Работа со строковыми функциями в JavaScript

В данном уроке мы будем изучать стандартные функции JavaScript.

Изучите методы и функции по следующим ссылкам, постарайтесь запомнить и законспектировать полученную информацию: length, toUpperCase, toLowerCase, substr, substring, slice, indexOf, replace,split, join.

Обязательно изучите каждую статью до конца - примеры в конце статьи специально подобраны так, чтобы подготовить вас к решению задач.

http://code.mu/books/javascript/base/rabota-so-strokovymi-funkciyami-v-javascript.html

***Свойство length***

Свойство length позволяет узнать длину строки. Под длиной понимается количество символов в ней.

Синтаксис

строка.length

Пусть в переменной str хранится строка. Давайте найдем ее длину:

var str = 'abcdefg';  
document.write(str.length);

Результат выполнения кода: 7

***Метод toUpperCase***

Метод toUpperCase производит преобразование строки в верхний регистр (из маленьких букв делает большие). При этом возвращается новая строка, а исходная строка не меняется.

См. также метод toLowerCase, которая преобразует строку в нижний регистр.

Синтаксис

***строка.toUpperCase();***

В данном примере строка из нижнего регистра преобразуется в верхний:

var str = "javascript+html";  
document.write(str.toUpperCase());

Результат выполнения кода: JAVASCRIPT+HTML

Исходная строка не меняется:

var str = "JavaScript";  
var newStr = str.toUpperCase();  
document.write('Новая строка: ' + newStr + '<br>');  
document.write('Исходная строка: ' + str);

Результат выполнения кода: Новая строка: JAVASCRIPT Исходная строка: JavaScript

***Метод toLowerCase***

Метод toLowerCase производит преобразование строки в нижний регистр (из больших букв делает маленькие). При этом возвращается новая строка, а исходная строка не меняется.

См. также метод toUpperCase, которая преобразует строку в верхний регистр.

Синтаксис

***строка.toLowerCase();***

В данном примере в строке все большие буквы преобразовались в маленькие:

var str = "Язык JAVASCRIPT";  
document.write(str.toLowerCase());

Результат выполнения кода: язык javascript

Исходная строка не меняется:

var str = "Язык JAVASCRIPT";  
var newStr = str.toLowerCase();  
document.write('Новая строка: ' + newStr + '<br>');  
document.write('Исходная строка: ' + str);

Результат выполнения кода: Новая строка: язык javascript Исходная строка: Язык JAVASCRIPT

***Метод substr***

Метод substr возвращает подстроку из строки (исходная строка при этом не изменяется).

Первый параметр задает номер символа, с которого метод начинает отрезать (учтите, что нумерация с нуля), а второй параметр - сколько символов отрезать.

Первый параметр может принимать отрицательные значения. В этом случае отсчет символа, с которого начинается обрезание, будет идти с конца строки. Причем последний символ имеет номер "-1", предпоследний - "-2" и так далее.

Второй параметр не является обязательным, если он не указан, то вырезаны будут все символы до конца строки.

См. также методы substring и slice, которые также возвращают подстроку.

Синтаксис

строка.substr(откуда отрезать, [сколько символов отрезать]);

В данном примере подстрока длиной 3 символа вырежется с нулевой позиции:

var str = 'Мне очень нравится JavaScript';  
document.write(str.substr(0, 3));

Результат выполнения кода: Мне

В данном примере подстрока вырезается с 4-той позиции и до конца строки (так как последний параметр не задан):

var str = 'Мне очень нравится JavaScript';  
document.write(str.substr(4));

Результат выполнения кода: очень нравится JavaScript

В данном примере подстрока вырезается с 10-го символа с конца и берется 4 символа:

var str = 'Мне очень нравится JavaScript';  
document.write(str.substr(-10, 4));

Результат выполнения кода: Java

В данном примере подстрока вырезается с 10-го символа с конца и берется вся оставшаяся строка до конца (так как последний параметр не задан):

var str = 'Мне очень нравится JavaScript';  
document.write(str.substr(-10));

Результат выполнения кода: JavaScript

В данном примере возвращается последний символ строки:

var str = 'Мне очень нравится JavaScript';  
document.write(str.substr(-1));

Результат выполнения кода: t

***Метод substring***

Метод substring возвращает подстроку из строки (исходная строка при этом не изменяется).

Первый параметр задает номер символа, с которого метод начинает отрезать (учтите, что нумерация с нуля), а второй параметр - номер символа, на котором следует закончить вырезание (символ с этим номером не включается в вырезанную часть).

Второй параметр не является обязательным, если он не указан, то вырезаны будут все символы до конца строки.

Если первый параметр больше второго, тогда substringing ведет себя, как будто они поменялись местами. Например, str.substringing(3, 0) == str.substringing(0, 3).

Если какой-то из параметров больше, чем длина строки - он считается равным длине строки.

Отрицательные значения приравниваются к нулю.

См. также методы substr и slice, которые также возвращают часть подстроки.

Синтаксис

строка.substr(откуда начать отрезать, [до куда отрезать]);

В данном примере вырезаются слова 'Мне очень'. Первая буква этой подстроки имеет номер 0, а последняя - номер 8. Значит, указываем, что мы будем резать с нулевого символа по 9-тый (на 1 больше, чем нам нужно):

var str = 'Мне очень нравится JavaScript';  
document.write(str.substring(0, 9));

Результат выполнения кода: Мне очень

Давайте вырежем слово 'очень':

var str = 'Мне очень нравится JavaScript';  
document.write(str.substring(4, 9));

Результат выполнения кода: очень

В данном примере подстрока вырезается с 4-той позиции и до конца строки (так как последний параметр не задан):

var str = 'Мне очень нравится JavaScript';  
document.write(str.substr(4));

Результат выполнения кода: очень нравится JavaScript

***Метод slice***

Метод slice возвращает возвращает подстроку из строки (исходная строка при этом не изменяется).

Первым параметром указывается номер символа строки, с которого начинается вырезание, а вторым параметром - номер символа, на котором закончится вырезание (при этом символ с этим номером не включится в вырезанную часть).

Второй параметр не является обязательным. Если его не указать - подстрока возьмется с указанного в первом параметре символа до конца строки.

Второй параметр также может принимать отрицательные значения. В этом случае отсчет символа, на котором закончится обрезание, начинается с конца строки. Причем последний символ имеет номер "-1", предпоследний - "-2" и так далее.

К примеру, если написать slice(1, -1), то вырезанная часть возьмется с первого символа включительно по последний не включая его. Если вам нужно включить и последний символ - нужно просто не задавать второй параметр для slice, вот так: slice(1).

Если второй параметр меньше первого - возвратится пустая строка.

См. также методы substr и substring, которые также возвращают часть подстроки.

Синтаксис

строка.slice(откуда отрезать, [докуда отрезать]);

В данном примере вырезаются слова 'Мне очень'. Первая буква этой подстроки имеет номер 0, а последняя - номер 8. Значит, указываем, что мы будем резать с нулевого символа по 9-тый (на 1 больше, чем нам нужно):

var str = 'Мне очень нравится JavaScript';  
document.write(str.substr(0, 9));

Результат выполнения кода: Мне очень

В данном примере подстрока вырезается с 4-той позиции и до конца строки (так как последний параметр не задан):

var str = 'Мне очень нравится JavaScript';  
document.write(str.substr(4));

Результат выполнения кода: очень нравится JavaScript

Пример

В данном примере из исходной строки извлекаются символы от позиции 0 до позиции -1 не включительно (-1 указывает на последний символ и он не включится в извлеченную часть):

var str = '12345';  
document.write(str.slice(0, -1));

Результат выполнения кода: 1234

Преимущество такого подхода в том, что вырезаться всегда будет часть строки, не включая последний символ, независимо от размера строки.

***Метод indexOf***

Метод indexOf осуществляет поиск подстроки (указывается первым параметром) в строке.

Метод вернет позицию первого совпадения, а если оно не найдено - то -1.

Вторым параметром (он необязателен) можно передать номер символа, откуда следует начинать поиск.

Метод чувствителен к регистру символов.

См. также метод lastIndexOf, которая также осуществляет поиск подстроки, но начинает поиск не с начала строки, а с конца.

Синтаксис

строка.indexOf(что ищем, [откуда начинать поиск]);

В данном примере в строке 'Я учу учу Javascript' ищется слово 'учу'. Метод вернет 2 (с этой позиции начинается первое слово 'учу' в строке):

var str = 'Я учу учу Javascript';  
document.write(str.indexOf('учу'));

Результат выполнения кода: 2

В данном примере в строке 'Я учу учу Javascript' также ищется слово 'учу', однако поиск начинается с 5-той позиции (я указал это вторым параметром). Метод вернет 6 (с этой позиции начинается второе слово 'учу' в строке):

var str = 'Я учу учу Javascript';

document.write(str.indexOf('учу', 5));

Результат выполнения кода: 6

В данном примере метод вернет -1, так как подстроки 'PHP' в строке 'Я учу Javascript' нету:

var str = 'Я учу Javascript';  
document.write(str.indexOf('PHP', 5));

Результат выполнения кода: -1

В данном примере метод вернет -1, так как он чувствителен к регистру (Javascript не равно JaVaScrIpt):

var str = 'Я учу Javascript';  
document.write(str.indexOf('JaVaScrIpt'));

Результат выполнения кода: -1

В данном примере метод вернет -1, так как после выбранной позиции совпадения отсутствуют:

var str = 'Я учу Javascript';  
document.write(str.indexOf('учу', 8));

Результат выполнения кода: -1

***Метод replace***

Метод replace осуществляет поиск и замену частей строки.

Первым параметром принимается подстрока, которую заменяем, а вторым - подстрока, на которую заменяем.

Синтаксис

строка\_в\_которой\_заменяем.replace(что заменяем, на что заменяем);

Первым параметром вместо строки может приниматься регулярное выражение.

Заменим в строке 'Я учу PHP' слово 'PHP' на слово 'JavaScript':

var str = 'Я учу PHP';  
var newStr = str.replace('PHP', 'JavaScript');  
document.write(newStr);

Результат выполнения кода: Я учу JavaScript

Метод заменяет только первое совпадение. В следующем примере в строке 'Я учу PHP PHP' заменится только первое слово 'PHP':

var str = 'Я учу PHP PHP';  
var newStr = str.replace('PHP', 'JavaScript');  
document.write(newStr);  
Результат выполнения кода: Я учу JavaScript PHP

Чтобы функция заменила все совпадения, следует воспользоваться так называемым глобальным поиском:

var str = 'Я учу PHP PHP';  
var newStr = str.replace(/PHP/g, 'JavaScript');  
document.write(newStr);

Результат выполнения кода: Я учу JavaScript JavaScript

Внимание! Чтобы правильно пользоваться приемом с глобальным поиском, вам нужно разобраться с регулярными выражениями.

***Метод split***

Метод split осуществляет разбиение строки в массив по указанному разделителю.

Разделитель указывается первым необязательным параметром. Если он не задан - вернется вся строка. Если он задан как пустые кавычки '' - то каждый символ строки попадет в отдельный элемент массива.

Вторым необязательным параметром можно указать максимальное количество элементов в получившемся массиве (см. примеры).

См. также метод join, который сливает элементы массива в строку.

Синтаксис

строка.split([разделитель], [максимальное количество элементов]);

В данном примере строка разбита по разделителю '-'. В результате получится следующий массив arr:

var str = 'Мне-очень-нравится-JavaScript';  
var arr = str.split('-');  
console.log(arr);

Результат выполнения кода: ['Мне', 'очень', 'нравится', 'JavaScript']

В данном примере строка также разбита по разделителю '-', однако вторым параметром указано максимальное количество элементов в получившемся массиве (2 элемента). Поэтому в новый массив запишется только 2 элемента:

var str = 'Мне-очень-нравится-JavaScript';  
var arr = str.split('-', 2);  
console.log(arr);

Результат выполнения кода: ['Мне', 'очень']

Давайте каждый символ строки запишем в отдельный элемент массива:

var str = 'abcde';  
var arr = str.split('');  
console.log(arr);

Результат выполнения кода: ['a', 'b', 'c', 'd', 'e']

Давайте первые 3 символа строки запишем в отдельный элемент массива:

var str = 'abcde';  
var arr = str.split('', 3);  
console.log(arr);

Результат выполнения кода: ['a', 'b', 'c']

Сейчас с помощью split разбивается строка с числами. Обратите внимание на то, что в результате получится массив строк ['1', '2', '3', '4', '5'], а не массив чисел [1, 2, 3, 4, 5]:

var str = '12345';  
var arr = str.split('');  
console.log(arr);

Результат выполнения кода: ['1', '2', '3', '4', '5']

Давайте перевернем символы строки в обратном порядке. Для этого разобьем строку в массив с помощью split по разделителю '' (этот разделитель положит каждый символ строки в отдельный элемент массива), перевернем этот массив с помощью reverse и затем сольем перевернутый массив обратно с помощью join:

var str = '123456789';  
var arr = str.split('');  
var arr = arr.reverse();  
var result = arr.join('');  
document.write(result);

Результат выполнения кода: 987654321

Упростим решение предыдущей задачи - сольем все команды в цепочку:

var str = '123456789';  
var result = str.split('').reverse().join('');  
document.write(result);

Результат выполнения кода: 987654321

Дана строка с цифрами. Найдем сумму цифр из этой строки. Для этого разобьем строку в массив, а затем переберем этот массив и найдем его сумму. Нас ждет подвох: split возвращает строки, поэтому при суммировании преобразуем эти цифры-строки в настоящие числа с помощью Number:

var str = '12345';  
var arr = str.split('');  
var sum = 0;for (var i = 0; i <= arr.length; i++) {  
 sum += Number(arr[i]); }  
document.write(sum);

Результат выполнения кода: 15

Дано число. Давайте запишем каждую цифру этого числа в отдельный элемент массива. Тут есть подвох - split применяется только к строкам, а у нас число. Преобразуем вначале число к строке с помощью String, а затем применим split:

var num = 12345;  
var str = String(num);  
var arr = str.split('');  
console.log(arr);

Результат выполнения кода:

['1', '2', '3', '4', '5']

***Функция join***

Функция join сливает строку в массив с указанным разделителем.

См. также функцию implode, которая является аналогом функции join;

Синтаксис

join(разделитель, строка);