# ПМ3 Разработка модулей ПО.

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

# Тема 2. Регулярные выражения.

# Лекция 1. Основы регулярных выражений

# Цель занятия:

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

# Учебные вопросы:

1. Что такое регулярные выражения?
2. Создание регулярных выражений в JavaScript.

# 1. Что такое регулярные выражения?

**Регулярные выражения** (RegExp) — это специальный язык шаблонов для работы со строками.

С их помощью можно искать, проверять, заменять и извлекать текст по заданному правилу.

**Зачем нужны?**

Они помогают автоматизировать работу со строками, там, где обычные методы уже не справляются.

**Основные задачи:**

- Проверка соответствия шаблону
- Поиск фрагментов текста
- Замена по шаблону
- Разделение строки по сложному разделителю

**Когда используются в разработке?**

- Валидация данных (email, телефон, пароль)
- Поиск и подсветка текста (поиск в редакторе, фильтрация)
- Обработка логов, CSV, HTML, JSON, markdown
- Нормализация и замена формата данных (например, даты или номера телефона)
- Написание парсеров и простых анализаторов текста

## ⚡ Итог:

Регулярные выражения — это универсальный инструмент для работы с текстом по правилам.

Они удобны, когда нужно не просто найти подстроку, а проверить или обработать текст по сложному условию.

# 2. Создание регулярных выражений в JavaScript.

## 1. Литерал регулярного выражения

- Записывается между косыми слэшами **/.../**
- Флаги указываются после второго слэша.

```
const re1 = /hello/;        // ищет "hello"
const re2 = /hello/i;       // ищет "hello", без учёта регистра
```

✅ Плюсы:

- Кратко и читаемо
- Используется чаще всего

❌ Минус:

- Нельзя динамически формировать строку-шаблон (например, из переменной)

## 2. Конструктор RegExp

Позволяет создавать регулярку из строки.

```
const word = "cat";
const re = new RegExp(word, "gi");
console.log("Cat catalog".match(re)); // ["Cat", "cat"]
```

✅ Плюсы:

- Удобно, если шаблон нужно собрать из переменных
- Флаги передаются как строка вторым аргументом

❌ Минус:

- Нужно экранировать \ дважды: в строках JS и в регулярке

```
const re = new RegExp("\\d+"); // эквивалент /\d+/
```

## Где использовать регулярные выражения?

Регулярки применяются в методах:

- у объекта RegExp → .test(), .exec()
- у строки (String) → .match(), .matchAll(), .replace(), .replaceAll(), .search(), .split()

### ◇ Итог

- Для статических шаблонов (заранее известные выражения) используют литерал /.../.
- Для динамических шаблонов (собранных из переменных) используют new RegExp().

# Контрольные вопросы:

- Что такое регулярные выражения? Для чего они нужны?
- Какими двумя способами можно создать регулярное выражение в JavaScript?

# Домашнее задание:

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