Молдавский Государственный Университет  
Факультет Математики и Информатики  
Департамент Информатики

**Индивидуальная работа №2  
по курсу “JavaScript/TypeScript”**

Проверил: Nartea Nichita  
Выполнил: студент группы I2302,  
Питропов Александр

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();

Файл index.js импортирует функцию updateRandomActivityEveryMinute из файла activity.js и вызывает её, запуская процесс обновления случайной активности с использованием внешнего API каждую минуту на веб-странице.

activity.js

export async function updateRandomActivityEveryMinute() {

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";

}

}

async function updateActivity(activity){

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

element.textContent = activity;

}

async function refreshActivity() {

const activity = await getRandomActivity();

updateActivity(activity);

setTimeout(refreshActivity, 60000);

}

await refreshActivity();

}

Этот файл содержит функции для работы с случайной активностью: getRandomActivity для получения активности, updateActivity для её обновления на странице и updateRandomActivityEveryMinute для обновления активности каждую минуту. Асинхронные операции обрабатываются с помощью ключевых слов async/await. Данные о случайной активности получаются с использованием API https://www.boredapi.com/api/activity/. Возможные ошибки при получении данных обрабатываются с помощью конструкции try...catch.

Вывод:

Проект успешно выполняет задачу с помощью использования асинхронных функций и обработки ошибок, что повышает надёжность приложения. Лабораторная работа №2 представляет собой отличный пример использования продвинутых возможностей JavaScript для создания полезного приложения. Она обеспечивает студентам возможность углубить свои знания в области асинхронного программирования, модульности и обработки ошибок.

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

Ссылка на репозиторий :

https://github.com/alexunderpitropov/individualka