**Блочная область видимости.**

В JavaScript присутствует функциональная область видимости. Это означает, что все переменные, объявленные c помощью ключевого слова var, будут видны в любом месте функции (даже если они объявлены внутри блока).  
  
function f(a) {

If (a<0){

Var I = 3;

}

Console.log(i);

}

F(-1)

В новой версии присутствует ключевое слово let, которое позволит объявлять переменные с блочной областью видимости.

Function f(a) {

If (a<0){

Let I = 3;

}

Console.log (i);

}

F(-1)

**Константы**

Const MAX=100; //нельзя перезаписать (let, var ….)

Let MAX = 100 //ERROR

**Значения параметров по умолчанию.**

В функциях появилась возможность объявлять у параметров значения по умолчанию:

Function setLevel (newLevel = 0){

}

SetLevel(); // new level=0

setLevel(5); // setlevel = 5

setLevel(undefined); // setlevel = 0

**Деструктуризация: массивы**

//для массивов

Let [x,y]=[10, 20, 30] //нестрогий matching

Console.log(x,y) // на выходе 10, 20

**Деструктуризация: объекты**

//for object

Let user = {name: “Ann”, location:{x:10, y:20}};

Let {name: username, location:{x:x, y:y}}=user;

Console.log(username, x, y); //”Ann”, 10, 20

**Деструктуризация: обмен переменных**

Let x =10;

Let y=20;

[x, y] = [y, x] //возможно

**Замена arguments: “rest” и “spread”**

**Объект arguments// ES3, ES5**

function format(pattern /\*, rest \*/) {

var rest = [].slice.call(arguments, 1); //СЛОЖНО

var items = rest.filter(function (x) { return x > 1});

return pattern.replace(“%v”, items);

}

format(“scores: %v”, 1, 5, 3); // scores: 5, 3

**Привет, “rest”// ES6 aka Harmony**

Function format(pattern, …rest) { // настоящий массив

var items = rest.filter(function (x) { return x > 1});

return pattern.replace(“%v”, items);

}

format(“scores: %v”, 1, 5, 3); // scores: 5, 3

**А также “spread”// ES6 aka Harmony**

function showUser(name, age, weight) {

return name + “:” + age + weight;

}

let user = [“Alex”, 28, 130];

showUser(…user);

// okshowUser.apply(null, user); // desugared

**“rest” массивов при деструктуризации// ES6 aka Harmony**

let userInfo = [“John”, 14, 21, 3];

let [name, ...scores] = userInfo;

console.log(name); // “John”

console.log(scores); // [14, 21, 3]

**Сокращенные нотации**

**Сокращения в деструктуризации**

let 3DPoint = {x: 20, y: 15, z: 1};

// полная нотация

let {x: x, y: y, z: z} = 3DPoint;

// сокращенная нотация

let {x, y, z} = 3DPoint;

**Короткий синтаксис функций -> функции**

// обычная функция

[1, 2, 3].map(function (x) { return x \* x; }); // [1, 4, 9]

// -> функция

[1, 2, 3].map((x) -> x \* x); // [1, 4, 9]

Синтаксически:

– опциональный return;

– -> вместо function

– Блочная скобки не обязательны

**-> функции: примеры**

// Пустая функция

let empty = ->;

// Скобки тела функции необязательны

let square= (x) -> x \* x;

// Функция без параметров

let getUser = -> users[current];

// Для сложных выражений скобки нужны

let users = [{name: “Mark”, age: 28}, {name: “Sarah”, age: 26}];

users.forEach((user, k) -> { if (k > 2) console.log(user.name, k) });

**Обычные функции: динамический this**

function Account(customer, cart) {

this.customer = customer;

this.cart = cart;

$(„#shopping-cart).on(click, function (event) { this.customer.purchase(this.cart); // error on click

});

}

Решения: var that = this; .bind(this)

**=> функции: лексический this**

function Account(customer, cart) {

this.customer = customer;

this.cart = cart;

$(„#shopping-cart).on(click, (event) =>

this.customer.purchase(this.cart); // OK

);

}

**Короткий синтаксис функций функции-блоки**

// обычная функция

[1, 2, 3].map(function (x) { return x \* x; }); // [1, 4, 9]

// функция-блок

[1, 2, 3].map {|x| x \* x}; // [1, 4, 9]

Синтаксически:

– опциональный return;

– |x| вместо function

– Необязательные скобки вызова

**Proxy объекты: мета уровень**

Proxy-объекты

/\* handler – обработчик мета-уровня \* proto – прототип прокси-объекта \*/

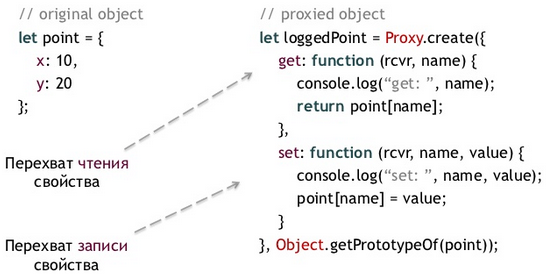
Proxy.create(handler, [proto])

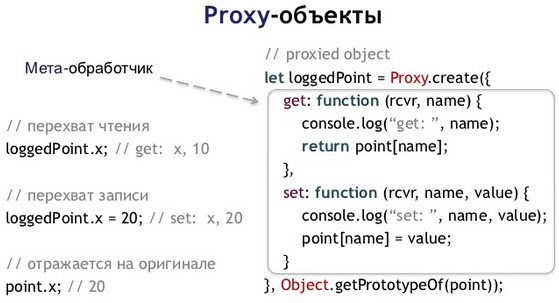
/\* handler – мета-обработчик \* call – проксирование вызова \* construct – проксирование конструирования \*/

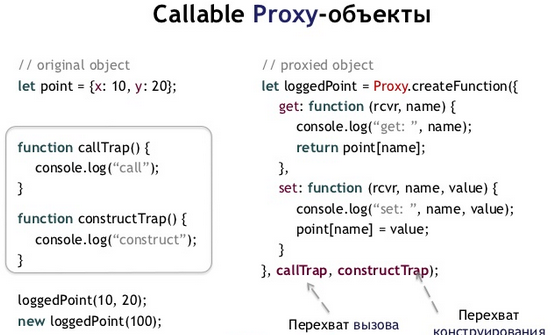
Proxy.createFunction(handler, [call, [construct]])

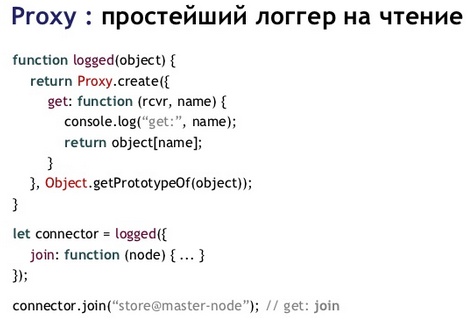
See: http://wiki.ecmascript.org/doku.php?id=harmony:proxies

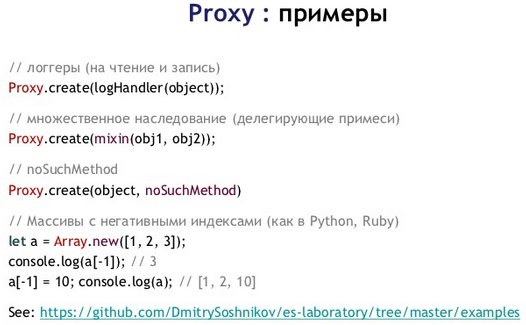
**Proxy-объекты**











**Система модулей**

