**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

###### ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

###### КЕМЕРОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

**ИНСТИТУТ ЦИФРЫ**

**ОТЧЁТ**

**О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

«Асинхронность - 2. Promise, async/await»

Студентки 2 курса, ФИТ-211 группы

**Колесник Полины Олеговны**

Направление 02.03.02 – «Фундаментальная информатика и информационные технологии»

Руководитель:

Доцент Зимин А. И.

Работа защищена

« »

“ ” 2023 г.

Кемерово 2023 г.

**ОТЧЁТ О ПРОДЕЛАННОЙ РАБОТЕ**

**1 задание**

let promise = new Promise(function(resolve, reject) {

    resolve(1);

    setTimeout(() => resolve(2), 1000);

});

promise.then(console.log);



**2 задание**

// Асинхронные функции

async function readConfig (name) {

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

        setTimeout(() => {

            resolve(console.log('(1) config from ' + name + ' loaded'))

        }, Math.floor(Math.random() \* 10000))

    })

}

async function doQuery (statement) {

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

        setTimeout(() => {

            resolve(console.log('(2) SQL query executed: ' + statement))

        }, Math.floor(Math.random() \* 1000))

    })

}

async function httpGet (url) {

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

        setTimeout(() => {

            resolve(console.log('(3) Page retrieved: ' + url))

        }, Math.floor(Math.random() \* 1000))

    })

}

async function readFile (path) {

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

        setTimeout(() => {

            resolve(console.log('(4) Readme file from ' + path + ' loaded'))

        }, Math.floor(Math.random() \* 1000))

    })

}

// Вызов функций

console.log('start')

readConfig('myConfig').

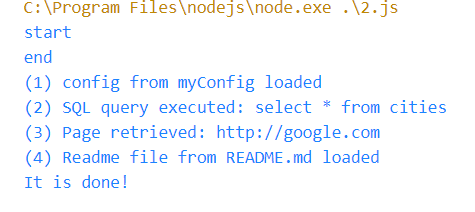
then(()=>doQuery('select \* from cities')).

then(()=>httpGet('http://google.com')).

then(()=>readFile('README.md')).

catch(()=>console.log('end'))

console.log('end')



**3 задание**

const x = 3

async function f1() {

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

        setTimeout(() => {

            const NewRes = x \*\* 2;

            console.log(`Промежуточный результат ${NewRes}`);

            resolve(NewRes);

        }, 0)

    })

}

async function f2(OldRes) {

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

        setTimeout(() => {

            let NewRes = OldRes + x \* 2;

            console.log(`Pезультат n = 2: ${NewRes}`)

            resolve(NewRes)

        }, 0)

    })

}

async function f3(OldRes) {

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

        setTimeout(() => {

            let NewRes = OldRes - 2;

            console.log(`Промежуточный результат ${NewRes}`)

            resolve(NewRes)

        }, 1000)

    })

}

async function f4(OldRes) {

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

        setTimeout(() => {

            let NewRes = OldRes \*\* 3;

            console.log(`Pезультат при n = 4: ${NewRes}`)

            resolve(NewRes)

        }, 10000)

    })

}

async function f5(OldRes) {

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

        setTimeout(() => {

            let NewRes = OldRes \* 3;

            console.log(`Промежуточный результат ${NewRes}`)

            resolve(NewRes)

        }, 1000)

    })

}

async function f6(OldRes) {

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

        setTimeout(() => {

            let NewRes = OldRes - 3;

            console.log(`Pезультат при n = 6: ${NewRes}`)

            resolve(NewRes)

        }, 1000)

    })

}

console.log('start')

f1().

then((a)=>f2(a)).

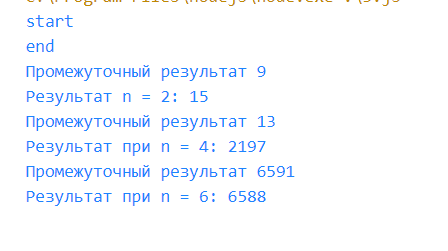
then((NewRes)=>f3(NewRes)).

then((NewRes)=>f4(NewRes)).

then((NewRes)=>f5(NewRes)).

then((NewRes)=>f6(NewRes))

console.log('end')



**4 задание**

let summ = function(value) {

    return new Promise(function(resolve, reject) {

        setTimeout(() => {

            (typeof value[0] === 'number' && typeof value[1] === 'number') ? resolve([value[0] + value[1], value[1], value[2] + 1]) : reject('Error');

        }, 2000);

    })

};

summ([4, 5, 0]).then(

    response => {

        console.log(response[0], response[2]);

        return summ([response[0], 8, response[2]])

    }

).then(

    response => {

        console.log(response[0], response[2]);

        return summ([response[0], 15,response[2]])

    }

).then(

    response => {

        console.log(response[0], response[2]);

        return summ([response[0], 2, response[2]])

    }

).then(

    response => {

        console.log(response[0], response[2]);

        return summ([response[0], 36, response[2]])

    }

).then(

    response => {

        console.log(response[0], response[2]);

        return summ([response[0], 4, response[2]])

    }

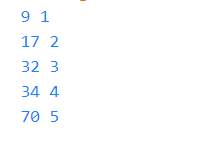
).catch(

    response => {

        console.log(response);

    }

)



let summ = function(value) {

    return new Promise(function(resolve, reject) {

        setTimeout(() => {

            (typeof value[0] === 'number' && typeof value[1] === 'number') ? resolve([value[0] + value[1], value[1], value[2] + 1]) : reject('Error');

        }, 2000);

    })

};

summ([4, '5', 0]).then(

    response => {

        console.log(response[0], response[2]);

        return summ([response[0], 8, response[2]])

    }

).then(

    response => {

        console.log(response[0], response[2]);

        return summ([response[0], 15,response[2]])

    }

).then(

    response => {

        console.log(response[0], response[2]);

        return summ([response[0], 2, response[2]])

    }

).then(

    response => {

        console.log(response[0], response[2]);

        return summ([response[0], 36, response[2]])

    }

).then(

    response => {

        console.log(response[0], response[2]);

        return summ([response[0], 4, response[2]])

    }

).catch(

    response => {

        console.log(response);

    }

)



**5 задание**

// Асинхронные функции

async function readConfig (name) {

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

        setTimeout(() => {

            resolve(console.log('(1) config from ' + name + ' loaded'))

        }, Math.floor(Math.random() \* 1000))

    })

}

async function doQuery (statement) {

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

        setTimeout(() => {

            resolve(console.log('(2) SQL query executed: ' + statement))

        }, Math.floor(Math.random() \* 1000))

    })

}

async function httpGet (url) {

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

        setTimeout(() => {

            resolve(console.log('(3) Page retrieved: ' + url))

        }, Math.floor(Math.random() \* 1000))

    })

}

async function readFile (path) {

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

        setTimeout(() => {

            resolve(console.log('(4) Readme file from ' + path + ' loaded'))

        }, Math.floor(Math.random() \* 1000))

    })

}

async function Await(){

    await true

    // Вызов функций

    console.log('start')

    await readConfig('myConfig');

    await doQuery('select \* from cities')

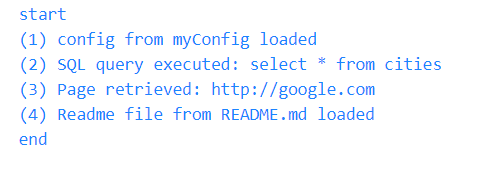
    await httpGet('http://google.com')

    await readFile('README.md')

    console.log('end')

}

Await()



const x = 3

async function f1() {

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

        setTimeout(() => {

            const NewRes = x \*\* 2;

            console.log(`Промежуточный результат ${NewRes}`);

            resolve(NewRes);

        }, 0)

    })

}

async function f2(OldRes) {

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

        setTimeout(() => {

            let NewRes = OldRes + x \* 2;

            console.log(`Pезультат n = 2: ${NewRes}`)

            resolve(NewRes)

        }, 0)

    })

}

async function f3(OldRes) {

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

        setTimeout(() => {

            let NewRes = OldRes - 2;

            console.log(`Промежуточный результат ${NewRes}`)

            resolve(NewRes)

        }, 1000)

    })

}

async function f4(OldRes) {

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

        setTimeout(() => {

            let NewRes = OldRes \*\* 3;

            console.log(`Pезультат при n = 4: ${NewRes}`)

            resolve(NewRes)

        }, 10000)

    })

}

async function f5(OldRes) {

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

        setTimeout(() => {

            let NewRes = OldRes \* 3;

            console.log(`Промежуточный результат ${NewRes}`)

            resolve(NewRes)

        }, 1000)

    })

}

async function f6(OldRes) {

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

        setTimeout(() => {

            let NewRes = OldRes - 3;

            console.log(`Pезультат при n = 6: ${NewRes}`)

            resolve(NewRes)

        }, 1000)

    })

}

async function Await(){

    await true

    // Вызов функций

    console.log('start')

    const readc = await f1();

    const NewRes = await f2(readc)

    const newRes2 = await f3(NewRes)

    const newRes3 = await f4(newRes2)

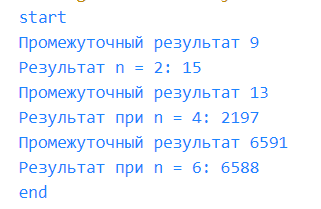
    const newRes4 = await f5(newRes3)

    await f6(newRes4)

    console.log('end')

}

Await()



const summ = async (value) => {

    let res = [...value];

    if (typeof res[0] !== 'number' || typeof res[1] !== 'number' || !Array.isArray(value) || value.length !== 3) {

        throw "Error";

    }

    res = [res[0] + res[1], res[1], res[2] + 1];

    return res;

};

let res = [4, 5, 0];

let count = 0;

const interval = setInterval(async () => {

    try {

        res = await summ(res);

        console.log(res[0], res[2]);

        count++;

        if (count === 5) {

            clearInterval(interval);

        }

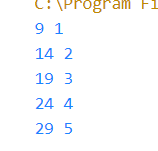
    } catch (error) {

        console.log(error);

        clearInterval(interval);

    }

}, 2000);



**6 задание**

async function wait() {

    await new Promise(resolve => setTimeout(resolve, 1000));

    return 10;

}

function f() {

    wait().then(console.log);

}

f();



**7 задание**

'use strict';

async function func(volume) {

    return new Promise(resolve => setTimeout(resolve, volume))

}

async function interviews()

{

    let number = 1;

    for (let i = 0; i < arguments.length; i++)

    {

        setTimeout(async()=>

        {

            console.log(arguments[i][0], "started the", number, "task");

            await func(arguments[i][1] \* 1000);

            console.log(arguments[i][0], "moved on to the defense of the", number, "task");

            await func(arguments[i][2] \* 1000);

            console.log(arguments[i][0], "completed the", number, "task");

            console.log(arguments[i][0], "is resting");

            await func(5000);

            number = 2;

            console.log(arguments[i][0], "started the", number, "task");

            await func(arguments[i][3] \* 1000);

            console.log(arguments[i][0], "moved on to the defense of the", number, "task");

            await func (arguments[i][4] \* 1000);

            console.log(arguments[i][0], "completed the", number, "task");

        });

    }

}

interviews(['Ivan', 5, 2, 7, 2], ['John', 3, 4, 5, 1], ['Sophia', 4, 2, 5, 1]);

