as verificações e validações impostas pelo mesmo

JAVASCRIPT

Localização (3 alternativas)

```
<!DOCTYPE html>
<html>
<head>
    <title>Exemplo</title>
    <script>
        //Código JavaScript
    </script>
    <script src="http://remote.server.com/jsExterno.js" />
</head>
<body>
    <script>
        //Código JavaScript
    </script>
</body>
</html>
```



JAVASCRIPT

Localização (3 alternativas)

- secção <head>
 - Scripts executados somente quando são invocados, ou quando um dado evento o ativa.
https://www.w3schools.com/js/tryit.asp?filename=tryjs_where_to_head
- secção <body>
 - Scripts executados quando a página é interpretada/gerada (deve colocar-se no final do body por questões de desempenho)
https://www.w3schools.com/js/tryit.asp?filename=tryjs_where_to_body
- Ficheiro externo
 - Guardar o script num ficheiro com a extensão “.js”, sem as tags <script> -
https://www.w3schools.com/js/tryit.asp?filename=tryjs_external
- Pode ser colocado nas três localizações simultaneamente!

JAVASCRIPT

- Vantagens da utilização de ficheiros externos
 - Reutilização: se pretendemos correr o mesmo script em várias páginas, podemos colocá-lo num ficheiro externo, evitando ter de escrever o mesmo script em cada uma das páginas
 - Manutenção: separa HTML do código
 - Cache
-

OUTPUT

1) document.getElementById("id").innerHTML

```
<body>
```

```
    <h1>My First Web Page</h1>
```

```
    <p>My First Paragraph.</p>
```

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

```
<script>
```

```
    document.getElementById("demo").innerHTML = 5 + 6;
```

```
</script>
```

```
</body>
```

My First Web Page

My First Paragraph.

11

JAVASCRIPT

2) document.write()- conveniente utilizar apenas para teste (apaga todo o html existente)

```
<body>
```

```
<h1>My First Web Page</h1>
```

```
<p>My first paragraph.</p>
```

```
<script>
```

```
    document.write(5 + 6);
```

```
</script>
```

```
</body>
```

JAVASCRIPT

3) `window.alert()` – uso de uma caixa de alerta

```
<body>
```

```
<h1>My First Web Page</h1>
```

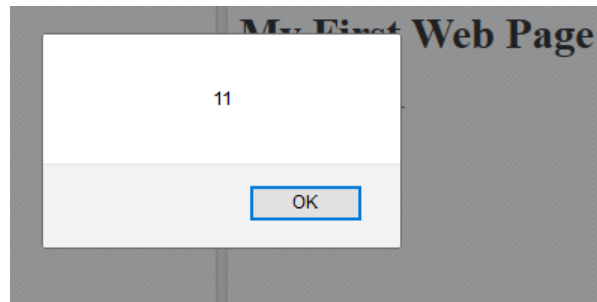
```
<p>My first paragraph.</p>
```

```
<script>
```

```
window.alert(5 + 6);
```

```
</script>
```

```
</body>
```



JAVASCRIPT

4) **console.log()** - Para fins de *debugging*, utiliza-se este método para apresentar dados.

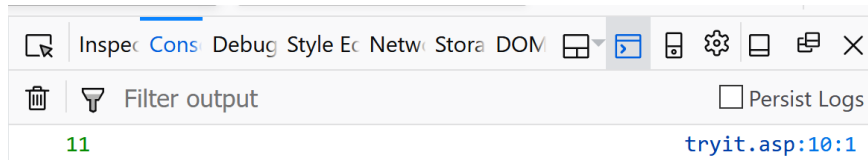
```
<body>
```

```
<h2>Activate debugging with F12</h2>
```

```
<p>Select "Console" in the debugger menu. Then click  
Run again.</p>
```

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

```
</body>
```



SINTAXE

- A linguagem Javascript é *case-sensitive*
- Cada instrução deve ser terminada com ;
`document.getElementById("demo").innerHTML = "Hello Dolly.";`
- As instruções são executadas pelo *browser* de forma sequencial
- As instruções podem ser agrupadas em blocos utilizando { e }
- Podem ser inseridos comentários utilizando // ou /* e */

SINTAXE

Linguagem *weakly typed*

- não é obrigatória a declaração de variáveis
- não é possível especificar o tipo de variável
- os tipos de dados podem ser convertidos automaticamente durante a execução

```
var answer = 42;  
x=5;  
var x=10, n;  
var x;  
answer = "Thanks for all the fish...";
```

- mais fácil de utilizar mas mais suscetível a erros de programação

VARIÁVEIS

- Variável – usadas para guardar valores de dados
- *Case-sensitive* - `mVariavel` é diferente de `mvariavel`
- Ter atenção palavras reservadas (por exemplo `var`)
- nomes com acentos não devem ser utilizados
- Nomes têm que começar com: letra, "_" ou "\$"
`var nome, $idade;`
- Podem conter letras, números, _ e \$
- Ser consistente nos nomes e na forma!
 - por exemplo, *lower camel case*, nomes com significado, prefixos,... `firstName`, `lastName`, `masterCard`, `interCity`.



VARIÁVEIS

- Declaração

```
var minhaVariavel;
```

- Atribuição

```
minhaVariavel = 30;  
var outraVariavel = "Olá";  
minhaVariavel = outraVariavel;
```

- Acesso a variáveis

- Valor de uma variável não inicializada

```
undefined
```

OPERADORES

Operadores aritméticos

Operator	Description	Example
+	Addition	$x = y + 2$
-	Subtraction	$x = y - 2$
*	Multiplication	$x = y * 2$
/	Division	$x = y / 2$
%	Modulus (division remainder)	$x = y \% 2$
++	Increment	$x = ++y$
--	Decrement	$x = --y$

Operadores de atribuição

Operator	Example	Same As
=	$x = y$	
+=	$x += y$	$x = x + y$
-=	$x -= y$	$x = x - y$
*=	$x *= y$	$x = x * y$
/=	$x /= y$	$x = x / y$
%=	$x \% = y$	$x = x \% y$

OPERADORES

Operadores lógicos

Operator	Description
&&	and
	or
!	not

Operadores relacionais

Operator	Description
==	is equal to
===	is exactly equal to (value and type)
!=	is not equal
>	is greater than
<	is less than
>=	is greater than or equal to
<=	is less than or equal to

OPERADORES

Tabela de verdade				
X	Y	X && Y	X Y	!x
V	V	V	V	F
V	F	F	V	F
F	V	F	V	V
F	F	F	F	V

TIPOS DE DADOS

- **boolean**: valores lógicos true e false

```
var x = true;  
var y = false;
```

- **string**: sequências de caracteres; utilizam-se `""` ou `' '` para representar *strings*

```
var carName = "Volvo XC60";    // Using double quotes  
var carName = 'Volvo XC60';    // Using single quotes
```

- **number**: inteiros, reais

```
var x1 = 34.00;                // with decimals  
Written                        without decimals  
var x2 = 34;                   //  
Written
```

- **array**

- **object**

TIPOS DE DADOS

Qual o tipo de dados?

Para saber o tipo de dados armazenado numa variável:

```
typeof(variavel); // retorna o tipo de dados armazenado
```

Resultados possíveis:

boolean

string

number

object

function

undefined

JAVASCRIPT

Conversão de dados

- `parseInt()`
 - Conversão de *string* para número inteiro
`a = parseInt(b)`
- `parseFloat()`
 - Conversão de *string* para número real
`c = parseFloat(d);`
- `String()`
 - Conversão de número, ou objeto, para *string*
`e = String(f);`
- Nem sempre é possível converter para número:
 - `var num = parseInt("teste");` => NaN (Not a Number)
 - `var num = parseInt("a123");` => NaN

JAVASCRIPT

Conversão implícita

- Em expressões que envolvem um valor numérico e uma *string*, o valor numérico é tratado como *string*

```
var t = "A minha idade: " + 21; => ?
```

```
var soma = "12" + "34"; => ?
```

- O JavaScript avalia as expressões da esquerda para a direita.
Diferentes sequências podem produzir resultados diferentes:

```
var bebidas = 1 + 4 + "cafés"; => ?
```

```
var bebidas = "cafés" + 1 + 4; => ?
```

FUNÇÕES

- Bloco de código desenvolvido para realizar uma dada tarefa
- Executadas por um evento ou quando for invocada uma função
- Podem retornar valores, utilizando a instrução "return"

```
function nomeFunção (param1,  
param2,...)  
{  
    // código a executar  
    [return valor;]  
}
```

Exemplo:

```
<body>  
<script>  
function myAdicionar(a, b) {  
    return a + b;  
}  
    document.write(myAdicionar(5,  
    5));  
</script>  
  
</body>
```


JAVASCRIPT

Scope – Visibilidade/Acessibilidade das variáveis/objetos

- **Variável global**

- pode ser utilizada em qualquer ponto do script
- é eliminada apenas quando a página é fechada
- a não declaração de uma variável (independentemente do local) faz com que esta seja global

- **Variável local**

- existe apenas na estrutura em que foi declarada
- podem existir variáveis locais com o mesmo nome em estruturas diferentes
- é eliminada assim que a estrutura é concluída

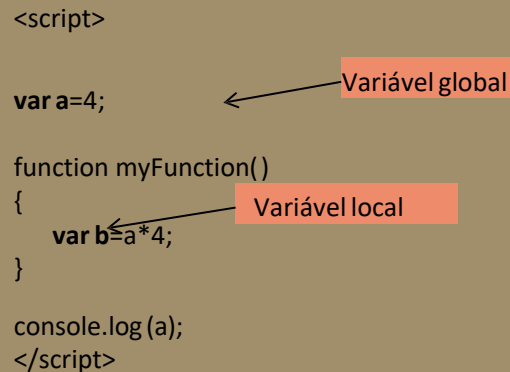
```
<script>

var a=4;

function myFunction()
{
    var b=a*4;

}

console.log(a);
</script>
```



JAVASCRIPT

Exemplo 1:

```
function square(num) {  
    total = num * num;  
}  
var total = 50;  
square(20);  
console.log(num); //??
```

Exemplo 2:

```
function square(num) {  
    var total = num * num;  
}  
square(20);  
console.log(total); //??
```



JAVASCRIPT

Exemplo 3:

```
function helloPeople(name)
{
    console.log(name); //??
}
```

```
helloPeople("Manel");
```

```
console.log(name); //??
```

JAVASCRIPT

Exemplo4:

```
function helloPeople(name)
{
    myName = name;
    console.log(name); //??
}
```

```
helloPeople("Manel");
console.log(myName); //??
```



JAVASCRIPT

Exemplo5:

```
var myName = "Maria";

function helloPeople(name)
{
    myName = name;
    console.log(name); //??
}

console.log(myName); //??

helloPeople("Manel");

console.log(myName); //??
```

SCOPE

Exemplo6:

```
function incValores(a, b)
{
    a++;
    b++;
    console.log(a + " " + b); //??
}

var a = 4;
var b = 8;

console.log(a + " " + b); //??

incValores(a, b);

console.log(a + " " + b); //??
```

INSTRUÇÃO CONDICIONAL IF ... ELSE

```
if ( condição ) {
    instruções se condição verdadeira
}
```

```
if ( condição ) {
    instruções se condição verdadeira
}
else {
    instruções se condição falsa
}
```

```
if ( condição1 ) {
    instruções se condição1 verdadeira
}
else {
    if (condição2)
        instruções se condição2 verdadeira else
        instruções se condição2 falsa
}
```

Exemplo:

Página Web que pede ao utilizador um número, mostrando uma mensagem com o valor do seu quadrado no caso em que o input é mesmo um valor numérico:

```
<h1>Exemplo if</h1>
  Introduza um número:
  <input type="text" id="myText">
  <button type="button"
onclick="myFunction()">Verificar</button>
<script >
function myFunction() {
var n = document.getElementById("myText").value;
if ( isNaN(n) )
    document.write("Número inválido!");
else
document.write("O quadrado do n° = "+n*n);
}
</script>
```

INSTRUÇÃO CONDICIONAL IF ... ELSE

Exercício:

- Escreva uma página Web que mostre uma mensagem de boas vindas consoante a hora do dia:
 - 6 – 12: "Bom dia"
 - 13 – 19: "Boa tarde"
 - 20 – 24, 1-5: "Boa noite"
- Para obter a hora do dia pode utilizar a seguinte instrução:
`Date().getHours();`

INSTRUÇÃO CONDICIONAL SWITCH

```
switch(n)
{
case 1:
    executa bloco código 1;
    break;
case 2:
    executa bloco código 2;
    break; default:
    outros casos;
}
```

Exemplo:

Página Web que pede um número correspondente ao mês (1,...,12), mostrando uma mensagem com o respetivo nome (janeiro,...,dezembro). Se for introduzido um número inválido mostra a mensagem "Mês inválido!".

```
<script>
    var n = Number(window.prompt("Mês?", ""));
    switch (n) {
        case 1:
            document.write("janeiro");
            break;
        ...
        case 12:
            document.write("dezembro");
            break;
        default:
            document.write("Número
inválido!");
    }
</script>
```

INSTRUÇÃO REPETIÇÃO FOR, WHILE E DO...WHILE

```
for (var=start;var<=end;var=var+inc)
{
    código a ser executado
}
```

```
while (var<=end)
{
    código a ser executado
}
```

```
do
{
    código a ser executado
} while (var<=end)
```

Possível utilizar as instruções break e continue nos ciclos de repetição!

Exemplo:

Página HTML que mostra mensagens com os diferentes níveis de cabeçalho (de h1 a h6):

```
<html>
<head><title>Exemplo Javascript</title></head>
<body>
<script>
    for (i = 1; i <= 6; i++) {
        document.write("<h" + i + ">Cabeçalho nº " + i);
        document.write("</h" + i + ">");
    }
</script>
</body>
</html>
```

EVENTOS

- Os eventos podem ocorrer por ações do utilizador ou automaticamente
 - cursor, teclado, formulário, *browser*...
- Exemplos:
 - Uma página da Web HTML terminou de carregar
 - Um campo de entrada HTML foi alterado
 - Um botão HTML foi clicado

EVENTOS

- Grande parte das vezes, quando acontecem eventos, queremos depois fazer alguma coisa.

```
<some-HTML-element some-event="some JavaScript">
```

```
<button onclick="displayDate()">The time is?</button>
```

```
<script>  
function displayDate() {  
    document.getElementById("demo").innerHTML = Date();  
}  
</script>
```

- Eventos declarados fora dos atributos HTML (boa prática)

```
<script>  
    document.getElementById("myBtn").onclick = function() {fazSoma()};  
</script>
```

EVENTOS

- Principais eventos:
 - onload, onunload – quando o utilizador entra ou sai da página
 - onclick – quando o utilizador clica sobre determinado elemento HTML
 - onfocus, onblur, onchange – para validações em campos de formulários
 - onsubmit – para validação de campos de formulários
 - onmouseover, onmouseout – por exemplo, para criar botões animados



JavaScript Reference HTML DOM Events

http://www.w3schools.com/jsref/dom_obj_event.asp

OBJETOS

- Todos os valores JavaScript, exceto os primitivos, são objetos.
 - Tipos primitivos:
 - string
 - number
 - boolean
 - null
 - Undefined
 - **Evitar declarar Strings, Numbers, e Booleans como objetos** (os tipos primitivos são mais rápidos)
-

OBJETOS NATIVOS DO JAVASCRIPT

- Object – objeto a partir do qual todos os restantes objetos do Javascript (nativos ou definidos pelo programador) são implementados:
http://www.w3schools.com/js/js_objects.asp
- Existe um conjunto de objetos (com propriedades e métodos) nativos do Javascript que podemos utilizar
- Os tipos primitivos de dados (numéricos, booleanos, strings,, ...) podem ser tratados como objetos:
 - Number (http://www.w3schools.com/js/js_obj_number.asp)
 - Boolean (http://www.w3schools.com/js/js_obj_boolean.asp)
 - String (http://www.w3schools.com/js/js_obj_string.asp)
- Outros objetos nativos do Javascript:
 - Math (http://www.w3schools.com/js/js_obj_math.asp) (difere dos outros pois é não instanciável)
 - Date (http://www.w3schools.com/js/js_obj_date.asp)
 - Array (http://www.w3schools.com/js/js_obj_array.asp)

ARRAYS

- Utilizados para guardar múltiplos valores numa variável
- Declaração de um array com elementos iniciais:

```
var arrayLetras = new Array("a", "b", "c"); (não é necessário o new) var  
arrayNumeros = [5, 23, 13, 12];  
var arrayMisto =[23, "Joana Tavares", false];
```

"a"	5
"b"	23
"c"	13
	12

ARRAYS

- `nomeArray[índice] = valor;`
- Operação de escrita:
`arrayLetras[1] ="x";`
`arrayLetras[3] ="d";`
`arrayLetras[5] ="z";`
- Operação de leitura:
`var letraEscolhida = arrayLetras[1];`
`document.write(letraEscolhida);`

"a"
"x"
"c"

"a"
"x"
"c"
"d"

"a"
"x"
"c"
"d"
undefined
"z"

ARRAYS

Propriedades

```
var fruits = ["Banana", "Orange", "Apple", "Mango"];  
var x= fruits.length;           // the length of fruits is 4
```



Javascript Array Object

http://www.w3schools.com/js/js_obj_array.asp

ARRAYS

Métodos

Método	Descrição
<code>concat()</code>	Joins two or more arrays, and returns a copy of the joined arrays
<code>join()</code>	Joins all elements of an array into a string
<code>pop()</code>	Removes the last element of an array, and returns that element
<code>push()</code>	Adds new elements to the end of an array, and returns the new length
<code>reverse()</code>	Reverses the order of the elements in an array
<code>shift()</code>	Removes the first element of an array, and returns that element
<code>slice()</code>	Selects a part of an array, and returns the new array
<code>sort()</code>	Sorts the elements of an array
<code>splice</code>	Adds/Removes elements from an array
<code>toString()</code>	Converts an array to a string, and returns the result
<code>unshift()</code>	Adds new elements to the beginning of an array, and returns the new length



STRINGS - PROPRIEDADES E MÉTODOS

```
var txt = "ABCDEFGHIIJKLMNOPQRSTUVWXYZ";
```

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

```
var y = "We are the so-called \"Vikings\" from the north."
```



Javascript String Object

http://www.w3schools.com/js/js_obj_string.asp

STRINGS - PROPRIEDADES E MÉTODOS

Method	Description
<code>charAt(index)</code>	Returns the character at the specified index.
<code>charCodeAt(index)</code>	Returns the Unicode value of the character at the specified index.
<code>concat(str1, str2, ...)</code>	Joins two or more strings, and returns a copy of the joined strings.
<code>fromCharCode()</code>	Converts Unicode values to actual characters.
<code>indexOf(subString)</code>	Returns the position of the first occurrence of a specified <code>subString</code> value. Returns -1 if the substring is not found.
<code>lastIndexOf(subString)</code>	Returns the position of the last occurrence of a specified <code>subString</code> value. Returns -1 if the substring is not found.
<code>match(regex)</code>	Searches the string and returns all matches to the regular expression.
<code>replace(subString/regex, replacementString)</code>	Searches the string for a match of the substring or regular expression and replaces the matched substring with a new substring.



DATE

- O objecto Date permite-nos trabalhar com datas (anos, meses, horas, etc.)

Fri Apr 21 2017 17:20:56 GMT+0100

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

```
<script>
```

```
document.getElementById("demo").innerHTML = Date();
```

```
</script>
```

- Vários métodos associados ao objecto Date: `getDay()`, `getDate()`;



Javascript Date Object

https://www.w3schools.com/jsref/jsref_obj_date.asp

JAVASCRIPT

- Math
 - O objecto Math permite efetuar operações matemáticas
 - sobre números
 - Várias propriedades contendo valores de constantes (PI, ...)
 - Vários métodos associados ao objecto Math:

- Javascript Math Object https://www.w3schools.com/js/js_math.asp

VALIDAÇÃO DE DADOS DE FORMULÁRIOS

- 3 formas de validar os dados de formulários:
 - Validadores do HTML 5
 - Novos tipos de input: email, number, url, ...
 - Novos atributos: required, placeholder, pattern, min, max, step, ...
 - Funções Javascript que processam os dados
 - Associar uma função Javascript a um evento que acontece num elemento HTML do formulário (p. ex., value changed, got focus, hit submit, etc.)
 - Dados enviados para o servidor para validação



JavaScript Form Validation

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

VALIDAÇÃO DE DADOS DE FORMULÁRIOS

- Validação utilizando o evento onsubmit:

`<form ... onsubmit="return validateForm()">`

- Se a função especificada no evento onsubmit devolver:
 - true, então os dados são enviados
 - false, então o formulário não é processado e os dados não são enviados

- Validação utilizando outros eventos associados aos elementos do formulário:

onclick, onfocus, onmouseover, onmouseout, onchange, ...

VALIDAÇÃO DE DADOS DE FORMULÁRIOS

```
function validateForm() {  
  var x = document.forms["myForm"]["fname"].value; if (x  
    == "") {  
      alert("Name must be filled out");  
      return false;  
    }  
}
```

</script>

</head>

<body>

<form name="myForm" action="/action_page_post.php" onsubmit="return validateForm()" method="post">

Name: <input type="text" name="fname">

<input type="submit" value="Submit">

</form>

BIBLIOGRAFIA

- Abreu, Luís – **Javascript 6**. FCA, 2015. ISBN: 978-972-722-815-7
- Javascript Basics - <https://www.udacity.com/courses/ud804>
- Interactivity with Javascript
<https://www.coursera.org/learn/javascript/>
- Object-Oriented JavaScript -
<https://www.udacity.com/courses/ud015>