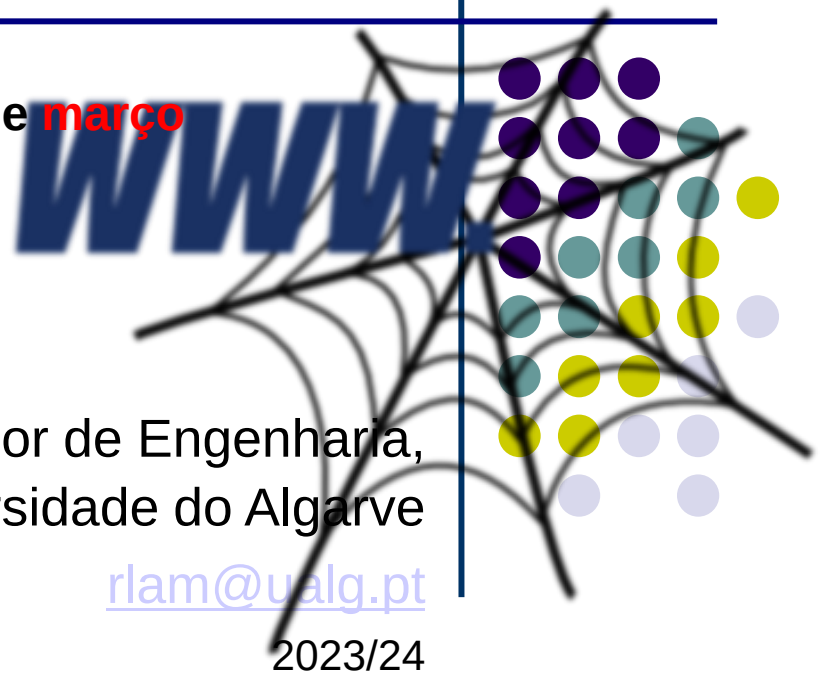
UAlg ISE
UNIVERSIDADE DO ALGARVE
INSTITUTO SUPERIOR DE ENGENHARIA

Tecnologias Web

5ª AULA, 11 de março



Roberto Lam, Instituto Superior de Engenharia,
Universidade do Algarve
rlam@ualg.pt

2023/24

JavaScript

Diversos

Sensibilidade ao tipo de letra

; no termino da instrução ou expressão

Comentários

Uso do espaço e mudança de linha

Palavras reservadas

```
break       for         throw
case        function    try
catch       if          typeof
continue    in          var
default     instanceof  void
delete      new         while
do          return      with
else        switch
finally     this
```

JavaScript

Sintaxe básica

Frases:

Expressões (produzem valores)

```
var aux= soma + resultado;
```

Instruções (não produzem resultados, produzem efeitos)

```
if(aux > 10){
        aux= soma;
}else
        aux= soma + resultado;
```

JavaScript

Sintaxe básica

Variáveis e tipos de dados:

Variáveis

```
var tipoNumber= 23;
tipoString="Olá Mundo";
var tipoBoolean=true;
var tipoArray=[tipoNumber, tipoString, tipoBoolean];
```

Obs:

```
A=tipoArray[0]; // A contem valor 23
B=tipoArray[1]; // B contem "Olá Mundo"
var num1=2, num3=10, num3=45;
```

JavaScript

Sintaxe básica

Tipos de dados:

```
typeof tipoNumber; //devolve number

typeof tipoString; // devolve string

typeof tipoBoolean; // devolve boolean

typeof tipoArray; // devolve object (coleção de dados)
```

Obs:

```
var A="ops"; // A contem valor "ops"
A=2451;      // A 2451
A=false;     // A contém valor false
A=null;      // ausência de dados

typeof A;
```

JavaScript

Comparação

Operador igualdade , não igualdade (==, !=)

Operador identidade, não identidade (===, !==)

```
tipoNumber=42;

tipoNumber==42;          // devolve true

tipoNumber==“42”;        // ??

tipoNumber===“42”        // ??
```

Obs:

```
var MeuArrayA=[1,2 3];
var MeuArrayB=[1,2 3];
MeuArrayA == MeuArrayB   // false
MeuArrayA === MeuArrayB  // false
MeuArrayB == MeuArrayB   // true
```

http://www.c-point.com/javascript_tutorial/jsgrpComparison.htm

JavaScript

Verdadeiro ou falso

Variáveis com valores; zero, null, undefined e string vazia são considerados false.

Obs:

```
aux=null;
if(aux){
        A=0;
}else
        A="segundo";
```

JavaScript

Mecanismos de controlo

Estruturas de decisão

if/ if else /else

```
if (expression) {
        console.log("expression é verdadeira");
} else
        if (expression2) {
                console.log("expression2 é verdadeira ");
        } else
                if (expression3) {
                        console.log("expression3 é verdadeira ");
                } else {
                        console.log(“todas expressions são falsas");
                }
```

JavaScript

switch/ case

```
switch (letra) {
        case "a":
        case "e":
        case "i":
        case "o":
        case "u":
          console.log(“letra é uma vogal");
          break;
        case "y":
          console.log(“a letra poderá ser uma vogal");
          break;
        default:
          console.log(“Não é vogal");
          break;
}
```

JavaScript

Ciclos

for (..; ..; ..)

```
for (var i = 0; i < 10; i++) {
  console.log("ciclo numero " + i);
}
```

while (..)

```
var times = 10;
while (times--) {
  console.log("viva!");
}
```

do{.. }while(..)

```
var times = 10;
do {
  console.log("viva !");
} while (times--)
```

JavaScript

Operadores lógicos

Negação lógica: **!**

```
if (! myVar) {
  // continua false
}
```

Conjunção lógica AND: **&&**

```
var truthyValue = 1;
var falseyValue = null;
// Left operand && right operand
true && truthyValue; // qual é o resultado??
falseyValue && truthyValue; // ???
```

JavaScript

Disjunção lógica OR: ||

```
falseyValue || truthyValue; // segundo valor => 1
truthyValue || falseyValue; // primeiro valor => 1
false || falseyValue || anotherFalseyValue || truthyValue;
// último é verdadeiro => 1
```

Combinação de operadores lógicos||

```
var element = document.body.firstChild;
if (element &&  (element.nodeName === "DIV" ||
  element.nodeName === "SPAN") &&   element.childNodes.length) {
    alert(“Todas expressões são verdadeiras");
}
```

JavaScript

Operações com números

Constantes e funções

- isNaN(x) Devolve true se x não é numero ou de passível conversão.
- Number.MAX_VALUE Maior numero que pode se expresso no JavaScript.
- Number.MIN_VALUE Menor numero que pode se expresso no JavaScript.
- Infinity Variable global que representa infinity. Resultado da divisão por zero.

JavaScript

Operações com números

Objecto **Math**

- Math.PI Valor aproximado de pi
- Math.abs(x) Valor absoluto de x
- Math.ceil(x) Proximo inteiro maior que x
- Math.floor(x) Menor inteiro menor do que x
- Math.max(x, y, …) Maior dos argumentos
- Math.min(x, y, …) Menor dos argumentos
- Math.random() Numero pseudoaleatório entre 0 e 1
- Math.round(x) Inteiro proximo a x
- Math.sqrt(x) Raiz quadrada d x

JavaScript

Conversão de números

parseInt()

```
var a=parseInt(“5.1”,10);
// a recebe 5
```

parseFloat()

```
var a=parseFloat(“1.5”,10);
// a recebe 1.5
```

toFixed()

```
var interest = 100 * 2.95 / 12;
interest.toFixed(0);
"24"
interest.toFixed(2);
"24.58"
```

JavaScript

Operações com strings

Caracteres *escape:* **\”**

```
"She said \"hello!\"";
'They say "hello!"';
```

Operador de concatenação: **+**

```
var a="hello" + " world";
// a tem "hello world"
var b="2" + "2";
// b tem "22"
```

Operadores de comparação: **<**, **>**

```
"Alpha" < "Zeta";
// true
"Zeta" > "alpha";
// false
```

JavaScript

Atributo: **length**

```
var a="abcdefghijklmnopqrstuvwxyz".length;
// a tem 26
```

Metodos: toUpperCase(), toLowerCase()

toUpperCase(), toLowerCase()
charAt(), charCodeAt()
slice(x[, y])
substr(x[, y])
split([delimiter, limit]) // divide string em substrings
indexOf(substring[, start]) // devolve a 1ª posição
lastIndexOf(substring[, start])

JavaScript

Funções e objectos

```
function functionName(arg1, arg2, argN) {
    // Block of statements
}
```

```
function dizOla(nome) {
    document.write("Ola, " + nome + "!");
    // console.log("Ola, " + nome + "!");
}
```

argumentos

Nome da função

Firebug, addon Firefox

JavaScript

Argumentos de funções

```
function myFuncao(arg0, arg1, arg2) {
        console.log(arg0);
        console.log(arg1);
        console.log(arg2);
}
```

Obs:

```
myFuncao("a", "b"); // chamada sem 3°argumento

//mostra Firebug (Fbg) "a", "b" e undefined
```

```
function myFuncao(arg0){ //se evocarmos c/ + N args
        console.log(arg0);
        console.log(argument[1]);
        console.log(argument[N]);
}
```

Obs:

```
myFuncao("a", "b", "c"); // chamada com mais 2 argumentos

//mostra no Fbg "a", "b" e "c"
```

JavaScript


```
function dizOla(nome) {
        aux=nome || "joao";
        console.log(aux)
}
```

Obs:

```
DizOla(); // chamada sem argumento, mostra Fbg joao

DizOla("Maria"); // chamada sem argumento, mostra Fbg Maria
```

Retorno dados funções

```
function Vezes(arg0, arg1){ //
        console.log(arg0*arg1);
        return(arg0*arg1);
}
```

Obs:

```
Vezes(23,2); // chamada com mais 2 argumentos

//a função devolve Fbg 26
```

JavaScript

Recursividade nas funções

```
function Factorial(arg0){ //
        if(arg0==1)
            return 1;
        else
            return(arg0*Factorial(arg0-1));
}
```

Obs:

```
Factorial(6); // chamada

//a função devolve Fbg 720
```

```
function Factorial(arg0){
var aux=1;
     for(i=1; i<= arg0; i++){
          aux=aux*I;
     }

return(aux);
}
```

JavaScript

Objectos = { atributos, propriedades}

```
var Objecto1 = {};
var objecto2 = {
    Atributo1:1980,
    Atributo2:"Março",
    "Strings podem": 123,
    12:"podem",
};
```

Obs:

```
console.log(objecto2["Strings podem"]);
Objecto2[Strings podem"]=15;
Objecto2[12]=" é confuso não?";
console.log(objecto2[12]);
console.log(objecto2["Strings podem"]);
console.log(objecto2.Atributo1);
```

JavaScript

Objectos = { atributos, propriedades}

```
var Pessoa = function(name){
    this.nome=name;
};
Pessoa.prototype.diz = function(message) {
      console.log(this.nome + " diz " + message);
};
Pessoa.prototype.mudaNome=function(novoNome) {
      this.nome=novoNome;
};
```

Utilização:

```
var Eu=new Pessoa("Roberto Lam"; // criação do objecto Eu
Eu.diz("olá");
Eu.mudaNome("rlam");
Eu.diz("olá");
```

JavaScript

Variáveis globais

```
var myVar = "global variable";

function myFunc() {
     var myVar = "scoped variable";
     console.log(myVar);
     console.log(window.myVar);
}

myFunc();

//"scoped variable"
//"global variable"
```

JavaScript

Funções globais

Timers

setTimeout(func, delay), //devolve um id para usar com
//ClearTimeout
clearTimeout();
setInterval(func, delay)

Obs:

```
function callback() {
 console.log("At least three seconds passed");
}

setTimeout(callback, 3000); // executa callback() única vez
setInterval(callback, 3000); // executa callback de 3 em 3 s
```

JavaScript

Processamento no cliente HTTP

Considerações

Inline e *remote script files*

Não bloqueio da carga dos *scripts*

Comportamento inconstante dos clientes HTTP (API *browser*)

Agressividade nas actualizações, no desenvolvimento.

Perguntas?

https://code.visualstudio.com