07 JavaScript (2/3)

Introdução às Tecnologias Web

Perguntas

JavaScript define?

```
var xpto = 10;
```

“use strict”;

Falsy e Truthy?

Qual o significado?

```
1  "use strict";
2
3  function somaValores (val1, val2) {
4      var soma = val1 + val2;
5
6      alert ("A soma dos dois valores é:" + soma);
7      return soma;
8  }
9
10 function mainFunction () {
11     var resultado;
12     var v1, v2;
13
14     v1 = 10;
15     v2 = 4 * v1;
16     resultado = somaValores (v1, v2);
17     if (resultado < 10) {
18         alert ("Resultado muito pequeno!");
19     } else {
20         alert ("Resultado = " + resultado);
21     }
22 }
23
24 window.onload = mainFunction;
25
```

Qual o significado?

```
1  "use strict";
2
3  function somaValores (val1, val2) {
4      var soma = val1 + val2;
5
6      alert ("A soma dos dois valores é:" + soma);
7      return soma;
8  }
9
10 function mainFunction () {
11     var resultado;
12     var v1, v2;
13
14     v1 = 10;
15     v2 = 4 * v1;
16     resultado = somaValores (v1, v2);
17     if (resultado < 10) {
18         alert ("Resultado muito pequeno!");
19     } else {
20         alert ("Resultado = " + resultado);
21     }
22 }
23
24 window.onload = mainFunction;
25
```

Resumo Aula Anterior

Introdução ao JavaScript

Define comportamento

Conceitos Nucleares

Operadores, expressões, comentários

Tipos, Valores e Variáveis

Primitivos, scope das variáveis

Decisões e Funções

if else, switch, function

Sumário

Ciclos

Objetos

Arrays

01

CICLOS

Ciclos

Permitem repetir partes do código

Existem 4 ciclos

- while
- do / while
- for
- for / in

Exemplo típico dos ciclos:

- iterar sobre os elementos de um array

while

```
while(expressão) {
    // código1
}
// código2
```

Expressão avaliada antes de entrar no ciclo

Enquanto **expressão** é *truthy* executa **código1**

Se **expressão** é *falsy*, "salta" para **código2**

Exemplo while

```
var count = 0;
while (count < 10) {
    console.log(count);
    count++;
}
```

do / while

```
do {
    // código1
} while(expressão);
// código2
```

Expressão avaliada no final do ciclo

Ciclo executado pelo menos uma vez

Enquanto **expressão** é *truthy* executa **código1**

Se **expressão** é *falsy*, "salta" para **código2**

Exemplo do / while

```
var count = 0;
do {
    console.log(count);
    count++;
} while (count < 10);
```

for

Facilitam a criação de ciclos

- já inclui um contador
- incrementa o contador
- testa uma condição

```
for (inicialização; teste; incremento) {
   // código1
}
```

Três expressões para a variável do ciclo

- Inicialização – realizada antes do início do ciclo
- Teste – realizado antes de cada iteração do ciclo
- Incremento – realizado no final de cada iteração do ciclo

Exemplo for

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

02

OBJETOS

Um objeto do dia a dia - Carro

Propriedades:

- Marca
- Modelo
- Cor
- Ano
- Combustível
- Matricula
- ...

Métodos (Ações)

- Start
- Stop
- Travar
- Acelerar
- ...

O que é um Objeto JS?

É um valor composto com múltiplos valores

Tem propriedades e métodos

Propriedades

São pares nome:valor

```
var carro = { marca:"Volvo", modelo:"XC60"}
```

Permitem guardar e recuperar valores pelo nome

```
var m = carro.marca;          // recuperar valor
carro.modelo = "XC90";        // guardar valor
```

Podem ser adicionadas/eliminadas de um objeto

```
delete carro.modelo;          // eliminar propriedade
carro.cor = "azul";           // adicionar propriedade
```

Métodos

Ações que se podem realizar no objeto

Ex: carro.start ();

São guardados em propriedades

```
Ex: var carro = {
      marca: ”Volvo”,
      modelo: ”XC60”,
      start: function() {
        // código da função
      }
    }
```

Guardar e Usar Propriedades

Usando o .

```
// Guardar valores
carro.marca = “BMW”;

// Usar valores
var c1 = carro.marca;
var c2 = carro.modelo;
```

Notação de arrays

```
// Guardar valores
carro[“marca”] = “BMW”;

// Usar valores
var c1 = carro[“marca”];
var c2 = carro[“modelo”];
```

Vantagens Notação de Arrays

Usa uma string, logo pode ser construída em run-time

```
var addr = "";
for (i = 0; i < 4; i++) {
  addr += customer["address" + i]+'\n';
}
```

Como Criar Objetos? - Literais

Lista de pares nome:valor dentro de { }

Exemplos:

```
var vazio = {};            // objeto sem propriedades

var ponto = { x:3, y:6 }; // duas propriedades
```

Como Criar Objetos? - Literais

```
var itw = {
    nome: “Introdução às Tecnologias Web”,
    sigla: “itw”,
    responsavel: {
        nome: “Manuel Fonseca”,
        email: “mjfonseca@ciencias.ulisboa.pt”
    }
}
```

Como Criar Objetos? – new Object()

Usando a keyword new

Exemplos:

```
var ponto = new Object();
ponto.x = 3;
ponto.y = 6;
```

Igual à anterior, mas menos eficiente

Preferível usar a criação usando literais!

Como Criar Objetos? – construtor

Métodos anteriores só criam um objeto

Permite criar vários objetos do mesmo tipo

Necessário definir um construtor (função)

Como Criar Objetos? – construtor

```
function ponto (xx, yy) {
    this.x = xx;
    this.y = yy;
}

var p1 = new ponto (4, 5);
var p2 = new ponto (10, 34);
```

A keyword this

Representa o objeto que é "dono" do código JS

Numa função representa o objeto que é "dono" da função

Num objeto, é o próprio objeto

A keyword this

Num construtor o *this* não tem valor. É um substituto do novo objeto

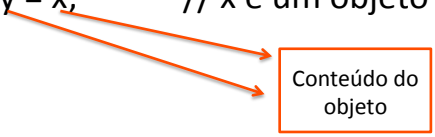
O valor do *this* passará a ser o novo objeto quando o construtor for usado para criar um objeto

```
function ponto (xx, yy) {
    this.x = xx;
    this.y = yy;
}
var p1 = new ponto (4, 5);
```

Objetos manipulados por Referência

Objetos são manipulados por referência

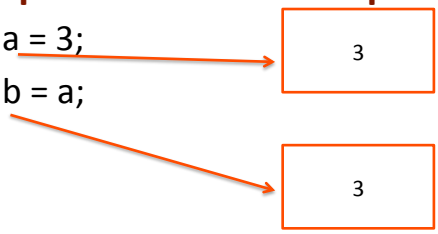
var y = x; // x é um objeto



Dados primitivos são manipulados por valor

var a = 3;

var b = a;



Aceder a objetos não existentes

Aceder a uma propriedade que não existe

Não é erro

Devolve undefined

Aceder a um objeto que não existe

É erro

Verificar se objeto não é null

```
Ex: if (p2) {
        var x3 = p2.x;
    }
```

03

ARRAYS

Arrays

Coleções ordenadas de valores (elementos)

Permitem guardar vários valores numa só variável

Cada elemento tem uma posição no array (índice)

Um array pode conter elementos de tipos diferentes

Elementos do array podem ser objetos ou outros arrays

O índice do primeiro elemento é 0 (zero)

Arrays

São dinâmicos (podem encolher ou esticar)

- Não é necessário especificar um tamanho

Um array pode ser esparso (ter índices sem valores)

Todos os arrays têm a propriedade *length*

- Dá o número de elementos

Arrays são um caso particular de objeto

- Os seus índices são propriedades cujo nome são inteiros

Criação de Arrays

```
// Array vazio
var vazio = [];

// Array com 5 elementos numéricos
var pares = [2, 4, 6, 8, 10];

// Array com 3 elementos de tipos diferentes
var mix = [ 1.1, true, "a" ];
```

Criação de Arrays

```
// Array usando expressões
var base = 8;
var table=[base, base+1, base+2];

// Array sem alguns elementos. Elementos não
// definidos ficam com o valor undefined
var vals = [2, , , 5, 9];
```

Evitar new Array()

Um Array pode ser criado usando new

```
var ponto = new Array ();
```

Mas devemos evitar!

```
var ponto = [4, 5];            // BOM!
var ponto = new Array (4,5)    // MAU!
```

Adicionar/Remover Elementos

Adicionar mais fácil usando a propriedade *length*

Garante que não ficam "buracos"

```
var frutas = ["Laranja", "Maçã",
  "Pera"];
frutas[frutas.length] = "Morango";
```

Remover usando *delete*

```
delete frutas[1]; //Elemento 1 (Maçã)
                  passa a undefined
```

Métodos

valueOf () – Devolve o array como uma string

toString () – Devolve o array como uma string

join () – Junta todos os elementos numa string, mas permite definir o separador (join (" * ");)

Métodos

pop () – Remove o último elemento

Devolve o elemento retirado

push () – Adiciona um elemento no fim

Devolve o novo tamanho do array

Pode adicionar mais que um elemento de uma vez

Métodos

shift () – Remove o primeiro elemento e move os outros para "baixo"

Devolve o elemento retirado

unshift() – Adiciona um elemento no início e move os outros para "cima"

Devolve o novo tamanho do array

Métodos

splice() – Adiciona novos elementos

```
frutas.splice (2, 0, "Morango", "Banana");
```

Insere na posição 2

Remove 0 elementos

["Laranja", "Maçã", **"Morango", "Banana"**, "Pera"]

sort () – Ordena por ordem alfabética

Por omissão o sort considera os valores strings

Elementos undefined ficam no fim

reverse () – Inverte a ordem dos elementos

Combinação com sort pode ordenar descendentemente

Sort de Números (e outros)

Necessário definir função de comparação

Deve devolver um valor negativo, zero, positivo

```
function (a, b) { return a-b; }
```

Ex: function (40, 100) devolve -60 -> 40 deve ficar antes do 100

```
frutas.sort(function(a, b){ return a-b;});
```

Juntar e Dividir Arrays

concat() – Cria novo array juntando outros

```
var carros=["BMW","Volvo","Audi","Fiat"]
var motas = ["KTM", "Honda"]
var veiculos = carros.concat(motas);
```

slice() – Retira uma parte do array para um novo

```
var car = carros.slice(1, 3);
```

Retira elementos da posição 1 à 3 (não incluída)

=> "Volvo","Audi"

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

04

MAIS CICLOS

break

Sai de um ciclo

```
for (var i = 0; i < a.length; i++)
  {
     if (a[i] == "X") {
          break;
     }
}
```

continue

Interrompe a iteração do ciclo e começa uma nova iteração

```
for(i = 0; i < d.length; i++) {
     if (!d[i]) { // d[i] é undefined
          continue;
     }
     total += d[i];
}
```

for com Arrays

```
var a = "Exemplo com Arrays";

for(var i=0; i<a.length; i++){
    console.log(a[i]);
}
```

for / in com Objetos

```
var relogio = {
    hora:17,
    min:21,
    seg:33 };

for(var p in relogio){
    console.log(relogio[p]);
}
```

05

RESUMINDO

Resumo

Ciclos

- while, do /while, for, for / in

Objetos

- Guardam vários valores
- Têm Propriedades e Métodos
- São manipulados por referência

Arrays

- Caso particular de objeto
- Têm vários métodos para os manipular

Próxima Aula