

ПМ3 Разработка модулей ПО.

РО 3.1 Понимать и применять принципы объектно-ориентированного и асинхронного программирования.

Тема 2. Регулярные выражения.

Лекция 3. Основы регулярных выражений

Цель занятия:

Познакомиться регулярными выражениями в JavaScript как инструментом для поиска и обработки текста.

Учебные вопросы:

1. Методы работы с регулярными выражениями в JavaScript.

1. Методы работы с регулярными выражениями в JavaScript.

Регулярные выражения в JS можно применять через методы объекта **RegExp** и строк (**String**).

◆ Методы у RegExp

test(str)

Проверяет: есть ли совпадение?

Возвращает true или false.

```
const re = /abc/;  
  
console.log(re.test("123abc456")); // true (строка содержит "abc")  
console.log(re.test("123456"));    // false (нет "abc")
```

exec(str)

Находит совпадение и возвращает подробную информацию:

- совпадение целиком,
- где оно нашлось (индекс),
- исходная строка.

```
const re = /abc/;  
console.log(re.exec("123abc456"));
```

```
// [ 'abc', index: 3, input: '123abc456', groups: undefined ]
```

Если совпадений нет → null.

◆ Методы у String

match(re)

Ищет совпадения.

- Без флага g → возвращает первое совпадение и подробности.
- С флагом g → возвращает массив всех совпадений.

```
"abc abc".match(/abc/);  
// ['abc', index: 0, input: 'abc abc', groups: undefined]  
  
"abc abc".match(/abc/g); // ['abc', 'abc']
```


search(re)

Возвращает индекс первого совпадения (позицию в строке) или -1, если не найдено.

```
"abc123".search(/123/);    // 3  
"abc123".search(/zzz/);    // -1
```

`replace(re, newStr)`

Заменяет совпадение на другой текст.

По умолчанию заменяет только первое совпадение.
Чтобы заменить все — нужен флаг `g`.

```
"2025-09-17".replace("-", ".");           // "2025.09-17"  
"2025-09-17".replace(/-/g, ".");         // "2025.09.17"
```

split(re)

Разбивает строку по разделителю.

Можно использовать регулярку, если разделителей несколько.

```
"a,b;c".split(/[.,;]/);    // ['a', 'b', 'c']
```

Краткие итоги:

- **test** — проверить, есть ли совпадение.
- **match** — получить совпадение/совпадения.
- **search** — узнать позицию.
- **replace** — замена.
- **split** — разбиение строки.

Практические примеры:

1. Проверка имени пользователя

Пусть нужно проверить, что в имени есть слово "Ivan".

```
const re = /Ivan/;

console.log(re.test("Ivan Petrov"));    // true
console.log(re.test("Petr Ivanov"));    // true (тоже содержит "Ivan")
console.log(re.test("Petrov"));        // false
```

2. Поиск всех имён "Anna" (без учёта регистра)

```
const text = "Anna likes coffee. anna goes to school. ANNA is happy.";
console.log(text.match(/anna/gi));
// ['Anna', 'anna', 'ANNA']
```

3. Замена домена в email

```
const email = "student@mail.com";  
const result = email.replace(/mail\.com/, "university.edu");  
console.log(result);  
// "student@university.edu"
```

4. Форматирование телефона

```
const phone = "8-999-123-45-67";  
// Заменим дефисы на пробелы:  
console.log(phone.replace(/-/g, " "));  
// "8 999 123 45 67"
```


5. Поиск года в тексте

```
const text = `я родился в 2000 году. сейчас мне 25 лет.  
Я окончил школу № 123 в 2007 году.`;  
console.log(text.match(/\d{4}/g));
```

Как работает `\d{4}/g`

- `\d` → цифра (0–9)
- `{4}` → ровно 4 раза подряд
- `g` → ищем все совпадения

6. Разделение списка email-адресов

```
const emails = "anna@mail.com; ivan@mail.com, petr@mail.com";  
console.log(emails.split(/[;,]/));  
// ["anna@mail.com", " ivan@mail.com", " petr@mail.com"]
```

7. Проверка, есть ли цифры в пароле

```
const password1 = "qwerty";  
const password2 = "qwerty123";  
  
console.log(/\d/.test(password1)); // false  
console.log(/\d/.test(password2)); // true
```

Контрольные вопросы:

- Что такое регулярные выражения? Для чего они нужны?
- Какими двумя способами можно создать регулярное выражение в JavaScript?
- Что делают флаги g, i, m?
- Чем отличается test от match?
- В чём разница между replace(/.../, ...) и replace(/.../g, ...)?

Домашнее задание:

1. <https://ru.hexlet.io/courses/regexp>