JavaScript Base

Занятие 3. Массивы и строки

IT Education Academy

Содержание

Что такое массив?

Создание массивов

Работа с массивами

Работа со строками

Методы масива

Что такое массив?

В программировании вообще:

Массив — набор однотипных элементов, расположенных в памяти непосредственно друг за другом, доступ к которым осуществляется по индексам.

В JavaScript’e:

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

Создание массивов

```
var arr = []; // Empty array
```

Массив fruits с тремя элементами:

```
var fruits = ["Apple", "Orange",
"Plum"];
```

Для доступа к элементам используется индекс. Индексы начинаются с `0` (нуля).

```
alert( fruits[0] ); // Apple
alert( fruits[1] ); // Orange
alert( fruits[2] ); // Plum
```

```
fruits[2] = 'Pear'; // now ["Apple",
"Orange", "Pear"]
fruits[3] = 'Lemon'; // now ["Apple",
"Orange", "Pear", "Lemon"]
```

Весь массив можно вывести с помошью **alert**.

```
alert( fruits ); //
Apple,Orange,Pear,Lemon
```

Длина массива (количество его элементов) содержится в свойстве **length**.

```
alert( fruits.length ); // 4
```

Особенности работы length

Длина **length** — не количество элементов массива, а последний индекс + 1.

```
var arr = [];
arr[1000] = true;

alert(arr.length); // 1001
```

Если элементы массива нумеруются случайно или с большими пропусками, то стоит подумать о том, чтобы использовать обычный объект. Массивы предназначены именно для работы с непрерывной упорядоченной коллекцией элементов.

Обычно нам не нужно самостоятельно менять **length**… Но при уменьшении **length** массив укорачивается.

Причем этот процесс необратимый, т.е. даже если потом вернуть **length** обратно — значения не восстановятся

```
var arr = [1, 2, 3, 4, 5];
arr.length = 2; // only 2 elems now
alert( arr ); // [1, 2]
arr.length = 5; // returning 5 elems
alert( arr[3] ); // undefined: there
is no values
```

Задача 1

1. Создайте массив из трех элементов: Яблоко, Груша, Апельсин
2. Выведите длинну массива
3. Вывидите второе значение из массива
4. Введите четвертое значение «Банан»
5. Вывидите четвертое значение из массива
6. Выведите длинну массива

Элементы массива

В массиве может храниться любое число элементов любого типа: строки, числа, объекты...

```
var arr = [ 1, 'String', { name: 'Ivan' }, true ];

alert( arr[2].name ); // Ivan

alert( arr ); // 1,String,[object Object],true
```

Основные операции с массивами

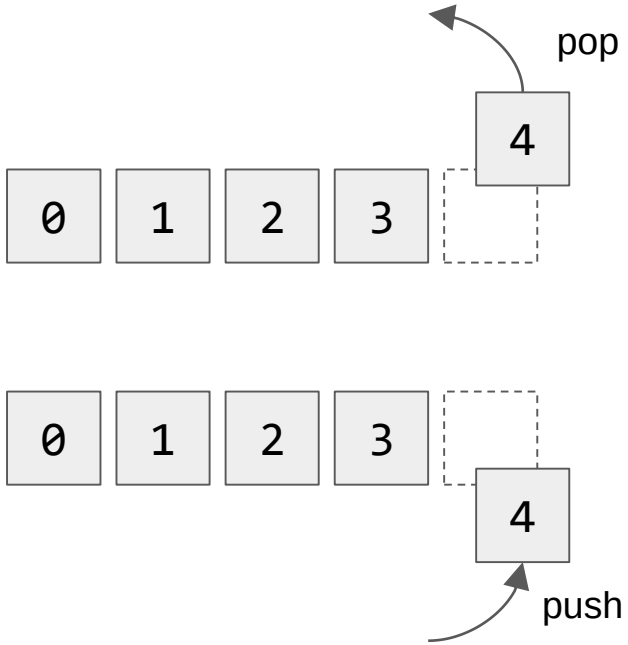
Конец массива

pop — удаляет *последний* элемент из массива и возвращает его.

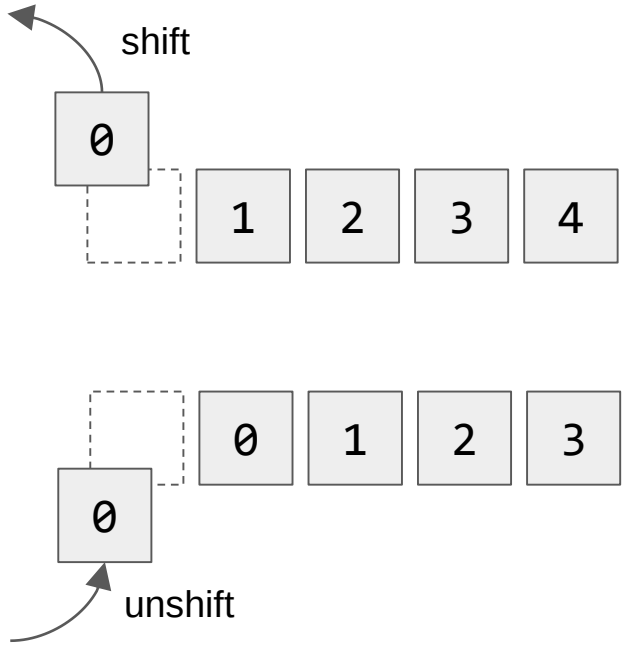
```
var fruits = ["Apple", "Orange",
"Pear"];
alert( fruits.pop() ); // "Pear"
alert( fruits ); // Apple, Orange
```

push — добавляет элемент в конец массива.

```
var fruits = ["Apple", "Orange"];
fruits.push("Pear");
alert( fruits ); // Apple, Orange,
Pear
```



Основные операции с массивами



Начало массива

shift — удаляет из массива первый элемент и возвращает его.

```
var fruits = ["Apple", "Orange",
"Pear"];
alert( fruits.shift() ); // Apple
alert( fruits ); // Orange, Pear
```

unshift — добавляет элемент в начало массива.

```
var fruits = ["Orange", "Pear"];
fruits.unshift("Apple");
alert( fruits ); // Apple, Orange,
Pear
```

Особенность push и unshift

Методы **push** и **unshift** могут добавлять несколько элементов

```
var fruits = ["Apple"];

fruits.push("Orange", "Peach");
fruits.unshift("Pineapple", "Lemon");

console.log( fruits ); // ["Pineapple", "Lemon", "Apple",
"Orange", "Peach"]
```

Задача 2

1. Создайте массив из трех элементов: Яблоко, Груша, Апельсин
2. Добавьте в конец значение «Персик»
3. Измените предпоследнее значение с конца на Лимон. Код замены предпоследнего значения должен работать для массивов любой длины
4. Добавьте в начало значения «Слива» и «Апельсин»
5. Удалите первое значение массива и выведите его с помошью **alert**
6. Пункты 2-5 делаем с последующим выводом массива в консоль

Перебор элементов

Для перебора элементов обычно используется цикл.

```
var arr = ["Apple", "Orange", "Pear"];

for (var i = 0; i < arr.length; i++) {
  console.log( arr[i] );
}
```

Задача 3

1. Создайте массив из таких элементов: 2, 4, 23, 55, 1, 3, 24, 33, 2
2. Посчитайте сумму элементов массива
3. Найдите наименьший и наибольший элемент массива (вывидите их индексы: `min: index1, max: index2`)

Многомерный массив в JavaScript

В JavaScript многомерные массисы создаются через вставку дгугих массивов в качестве элементов первого (получается массив массивов)

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

alert( matrix[1][1] ); // 5
```

Задача 4

1. Создайте многомерный массив 3*3, заполните его случайными числами (генерация случайного числа от min до max включительно: `var rand = min + Math.floor(Math.random() * (max + 1 - min));`)
2. Вывидите в консоль элементы матрицей
3. Найдите сумму по каждому ряду и столбцу массива

Создание массива через new Array

Существует еще один синтаксис для создания массива, он редко используется, т.к. квадратные скобки [] короче.

Кроме того, у него есть одна особенность. Если у него один аргумент-число `new Array(number)`, то он создает массив без элементов, но с заданной длиной.

```
var arr = new Array(2, 3);
alert( arr[0] ); // 2, array [2, 3]
arr = new Array(2); // creates array with two elements
alert( arr[0] ); // undefined! there are two undefined elements
```

Работа со строками

Символ	Описание
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Tab
\uNNNN	Символ в кодировке Юникод с шестнадцатеричным кодом `NNNN`. Например, `\u00A9` — юникодное представление символа копирайт ©

Строки могут содержать специальные символы (в таблице слева).

Так же надо экранировать обратным слешем (\) некоторые другие символы:

```
var str = 'I\'m a JavaScript programmer';
alert( str ); // I'm a JavaScript programmer
var str = 'backslash \\';
alert( str ); // backslash \
```

Работа со строками

Обратите внимание, `str.length` — это *свойство* строки, а `str.charAt(pos)` — *метод*, т.е. функция.

Обращение к методу всегда идет со скобками, а к свойству — без скобок.

Получение длины строки

```
var str = "My\n"; // 3, третий — перевод строки

alert( str.length ); // 3
```

Доступ к символам

```
alert( str.charAt(0) ); // M
alert( str[0] ); // M
```

Разница между этими способоми заключается в том, что если **символа нет** — **charAt** выдает **пустую строку**, а **скобки** — **undefined**.

Изменения строк

Мы не можем заменить символ внутри строки. Как только строка создана — она такая навсегда.

Можно только создатьновую строку и присвоить в переменную вместо старой.

```
var str = "string";

str = str[3] + str[4] + str[5];

alert( str ); // ing
```

Задача 5

1. Создайте строку: 'Ваше ім'я, по-батькові' (в одинарных кавычках с переводом строки в конце)
2. Вывидите в консоль длинну строки
3. Вывидите в консоль восьмой символ
4. Создайте новую строку из символов 2, 3, 13, 17 и выведите ее
5. Выведите с консоль длинну новой строки

Некоторые методы работы со строками

Изменение регистра букв строки

```
alert( "Internet".toUpperCase() ); // INTERNET
alert( "Internet".toLowerCase() ); // internet
```

Поиск подстроки

```
var str = "Widget with id";

alert( str.indexOf("Widget") ); // 0, "Widget"  at the beginning of string
alert( str.indexOf("id") ); // 1, "id" found on position 1, look closer
alert( str.indexOf("widget") ); // -1, not found (different case)
alert(str.indexOf("id", 2)) // 12, searching from position 2
```

Взятие подстроки: substr, substring, slice.

```
var str = "Widget with id";

alert(str.substring(0,1)); // "W", символы с позиции 0 по 1 не включая 1.
alert(str.substring(5)); // t with id, символы с позиции 2 до конца
alert(str.substr(2,4)); // dget, со 2-й позиции 4 символа

alert(str.slice(0,1)); // "W", символы с позиции 0 по 1 не включая 1, как substring, но:
alert( "testme".slice(1, -1) ); // "estm", от 1 позиции до первой с конца.
```

Задаче 6

1. Создайте массив строк: «ватафон», «граммафон», «турболон», «патифон», «графогон»

2. Выведите только те строки, которые содержат «фон»

3. Преобразуйте первый символ каждой выводимой строки в верхний регистр (сделать букву большой)

Важно помнить:

Все строки имеют внутреннюю кодировку Юникод.

Неважно, на каком языке написана страница, находится ли она в `windows-1251` или `utf-8`. Внутри JavaScript-интерпретатора все строки приводятся к единому «юникодному» виду. Каждому символу соответствует свой код.

Методы: масив <-> строка

split — разбивает строку по разделителю, превращая ее в массив.

```
var names = 'John, Jane, Rob, Kate';

var arr = names.split(', ');

for (var i = 0; i < arr.length; i++) {
  alert( 'You've got a message, ' +
arr[i] );
}
```

join — делает в точности противоположное **split**. Он берет массив и склеивает его в строку, используя разделитель.

```
var arr = ['John', 'Jane', 'Rob',
'Kate'];

var str = arr.join('; ');

alert( str ); // John; Jane; Rob; Kate
```

Разбивка по буквам: `alert( "test".split('') ); // t,e,s,t`
Маленький трюк: `alert( "testing".split('').reverse('').join('') ); // gnitset`

Метод slice

Метод **slice(begin, end)** копирует участок массива от begin до end, не включая end. Исходный массив при этом не меняется.

```
var arr = ["Почему", "надо", "учить",
"JavaScript"];

var arr2 = arr.slice(1, 3); //
элементы 1, 2 (не включая 3)

alert( arr2 ); // надо, учить
```

Аргументы ведут себя так же, как и в строковом **slice**:

- Если не указать **end** — копирование будет до конца массива.
- Можно использовать отрицательные индексы, они отсчитываются с конца.
- Если вообще не указать аргументов — скопируется весь массив.

Метод splice и sort

arr.splice(index[, deleteCount, elem1, ..., elemN])

Удалить **deleteCount** элементов, начиная с номера **index**, а затем вставить **elem1**, ..., **elemN** на их место. Возвращает массив из удалённых элементов.

```
var arr = ["I", "am", "studying", "JavaScript"];

// удалить 3 первых элемента и добавить другие вместо них
arr.splice(0, 2, "We", "are")

alert( arr ) // ["We", "are", "studying", "JavaScript"]
```

Метод **sort()** сортирует массив на месте.

```
var arr = [ 1, 2, 15 ];

arr.sort();

alert( arr );  // 1, 15, 2
```

По умолчанию **sort** сортирует, преобразуя элементы к строке.

Поэтому и порядок у них строковый, ведь "2" > "15".

Для указания своего порядка сортировки в метод arr.sort(fn) нужно передать функцию fn от двух элементов, которая умеет сравнивать их.

https://learn.javascript.ru/array-methods#метод-splice

Поиск елемента в массиве

Метод «**arr.indexOf(searchElement[, fromIndex])**» возвращает номер элемента searchElement в массиве arr или -1, если его нет.

Для поиска используется строгое сравнение ===.

```
var arr = [1, 0, false];

alert( arr.indexOf(0) ); // 1
alert( arr.indexOf(false) ); // 2
alert( arr.indexOf(null) ); // -1
```

Как вы могли заметить, по синтаксису он полностью аналогичен методу **indexOf** для строк.

Метод «**arr.lastIndexOf(searchElement[, fromIndex])**» ищет справа-налево: с конца массива или с номера **fromIndex**, если он указан.

Задача 6

1. Спросить у пользователя список городов, через запятую
2. Создать из городов массив
3. Отсортировать массив по алфавиту в обратную сторону (Я -> А)
4. Заменить в массиве город «Москва» на «Вена»
5. Вывести сообщение: «Тур Город1 — ГородN» (первый и последний в массиве)

Метод concat

Метод arr.concat(value1, value2, … valueN) создаёт новый массив, в который копируются элементы из arr, а также value1, value2, ... valueN.

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

alert( newArr ); // 1,2,3,4
```

Если аргумент concat – массив, то concat добавляет элементы из него.

```
var newArr = arr.concat([3, 4], 5);
alert( newArr ); // 1,2,3,4,5
```

Задача 7

1. Создать 2 массива: 1,6,3,7 и 2,5,4,8
2. Объеденить их
3. Отсортировать и вывести

Ссылки

https://learn.javascript.ru/string

https://learn.javascript.ru/array

https://learn.javascript.ru/array-methods

Inspiration: http://codepen.io/

Тест на CSS селекторы: http://flukeout.github.io/

Прохождение игры с помощью JS: https://www.codingame.com/games/puzzles/