
IT-ACADEMY

Строки JavaScript

Строки

Строки JavaScript используются для хранения текста и управления им.

Строка JavaScript - это ноль или более символов, записанных в кавычках.

```
var x = "John Doe";
```

```
var carName1 = "Volvo XC60";  // Double quotes
var carName2 = 'Volvo XC60';  // Single quotes
```

Вы можете использовать кавычки внутри строки, если они не соответствуют кавычкам, окружающим строку:

```
var answer1 = "It's alright";
var answer2 = "He is called 'Johnny'";
var answer3 = 'He is called "Johnny"';
```

Длина строки

Чтобы узнать длину строки, используйте встроенное свойство длины:

```
<p id="demo"></p>
<script>
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
var sln = txt.length;
document.getElementById("demo").innerHTML = sln;
</script>
```

Пропуск символов

Поскольку строки должны быть записаны в кавычках, JavaScript неправильно поймет эту строку:

```
var x = "We are the so-called "Vikings" from the north.";
```

Строка будет перерезана до "We are the so-called ".
Решение избежать этой проблемы - использовать escape-символ обратной косой черты.
Управляющий символ обратной косой черты (\) превращает специальные символы в строковые символы:

Пропуск символов

Code	Result	Description
\'	'	Single quote
\"	"	Double quote
\\	\	Backslash

```
<p id="demo"></p>

<script>
var x = "We are the so-called \"Vikings\" from the
north.";
document.getElementById("demo").innerHTML = x;
</script>
```

Пример 2

```
<p id="demo"></p>

<script>
var x = 'It\'s alright.';
document.getElementById("demo").innerHTML = x;
</script>


var x = "The character \\ is called backslash.";
```

В JavaScript действительны шесть других escape-последовательностей:

Приведенные выше 6 escape-символов изначально были разработаны для управления пишущими машинками, телетайпами и факсами. Они не имеют смысла в HTML.

Код	Резульат
\b	Backspace
\f	Form Feed
\n	New Line
\r	Carriage Return
\t	Horizontal Tabulator
\v	Vertical Tabulator

Прерывание длинных строк кода

Для лучшей читаемости программисты часто избегают строк кода длиннее 80 символов.

Если оператор JavaScript не умещается в одной строке, лучше всего его прервать после оператора:

```
document.getElementById("demo").innerHTML = \
"Hello Dolly!";
```

Строки могут быть объектами

Обычно строки JavaScript предоставляют собой примитивные значения, созданные из литералов:

var firstName = "John";
Наши строки могут быть устойчивыми как объекты с также ключевым словом new:

var firstName = new String("John");

```
var y = new String("John");

// typeof x will return string
// typeof y will return object
```

Не создавайте строки как объекты. Это снижает скорость выполнения. Ключевое слово усложняет код. Это может привести к неожиданным результатам:

Пример 3

```
var x = "John";
var y = new String("John");

// (x == y) is true because x and y have equal values

var x = "John";
var y = new String("John");

// (x === y) is false because x and y have different types (string and object)

var x = new String("John");
var y = new String("John");

// (x == y) is false because x and y are different objects

var x = new String("John");
var y = new String("John");

// (x === y) is false because x and y are different objects
```

Обратите внимание на разницу между (x == y) и (x === y).
Сравнение двух объектов JavaScript всегда возвращает false.

Строковые методы JavaScript

Строковые методы помогают работать со строками.

Строковые методы и свойства

Примитивные значения, такие как «Джон Доу», не имеют свойств или методов (потому что они не являются объектами).
С помощью JavaScript методы и свойства также доступны для примитивных значений, потому что JavaScript обрабатывает примитивные значения как объекты при выполнении методов и свойств.

Длина строки
lengthСвойство возвращает длину строки:

```
<p id="demo"></p>

<script>
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
var sln = txt.length;
document.getElementById("demo").innerHTML = sln;
</script>
```

Поиск строки в строке

indexOf() метод возвращает позицию первого вхождения указанного текста

```
var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("locate");
```

JavaScript считает позиции с нуля.
0 - первая позиция в строке, 1 - вторая, 2 - третья ...

Метод lastIndexOf () возвращает индекс последнего вхождения указанного текста в строку:

```
<p id="demo"></p>

<script>
var str = "Please locate where 'locate' occurs!";
var pos = str.lastIndexOf("locate");
document.getElementById("demo").innerHTML = pos;
</script>
```

Методы

И indexOf (), и lastIndexOf () возвращают -1, если текст не найден.

Оба метода принимают второй параметр в качестве начальной позиции для поиска:

```
var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("locate", 15);
```

Методы lastIndexOf () выполняют поиск в обратном направлении (от конца к началу), что означает: если второй параметр равен 15, поиск начинается с позиции 15 и выполняется до начала строки.

```
h2>JavaScript String Methods</h2>

<p>Both indexOf(), and lastIndexOf() return -1 if the text is not found:</p>

<p id="demo"></p>

<script>
var str = "Please locate where 'locate' occurs!";
var pos = str.indexOf("John");
document.getElementById("demo").innerHTML = pos;
</script>
```

Поиск строки в строке

Метод search () ищет в строке указанное значение и возвращает позицию совпадения:

```
var str = "Please locate where 'locate' occurs!";
var pos = str.search("locate");
```

Ты заметил?
Два метода, indexOf () и search (), равны?

Они принимают одни и те же аргументы (параметры) и возвращают одно и то же значение?

Эти два метода НЕ равны. Вот отличия:

Метод search () не может принимать второй аргумент начальной позиции.
Метод indexOf () не может принимать мощные поисковые значения (регулярные выражения).
Вы узнаете больше о регулярных выражениях в следующей главе.

Извлечение части строк

Есть 3 метода извлечения части строки:

slice(start, end) (начало, конец)

substring(start, end) (начало, конец)

substr(start, length) (начало, длина)

slice () извлекает часть строки и возвращает извлеченную часть в новой строке.

Метод принимает 2 параметра: начальную позицию и конечную позицию (конец не включен).

В этом примере часть строки вырезается из позиции 7 до позиции 13:

```
var str = "Apple, Banana, Kiwi";
var res = str.slice(7, 13);
```

Помните: JavaScript считает позиции с нуля.
Первая позиция - 0.

строки

Если параметр отрицательный, позиция отсчитывается от конца строки.

В этом примере часть строки вырезается из позиции -12 в позицию -6:

```
var str = "Apple, Banana, Kiwi";
var res = str.slice(-12, -6);
```

Отрицательные позиции не работают в Internet Explorer 8 и ранее.

Метод substring ()
substring () похожа на slice ().

```
var str = "Apple, Banana, Kiwi";
var res = str.substring(7, 13);
```

Если вы опустите второй параметр, substring () вырежет оставшуюся часть строки.

Метод substr ()

substr () похожа на slice ().

Разница в том, что второй параметр указывает длину извлеченной части.

```
var str = "Apple, Banana, Kiwi";
var res = str.substr(7, 6);
```

```
Banana
```

```
var str = "Apple, Banana, Kiwi";
var res = str.substr(7);
```

```
Banana, Kiwi
```

Если первый параметр отрицательный, позиция отсчитывается от конца строки.

```
var str = "Apple, Banana, Kiwi";
var res = str.substr(-4);
```

```
Kiwi
```

Замена содержимого строки

Метод replace () заменяет указанное значение другим значением в строке:

```
str = "Please visit Microsoft!";
var n = str.replace("Microsoft", "W3Schools");
```

Метод replace () не изменяет строку, для которой он вызывается. Возвращает новую строку.

По умолчанию метод replace () заменяет только первое совпадение:

```
str = "Please visit Microsoft and Microsoft!";
var n = str.replace("Microsoft", "W3Schools");
```

По умолчанию метод replace () чувствителен к регистру. Записать MICROSOFT (в верхнем регистре) не получится:

Строка преобразуется в нижний регистр с помощью toLowerCase ():

```
var text1 = "Hello World!";       // String
var text2 = text1.toLowerCase();  // text2 is text1 converted to lower
```

Метод concat ()

concat () объединяет две или более строк:

```
var text1 = "Hello";
var text2 = "World";
var text3 = text1.concat(" ", text2);
```

```
<p id="demo"></p>

<script>
var text1 = "Hello";
var text2 = "World!";
var text3 = text1.concat(" ",text2);
document.getElementById("demo").innerHTML = text3;
</script>
```

Все строковые методы возвращают новую строку. Они не изменяют исходную строку. Формально сказано: строки неизменяемы: строки не могут быть изменены, их можно только заменить.

String.trim ()

Метод trim () удаляет пробелы с обеих сторон строки:

```
var str = "       Hello World!        ";
alert(str.trim());
```

Метод trim () не поддерживается в Internet Explorer 8 и более ранних версиях.

Вы также можете использовать решение для замены выше, чтобы добавить функцию обрезки в JavaScript String.prototype:

```
if (!String.prototype.trim) {
  String.prototype.trim = function () {
    return this.replace(/^[\s\uFEFF\xA0]+|[\s\uFEFF\xA0]+$/g, '');
  };
}
var str = "       Hello World!        ";
alert(str.trim());
```

javaScript String Padding

В ECMAScript 2017 добавлено два метода String: padStart и padEnd для поддержки заполнения в начале и в конце строки.

```
let str = "5";
str = str.padStart(4,0);
// result is 0005

let str = "5";
str = str.padEnd(4,0);
// result is 5000
```

Заполнение строк не поддерживается в Internet Explorer.

Firefox и Safari были первыми браузерами с поддержкой строкового заполнения JavaScript.

Извлечение строковых символов

Есть 3 метода извлечения строковых символов:

charAt (позиция)
charCodeAt (позиция)
Доступ к собственности []

```
var str = "HELLO WORLD";
str.charAt(0);            // returns H
```

Метод charCodeAt ()
Метод charCodeAt () возвращает юникод символа по указанному индексу в строке:

Метод возвращает код UTF-16 (целое число от 0 до 65535).

```
var str = "HELLO WORLD";

str.charCodeAt(0);         // returns 72
```

Доступ к свойствам

ECMAScript 5 (2009) разрешает доступ к свойствам [] для строк:

```
var str = "HELLO WORLD";
str[0];                  // returns H
```

Доступ к свойствам может быть немного непредсказуемым:

Не работает в Internet Explorer 7 или более ранней версии.
Это делает строки похожими на массивы (но это не так).
Если символ не найден, [] возвращает значение undefined, a charAt () возвращает пустую строку.
Он доступен только для чтения. str [0] = "A" не выдает ошибки (но не работает!).

```
var str = "HELLO WORLD";
str[0] = "A";            // Gives no error, but does not work
str[0];                  // returns H
```

Если вы хотите работать со строкой как с массивом, вы можете преобразовать ее в массив.

Преобразование строки в массив

Преобразование строки в массив
Строку можно преобразовать в массив с помощью метода split ():

```
var txt = "a,b,c,d,e";   // String
txt.split(",");          // Split on commas
txt.split(" ");          // Split on spaces
txt.split("|");          // Split on pipe
```

Если разделитель опущен, возвращаемый массив будет содержать всю строку в индексе [0].

Если разделитель "", возвращаемый массив будет массивом отдельных символов:

```
var txt = "Hello";       // String
txt.split("");           // Split in
characters
```

Методы String HTML Wrapper
Методы оболочки HTML возвращают строку, заключенную в соответствующий тег HTML.

Это нестандартные методы, и они могут работать не во всех браузерах.

anchor () Создает привязку
big () Отображает строку с использованием большого шрифта
blink () Отображает мигающую строку
bold () Отображает строку жирным шрифтом
fixed () Отображает строку с использованием шрифта с фиксированным шагом
fontcolor () Отображает строку с использованием указанного цвета
fontsize () Отображает строку заданного размера
курсив () Отображает строку курсивом
link () Отображает строку как гиперссылку
small () Отображает строку мелким шрифтом
strike () Отображает строку с зачеркиванием
sub () Отображает строку как нижний текст
sup () Отображает строку в виде надстрочного текста

Строковые методы

Описание метода
charAt () Возвращает символ по указанному индексу (позиции)
charCodeAt () Возвращает Unicode символа по указанному индексу
concat () Объединяет две или более строк и возвращает новые соединенные строки
endWith () Проверяет, заканчивается ли строка указанной строкой / символами
fromCharCode () Преобразует значения Unicode в символы
includes () Проверяет, содержит ли строка указанную строку / символы
indexOf () Возвращает позицию первого найденного вхождения указанного значения в строке
lastIndexOf () Возвращает позицию последнего найденного вхождения указанного значения в строке
localeCompare () Сравнивает две строки в текущей локали
match () Ищет в строке совпадение с регулярным выражением и возвращает совпадения.
repeat () Возвращает новую строку с указанным количеством копий существующей строки
replace () Ищет в строке указанное значение или регулярное выражение и возвращает новую строку, в которой указанные значения заменяются
search () Ищет в строке указанное значение или регулярное выражение и возвращает позицию совпадения.
slice () извлекает часть строки и возвращает новую строку
split () Разбивает строку на массив подстрок
startWith () Проверяет, начинается ли строка с указанных символов
substr () Извлекает символы из строки, начиная с указанной начальной позиции и до указанного количества символов.
substring () Извлекает символы из строки между двумя указанными индексами

Спасибо