Задание 1

function counter(n) {

    let current = n;

    let intervalId = setInterval(() => {

        console.log(current);

        if (current === 0) {

            clearInterval(intervalId);

        } else {

            current--;

        }

    }, 1000);

}

function createCounter(n) {

    let current = n;

    let intervalId;

    return {

        start() {

            intervalId = setInterval(() => {

                console.log(current);

                if (current === 0) {

                    clearInterval(intervalId);

                } else {

                    current--;

                }

            }, 1000);

        },

        pause() {

            clearInterval(intervalId);

        },

        stop() {

            clearInterval(intervalId);

            current = n;

        }

    };

}

// Пример использования:

counter(5); // Запускает счетчик от 5 до 0

const myCounter = createCounter(5);

myCounter.start(); // Запускает счетчик от 5 до 0

Задание 2

function delay(N) {

    return new Promise(resolve => {

        setTimeout(() => resolve(), N \* 1000);

    });

}

// Решение задачи со счетчиком

function countDownFrom(N) {

    if (N >= 0) {

        console.log(N);

        return delay(1).then(() => countDownFrom(N - 1));

    }

}

countDownFrom(5); // Запускает счетчик от 5 до 0

// Функция для получения первого репозитория по имени пользователя

function getFirstRepository(username) {

    return fetch(`[https://api.github.com/users/${username}`](https://api.github.com/users/$%7Busername%7D))

        .then(response => response.json())

        .then(user => {

            return fetch(user.repos\_url);

        })

        .then(response => response.json())

        .then(repos => repos[0].name);

}

// Пример использования:

getFirstRepository('iliakan').then(repoName => {

    console.log(`First repository for iliakan: ${repoName}`);

});

Задание 3

class HttpError extends Error {

    constructor(response) {

        super(`${response.status} for ${response.url}`);

[this.name](http://this.name/) = 'HttpError';

        this.response = response;

    }

}

async function loadJson(url) {

    let response = await fetch(url);

    if (response.status == 200) {

        return await response.json();

    } else {

        throw new HttpError(response);

    }

}

async function getGithubUser() {

    let name;

    do {

        name = prompt("Введите логин?", "iliakan");

        try {

            let user = await loadJson(`[https://api.github.com/users/${name}`](https://api.github.com/users/$%7Bname%7D));

            alert(`Полное имя: ${[user.name](http://user.name/)}.`);

            return user;

        } catch (err) {

            if (err instanceof HttpError && err.response.status == 404) {

                alert("Такого пользователя не существует, пожалуйста, повторите ввод.");

            } else {

                throw err;

            }

        }

    } while (true);

}

getGithubUser();