### ========================= ТЕОРІЯ ================

### Масиви

На відміну від деяких мов програмування, масив в JavaScript є складним, впорядкованим, неоднорідним типом даних. Тобто, в масиві можна зберігати елементи різних типів. Кожен елемент характеризується своїм індексом – унікальним номером.

Синтаксис для створення нового масиву - квадратні дужки зі списком елементів усередині.  
var arr = []; //Порожній масив

var fruits = ["Яблуко", "Апельсин", "Слива"]; // Масив фруктів з трьома елементами

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

Враховуючи, що масив є об'єктом, його можна створити за допомогою конструктора new Array(). Якщо в як параметр передати одне ціле число то буде створено порожній масив відповідної довжини, в решті випадків буде створено масив, що складається з аргументів конструктора. Виклик конструктора із від’ємним аргументом приводить до помилки.

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

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

arr=new Array(-3); // Помилка

Через alert можна вивести масив цілком. При цьому значення його елементів будуть перераховані через кому:

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

Загальне число елементів, збережених в масиві, міститься в його властивості **length**:

alert (fruits.length); //3

Зазвичай нам не потрібно самостійно міняти length. При зменшенні length масив вкорочується. Причому цей процес необоротний, тобто навіть якщо потім повернути length назад - значення не відновляться. Найпростіший спосіб очистити масив - присвоїти властивості length нульове значення.

Елементи нумеруються, починаючи з нуля. Щоб отримати потрібний елемент з масиву - вказується його індекс у квадратних дужках:

alert (fruits [0]); // Яблуко

alert (fruits [2]); // Слива

Елемент можна завжди замінити:

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

Або додати:

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

Слід врахувати, що 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

Приклад роботи з масивами.

<script>

var a = []; //Спочатку масив порожній

//-------- Генерування елементів масиву ----

for (var i = 0; i < a.length; i++) {

a[i] = 0.5-Math.random();

}

//--------- Знайти суму усіх ------------

var s = 0;

for (var i = 0; i < a.length; i++) {

s += a[i];

}

document.write("sum=" + s);

//----------

var s = 0;

for (var i = a.length/2; i < a.length; i++) // Як змінюється номер елемента

{ //Що робити з кожним елеменом з таким номером

s += a[i];

}

document.write("sum=" + s);

//----------- СУМА ДО ПЕРШОГО ДОДАТНЬОГО -----------

//------ Перший спосіб ----

var s = 0;

var i = 0;

while (i < a.length && a[i]<=0) {

s += a[i];

i++;

}

//------ Другий спосіб -----

var s = 0;

for (var i = 0; i < a.length; i++) {

if (a[i] > 0)

break;

s += a[i];

}

//------- Третій спосіб -----

var s = 0;

for (var i = 0; i < a.length && a[i]<=0; i++) {

s += a[i];

}

</script>

Приклад.

Задача 1. Дано послідовність платіжок протягом року. Знайти сумарну кількість грошей за:

1. за весь рік;
2. у першій половині року;
3. у другій половині року;
4. за літо;
5. за ІІ квартал;
6. за парні місяці (з парними номерами);
7. за місяці, які є початковими у сезоні (весна, літо, осінь, зима).

<script>

var paymants = [100, 210, 300, 23, 45, 233, 100, 210, 300, 23, 45, 233];

//1 за весь рік (суму усіх елементів)

var sumYear = 0;

for (var i = 0; i < paymants.length; i++) {

sumYear += paymants[i];

}

document.write("Summ per year =" + sumYear);

//2 у першій половині року;

var sumFirstYearHalf = 0;

var t = paymants.length / 2;

for (var i = 0; i < t; i++) {

sumFirstYearHalf += paymants[i];

}

document.write("Summ per first half of year =" + sumFirstYearHalf);

//3 у другій половині року;

var sumSecondHalfOfYear = 0

t = paymants.length / 2;

for (var i = t; i < paymants.length; i++) {

sumSecondHalfOfYear += paymants[i];

}

document.write("Summ per second half of year =" + sumSecondHalfOfYear);

//4) за літо;

var sumSummer = 0;

for (var i = 5; i <= 7; i++) {

sumSummer += paymants[i];

}

document.write("Summ per summer =" + sumSummer);

//5) за ІІ квартал.

var sumKv2 = 0;

for (var i = 3; i <= 5; i++) {

sumKv2 += paymants[i];

}

document.write("Summ per Kv2 =" + sumKv2);

//6) за парні місяці (оскільки нумерація починається від 0, то нам треба розглядати непарні місяці)

var sumP = 0;

for (var i = 1; i <paymants.length; i+=2) {

sumP += paymants[i];

}

document.write("SummP =" + sumP);

//7) за місяці, які є початковими у сезоні (весна, літо, осінь, зима)

var sumPS = 0;

for (var i = 2; i < paymants.length; i += 3) {

sumPS += paymants[i];

}

document.write("SummPS =" + sumPS);

</script>

Задача 2. Дано послідовність оцінок учня. Підрахувати кількість:

1. двійок
2. кількість хороших оцінок (добре, відмінно);
3. кількість оцінок, які нижче середнього.

<script>

//Задача 2. Дано послідовність оцінок учня. Підрахувати кількість:

//1) двійок

//2) кількість хороших оцінок (добре, відмінно);

//3) кількість оцінок, які нижче середнього.

//Спитаємо скільки оцінок

var scoresCount = parseInt(prompt("Scores count =", "5"));

//Створимо масив для оцінок

var scores = new Array(scoresCount);

//Введемо оцінки учня

for (var i = 0; i < scores.length; i++) {

scores[i] = parseInt( prompt("Scores №"+(i+1) ,"0") );

}

//Рахуємо кількість двійок

var count2 = 0;

for (var i = 0; i < scores.length; i++) { //Як змінюються номери елементів "і"

if (scores[i] == 2) //Що треба зробити з кожним елементом з номером "і",тобто "scores[i]"

count2++;

}

document.write("Count 2 =" + count2);

//2) кількість хороших оцінок (добре, відмінно);

var countGood = 0;

for (var i = 0; i < scores.length; i++) { //Як змінюються номери елементів "і"

if (scores[i]>=4) //Що треба зробити з кожним елементом з номером "і",тобто "scores[i]"

countGood++;

}

document.write("Count good =" + countGood);

//3) кількість оцінок, які нижче середнього.

//---------- Знаходимо суму ----

var sum = 0;

for (var i = 0; i < scores.length; i++) {

sum += scores[i];

}

var average = sum / scores.length;

//---------- Рахуємо кількість тих, які більші за середнє

var countLessThanAversge = 0;

for (var i = 0; i < scores.length; i++) {

if (scores[i]<average) {

countLessThanAversge++;

}

}

document.write("Count less than aversge ="+countLessThanAversge);

</script>

Можна присвоювати цілі масиви, але необхідно врахувати що при цьому відбувається копіювання вказівника на масив, а не значень масиву. Це може викликати непередбачувані наслідки, тому доводиться копіювати масиви поелементно:

var a=[1,2,3],b=a;

b[1]='Ой!';

alert(a[1]); // Ой!, а не 2, як очікувалось

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 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 2 Знайти всі прості числа до 1000 та їх суму за допомогою решета Ерастрофена.

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 р 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);

Масиви в JavaScript можуть містити в якості елементів інші масиви. Цим можна скористатись для створення багатовимірних масивів, наприклад матриць. Звертатись до елемента необхідно беручи кожен індекс в квадратні дужки.

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 ініціацію: 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 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 |

var n=+prompt('Розмірність',''),i,j,l,e=1;

var m=[]; //ініція матриці

for(i=0;i<n;i++){

m[i]=[];

};

for ( l=0;l<=(n-1)/2;l++){

for (j=l;j<n-l;j++){m[l][j]=(e++)}; //перший рядок

for (i=l+1;i<n-l;i++){m[i][n-l-1]=(e++)}; //останній стовпець

for (j=n-l-2;j>=l;j--){m[n-l-1][j]=(e++)}; //останній рядок

for (i=n-l-2;i>l;i--){m[i][l]=(e++)}; //перший стовпець

};

//вивід таблиці за допомогою тегів

document.writeln('<table border="2" cellspacing="0">');

for (i=0;i<n;i++){

document.write('<tr>');

for (j=0;j<n;j++){

document.write('<td>'+m[i][j]+'</td>');

};

document.writeln('</tr>');

}; document.writeln('</table>');

=================== ПРИКЛАДИ ==================

Задача 1. Дано послідовність платіжок протягом року. Знайти сумарну кількість грошей за:

1. у першій половині року;
2. у другій половині року;
3. за літо;
4. за ІІ квартал.

Задача 2. Дано послідовність оцінок учня. Підрахувати кількість:

1. двійок
2. кількість хороших оцінок (добре, відмінно);
3. кількість оцінок, які нижче середнього.

Задача 3. Дано послідовність цін товарів та назв (у окремих масивах). Вивести на екран ті, які може собі дозволити користувач (кількість грошей задається).

Індивідуальні завдання

Описати літерал, що зберігає \_\_\_\_ та вивести його.

1. оцінки за екзамен в національній системі (відмінно, добре, …).
2. дні тижня.
3. місяці року.
4. жіночі імена (щонайменше 7 шт.)
5. чоловічі імена (щонайменше 7 шт.)
6. кольори веселки.
7. прізвища студентів Вашої групи.
8. прізвища студентів Вашої групи.
9. назви фруктів (щонайменше 7 шт.)
10. назви овочів (щонайменше 7 шт.)
11. оцінки за екзамен в системі ECTS (A,B,…).
12. Дано масив розмірності N. Назвемо серією групу однакових елементів що йдуть поспіль, а довжиною серії - кількість цих елементів (довжина серії може дорівнювати 0. Вивести масив, що містить довжини всіх серій вихідного масиву.
13. Дано масив розмірності N. Назвемо серією групу однакових елементів що йдуть поспіль, а довжиною серії - кількість цих елементів (довжина серії може дорівнювати 0. Перетворити масив так, щоб замість кожної серії йшов один елемент, що утворює серію, а її довжину.
14. Дано цілочисловий масив розмірності N. Якщо він є перестановкою, то вивести ТАК, у противному випадку вивести номер першого недопустимого елемента.
15. Дано масив розмірності N і число k ( k<N). Здійснити циклічний зсув елементів масиву вліво на k позицій. Наприклад для масиву [1,2,3,4,5,6] та k=2 після виконання зсуву отримуємо [3,4,5,6,1,2].
16. Дано числовий масив розмірності N. Знайти кількість його локальних мінімумів. Локальним мінімумом називається елемент, значення якого менше від сусідів
17. Дано масив розмірності N. Визначити максимальну кількість його однакових елементів.
18. Дано масив розмірності N і число k ( k <N). Здійснити циклічний зсув елементів масиву вправо на k позицій. Наприклад для масиву [1,2,3,4,5,6] та k=2 після виконання зсуву отримуємо [5,6,1,2,3,4].
19. Дано числовий масив розмірності N. Вважати, що його елементи утворюють числову послідовність, утворити новий масив що містить найдовшу зростаючу підпослідовність.
20. Дано числовий масив розмірності N. Знайти кількість його локальних максимумів.
21. Дано масив розмірності N. Назвемо серією групу однакових елементів що йдуть поспіль, а довжиною серії - кількість цих елементів (довжина серії може дорівнювати. Додати після кожної серії елемент з текстом «серія k», де k – порядковой номер серії
22. Дано числовий масив розмірності N. Знайти кількість елементі, що більші за перший.

**Методи для робото з масивами**

Масиви є об’єктами і мають численні методи для полегшення роботи з ними.

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;Василий

splice (позиція, [лічильник видалення, елемент1, елемент2, …, елементN]) - універсальний метод для роботи з масивами. Вміє все: видаляти елементи, вставляти елементи, замінювати елементи - по черзі і одночасно. Він видаляє вказану кількість елементів, починаючи з позиції, а потім вставляє елементи на їх місце. Наступні за видаленими елементами зсуваються, щоб заповнити їх місце. Метод splice повертає масив з видалених елементів. Допускається використання від'ємного номера позиції, яка в цьому випадку відраховується з кінця.

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. Для пошуку використовується суворе порівняння ===.

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]) має аналогічні параметри, але 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

Методу sort (функція порівняння) вміє сортувати будь масиви, якщо вказати функцію fn 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});

Приклад. Скласти скрипт для сортування масиву arr в порядку, коли спочатку йдуть всі парні чила, потім – непарні.

function evenCompare (a, b) {

  // Числа однієї парності - сортуються звичайним чином

if (a%2 == b%2) return a - b;

// Інакше, якщо a - парне, то воно менше

if (a% 2 == 0) return -1;

// Лишився один варіант: a - непарне, і b – парне

return 1;

}

var arr = [5, 2, 1, -10, 8];

arr.sort (evenCompare);

alert (arr); / / -10, 2, 8, 1, 5

Використання випадкових чисел для функції порівння дозволить перемішати елементи масиву випадковим чином. Math.random() повертає результат що належить [0,1).

var arr = [1, 2, 3, 4, 5];

arr.sort (function () { return Math.random () - 0.5;});

alert (arr); / / елементи у випадковому порядку, наприклад [3,5,1,2,4]

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);

map (callback) створює новий масив, який складатиметься з результатів виклику callback (item, i, arr) для кожного елемента arr.

Приклад. Скласти скрипт, що збільшує кожен елемент масиву на його індекс;

function change(x,i,arr){

return x+i;

}

alert(a.map(change));

every (callback) повертає true, якщо виклик callback поверне true для кожного елемента масива.

some (callback) повертає true, якщо виклик callback поверне true для якогось елемента масива.

Приклад. Чи всі елементи массиву додатні, чи є хоча б один додатній.

var arr = [1, -1, 2, -2, 3];

function isPositive (number) {return number> 0};

alert (arr.every (isPositive)); / / false, не всі додатні

alert (arr.some (isPositive)); / / true, є хоч одне додатне

reduce (reduceCallback [, initialValue]) застосовує функцію reduceCallback по черзі до кожного елементу масиву зліва направо, зберігаючи при цьому проміжний результат.  
Аргументи функції reduceCallback (previousValue, currentItem, index, arr):  
     previousValue - останній результат виклику функції, він же проміжний результат.  
     currentItem - поточний елемент масиву, елементи перебираються по черзі зліва-направо.  
     index - номер поточного елемента.  
     arr - масив.  
Значення previousValue при першому виклику одно initialValue. Якщо initialValue немає, то воно дорівнює першому елементу масиву, а перебір починається з другого.

Приклад. Скласти скрипт для знаходження суми всіх елементів та кількості додатних елементів в масиві.

function kilk(result,x,i,arr){

if (x>0) {result++};

return result;

}

function suma(result,x,i,arr){

result+=x;

return result;

}

var a=[1,2,-3,4,-5];

alert(a.reduce(kilk,0));

alert(a.reduce(suma));