UNINOVE

Definição e Aplicação de Vetores e Matrizes

APRESENTAR OS CONCEITOS DE VETORES E MATRIZES EM JAVASCRIPT.

AUTOR(A): PROF. EDSON MELO DE SOUZA

1. Introdução

Habitualmente um programa de computador necessita de variáveis para poder realizar as tarefas designadas pelo programador. E, dentro deste universo, nos deparamos em certas situações em que uma variável deve armazenar mais de um valor. Por esse motivo, será estudado neste tópico a utilização de vetores (*arrays*) e matrizes (arrays multidimensionais).

2. Vetores ou *arrays*

Para iniciar o estudo dos vetores, vamos descrever um problema real. Imagine que você necessita criar uma lista de nomes dos seus colegas de classe e, para isso, deverá criar uma variável para guardar os registros. Veja o exemplo:

var nome1 = “João”;

var nome2 = “Maria”;

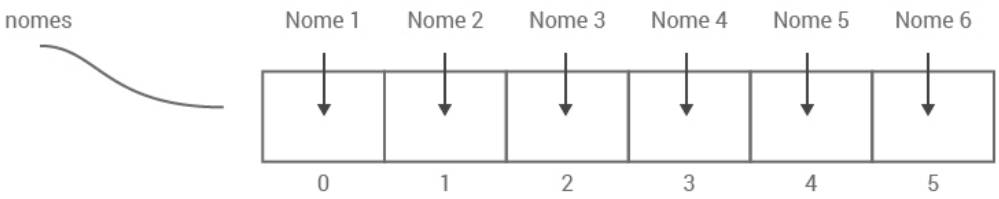
var nome3 = “Paulo”;

Você já percebeu onde isso vai parar caso a turma tenha 50 alunos? Mesmo que você crie as 50 variáveis, como encontrar a que contenha o nome que deseja encontrar? E se entrar um aluno novo? Para resolver esse e outros problemas, fazemos uso de variáveis especiais conhecidas com vetores ou *arrays*.

Segundo Teruel (2016), *“Arrays* são um tipo de variável que permitem armazenar vários valores, ao contrário de uma variável simples. ”. Os *arrays* podem ser definidos de duas formas: unidimensional (vetores) e multidimensionais (matrizes).

2.1 *Arrays* unidimensionais

Este tipo de *array* pode armazenar dados em uma única dimensão que é representada em uma linha com divisão em colunas. Pense como se você tivesse uma caixa comprida grande com várias divisões internas (mini caixas), onde é possível guardas "coisas" ou valores em cada uma dessas divisões. Na figura a seguir é mostrada a estrutura de um *array* de uma única dimensão.



Legenda: REPRESENTAçãO DE UM ARRAY UNIDIMENSIONAL

Vamos considerar que o nosso *array*, figura acima, é para guardarmos nomes de colegas de classe. Então, vamos chamar esse *array* de "nomes", pois armazenará mais de um nome. Perceba que na parte superior da figura estão escritos Nome 1, Nome 2, etc., com uma seta apontado para dentro do retângulo (se lembra da mini caixa?), pois é, esses valores (Nome 1, Nome 2, etc.) ficam guardados ou armazenados dentro do *array* em um "local" específico, mais conhecido como posição ou coluna, que são identificadas por índices, ou seja, um valor que representa a posição dentro do *array*.

ATENçãO!!!

Em Javascript, o índice de um array SEMPRE se inicia em zero (0)

Os arrays podem armazenar diversos tipos de dados como textos, caracteres, inteiros, duplos, entre outros e, em Javascript, um *array* pode ser declarado das seguintes formas:

var nome_do_array = ["valor 1", "valor 2", etc...];

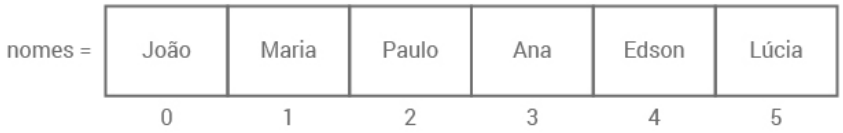
Note que a declaração é diferente de uma variável comum, pois recebe um par de colchetes "[]", o que indica que se trata de um *array*. A outra forma é a seguinte:

var nome_do_array = new Array("valor 1", "valor 2", etc...);

No exemplo a seguir, vamos criar então a nossa lista de colegas de classe utilizando a primeira forma:

var alunos = ["João", "Maria", "Paulo", "Ana", "Edson", "Lúcia"];

Após a declaração, fizemos o preenchimento do *array* com os nomes, indicando, para cada um, em qual posição deverá ficar usando os números para os índices. Veja na figura a seguir como ficou então o nosso *array* após o preenchimento:



Legenda: ESTRUTURA DE UM ARRAY UNIDIMENSIONAL COM DADOS ARMAZENADO

Para acessar o elemento utilizamos a seguinte instrução:

Alunos[0] = “João”

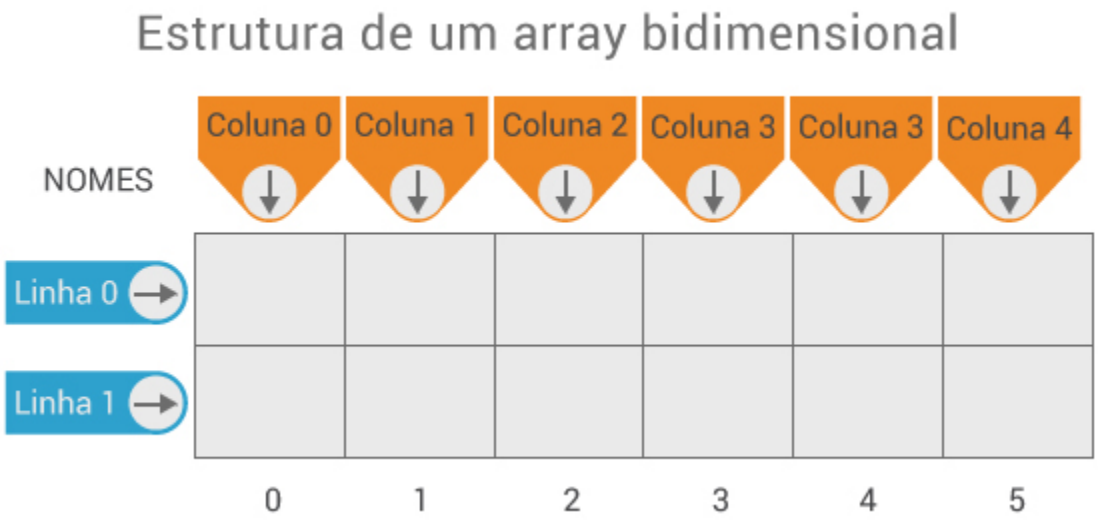
Alunos[3] = “Ana”

Para outros tipos de dados, os procedimentos são os mesmos.

2.2 *Arrays* multidimensionais

Existem casos que um *array* com apenas uma dimensão não é suficiente para atender determinadas necessidades. Imagine uma situação em que é necessário armazenar o nome de um colega de classe e também o seu telefone. Se pensarmos em um *array* com apenas uma dimensão, como isso seria feito? Por esse motivo é que utilizaremos um *array* com duas dimensões.

Um *array* bidimensional não difere em nada de um *array* unidimensional, exceto pelo fato de ter mais dimensões. Vamos ver então a representação gráfica deste tipo de *array* na figura a seguir:



Legenda: ESTRUTURA DE UM ARRAY BIDIMENSIONAL

Perceba que agora existe mais de uma referência para os índices, ou seja, antes havia apenas o índice da coluna, pois só existia uma linha, agora toda referência passará a ser o par (linha, coluna). Vamos ver como declaramos esse *array* de alunos para guardar o nome e o telefone e qual será seu resultado.

```
var alunos = [
        ["João", "11 3232-5566"],
```

```
            ["Maria", "21 4412-6859"],
            ["Paulo", "21 5555-8787"],
            ["Ana", "11 6264-5956"],
            ["Edson", "11 2345-4564"],
            ["Lucia", "11 3578-9512"],
        ] ;
```

A seguir é mostrado o acesso ao conjunto dos elementos:

alunos[2][0] = “Paulo”

alunos[2][1] = “21 5555-8787”

alunos[5][0] = “Lucia”

alunos[5][1] = “11 3578-9512”

alunos[5][0] = “Maria”

alunos[5][1] = 21 4412-6859”

Perceba que a variável foi declarada com dois colchetes “[][]”, pois um faz referência a linha e outro a coluna. Uma relação de matrizes com outras situações pode ser exemplificada com uma planilha do Excel, onde temos linhas e colunas (A1, C4, etc.). Para recuperar um valor do *array*, é necessário informar os dos índices, linha e coluna, caso contrário, não será possível recuperar o valor armazenado.

2.3 Limites do *Array*

É de extrema importância que seja verificado o tamanho de um *array* antes de sua manipulação (próximo item a ser abordado), seja ele de uma ou mais dimensões, porque se for ultrapassado o seu tamanho, ocorrerá um erro em tempo de execução.

Para isso é importante sempre lembrar que o índice de um *array* é iniciado em zero (0), ou seja, se um *array* contiver 10 elementos, o seu maior índice será o número de elementos menos um (elementos-1).

2.4 Declarando e Percorrendo *Arrays*

Agora chegou a hora de colocarmos em prática a teoria sobre *arrays* utilizando a JSP. Vamos colocar então a mão na massa! Nos exemplos a seguir serão realizadas diversas manipulações com *arrays* e, para facilitar o entendimento, cada instrução será descrita utilizando comentários diretamente no código.

Procure estudar os códigos e entender o que está acontecendo antes de partir para a codificação. Ah, uma ótima prática para poder aprender e dominar a programação é NÃO COPIAR e COLAR os códigos, mas digita-los, pois assim haverá um melhor entendimento.

2.4.1 *Arrays* unidimensionais

Neste primeiro exemplo vamos declarar um *array* de uma dimensão, atribuir valores e acessa-lo, tanto manualmente, quanto utilizando laços de repetição, pois laços são extremamente úteis para acessar valores contidos em *arrays*.

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title></title>

        <script type="text/javascript">

        // criação da função
        function alunos01(){

                // declara uma variável do tipo array e atribui os valor
                var alunos = ["João", "Maria", "Paulo", "Ana", "Edson",


                // mostra os elementos do array individualmente, de acor
                document.write("<h3>Declaração do array</h3>");
                document.write('var alunos = ["João", "Maria", "Paulo",
                document.write("<br><h3>Exibindo individualmente</h3>");
                document.write(" alunos[2] = " + alunos[2] + "<br>");
                document.write(" alunos[5] = " + alunos[5] + "<br>");
                document.write(" alunos[1] = " + alunos[1] + "<br>");

                document.write("<h3>Exibindo com um laço (for)</h3>");
                // laço de reptição para exibir os valores armazenados n
                for (var i = 0; i < 6; i++) {

                        // escreve no corpo do html
                        document.write(" alunos[" + i + "] = " + alunos[
                }
        }
        </script>

</head>
<!-- Executa a função alunos01() ao carregar a página-->
<body onload="alunos01();">

</body>
```

```
40. </html>
```

Declaração do array

Var alunos = ["João", "Maria", "Paulo", "Ana", "Edson", "Lucia"];

Exibindo individualmente

alunos[2] = Paulo
alunos[5] = Lucia
alunos[1] = Maria

Exibindo com um laço (for)

alunos[0] = João
alunos[1] = Maria
alunos[2] = Paulo
alunos[3] = Ana
alunos[4] = Edson
alunos[5] = Lucia

Uma estratégia para não ter problemas com o tamanho de um *array* é utilizar um dos seus métodos, isso mesmo, após ser criado, um *array* possui métodos e campos que podem ser acessados. Isso é importante, pois, além de não causar problemas com os índices, ainda deixa o programa mais independente.

Na linha 14 é declarado e inicializado o array com os valores. Nas linhas de 21 a 23 são mostrados os valores acessando diretamente a posição dentro do array. O bloco de repetição, linhas 27 a 32, mostra TODOS os elementos de acordo com a ordem de declaração.

ATENçãO!!!

Após criado um *array*, não será possível redimensionar o seu tamanho. Para que isso seja possível, será necessário criar outro *array* e então copiar para eles os valores atuais.

No próximo exemplo, vamos percorrer o mesmo *array* anterior, mas agora com uma alteração que pegará o seu tamanho automaticamente.

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.         <meta charset="utf-8">
5.         <meta http-equiv="X-UA-Compatible" content="IE=edge">
6.         <title></title>
7.
8.         <script type="text/javascript">
9.
10.        // criação da função
11.        function alunos02(){
12.
13.                // declara uma variável do tipo array e atribui os valor
14.                // utilizando a notção new Array
15.                var alunos = new Array("João", "Maria", "Paulo", "Ana",
16.
17.                // laço de reptição para exibir os valores armazenados n
18.                for (var i = 0; i <= alunos.length-1; i++) {
19.
20.                        // escreve no corpo do html
21.                        document.write(alunos[i] + "<br>");
22.                }
23.        }
24.        </script>
25.
26. </head>
27. <!-- Executa a função alunos02() ao carregar a página-->
28. <body onload="alunos02();">
29.
30. </body>
31. </html>
```

Perceba que nas linhas 18 foi escrito *alunos.length-1*. Essa instrução fornece o tamanho do *array*, não havendo mais a necessidade de informar manualmente, como no exemplo anterior.

2.4.2 Arrays Bidimensionais

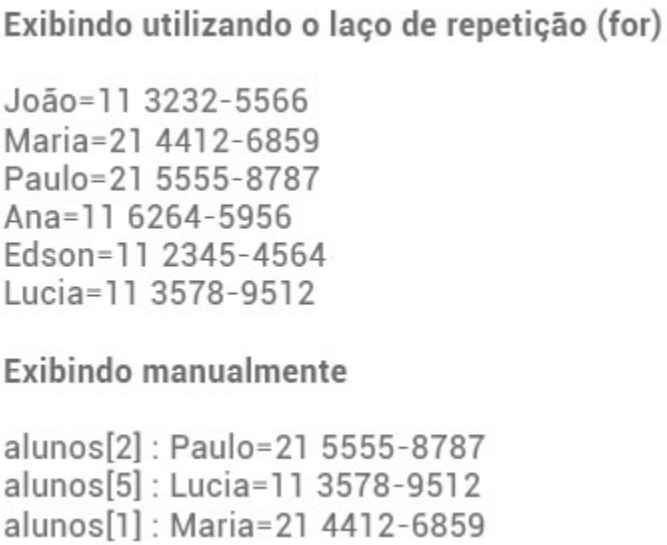
Agora que já sabemos como percorrer um *array* de uma dimensão, vamos partir para duas dimensões. O funcionamento é o mesmo, exceto que agora utilizaremos dois laços ao invés de um só, mas porquê? Simples, porque um laço (mais externo) vai controlar as linhas e outro as colunas (mais interno).

O algoritmo é o seguinte: Para cada linha do *array*, percorrer todas as suas colunas!

Vamos usar o mesmo exemplo anterior, só que agora modificado para guardar o nome e o telefone dos alunos. Depois vamos acessar os valores manualmente e também utilizando um laço.

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title></title>

        <script type="text/javascript">

        // criação da função
        function alunos03(){

                // declara uma variável do tipo array bidimensional
                var alunos = [
                                ["João", "11 3232-5566"],
                                ["Maria", "21 4412-6859"],
                                ["Paulo", "21 5555-8787"],
                                ["Ana",  "11 6264-5956"],
                                ["Edson", "11 2345-4564"],
                                ["Lucia", "11 3578-9512"],
                        ];

                // laço de reptição para exibir os valores armazenados n
                document.write("<h3>Exibindo utilizando o laço de repeti
                for (var i = 0; i < alunos.length; i++) {
                        for (var j = 0; j < 1; j++) {
                                // escreve no corpo do html a linha e a
                                document.write(alunos[i][0] + "=" + alun
                        }
                        document.write("<br>");
                }

                // Exibe a linha e a coluna manualmente
                document.write("<br><h3>Exibindo manualmente</h3>");
                document.write("alunos[2] : " + alunos[2][0] + "=" + alu
                document.write("alunos[5] : " + alunos[5][0] + "=" + alu
                document.write("alunos[1] : " + alunos[1][0] + "=" + alu
        }
        </script>
```

```
40.
41. </head>
42. <!-- Executa a função alunos03() ao carregar a página-->
43. <body onload="alunos03();">
44.
45. </body>
46. </html>
```



SAIBA MAIS!

TEO - https://www.todoespacoonline.com/w/2014/04/lacos-em-javascript/ (https://www.todoespacoonline.com/w/2014/04/lacos-em-javascript/)

iMasters - https://imasters.com.br/artigo/21197/javascript/entendendo-arrays-no-javascript/?trace=1519021197&source=single (https://imasters.com.br/artigo/21197/javascript/entendendo-arrays-no-javascript/?trace=1519021197&source=single)

Resumo

Neste tópico foram apresentados os arrays e as matrizes em Javascript que são fundamentais para manipulação de dados quando não há possibilidade de trabalhar com variáveis simples, além de exemplos de criação e acesso aos dados armazenados.

Conclusão

Este tópico mostrou uma breve introdução aos arrays em Javascript, entretanto, há muito mais recursos que podem ser utilizados para que as estruturas de dados possam ser manipuladas em uma aplicação. Portanto, após "saborear" esta pequena introdução, acesse os links disponíveis para consulta e também a bibliografia para ficar "fera" no assunto.

Utilize os códigos de exemplo e faça seus próprios programas para treinar. Se ficou alguma dúvida, releia o conteúdo, os links de apoio e converse com o seu professor.

Bons estudos!

ATIVIDADE FINAL

Qual a definição de um *array*?

A. É um tipo de variável que permitem armazenar vários valores, ao contrário de uma variável simples.
B. São variáveis que podem armazenar somente valores inteiros.
C. É uma estrutura exclusiva da linguagem Javascript.
D. São estruturas complexas para armazenamento de variáveis simples.

O que é um índice de um array?

A. É o valor que representa a posição dentro do *array*.
B. É a declaração de um array.
C. São os dados contidos em um array
D. É a especificação para um array de mais de uma dimensão.

Considerando o fragmento da instrução: var variável = [[1, 2, 3]] ..., é possível afirmar que se trata de:

A. Criação de um array bidimensional.
B. Criação de um array unidimensional.
C. Acesso aos valores do array.
D. Variável dinâmica.

REFERÊNCIA

ALVAREZ, Miguel Angel. O que é Javascript. Criarweb. com. Disponível em:< http://www. criarweb. com/artigos/184. php>. Acessado em 30 abr. 2017.

BERNICHE, Erica Fernanda; DE ALMEIDA NERIS, Vânia Paula. Análise Comparativa entre Bibliotecas Javascript para o Desenvolvimento de Interfaces Web Ricas. Revista TIS, v. 2, n. 1, 2013.

PEREIRA, Alexandre; POUPA, Carlos. Linguagens Web. Edições Sílabo (2005-2ª edição), 2005.

RAMOS, Miguel Esteban; VALENTE, Marco Tulio. Análise de Métricas Estáticas para Sistemas JavaScript. In: II Brazilian Workshop on Software Visualization, Evolution, and Maintenance (VEM 2014). 2014. p. 30-37.

SEBESTA, Robert W. Conceitos de linguagens de programação. Bookman Editora, 2009.

TERUEL, Evandro Carlos. Programação orientada a objetos com JAVA sem mistérios. São Paulo: Universidade Nove de Julho – UNINOVE. 2016. 3876 p. il.