Молдавский Государственный Университет  
Факультет Математики и Информатики  
Департамент Информатики

**Лабораторная работа №1  
по курсу “JavaScript/TypeScript”**

Проверил: Nartea Nichita  
Выполнил: студент группы I2302,  
Михайлов Пётр

Кишинев, 2024

Цель индивидуальной работы:

Ознакомить студентов с продвинутыми функциями JavaScript, включая асинхронный JavaScript, модули и обработку ошибок.

Описание задания:

Разработать мини-приложение для предложения капитану Смиту нового занятия при каждом обновлении. Приложение должно обновляться каждую минуту и предоставлять случайную активность.

Структура проекта:

1. Файл index.html: основная HTML-структура веб-страницы.
2. Файл index.css: определение стилей страницы.
3. Файл index.js: основной файл JavaScript, который импортирует функцию для обновления активность и запуская её.
4. Файл activity.js: файл с логикой получения данных о случайной активности и их обновления.

Анализ кода:

index.js

import { updateRandomActivityEveryMinute } from './activity.js';

updateRandomActivityEveryMinute();

В этом файле импортируется функция из модуля activity.js, которая затем вызывается.

activity.js

console.log("1");

export async function getRandomActivity() {

  try {

    const response = await fetch('https://www.boredapi.com/api/activity/');

    const data = await response.json();

    return data.activity;

  } catch (error) {

    console.error("Ошибка при получении данных:", error);

    return "К сожалению, произошла ошибка";

  }

}

export async function updateActivity(activity){

  const element = document.querySelector("#activity");

  element.textContent = activity;

}

export async function updateRandomActivityEveryMinute() {

  async function refreshActivity() {

    const activity = await getRandomActivity();

    updateActivity(activity);

  }

  await refreshActivity();

  setInterval(refreshActivity, 60000);

}

Этот файл содержит функции для получения случайной активности (getRandomActivity), ее обновления на странице (updateActivity) и обновления активности каждую минуту (updateRandomActivityEveryMinute).

Используются асинхронные функции и ключевые слова async/await для работы с асинхронными операциями.

Для получения данных о случайной активности используется API https://www.boredapi.com/api/activity/.

Обработка возможных ошибок при получении данных реализована с помощью конструкции try...catch.

Вывод:

Данный проект успешно реализует поставленную задачу. Использование асинхронных функций и обработка ошибок делает приложение надежным.

Лабораторная работа №2 демонстрирует применение продвинутых функций JavaScript для создания полезного приложения. Она позволяет студентам углубить свои знания в области асинхронного программирования, модульности и обработки ошибок.

Ссылка на репазиторий Git:

https://github.com/mihailovp04/JavaScript/tree/main/LI2

Ответы на контрольные вопросы:

1. Какое значение возвращает функция fetch?

- Функция fetch() возвращает объект Promise.

2. Что представляет собой Promise?

- Promise представляет собой объект, представляющий результат асинхронной операции. Он может быть либо в состоянии ожидания, либо выполнен, либо отклонен.

3. Какие методы доступны у объекта Promise?

- У объекта Promise доступны методы:

- then(): обработка успешного выполнения операции.

- catch(): обработка ошибок.

- finally(): выполнение действий после завершения операции, независимо от ее результата.

4. Каковы основные различия между использованием async/await и Promise?

- async/await позволяет писать асинхронный код в синхронном стиле, делая его более читаемым.

- async/await автоматически обрабатывает промисы, что упрощает их использование.

- С помощью async/await легче обрабатывать ошибки с помощью конструкции try...catch.

Список использованных источников

1. Chat GPT
2. Современный учебник JavaScript
3. Руководство по JavaScript
4. YouTube