Функции

Аргументы. Область видимости. Всплытие. Замыкание

Жигалов Сергей

Функции

- Объединить группу действий
- Не повторять код (DRY)
- Рекурсивный вызов
- Создать область видимости*

Аргументы

Васнецов. "Три богатыря"

Аргументы

```
function min(a, b) {
    return a < b ? a : b;
}
```

```
min(2, 7);     // 2
min(3, 4, 2); // 3
min(13);       // undefined
```

Аргументы

```
function min(a, b) {
    return a > b ? b : a;
}
```

```
min(2, 7);      // 2
min(13);        // 13
```

Аргументы

```
function min(a, b) {
    if (b === undefined) {
        return a;
    }

    return a < b ? a : b;
}
```

```
min(2, 7);      // 2
min(13);        // 13
```

Аргументы. Значения по умолчанию

```
function min(a, b) {
    b = b || Infinity;

    return a < b ? a : b;
}
```

```
min(2, 7);      // 2
min(13);        // 13
```

Аргументы. Значения по умолчанию

```
function getCost(price, count) {
    count = count || 1;

    return price * count;
}
```

```
getCost(27.70, 10); // 🍏 277₽
getCost(49.90);     // 🍅 49.9₽
```

```
getCost(9999, 0);   // 🚀 9999 ???
```

Аргументы. Значения по умолчанию

```
function getCost(price, count) {
    if (count === undefined) {
        count = 1;
    }

    return price * count;
}
```

```
getCost(27.70, 10); // 🍏 277₽
getCost(49.90);     // 🍅 49.9₽
getCost(99999, 0);  // 🚀 0₽
```

Именованные аргументы

```
function BMI(params) {
    var height = params.height;

    return params.weight / (height * height);
}
```

```
BMI({ weight: 60, height: 1.7 }) // 20.7
```

Именованные аргументы. Достоинства

- Удобно, если несколько необязательных аргументов
- Неважен порядок аргументов
- Неограниченное число аргументов
- Легко рефакторить код

Именованные аргументы. Недостатки

- Неявный интерфейс
- Неудобно работать с аргументами

arguments

Объект arguments - это подобный массиву объект, который содержит аргументы, переданные в функцию.

Arguments object - JavaScript | MDN

```
function example() {
    arguments[1];      // 12
    arguments.length; // 2
}

example(3, 12);
```

arguments

```
function sum() {
    var a = arguments[0] || 0;
    var b = arguments[1] || 0;

    return a + b;
}
```

```
sum(3, 12);      // 15
sum(45);         // 45
sum(2, 4, 8);    // 6
```

arguments

```
function sum() {
    var sum = 0;

    for(var i = 0; i < arguments.length; i++) {
        sum += arguments[i];
    }

    return sum;
}
```

```
sum(2, 4, 8);   // 14
```

arguments

```
function sum() {
    var args = [].slice.call(arguments);

    return args.reduce(function (sum, item) {
        return sum + item;
    });
}
```

```
sum(2, 4, 8);   // 14
```

Методы функции. Call

```
function example() {
    [1, 2].slice();           // [1, 2]
    [].slice.call([3, 4]);    // [3, 4]

    [].slice.call(arguments); // [5, 6]
}

example(5, 6);
```

Объявление функции

Васнецов. "Витязь на распутье"

Объявление функции

```
// function declaration
function add(a, b) {
    return a + b;
}
```

```
// function expression
var add = function (a, b) {
    return a + b;
}
```

Объявление функции. function declaration

```
add(2, 3);  // 5

function add(a, b) {
    return a + b;
}
```

Объявление функции. function expression

```
add(2, 3);  // TypeError

var add = function (a, b) {
    return a + b;
}
```

Named function expression

```
var factorial = function inner(n) {
    return n === 1 ?
        1 : n * inner(n - 1);
}
```

```
typeof factorial; // 'function'
typeof inner;     // ReferenceError
```

```
factorial(3);   // 6
```

Конструктор Function

```
var add = new Function('a', 'b', 'return a + b');
```

```
add(2, 3);  // 5
```

Область видимости

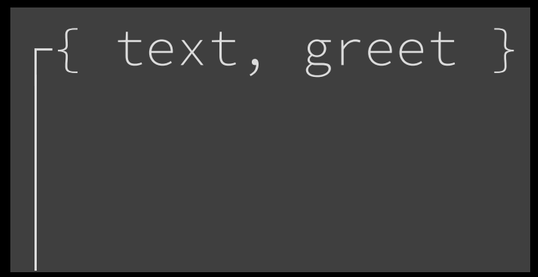
Глобальный объект

Код:

```
var text = 'Привет';

function greet() {
}
```

Область видимости:




```
global.text; // 'Привет'
```

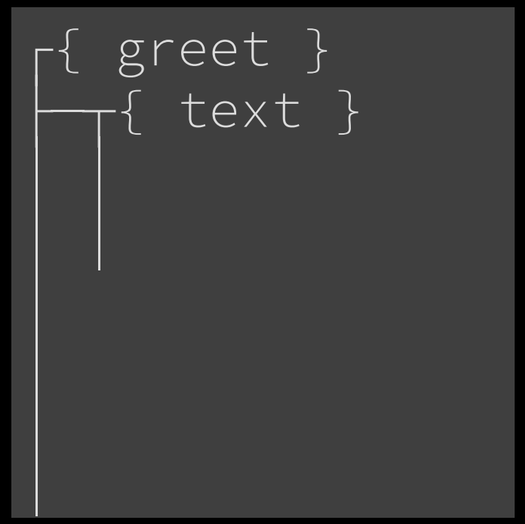
Создание области видимости

Код:

```
function greet() {
    var text = 'Привет';
    text; // 'Привет'
}

text;   // ReferenceError:
        // text is not defined
```

Область видимости:



✕ Нет блочной области видимости*

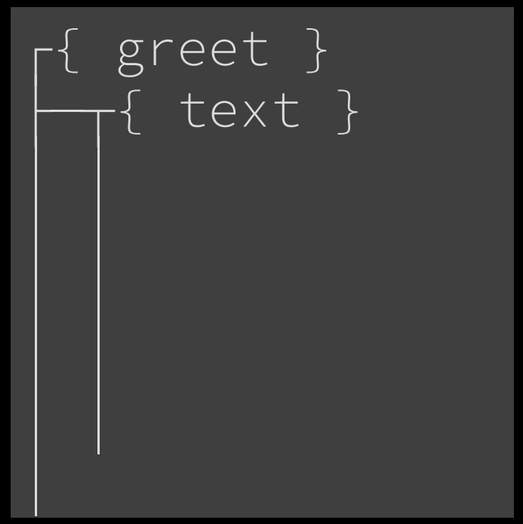
* Standard ECMA-262 5.1 Edition

Код:

```
function greet() {
    if (true) {
        var text = 'Привет';
    }

    text; // 'Привет'
}
```

Область видимости:



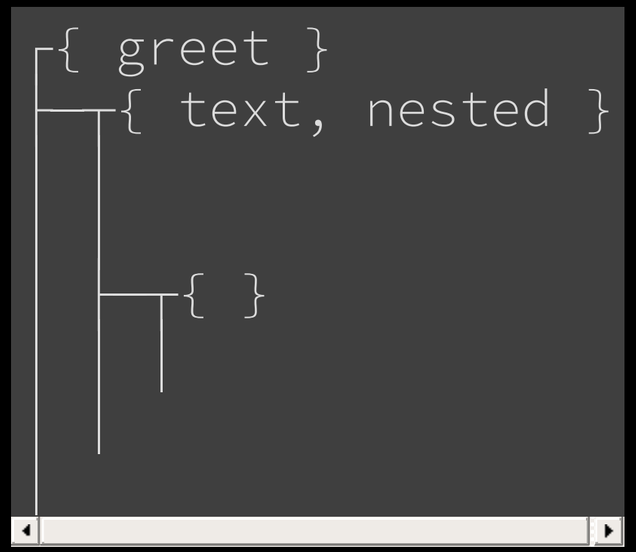
Вложенные функции

Код:

```
function greet() {
    var text = 'Привет';

    function nested() {
        text; // 'Привет'
    }
}
```

Область видимости:



Затенение

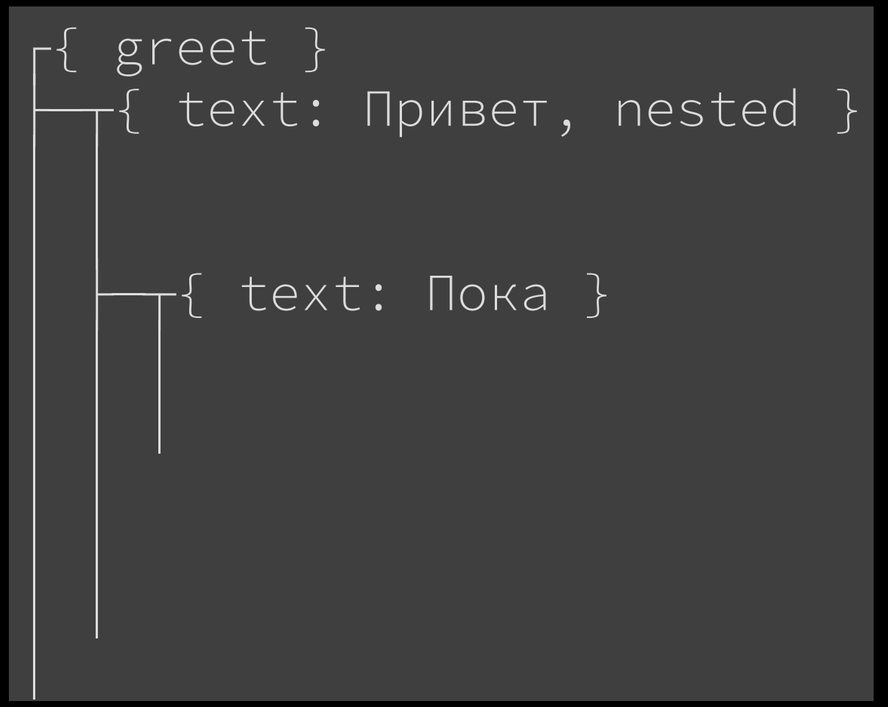
Код:

```
function greet() {
    var text = 'Привет';

    function nested() {
        var text = 'Пока';
        text; // 'Пока'
    }

    text; // 'Привет'
}
```

Область видимости:



Всплытие

Выполнение кода

1. Инициализация
   1. `function declaration`
   2. `var`
2. Собственно выполнение

Всплытие функций

Код:

```
add(2, 3);

function add(a, b) {
    return a + b;
}
```

Значение переменных:

```
{ add: function }
```

Всплытие функций

Код:

```
add(2, 3);  // 5

function add(a, b) {
    return a + b;
}
```

Значение переменных:

```
{ add: function }
```

Всплытие переменных

Код:

```
add(2, 3);

var add = function (a, b) {
    return a + b;
}
```

Значение переменных:

```
{ add: undefined }
```

Всплытие переменных

Код:

```
add(2, 3); // TypeError

var add = function (a, b) {
    return a + b;
}
```

Значение переменных:

```
{ add: undefined }
```

Всплытие переменных

Код:

```
add(2, 3); // TypeError

var add = function (a, b) {
    return a + b;
}
```

Значение переменных:

```
{ add: undefined }
```

Всплытие переменных

Код:

```
add(2, 3); // TypeError

var add = function (a, b) {
    return a + b;
}
```

Значение переменных:

```
{ add: function }
```

Всплытие переменных

Код:

```
function greet() {
    var text = 'Привет';
}
```

Значение переменных:

```
{ greet: function }
```

Всплытие переменных

Код:

```
function greet() {
    var text = 'Привет';
}
```

Значение переменных:

```
{ greet: function, text: undefined }
```

Всплытие переменных

Код:

```
function greet() {
    var text = 'Привет';
}
```

Значение переменных:

```
{ greet: function, text: 'Привет' }
```

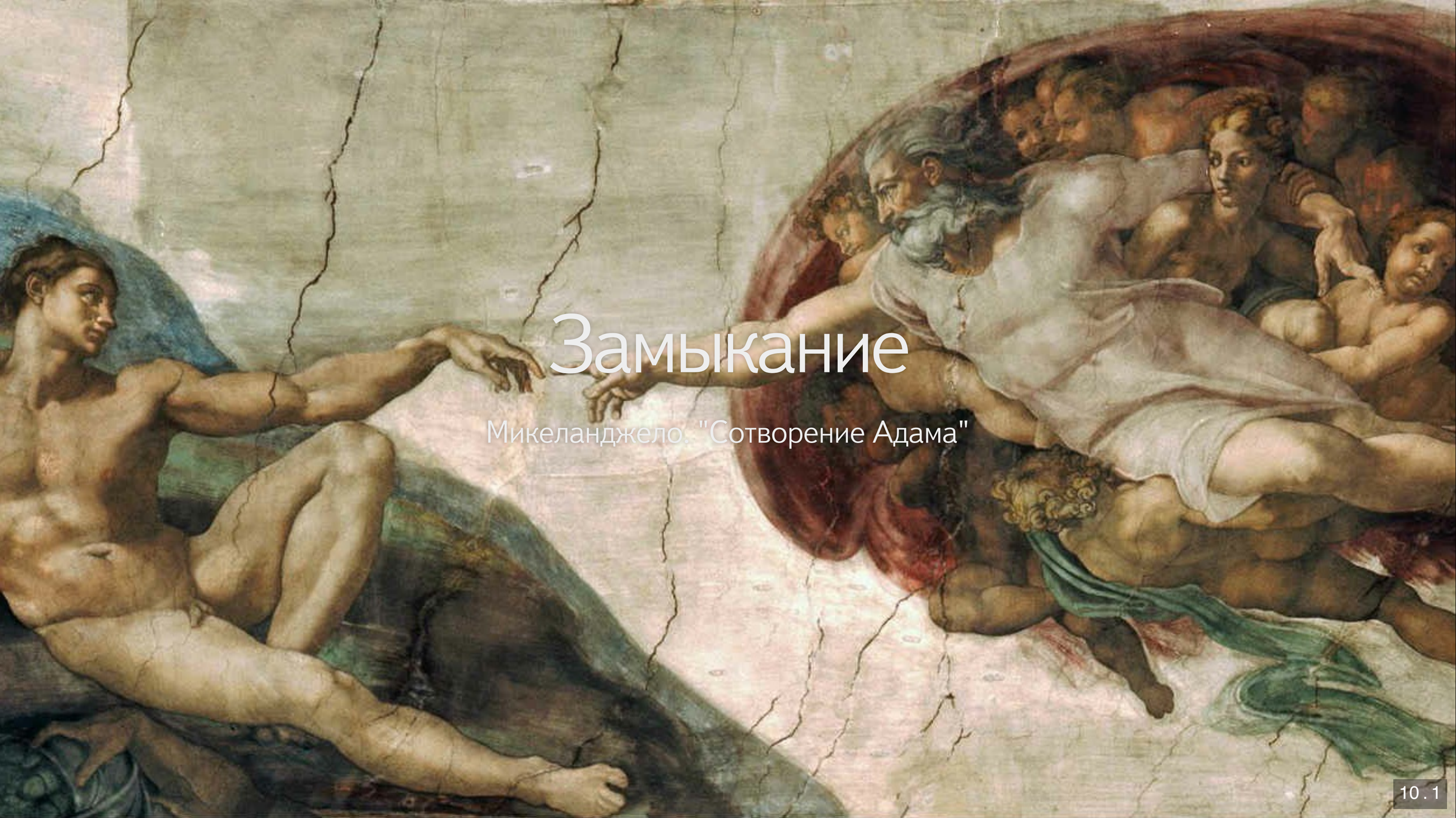
Всплытие переменных

Код:

```
function greet() {
    var text = 'Привет';
}
```

Значение переменных:

```
{ greet: function }
```



Замыкание

Микеланджело. "Сотворение Адама"

Замыкание — это функция вместе со всеми внешними переменными, которые ей доступны.

Ссылка на переменные

Код:

```
function greet() {
    var text = 'Привет';
}

greet();
```

Счётчик ссылок:

```
{ text: 1 }
```

Ссылка на переменные

Код:

```
function greet() {
    var text = 'Привет';
}

greet();
```

Счётчик ссылок:

```
{ text: 0 }
```

Ссылка на переменные

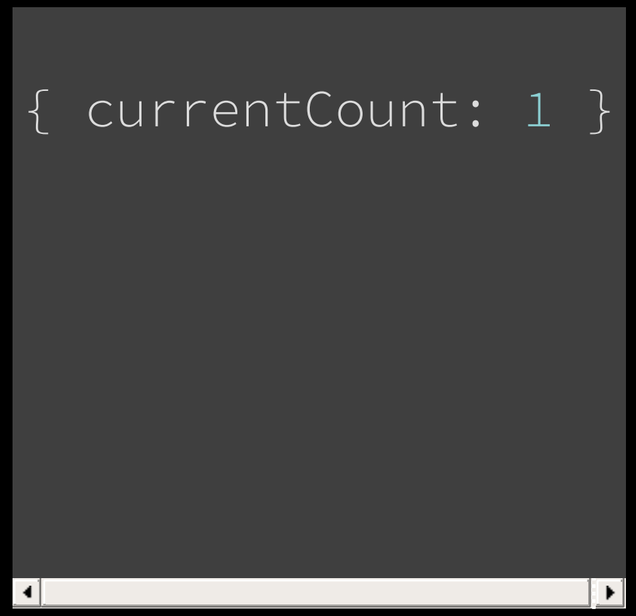
Код:

```
function makeCounter() {
    var currentCount = 0;

    return function () {
        return currentCount++;
    };
}

var counter = makeCounter();
```

Счётчик ссылок:



Ссылка на переменные

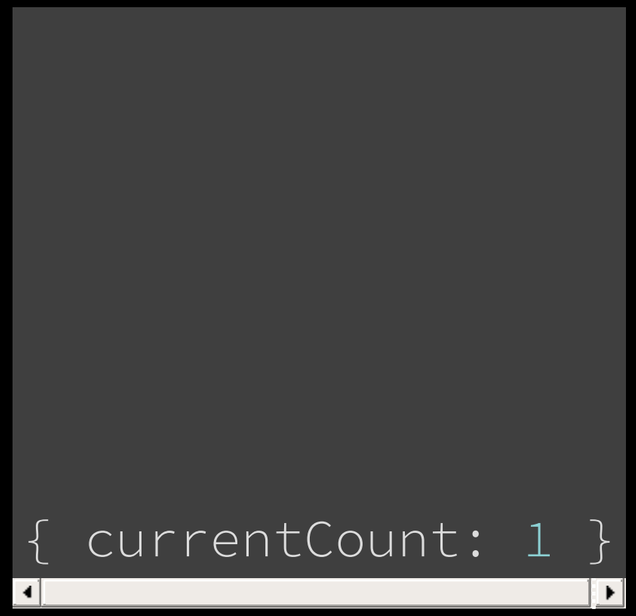
Код:

```
function makeCounter() {
    var currentCount = 0;

    return function () {
        return currentCount++;
    };
}

var counter = makeCounter();
```

Счётчик ссылок:



Замыкание

Код:

```
function makeCounter() {
    var currentCount = 0;

    return function () {
        return currentCount++;
    };
}
```

Область видимости:

```
{ makeCounter } // 1
  { currentCount } // 2

    { } // 3
```

```
var counter = makeCounter();
counter(); // 0
counter(); // 1
counter(); // 2

var yetAnother = makeCounter();
yetAnother(); // 0
```

Замыкание

```
function greet(name) {
    return function () {
        return 'Привет, ' + name;
    }
}

var helloWorld = greet('мир!');
```

```
helloWorld(); // "Привет, мир!"
```

Модуль

```
function format(date) {
    return date.toGMTString()
}

function getDateString(date) {
    date = date || new Date();
    return format(date);
}
```

```
getDateString();
// "Thu, 27 Oct 2016 10:56:52 GMT"
```

```
function format() {
    return '💥';
}
```

```
getDateString();
// "💥"
```

IIFE

immediately-invoked function expression

IIFE

```
var getDateString = (function () {
    function format(date) {
        return date.toGMTString()
    }

    return function getDateString(date) {
        date = date || new Date();
        return format(date);
    }
}());
```

IIFE

```
(function () {
}());
```

```
(function () {
})();
```

```
!function () {
}(); // X
```

```
void function () {
}(); // X
```

Почитать

- Область видимости в JavaScript ...
- Замыкания в JavaScript
- Замыкания, область видимости
- Лексическая область видимости