ГУАП

КАФЕДРА № 41

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ассистент |  |  |  | Н. И. Мирошниченко |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №2 |
| Практическое задание №2 по курсу: ОСНОВЫ FRONTEND-РАЗРАБОТКИ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ гр. № | 4217 |  |  |  | В. А. Милованов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2025

**Цель работы**

Формирование практических навыков работы с асинхронным кодом в языке JavaScript, используя callback-функции и объект Promise.

**Ход работы**

1. Напишите функцию `filterArray`, которая принимает массив чисел и callback-функцию. Функция `filterArray` должна вызывать callback функцию для каждого элемента массива и возвращать новый массив, содержащий только те элементы, для которых callback-функция вернула `true`.

Напишите 2 примера применения этой функции. К примеру, для фильтрации четных и нечетных значений массива.

Была реализована функция filterArray(), принимающая аргументы в виде arr массива и callback функции. Для обработки массива использовался метод filter() (см. рис. 1).

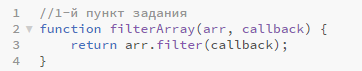


Рисунок 1 – Функция filterArray()

Для проверки работы функции был создан array массив и две стрелочные функции проверяющие четность и нечетность элементов массива. Вывод результатов производился с помощью метода alert() (см. рис. 2-3).

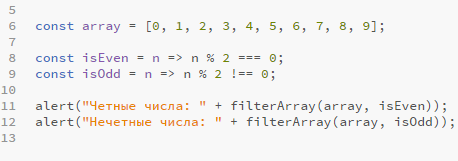


Рисунок 2 – Проверка работы функции

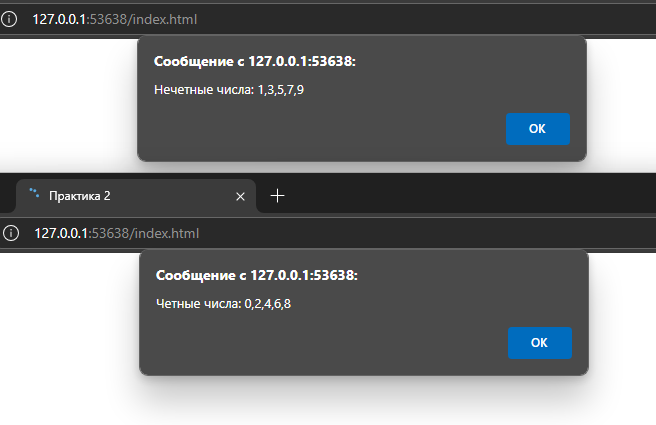


Рисунок 3 – Результат вывода

2-я часть задания

2. Напишите асинхронную функцию `fetchData`, которая принимает URL в качестве параметра и возвращает Promise. Функция должна использовать `fetch` для получения данных с указанного URL. Если запрос прошел успешно (статус ответа 200), Promise должен быть разрешен с полученными данными в виде строки. Если запрос не удался (любой другой статус), Promise должен быть отклонен с сообщением об ошибке.

Была создана функция fetchData(), которая отправляет запрос по указанному url параметру. На случай возникновения ошибок при отправке запроса использовались условие !request.ok и конструкция try…catch для отклонения Promise при неудавшемся запросе.



Рисунок 4 – Функция fetchData()

**Вывод**

В ходе выполнения практической работы мной были освоены навыки работы с JavaScript. Были изучены и реализованы важные аспекты работы с функциями, асинхронными запросами и обработкой ошибок. Создана функция filterArray, которая позволяет фильтровать массив чисел на основе переданной callback-функции. Это продемонстрировало полезность колбэков при динамической обработке данных, например, для фильтрации четных и нечетных чисел.

Кроме того, была разработана асинхронная функция fetchData, выполняющая HTTP-запрос с использованием fetch() и async/await. Такой подход делает взаимодействие с асинхронными операциями более удобным и читабельным. Важным этапом работы стала реализация корректной обработки ошибок. Для этого использовались два уровня контроля: проверка статуса HTTP-ответа с выбросом ошибки в случае неудачи и глобальный try...catch, перехватывающий возможные ошибки сети или сбои при обработке данных. В результате выполнения работы были закреплены ключевые принципы современной веб-разработки.