**Создание регулярного выражения**

1. Литерал регулярного выражения:

const regex = /pattern/;

2. Конструктор RegExp:

const regex = new RegExp('pattern');

Основные символы

**• .:** Соответствует любому одиночному символу, кроме новой строки.

const regex = /c.t/;

console.log(regex.test('cat')); // true

**• ^**: Начало строки.

const regex = /^Hello/;

console.log(regex.test('Hello world')); // true

**• $**: Конец строки.

const regex = /world$/;

console.log(regex.test('Hello world')); // true

**• \***: Соответствует нулю или более повторений предыдущего символа.

const regex = /bo\*/;

console.log(regex.test('booo')); // true

**• +**: Соответствует одному или более повторений предыдущего символа.

const regex = /bo+/;

console.log(regex.test('booo')); // true

**• ?**: Соответствует нулю или одному повторению предыдущего символа.

const regex = /colou?r/;

console.log(regex.test('color')); // true

console.log(regex.test('colour')); // true

**• d**: Соответствует любой цифре (эквивалентно [0-9]).

const regex = /d+/;

console.log(regex.test('123')); // true

**• D**: Соответствует любому нецифровому символу.

const regex = /D+/;

console.log(regex.test('abc')); // truе

**• w**: Соответствует любому алфавитно-цифровому символу или подчеркиванию (эквивалентно [A-Za-z0-9\_]).

const regex = /w+/;

console.log(regex.test('hello\_123')); // true

**• W**: Соответствует любому не алфавитно-цифровому символу.

const regex = /W+/;

console.log(regex.test('!@#')); // true

**• s**: Соответствует любому пробельному символу (включая пробел, табуляцию и новую строку).

const regex = /s+/;

console.log(regex.test(' ')); // true

**• S**: Соответствует любому непробельному символу.

const regex = /S+/;

console.log(regex.test('abc')); // true

**Квантификаторы**

**• {n}:** Ровно n повторений предыдущего символа.

const regex = /a{3}/;

console.log(regex.test('aaa')); // true

**• {n,}:** n или более повторений предыдущего символа.

const regex = /a{2,}/;

console.log(regex.test('aaaa')); // true

**• {n,m}:** От n до m повторений предыдущего символа.

const regex = /a{2,4}/;

console.log(regex.test('aaa')); // true

**Группы и альтернативы**

**• Группы ():** Используются для группировки частей выражения.

const regex = /(abc)+/;

console.log(regex.test('abcabc')); // true

**• Альтернативы** |: Соответствует любой из альтернатив.

const regex = /cat|dog/;

console.log(regex.test('cat')); // true

**Обратные ссылки**

Обратные ссылки позволяют ссылаться на ранее захваченные группы.

const regex = /(a)1/;

console.log(regex.test('aa')); // true

**▎Флаги**

• g: Глобальный поиск.

• i: Нечувствительный к регистру поиск.

• m: Многострочный поиск.

• s: Позволяет . соответствовать символам новой строки.

• u: Включает поддержку Unicode.

• y: "Стик" режим, поиск начинается с текущей позиции в строке.

const regex = /hello/i; // Нечувствительный к регистру поиск "hello"

**Методы работы с регулярными выражениями**

1. test(): Проверяет, соответствует ли строка регулярному выражению.

const regex = /hello/;

console.log(regex.test('hello world')); // true