

*** Масиви та методи
роботи з ними**

Масив - це різновид об'єкту, що зберігає в собі пронумеровані значення.

Масив можна задати за допомогою наступного синтаксису:

```
1 var arr = [];
```

За допомогою методу можна **typeof** переконатися, що масив це об'єкт.

```
> typeof arr  
< "object"  
>
```

Створимо масив з трьох елементів:

```
1 var fruits = ["Яблоко", "Апельсин", "Слива"];
```

Щоб отримати доступ до певного елемента масиву - вказуємо його номер у квадратних дужках. Елементи масиву нумеруються з нуля!!!

```
> var arr = [0,1,2,3]  
< undefined  
> arr[2]  
< 2  
> |
```

Загальна кількість елементів масиву визначається за допомогою методу `length`:

```
1 var fruits = ["Яблоко", "Апельсин", "Груша"];
2
3 alert( fruits.length ); // 3
```

Через `alert` можна вивести як одиничний елемент, так і весь масив.

```
1 var fruits = ["Яблоко", "Апельсин", "Груша"];
2
3 alert( fruits ); // Яблоко,Апельсин,Груша
```

```
> alert(arr[1])
< undefined
```

В масиві можна зберігати як числа, рядки та об'єкти

```
1 // микс значень
2 var arr = [ 1, 'Имя', { name: 'Петя' }, true ];
3
4 // получить объект из массива и тут же -- его свойство
5 alert( arr[2].name ); // Петя
```

Методи pop/push, shift/unshift

Метод **pop** видаляє останній елемент масиву і повертає його.

```
1 var fruits = ["Яблоко", "Апельсин", "Груша"];
2
3 alert( fruits.pop() ); // удалили "Груша"
4
5 alert( fruits ); // Яблоко, Апельсин
```

Метод **push** додає елемент в кінець масиву.

```
1 var fruits = ["Яблоко", "Апельсин"];
2
3 fruits.push("Груша");
4
5 alert( fruits ); // Яблоко, Апельсин, Груша
```

Метод **shift** видаляє перший елемент масиву і повертає його.

```
1 var fruits = ["Яблоко", "Апельсин", "Груша"];
2
3 alert( fruits.shift() ); // удалили Яблоко
4
5 alert( fruits ); // Апельсин, Груша
```

Метод **unshift** додає елемент в початок масиву.

```
1 var fruits = ["Апельсин", "Груша"];
2
3 fruits.unshift('Яблоко');
4
5 alert( fruits ); // Яблоко, Апельсин, Груша
```

Метод **unshift** та **push** можуть додати по декілька елементів.

```
1 var fruits = ["Яблоко"];
2
3 fruits.push("Апельсин", "Персик");
4 fruits.unshift("Ананас", "Лимон");
5
6 // результат: ["Ананас", "Лимон", "Яблоко", "Апельсин", "Персик"]
7 alert( fruits );
```

Вивід масиву з дірками.

То при виводі в більшості браузерів з'являються «зайві» коми.

```
1 var a = [];  
2 a[0] = 0;  
3 a[5] = 5;  
4  
5 alert( a ); // 0,,,,,5
```

Використовуємо **length** для укорочення масиву.

```
var arr = [1, 2, 3, 4, 5];  
  
arr.length = 2; // укоротить до 2 елементів  
alert( arr ); // [1, 2]
```

Метод `split`

Трансформує рядок у масив.

```
1 var names = 'Маша, Петя, Марина, Василь';  
2  
3 var arr = names.split(', ');  
4
```

Якщо вказати другий аргумент, то він вводить обмеження на кількість елементів в масиві.

```
1 alert( "a,b,c,d".split(',', 2) ); // a,b
```

За допомогою цього методу можна розбити рядок по буквам

```
1 var str = "тест";  
2  
3 alert( str.split('') ); // т,е,с,т
```

Метод `join`

Цей метод бере і склеює його в рядок, використовуючи `str` як роздільник.

```
1 var arr = ['Маша', 'Петя', 'Марина', 'Василий'];
2
3 var str = arr.join(';');
4
5 alert( str ); // Маша;Петя;Марина;Василий
```

Видалення елемента

Видалити елемент масиву можна за допомогою методу **`delete`**, але при видаленні утворилась дірка.

```
1 var arr = ["Я", "иду", "домой"];
2
3 delete arr[1]; // значение с индексом 1 удалено
4
5 // теперь arr = ["Я", undefined, "домой"];
6 alert( arr[1] ); // undefined
```


Метод `splice`

Починаючи з першої позиції видалимо 1 елемент.

```
1 var arr = ["Я", "изучаю", "JavaScript"];
2
3 arr.splice(1, 1); // начиная с позиции 1, удалить 1 элемент
4
5 alert( arr ); // осталось ["Я", "JavaScript"]
```

Видалимо три перших елемента і замінимо їх на інші.

```
1 var arr = ["Я", "сейчас", "изучаю", "JavaScript"];
2
3 // удалить 3 первых элемента и добавит другие вместо них
4 arr.splice(0, 3, "Мы", "изучаем")
5
6 alert( arr ) // теперь ["Мы", "изучаем", "JavaScript"]
```

Метод `splice` повертає масив з видалених елементів.

```
1 var arr = ["Я", "сейчас", "изучаю", "JavaScript"];
2
3 // удалить 2 первых элемента
4 var removed = arr.splice(0, 2);
5
6 alert( removed ); // "Я", "сейчас" <-- array of removed elements
```

Метод **Slice**

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

```
1 var arr = ["Почему", "надо", "учить", "JavaScript"];
2
3 var arr2 = arr.slice(1, 3); // элементы 1, 2 (не включая 3)
4
5 alert( arr2 ); // надо, учит
```

Сортування методом **sort(fn)**

Для вказівки свого порядку сортування в метод `arr.sort(fn)` потрібно передати функцію `fn` від двох елементів, яка вміє порівнювати їх. Якщо цю функцію не вказати, то елементи сортуються як рядки.

```
1 function compareNumeric(a, b) {
2   if (a > b) return 1;
3   if (a < b) return -1;
4 }
5
6 var arr = [ 1, 2, 15 ];
7
8 arr.sort(compareNumeric);
9
10 alert(arr); // 1, 2, 15
```

Метод **reverse**

Метод `arr.reverse ()` змінює порядок елементів в масиві на зворотний.

```
1 var arr = [1, 2, 3];
2 arr.reverse();
3
4 alert( arr ); // 3,2,1
```

Метод **concat**

Метод `arr.concat (value1, value2, ... valueN)` створює новий масив, в який копіюються елементи з `arr`, а також `value1, value2, ... valueN`.

```
1 var arr = [1, 2];
2 var newArr = arr.concat(3, 4);
3
4 alert( newArr ); // 1,2,3,4
```

Метод **indexOf**

Метод «`arr.indexOf (searchElement [, fromIndex])`» повертає номер елемента `searchElement` в масиві `arr` або `-1`, якщо його немає.

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