**Методы массивов и строк в JavaScript**

**Массивы. Вступление**

Все массивы в JavaScript динамические (их размер можно изменять по ходу программы) и уже имеют набор определенных методов и свойств. Наиболее часто используются:

* Свойство Array.length – возвращает длину массива.
* Метод Array.push( element ) – добавляет element в массив Array с конца. Element будет последним в массиве.
* Метод Array.pop( ) – удаляет последний элемент из массива Array.
* Метод Array.shift( ) – удаляет первый элемент из массива Array.
* Метод Array.unshift( element ) – добавляет element в массив Array с начала. Element будет первым в массиве.

Разберем на примере:

var numbers = [1, 2, 3]  
//Добавим 4 элемента в массив. По 2 с каждой стороны  
numbers.push(4);  
numbers.push(5);  
numbers.unshift(0);  
numbers.unshift(-1);  
//Удалим по одному с каждой стороны  
numbers.pop();  
numbers.shift();  
console.log(numbers); // [ 0, 1, 2, 3, 4 ]

В предыдущем уроке вы уже познакомились с методом перебора массива forEach. Есть еще несколько способов перебора массива.

Перебор с помощью цикла for и индекса (единственный способ внести изменения в элементы):

var numbers = [1, 2, 3, 4];  
for (var i = 0; i < numbers.length; i++) {  
 numbers[i] \*= 2;  
}  
console.log(numbers); // [ 2, 4, 6, 8 ]

Перебор с помощью цикла for of поэлементно (используется для работы с элементами, не вносит изменения в первоначальный массив):

var numbers = [1, 2, 3, 4];  
for (var element of numbers) {  
 console.log(element += 2);  
}  
/\*  
3  
4  
5  
6  
\*/

Разберем не так часто используемые, но все равно полезные методы.

**Методы для изменения содержимого**

1) Splice ()

Синтаксис:

array.splice(index,[ deleteCount, elem1, ..., elemN]);

Метод работы:

Он начинает с позиции index, удаляет deleteCount элементов и вставляет elem1, ..., elemN на их место. Возвращает массив из удалённых элементов.

Пример:

var arr = ["Поешь", "еще", "этих", "французских", "булок"];  
// удалить 3 первых элемента и заменить их другими  
arr.splice(3, 1, "немецких", "и", "голландских");  
console.log(arr) // теперь [ 'Поешь', 'еще', 'этих', 'немецких', 'и', 'галандских', 'булок' ]

2) Slice ()

Синтаксис:

array.slice([start], [end]);

Метод работы:

Он возвращает новый массив, в который копирует элементы, начиная с индекса start и до end (не включая end).

Пример:

var arr = ["Поешь", "еще", "этих", "французских", "булок"];  
// скопировать элементы 1,2,3  
var newArr = arr.slice(1, 4)  
console.log(newArr) // [ 'еще', 'этих', 'французских' ]

3) Concat ()

Синтаксис:

array.concat(arg1, arg2...);

Метод работы:

Он принимает любое количество аргументов, которые могут быть как массивами, так и примитивными значениями. В результате мы получаем новый массив, включающий в себя элементы из array, а также arg1, arg2 и так далее…

Пример:

var arrFirst = ["Поешь", "еще", "этих", "французских", "булок"];  
var arrSecond = ["да", "выпей"];  
//Объединим 2 массива и добавим 2 аргумента  
var newArray = arrFirst.concat(arrSecond, "же", "чаю");  
console.log(newArray) // теперь ['Поешь', 'еще', 'этих', 'французских', 'булок', 'да', 'выпей', 'же', 'чаю']

**Методы для поиска по массиву**

1) indexOf()/lastIndexOf() и includes()

Синтаксис:

array.indexOf(item, from);  
array.lastIndexOf(item, from);  
array.includes(item, from);

Метод работы:

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

Пример:

var array =[ "Поешь", "еще", "этих", "французских", "булок", "да", "выпей", "же", "чаю"];  
console.log(array.indexOf("этих", 0)); // 2  
console.log(array.lastIndexOf("этих", 8)); // 2  
console.log(array.includes("этих", 0)); // true

2) find() и findIndex()

Синтаксис:

array.find(function(item, index, array) {

// если true - возвращается текущий элемент и перебор прерывается

// если все итерации оказались ложными, возвращается undefined

});

array.findIndex(function(item, index, array) {

// если true - возвращается текущий индекс и перебор прерывается

// если все итерации оказались ложными, возвращается undefined

});

Метод работы:

Этот метод, также как и forEach, использует как аргумент callBack. СallBack в методах – это функция, которая принимает аргументы (уже прописанные в языке) и возвращает true/false или undefined в зависимости от того, какое условие прописано в этой функции. В callBack метода find могут присутствовать 3 аргумента, называть их можно как угодно, главное, чтобы программист сам понимал обозначения. Первый аргумент всегда будет сам элемент массива, если нужно (это не обязательно) можно добавить еще индекс и сам массив.

Пример:

var array = [  
 {id: 1, word: "Поешь"},  
 {id: 2, word: "еще"},  
 {id: 3, word: "этих"},  
 {id: 4, word: "французских"},  
 {id: 5, word: "булок"}  
];  
//CallBack, который вернет нам элемент, если word будет равен "французских"  
function callBack(element) {  
 return element.word === "французских";  
}  
var element = array.find(callBack);  
console.log(element.id); //4

**Преобразование массива**

1) Sort()

Метод sort по умолчанию написан для string и работает по принципу сравнения строк (лексикографический порядок). Но у нас есть возможность самостоятельно прописать callBack для сравнения любых типов.

Синтаксис:

array.sort(function (a, b) {

//условие при котором 1, то есть а больше b

//условие при котором -1, то есть b больше a

//условие при котором 0, то есть a и b равны

});

Метод работы:

В callBack метода sort должны входить 2 аргумента, называть их можно как угодно. Первый аргумент всегда будет первый элемент из пары для сравнения, второй аргумент соответственно второй элемент.

Пример с сортировкой чисел:

function compareNumeric(a, b) {  
 if (a > b) return 1;  
 if (a === b) return 0;  
 if (a < b) return -1;  
}

//быстрая сортировка меньше 0 => false, больше 0 => true  
function compareNumericFast(a, b) {  
 return a - b;  
}  
var arr = [34, 2, 1, 15];  
var arrForFast = [34, 2, 1, 15];  
arr.sort(compareNumeric);  
arrForFast.sort(compareNumericFast);  
console.log(arr); // [ 1, 2, 15, 34 ]  
console.log(arr); // точно так же [ 1, 2, 15, 34 ]

Пример с прописанной для объектов сортировкой:

var array = [  
 {id: 5, word: "булок"},  
 {id: 2, word: "еще"},  
 {id: 3, word: "этих"},  
 {id: 4, word: "французских"},  
 {id: 1, word: "Поешь"}  
];  
function compareNumeric(a, b) {  
 if (a.id > b.id) return 1;  
 if (a.id === b.id) return 0;  
 if (a.id < b.id) return -1;  
}  
array.sort(compareNumeric);  
console.log(array);   
/\*

[  
 { id: 1, word: 'Поешь' },  
 { id: 2, word: 'еще' },  
 { id: 3, word: 'этих' },  
 { id: 4, word: 'французских' },  
 { id: 5, word: 'булок' }  
]  
\*/

2) reverse ()

Синтаксис:

array.reverse();

Метод работы:

Меняет порядок элементов в массиве на обратный.

Пример:

var array = [1, 2, 3, 4, 5];  
console.log(array.reverse()); // [ 5, 4, 3, 2, 1 ]

3) join ()

Синтаксис:

array.join(str);

Метод работы:

Разбивает строку на элементы массива, используя str как разделитель.

Пример:

var array = ['Поешь', 'еще', 'этих', 'французских', 'булок'];  
var str = array.join(' ');  
console.log(str); //Поешь еще этих французских булок

4) reduce() и reduceRight()

Методы reduce и reduceRight похожи на метод forEach , но они немного сложнее. Они используются для вычисления какого-нибудь единого значения на основе всего массива.

Синтаксис:

arr.reduce(function (previousValue, item, index, array) {

// ...

}, [initial]);

Метод работы:

В callBack метода sort должны входить как минимум 2 аргумента (если нужно, можно добавить еще 2), называть их можно как угодно.

Аргументы:

* previousValue – результат предыдущего вызова этой функции, равен initial при первом вызове (если передан initial). (Обязательный)
* item – очередной элемент массива. (Обязательный)
* index – его индекс. (Не обязательный)
* array – сам массив. (Не обязательный)

Например, попробуем написать аналог join:

var array = ['Поешь', 'еще', 'этих', 'французских', 'булок'];  
function callBackJoin(previousValue, item, index, array) {  
 var word = index !== array.length - 1 ? item + " " : item;  
 return previousValue + word;  
}  
var value = array.reduce(callBackJoin, "");  
console.log(value); //Поешь еще этих французских булок

**Строки. Вступление**

Строка — это любая переменная, которая содержит в себе текст. Она тоже имеет свои методы и свойства. Тип String также можно перебрать методами for(по индексу) и for of аналогично массивам.

Мы можем получать отдельные символы строки с помощью квадратных скобок и номера элемента (начиная с 0). Но только для чтения, изменять посимвольно строки мы не можем. Поэтому, при необходимости поменять один символ, придется создавать новую строку и склеивать ее по частям.

Поэтому:

var str = 'js';  
str[0] = 'J'; // ошибка  
console.log(str); // не работает, все равно js  
// но  
var newStr = "J" + str[1];  
console.log(newStr); // Js

**Методы изменения строки**

1) toLowerCase() и toUpperCase()

Синтаксис:

str.toLowerCase();

str.toUpperCase ();

Метод работы:

Меняют регистр всех символов на заглавный (toUpperCase) или прописной (toLowerCase).

Пример:

var str = 'JavaScript';  
console.log(str.toUpperCase()); // JAVASCRIPT  
console.log(str.toLowerCase()); // javascript  
console.log(str[0].toLowerCase()); // j

2) replace()

Синтаксис:

str.replace(substr, newSubstr);

Метод работы:

Меняет подстроку substr на newSubstr в строке str.

Пример:

var str = 'JavaScript';  
console.log(str.replace(‘S’,’C’)) // JavaCript

**Поиск подстроки**

1) indexOf()

Синтаксис:

str.indexOf(substr, index);

Метод работы:

Ищет подстроку substr в строке str, начиная с индекса index, и возвращает индекс, на котором располагается совпадение, либо -1 при отсутствии совпадений.

Пример:

var str = 'JavaScript';  
console.log(str.indexOf('Scr', 0)); // 4

2) includes()

Синтаксис:

str.includes(substr, index);

Метод работы:

Возвращает true, если в строке str есть подстрока substr, либо false, если нет.

Пример:

var str = 'JavaScript';  
console.log(str.includes('Scr', 0)); // true

3) startsWith() и endsWith()

Синтаксис:

str.startsWith(substr);

str.endsWith(substr);

Метод работы:

Проверяют начинается ли (startsWith) и заканчивается ли (endsWith) строка подстрокой substr.

Пример:

var str = 'JavaScript';  
console.log(str.startsWith("Java")); // true, "Java" — начало "JavaScript"  
console.log(str.endsWith("ipt")); // true, "ipt" — окончание "JavaScript"

**Получение подстроки**

1) slice()

Синтаксис:

str.slice(startIndex, endIndex);

Метод работы:

Возвращает подстроку, состоящую из символов под номерами начиная с startIndex и заканчивая endIndex – 1(то есть str[endIndex] не входит).

Пример:

var str = 'JavaScript';  
console.log(str.slice(4, 7)); // “Scr”, с позиции 4 и до 7 не включительно

**Сравнение строк**

1) localeCompare ()

Синтаксис:

str. localeCompare(str2);

Метод работы:

Возвращает число, которое показывает, какая строка больше (выше по алфавиту) в соответствии с правилами языка:

* -1, если str меньше str2.
* 1, если str больше str2.
* 0, если строки равны.

Пример:

var str = 'JavaScript';  
console.log(str.localeCompare("Angular")); // 1 т.к Angular по алфавиту выше JavaScript

**Разбиение строки на массив**

1) split ()

Синтаксис:

str. split(comp);

Метод работы:

Возвращает массив, состоящий из элементов строки. Как разделитель (подстрока-показатель конца элемента и начала нового) будет использоваться comp.

Пример – строка разбита на слова (разделитель - пробел):

var str = "Сегодня я никуда не пойду";  
console.log(str.split(" ")); //['Сегодня', 'я', 'никуда', 'не', 'пойду']

**Специальные символы**

Специальные символы – это символы для форматирования текста. В строке они будут выглядеть как обычные символы, но компьютер прочитает их по-своему и заменит на привычный нам вид.

Например:

* \n - символ перевода строки.
* \’ или \” – символ кавычек в тексте.
* \t – символ табуляции.
* \xXX – символ с шестнадцатеричным юникодным кодом XX, например, '\x7A' — то же самое, что 'z'.
* \uXXXX– символ в кодировке UTF-16 с шестнадцатеричным кодом XXXX, например, \u00A9 — юникодное представление знака копирайта, ©.
* \u{X…XXXXXX} (от 1 до 6 шестнадцатеричных цифр) - символ в кодировке UTF-32 с шестнадцатеричным кодом от U+0000 до U+10FFFF. Некоторые редкие символы кодируются двумя 16-битными словами и занимают 4 байта. Так можно вставлять символы с длинным кодом.

Пример использования в коде:

console.log("\u{1F60D}");  
// 😍, лицо с улыбкой и глазами в форме сердец  
console.log("Он сказал :\n\'Сегодня я никуда не пойду\'")  
/\*  
Он сказал :   
'Сегодня я никуда не пойду'  
\*/