1. Переменные

"use strict";//директива которая указывает скриптовому файлу,

//что мы работаем в современном режиме, !использовать всегда!

//прописывать в начале скриптового файла

let number = 5; // = - присваивание //let - переменная изменяемая

const leftBorderWidth = 1; // цифра не может быть первой в названиии

//переменной,нельзя использовать зарезирвированные слова.

number = 10;

console.log(number);

const obj = { //прямых констант в js небывает. что такое прямая константа

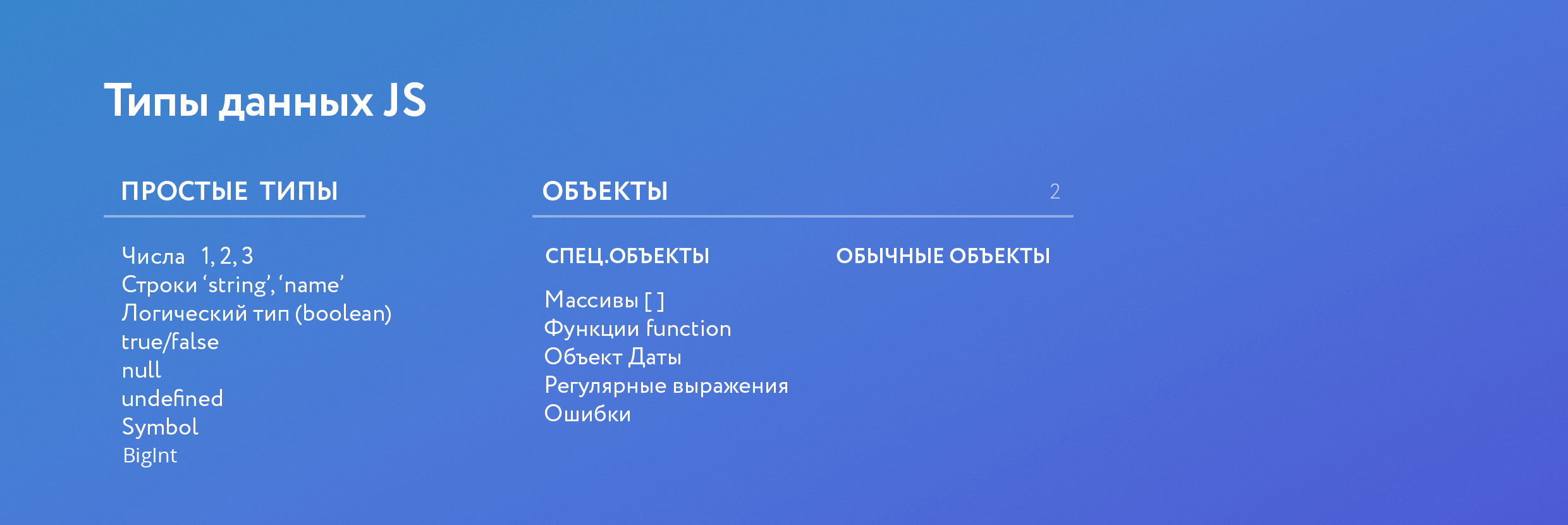
    a:50

};

obj.a = 10;

console.log(obj);

1. Типы данных



Undefined – что-то существует, но этого нет (стоит холодильник, но он пустой )

BigInt – работа с большими числами. (ограничение с обычными числами. Мы не можем задать число, которое больше чем 2 в 53 степени)

"use strict";

//тип данных - число

let number = 4.6; //числа, даже дробные

console.log(4/0);//infinity

console.log('string'\*9);//nan - не число

//типданных - строка

const persone = "5";

//логичесий тип данных (boolean)

const bool = true;

//тип анных null И undefined

console.log(something);

let und;

console.log(und);

//тип данных - объект

const obj = {// это объект

    name: "john",// свойства объекта

    age: 25,// свойства объекта

    isMarried: false// свойства объекта

};

console.log(obj.name); // обратились к значению свойства name  объекта

//массив это частный случай объекта, о не является типом данных

console.log(obj["name"]);//конструкция в квадратных скобках

//используется для хранения данных идущих по порядку

let arr = ['plum.png', 'orange.jpg', 'apple.bmp', 6, {}, []];

console.log(arr[1]);// в программировании исчисление начинается с 0

//[Running] node "e:\3.Projects JS\first\tempCodeRunnerFile.js"

//orange.jpg

1. Простое общение с пользователем

"use strict";

//alert('hello');

//const result = confirm('are you here?');

//console.log(result);

//происходит запись результата

//const answer = prompt("Вам есть 18?", "17");

//console.log(answer); // ответ записывает от пользователя, тип данных строка

//вся информация от пользователя птип данных является строка,

//если не преобразован тип данных программистом

//const answer = +prompt("Вам есть 18?", "18");

//console.log(typeof(answer)); в этом случаи будет number тип данных

const answers = [];

//answers[0]=prompt ('как ваше имя', '');

//answers[1]=prompt ('ваша ффамилия', '');

//answers[2]=prompt ('сколько вам лет', '');

//document.write(answers);

console.log(typeof(answers));

console.log(typeof(null));//ошибка, но такая вот особенность

4.Интерполяция

Это значит, что мы можем вставлять внутри строки значение переменной.

const category = 'toys';

//console.log('https://someurl.com/' + category); // конкоцинация (https://someurl.com/toys)

console.log(`https://someurl.com/${category}/5`);//не забывать про апострофы

const user = 'Ivan';

console.log(`Privet, ${user}`);

5.Операторы

console.log('arr' + '- object'); // конкоцинация, строки сложатся и правратятся

//в большую строку -  arr- object;

console.log(4 + '- object'); // тоже самое

console.log(4 + +'5');// 9, унарный плюс +"5", использует один аргумент

//икримент и дикримент

let incr = 10,

    decr = 10;

    incr++;//увеличение на единицу(постфикснаяя форма записи)

    decr--;//уменьшение знач на ед.(постфикснаяя форма записи)

   // ++incr;//(префиксная форма записи)

// --decr;//(префиксная форма записи)

    console.log(incr); //11 тоже самое будет и при префиксной форме

    console.log(decr); // 9

    console.log(incr++); //10

    console.log(decr--); //10

    /\*в постфиксной фоме возвращается старое значение,

     а после произвоит действие\*/

     console.log(5%2); // 1 - выводит остаток

     console.log(2\*4 == 8); //true сравнение происходит по значению, а не по типу данных

     // если сравнивать по типу данных то

     console.log(2\*4 === '8'); // false, строгое сравнение, сравнивает и по тиру данных

     /\* && - оператор и - работает когда два либо более значения являются правдивыми

        || - оператор или - работает когда хотябы одно значение является правдивым

      \*/

    const isChecked = true;

          isClose = false;

          console.log(isChecked || isClose); // выдает true

          const isChecked = true;

          isClose = false;

          console.log(isChecked && isClose); // выдает false, true выдаст, когда оба значения будут true

          // ! - оператор отрицания

           console.log(2 + 2 \* 2 === 8);//false

           console.log(2 + 2 \* 2 != 8); // true

           console.log(2 + 2 \* 2 !== '6');//true

callback – исполнение строго заданное, пока не выполнится заданная функция, втрая выполнятся не будет

'use strict';

function first() {

    //do something

    setTimeout(function() { //эмуляция задержки на 0,5с

        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 прошёл этот урок');

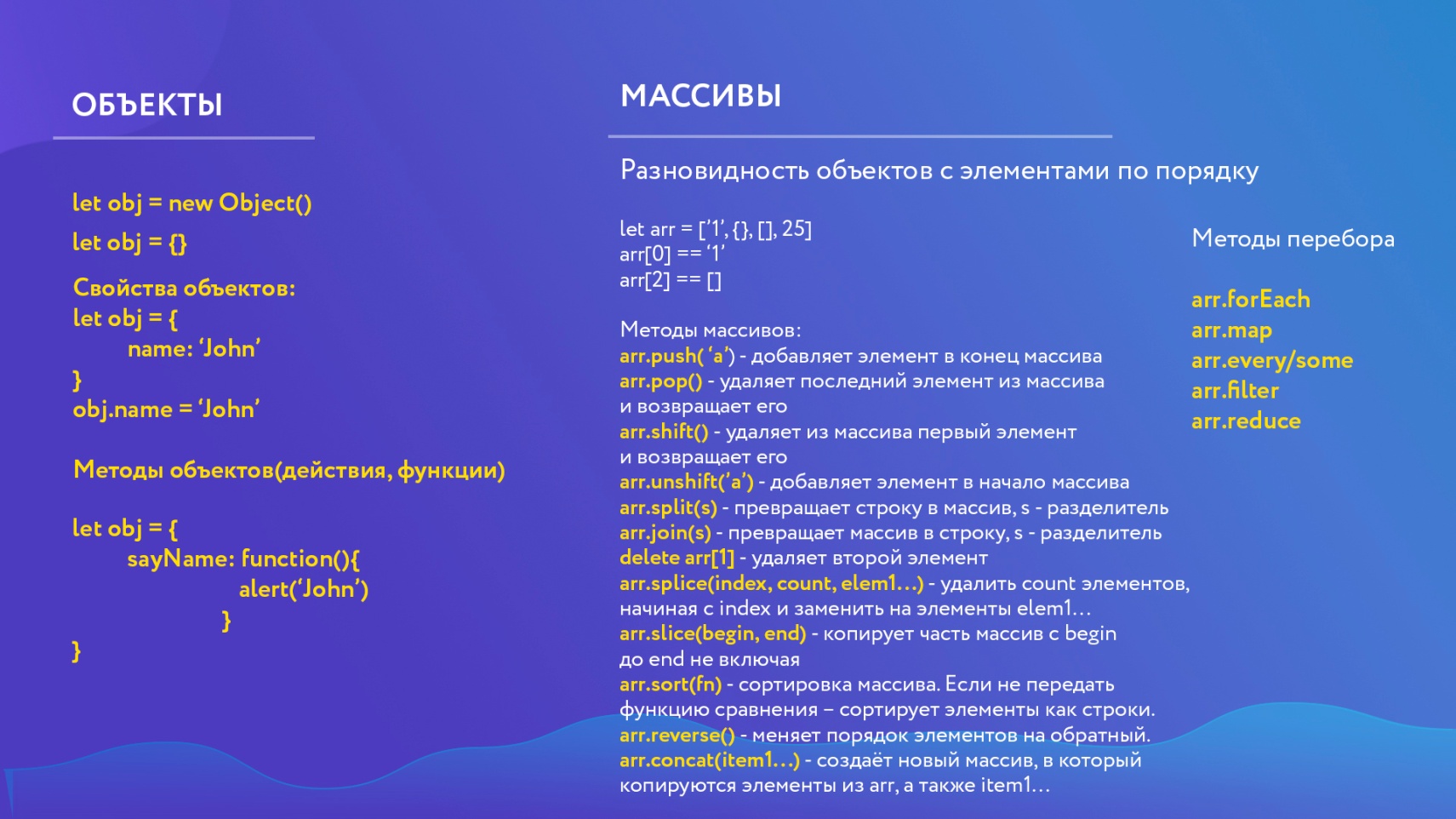
}

learnJS('JAvascript', done);

//круглых скобок нет, потому что мы не вызываем функцию,

// а передаем, что бы она была использована

Объекты, деструктуризация объектов



'use strict';

//обекты деструктуризация объектов

const options = {

    name: 'test',

    width: 1024,

    height: 1024,

    colors:{

        border: 'black',

        bg: 'red'

    },

    //методы - это действия которые может совершать объект

    //а этот метод мы создаем сами

    makeTest: function(){

        console.log('Test');

    }

};

options.makeTest();

//деструктуризация

const {border, bg} = options.colors;

//мы вытащили кусочки вложеные в свойства в качестве отдельнй переменной

console.log(border);

//console.log(Object.keys(options).length); // получение количества элементов внутри объекта

//это встроенныеметод

//ЗАПОМНИТЬ!!!

//console.log(options.name);

//delete options.name;//опертор удаления

//console.log(options);

//{ width: 1024, height: 1024, colors: { border: 'black', bg: 'red' } }

//без строчки тест

//прием счетчика

/\*let counter = 0;

for (let key in options) { //роемся в свойствах

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

        for (let i in options[key]){

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

            counter++;

        }

    } else {

        console.log(`Свойства ${key} имеет значение ${options[key]}`);//key это name width и т.д

        //таким обазом мы будем получать значение этого ключа которые перебирается в этой итерации

        //значений цикла

        counter++;

    }

}

console.log(counter);

\*/

//ИТОГ

/\*

чтобы перебрать мы можем использовать конструкцию for in

у объектов бывают встроенные методы  встроенные свойства

так же мы можем и создавать методы

 \*/

021 массивы и псевдомассивы

"use strict";

//Массивы и псевдомассивы

const arr = [1, 2, 3, 6, 8];

arr.sort(compareNum);

console.log(arr);

function compareNum(a,b) {

    return a - b;

}

//почитать алгоритмы

//arr[99] = 0;

//сonsole.log(arr.length);

//стостоит из последнего индекса в массиве + 1

//arr.forEach(function(item, i, arr){

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

//});

//arr.pop();//удление последнего элемента с массива

//[ 1, 2, 3, 6 ]

//arr.push(10);//добавляет заданный элемент в конец массива

//[ 1, 2, 3, 6, 8, 10 ]

//console.log(arr);

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

  //  console.log(arr[i]);

//} //перебор элементов

//for (let value of arr) {

   // console.log(value);

//}//второй пособ перебора элементов но:

/\*

работает с массивоподобными сущностями

 \*/

 const str = prompt('', '');

 const products = str.split('', '');

 products.sort();//сортирует элементы как строки

 console.log(products.join('; '));

Передача по ссылке или по значению, оператор spread

'use strict';

//передача по ссылке или по значению, оператор spread

// let a = 5,

//     b = a;

//     b = b +5;

//     console.log(b);

//     console.log(a);

    //const obj = {

       // a: 5,

       // b: 1

    //};

    //const copy = obj;//передаёт ссылку, мы не скопировали новый объект

    //а передает именно ссылку на существующий объект

//copy.a = 10;

//console.log(copy);

//console.log(obj);

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;//измениться везде

console.log(newNumbers);

console.log(numbers);

/\*{ a: 10, b: 5, c: { x: 10, y: 4 } }

{ a: 2, b: 5, c: { x: 10, y: 4 } }

потому что создана поверхностная копия

\*/

const add = {

    d: 17,

    e: 20

};

const clone = Object.assign(numbers, add);

clone.d = 225;

// console.log(add);

// console.log(clone);

const oldArray = ['a', 'b', 'c'];

const newArray = oldArray.slice();//метод позволяющий скопировать старый массив

newArray[1] = 'asdfsdfsdfdfs';

console.log(newArray);

console.log(oldArray);

//четвертый способ создания поверхностной копии

// spread оператор разворота, разворачивает структуру и превращает ее

// в набор данных

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

    blogs = ['wordpess', 'livejounal', '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 = [2, 5, 7];//информация пришелшая с сервера

    //нам нужно ередать ее в функцию

    log(... num);

//еще один способ

const array = ['a', 'b'];

const newAarray = [...array];

const q = {

    one: 1,

    two: 2

};

const newObj = {...q};