Частное учреждение образования

«Колледж бизнеса и права»

ОТЧЕТ

ПО Лабораторной работе №29

ОП Т.094001

Руководитель практики

от колледжа (Рогалевич А. В.)

Учащийся (Бакулин А. И.)

2024

**Задание 1.**

Создайте функцию, которая принимает на вход два аргумента - URL и таймаут в миллисекундах. Функция должна возвращать промис, который выполнится через указанный таймаут и вернет данные, полученные по указанному URL.

**Задание 2.**

Напишите функцию, которая принимает на вход массив промисов и возвращает новый промис, который выполнится, когда все промисы в массиве будут выполнены. Результатом должен быть массив значений, возвращенных каждым из промисов.

**Задание 3.**

Создайте промис, который выполнится через случайное время (от 1 до 5 секунд) и вернет случайное число от 1 до 100. Затем напишите функцию, которая будет вызывать этот промис и выводить результат в консоль.

**Задание 4.**

Напишите функцию, которая принимает на вход массив объектов, у каждого из которых есть поле "url". Функция должна создавать промис для каждого URL и возвращать массив промисов. Когда все промисы будут выполнены, функция должна вернуть массив объектов, в которых к каждому URL добавлено новое поле "data" с данными, полученными по этому URL.

**Задание 5.**

Напишите функцию, которая принимает на вход массив промисов и таймаут в миллисекундах. Функция должна возвращать новый промис, который выполнится через указанный таймаут и вернет массив результатов выполнения каждого из переданных промисов. Если какой-то из промисов не выполнится за указанный таймаут, то он должен быть прерван и вместо результата вернуться ошибку.

**Код**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Promises Tasks</title>

<style>

    /\* CSS стили могут быть добавлены здесь \*/

</style>

</head>

<body>

    <script>

        // Задание 1

        function fetchDataWithTimeout(url, timeout) {

            return new Promise((resolve, reject) => {

                setTimeout(() => {

                    fetch(url)

                        .then(response => {

                            if (!response.ok) {

                                throw new Error('Network response was not ok');

                            }

                            return response.json();

                        })

                        .then(data => resolve(data))

                        .catch(error => reject(error));

                }, timeout);

            });

        }

        // Задание 2

        function waitForAll(promises) {

            return Promise.all(promises);

        }

        // Задание 3

        function getRandomNumberWithDelay() {

            return new Promise((resolve, reject) => {

                const randomDelay = Math.floor(Math.random() \* 5000) + 1000; // от 1 до 5 секунд

                setTimeout(() => {

                    const randomNumber = Math.floor(Math.random() \* 100) + 1; // от 1 до 100

                    resolve(randomNumber);

                }, randomDelay);

            });

        }

        // Задание 4

        function fetchDataForUrls(urls) {

            const promises = urls.map(urlObj => {

                return new Promise((resolve, reject) => {

                    fetch(urlObj.url)

                        .then(response => {

                            if (!response.ok) {

                                throw new Error('Network response was not ok');

                            }

                            return response.json();

                        })

                        .then(data => {

                            urlObj.data = data;

                            resolve(urlObj);

                        })

                        .catch(error => reject(error));

                });

            });

            return Promise.all(promises);

        }

        // Задание 5

        function executePromisesWithTimeout(promises, timeout) {

            return Promise.allSettled(

                promises.map(p => Promise.race([p, new Promise((\_, reject) => setTimeout(() => reject(new Error('Promise timeout')), timeout))]))

            );

        }

        // Пример использования функций:

        const url = 'https://jsonplaceholder.typicode.com/posts/1';

        fetchDataWithTimeout(url, 3000)

            .then(data => console.log('Task 1:', data))

            .catch(error => console.error('Task 1 Error:', error));

        const promises = [

            fetch(url),

            fetch(url),

            fetch(url)

        ];

        waitForAll(promises)

            .then(data => console.log('Task 2:', data))

            .catch(error => console.error('Task 2 Error:', error));

        getRandomNumberWithDelay()

            .then(data => console.log('Task 3:', data))

            .catch(error => console.error('Task 3 Error:', error));

        const urls = [{ url: 'https://jsonplaceholder.typicode.com/posts/1' }, { url: 'https://jsonplaceholder.typicode.com/posts/2' }];

        fetchDataForUrls(urls)

            .then(data => console.log('Task 4:', data))

            .catch(error => console.error('Task 4 Error:', error));

        executePromisesWithTimeout(promises, 2000)

            .then(data => console.log('Task 5:', data))

            .catch(error => console.error('Task 5 Error:', error));

    </script>

</body>

</html>

**Реализация**

