//console.log(1);

//"use strict"; //команда для работы в современном режиме

//let number = 1;

//const leftBorderWidth = 1;

//number = 10;

//console.log(number);

//const obj = {

//пример как поменять const

// a: 50,

//};

//obj.a = 04;

//console.log(obj);

/\* const vehicleBodyWidth = 5000;

const vehicleBodyLenght = 4000;

console.log(

  "ширина кузова автомобиля: " +

    vehicleBodyWidth +

    ", длина" +

    vehicleBodyLenght

); \*/

//['fdsf', 'DFG' , 'FDF'].map(a =>)

//const userName = 'John';

//let userNumber = 25;

//number = 24;

//числа

/\* let number = 4.5;//без кавычек число

console.log(4/0); //Infinity неправильное действие

console.log('strict' \* 7); //NaN non an number не число

 \*/

//строка

//const persone = "Alex"; // все в кавычках

//булиновые

//const bool = true; //два значение true or falce

//console.log(something); //null

/\* let und;

console.log(und); \*/ //undefined пусто

/\* const obj = {

  name: "John",

  age: 25,

  isMarried: false

}; \*/

// обьект это хранение данных в парном форматеБ т.е. есть ключ и значение. Порядок не имеет значение

//console.log(obj.name);

//console.log(obj['name']); //редко

//let arr = ['plum.png', 'orange.ipg', 6, 'apple', {}]; //классический масив - частный случай обьекта - хранение данных строго по порядкуБ можем разные тип данных вносить - каждый тип данных строго по порядку 0,1,2,3 и т.д. В принципе тоже есть ключ но доступ к нему только через номер по порядку. Квадратный скобки это масив

//let arrObj = { // можна и так записать массив и тут можна создавать вутри обьекты

//0: 'plum.png',

//1: 'orange.ipg',

//2: 6,

//3: 'apple',

//4: {}

//abc: {

//df: [{}, {}],

//def: {

//    }

// }

//};

//arrObj.b = '1111';

//arrObj['b'] = '1111';

//console.log(arr[0]); //в программировании нумерация начинается с "0"

//console.log(arrObj[1]);

//console.log(arrObj[b]); // ошибка, думает что переменная - надо преобразовать в строку

//console.log(arrObj['b']);

//console.log(arrObj.b);

//Взаимодействие с пользователем

//alert("Hello");

//const result = confirm("Are you here?");

//console.log(result);

//const answer = +prompt("Are you 18?", "18");

//console.log(typeof answer); //typeof показывает какой тип вывода данных строка или

//console.log(answer + 5);

//const ansvers = [];

//ansvers[0] = prompt("What is you name?", "");

//ansvers[1] = prompt("What is you last name?", "");

//ansvers[2] = prompt("how much you old?", "");

//document.write(ansvers); устаревший вариант

//console.log(typeof ansvers);

//Интерполяция - внутри строки можем вставить значение переменной

//const category = "toys";

//console.log('https://someurl.com/' + category + '/' + '4'); // неудобно и был придуман Интерполяция

//console.log(`https://someurl.com/${category}/5`); // обезательно использование косых кавычек ``

//const user = "Dima";

//alert(`Hello, ${user}`);

//Операторы (+ - \* / % \*\*)

//console.log("arr" + " - object");

//console.log(4 + " - object");

//console.log(4 + +" - object"); //NaN second + - унарный

//console.log(4 + +"5");

//let incr = 10,

//decr = 10;

//incr++; //оператор инкримента, т.е. увелечение на 1

//decr--; //операт декремента, т.е. уменшение на 1

//++incr;

//--decr;

//console.log(incr++); //так не работает

//console.log(--decr); //так  работает

//console.log(5%2); //% значит что 5 делится на 5 2 раза и получаем 1

//console.log(2\*2 == 4); // == сравнение и неважно просто число или ""

//console.log(2\*2 === "4"); // === строгое сравнение и если "" то уже это неправда

//console.log(2\*2 + 2 === "4");

//console.log(2\*2 + 2 != "4"); // != знак

// && - and - работает когда минимум два варианта правдивы

// || - or - работает когда один из вариантов правда

// ! - оператор отрицание меняет значение на противоположное

//const isCheked = true,

//      isClose = false;

//console.log(isCheked && isClose);

//console.log(!isCheked || isClose);

/\* \* Задание на урок:

1) Создать переменную numberOfFilms и в неё поместить ответ от пользователя на вопрос:

'Сколько фильмов вы уже посмотрели?'

2) Создать объект personalMovieDB и в него поместить такие свойства:

    - count - сюда передается ответ на первый вопрос

    - movies - в это свойство поместить пустой объект

    - actors - тоже поместить пустой объект

    - genres - сюда поместить пустой массив

    - privat - в это свойство поместить boolean(логическое) значение false

3) Задайте пользователю по два раза вопросы:

    - 'Один из последних просмотренных фильмов?'

    - 'На сколько оцените его?'

Ответы стоит поместить в отдельные переменные

Записать ответы в объект movies в формате:

    movies: {

        'logan': '8.1'

    }

Проверить, чтобы все работало без ошибок в консоли \*/

/\* const numberOfFilms = +prompt("Сколько фильмов вы уже посмотрели?", "");

const personalMovieDB = {

  count: numberOfFilms,

  movies: {},

  tactors: {},

  genres: [],

  privat: false,

};

const a = prompt("Один из последних просмотренных фильмов?", ""),

  b = prompt("На сколько оцените его?", ""),

  c = prompt("Любимый актер?", ""),

  d = prompt("Любимая актриса?", "");

personalMovieDB.movies[a] = b;

personalMovieDB.actors[c] = d;

console.log(personalMovieDB); \*/

// условия if - "если" работает как на числа так и на строки

/\* if (4 == 9) {

  console.log("Ok!");

} else {

  console.log("Error");

}

const num = 50;

if (num < 49) {

  console.log("Error!");

} else if (num > 100) {

  console.log("Много");

} else {

  console.log("Ok!");

}

num === 50 ? console.log("Ok!") : console.log("Error!"); \*/

/\* тернарный оператор - в его работе участвует три аргумента

бинарный - 2 аргумента

унарный - 1 \*/

/\* const num = 50;

switch (num) {

  case 49:

    console.log("Error!");

    break;

  case 100:

    console.log("Error!");

    break;

  case 51:

    console.log("Ok");

    break;

  default: //если дейсвие не выполняется

    console.log("Next");

    break;

} \*/

//Логические операторы - "и", "или", оператор отрецания "не"

//пример - я наемся если в меню будет и гамбургер и картошка фри

/\* const hamburger = true;

const fries = false;

if (hamburger && fries) {

  console.log('Я сыт');

}

console.log((hamburger && fries)); \*/ // результат логического пути булитовое значение true

/\* const hamburger = 3;

const fries = 1;

const cola = 0;

console.log(hamburger === 3 && cola && fries);

//правда идет от 1 ("0" это уже не правда) и если возле значения нет сравнения то считается от правда от "1"

//неправда это 5 сущностей - 0, null, NaN, inderferno,

console.log(1 && 0);

console.log(1 && 5);

console.log(null && 5);

console.log(0 && 'ggggggggjghdj');

//оператор 'i' ищет значение и если находит неправильное то возвращает его обратно и дальше код не идет...cola.есди все правильные то возвращает последнее

if (hamburger === 3 && cola === 1 && fries === 1) {

  console.log('Все сыты');

} else {

  console.log('Мы уходим');

}

console.log((hamburger && fries)); \*/

/\* const hamburger = 3;

const fries = 3;

const cola = 0;

const nuggets = 2;

if ((hamburger === 3 && cola === 2) || (fries === 3 && nuggets)) {

  console.log("Все сыты");

} else {

  console.log("Мы уходим");

}

console.log((hamburger === 3 && cola === 2) || (fries === 3 && nuggets));

let johnReport,

  alexReport,

  samReport,

  mariaReport = "done";

console.log(johnReport || alexReport || samReport || mariaReport);

console.log(!0); // неправда возвращает \*/

//Циклы

//let num = 50;

//1) while - пока что какоето условие выполняется мы будем выполнять какоето действие

//while (num <= 55) {

//console.log(num);

//num++;

//}

//2) застивить чтото сделать а потом проверить условиеБ если необходимо выйти из цикла

/\* do {

  console.log(num);

  num++;

} while (num < 55);

for (let i = 1; i < 8; i++) {

  console.log(i);

} \*/

/\* for (let i = 1; i < 8; i++) {

  console.log(num);

  num++;

} \*/

/\* for (let i = 1; i < 10; i++) {

  if (i === 6) {

    //break;прерываем цикл

    continue; //пропускаем значение с "если"и дальше продолжаем цикл

  }

  console.log(i);

} \*/

//Метки в цикле

/\* for (let i = 0; i < 3; i++) {

  console.log(i);

  for (let j = 0; j < 3; j++) {

    console.log(j);

  }

} \*/

//\*

//\*\*

//\*\*\*

//\*\*\*\*

//\*\*\*\*\*

//\*\*\*\*\*\*

/\* let result = "";

const lenght = 7;

for (let i = 1; i < lenght; i++) {

  for (let j = 0; j < i; j++) {

    result += "\*";

  }

  result += "\n"; //перенос строки//

}

console.log(result);

first: for (let i = 0; i < 3; i++) {

  console.log(`First level: ${i}`);

  for (let j = 0; j < 3; j++) {

    console.log(`Second level: ${j}`);

    for (let k = 0; k < 3; k++) {

      if (k === 2) continue first;

      console.log(`Third level: ${k}`);

    }

  }

}

first: for (let i = 0; i < 3; i++) {

  console.log(`First level: ${i}`);

  for (let j = 0; j < 3; j++) {

    console.log(`Second level: ${j}`);

    for (let k = 0; k < 3; k++) {

      if (k === 2) break first;

      console.log(`Third level: ${k}`);

    }

  }

}

 \*/

/\*

function firstTask() {

  for (let i = 20; i >= 10; i--) {

    if (i === 13) break;

    console.log(i);

  }

}

for (let i = 5; i <= 10; i++) {

  console.log(i);

}

for (let i = 2; i <= 10; i++) {

  if (i % 2 === 0) {

    console.log(i);

  }

}

let i = 2;

while (i <= 16) {

  if (i % 2 === 0) {

    i++;

    continue;

  } else {

    console.log(i);

  }

  i++;

}

const arrayOfNumbers = [];

for (let i = 5; i < 11; i++) {

  arrayOfNumbers[i - 5] = i;

}

console.log(arrayOfNumbers);

return arrayOfNumbers;

const arr = [3, 5, 8, 16, 20, 23, 50];

const result = [];

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

  result[i] = arr[i];

}

console.log(result);

return result;

const data = [5, 10, "Shopping", 20, "Homework"];

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

  if (typeof data[i] === "number") {

    data[i] = data[i] \* 2;

  } else if (typeof data[i] === "string") {

    data[i] = `${data[i]} - done`;

  }

}

console.log(data);

return data;

const data = [5, 10, "Shopping", 20, "Homework"];

const result = [];

for (let i = 1; i < data.length; i++) {

  result[i - 1] = data[data.length - i];

}

console.log(result);

return result;

const lines = 5;

let result = "";

for (let i = 0; i < lines; i++) {

  for (let j = 0; j < lines - i; j++) {

    result += " ";

  }

  for (let j = 0; j < 2 \* i + 1; j++) {

    result += "\*";

  }

  result += "\n";

}

console.log(result); \*/

/\* Задание на урок:

1) Автоматизировать вопросы пользователю про фильмы при помощи цикла

2) Сделать так, чтобы пользователь не мог оставить ответ в виде пустой строки,

отменить ответ или ввести название фильма длинее, чем 50 символов. Если это происходит -

возвращаем пользователя к вопросам опять. (К любой строке можно обратиться как

str.length - и получить её длину)

3) При помощи условий проверить  personalMovieDB.count, и если он меньше 10 - вывести сообщение

"Просмотрено довольно мало фильмов", если от 10 до 30 - "Вы классический зритель", а если больше -

"Вы киноман". А если не подошло ни к одному варианту - "Произошла ошибка"

4) Потренироваться и переписать цикл еще двумя способами\*/

/\* ("use strict");

const numberOfFilms = +prompt("Сколько фильмов вы уже посмотрели?", "");

const personalMovieDB = {

  count: numberOfFilms,

  movies: {},

  tactors: {},

  genres: [],

  privat: false,

};

// цикл начинается с нуля. в данном случае необходимо что бы он сработал 2 раза. начинается с нуля, потом плюс 1 и на 2 закачивается

for (let i = 0; i < 2; i++) {

  const a = prompt("Один из последних просмотренных фильмов?", ""),

    b = prompt("На сколько оцените его?", "");

  //null появляется если нажимаем отмена и в данном случае проверяем значение на наличие этих отмен, "" пустая строка на нее тоже проверяем, lenght (длина строки) не более чем 50 символов -> и если все хорошо то выполняется действие. если это не выполнилось задаем итератор i с двумя минусами и начинаем сначала. и так происходит пока пользователь не внесет правильные данные

  if (a != null && b != null && a != "" && b != "" && a.length < 50) {

    personalMovieDB.movies[a] = b;

    console.log("done");

  } else {

    console.log("error");

    i--;

  }

}

if (personalMovieDB.count < 10) {

  console.log("Просмотрено довольно мало фильмов");

} else if (personalMovieDB.count >= 10 && personalMovieDB.count < 30) {

  console.log("Вы классический зритель");

} else if (personalMovieDB >= 30) {

  console.log("Вы киноман");

} else {

  console.log("Произошла ошибка");

}

console.log(personalMovieDB); \*/

//("use strict");

// функсии - ()-для аргументовб внутри фигрных скобок действия, !!! что бы функция заработала нужно  вызвать к имени и обезательно круглые скобки+;

//главное в функции ее имя - как правило это глагол с припиской над чем выполняется действие.

//в скобках задается аргумент который мы задаем потом в дейсвтии, а отдельно им даем значение аргументов можно задавать много

//если мы обьявляем внутри функции переменную снаружи она не доступна, а внутри сней можно работать

// переменную нужно задавать глобально вне функции, а менять можно только без использования let

//замыкание функции - это поиск функцией переменных на уровень выше до самого верха - со всеми внешними переменными какие ей доступны

// return (ret) - останавливает выполнение функции, после него ничего не задается

/\* let num = 20;

function showFirstMessage(text) {

  console.log(text);

  let num = 10;

  console.log(num);

}

showFirstMessage("Hello world!");

console.log(num);

//

function calc(a, b) {

   return (a + b);

}

console.log(calc(4, 3));

console.log(calc(5, 10));

console.log(calc(12, 15));

//

function ret() {

  let num = 50;

  return num;

}

const anotherNumm = ret();

console.log(anotherNumm);

 \*/

//три вида функций

//1 как обычно используем Function Declaration  создается до начала выполнения скрипта можна вызвать перед выполнением

// function ret() {

//код

//}

//2 Function Expression - создается только тогда когда доходит поток кода , можно вызвать только после обьявлени и выводится через переменную

/\* const logger = function() {

  console.log('Hello')

};

logger(); \*/

//3 Стрелочные функции - не имеют своего контекста

// ()=>

/\* const calc = (a, b) => {

  return a+b;

} \*/

//умение писать код по принципе оптимальности и минамально доступным способом - пример конвертации валют

/\* const usdCurr = 28;

const evrCurr = 32;

function convert(amount, curr) {

  console.log(curr \* amount);

}

convert(500, usdCurr);

convert(500, evrCurr);

 \*/

// любая функция может вернет чтото как результат своей работы

//return

// конвертация валют и следующее оформлени скидки

/\* const usdCurr = 32;

const discount = 0.9;

function convert(amount, curr) {

  return curr \* amount;

}

function promotion(result) {

  console.log(result \* discount);

}

const res = convert(500, usdCurr);

promotion(res);

// 2 вариант - promotion (convert(500, usdCurr)); // производится вызов функции и возвращает результат (число), это число в качестве уже аргумента идет в др.функцию и производится вычесление

// return - по правило после него фольше ничего не выполняется и нельзя переносить строку,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,

function test() {

  for (let i = 0; i < 5; i++) {

    console.log(i);

    if (i ===3) {return}

  }

  console.log('Done');

}

test();

//функция даже пустая всегда чтото возвращает как пример - выдает значение и потом underfined - на примере доказательство что пустая функция вернет:

function doNothing() {};

console.log(doNothing () === undefined); \*/

/\* function sayHello(name) {

  return `Привет, ${name}`;

}

sayHello("Aнтон");

function returnNeighboringNumbers(num) {

  return [num - 1, num, num + 1];

}

returnNeighboringNumbers(5);

function getMathResult(num, times) {

  if (typeof times !== "number" || times <= 0) {

    return num;

  }

  let str = "";

  for (let i = 1; i <= times; i++) {

    if (i === times) {

      str += `${num \* i}`;

    } else {

      str +=`${num\*1}---`;

    }

  }

  return str;

}

getMathResult (10, 5); \*/

// числа и строки

/\* const str = 'test';

const arr = [1, 2, 3];

//console.log(str.toUpperCase());

console.log(str.toLocaleLowerCase());

console.log(str);

//console.log(arr.length);

//console.log(str.length);

//поиск подстроки

const fruit = 'Some fruit!';

//console.log(fruit.indexOf('fruit'));

console.log(fruit.indexOf('a'));

//обрезание символов

const logg = 'Hello world!';

//console.log(logg.slice(5, 11));

//console.log(logg.substring(5, 11)); //то же самое что и 1 вариант только нельзя использовать минусовые значение

console.log(logg.substr(6, 5)); // сколько символов необходимо вырезать

// числа через Math

const num = 12.2;

console.log(Math.round(num));

const test = "12.2px";

console.log(parseInt(test)); //преобразование строки в число

console.log(parseFloat(test));

 \*/

/\* Задание на урок:

1) Первую часть задания повторить по уроку

2) Создать функцию showMyDB, которая будет проверять свойство privat. Если стоит в позиции

false - выводит в консоль главный объект программы

3) Создать функцию writeYourGenres в которой пользователь будет 3 раза отвечать на вопрос

"Ваш любимый жанр под номером ${номер по порядку}". Каждый ответ записывается в массив данных

genres

P.S. Функции вызывать не обязательно \*/

/\* let numberOfFilms;

function start() {

  numberOfFilms = +prompt("Сколько фильмов вы уже посмотрели?", "");

  while (numberOfFilms == "" || numberOfFilms == null || isNaN(numberOfFilms)) {

    numberOfFilms = +prompt("Сколько фильмов вы уже посмотрели?", "");

  }

}

start();

const personalMovieDB = {

  count: numberOfFilms,

  movies: {},

  actors: {},

  genres: [],

  privat: false,

};

function rememberMyFilms() {

  for (let i = 0; i < 2; i++) {

    const a = prompt("Один из последних просмотренных фильмов?", "").trim(), // добавление этого метода позволяет обрезать пробелы

      b = prompt("На сколько оцените его?", "");

    if (a != null && b != null && a != "" && b != "" && a.length < 50) {

      personalMovieDB.movies[a] = b;

      console.log("done");

    } else {

      console.log("error");

      i--;

    }

  }

}

rememberMyFilms();

function detectPersonalLevel() {

  if (personalMovieDB.count < 10) {

    console.log("Просмотрено довольно мало фильмов");

  } else if (personalMovieDB.count >= 10 && personalMovieDB.count < 30) {

    console.log("Вы классический зритель");

  } else if (personalMovieDB.count >= 30) {

    console.log("Вы киноман");

  } else {

    console.log("Произошла ошибка");

  }

}

function showMyDB(hidden) {

  if (!hidden) {

    console.log(personalMovieDB);

  }

}

showMyDB(personalMovieDB.privat);

function writeYourGenres() {

  for (let i = 1; i <= 3; i++) {

    personalMovieDB.genres[i - 1] = prompt(`Ваш любимый жанр под номером ${i}`);

  }

}

writeYourGenres(); \*/

//вычисление площади и обьема

/\* function calculateVolumeAndArea(lenght) {

  if (typeof lenght !== "number" || lenght < 0 || !Number.isInteger(lenght)) {

    return "При вычислении произошла ошибка";

  }

  let volume = 0,

    area = 0;

  volume = lenght \* lenght \* lenght;

  // length \*\* 3 - это тоже самое, что и выше или варианты через цикл.

  // \*\* - это оператор степени, напоминаю. Но онлайн редактор его не принимает

  area = 6 \* (lenght \* lenght);

  return `Объем куба: ${volume}, площадь всей поверхности: ${area}`;

}

calculateVolumeAndArea(5);

//функция, которая будет опеределять номера купе по преданому ей номеру места

function getCoupeNumber(seatNumber) {

  if (typeof(seatNumber) !== 'number' || seatNumber < 0 || !Number.isInteger(seatNumber)) {

    return "Ошибка. Проверьте правильность введенного номера места";

  }

  if (seatNumber === 0 || seatNumber > 36) {

    return "Таких мест в вагоне не существует";

  }

   for (let i = 4; i<=36; i = i + 4) {

    if (seatNumber <= i) {

      return Math.ceil(i / 4);

    }

   }

}

getCoupeNumber(33);

// функция принимает в себя целое число минут и возвращает время в нужном формате строки

function getTimeFromMinutes(minutesTotal) {

  if (typeof(minutesTotal) !== 'number' || minutesTotal < 0 || !Number.isInteger(minutesTotal)) {

    return "Ошибка, проверьте данные"

  }

  const hours = Math.floor(minutesTotal / 60);

  const minutes = minutesTotal % 60;

  let hoursStr = '';

  switch (hours) {

    case 0:

      hoursStr = 'часов';

      break;

    case 1:

      hoursStr = 'час';

      break;

    case 2:

    case 3:

    case 4:

      hoursStr = 'часа';

      break;

    default:

      hoursStr = 'часов';

  }

  return `Это ${hours} ${hoursStr} ${minutes} минут`;

}

getTimeFromMinutes(250);

// функция которая принмает в себя 4 числа и возвращает большое из них

function findMaxNumber(a,b,c,d) {

  // Самое простое - это использовать Math.max :)

    // А оптимизировать такую проверку мы научимся совсем скоро

  if (typeof(a) !== 'number'||

      typeof(b) !== 'number'||

      typeof(c) !== 'number'||

      typeof(d) !== 'number') {

        return 0;

      } else {

        return Math.max(a,b,c,d);

      }

}

findMaxNumber(2, 10, 5.7, 3);

findMaxNumber(1, 7, '6', '11');

function fib(num) {

    if (typeof(num) !== 'number' || num <= 0 || !Number.isInteger(num)) {

      return "";

    }

    let result = 0;

    let first = 0;

    let second = 1;

    for (let i = 0; i < num; i++) {

      if (i + 1 === num) {

        result += `${first}`;

      } else {

        result += `${first}`;

      }

    let third = first + second;

    first = second;

    second = third;

    }

    return result;

}

fib(5); \*/

// callback  - указываем какая функция должна выполняться после того как дpугая выполнилась

/\* function first() {

  // Do something

  setTimeout(function () {

    console.log(1);

  }, 500);

}

function second() {

  console.log(2);

}

first();

second();

//

function learnJS(lang, callback) {

  console.log(`я учу: ${lang}`);

  callback();

}

function done() {

  console.log('я прошел этот урок');

}

learnJS('JavaScript', done); \*/

// обьектыБ дестуктуризация

//const obj = new Object();

/\* const options = {

  name: "test",

  width: 1024,

  height: 1024,

  colors: {

    border: "black",

    bg: "red",

  },

  //можем создать метод который будет работать внутри

  makeTest: function () {

    console.log("Test");

  }

};

options.makeTest (); // обезательно надо запускать

// деструктуризация - это возможность работать с каждым свойствой отдельно

const {border, bg} = options.colors;

console.log(border); \*/

//console.log(Object.keys(options).length); // получаем масив с всеми ключами (name, w< h< colors)

//console.log(options.name);

//delete options.name; //способ удаление (исключение) значения

//console.log(options);

//для переборки функций в обьекти используем -> for (let ..in...) {} если внутри ести еще обьекты то запускаем исключение через typeof

/\* let counter = 0; // для пересчета свойств в обьекте

for (let key in options) {

  if (typeof (options[key]) === 'object') {

    for (let i in options[key]) {

      console.log(`Свойство ${i} имеет значение ${options[key][i]}`);

      counter++;

    }

  }  else {

    console.log(`Свойство ${key} имеет значение ${options[key]}`);

    counter++;

  }

}

console.log(counter); \*/

//масивы и псевдомасивы - методы и свойства/ псевдомасивы похожи на масивы, только к ним не применяются методы и они отбражают элементы строго по порядку

//length - количество элементов которые находятся внутри

/\* const arr = [1, 21, 13, 26, 8];

arr.sort(compareNum);

console.log(arr);

function compareNum(a, b) {

  return a - b;

} \*/

//arr[99] = 0;

//console.log(arr.length);

//console.log(arr);

// arr.pop(); // метод удаления последнего значения

//arr.push(10); // метод добавления к последнему значения

//arr.forEach() // позволяет гибко перебрать все элементы

//arr.forEach(function (item, i, arr) {

//  console.log(`${i}: ${item} внутри масива ${arr}`);

//});

//console.log(arr);

// перебор элементов внутри масива

// 1

//for (let i = 0; i < arr.length; i++) {

//  console.log(arr[i]);

//}

//2 в методе for (let ... of) можно использовать breake (остановка) или contine (пропуск)

//for (let value of arr) {

//  console.log(value);

//}

// МЕТОДЫ split превращает строку в масив // join превращает масив в строку// sort - сортировка массива как строки, если в масиве числа то пописуем функцию.

/\* const str = prompt("", "");

const product = str.split(", ");

product.sort();

console.log(product.join("; "));

const arr = [1, 21, 13, 26, 8];

arr.sort(compareNum);

console.log(arr);

function compareNum(a, b) {

  return a - b;

} \*/

// передача данных по ссылке или по значению / есть поверхностное (простые значение) или глубокое клонирование

/\* let a = 5,

    b = a;

b = b + 5;

 console.log(b);

 console.log(a);

//

 const obj = {

  a: 5,

  b: 1

};

const copy = obj; // передается не значение а ссылка на предыдущую структуру

copy.a = 10;

console.log(copy);

console.log(obj); \*/

// клонирование обьекта

/\* function copy(mainObj) {

  let objCopy = {};

  let key;

  for (key in mainObj) {

    objCopy[key] = mainObj[key];

  }

  return objCopy;

}

const numbers = {

  a: 2,

  b: 5,

  c: {

    x: 7,

    y: 4,

  }

};

const newNumbers = copy(numbers);

newNumbers.a = 10;

newNumbers.c.x = 10;

//onsole.log(newNumbers);

//console.log(numbers);

//добавление новых значений в обьект

const add = {

  d: 17,

  e: 25

};

//console.log(Object.assign(numbers), add);

// создаем пустой обьект и клонируем

const clone = Object.assign({}, add);

clone.d = 20;

//console.log(add);

//console.log(clone);

//клонирование масива

const oldArrey = ['a', 'b', 'c'];

const newArrey = oldArrey.slice();

newArrey[1] = 'ddddddd';

console.log(newArrey);

console.log(oldArrey);

// оператор разворота (...)

const video = ['youtube', 'vimeo', 'hdrezka'],

      blogs = ['wordpress', 'livejournal', 'blogger'],

      internet = [...video, ...blogs, 'instagram', 'fb'];

console.log(internet);

//

function log(a, b, c) {

  console.log(a);

  console.log(b);

  console.log(c);

}

const num = [2, 4, 7];

log(...num);

// спред оператор для разворота обьектов

const arrey = ["a", "b"];

const newAarrey = [...arrey];

//

const g = {

  one: 1,

  two: 2

};

const newObj = {...g}; \*/

/\* const personalPlanPeter = {

  name: "Peter",

  age: "29",

  skills: {

    languages: ["ru", "eng"],

    programmingLangs: {

      js: "20%",

      php: "10%",

    },

    exp: "1 month",

  },

  showAgeLangs: function (plan) {

    const { age } = plan;

    const { languages } = plan.skills;

    let str = `Мне ${age} и я владею языками:`;

    languages.forEach(function (lang) {

      str += `${lang.toUpperCase()}`;

    });

    return str;

  }

};

personalPlanPeter.showAgeLangs(personalPlanPeter);

function showExperience(plan) {

  const { exp } = plan.skills;

  return exp;

}

showExperience(personalPlanPeter);

function showProgrammingLangs(plan) {

  let str = '';

  const {programmingLangs} = plan.skills;

  for (let key in programmingLangs) {

    str += `Язык ${key} изучен на ${programmingLangs[key]}\n`;

  }

  return str;

}

showProgrammingLangs(personalPlanPeter);

//

const family = ['Peter', 'Ann', 'Alex', 'Linda'];

function showFamily(arr) {

    let str = '';

    arr.length === 0 ? str = 'Семья пуста' : str = 'Семья состоит из: ';

    arr.forEach(member => {

      str += `${member}`;

    });

    return str;

}

showFamily(family);

// нижний регистр

const favoriteCities = ['liSBon', 'ROME', 'miLan', 'Dublin'];

function standardizeStrings(arr) {

    arr.forEach(city => {

      console.log(city.toLowerCase());

    });

}

standardizeStrings(favoriteCities);

// вернуть предложние в обратном порядке букв

const someString = 'This is some strange string';

function reverse(str) {

    if(typeof(str) !== 'string') {

    return "Ошибка!";

  }

  // применяем методы split (Разбить строку на подстроки, используя указанный разделитель, и вернуть их в виде массива.)reverse(переворачивает элементы массива на место. Этот метод изменяет массив и возвращает ссылку на тот же массив.)join (Добавляет все элементы массива в строку, разделенную указанной строкой-разделителем.)

  return str.split('').reverse('').join('');

}

reverse(someString);

// банкомат с двумя валютами - функция раюоты с валютами

const baseCurrencies = ['USD', 'EUR'];

const additionalCurrencies = ['UAH', 'RUB', 'CNY'];

function availableCurr(arr, missingCurr) {

  let str = '';

  arr.length === 0 ? str = 'Нет доступных валют' : str = 'Доступные валюты:\n';

  arr.forEach(function (curr, i) {

    if (curr !== missingCurr) {

      str += `${curr}\n`;

    }

  });

  return str;

}

availableCurr([...baseCurrencies, ...additionalCurrencies], 'CNY'); \*/

// обьектно-ориентиованное програмирование / прототипно ориентированное програмирование

/\* let str = "some";

let strObj = new String(str);

//строка превращается в обьект

//console.log(typeof(str));

//console.log(typeof(strObj));

console.log([1, 2, 3]);

//prototype некоторые значение могут брать свойства и методы из своего прототипа

const soldier = {

  health: 400,

  armor: 100,

  sayHello: function () {

    console.log('Hello');

  }

};

const jonh = Object.create(soldier); // мы создаем новый обьект который прототип от солдата

//const john = {

  //health: 100/

//};

//john.\_\_proto\_\_ = soldier; устаревший метод

Object.setPrototypeOf(john, soldier); //установили прототип от джона для солдат

//console.log(john.armor);

john.sayHello();

 \*/

/\* Задание на урок:

1) У нас уже есть рабочее приложение, состоящее из отдельных функций. Представьте, что

перед вами стоит задача переписать его так, чтобы все функции стали методами объекта personalMovieDB

Такое случается в реальных продуктах при смене технологий или подхода к архитектуре программы

2) Создать метод toggleVisibleMyDB, который при вызове будет проверять свойство privat. Если оно false - он

переключает его в true, если true - переключает в false. Протестировать вместе с showMyDB.

3) В методе writeYourGenres запретить пользователю нажать кнопку "отмена" или оставлять пустую строку.

Если он это сделал - возвращать его к этому же вопросу. После того, как все жанры введены -

при помощи метода forEach вывести в консоль сообщения в таком виде:

"Любимый жанр #(номер по порядку, начиная с 1) - это (название из массива)"\*/

("use strict");

/\* const personalMovieDB = {

  count: 0,

  movies: {},

  actors: {},

  genres: [],

  privat: false,

  start: function () {

    personalMovieDB.count = +prompt("Сколько фильмов вы уже посмотрели?", "");

    while (

      personalMovieDB.count == "" ||

      personalMovieDB.count == null ||

      isNaN(personalMovieDB.count)

    ) {

      personalMovieDB.count = +prompt("Сколько фильмов вы уже посмотрели?", "");

    }

  },

  rememberMyFilms: function () {

    for (let i = 0; i < 2; i++) {

      const a = prompt("Один из последних просмотренных фильмов?", "").trim(), // добавление этого метода позволяет обрезать пробелы

        b = prompt("На сколько оцените его?", "");

      if (a != null && b != null && a != "" && b != "" && a.length < 50) {

        personalMovieDB.movies[a] = b;

        console.log("done");

      } else {

        console.log("error");

        i--;

      }

    }

  },

  detectPersonalLevel: function () {

    if (personalMovieDB.count < 10) {

      console.log("Просмотрено довольно мало фильмов");

    } else if (personalMovieDB.count >= 10 && personalMovieDB.count < 30) {

      console.log("Вы классический зритель");

    } else if (personalMovieDB.count >= 30) {

      console.log("Вы киноман");

    } else {

      console.log("Произошла ошибка");

    }

  },

  showMyDB: function (hidden) {

    if (!hidden) {

      console.log(personalMovieDB);

    }

  },

  toggleVisibleMyDB: function () {

    if (personalMovieDB.privat) {

      personalMovieDB.privat = false;

    } else {

      personalMovieDB.privat = true;

    }

  },

  writeYourGenres: function () {

    //for (let i = 1; i <= 3; i++) {

    //let genre = prompt(`Ваш любимый жанр под номером ${i}`);

    //if (genre === "" || genre == null) {

    //  console.log("Вы ввели некорруктные данные или не ввели их вообще");

    //  i--;

    //} else {

    //  personalMovieDB.genres[i - 1] = genre;

    //}

    for (let i = 1; i < 2; i++) {

      let genres = prompt(`Введити ваши любимые жанры через запятую`).toLowerCase(); //toLowerCase переделывает все в нижний регистр

      if (genres === "" || genres == null) {

        console.log("Вы ввели некорруктные данные или не ввели их вообще");

        i--;

      } else {

        personalMovieDB.genres = genres.split(", "); //split Разбить строку на подстроки, используя указанный разделитель, и вернуть их в виде массива.

        personalMovieDB.genres.sort(); //sort сортирует по алвавиту

      }

    }

    personalMovieDB.genres.forEach((item, i) => {

      console.log(`Любимый жанр ${i + 1} - это ${item}`);

    });

  }

};

 \*/

//

// есть магазин - посчитать бюджет по отоплению согласно площадей

/\* const shoppingMallData = {

  shops: [

      {

          width: 10,

          length: 5

      },

      {

          width: 15,

          length: 7

      },

      {

          width: 20,

          length: 5

      },

      {

          width: 8,

          length: 10

      }

  ],

  height: 5,

  moneyPer1m3: 30,

  budget: 50000

}

function isBudgetEnough(data) {

  let squire = 0;

  let volume = 0;

  data.shops.forEach(shop => {

    squire = shop.width \* shop.length;

  });

  volume = data.height \* squire;

  if (data.budget - (volume\* data.moneyPer1m3) >=0) {

    return 'Бюджета достаточно';

  } else {

    return 'Бюджета недостаточно';

  }

}

isBudgetEnough(shoppingMallData);

// распределяем игроков по алфавиту и группам и если ктото + или + добавляем в 4 строку

const students = ['Peter', 'Andrew', 'Ann', 'Mark', 'Josh', 'Sandra', 'Cris', 'Bernard', 'Takesi', 'Sam'];

function sortStudentsByGroups(arr) {

  arr.sort();

  const a = [], b = [], c = [], rest = [];

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

      if (i < 3) {

          a.push(arr[i]);

      } else if (i < 6) {

          b.push(arr[i]);

      } else if (i < 9) {

          c.push(arr[i]);

      } else {

          rest.push(arr[i]);

      }

  }

  return [a,b,c, `Оставшиеся студенты: ${rest.length === 0 ? '-' : rest.join(', ')}`]

}

sortStudentsByGroups(students); \*/

//динамическая тиризация - это когда число може стать строкойЮ строка чилслом а обьект булиновым значением

// to String

//1) редко

/\* console.log(typeof(String(null)));

console.log(typeof(String(null)));

//2) контонтинация - это сложение строки с чем-то получаем строку

console.log(typeof(5 + ''));

//позволяет перейти в указанный каталог

const num = 5;

console.log("https//facebook/catalog/" + num);

//задаем стили

const fontSize = 26 + 'px';

// to Number

//1) редко

console.log(typeof(Number('fgdg')));

//2) унарный плюс - до строки добавляем +

console.log(typeof(+'5'));

//3) parseInt

console.log(typeof(parseInt("15px", 10)));

// Present

let answer = +prompt("Ytllu", "")

//to Boolean

//0, '', null, undefined, NaN = false

// все остальное true

//1)

let switcher = null;

if(switcher) {

  console.log('Working....');

}

switcher = 2;

if(switcher) {

  console.log('Working....');

}

//2) редко

console.log(typeof(Boolean('4')));

//3) редко

console.log(typeof(!!"4444"));  \*/

//замыкание

/\* let number = 5;

debugger;

function logNumber() {

  let number = 4;

  debugger;

  console.log(number);

}

number = 6;

logNumber();

debugger; \*/

// обьяснение замыкания

/\* function createCounter() {

  let counter = 0;

  const myFunction = function () {

    counter = counter + 1;

    return counter;

  };

  return myFunction;

}

const increment = createCounter();

const c1 = increment();

const c2 = increment();

const c3 = increment();

console.log(c1, c2, c3);

 \*/

//  Какое будет выведено значение: let x = 5; alert( x++ ); ? сначала 5. если поменяем ++ то 6

//let x = 5;

//alert(x++);

//  Чему равно такое выражение: [ ] + false - null + true ? пустой масив будет строка (если бы было одно действие). нематематичские действия приводят к NaN

//[] + false - null + true; // NaN

//console.log([] + false - null + true);

//  Что выведет этот код: let y = 1; let x = y = 2; alert(x); ? последовательное присваивание справа налево //2

//let y = 1;

//let x = (y = 2);

//alert(x);

//  Чему равна сумма [ ] + 1 + 2? пустой масив будет строка и следующие примитивные значение превращает в строку - "12"

//console.log([ ] + 1 + 2);

//  Что выведет этот код: alert( "1"[0] )?

//alert("1"[0]); // строка примитивная с одного значения и в масиве будет выводится под 1 номером и получаем 1

//  Чему равно 2 && 1 && null && 0 && undefined ? оператор && (и) всегда запинается на лжи(false) null = false

//console.log(2 && 1 && null && 0 && undefined);

//  Есть ли разница между выражениями? !!( a && b ) и (a && b)? !! возвращают булиновые значение и естественно не будет равно

//console.log(!!(1 && 2) === (1 && 2));

//  Что выведет этот код: alert( null || 2 && 3 || 4 );

// смотрим таблицу приоритетов "и" выше сначала тогда && возвращает большее значение, потом null = false и возвращается 3, и 3 "или" 4 возвращает правду - 3

//? оператор || (или) запинается на правде

//оператор && (и) всегда запинается на лжи(false)

//          (2)3  (1)3  (3)3

//alert(null || (2 && 3) || 4);

//  a = [1, 2, 3]; b = [1, 2, 3]; Правда ли что a == b ?

// нет это разные хранилище информации и разными данными и неважно что они выглядят одинаково

//const a = [1, 2, 3];

//const b = [1, 2, 3];

//console.log(a == b);

//  Что выведет этот код: alert( +"Infinity" ); ? выведет Infinity, но это будет number, так как стоит унарный плюч

//alert(+"Infinity");

//  Верно ли сравнение: "Ёжик" > "яблоко"? false нет так как при сравнении буква в нижнем регистре выше чем в верхнем

//console.log("Ёжик" > "яблоко");

//  Чему равно 0 || "" || 2 || undefined || true || falsе ?

// оператор || (или) запинается на правде - ответ 2

//console.log(0 || "" || 2 || undefined || true || falsе);

//

/\* const restorantData = {

  menu: [

    {

      name: "Salad Caesar",

      price: "14$",

    },

    {

      name: "Pizza Diavola",

      price: "9$",

    },

    {

      name: "Beefsteak",

      price: "17$",

    },

    {

      name: "Napoleon",

      price: "7$",

    },

  ],

  waitors: [

    { name: "Alice", age: 22 },

    { name: "John", age: 24 },

  ],

  averageLunchPrice: "20$",

  openNow: true,

};

function isOpen(prop) {

  let answer = "";

  prop ? (answer = "Закрыто") : (answer = "Открыто");

  return answer;

}

console.log(isOpen(restorantData.openNow));

function isAverageLunchPriceTrue(fDish, sDish, average) {

  if (

    +fDish.price.slice(0, -1) + sDish.price.slice(0, -1) <

    +average.slice(0, -1)

  ) {

    return "Цена ниже средней";

  } else {

    return "Цена выше средней";

  }

}

console.log(

  isAverageLunchPriceTrue(

    restorantData.menu[0],

    restorantData.menu[1],

    restorantData.averageLunchPrice

  )

);

function transferWaitors(data) {

  const copy = Object.assign({}, data);

    // Нам просто нужно менять весь массив данных,

    // а не лезть напрямую менять каждого из сотрудников

    // Так как это верхний уровень объекта, то значение

    // будет меняться только у копии

  copy.waitors = [{ name: "Mike", age: 32 }];

  return copy;

}

transferWaitors(restorantData); \*/