
Java Script
Массивы

Массивы

Массив – разновидность объекта, которая предназначена для хранения пронумерованных значений и предлагает дополнительные методы для удобного манипулирования такой коллекцией.

По сути, массив - это просто переменная, которая отличается тем, что в обычной переменной содержится одно значение, а в массиве может содержаться неограниченное кол-во значений, причём именно в JS они могут быть разного типа в одном массиве. То есть в одном массиве может содержаться элемент типа string, типа number и типа boolean одновременно.

Создание массива

Массив создаётся практически так же как и обычная привычная нам переменная.

Например, попробуем создать массив с именем arr (задавать ему имя array нельзя, так как это зарезервированное значение в языке Java Script)

```
<script>
  var arr = new Array();
</script>
```

Наполнение массива

В примере на предыдущем слайде мы создали массив arr. В том примере он абсолютно пустой.

Попробуем наполнить его, причём значениями разного типа, и вывести в нашем документе:

```
<script>
  var arr = new Array("hi", 2.5, 3, true);
  document.write(arr);
</script>
```

Элементы массива

Все элементы массива нумеруются автоматически, причем первый элемент является нулевым, а не первым элементом.
Давайте попробуем вывести первый и третий элемент нашего массива (для этого на самом деле нужно будет вывести элемент 0 и элемент 2):

```
<script>
  var arr = new Array("hi", 2.5, 3, true);
  document.write(arr[0] + "<br>" + arr[2]);
</script>
```

Результат: вывелось hi и 3.

Длина массива

Каждый массив имеет свою длину. Для того, чтобы определить длину массива, существует специальная функция.

Сейчас при помощи этой функции выведем в документе длину нашего массива:

```
<script>
  var arr = new Array("hi", 2.5, 3, true);
  document.write(arr.length);
</script>
```

Массивы и циклы

При помощи циклов мы можем работать с массивами. Например, выведем все значения массива:

```
<script>
  var arr = new Array("hi", 2.5, 3, true);
  for (var i = 0; i < arr.length; i++) {
  document.write(arr[i] + "<br>");
  }
</script>
```

Сортировка пузырьком

```
23      let myArr = [23, 2, 4, 6, 0, 1, 3, 45, 5, 6, 0, 3, 10, 11, 13, 12];
24      console.log(bubbleSort(myArr));
25
26      // 0 0 1 2 3 3 4 5 6 6 10 11 12 13 23 45
```

6 5 3 1 8 7 2 4

Создание массива при помощи цикла:

```
<script>
var arr = new Array();
for (var i = 0; i < 10; i++) {
  arr[i] = i * 3;
  document.write(arr[i] + "<br>");
}
</script>
```

Как посчитать сумму элементов массива:

```
<script>
var arr = new Array();
for (var i = 0; i < 4; i++) {
  arr[i] = i * 3;
}
var summ = 0;
for (i = 0; i < arr.length; i++) {
  summ += arr[i];
}
  document.write(summ);
</script>
```

Многомерные массивы

Многомерные массивы - это массивы, в которых каждым элементом массива является другой массив.

Создаём двумерный массив:

```
<script>
  var arr_1 = new Array();
  var arr_2 = new Array();
  var arr_3 = new Array();
  for(var i = 0; i < 5; i++) arr_1[i] = i;
  for(var i = 0; i < 10; i++) arr_2[i] = i;
  for(var i = 0; i < 15; i++) arr_3[i] = i;

  var arr = new Array(arr_1, arr_2, arr_3);
  console.log(arr);
</script>
```

Обращение к элементам многомерного массива:

```
<script>
  var arr_1 = new Array();
  var arr_2 = new Array();
  var arr_3 = new Array();
  for(var i = 0; i < 5; i++) arr_1[i] = i;
  for(var i = 0; i < 10; i++) arr_2[i] = i;
  for(var i = 0; i < 15; i++) arr_3[i] = i;

  var arr = new Array(arr_1, arr_2, arr_3);
  console.log(arr);
  document.write(arr[1][2]);
</script>
```

Как вывести многомерный массив?

IT SHATLE

```
<script>
  var arr_1 = new Array();
  var arr_2 = new Array();
  var arr_3 = new Array();
  for(var i = 0; i < 5; i++) arr_1[i] = i;
  for(var i = 0; i < 10; i++) arr_2[i] = i;
  for(var i = 0; i < 15; i++) arr_3[i] = i;

  var arr = new Array(arr_1, arr_2, arr_3);

for (var x = 0; x < arr.length; x++) {
  for (var j = 0; j < arr[x].length; j++) {
    document.write(arr[x][j] + "");
  }
  document.write("<br>")
}


</script>
```

Методы для работы с массивами

У всех объектов, в том числе и у массивов, есть методы toString () и valueOf( ) .

Метод toString (), будучи вызванный для массива, возвращает строку из строковых эквивалентов значений массива, разделенных запятыми.

Метод valueOf ( )- не изменяет исходное содержимое массива и возвращает все элементы массива.

Метод join

Метод **join** объединяет элементы массива в строку с указанным разделителем (он будет вставлен между элементами массива).

Разделитель задается параметром метода и не является обязательным. Если он не задан - по умолчанию в качестве разделителя возьмется **запятая**. Если вы хотите слить элементы массива **без разделителя** - укажите его как пустую строку ''

Методы push, pop, shift, unshift

Метод push () принимает любое количество аргументов и добавляет их в конец

массива, возвращая его новую длину. Метод pop () извлекает последний элемент

массива, уменьшает длину массива на 1 и возвращает элемент.

Метод shift() удаляет первый элемент массива, возвращая его и уменьшая длину массива на 1.

Метод u n s h i f t (), обратный методу shi ft (): он добавляет любое количество элементов в начало массива и возвращает его новую длину.

Методы изменения порядка следования элементов

Для изменения порядка следования элементов, уже находящихся в массиве, используются методы reverse( ) и sort(). Метод reverse () просто изменяет порядок следования элементов в массиве на обратный, например:

```
var values = [1, 2, 3, 4, 5 ] ;

values . reverse ();

alert (values); // 5,4, 3,2,1
```

Метод sort

По умолчанию метод sort () располагает элементы по возрастанию: наименьшее значение первым, а наибольшее последним. Чтобы отсортировать массив, он вы зывает функцию приведения типов String () для каждого элемента, а затем срав нивает возвращенные строки. Это происходит, даже если массив содержит только числа, например:

```
var values = [0, 1, 5, 10, 15] ;
values . sort();
alert (values); // 0,1, 10,15, 5
```

Методы манипулирования элементами

Над элементами массивов можно выполнять различные операции. Например, метод concat() позволяет создать новый массив на основе текущего. Сначала он создает копию массива, а затем добавляет аргументы в его конец и возвращает новый массив. Если метод concat () вызван без аргументов, он просто возвращает копию массива. Если передать в метод concat() один или несколько массивов, все их элементы будут добавлены в конец результата. Значения, которые не яв ляются массивами, просто добавляются в конец итогового массива. Рассмотрим пример:

Пример:

```
var colors = ["red", "green", " Ыlue" ];

var colors2 = colors . concat ( "yellow", [ "Ыlack" , " brown " ]);

alert (colors); // red , green, Ыlue

alert (colors2) ; // red, green, Ыlue, yellow, Ыlack,brown
```

Метод slice

Метод slice ( ) создает массив с одним или более элементами, уже содержащи мися в массиве. Он принимает один или два аргумента: начальную и конечную позиции элементов, которые нужно возвратить. Если аргумент только один, ме тод возвращает все элементы с этой позиции до конца массива. Если аргументов два, метод возвращает все элементы между начальной и конечной позициями, не включая конечный элемент. Эта операция никак не влияет на исходный массив.

Рассмотрим пример:

Пример:

```
var colors = ["red", " green", "Ыlue", "yellow" , " purple" ];

var colors2 = colors . slice(1) ;

var colors3 = colors . slice(1,4) ;

alert ( colors2) ;

alert ( colors3);
```

Самый мощный - метод splice

Самым мощным методом для работы с массивами является splice ( ) . Он используется в основном для вставки элементов в середину массива, но есть и два других способа его применения.

1. Удаление

Удаление. Из массива можно удалить любое количество элементов, указав позицию первого элемента, подлежащего удалению, и количество удаляемых элементов. Например, вызов splice (0, 2) удаляет первые два элемента.

2. Вставка

Вставка. Элементы можно вставить в массив в конкретной позиции, указав три или более аргументов: начальную позицию, 0 (количество удаляемых элементов) и элемент, который нужно вставить. С помощью четвертого, пятого и т. д. пара метров можно вставить дополнительные элементы. Например, вызов splice (2, 0, " red ", " green») вставляет в массив строки " red " и "green" начиная с позиции 2.

3. Замена

Замена. При вставке элементов в конкретной позиции можно одновременно удалить элементы, которые уже есть в массиве. Для этого нужно указать три или более аргументов: начальную позицию, количество удаляемых элементов и любое количество вставляемых элементов. Вставляемых элементов может быть больше или меньше, чем удаляемых. Например, вызов splice (2, 1, " red ", "green") удаляет один элемент в позиции 2, а затем вставляет в этой же позиции строки " red " и "green".

Удаление элемента:

```
var colors = ["red", " green", " Ыue" ]; // удаление первого элемента


var removed = colors . splice (0,1);

alert (colors); // greeп, Ыue

alert (removed); // red - массив с одним элементом
```

Вставка двух элементов

```
// вставка двух элементов в позиции 1

removed = colors . splice (1, 0, "yellow" , "orange" );

alert (colors); // greeп, yellow, orange, Ыue

alert (removed); // пустой массив
```

Вставка и удаление

```
// вставка двух значений и удаление одного

removed = colors . splice (l, 1, " red", " purple " );

alert (colors); // greeп, red, purple, oraпge, Ыue

alert (removed); // yellow - массив с одним элементом
```

Задание 1:

Создайте массив, длиной в 6 элементов, каждый из которого больше предыдущего на 5, после этого найдите среднее значение из этих чисел.

Задание 2.

Есть массив с неизвестным кол-вом элементов. Как вывести последний элемент?

Задание 3

У нас снова массив с неизвестным кол-вом элементов.

Напишите код, в котором предпоследнему элементу задается значение “предпоследний элемент”.

Задача 4:

Задача из 5 шагов-строк:

Создайте массив styles с элементами «Джаз», «Блюз».
Добавьте в конец значение «Рок-н-Ролл»
Замените предпоследнее значение с конца на «Классика». Код замены предпоследнего значения должен работать для массивов любой длины.
Удалите первое значение массива и выведите его alert.
Добавьте в начало значения «Рэп» и «Регги».
Массив в результате каждого шага:
Джаз, Блюз
Джаз, Блюз, Рок-н-Ролл
Джаз, Классика, Рок-н-Ролл
Классика, Рок-н-Ролл
Рэп, Регги, Классика, Рок-н-Ролл

Поиск элемента в массиве

Методы arr.indexOf, arr.lastIndexOf и arr.includes имеют одинаковый синтаксис и делают по сути то же самое, что и их строковые аналоги, но работают с элементами вместо символов:

- `arr.indexOf(item, from)` ищет `item`, начиная с индекса `from`, и возвращает индекс, на котором был найден искомый элемент, в противном случае `-1`;
- `arr.lastIndexOf(item, from)` – то же самое, но ищет справа налево.
- `arr.includes(item, from)` – ищет `item`, начиная с индекса `from`, и возвращает `true`, если поиск успешен.

find и findIndex

let result = arr.find(function(item, index, array) {}

Возвращает элемент, удовлетворяющий условию внутри функции.

```
let arr = [
    {
        name: 'Андрей',
        id: 1
    },
    {
        name: 'Саша',
        id: 2
    },
    {
        name: 'Коля',
        id: 3
    }
];
let result = arr.find(item => {
    return item.id === 2;
});

// {
//     name: 'Саша',
//     id: 2
// }
```

filter

let result = arr.filter(function(item, index, array) {}

Возвращает все элементы, удовлетворяющие условию внутри функции.

```
let arr = [1, 5, 334, 34, 12, 90];
let result = arr.filter(item => {
        return item > 10;
});
// 334, 34, 12, 90
```

forEach

arr.forEach(function(item, index, array) {}

```
let arr = [1, 5, 334, 34, 12, 90];
arr.forEach(item => {
    alert(item);
});
```

map

let result = arr.map(function(item, index, array) {}

Он вызывает функцию для каждого элемента массива и возвращает массив результатов выполнения этой функции.

```
let arr = [1, 5, 334, 34, 12, 90];
let result = arr.map(item => {
    return item + 10;
});
// 11, 15, 344, 44, 22, 100
```

reduce/reduceRight

let value = arr.reduce(function(previousValue, item, index, array) { // ...

}, [initial]);

Функция применяется по очереди ко всем элементам массива и «переносит» свой результат на следующий вызов

```
let arr = [1, 5, 334, 34, 12, 90];
let result = arr.reduce((sum, item) => {
    return sum + item;
});
// 476
```

Задание 1:

Существует массив из какого-то количества элементов строковых данных. Необходимо создать новый массив из длин каждого из элементов исходного массива.

Задание 2:

Существует цикл **for**, который перебирает массив со строковыми и числовыми данными **[“Anna”, 12, “Sam”, 9, “Kate”, 10, “Ron”, 9]** и выводит сначала строки, а затем числа:

```
for (let i = 0; i < arr.length; i++) {
    if (typeof arr[i] === 'string') {
        console.log(arr[i] + ' - string value');
    } else {
        console.log(arr[i] + ' - number value');
    }
}
```

Необходимо переписать данный цикл с помощью forEach() метода

Задание 3:

Существует массив [1, 4, 2, 67, 34, 2, 50, 23, 11, 10, 5, 4, 9, 21] . Необходимо создать новый массив из значений данного, которые больше 10.

Создайте калькулятор для введённых значений

Напишите код, который:

Запрашивает по очереди значения при помощи prompt и сохраняет их в массиве.
Заканчивает ввод, как только посетитель введёт 15.
При этом ноль 0 не должен заканчивать ввод, это разрешённое число.
Выводит сумму всех значений массива