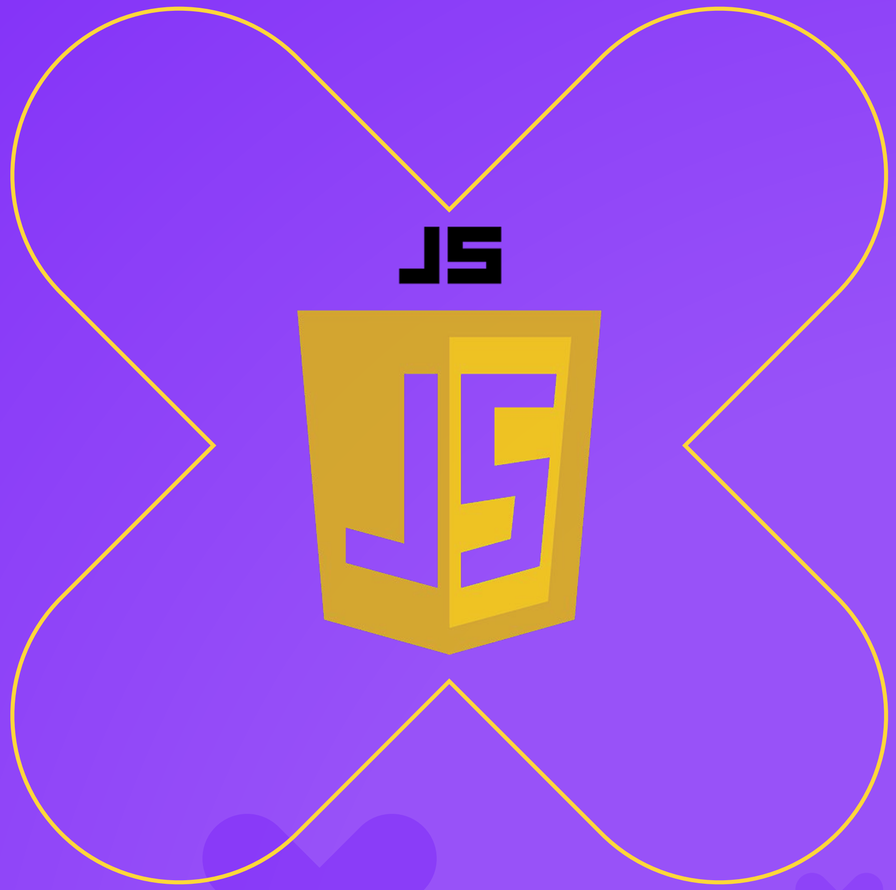
STEP ACADEMY

ЗАНЯТИЕ №12

# Строки

## Тип данных string



# Строковый тип данных

Строковый тип данных предназначен для сохранения в переменную набора символов (текста).

Строки записываются в кавычках: парных, одиночных и косых.

```
"string"
'string'
`string`
```

Кавычки нужны для того, чтобы отличать текст от кода.

# Литералы

В программировании часто встречается понятие “литерал”.
Литерал - это конкретное значение. Каждый раз когда вы пишите какое-то число, значение true или false, строку или массив - это литерал.

```
let s = "my string";
```

my string - это литерал

# Экранируемые символы

В строках можно передать особые символы, такие как перенос строки, табуляция и т.д. Для этого используется символ \

Вот некоторые символы:

\n  Перевод строки (New line)
\r Возврат каретки (Carriage return)
\t Табуляция (Tab)
\' Апостроф или одинарная кавычка
\" Двойная кавычка
\\ Обратная косая черта (Backslash)

# Разница между " ' `

Разница между одиночной кавычкой и двойной несущественна - в двойных кавычках можно не экранировать одиночную, в одиночных - можно не экранировать двойные.

Запись строки в косых кавычках (апострофах) значительно выше - это так называемые шаблонные строки.

# Шаблонные строки

Есть несколько особенностей шаблонных строк.

1. Шаблонные строки допускают разрывы внутри строки, в отличие от обычных.

```
"одна
строка" //тут ошибка
`одна
строка` //а тут нет
```

# Шаблонные строки

2. В шаблонные строки можно легко встраивать Javascript выражения. Для этого нужно будет вставить комбинацию символов ${}.
В фигурных скобках можно указывать выражения.

```
let a = 10;
let str = `значение переменной = ${a}`
```

# Работа со строками

Строки в Javascript работают как коллекция (набор) символов, поэтому мы можем получить любой символ в строке по индексу.

```
let str = "слово"
console.log(str[0]) //получим "с«
```

ВАЖНО! В отличие от массива или других коллекций изменить отдельный символ нельзя.

Также можно использовать запись str.length для получения длины строки.

# Функции для работы со строками

Функции для работы с массивами изменяли массивы.

Функции для работы со строками создают и возвращают копии исходной строки.

**ЭТО ВАЖНО!**

# Изменение регистра строки

```
let str = "слово"

let lowStr = str.toLowerCase();//заглавные буквы

let upStr = str.toUpperCase();//строчные буквы
```

# Вырезание подстроки

str.substring(2, 5) //вырежет подстроку с 2 символа до 5 (не включительно).

str.substring(2) //вырежет подстроку с 2 символа до конца.

slice работает почти так же, как и substring, но в slice можно писать отрицательные значения.

Отрицательные значения будут считаться от конца строки.

# Превращаем строку в массив

str.split(“,”); //выдаст массив из подстрок, разделенных запятой

str.split(“”); //выдаст массив из символов строки

# Убираем пробелы

str.trim() //удалит все пробельные символы в начале и конце строки.

Есть вариации работы функции trim - trimLeft и trimRight

# Повторяем строку

str.repeat(5); //повторит строку указанное количество строк

# Поиск по строке

str.indexOf(“a”); //вернет индекс первого вхождения буквы “a”

Если буква “a” не встречается в строке, тогда получим значение -1.

# Конкатенация строк

Строки можно соединять применяя к ним операцию +

# Конец