16. Функции, стрелочные ф-ции (ES6)

// 1. FUNCTION DECLARATION

function showFirstMesagge() {

    console.log('Hello world');

}

showFirstMesagge();

let num = 20;

function showFirstMesagge(text) {

    console.log(text);

    let num = 10;

}

showFirstMesagge('Hello world');

console.log(num);

// function calc(a, b) {

//     return a + b;

// }

// console.log(calc(4, 7));

// console.log(calc(5, 12));

// выходит num наружу

function ret() {

    let num = 50;

    return num; // выходит num наружу (с функции)

}

const actors = ret();

console.log(actors);

// Замыкание функций

let num2 = 20;

function showFirstMesagge(text) {

    console.log(text);

    console.log(num2); // ищет вышестоящее значение

}

showFirstMesagge('Hello world');

console.log(num2);

// 2. FUNCTION EXPRESSION

const log = function() {

    console.log('Hello world');

};

log();

// 3. СТРЕЛОЧНЫЕ ФУНКЦИИ

const log1 = () => {

    console.log('Hello world');

};

log1();

const calc = (a,b) => {

    return a + b;

};

console.log(calc(4, 7));

console.log(calc(5, 12));

**17. Методы и свойства строк и чисел**

// 17. Методы и свойства строк и чисел

const str = 'test';

console.log(str.length); // свойство строки

// методы строк

const strUpper = str.toUpperCase(); // метод строки

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

console.log(str);

console.log(strUpper);

// метод поиск подстроки

const fruit = 'Some fruit';

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

// метод slice()

const logg = 'Hello world';

console.log(logg.slice(6, 11)); // вырезает кусочек строки с 6 по 11 индекс. (получаем world)

// метод substring()   тоже самое что и slice()

// метод substr()

console.log(logg.substr(6, 4)); // вырезает кусочек с 6 индекса (2 цифра сколько вырезать)

// Числа

const number = 12.2;

console.log(Math.round(number)); // округление

const test = '12.2px';

console.log(parseInt(test)); // метод превращает строку в целое число (12)

console.log(parseFloat(test)); //метод превращает строку в число (12.2)

19. Callback- функции

/19. Callback- функции

function learnJS(lang, callback) {

    console.log(`Я учу ${lang}`);

    callback();

}

function done() {

    console.log('Я прошел этот урок');

}

learnJS('JS', done);

learnJS('JS', function(){

    console.log('Я прошел этот урок');

});

20. Объекты, деструктуризация объектов (ES6)

// 20. Объекты, деструктуризация объектов (ES6)

const options = {

    name: 'text',

    widht: 1024,

    height: 1024,

    colors: {

        border: 'black',

        bg: 'red'

    },

    makeTest: function() {   // делаем метод в объекте

        console.log('test');

  }

};

options.makeTest();  // запускаем метод    круто!!!

//деструктуризация объектов (ES6)

const {border, bg} = options.colors;

console.log(border);

// узнать количество свойств в объекте

console.log(Object.keys(options).length);

// узнать свойство в объекте

//console.log(options.name);

// перебор свойств в объекте

for (let key in options) {

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

        for (let i in options[key]) {

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

        }

    } else {

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

    }

}

**21. Массивы и псевдомассивы**

const arr = [1, 2, 3, 6, 8];

arr[99] = 7;

console.log(arr.length); // свойство length +1 !!!

console.log(arr);

// методы которые работают с концом массива

arr.pop(); // удаляет последнее элеиент массива [1, 2, 3, 6]

arr.push(10); // добавляет в конец элемент массива [1, 2, 3, 6, 8, 10]

// перебор элементов в массиве

// метод forEach !!!

arr.forEach(function(item, index, arr){

    console.log(item);

});

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

    console.log(arr[i]);

}

for (let i of arr) {

    console.log(i);

}

// метод split('.')          из строки (через точку('.')) превращает в массив

const str3 = prompt("", "");

const products = str3.split('.');

console.log(products);

// метод join(";")   наоборот       из массиа (через (";")) превращает в строку

const str4 = prompt("", "");

const products1 = str4.split('.');

products1.sort();     // метод sort() сортирует строки

console.log(products1.join(";"));

// метод sort()

const arr2 = [21, 5, 10, 82, 9];

arr2.sort(compareNum);

console.log(arr2);

function compareNum(a, b) {  // с этой функцией метод sort сортирует числа по порядку.

    return a - b;

}

**22. Передача по ссылке или по значению, Spread  
оператор (ES6-ES9)**

**Клонирование объектов**

# // Object.assign()

const numbers = {

    a: 10,

    b: 12,

    c: {

        d: 2,

        e: 5

    }

};

const add = {

    f: 10,

    k: 2

};

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

clone.a = 20;

clone.d = 10;

console.log(clone);

console.log(numbers);

# // Spread

// Оператор разворота Spread (...)

//массивы

const video = ['youtube', 'vimio', 'rutube'],

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

    internet = [...video, ...blogs, 'vk', 'facebook'];

    console.log(internet);

function log(a, b, c) {

    console.log(a);

    console.log(b);

    console.log(c);

}

const num = [1, 5, 10];

log(...num);

const array = ['a', 'b'];

const newArray = [...array];

//объекты

const obj = {

    a: 1,

    b:5,

    c: {

        f: 15,

        g: 20

    }

};

const newObj = {...obj};

newObj.a = 10;

console.log(newObj);

console.log(obj);

# 23. Основы ООП, прототипно-ориентированное наследование

const soldier = {

    health: 400,

    armor: 100,

    sayHello: () => {

        console.log('Hello');

    }

};

// const jonh = {

//     health: 100

// };

//jonh.\_\_proto\_\_ = soldier; // прототип старый \_\_proto\_\_

//Object.setPrototypeOf(jonh, soldier); // прототип новый с помощью Object.setPrototypeOf

const jonh = Object.create(soldier); // создаем новый объект-прототип с помощью Object.create

jonh.health = 100;

console.log(jonh.health);

jonh.sayHello();