MÁSODIK ÓRA:

Háziként le lehet adni a zh megoldást is.

(ez alapján kell megcsinálni a zh-t is)

console.log('Negyedik feladat:')

function getParams2(value2){

let rplc = value2.replace('=', ':');

let rplc2=rplc.replace(' ',', ');

return rplc2;

}

function getParams(url) {

let spt = url.split('?')[1];

let value = spt.split('&');

var value3 = value.map(getParams2);

return value3.join();

}

console.log(getParams("http://example.com/path/to/page?firstname=john&lastname=doe&email=jdoe@example.com"));

const people = require('./people').people;

console.log('Első feladat:')

//Write a function that return the number of people that have starships

function HasStarShip(person){

return person.starships.length>0;

}

function cntHasStarShip() {

//return people.filter(person => person.starships.length>0).length;//egy szűrt tömböt ad vissza

return people.filter(HasStarShip).length;

}

console.log(cntHasStarShip());

console.log('Második feladat:');

//Add a vehiclesCount property to each object, the value of the property is number of items of the array of vehicles property

function addVehiclesCount() {

people.forEach(person =>

person.vehiclesCount = person.vehicles.length

);

}

addVehiclesCount();

console.log("Vehicles Cnt:");

console.log("--------------");

people.forEach(p=>console.log(p.name, p.vehiclesCount));

//Sort the people objects by the number of items in the array of films property

function greater(person1, person2){

return person1.films.length - person2.films.length;

}

function sortByFilmsCnt() {

people.sort(greater);

//people.sort((people1,people2)=>person1.films.length - person2.films.length)

}

console.log(' ');

console.log('Harmadik feladat:')

console.log(' ');

console.log("Sorted by films: ");

console.log("-----------------");

sortByFilmsCnt();

people.forEach(p=>console.log(p.name, p.films.length));

//Sort the people objects by the number of items in the starships array and within that by the number of items in the vehicles array

/\*function more(person111, person222){

return person111.starships.length - person222.starships.length;

}

function count(person11, person22){

return more(person11.vehiclesCount - person22.vehiclesCount);

}\*/

nction sortByStarsipsandVehicles() {

people.sort((person1, person2)=>{

let stord = person1.starships.length - person2.starships.length;

return stord == 0 ? personfu

1.vehiclesCount-person2.vehiclesCount:stord;

});

}

console.log(' ');

console.log('Negyedik feladat:')

console.log(' ');

console.log("Sorted by starShip and Vehicles: ");

console.log("-----------------");

sortByStarsipsandVehicles();

people.forEach(p=>console.log(p.name, p.starships.length, p.vehiclesCount))

//Count the total number of vehicles

function countVehicles(person) {

let temp=0;//mindig adni kell neki kezdő értéket, különben nem mefelelően íródik ki.

for(var i=0; i< person.length; i++){

temp+=person[i].vehiclesCount;

}

return temp;

}

console.log(' ');

console.log('Ötödik feladat:')

console.log(' ');

console.log("Total number of vehicles: ");

console.log(countVehicles(people));

CALLBACK

console.log(arguments);

setTimeout(()=>console.log('test'),500);

console.log('test 2'); //ha ezt futtajtuk, akkor is a teszt2 fog előbb kiiródni, abban az esetben is, ha az 500 helyett 0-t irunk; a főszáll az fut tovább

function readFileCallBack(err, data){//első mindig a hiba, második a file tartalma

if(err){

console.log(err);

}else{

console.log(data.length);

}

}

const fs= require('fs');

fs.readFile('./exercises.js', readFileCallBack);

console.log(fs.readFileSync('./exercises.js'));//a visszatérési értéke a puffer; érdemes igy használni, mert ez nem foglalja le a node-t

// a fenti esetben a callback függvény miatt, a 2. kiiratás fog lefutni később

//Mi van akkor, ha ahibát azelőtt megtaláljuk, mielőtt meghívnánk a readfile függvényt? a hiba hamarabb megjelenik

function read(path, callback){

//elsőként megnézzük, hogy átadott e valamit a path-nak

if(path==undefined){

//callback(new Error('Missing fileName'),null);//mivel itt nincs eredménye, mert megáll ezért lesz a 2. paraméter null

process.nextTick(()=> callback(new Error('Missing fileName'),null));//aszinkron mód fog futni, nem a callback kövertkező lépéseként kerül be, még mielőtt a callback viewból olvasna , azelőtt lesz belerakva.

}else{

fs.readFile(path, callback);

}

}

read('./exercises.js', readFileCallBack);

read(undefined, readFileCallBack);

//Csinálj egy függvényt, ami kiirja a konzolra az argumentumot amit kap.

function echoSync(text){

if(!text){

console.error('error sync');//ezzel tudunk kiirni hibát, böngészőbe priossal írja ki.

}

console.log(text);

}

//Aszinkron változat

//Callback függvény

function logCallback(error, text){

console.log(error, text);

}

function echo(text,callback){

// if(!text)

// callback('error', null); ez a rész nem jó.

setTimeout(()=>{

if(text){

callback (null, text);//aszinkron esetében érdemes a time outot használni, ennek már kell egy callback függvény is.

}else{

callback('error Async', null);

}

},0);

}

echo('Echo async', logCallback);

echoSync('Echo sync');

Ez FELADAT NEM JÓ, MEG KELL NÉZNI HOGY MIÉRT NEM

//Csinálj egy függvényt, ami kiirja a konzolra az argumentumot amit kap.

function echoSync(text){

if(!text){

console.error('error sync');//ezzel tudunk kiirni hibát, böngészőbe priossal írja ki.

}

console.log(text);

}

//Aszinkron változat

//Callback függvény

function logCallback(error, text){

console.log(error, text);

}

function echo(text,callback){

// if(!text)

// callback('error', null); ez a rész nem jó.

setTimeout(()=>{

if(text){

callback (null, text);//aszinkron esetében érdemes a time outot használni, ennek már kell egy callback függvény is.

}else{

callback('error Async', null);

}

},0);

}

echo('Echo async', logCallback);

echoSync('Echo sync');

//Szinkron

let strg='Ez itt egy szöveg.';

function upperSync(str){

console.log(str.toUpperCase());

}

upperSync(strg);

//Asszinkron

function callBack(error, text){

//console.log(error, text.toUpperCase());

console.log(error, text);

}

function upperCase(text){

return text.toUpperCase();

}

function upper(text, callBack){

setTimeout(()=>{

if(!text){

callBack('Hiányzó text', null);

}else{

callBack(null, text.toUpperCase());

}

},0);

}

upper(strg);

//create a module that exports a getOSData() method

//the method returns an object with the following properties:

// {

// platform: "win32",

// os: "Windows\_NT 10.0.17763",

// arch: "x64",

// cpus: 4,

// mem: {

// free: 2.2774009704589844,

// total: 11.869091033935547

// },

// network: {

// ethernet: "89.133.189.223",

// wireless: ""

// }

// }

//use the os module APIs to query computer data

//https://nodejs.org/api/os.html

const os = require('os');//modulra kell hivatkozni

function getOSData(){

return {

platform: os.platform(),

os: `${os.type()} ${os.release}`,

arch: os.arch(),

cpus: os.cpus().length,

mem:{

free: os.freemem() /1024 /1024 /1024,

total: os.totalmem() /1024 /1024 /1024,

},

network: {

ethernet: getIPv4(os.networkInterfaces()['Ethernet'])

}

};

}

function getIPv4(interface){

let address = interface.filter(int =>int.family == 'IPv4').map(int=>int.address);

if(address.length>0){

return address[0];

}else{

return'';

}

}

module.exports = {//object

//kell, hogy más modulok számára elérhető legyen.

//getOSData: getOSData //property ha a property neve és a függvény megegyezik akkor a 6-os szabványtól, nem kell ezzel foglalkozni elég ha úgy rjuk hogY:

getOSData

};

Házi feladat: objektek feldolgozásának gyakorlása: repoban: peopleapi.js!