# Тема 4. Строки и шаблонные строки. Методы строк.


хекслет колледж

# Цель занятия:

Познакомить студентов с основами работы со строками в JavaScript, включая методы строк и шаблонные строки, а также показать практическое использование для форматирования текста и вывода данных.

# Учебные вопросы:

1. Понятие строки в JavaScript
2. Шаблонные строки (Template literals)
3. Основные методы строк
4. Конкатенация строк

# 1. Понятие строки в JavaScript

Строка (string) — это тип данных, предназначенный для хранения текста.
Строки используются для:

- слов и предложений,
- вывода сообщений пользователю,
- хранения имён, адресов, описаний,
- работы с HTML-текстом.

Примеры строк:

```
let name = "Иван";
let city = ‘Алматы';
let message = "Добро пожаловать!";
```

# Способы создания строк

В JavaScript строку можно записать:

- в двойных кавычках "...",
- в одинарных кавычках '...'.

Оба способа равноценны.

Выбор кавычек — вопрос стиля, главное — использовать их последовательно.

# Длина строки

Длина строки — это количество символов в ней.

Для получения длины используется свойство **length**:

```
let text = "Привет";
console.log(text.length); // 6
```

# Неизменяемость строк

Строки в JavaScript являются неизменяемыми (immutable):
нельзя изменить отдельный символ напрямую,
любые изменения создают новую строку.

# Итог:

- Строка — это тип данных для хранения текста.
- Строки записываются в одинарных или двойных кавычках.
- Строки имеют длину, доступную через length.
- Строки в JavaScript неизменяемы.

# 1. Введение

Улучшение

# 2. Шаблонные строки (Template literals)

Шаблонные строки (template literals) — это современный способ работы со строками в JavaScript, который:
упрощает создание строк,
делает код более читаемым,
позволяет удобно вставлять переменные и выражения.

Шаблонные строки записываются с помощью обратных кавычек:

`...`

# Создание шаблонной строки

Пример простой шаблонной строки:

```
let text = `Привет, мир!`;
console.log(text);
```

# Вставка переменных и выражений

В шаблонных строках можно вставлять:
переменные,
математические выражения,
результаты вычислений.
Для этого используется конструкция **${...}**:

```
let userName = "Анна";
let userAge = 20;

let message = `Меня зовут ${userName}, мне ${userAge} лет.`;
console.log(message);
```

Внутри ${} можно писать любое корректное выражение JavaScript:

```
let a = 5;
let b = 3;
console.log(`Сумма: ${a + b}`); // Сумма: 8
```

# Многострочные строки

Шаблонные строки позволяют писать текст в несколько строк без специальных символов:

```
let text = `Первая строка
Вторая строка
Третья строка`;

console.log(text);
```

Это удобно для:

- сообщений,
- описаний,
- HTML-разметки (на базовом уровне).

# Сравнение с обычными строками

Без шаблонных строк:

```
let message = "Привет, " + name + "! Тебе " + age + " лет.";
```

С шаблонными строками:

```
let message = `Привет, ${name}! Тебе ${age} лет.`;
```

Второй вариант короче и легче читается.

# Итог:

- Шаблонные строки записываются в обратных кавычках `.
- ${...} позволяет вставлять значения и выражения внутрь строки.
- Поддерживаются многострочные строки.
- Шаблонные строки позволяют удобно формировать текст, комбинируя строки и данные без сложной конкатенации.

# 3. Основные методы строк

## Изменение регистра

**toUpperCase()**

Преобразует строку в верхний регистр.

```
let word = "hello";
console.log(word.toUpperCase()); // HELLO
```

## toLowerCase()

Преобразует строку в нижний регистр.

```
let word = "HELLO";
console.log(word.toLowerCase()); // hello
```

# Удаление пробелов

**trim()**
Удаляет пробелы в начале и в конце строки.

```
let input = "   текст   ";
console.log(input.trim()); // "текст"
```

Полезно при обработке данных, введённых пользователем.

# Поиск и проверка наличия подстроки

**includes()**

Проверяет, есть ли подстрока в строке.

Возвращает true или false.

```
let text = "Изучаем JS";
console.log(text.includes("JS")); // true
```

**startsWith()**
Проверяет, начинается ли строка с указанного текста.

```
console.log("Hello world".startsWith("Hello")); // true
```

**endsWith()**
Проверяет, заканчивается ли строка указанным текстом.

```
console.log("index.html".endsWith(".html")); // true
```

# Получение части строки (срез)

**slice(start, end)**

Возвращает часть строки от индекса start до end (не включая end).

```
let str = "JavaScript";
console.log(str.slice(0, 4)); // Java
```

# Замена части строки

**replace()**
Заменяет первую найденную подстроку на новую.

```
let text = "Я изучаю JS";
console.log(text.replace("JS", "JavaScript"));
// Я изучаю JavaScript
```

# Разделение строки

**split()**
Разделяет строку на массив по указанному разделителю.

```
let sentence = "HTML CSS JavaScript";
let words = sentence.split(" ");
console.log(words); // ["HTML", "CSS", "JavaScript"]
```

# Другие методы строк:

https://learn.javascript.ru/string

# Итог:

- Методы строк позволяют анализировать и преобразовывать текст.
- length, trim(), includes(), slice(), replace(), split() — наиболее часто используемые.
- Методы упрощают работу с пользовательским вводом и текстовым содержимым сайта.
- Все методы строк не изменяют исходную строку, а возвращают новую.

# Итог лекции:

- Строки используются для сообщений, данных и интерфейса.
- Методы строк помогают проверять, обрабатывать и форматировать текст.
- Шаблонные строки делают код проще и понятнее, используются для многострочного текста.

# Домашнее задание:

https://ru.hexlet.io/courses/js-basics

хекслет колледж

@HEXLY.KZ