Створення масиву

|  |  |  |
| --- | --- | --- |
| Зміст | Загальний вигляд | Приклад |
| створення порожнього масиву | назва\_масиву = [ ]  назва\_масиву = new Array(); | var arr = []; //Порожній масив  var t = new Array(); //Порожній масив |
| створення за допомогою літерала (вказуємо список значень) | назва\_масиву = [ список\_значень ] | var fruits = ["Яблуко", "Апельсин", "Слива"]; |
| за допомогою конструктора (створюється масив, усі елементи якого спочатку undefined) | назва = new Array( кількість\_елементів ) | arr=new Array(3); // arr = [,,] |
| за допомогою конструктора (вказуємо значення усіх елементів) | назва = new Array( список значень ) | var arr = new Array(2,3); // еквівалентно arr = [2, 3] |

Визначення чи є масивом

|  |  |  |
| --- | --- | --- |
| Зміст | Загальний вигляд | Приклад |
| дозволяє з’ясувати, чи є змінна масивом | Array.isArray( об’єкт\_масиву ) | if ( Array.isArray(fruits ) )  alert(“fruits є масивом”); |

Властивості масиву

|  |  |  |
| --- | --- | --- |
| Зміст | Загальний вигляд | Приклад |
| кількість елементів | назва\_масиву . **length** | var fruits = ["Яблуко", "Апельсин", "Слива"];  alert (fruits.**length**); //3 |
|  |  |  |
|  |  |  |

Доступ до елементів масиву

|  |  |  |
| --- | --- | --- |
| Зміст | Загальний вигляд | Приклад |
| зчитування значення  (у квадратних дужках необхідно вказати індекс у квадратних дужках) | змінна = назва\_масиву [ **індекс** ] | var fruits = ["Яблуко", "Апельсин", "Слива"];  alert (fruits **[0]**); // Яблуко  alert (fruits **[2]**); // Слива |
| зміна значення | назва\_масиву [ **індекс** ] = значення | var fruits = ["Яблуко", "Апельсин", "Слива"];  fruits **[0]**= “Ківі”**;**  //fruits = ["**Ківі**", "Апельсин", "Слива"]; |
|  |  |  |

Додавання елементів

|  |  |  |
| --- | --- | --- |
| Зміст | Загальний вигляд | Приклад |
| додавання в кінець |  | var newLength = fruits.push('Апельсин'); |
| додавання на початок |  | var newLength = fruits.unshift('Клубника') |
| вставка в середину |  | var removedItems = fruits.splice(pos, 0, “Груша”); |

Видалення елементів

|  |  |  |
| --- | --- | --- |
| Зміст | Загальний вигляд | Приклад |
| видалення з кінця | видалений\_елемент = назва\_масиву . pop() | var last = fruits.pop(); |
| видалення з початку | видалений\_елемент = назва\_масиву . shift() | var first = fruits.shift(); |
| видалення одного елемента у вказаній позиції | видалений\_елемент = назва\_масиву . splice(pos, 1)  pos - позиція елемента, який треба видалити | var removedItem = fruits.splice(pos, 1) |
| видалення декількох елементів починаючи з вказаної позиції | видалені\_елементи = назва\_масиву . splice(pos, n)  pos – початкова позиція, n – кількість елементів для видалення | var removedItems = vegetables.splice(pos, n); |
| видалення існуючих і вставка нових елементів | arr.splice(pos[, deleteCount, elem1, ..., elemN])  pos – позиція, з якої починається видалення  deleteCount – кількість елементів, які потрібно видалити  elem1, ..., elemN – список елементів, які потрібно вставити | var arr = ["Я", "зараз", "вивчаю", "JavaScript"];  // Видалитит 3 перших елементи і замість них додати слова : "Ми", "вивчаємо"  arr.splice(0, 3, "Ми", "вивчаємо")  alert( arr ) // ["Ми", "вивчаємо", "JavaScript"] |
| видалення значення елемента (сам елемент залишається і дорівнює undefined) | delete назва\_масиву [ індекс ] | var arr = ["Я", "йду", "додому"];  **delete** arr[1]; // видалити значення з інексом 1  alert( arr); // arr = ["Я", **undefined**, "додому"]; |
| використання властивості length  (якщо властивості length присвоїти значення, яке є меншим за справжню довжину масиву, то масив зменшується до вказаної кількості елементів) | назва\_масиву . length = нова\_довжина\_масиву | var arr = ["Я", "зараз", "вивчаю", "JavaScript"];  arr.**length** = 2; //довжина=4, а ми присвоюємо 2  alert( arr ) // ["Я", "зараз"] |

Пошук елементів

|  |  |  |
| --- | --- | --- |
| Зміст | Загальний вигляд | Приклад |
| знаходження індексу першого входження елемента (повертає -1, якщо немає елемента) | назва\_масиву . indexOf() | var fruits = ["Яблуко", "Апельсин", "Слива"];  var pos = fruits.indexOf("Слива”);  alert(pos); // 2  pos = fruits.indexOf('Банан');  alert(pos); // -1 (елементу немає в масиві) |
| знаходження індексу останнього входження (пошук з кінця) | назва\_масиву . lastIndexOf () | var a=[ 11, 55,23, 11, 90]  var pos = a. lastIndexOf (11);  alert(a); // 3 (індекс останнього входження 11) |

Копіювання частини елементів

|  |  |  |
| --- | --- | --- |
| Зміст | Загальний вигляд | Приклад |
| копіювання всього масиву | назва\_масиву . slice () | var shallowCopy = fruits.slice(); |
| копіювання частини масиву | назва\_масиву . slice ( start, end )  start – індекс початку копіювання  end – індекс кінцевого елементу (до якого елемента буде здійснюватися копіювання; сам елемент не входить до скопійованої частини) | var a = [25, 12, 18, 46, 35, 26];  var t = a.slice(2, 4);  alert(t); // [18, 46] |

Інші методи

|  |  |  |
| --- | --- | --- |
| Зміст | Загальний вигляд | Приклад |
| обертання масиву (елементи будуть розміщені у зворотному порядку) | назва\_масиву .reverse() | var a = [25, 12, 18, 46, 35, 26];  a.reverse();  document.write(a); // [26,35,46,18,12,25] |
| об’єднання масиву  (повертає **новий масив**, який складається з елементів даного масиву і масиву, з яким виконується об’єднання) | масив1 . concat(масив2 )  масив . concat(список\_нових\_масивів )  масив . concat( список\_нових\_елементів ) | var a1 = [1, 2, 3];  var a2 = [4, 5];  var a3 = a1.concat(a2);  document.write(a3); // 1,2,3,4,5  //--------------------------  var a1 = [1, 2, 3];  var a2 = [4, 5];  var a3 = [6,7];  var a4 = a1.concat(a2,a3);  document.write(a4); // 1,2,3,4,5,6,7  //--------------------------  var a1 = [1, 2, 3];  var a2 = a1.concat(4,5);  document.write(a2); // 1,2,3,4,5 |
| створення рядка з елементів масиву  (елементи розділяються вказаним знаком-розділювачем) | масив . join( символ-розділювач )  масив . join() //символом розділювачем є кома | var arr = ['Марія', 'Петро', 'Марина', 'Василь'];  var str = arr.**join(';')**;  alert( str ); // "Марія;Петро;Марина;Василь"  //-------------------------------------  var arr = ['Марія', 'Петро', 'Марина', 'Василь'];  var str = arr.join();  alert(str); // Марія,Петро,Марина,Василь |
| створення масиву шляхом розбиття рядка, у якому елементи розділені деяким символом-розділювачем |  | var arr=” Маша;Петя;Марина;Василий”.split(“;”) |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Призначення** | **Що повертає** | **Особливості функції-колбека** | **Приклад** |
| **sort** | упорядкування елементів за вказаним правилом | змінюється сам масив | function compareRule(item1, item2)  {  ...  }  Функція-колбек задає правило попарного порівняння елементів під час сортування і повинна повертати:   * число більше за нуль, якщо item1>item2; * число менше за нуль, якщо item1<item2; * нуль, якщо item1 = item2; | //--------- без правила порівняння ---  var a = [12, 3, -21, 4, -56, 4, 3];  a.sort(); //елементи порівнюються як string  alert(a); // -21,-56,12,3,3,4,4  //--задаємо правило порівняння (за зростанням)-  function rule(item1, item2) {  if (item1 > item2)  return 1;  else  if (item1 < item2)  return -1;  else  return 0;  }  var a = [12, 3, -21, 4, -56, 4, 3];  a.sort();  alert(a); // 56,-21,3,3,4,4,12  //-- З використанням лямбда-виразів ---  var a = [12, 3, -21, 4, -56, 4, 3];  a.sort((a, b) => a - b);  alert(a); |
| **ForEach** | Коли треба переглянути елементи не змінюючи їх, або треба змінити елементи існуючого масиву (новий масив не формуємо). Елементи перебираються по черзі зліва-направо. | Нічого не повертає | function callback(itemCopy, index, array)  { . . .}  itemCopy – копія поточного елемента масиву (окрема змінна); index – номер поточного елемента; array – посилання на масив, для якого викликано метод.  Функція-колбек нічого не повертає.  Якщо хочемо змінювати елементи використовуємо посилання на масив (array[index]= нове\_значення) | Вивести парні елементи.  var a = [1, 2, 3, 4];  function isEven(x, i, arr) {  if (x % 2 == 0)  alert(x);  }  a**.forEach(isEven);**  Усі елементи помножити на 2  var a = [1, 2, 3];  a.**forEach((x, i, arr) => { arr[i] \*= 2;})**  alert(arr); // [2,4,6] |
| **Map** | Коли треба сформувати новий масив на основі існуючого (кількість елементів нового масиву співпадає з кількістю існуючого) | Повертає новий масив, який сформовано на основі даного (правило утворення елементів нового масиву задає функція-колбек) | function callback(itemCopy, index, array)  {  . . .  return **обчислене\_значення\_елемента\_нового\_масиву**  }  Функція-колбек задає правило утворення елементів нового масиву і повертає обчислене значення елемента нового масиву на основі поточного значення itemCopy початкового масиву | Створити новий масив, у якому до кожного елемента додано 20  function rule(x, i, arr) {  return x + 20;  }  var a = [12, 3, -21, 4, -56, 4, 3];  var newArr=a.map(rule);  alert(newArr); //32,23,-1,24,-36,24,23  Створити новий масив, у якому усі елементи помножено на 7  var a = [1, 2, 3];  var newArr=a.map((x)=>x\*7);  alert(newArr); //7,14,21 |
| **Filter** | Коли треба сформувати новий масив з елементів даного масиву, які задовольняють деякій умові (тобто елементи існуючого або входять у новий масив, або не входять) | Повертає масив елементів, які задовольняють заданій умові | function callback(itemCopy, index, array)  {  . . .  return **результат\_перевірки\_умови\_фільтрації**  }  Фунція-кобек повертає True (коли елемент підходить за фільтром) або False (коли елемент не підходить за фільтром) | Дано масив цін. Сформувати масив цін, які більші за 100  function rule(x, i, arr) {  return x > 100;  }  var a = [450, 13, 23, 900];  var newArr = a.filter(rule);  alert(newArr); // 450,900  //-------------------------------------  Дано масив років народження працівників. Вивести тих, які народилися між 1980 і 2003 роками.  var a = [1990, 2017, 2006, 2001];  var newArr = a.filter((x)=> x>=1980&&x<=2003);  alert(newArr); // 1990,2001 |
| **Every** | Коли треба перевірити чи для усіх елементів масиву виконується деяка умова (яку записуємо у функції) | every повертає true, якщо усі елементи задовольняють заданій умові | function callback(itemCopy, index, array)  {  . . .  return **результат\_перевірки\_умови**  }  Фунція-кобек повертає True або False (у функції повертаємо результат перевірки умови) | Дано масив оцінок учня. З’ясувати, чи є учень двійочником.  function isNotBad(mark) { // чи не двійка  return mark > 3;  }  var marks = [8, 11, 9, 10, 9];  if (marks.**every(isNotBad)**)  alert("Не двійочник");  else  alert("Двійочник"); |
| **Some** | Коли треба перевірити чи хоча би для одного з елементів виконується умова (яку записуємо у функції) | some повертає true, якщо хоча би один елемент задовольняє заданій умові | function callback(itemCopy, index, array)  {  . . .  return **результат\_перевірки\_умови**  }  Фунція-кобек повертає True або False (у функції повертаємо результат перевірки умови) | Дано масив оцінок учня. З’ясувати, чи є у нього оцінки 10.  var marks = [8, 10, 9, 11, 9];  if (marks.**some((mark)=>mark==10)**)  alert("Є оцінка 10");  else  alert("Немає оцінки 10"); |
| **reduce** | reduce(callback [, initialValue])  послідовно викликає функцію callback один раз для кожного елемента масиву (за виключенням порожніх елементів undefined), обчислюючи при цьому деяку величину. Значення величини при цьому поступово (зліва-направо) обраховується (накопичується) на основі кожного елемента масиву. Наприклад, при знаходженні суми до величини суми поступово додається кожен елемент масиву. | повертає останнє обчислене значення акумульованої величини | function callback(**previousValue**,itemCopy,index,array)  {  . . .  }  previousValue – результат виклику функції для попереднього елемента (проміжковий результат акумульованої величини). previousValue при першому виклику дорівнює;  initialValue. Якщо initialValue не вказано, то previousValue спочатку дорівнює першому елементу масиву, а перебір починається з другого | Знайти кількість додатних  function kilk(result, x, i, arr) {  if (x > 0) { result++ };  return result;  }  var a = [1, 2, -3, 4, -5];  alert(a.**reduce**(kilk, **0**)); //спочатку result=0  //-------------------------------  Знайти суму елементів масиву  function suma(result, x, i, arr) {  result += x;  return result;  }  var a = [1, 2, -3, 4, -5];  alert(a.**reduce**(suma)); //спочатку result=a[0] |
| **reduceRight** | reduceRight( callback [, initialValue])  послідовно викликає функцію callback один раз для кожного елемента масиву (за виключенням порожніх елементів undefined), обчислюючи при цьому деяку величину. Значення величини при цьому поступово (зліва-направо) обраховується (накопичується) на основі кожного елемента масиву. Наприклад, при знаходженні суми до величини суми поступово додається кожен елемент масиву. | повертає останнє обчислене значення акумульованої величини | function callback(**previousValue**,itemCopy,index,array)  {  . . .  }  previousValue – результат виклику функції для попереднього елемента (проміжковий результат акумульованої величини). previousValue при першому виклику дорівнює;  initialValue. Якщо initialValue не вказано, то previousValue спочатку дорівнює першому елементу масиву, а перебір починається з другого | Початкове значення суми дорівнює останньому елементу. Поступово розглядаючи елементи справа-наліво додаємо наступний елемент тільки у тому випадку, якщо його значення менше за поточне значення суми.  var a = [1, 2, 3];  alert(a.reduceRight(  (currentResult, x) =>  x < currentResult ?  currentResult + x :  currentResult)  ); //6 |

Посилання на довідник з методів: <https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array>

Приклад. Упорядквати за спаданням

function f(a,b) {

if (a > b) {

return -1;

}

if (a < b) {

return 1;

}

return 0;

}

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

var a = ["Ivan", "Petro", "Anton", "Sudor"];

a.sort(f);

document.write(a);

Приклад. Упорядкувати за зростанням за другою буквою

function f(a,b) {

if (a[1] > b[1]) {

return 1;

}

if (a[1] < b[1]) {

return -1;

}

return 0;

}

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

var a = ["Ivan", "Petro", "Anton", "Sudor"];

a.sort(f);

document.write(a);

//Задача.Кожне парне число збільшити на 5.

function f(x, i, arr) {

if (arr[i]%2==0) {

arr[i] += 5;

}

}

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

var a = [2, 11, 3, 4, 24, 3];

a.forEach(f);

document.write(a);

//Задача. Дано масив років народження працівників.

//Вивести тих, які народилися між 1980 і 2003 роках.

function f(x, i, arr) {

if (x >= 1980 && x <= 2003) {

document.write(x+", ");

}

}

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

var a = [1990,2001,1950,1967,2000];

a.forEach(f);

------------------

//Ця ж задача за допомогою filter

function f(x, i, arr) {

return (x >= 1980 && x < 2003);

}

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

var a = [1990, 2001, 1950, 1967, 2000];

var b = a.filter(f);

document.write(b);

//Задача.Дано масив імен працівників.

//Сформувати новий масив, який складається з

//перших літер імен.

function f(x, i, arr) {

return x[0];

}

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

var a = ["Ivan","Petro","Olja"];

var b=a.map(f);

document.write(b);

Задача. Дано масив показників термометра протягом місяця. З’ясувати, чи є серед них дні, коли температура була меншою за 5 градусів.

function f(x, i, arr) {

return x<5;

}

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

var a = [10, 15, 1, 4, 32];

if (a.some(f)) {

document.write("We have day with <5");

}

else {

document.write("All days >=5 ");

}

Задача. Дано масив показників термометра протягом місяця. З’ясувати, чи усі показники є більшими за 10 градусів.

function f(x, i, arr) {

return x>10;

}

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

var a = [10, 15, 1, 4, 32];

if (a.every(f)) {

document.write("All days >10");

}

else {

document.write("There is day <=10 ");

}

Задача. Знайти добуток від’ємних елементів масиву.

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

if (x<0) {

result \*= x;

}

return result;

}

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

var a = [17, 2, -3, 4, -5];

var c = a.reduce(f,1);

alert(c);

Задача. Знайти суму парних елементів

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

if (x%2==0) {

result += x;

}

return result;

}

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

var a = [17, 2, 3, 4, 5];

var c = a.reduce(f, 0);

document.write(c);