Строки

В JavaScript любые текстовые данные являются строками.

Строки создаются при помощи двойных или одинарных кавычек:

```
1  var text = "моя строка";
2
3  var anotherText = 'еще строка';
4
5  var str = "012345";
```

В JavaScript нет разницы между двойными и одинарными кавычками.

Специальные символы

Строки могут содержать специальные символы. Самый часто используемый из таких символов – это «перевод строки».

Он обозначается как \n, например:

```
alert( 'Привет\nМир' ); // выведет "Мир" на новой строке
```

Экранирование специальных символов

Если строка в одинарных кавычках, то внутренние одинарные кавычки внутри должны быть экранированы, то есть снабжены обратным слешем \', вот так:

```
var str = 'I\'m a JavaScript programmer';
```

В двойных кавычках – экранируются внутренние двойные:

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

Экранирование специальных символов

Если строка в одинарных кавычках, то внутренние одинарные кавычки внутри должны быть экранированы, то есть снабжены обратным слешем \', вот так:

```
var str = 'I\'m a JavaScript programmer';
```

В двойных кавычках – экранируются внутренние двойные:

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

Методы и свойства

- ✔ Все значения в JavaScript, за исключением null и undefined, содержат набор вспомогательных функций и значений, доступных «через точку».

- ✔Такие функции называют «методами», а значения – «свойствами».

Длина

✔ Получение длины строки

```
var str = "My\n"; // 3 символа. Третий - перевод строки

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

Доступ к символам

✔ Чтобы получить символ, используйте вызов *charAt(позиция)*. Первый символ имеет позицию 0

```
var str = "jQuery";
alert( str.charAt(0) ); // "j"
```

✔Также для доступа к символу можно использовать квадратные скобки:

```
var str = "Я - современный браузер!";
alert( str[0] ); // "Я"
```

✔Разница между этим способом и *charAt* заключается в том, что если символа нет – *charAt* выдает пустую строку, а скобки – *undefined*

Доступ к символам

✔ Чтобы получить символ, используйте вызов *charAt(позиция)*. Первый символ имеет позицию 0

```
var str = "jQuery";
alert( str.charAt(0) ); // "j"
```

✔Также для доступа к символу можно использовать квадратные скобки:

```
var str = "Я - современный браузер!";
alert( str[0] ); // "Я"
```

✔Разница между этим способом и *charAt* заключается в том, что если символа нет – *charAt* выдает пустую строку, а скобки – *undefined*

Изменения строк

✔ Содержимое строки в JavaScript нельзя изменять. Нельзя взять символ посередине и заменить его. Как только строка создана – она такая навсегда.

Смена регистра

✔Методы *toLowerCase*() и *toUpperCase*() меняют регистр строки на нижний/верхний

```
72 alert( "Интерфейс".toUpperCase() );
73 alert( "Интерфейс".toLowerCase() );
```

Поиск подстроки

✔Для поиска подстроки есть метод *indexOf(подстрока[, начальная_позиция])*.

✔Он возвращает позицию, на которой находится подстрока или -1, если ничего не найдено.

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

alert( str.indexOf("Widget") );
// 0, т.к. "Widget" найден прямо в начале str

alert( str.indexOf("id") ); // 1, т.к. "id"
найден, начиная с позиции 1

alert( str.indexOf("widget") ); // -1, не
найдено, так как поиск учитывает регистр
```

Поиск подстроки

✔Необязательный второй аргумент позволяет искать, начиная с указанной позиции. Например, первый раз "id" появляется на позиции 1. Чтобы найти его следующее появление – запустим поиск с позиции 2

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

alert(str.indexOf("id", 2)) // 12, поиск начат с позиции 2
```

Взятие подстроки: substring.

✔Метод *substring(start, end)* возвращает подстроку с позиции *start* до *end*, но не включая *end*

```
var str = "stringify";
alert(str.substring(0,1)); // "s", символы с позиции 0 по 1 не включая 1.
```

✔Если аргумент end отсутствует, то идет до конца строки

```
var str = "stringify";
alert(str.substring(2)); // ringify, символы с позиции 2 до конца
```

Взятие подстроки: substr.

- ✔Метод **substr(start [, length])**
- ✔Первый аргумент имеет такой же смысл, как и в substring, а второй содержит не конечную позицию, а количество символов.
- ✔Если второго аргумента нет – подразумевается «до конца строки».

```
var str = "stringify";
str = str.substr(2,4); // ring, со 2-й позиции 4 символа
alert(str)
```

Взятие подстроки: slice.

- ✔Метод **slice(start [, end])**
- ✔Возвращает часть строки от позиции start до, но не включая, позиции end. Смысл параметров – такой же как в substring.

Взятие подстроки: slice и substring.

Различие между *substring* и *slice* – в том, как они работают с отрицательными и выходящими за границу строки аргументами:

```
alert( "testme".substring(-2) ); // "testme", -2 становится
```

если *start* > *end*, то аргументы меняются местами, т.е. возвращается участок строки между *start* и *end*

```
alert( "testme".substring(4, -1) ); // "test"
// -1 становится 0 -> получили substring(4, 0)
// 4 > 0, так что аргументы меняются местами -> substring(0, 4) = "test"
```

Взятие подстроки: slice и substring.

В методе slice отрицательные значения отсчитываются от конца строки:

```
alert( "testme".slice(-2) ); // "me", от 2 позиции с конца
```

```
alert( "testme".slice(1, -1) ); // "estm", от 1 позиции до первой с конца.
```

trim()

Метод *trim* обрезает пробелы в начале и в конце строки

Синтаксис

```
str.trim()
```

Пример

```
var orig = '   foo  ';
console.log(orig.trim()); // 'foo'

var orig = 'foo    ';
console.log(orig.trim()); // 'foo'
```