Filter  
name = [‘KiRill’, ‘IvAn’, ‘VoVa’, ‘DinIs’]

shortName = name.filter((elem) => {  
 return elem.length < 5  
});

Map

lowerName = name.map((elem) => {

return elem.toLowerCase();

});

Rest и reduce

Function max(…numbers) {

Numbers.reduce((maxNumber, currentElem) => {  
 if (maxNumber < currentElem) {  
 return currentElem  
 } else {  
 return maxNumber  
 }  
 })

}

Spread

function foo(x, y, z) {   
 console.log(x, y, z);   
}   
let arr = [1, 2, 3];   
foo(...arr); *// 1 2 3*

Obj Spread

*let* user = {

name: 'Ivan',

root: 'user',

age: 20

};

*let* admin = {

name: 'Kirill',

root: 'admin'

},

X = 10;

*const* newAdm = {

...user,

...admin,

X,

Method () {  
 …  
 }

};

*console*.log(newAdm);

result - { name: 'Kirill', root: 'admin', age: 20 }

Упрощение объектов

*const* x = 10, y = 20;

*const* coord = {

x,

y,

calcSqr() {

*console*.log(this.x \* this.y);

}

}

coord.calcSqr();

Создание поверхностной копиии объекта

Const res = Object.assign({}, obg1, obg2);

Деструкторизация объектов

*let* user = {

name: {

first = 'Kirill',

second = 'Fyodorov'

},

root: 'user',

age: 20

};

*const* {name: {first: firstName, second: secondName}, root, age, connect, …otherParam = ‘localhost’} = user;

*console*.log(secondName); //рузультат – Fyodorov

*console*.log(connect); //рузультат – localhost – задание некотрым свойствам значений по умолчанию

*function* connect({

*host* = 'localhost',

*port* = 3000,

*user* = 'default'

} = {}) {

*console*.log(host, port, user);

}

connect({

port: 232

}); // результат localhost, 232, default

Деструкторизация массивов

*let* mas = [1, 2, 3, 5];

*const* [a, b, c] = mas; //a == 1, b ==2, c==3

*let* mas = [1, 2, 3, 5];

*const* [, , c] = mas; //c==3

*let* mas = [[1, 2], [3, 5]];

*const* [[a, b], [c, d]] = mas;

Прототипы

*function* Animal(*name*, *voice*) {

this.name = name;

this.voice = voice;

}

*Animal*.prototype.say() {

*console*.log(`${this.name} goes ${this.voice}`);

}

*const* dog = new Animal('dog', 'woof');

*const* cat = new Animal('cat', 'meow');

dog.say();

cat.say();

Классы

*class* Animal {

*constructor* (*name*, *voice*) {

this.name = name;

this.voice = voice;

}

say() {

*console*.log(`${this.name} ${this.voice}`)

}

}

*class* Bird extends *Animal* {

*constructor* (*name*, *voice*, *canFly*) {

super(name, voice);

super.say();

this.canFly = canFly;

}

say() {

*console*.log(`${this.name} dont like fly`)

}

};

*const* duck = new Bird ('duck', 'quack', true);

duck.say();

РЕЗУЛЬТАТ –

duck quack

duck dont like fly