|  |  |
| --- | --- |
| ДИСЦИПЛИНА | Технологии индустриального программирования |
| ИНСТИТУТ | Институт перспективных технологий и индустриального программирования |
| КАФЕДРА | Кафедра индустриального программирования |
| ВИД УЧЕБНОГО МАТЕРИАЛА | Практическое занятие |
| ПРЕПОДАВАТЕЛЬ | Адышкин Сергей Сергеевич |
| СЕМЕСТР | 2 семестр, 2024-2025 гг. |

# 

# Практическое занятие №2: Продвинутая разработка JavaScript

## Тема: Асинхронность: Promises и async/await

## Цели занятия:

- Понять основы работы с асинхронным кодом в JavaScript.

- Научиться использовать Promises для обработки асинхронных операций.

- Освоить синтаксис async/await.

## Инструкции:

1. Создайте функцию, которая возвращает Promise и симулирует задержку с использованием setTimeout.

2. Напишите цепочку then/catch для обработки результатов Promises.

3. Перепишите код с использованием async/await.

## Примеры кода:

```javascript  
const delayedPromise = (message, delay) => {  
 return new Promise((resolve, reject) => {  
 setTimeout(() => {  
 if (message) {  
 resolve(message);  
 } else {  
 reject('No message provided');  
 }  
 }, delay);  
 });  
};  
  
// Using then/catch  
delayedPromise('Hello, Async!', 2000)  
 .then((message) => console.log(message))  
 .catch((error) => console.error(error));  
  
// Using async/await  
async function runAsync() {  
 try {  
 const message = await delayedPromise('Hello, Async/Await!', 2000);  
 console.log(message);  
 } catch (error) {  
 console.error(error);  
 }  
}  
runAsync();  
```

## Дополнительные задания:

- Реализуйте функцию с несколькими последовательными асинхронными вызовами.

- Добавьте обработку ошибок в цепочке then/catch.

- Используйте Promise.all для параллельного выполнения асинхронных операций.