Тема 12. Масиви.

Синтаксис для створення нового масиву - квадратні дужки зі списком елементів усередині.

var arr = [];

var fruits = ["Яблуко", "Апельсин", "Слива"];

var arr = [1, 'Ім\'я', [1,2,3], true];

Якщо в **new Array().** як параметр передати одне ціле число то буде створено порожній масив відповідної довжини, в решті випадків буде створено масив, що складається з аргументів конструктора.

var arr = new Array(2,3); // еквівалентно arr = [2, 3]

var arr=new Array(3); // arr = [,,]

var arr=new Array(-3); // Помилка

Через alert можна вивести масив цілком.

alert (fruits)**;** // Яблуко, Апельсин, Слива

length містить загальне число елементів, збережених в масиві

alert (fruits.length); //3

Найпростіший спосіб очистити масив - присвоїти властивості length нульове значення.

Щоб отримати потрібний елемент з масиву - вказується його індекс у квадратних дужках:

alert (fruits [0]); // Яблуко

alert (fruits [2]); // Слива

Елемент можна завжди замінити або додати:

fruits [2] = 'Груша'; // тепер ["Яблуко", "Апельсин", "Груша"]

fruits [3] = 'Лимон'; // тепер ["Яблуко", "Апельсин", "Груша", "Лимон"]

При додаванні елемента в позицію, що перевищує поточну розмірність масиву приводить до збільшення розмірності. При цьому, елементи значення яких не вказано явно не займають місце в пам'яті. При виводі масиву через alert на їх позиції нічого не, але відповідні коми виводяться. При звертанні безпосередньо до них значення вважається рівне **undefined.**

```
var arr[1,2,3];
arr[5]=5; // в результаті отримаємо масив з 6 елементів [1,2,3,,,5]
alert(a[4]); // undefined
alert(a.length); //6
```

При присвоюванні масивів відбувається копіювання вказівника на масив, а не значень масиву:

```
var a=[1,2,3],b=a;
b[1]='Ой!';
alert(a[1]); // Ой!, а не як очікувалось 2

var a=[1,2,3],b=[];
for (var i=0;i<a.length;i++){b[i]=a[i]};
b[1]='Ой!';
alert(a[1]); //2 як і має бути
```

Приклад 12.0 Знайти суму n чисел.

```
var n=+prompt('n=','');
var a=[];
for (var i=0;i<n;i++){
  a[i]=+prompt('Введіть число:','');
};
var s=0;   //шукаємо суму
for (i=0; i<n; i++){
        document.writeln(a[i]);
        s=s+a[i];
  };
document.writeln(' s= ', s );
```

Приклад 12.0.1 Знайти максимальне число

```
var n=+prompt('n=','');
var a=[];
for (var i=0;i<n;i++){
 a[i]=+prompt('Введіть число:','');
 document.writeln(a[i]);
};
var max=a[0];
for (i=0; i<n; i++){
       if (a[i]>=max) {max=a[i];};
 };
document.writeln(' max= ', max );
```

Приклад 12.0.2. Відсортувати масив по спаданню

```
var n=+prompt('n=','');
var a=[];
for (var i=0;i<n;i++){
 a[i]=+prompt('Введіть число:','');
 document.writeln(a[i]);
};
var max=a[0];
for (i=0; i<n; i++){
      for (j=i+1; j<n; j++){
            if (a[j]>a[i]) {b=a[i];a[i]=a[j],a[j]=b;};

      }
 };
 document.writeln('<br>');
for (var i=0;i<n;i++){
 document.writeln(a[i]);
};
```

3 8 2 4 6

8 6 4 3 2

Приклад 12.1 Створити скрипт, який серед введених прізвищ знаходить однофамільців.

```
var n=+prompt('n=','');
for (var i=0;i<n;i++){
 a[i]=prompt('Введіть прізвище:','');
};
var o=[];
next:for (i=0;i<n;i++){
 for (var j=i+1;j<n;j++){
       if (a[i]==a[j] ) {      //знайшли однофамільця
            for (var l=0;l<o.length;l++){
                 if(a[i]==o[l]) continue next;  //він вже є в списку
однофамільців
            }; //l
            o[o.length]=a[i];  //додаємо до списку однофамільців
            continue next;
       }; //if
 };  // j
}; // i
if(o.length){alert(o)} else {alert('однофамільців нема')};
```

Приклад 12.2 **Знайти всі прості числа до 1000 та їх суму**

```
var c=[];
const n=1000;
for(var i=1;i<n;i++){c[i]=true};       //вважаємо всі числа простими
var p=2;
while (p<=Math.sqrt(n)){
 for (i=2*p; i<n; i=i+p){  //всі числа кратні p вважаємо не простими
       c[i]=false;
 };
 i=p+1; //шукаємо наступне просте число
 while (!c[i]){
       i++;
 };
 p=i;
};
var s=0;  //шукаємо суму
for (i=1;i<n;i++){
 if (c[i]){
       document.writeln(i);
       s=s+i;
 };
};
alert('Сума рівна'+s);
```

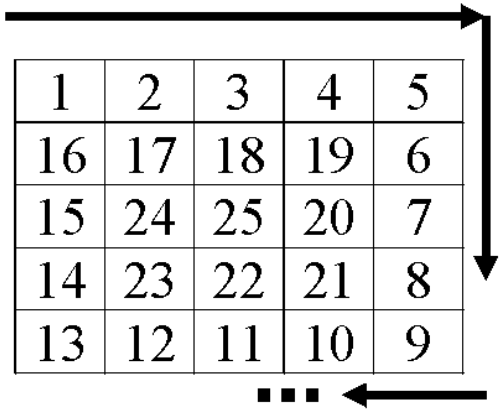
```
var matrix = [
        [1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]
];
```

alert (matrix [1][2]); // елемент в другому рядку, третьому стовпцю рівний 6

Якщо матрицю неможливо задати явно, то щоб інтерпретатор зрозумів з чим має справу перед її подальшою обробкою необхідно провести **ініціалізацію:** спочатку описати матрицю як масив, а потім кожному елементу присвоїти порожній масив.

```
var matrix=[]; // матриця розміром n*m
for(i=0;i<n;i++){
 matrix[i]=new Array(m); // або matrix[i]=[];
};
```

Приклад 12.3 Заповнити матрицю послідовністю натуральних чисел по спіралі(починаючи з верхнього лівого кута, за годинниковою стрілкою). Вивести результат у вигляді таблиці.



1	2	3	4	5
16	17	18	19	6
15	24	25	20	7
14	23	22	21	8
13	12	11	10	9

split(роздільник) дозволяє перетворити рядок в масив, розбивши її по роздільнику.

```
var names = 'Петя, Марина, Василь';
var arr = names.split (','); // arr = ['Петя', 'Марина', 'Василь']
for (var i = 0; i <arr.length; i + +) {
    alert (arr [i]);
}
```

Виклик str.split ('') розіб'є рядок на літери.

join (роздільник) Протилежний метод до split. Він склеює масив в рядок, використовуючи роздільник.

```
var arr = ['Петя', 'Марина', 'Василь'];
alert(arr.join(';')); //Петя;Марина;Василь
```

splice (позиція, [лічильник видалення, елемент1, елемент2, …, елементN]) - видаляє вказану кількість елементів, починаючи з позиції, а потім вставляє елементи на їх місце. Наступні за віддаленими елементами зсуваються, щоб заповнити їх місце. Метод splice повертає масив з видалених елементів.

var a=[0,1,2,3,4,5,6,7];

a.splice (1, 1); // видалити 1 елемент, починаючи з позиції 1, a=[0,2,3,4,5,6,7];

a.splice(-3,2); //видалити 2 елементи, починаючи з 3 від кінця позиції, a=[0,2,3,4,7]

a.splice(0, 3, 5, 5); // видалити 3 елементи на початку і додати дві 5, a=[5,5,4,7]

a.splice(3, 0, 1); // додати без видалення a=[5,5,4,1,7]

var b=a.splice (1, 2,'видалено 5,4'); //a=[5,'видалено 5,4',1,7]; b=[5,4];

slice (begin, end) копіює ділянку масиву від begin до end, не включаючи end. Вихідний масив при цьому не змінюється.

var arr = ["Чому", "треба", "вчити", "JavaScript"];

var a = arr.slice (1,3); / / елементи 1, 2 (не включаючи 3) a=["треба", "вчити"];

reverse() змінює порядок елементів у масиві на зворотний.

indexOf (searchElement [, fromIndex]) повертає номер елемента **searchElement** в масиві, або -1, якщо його немає. Пошук починається з номера fromIndex, якщо він зазначений. Якщо ні - з початку масиву.

```
var arr = [1, 0, false,1];
alert (arr.indexOf (0)); // 1
alert (arr.indexOf (false)); // 2
alert (arr.indexOf (null)); // -1
alert (arr.indexOf (1,2)); //3
```

lastIndexOf (searchElement [, fromIndex]) шукає останнє входження елементу, переглядаючи масив справа-наліво з кінця або з номера **fromIndex**, якщо він зазначений.

Метод **sort()** сортує масив. Він сортує в порядку зростання, перетворюючи елементи до рядкового типу. Це дає неправильні результати для числових масивів.

```
var arr = [1, 2, 15,36,305];
arr.sort ();
alert (arr); / / 1, 15, 2, 305, 36
```

sort (функція порівняння) вміє сортувати будь масиви, якщо вказати функцію порівняння від двох елементів, яка вміє порівнювати їх. Вона повинна повертати додатне значення, якщо перший елемент більший, від'ємне значення, якщо другий більший, якщо рівні – то 0.

```
function compareNumber (a, b) {
 if (a>b) {return 1};
 if (a<b) {return -1};
 return 0;
 //а можна просто return a-b
}
var arr = [1,2,15,36,305];
arr.sort (compareNumber);
alert (arr); / / 1, 2, 15, 36, 305
```

Приклад. Скласти скрипт для сортування масиву arr порядку спадання

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

forEach (функція) викликає функцію для кожного елемента масиву. Функція викликається з параметрами (item, i, arr): item – значення поточного елементу масиву; i - його номер; arr - масив, що перебирається. Зміна значення item не приводить до зміни відповідного елементу, для цього слід звертатись до arr[i].

Приклад. Скласти скрипт, що виводить кожен елемент масиву в окремий рядок у форматі a[i]=…

```
function print(x,i,arr){
 document.writeln('a['+i+']='+x+'<br>');
}
var a=[2,3,5];
a.forEach(print);
alert(a);
```

filter (callback) створює новий масив, в який увійдуть тільки ті елементи вихідного масиву, для яких виклик callback (item, i, arr) поверне true

Приклад. Скласти скрипт, що знаходить всі парні елементи на непарних місцях.

```
function condition(x,i,arr){
 return (i%2!=0)&&(x%2==0);
}
var b=a.filter(condition);
```

push (елемент) додає елемент в кінець.
pop() видаляє і повертає останній елемент.
unshift (елемент) додає елемент в початок.
shift() видаляє і повертає перший елемент.