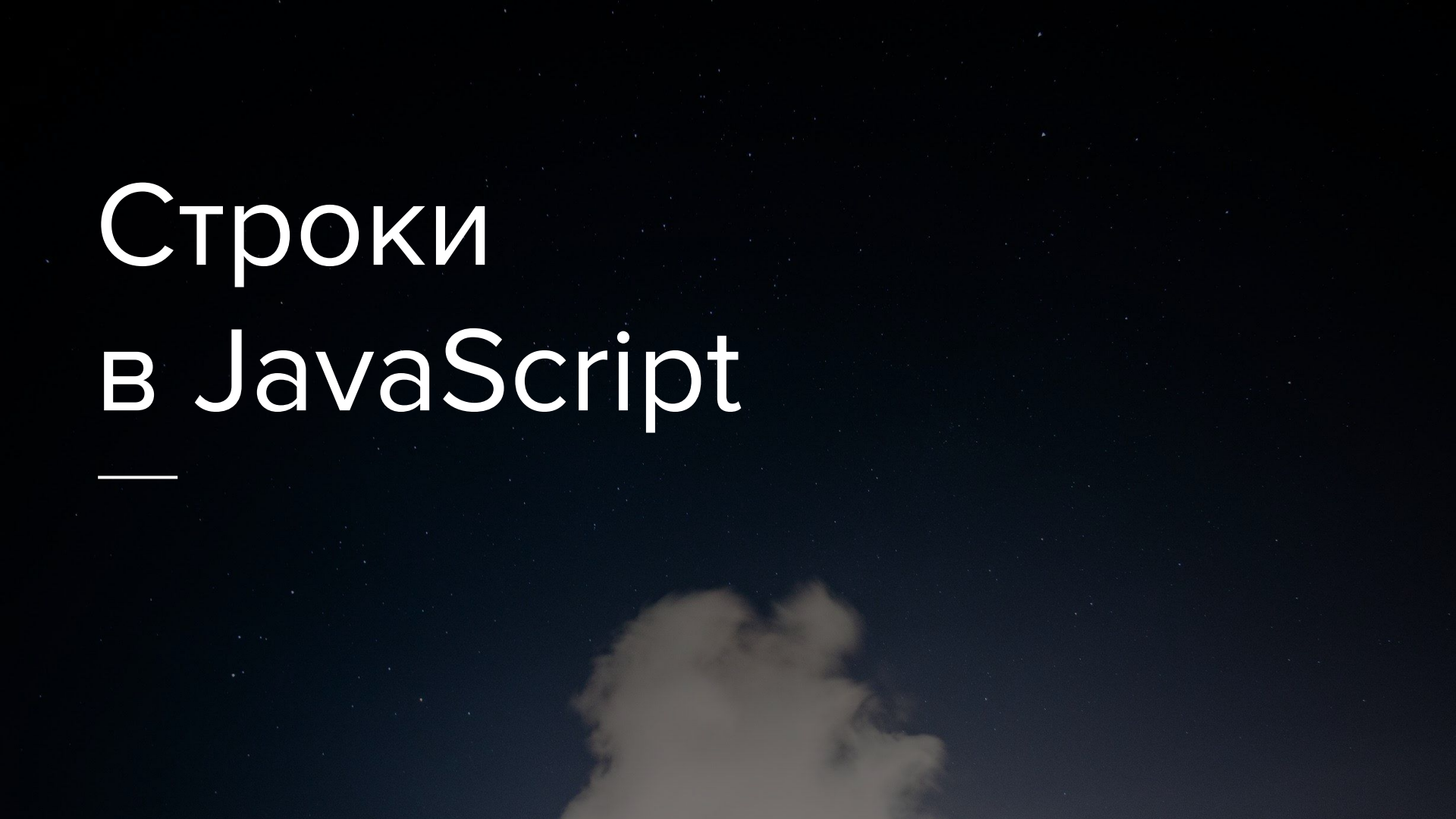
Строки
в JavaScript



Строки

Строки полезны для хранения данных, которые можно представить в текстовой форме.

Наиболее **частые операции со строками**:

- проверка их длины,
- построение строки с помощью операций строковой конкатенации + и +=,
- проверка на существование или местоположение подстрок,
- извлечение подстрок с помощью метода substring() и тд.

Внутренним форматом строк, вне зависимости от кодировки страницы, является Юникод (Unicode).

Содержимое строки в JavaScript нельзя изменять, можно лишь создать новую строку на основании старой.

Для **создания строковой переменной** используются двойные или одинарные кавычки.

```
let str = "Строковая переменная";
let str2 = 'Другая строка';
```

В строках могут быть использованы **специальные символы**:

- перенос строки \n
- табуляция \t
- \uNNNN символ в кодировке Юникод с шестнадцатеричным кодом `NNNN`. Например, \u00A9 и др

Для отображения специальных символов в строке, их нужно **экранировать** - поставить обратный слеш перед символом, например, \\n

Если строка в одинарных (двойных) кавычках, **то внутренние одинарные** (двойные) **кавычки тоже должны экранироваться**:

```
let str = "Строковая \"Строка в кавычках\" переменная";
```

Методы и свойства строк

Свойство **length** - **получение длины строки**.

Доступ к символам (нумерация начинается с 0)**:**

1) метод **charAt(индекс)** - получение символа по индексу
   если символа нет, метод вернет пустую строку
2) квадратные скобки:
   ```
   let str = "Hello";
   str[0];  // "H"
   ```
   если символа нет, результат будет равен undefined

Смена регистра

1) метод **toLowerCase()** меняет регистр на нижний
2) метод **toUpperCase()** меняет регистр на верхний

Методы и свойства строк

Метод **indexOf(searchValue[, fromIndex])** - используется для поиска подстроки.
Возвращает индекс первого вхождения указанного значения (**searchValue**) в строку. Поиск начинается с индекса **fromIndex**, если не указан (или меньше 0) поиск осуществляется с начала строки.
Если fromIndex >= str.length, метод вернёт -1.
Возвращает -1, если ничего не найдено.

Чтобы найти **все вхождения подстроки**, **нужно запустить indexOf в цикле**.

Методы и свойства строк

Взятие подстроки: substring, substr, slice

Метод **substring(indexA[, indexB])** - возвращает подстроку с позиции **indexA** до, но не включая **indexB**.

- Если **indexA** равен **indexB**, метод substring() вернёт пустую строку;
- Если аргумент **indexB** опущен, метод извлечёт символы до конца строки;
- Если любой из аргументов меньше, либо равен нулю или равен NaN, он будет трактоваться как равный 0;
- Если любой из аргументов больше, чем длина строки, он будет трактоваться как равный длине строки
- Если **indexA** будет больше **indexB**, то метод сработает так, как если бы аргументы были поменяны местами;

Методы и свойства строк

Взятие подстроки: substring, substr, slice

Метод **substr(start[, length])** - возвращает из строки
указанное количество символов (**length)**, начиная с позиции (**start).**

- если **start** - отрицательное число, оно трактуется как (длина_строки - start);
- если **start** больше, либо равен длине строки, метод вернёт пустую строку;
- если **start** - отрицательное число, метод использует его как индекс символа, начиная с конца строки.
- если **start** - отрицательное число и по модулю больше длины строки, метод будет использовать 0 в качестве начального индекса;
- если **length** равен нулю или отрицателен, метод substr() вернёт пустую строку; если **length** опущен, метод substr() извлечёт все символы до конца строки.

Методы и свойства строк

Взятие подстроки: substring, substr, slice

Метод **slice(beginSlice[, endSlice])** - возвращает часть строки от **beginSlice** до, но не включая, **endSlice**. Смысл параметров как в substring.

- если **beginSlice** отрицателен, то трактуется как (длина_строки + beginSlice)
- если **endSlice** опущен, slice() извлечёт всё до конца строки.
- если **endSlice** отрицателен, то трактуется как (длина_строки + beginSlice)

Методы строк

“добавленные ES6/ES2015”

Метод **includes(searchString[, position])** - проверяет, содержит ли строка заданную подстроку (searchString), и возвращает, true или false.

Поиск начинается с начала строки, если не задана позиция в строке (position), с которой нужно начинать поиск строки.

Метод includes() **является регистрозависимым**

Метод **endsWith(s)** – возвращает true, если строка заканчивается подстрокой s.

Метод **startsWith(s)** – возвращает true, если строка начинается со строки s.

Метод **repeat(times)** – повторяет строку times раз.

Сравнение строк

Все строки **имеют внутреннюю кодировку Юникод**.
Внутри JavaScript-интерпретатора все строки приводятся к единому виду.
Каждому символу соответствует свой код.

При сравнении с помощью операторов сравнения символы сравниваются не по алфавиту, а **по коду**.

String.fromCharCode(code) - возвращает символ по коду code
str.charCodeAt(pos) - возвращает код символа на позиции pos

Метод str1.**localeCompare**(str2) - возвращает:

- **-1**, если str1 < str2,
- **1**, если str1 > str2
- **0**, если они равны.

Дополнительный материал:

- **Подробнее о localeCompare**
- **Интернационализация в JS**

Строки - шаблоны

“возможности ES6/ES2015”

Обратные кавычки ` ` используются для:

- **переноса строки**

Например, ***console***.log(`**строка**
в обратных кавычках`);

- **интерполяции** - вставки выражений при помощи ${...}

Например,

```
'use strict';
let height = 100;
let width = 20;

console.log(`высота равна ${height},
            ширина равна ${width},
              площадь равна ${height * width}`);
```

Функции шаблонизации “возможности ES6/ES2015”

используются **для написания собственных функций шаблонизации строк**.

Название такой функции ставится перед первой обратной кавычкой.

Функция будет автоматически вызвана и получит в качестве аргументов:

1) строку, разбитую по вхождениям параметров ${...}
2) параметры, добавленные через ${...}.

```
let login = 'user495';

let str = myFunc`Логин пользователя ${login}. \n Пароль скрыт`;

function myFunc(strings, ...values) {

   console.log(strings, values);

}
```