Introdução à Tecnologia Web

JavaScript
Arrays

Profª MSc. Elizabete Munzlinger
www.elizabete.com.br

JavaScript – Arrays

Índice

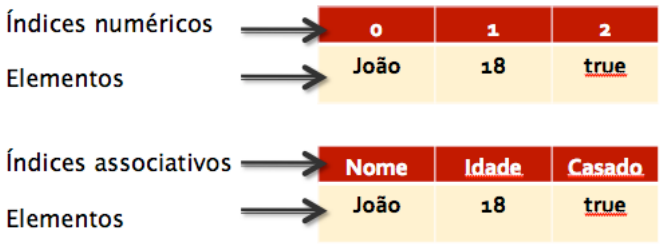
*Itens com * não fazem parte do conteúdo programático deste ano. São para estudo extra.*
Explicações e exemplos complementares vistos em sala de aula.
Não utilizar este documento como única fonte de estudo!

1 – ARRAYS (ARRANJOS)

Um array é uma variável "particionada" capaz de armazenar diversas outras variáveis (elementos). Em JavaScript, os elementos a serem armazenados no array podem ser de qualquer tipo (numérico, cadeia de caracteres, booleano, objeto). Assim, quando um array armazena outro array em uma de suas posições, está aumentando um uma dimensão (bidimensional, tridimensional ...)

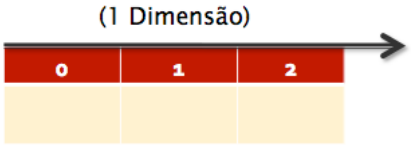
Os dados podem ser acessados individualmente através de sua posição dentro do array. A posição é dada por um índice, que em JavaScript, pode ser numérico ou associativo (textual).

Em JavaScript, o array é considerado um objeto (objeto do tipo Array). Funciona como uma estrutura de dados especial também conhecido por vetor (para arrays unidimensionais) ou matriz (para arrays bidimensionais).



1.1 – Array unidimensional (uma dimensão)

Em JavaScript, o array unidimensional armazena variáveis de qualquer tipo com exceção de outro array. Assim, o array cresce em uma direção, o que significa que pode continuar a receber novos elementos se necessário.



a) Criando um array unidimensional

Ao criar um array em JavaScript, pode-se definir ou ocultar o tamanho inicial do array. Em JavaScript, mesmo definindo um tamanho inicial é possível acrescentar novos elementos posteriormente. Podemos criar um array de 3 formas: regular, condensada e literal.

Exemplo 1: Forma regular com índices numéricos ou associativos

```
<script language="javascript">
<!--
// COM ÍNDICES NUMÉRICOS
var nomes = new Array();     <- usamos o construtor de objetos "new"
nomes[0] = "Maria";          <- armazena cada informação em um índice diferente.
nomes[1] = "João";
nomes[2] = "Zuleica";

// COM ÍNDICES ASSOCIATIVOS
var dados = new Array();     <- usamos o construtor de objetos "new"
nomes["nome"] = "Maria";     <- armazena cada informação em um índice associativos diferente.
nomes["idade"] = 23;
nomes["cpf"] = "123.456.789-00";
-->
</script>
```

Exemplo 2: Forma condensada

```
<script language="javascript">
<!--
var nomes = new Array("Maria", "João", "Zuleica");<- usamos o construtor de objetos "new"
-->                                               armazena cada informação separada por vírgula
</script>                                         diretamente dentro do parênteses do Array
```

Exemplo 3: Forma literal *

```
<script language="javascript">
```

```
<!--
var nomes = ["Maria", "João", "Zuleica"];  <- sem o uso do construtor "new" e uso de colchetes
-->
</script>
```

b) Tamanho de um array

Através da propriedade **length**, podemos saber a quantidade de elementos que contém um array. Saber o tamanho de um array é particularmente importante para o caso de percorrer o array para acessar todos os dados.

Exemplo:

```
<script language="javascript">
var dados = new Array("Maria", "casada", 23);
alert(dados.length);        // apresenta o numero de elementos (3)
</script>
```

c) Acessando dados específicos de um array

Ler dados de um array significa acessar o conteúdo de cada elemento do array.

Exemplo:

```
<script language="javascript">

// ACESSANDO DADOS COM ÍNDICES NUMÉRICOS
var dados = new Array("Maria", "casada", 23);
document.write(dados[0]);   // acessa o elemento do índice 0 -> Maria

// ACESSANDO DADOS COM ÍNDICES ASSOCIATIVOS
var dados = new Array();    <- usamos o construtor de objetos "new"
nomes["nome"] = "Maria";    <- armazena cada informação em um índice associativos diferente.
nomes["idade"] = 23;
nomes["cpf"] = "123.456.789-00";
document.write(dados["nome"]);      // acessa o elemento associado ao índice "nome" -> Maria
</script>
```

d) Acessando todos os dados de um array

Para ler todos os dados do array podemos usar uma estrutura de controle de "repetição". Em JavaScript podemos usar o FOR para ler conteúdo armazenado em índices numéricos. Para ler elementos armazenados em índices associativos usamos o FOR IN, que lê qualquer tipo de índice.

Exemplo com FOR:

```
<script language="javascript">
var cliente = new Array("João", 18, true);        <- cria o array
for (i=0; i< cliente.length; i++) {               <- percorre do indice 0 ao comprimento do array
      document.write(cliente[i] + "<br>");        <- comando de impressão para o índice atual
}
</script>
```

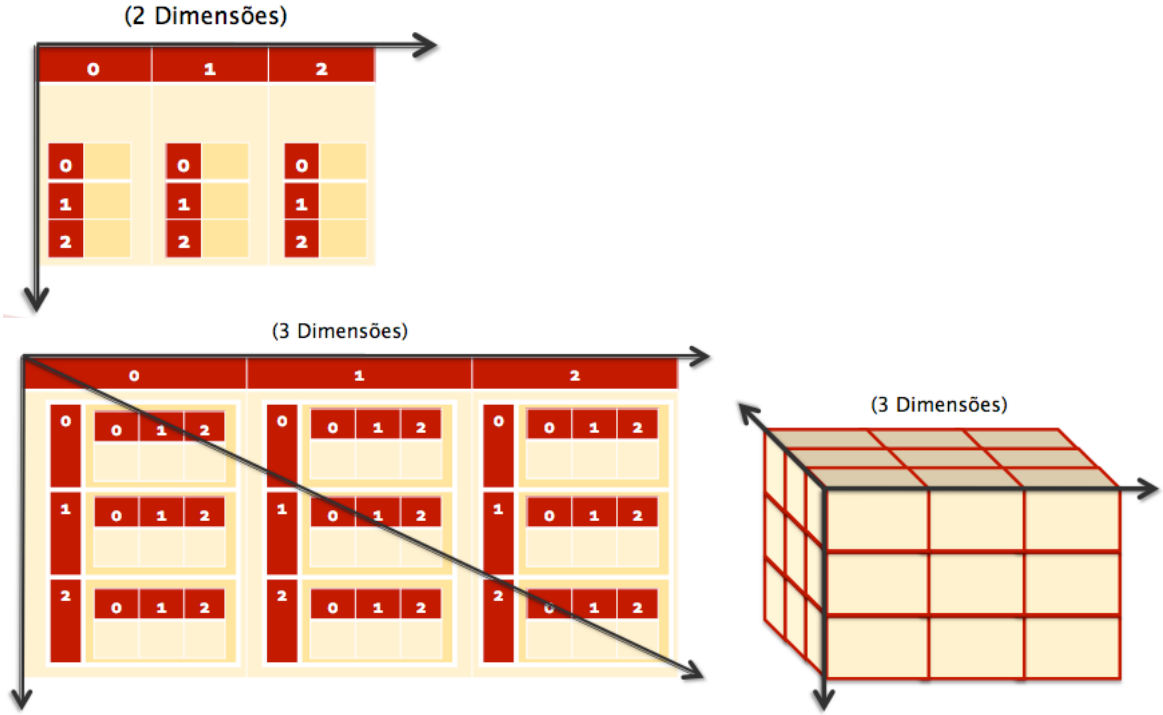
Exemplo com FOR IN:

```
<script language="javascript">
var cliente = new Array(3);
cliente["nome"] = "João";
cliente["idade"] = 18;
cliente["casado"] = true;

for (i in cliente) {                        <- percorre todo e qualquer índice do array cliente
      document.write(cliente[i] + "<br>");
}
</script>
```

1.2 Array multidimensional (mais de uma dimensão) *

Em JavaScript, um array multidimensional é um array que armazenas outros arrays como elementos. Um array multidimensional pode continuar a receber novos elementos se necessário, então são os arrays internos que recebem esses novos elementos.



a) Criando e inserindo dados em um array bidimensional

Um array bidimensional é criado a partir de outros arrays existentes. O procedimento de criação é semelhante ao do array unidimensional.

Exemplo:

```
<script language="javascript">

var cliente1 = new Array(“João”, 18, true);// cria 1º array Unidim.
var cliente2 = new Array(“Maria”, 22, true);       // cria 2º array Unidim.
var cliente3 = new Array(“Carlos”, 33, false);// cria 3º array Unidim.

var clientes = new Array();
clientes[0] = cliente1;      // recebe a o 1º array como elemento
clientes[1] = cliente2;      // recebe a o 2º array como elemento
clientes[2] = cliente3;      // recebe a o 3º array como elemento

// OU

var clientes = new Array( new Array("João", 18, true), new Array("Maria", 22, true), new
Array("Carlos", 33, false));

</script>
```

b) Lendo dados específicos de um array bidimensional

Ler dados de um array significa acessar o conteúdo de cada elemento do array.

Exemplo:

```
<script language="javascript">
document.write(clientes);    // imprime todos os dados de todos os
                             // arrays (separado por virgula)

document.write(clientes[0]); // acessa o elemento do índice 0 no array
                             // ou seja, imprime todos os dados do
                             // array cliente1 (separado por virgula)
document.write(clientes[0][1]); // acessa o elemento/conteúdo 18
document.write(clientes[2][0]); // acessa o elemento “Carlos”
```

```
</script>
```

Para ler todos os dados do array podemos usar uma estrutura de controle de "repetição". Em JavaScript podemos usar o FOR para ler conteúdo armazenado em índices numéricos, e o FOR IN para ler elementos armazenados em índices numéricos e principalmente associativos.

Exemplo com FOR aninhados:

```
<script language="javascript">
// dado o array bidimensional clientes criado acima

for (i=0; i< clientes.length; i++) {
      for (j=0; j< clientes[1].length; j++) {
            document.write(clientes[i][j] + "<br>");
      }
}
</script>
```

Exemplo com FOR IN aninhados:

```
<script language="javascript">
// dado o array bidimensional clientes criado acima

for (i in clientes) {
      for (j in clientes[i]) {
            document.write(clientes[i][j] + "<br>");
      }
}
</script>
```

Referências Bibliográficas

1. CARDOSO, Márdel. ***Desenvolvimento web para o ensino superior***. Rio de Janeiro: Axcel Books, 2004.
2. CSS.***W3C Recomendation: Cascading Style Sheets Home Page***. Disponível online em: [http://www.w3.org/Style/CSS/]
3. DAMIANI, Edgard. ***Javascript - Guia de Consulta Rápida***. São Paulo: Novatec, 2001.
4. DOCTYPE. ***Recommended list of DOCTYPE W3C***. Disponível online em: [http://www.w3.org/QA/2002/04/valid-dtd-list.html]
5. FREEMAN, Eric, FREEMAN, Elisabeth. ***Use a Cabeça: HTML com CSS e XHTML***. Rio de Janeiro: Alta Books, 2008.
6. GOODMAN, Danny. ***Javascript : a Biblía***. Rio de Janeiro: Campus, 2001.
7. GOODMAN, Danny. ***Javascript e DHTML: Guia Prático***. Rio de Janeiro: Campus, 2008.
8. HTML. ***HTML Validator W3C***. Disponível online em: [http://validator.w3.org/]
9. HTML. ***HTML Working Group***. Disponível online em: [http://www.w3.org/html/wg/]
10. HTML. ***Overview of HTML***. Disponível online em: [http://www.w3.org/html/]
11. HTML. ***W3C Recomendation: HTML 4.01 Specification***. Disponível online em: [http://www.w3.org/TR/html4/]
12. MACEDO, Marcelo da Silva. ***Construindo sites adotando padrões Web***. Rio de Janeiro: Ciência Moderna, 2004.
13. MARCONDES, Christian Alfim***. HTML fundamental 4.0***. São Paulo: Érica, 2005.
14. MORRISON, Michael. ***Use a Cabeça: Javascript***. Rio de Janeiro: Alta Books, 2008.
15. NEGRINO, Tom; SMITH, Dori. ***Javascript para a World Wide Web***. Rio de Janeiro: Campus, 2001.
16. SILVA, Maurício Samy. ***JavaScript - Guia do programador***. São Paulo: Novatec, 2010.
17. W3C. ***JavaScript-Based Style Sheets***. Disponível online em: [http://www.w3.org/Submission/1996/1/WD-jsss-960822]
18. W3C. ***Scripts***. Disponível online em: [http://www.w3.org/TR/REC-html40/interact/scripts.html]
19. W3C. ***W3C Standards***. Disponível online em: [http://www.w3.org/standards/]
20. W3C. ***W3C Tutorials***. Disponível online em: [http://www.w3.org/wiki/Category:Tutorials]
21. W3C. ***Word Wide Web Consortium***. Disponível online em: [http://www.w3.org/]
22. W3Schools. ***JavaScript Tutorial***. Disponível online em: [http://www.w3schools.com/js/default.asp]
23. Webdesign. ***Revista Webdesign***. Rio de Janeiro: Artecom, 2010.
24. Wide. ***Revista Wide***. Rio de Janeiro: Artecom, 2011.
25. XHTML. ***W3C Recomendation: XHTML 1.0 The Extensible HyperText Markup Language (Second Edition). A Reformulation of HTML 4 in XML 1.0***. Disponível online em: [http://www.w3.org/TR/xhtml1/]