ALGORITMOS E LÓGICA DE PROGRAMAÇÃO
03/06 – Array

Prof.: Eduardo Tavares

Fatec Matão
Luiz Marchesan

Desenvolvimento de Software Multiplataforma
Fatec Matão

Os Arrays são pares do tipo inteiro-valor para se mapear valores a partir de um índice numérico. Em JavaScript os Arrays são objetos com métodos próprios. Um objeto do tipo Array serve para se guardar uma coleção de ítens em uma única variável.

Exemplo:

```
var arr = new Array();
// Por ser um objeto podemos usar o "new" em sua criação
var arr = new Array(elem1,elem2, ... ,elemN);
// Dessa forma criamos um array já iniciado com elementos.
var arr = [1,2,3,4];
// outra forma é iniciar um array com elementos sem usar o "new".
var arr = new Array(4);
// Dessa forma criamos um array vazio de 4 posições.
```

Para acessar as variáveis dentro de um array basta usar o nome do array e o índice da variável que se deseja acessar.

Exemplo:

```
arr[0] = "Até mais e obrigado pelos peixes";
arr[1] = 42;
document.write(arr[1]);
//imprime o conteúdo de arr[1]
```

Do mesmo modo, pode-se fazer atribuições ou simplesmente ler o conteúdo da posição.

Em JavaScript os arrays podem conter valores de tipos diferentes sem nenhum problema; podemos colocar em um mesmo array inteiros, strings, booleanos e qualquer objeto que se desejar.

Agora faremos uma breve descrição dos métodos que o objeto Array traz consigo.

concat(item1,item2, ... ,itemN) - agrupa dois ou mais arrays e retorna o resultado

Exemplo:

```
var x = ["a vida","o universo"];
var y = ["e tudo mais"];
var resultado =  x.concat(y);
// Isso geraria o array resultado contendo
// ["a vida","o universo","e tudo mais"]
```

join(separador) - agrupa todos os elementos contidos no array em uma string separados pelo que estiver na variável separador, se um caractere separador não for fornecido a vírgula será usada como padrão. Para tal operação, os elementos do array são convertidos para strings caso já não o sejam.

Exemplo:

```
var x = ["a vida","o universo","e tudo mais"];
var str = x.join("\ ");
// Após a execução do código str terá "a vida o universo
// e tudo mais" pois os valores foram agrupados com
// espaço como separador
```

pop() - o último elemento do array é removido e retornado.

Exemplo:

```
var vetor = [4,8,15,16,23,42];
var resposta = vetor.pop();
// Assim resposta irá conter o valor 42.
```

push(item1,item2, ... ,itemN) - insere os N itens no final do array, na ordem que eles foram passados e retorna o novo tamanho (*length*) do array.

Exemplo:

```
var vetor = ["Arthur","Ford",];
var retorno = vetor.push("Marvin");
// Agora o vetor possui ["Arthur","Ford","Marvin"]
// e o valor de retorno é 3.
```

reverse() - retorna o array com os elementos na ordem inversa.

Exemplo:

```
var vetor = [1,2,3,4];
var inverso = vetor.reverse();
// Agora tanto inverso quanto vetor possuem [4,3,2,1]
```

shift() - o primeiro elemento do array é removido e retornado.

Exemplo:

```
var vetor = [4,8,15,16,23,42];
var elemento = vetor.shift();
// Depois de executado elemento terá o valor 4
// e o vetor será [8,15,16,23,42];
```

slice(inicio,fim) - retorna um array contendo os elementos de inicio até fim, mas sem incluir o elemento da posição fim. Caso fim não seja declarado, retorna um array de inicio até o final do array. Se inicio for negativo, o valor será contado a partir do final do array, por exemplo, -2 para indicar a penúltima posição, o mesmo acontece com fim caso seja negativo. Podemos imaginar o sinal de negativo apenas como um inversor no sentido que os elementos são considerados (inicio-fim para valores positivos e fim-inicio para valores negativos).

Exemplo:

```
var meuarray = [1,2,3,4];
var meio = meuarray.slice(1,3);
var meio = meuarray.slice(-3,-1);
// nas duas atribuições acima a variável meio receberia
// o array [2,3] pois -3 equivale a posição do elementos 2
// e a -1 equivale a posição do elemento 4. é preciso lembrar
// que elemento fim não é incluido no array de retorno.
```

sort(comparador) - método que ordena os elementos objeto do Array por ordem alfabética. Isto significa que ele não funcionará para números; para ele o número 33 virá antes do número 7 pois o primeiro algarismo de 33 é menor. Para comparar números ou outros tipos de elementos você pode fornecer ao método um comparador.

Este deve ser uma função com as seguintes características:

- receber dois argumentos: x e y.
- retornar um valor negativo se x < y.
- retornar zero se x == y.
- retornar um valor positivo se x > y.

Caso contrário, o método terá o comportamento padrão descrito acima.

Exemplo 1:

```
var vetor = ["d","y","a","n","m"];
vetor.sort();
// Após a execução vetor conterá o array
["a","d","m","n","y"].
```

Exemplo 2:

```
// Uma função simples para comparar valores numéricos
// os detalhes sobre a utilização de funções serão abordadas
// mais adiante.
function compara(x,y) {
    if (x < y)
        return -1;
    else if (x == y)
        return 0;
    else
        return 1;
}
var vetor = [5,45,1,4,16];
var errado = vetor.sort();
var certo = vetor.sort(compara);
// O array errado irá conter [1,16,4,45,5]
// e o array certo irá conter [1,4,5,16,45]
```

splice(inicio, quantidade, elem1, ... , elemN) - este método pode ser usado para remover, inserir ou substituir elementos em um Array. Os argumentos "inicio" e "quantidade" são obrigatórios, mas os elementos posteriores são opcionais.

Para apenas remover elementos do array devemos usar o método com: inicio sendo o índice do primeiro elemento a ser removido, quantidade sendo o numero de elementos a remover e nenhum outro argumento.

Exemplo 1:

```
var array = [1,2,3,4,5,6];
array.splice(2,2);
// Como resultado teremos em array o vetor [1,2,5,6]
// apenas deletamos dois elementos partindo do índice 2.
```

Para inserir um elemento sem remover nenhum outro, basta utilizar o método com parâmetro inicio sendo o índice da posição onde serão inseridos os elementos, quantidade deve ser 0 pois não desejamos remover nenhum elemento e cada elemento a ser inserido deve ser um novo argumento subsequente.

Exemplo 2:

```
var array = [1,2,5,6];
array.splice(2,0,3,4);
// Como resultado teremos em array o vetor [1,2,3,4,5,6]
// Isso significa que a partir do índice 2 deletamos 0 elementos
// e inserimos os valores 3 e 4
```

Para substituir elementos do array, precisamos que inicio contenha o índice a partir do qual os elementos serão substituídos, quantidade deve conter o número de elementos que serão substituídos e mais um argumento para cada elemento quer será inserido substituindo os elementos originais. Note que se houver menos elementos do que o valor em quantidade, o excedente será apenas removido sem substituição e se houver mais elementos do que posições deletadas o excedente será apenas inserido após as substituições.

Exemplo 3:

```
var array = [1,2,3,4,5,6];
array.splice(2,2,7,8);
// Como resultado teremos em array o vetor [1,2,7,8,5,6]
// Ou seja, a partir do índice 2 deletamos 2 elementos e inserimos
// os elementos 7 e 8 no lugar deles
```

Em todos os casos, o método retorna um array contendo os elementos que foram removidos se houve algum.

unshift(item1,item2, ... ,itemN) - adiciona um ou mais elementos ao início do array na ordem que os argumentos foram fornecidos e retorna o novo número de ítens.

Exemplo:

```
var vetor = [4,2,6,1];
var tamanho = vetor.unshift(5,3);
// Após a execução a variável tamanho irá conter o valor 6
// e vetor será [5,3,4,2,6,1];
```

toString() - este método se comporta da mesma maneira que o método join quando chamado sem nenhum parâmetro, ou seja, ele separa os elementos por vírgula.

Exemplo:

```
var vetor = ["Hiro","Peter","Claire"];
var result = vetor.toString();
// result irá conter a string "Hiro,Peter,Claire"
```

length - propriedade que contém o tamanho do array, ou seja, a quantidade de elementos contidos nele.

objeto.propriedade

Exemplo:

```
var vetor = ["a","b","c","d"];
var tam = vetor.length;
// A variável tam irá conter o valor 4 porque existem quatro
// elementos no array
```