JavaScript

IFRN – INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIAS E TECNOLOGIAS DO RIO GRANDE DO NORTE

Arrays

Array é um objeto destinado a armazenar uma coleção ordenada de dados indexados sequencialmente a partir do zero. Para criar um array, usamos o construtor ***Array()*** precedido do operador ***new***.

Exemplo:

var meuArray = new Array();

Para polular um array, passamos uma lista de argumentos ao construtor contendo os dados do array conforme mostrado a seguir:

var meuArray = new Array(“casa”, “rio”, 5, “Natal”, 124);

Os dados são denominados elementos do array e devem ser separados com vírgula.

Array

Arrays podem conter dados do tipo objeto como mostrado a seguir:

```
var meuArray = new Array("casa", "rio", {x: 14, y: "mar"});
```

Para ler um dado do tipo objeto contido em um array, usamos a sintaxe mostrada a seguir:

```
meuArray[2].x;      // Retorna o número 14
meuArray[2].y;      // Retorna a string mar
```

Usando uma sintaxe alternativa:

```
meuArray[2]["x"];      // Retorna o número 14
meuArray[2]["y"];      // Retorna a string mar
```

Array

Outra forma de criar um array é defini-lo vazio e populá-lo posteriormente:

```
var a = new Array();
 a[0] = 3.1416;
 a[1] = "prego";
 a[2] = true;
 a[3] = "flor";
 a[4] = {nome: "José Antônio", cidade: "Natal"};
```

Resulta em:

a = [3.1416, "prego", true, "flor", {nome: "José Antônio", cidade: "Natal"};

Array

Propriedades do objeto Array

As propriedades do objeto Array são listadas a seguir:

- constructor;
- prototype;
- length.

Array

construct

A propriedade constructo é uma referência à função que cria um objeto.

Exemplo: no script a seguir vamos criar uma array e através da caixa alert retornar a função que criou o array.

```
<script type="text/javascript>
    var arr = [3, "a", 6, 4, "d", "e"];

    Windows.onload = function() {
        alert(arr.constructor);
    }
</script>
```

Array

prototype

A propriedade prototype permite adicionar novas propriedades e/ou métodos a um objeto já existente.

```
<script type="text/javascript>
    function Cilindro(r,h) {
        this.raioBase = r,
        this.altura = h
    };
    Cilindro.prototype.cor = 'null';     //criamos a propriedade cor
    Cilindro.cor = 'azul';               definimos a cor usando a propriedade recém-criada.
    Windows.onload = function() {
        alert(Cilindro.cor);
    };
</script>
```

Array

length

A propriedade length retorna a quantidade de elementos de um array:

```
var arr = new Array(1,2,3,"a", "b", "c", {nome: "José", cidade: "Natal"}

var x = arr.length;     //O valor é sete
```

Array

Métodos

Os métodos do objeto array são:

concat()	every()	filter()	forEach()	indexOF()	lastIndex()
join()	map()	pop()	push()	reduce()	reduceRight()
reverse()	shift()	some()	splice()	toLocaleString()	toString()
unshift()					

Nota: nem todos estão implementados em todos os navegadores.

Array

concat(arg1, arg2, ..., argn) : este método aplicado a um array existente lhe acrescenta os elementos definidos nos seus argumentos:

```
var arr = [1, 2, "a", "b"];

var a1 = arr.concat("m", 32);          // Resulta em a1 = [1, 2, "a", "b", "m", 32];

var a2 = arr.concat("[gol", 456]);     // Resulta em a2 = [1, 2, "a", "b", "gol", 456];

var a3 = arr.concat({x: 3, y: "boi"}); // Resulta em a3 = [1, 2, "a", "b", {x: 3, y: "boi"}]
```

Array

every(função(elem, ind, obj) [,thisObjeto])

Este método destina-se a percorrer cada um dos elementos de um array e executar uma função callback. Assim que a função encontrar um elemento que não satisfaça as condições impostas no corpo da função, retorna ***false***. Caso todos os elementos do array satisfaçam a função, retorna ***true***.

A função call-back é obrigatória e recebe três parâmetros:

- elem : o valor de cada elemento do array;
- Ind : o índice de cada elemento do array;
- Obj : o objeto array.

- thisObjeto é opcional: define um objeto a ser usado como this na função call-back. Se for omitido, é usado o objeto global sendo percorrido.

Nota: alguns navegadores suportam este método e outros não.

Array

Exemplo 1:

```
var arr = [21, 3, 18, 290];
function funcaoUm(elem, ind, obj) {
        return (typeof elem == "number");
};
var x = arr.every(funcaoUm);
alert(x);                          // Retorna true, todos os elementos do array são números.
```

Exemplo 2:

```
var arr = [21, 3, 18, "a", 290];
function funcaoDois(elem, ind, obj) {
        return (typeof elem == "number");
};
var x = arr.every(funcaoDois);
alert(x);                          // Retorna false, todos os elementos do array são números.
```

Array

filter(função(elem, ind, obj) [, thisObjeto])

Este método destina-se a filtrar elementos em um array e retornar o array após aplicação do filtro. Não altera o array original e adminite um argumento obrigatório, que é a função de filtragem, e um argumento opcional.

Tem características idênticas ao método every.

Array

Exercício 1:

```
var arr = [21, 3, 18, "a", 290, "b", 7];
function filtrarArray(elem, ind, obj) {
    return (elem >= 18);
};
arr.filter(filterArray);    // Retorna [21, 18, 290]
```

Exercício 2:

```
var arr = [21, 3, 18, "a", 290, "b", 7];
function filtrarArray(elem, ind, obj) {
    return (elem == "string");
};
arr.filter(filterArray);    // Retorna ["a", "b"]
```

Exercício 3:

```
var arr = [21, 3, 18, "a", 290, "b", 7];
function filtrarArray(elem, ind, obj) {
    return (ind < 4);
};
arr.filter(filterArray);    // Retorna [21, 3, 18, "a"]
```

Array

forEach(função(elem, ind, obj) [, thisObjeto])

Este método destina-se a percorrer cada um dos elementos em um array e executar uma função callback. Não altera o array original e adminite um argumento obrigatório, que é a função de filtragem, e um argumento opcional.

Exemplo:

```
var arr = [21, 3, 18, “a”, 290];
var msg = “”;
function funcaoUm(elem, ind, obj) {
    msg += “arr[“ + ind + “] = “ + elem + “\n”;
}
arr.forEach(funcaoUm);
alert(msg);
```

Array

indexOf(elemento, [arg2, true ou false])

Este método retorna o índice do elemento de um array. Adminte um argumento obrigatório que é o elemento do array cujo índice desejamos saber. Caso o elemento não exista, retorna -1.

Exemplo:

```
var arr = [1, 2, "a", "b", "2", "c", 1, "a", "45"];
var a = arr.indexOf("b")        // Retorna 3
var a = arr.indexOf("a")        // Retorna 2
var a = arr.indexOf(23)         // Retorna -1
var a = arr.indexOf(2)          // Retorna 1
var a = arr.indexOf(45)         // Retorna -1
```

Array

O segundo argumento deste método é facultativo e, quando definido, estabelece o índice do elemento do array a partir do qual (inclusive) deverá iniciar-se a busca.

Exemplo:

```
var arr = [1, 2, “a”, “b”, “2”, “c”, 1, “a”, “45”];
var a = indexOf(1);              // Retorna 0
 var a = indexOf(1,2);           // Retorna 6 – índice do primeiro elemento 1 no array
                                 //começando a busca no terceiro elemento.
```

Array

lastIndexOf(elemento, [arg2, true ou false])

A única diferença desse método em relação a indexOf() é que ele retorna o último índice do elemento procurado caso haja mais de um. Caso o elemento não exista no array, retorna -1.

Exemplo:

```
var arr = [1, 2, "a", "b", "2", "c", 1, "a", "45"];
var a = arr.indexOf("a")          // Retorna 2
var a = arr.lastIndexOf("a")      // Retorna 7
```

Array

join(arg)

Este método transforma os elementos de um array em uma string. Admite um argumento opcional que se destina a criar um separador para os elementos do array. Se não for passado o separador, então será adotado o padrão, que é vírgula (,);

Exemplo:

```
var arr= [1, 2, "a", "b"];
var a = arr.join();          // Resulta na string a = "1,2,a,b"
var a = arr.join(" – ");     // Resulta na string a = "1 – 2 – a – b"
var a = arr.join("*");       // Resulta na string a = "1*2*a*b"
```

Array

map(função(elem, ind, obj) [, thisObjeto])

Este método destina-se a percorrer cada um dos elementos de um array e modifica-los conforme definido em uma função call-back. Não altera o array original e admite um argumento obrigatório, que é a função call-back, e um argumento opcional.

Exemplo:

```
var arr= [21, 3, 18, 290];
function funcaoUm(elem, ind, obj) {
    return (elem = elem*10);
}
var x = arr.map(funcaoUm);
alert(x);                                   // Retorna [210, 30, 180, 2900]
```

Array

pop()

Este método remove o último elementos de um array e retorna o valor que foi removido.

Exemplo:

```
var arr= [1, 2, “a”, “b”];
var a = arr.pop();          // Resulta em a = “b”  e a.length = 3
```

Array

push(arg1, arg2, ..., argn)

Este método acrescenta os argumentos no final de um array e retorna a nova quantidade de elementos do array.

Exemplo:

```
var arr= [1, 2, “a”, “b”];
var a = arr.push(5, 6, 7, “m”);        // Resulta em a = 8  e arr = [1, 2, “a”, “b”, 5, 6, 7, “m”]
```

Array

reduce(função (v1, v2, ind, arr)[, vInicial])

Este método destina-se a percorrer cada um dos elementos do array e executar uma função callback. Não altera o array original e admite um argumento obrigatório, que é a função e, um opcional.

A função requer quatro argumentos:

- v1 (valor anterior),
- v2 (valor corrente),
- ind (índice de cada elemento) e
- arr (o array sendo percorrido).

Se for especificado o argumento opcional valorInicial, o valor de v1 será igual a ele e o valor de v2, igual ao valor do primeiro elemento do array. Se não for especificado valorInicial, o valor de v1 será igual ao valor do primeiro elemeto do array e o valor de v2, igual ao valor do segundo elemento do array.

Array

Exemplo 1:

```
var arr= [1, 4, 7, 15];
function funcaoUm(v1, v2, ind, arr) {
    return v1 + v2;
}
var x = arr.reduce(functionUm);
alert(x);
```

Resulta:

- primeira chamada: v1=1 e v2=4, retonando 1 + 4 = 5
- Segunda chamada: v1=4 e v2=7, retornando 5 + 7 = 12
- Terceia chamada: v1=12 e v2=15, retornando 12 + 15=27

O valor final retornado pela função é 27.

Array

Exemplo 2:

```
var arr= [1, 4, 7, 15];
function funcaoUm(v1, v2, ind, arr) {
    return v1 + v2;
}
var x = arr.reduce(functionUm, 100);
alert(x);
```

Resulta:

- primeira chamada: v1=100 e v2=1, retonando 100 + 1 = 101
- Segunda chamada: v1=101 e v2=4, retornando 101 + 4 = 105
- Terceia chamada: v1=105 e v2=7, retornando 105 + 7=112
- Quarta chamada: v1=112 e v2=15, retornando 112 + 15 = 127

O valor final retornado pela função é 127.

Array

reduceRight(função (v1, v2, ind, arr)[, vInicial])

Este método destina-se a percorrer cada um dos elementos do array e executar uma função callback. A diferença entre este método e o método reduce() é que ele percorre o array do último elemento do array para o primeiro..

A função requer quatro argumentos:

- v1 (valor anterior),
- v2 (valor corrente),
- ind (índice de cada elemento) e
- arr (o array sendo percorrido).

Se for especificado o argumento opcional valorInicial, o valor de v1 será igual a do último elemento do array e o valor de v2 do penúltimo elemento do array.

Array

Exemplo 1:

```
var arr = [1,4,7,15];
function funcaoUm(v1,v2,ind, arr) {
    return v1 + v2;
}
var x = arr.reduceRight(funcaoUm);
alert(x);
```

- 1ª chamada: v1=15 e v2=7, retornando 15 + 7=22
- 2ª chamada: v1=22 e v2=4, retornando 22 + 4=26
- 3ª chamada: v1=26 e v2=1, retornando 26 + 1=27

Array

reverse()

Este método inverte a ordem dos elementos do array. Não cria um novo array com os elementos invertidos, e sim, altera o array existente.

Exemplo:

```
var arr = [1, 2, “a”, “b”];
arr.reverse();                    // Resulta arr = [“b”, “a”, 2, 1]
```

Array

shift()

Este método remove o primeiro elemento de um array e retorna o valor que foi removido.

Exemplo:

```
var arr = [1, 2, "a", "b"];
var a = arr.shift();                    // Resulta em a = 1 e arr.length = 3
```

Array

slice(arg1, arg2)

Este método retorna um subarray do array. Admite um ou dois argumentos que definem o índice inicial e o final do subarray a extrair.

As diretrizes de extração do subarray são:

- O subarray extraído contém o elemento definido no índice inicial (arg1), mas não o elemento definido no índice final (arg2);
- Se for definido apenas um índice (arg1), o subarray extraído começa com o elemento de índice arg1 inclusive e vai até o último elemento;
- Argumentos negativos revertem o inicio de contagem do índice para o último elemento, ou seja, -1 é o último elemento, -2 o penúltimo, e assim por diante.

Array

Exemplo:

```
var arr = [1, 2, 3, 4, “a”, “b”, “c”];
var a = arr.slice(2,5);          // Resulta em a = [3, 4, “a”]
var a = arr.slice(3);            // Resulta em a = [4, “a”, “b”, “c”]
var a = arr.slice(-5,6);         // Resulta em a = [3, 4, “a”, “b”]
var a = arr.slice(-6,-4);        // Resulta em a = [2, 3]
```

Array

sort(função)

Este método se destina a ordenar os elementos de um array. Admite uma função como argumento opcional para definir como será processada a ordenação dos elementos. Se o argumento for omitido, a ordenação será em ordem alfabética crescente.

Exemplo 1:

```
var arr = [“manga”, “laranja”, “limão”, “abacate”, “banana”];
arr.sort();          // Resulta em arr = [“abacate”, “banana”, “laranja”, “limão”, “manga”]
```

Exemplo 2:

```
var arr = [2, 128, -32, 47, 34, 111, -67];
arr.sort(function(a,b) {
    return a-b;
});                  // Resulta em arr = [-67, -32, 2, 34, 47, 111, 128]
```

Array

splice(arg1 [, arg2, arg3, ..., argn])

Este método se destina a inserir ou a remover elementos de um array, isto é, podemos somente inserir, somente remover ou inserir e remover ao mesmo tempo elementos em um array. Admite um argumento obrigatório e vários opcionais, cuja finalidade são descritas a seguir:

- O 1º argumento define a posição onde se iniciará a inserção ou remoção. Atenção: posição não é índice. O primeiro elemento assume posição 1 e seu índice é zero;
- O 2º elemento define a quantidade de elementos a remover. Se esse argumento for omitido serão removidos todos os elementos do aray a partir da posição fornecida pelo 1º argumento;
- Do 3º argumento em diante, definimos os elementos a inserir a partir da posição definida no primeiro argumento.

Array

Exemplo:

```
var arr = [2, 5, 9, 7, 1, 6, 8, 3, 10, 4];
arr.splice(5);                  // Resulta em arr = [2, 5, 9, 7, 1]
 arr.splice(3, 4);              // Resulta em arr = [2, 5, 9, 3, 10, 4]
 arr.splice(5, 0, 13, 54, 32);  // Resulta em arr = [2, 5, 9, 7, 1, 13, 54, 32, 6, 8, 3, 10, 4]
 arr.splice(2, 6, 21, 43);      // Resulta em arr = [2, 5, 21, 43, 10, 4]
```

Array

toString() e toLocaleString()

Este método converte cada elemento do array em uma string e retorna uma lista dos elementos separados por vírgula.

Exemplo:

```
var arr = [16, 7, 2, 23];
arr.toString();                    // Resulta na string arr = “16, 7, 2, 23”
```

Array

unShift(args)

Este método insere elementos no início de um array e retorna a nova quantidade de elementos do array.

Exemplo:

```
var arr = [1, 2, “a”, “b”];
var a = arr.unshift(2, 4, 9);          // Resulta em a = 7 e arr = [2, 4, 9, 1, “a”, “b”]
```