ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Доцент |  |  |  | О. И. Красильникова |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ПРАКТИЧЕСКОЙ РАБОТЕ №2 |
| по курсу: JavaScript, его библиотеки и фреймворки в Frontend-разработке |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 4134К |  |  |  | Д. В. Самарин |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург

2024

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

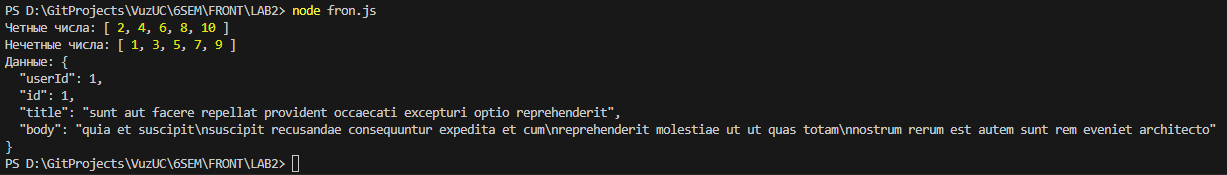
Задание:  
1.  Напишите функцию `filterArray`, которая принимает массив чисел и callback функцию. Функция `filterArray` должна вызывать callback функцию для каждого элемента массива и возвращать новый массив, содержащий только те элементы, для которых callback функция вернула `true`. Напишите 2 примера применения этой функции. К примеру, для фильтрации четных и нечетных значений массива.

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.

Ход работы:

В ходе работы был реализован js скрипт.

Ниже предоставлена возможность наглядно ознакомиться с результатом:



Листинг fron.js

|  |
| --- |
| // Задание 1: Функция filterArray  function filterArray(array, callback) {      const filteredArray = [];      array.forEach(element => {        if (callback(element)) {          filteredArray.push(element);        }      });      return filteredArray;    }      // Примеры применения функции filterArray    const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];      // Фильтрация четных значений    const evenNumbers = filterArray(numbers, number => number % 2 === 0);    console.log('Четные числа:', evenNumbers);      // Фильтрация нечетных значений    const oddNumbers = filterArray(numbers, number => number % 2 !== 0);    console.log('Нечетные числа:', oddNumbers);        // Задание 2: Асинхронная функция fetchData    function fetchData(url) {      return new Promise((resolve, reject) => {        fetch(url)          .then(response => {            if (response.status === 200) {              resolve(response.text());            } else {              reject(`Ошибка ${response.status}: ${response.statusText}`);            }          })          .catch(error => reject(error));      });    }      // Пример использования функции fetchData    fetchData('https://jsonplaceholder.typicode.com/posts/1')      .then(data => {        console.log('Данные:', data);      })      .catch(error => {        console.error('Ошибка:', error);      }); |

Вывод:

Выполнив практическую работу №1, я сформировал практические навыки работы с асинхронным кодом в языке JavaScript, используя callback-функции и объект Promise.