let fierstName="Jon"; /\*создание переменной\*/

const age=30; /\*константа с постоянным значением,как в java\*/

let lastName="N`uman";

/\* const age=35; <-- нельзя присваивать другое значение\*/

/\*в отличие от const, let изменять можно, т.е. присваивать новые значения \*/

let floatingPoint = 0.9; /\* можно int, float \*/

let fullName = fierstName+ " "+lastName;/\*конкатинация строк как в java\*/

let interpolation=`${fierstName} ${lastName}`; /\*interpolation-что-то что работает как конкатинация строк, но синтаксис слегка другой \*/

console.log(interpolation, fullName, age, floatingPoint);/\*аналог Syso, вывод в консоль \*/

/////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

if(fierstName === "Mary" && lastName==="Wort"){ /\* if-как в java только с 3-мя знаками равно \*/

console.log("its Mary"); /\* можно применять операторы сравнения или логические операторы \*/

}

else{

console.log("not Mary");

}

//////////////////////////////////////////////////////// Методы ////////////////////////////////////////////////////////////////////

function nazvanieFunction( ){

console.log("можно делать методы и вызывать их так-же как в Java");

}

nazvanieFunction(); //вызов метода

/////////////////////////////////////////////////// создание обьекта /////////////////////////////////////////////////////////////////////

let person\_object = { /\* синтаксис создание обьекта \*/

name:"Steev",

sername: "Madison",

age: 30,

'city-adress': "Mosty", //когда в названии свойства есть дефис, необходимо брать название в одинарные кавычки.

greeting:function(){

console.log("Обьекту можно добовлять методы, это делается в теле обьявления обьекта, вызывать через точечную нотацию");

console.log("Если надо обратится к переменной обьекта при создании метода, т.е. будучи в блоке обьявления обьекта, необходимо использовать точечную нотацию или this. далее пример "+person\_object.name+" "+this.name);

}

}

person\_object.email="person\_object@email.com"; /\*способ добавления свойств обьекту - в java это пришлось бы делать в классе \*/

console.log(person\_object);

console.log("Чтобы получить отдельное сволйство у обьекта используй точечную нотацию "+person\_object.age)

console.log("Но если у имени свойства есть дефис то используй следущий синтаксис: название\_обьекта['название-свойства']"+

"иначе будет undifined, далее пример-"+ person\_object['city-adress']);

person\_object.greeting();

////////////////////////////////////////////////////// JSON vs JS /////////////////////////////////////////////////////////////////

let etoJSON\_iz\_obekta\_person\_object=JSON.stringify(person\_object);// построение JSON из обьекта созданного выше

console.log("Это уже JSON а не обьект "+etoJSON\_iz\_obekta\_person\_object);

let object\_from\_JSON=JSON.parse(etoJSON\_iz\_obekta\_person\_object);

console.log("Это опять обьект конвертированный из JSON, обратимся к его полю email: "+object\_from\_JSON.email);// из JSON в object

// Необходимо помнить, что конвертируя объект из JSON-а мы теряем все методы/функции, методы это реализация в javascript, JSON 'заточен' чтобы быть кросс платформенным средством передачи данных на другие языки програмирования и не может в себе нести дополнительную нагрузку реализации методов, только данные.

///////////////////////////////////////////////// arrays and navigating in it //////////////////////////////////////////////////////////

let array = []; //empty array

let array1 = [ 'Дерево', 'яблоко', 'bitcoin', 30, {object\_from\_JSON}]; //массивы могут быть с разными типами данных: String, int, object..

console.log(array1.length);

console.log(array1);

let objectExample = {

name: "обьект для испытаний",

array\_in\_Object: ['faceBook', 'Tik-tok', {name: 'какоето имя', city: 'Mosty'}],//сразу пишеш название без let(особенность создания полей в JS у обьекта)

}

console.log("ПОИСК ПО МАССИВУ "+objectExample.array\_in\_Object[2].city);// у обьекта есть массив(достать его можно через точечную нотацию), массив-это обьект,к элементам массива обрасчаемся через индэкс, но в массиве, одним из элементов является обьект с полями, в этом обьекте мне нужено к примеру значение поля city, обрасчаемся к нему через точечную нотацию и по имени поля.)

objectExample.array\_in\_Object.push('YouTube');//можно добовлять обьекты в массив

objectExample.array\_in\_Object.pop(); //удаляет последний элемент и возврасчает его, т.е. если создать переменную то элемент какбы перейдёт из массива в переменную, при это при любом раскладе её в массиве не будет.

for(let i=0;i<objectExample.array\_in\_Object.length;i++){ //классичечкий for-чик

console.log(objectExample.array\_in\_Object[i]);

}

objectExample.array\_in\_Object.forEach(logItems);//данная структура аналог цикла for, но здесь на обьекте вызываем массив, затем forEch

function logItems(item, index, array){ //который должен в параметрах принимать так-называемую callBackFunction, т.е. функцию. Описание

console.log(item+" "+index);//функции <-- здесь, можно вызвать элемент, индэкс и массив, я х.з.

}

// можно немного упростить данную инструкцию,

objectExample.array\_in\_Object.forEach(function(item, index, array){//это ananimys funktion, всё тоже

console.log(array.length);//что и в верху, но я не определяю метод отдельно, а прописываю прямо

}) //в forEach, как бы наверно экономится место,но слегка запутывает.

//ещо упрастить

objectExample.array\_in\_Object.forEach ( (item, index, array) => {//убираем слово function, ставим =>

console.log(array.length); //приходим к концэпту ()=> {}

})

//ещо

objectExample.array\_in\_Object.forEach ( console.log);

// так-как consol.log метод, то он подходит в качестве callBackfunction. Смотри в консоль, думаю там можно поигратся с параметрами

let tikTok\_check =false;

objectExample.array\_in\_Object.forEach( (item, index, array) => { //пример применения forEach

if(item === "Tik-tok"){

tikTok\_check=true;

}

})

console.log(tikTok\_check);

*console*.log('В данной вкладке описываются команды которые будут выполнены до того как будет отправлен риквест');

*console*.clear(); //убирает/чистит консоль.

