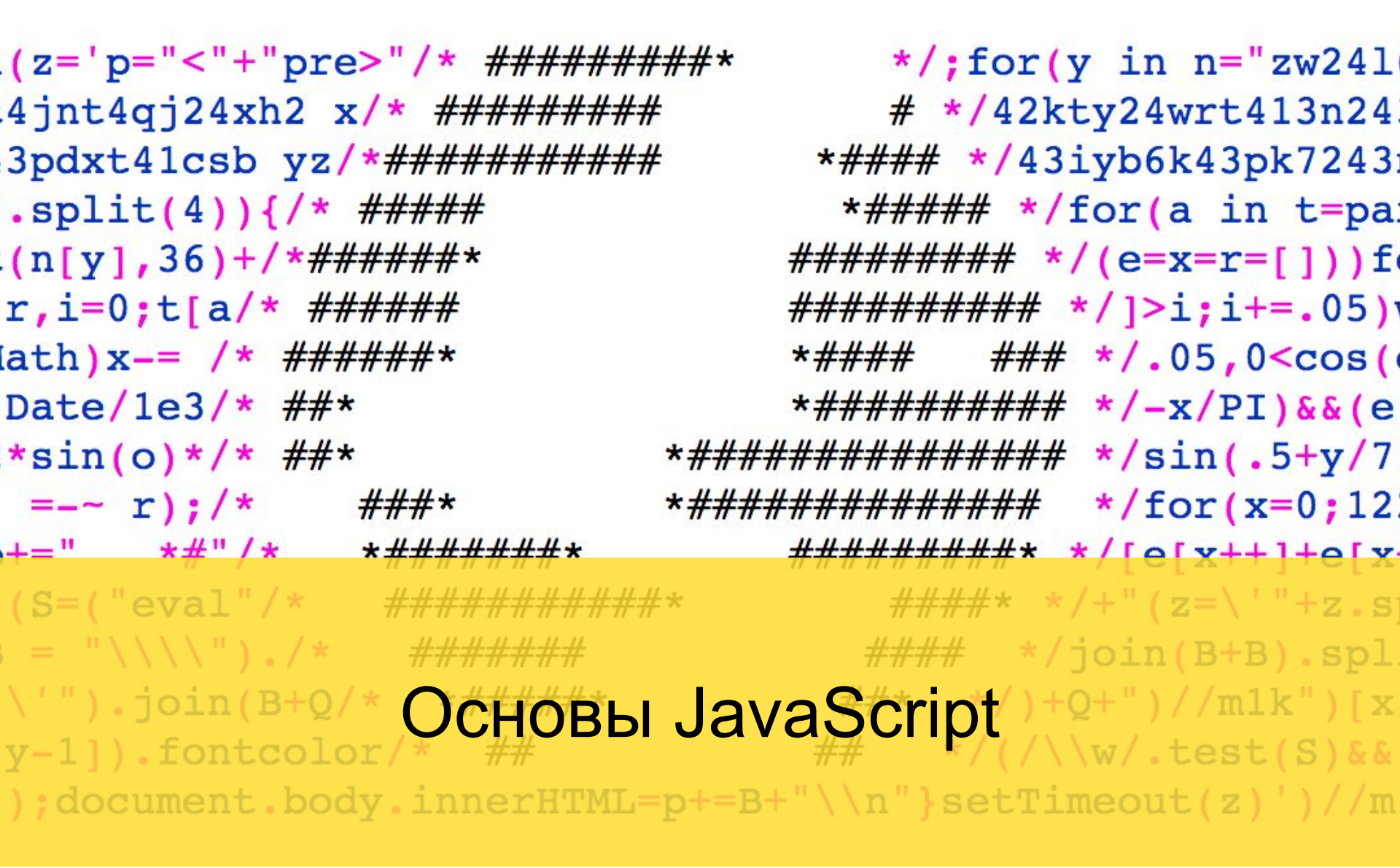


Основы JavaScript

Tinkoff.ru

Давайте знакомиться

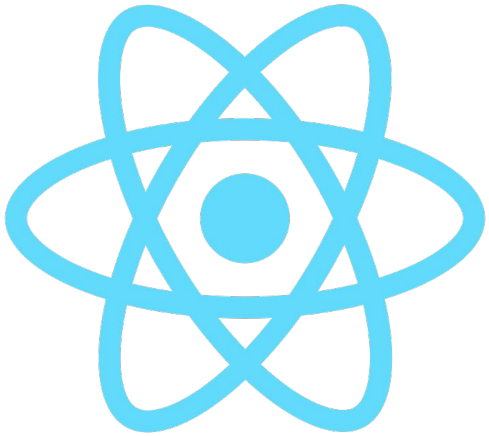
Андрей Алексеев

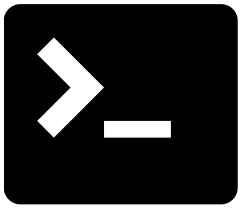
t.me/aalexeev239

старший разработчик

в Тинькофф для бизнеса

Курс





Фронтенд в Тинькофф

2 больших направления:

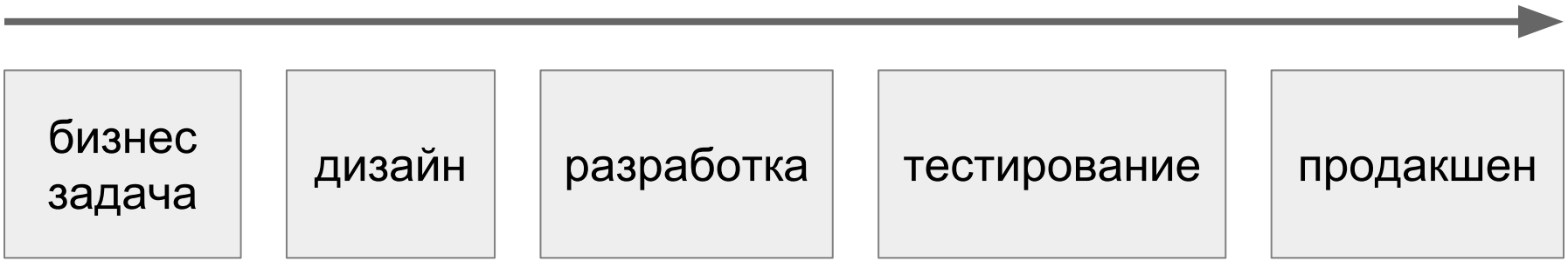
платформа для физлиц / для бизнеса

120+ фронтенд разработчиков

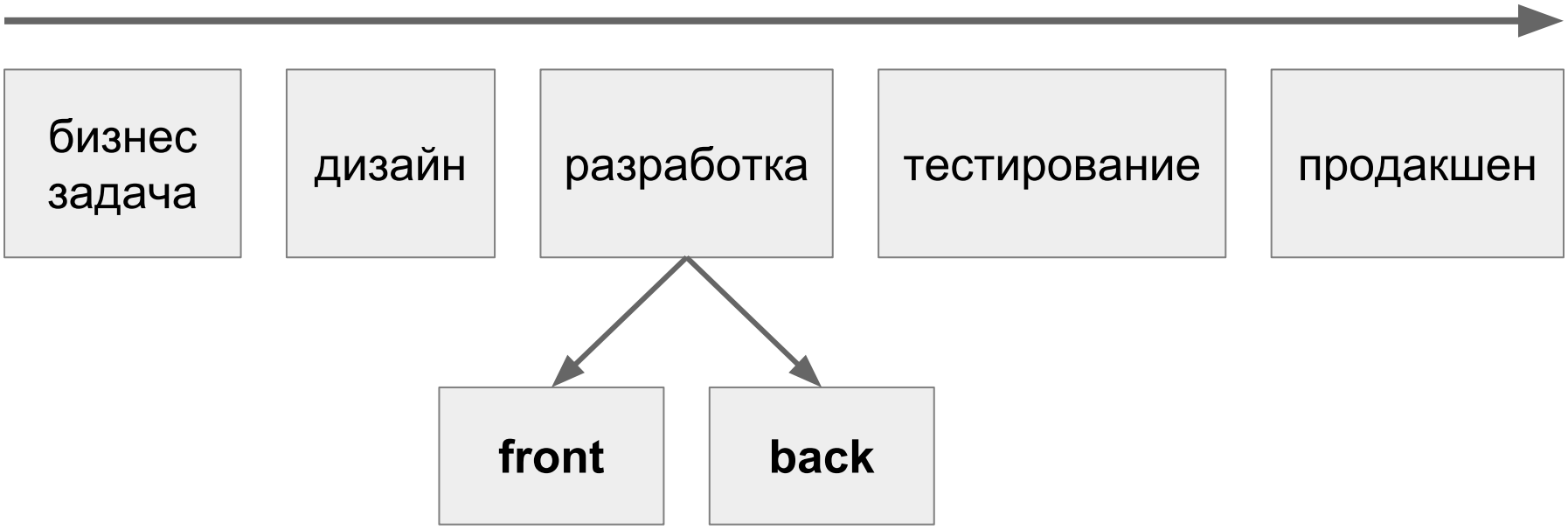
6 центров разработки: мск, спб, нн, екб, нск, инп

массивный стек технологий

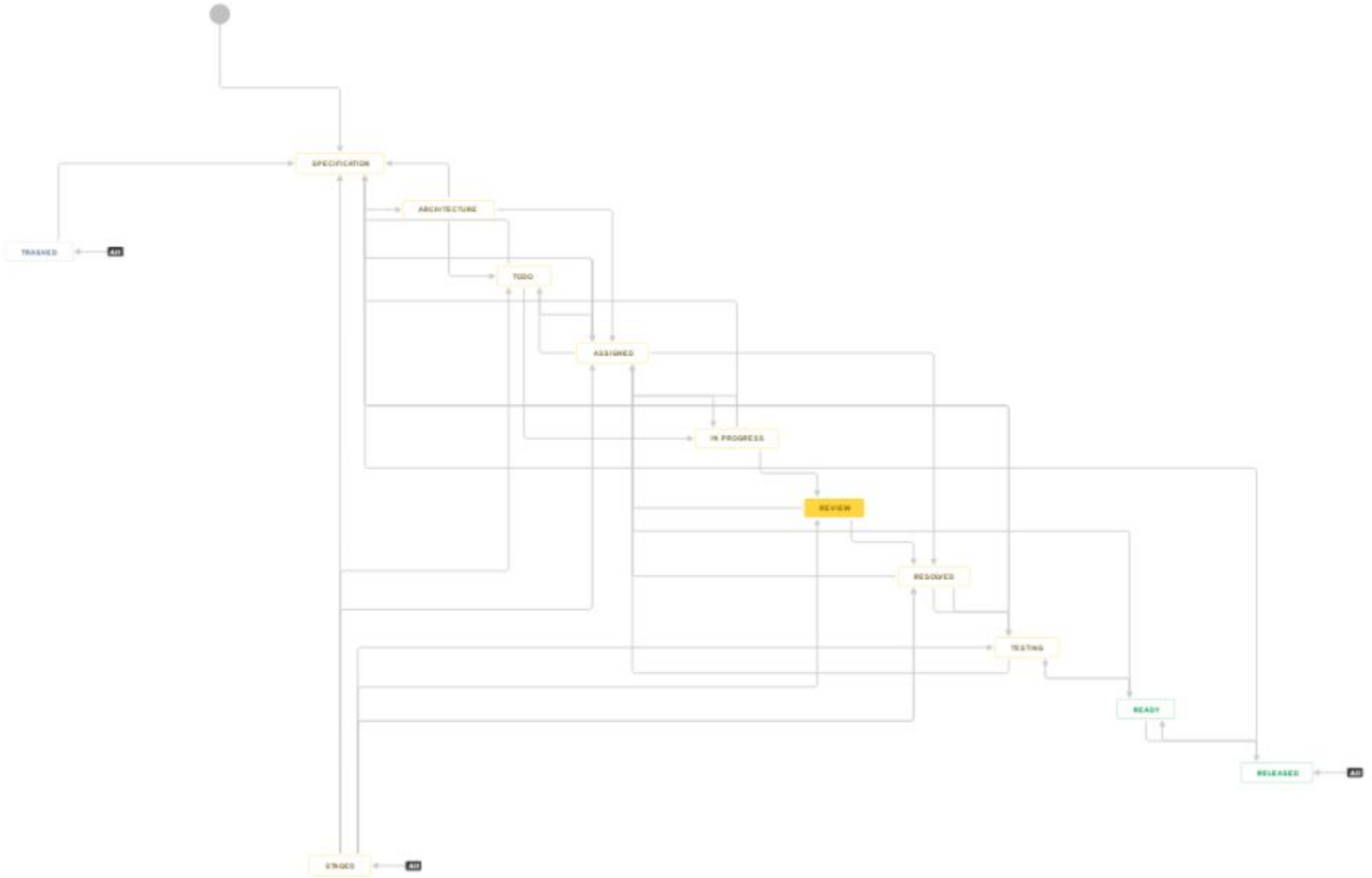
Процесс разработки



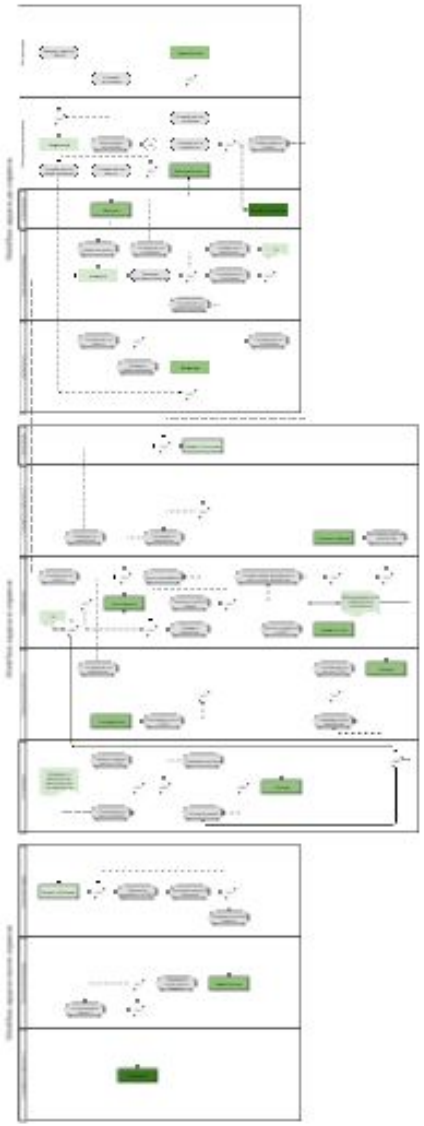
Процесс разработки



На самом деле



И даже так



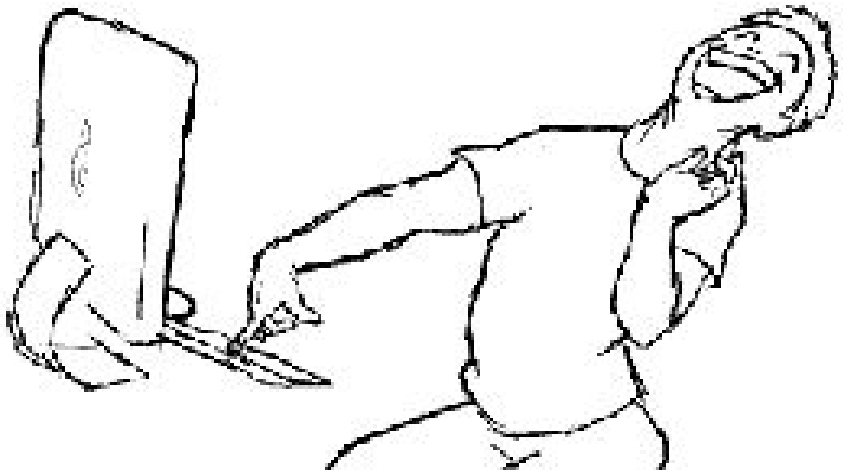
Чем занимаются фронтендеры

продуктовые задачи

unit-тесты

e2e-тесты

рефакторинг



ECMASCRIPT 2017

История

1995  **LiveScript**

1996  **JavaScript**

JScript 

1999 **ECMAScript, ES3**

2009 **ECMAScript, ES5**

2015 **ES6, ES2015**

2016 **ES7, ES2016**

2017 **ES8, ES2017**

История

На хабре:

Краткая история JavaScript. Часть 1

Краткая история JavaScript. Часть 2

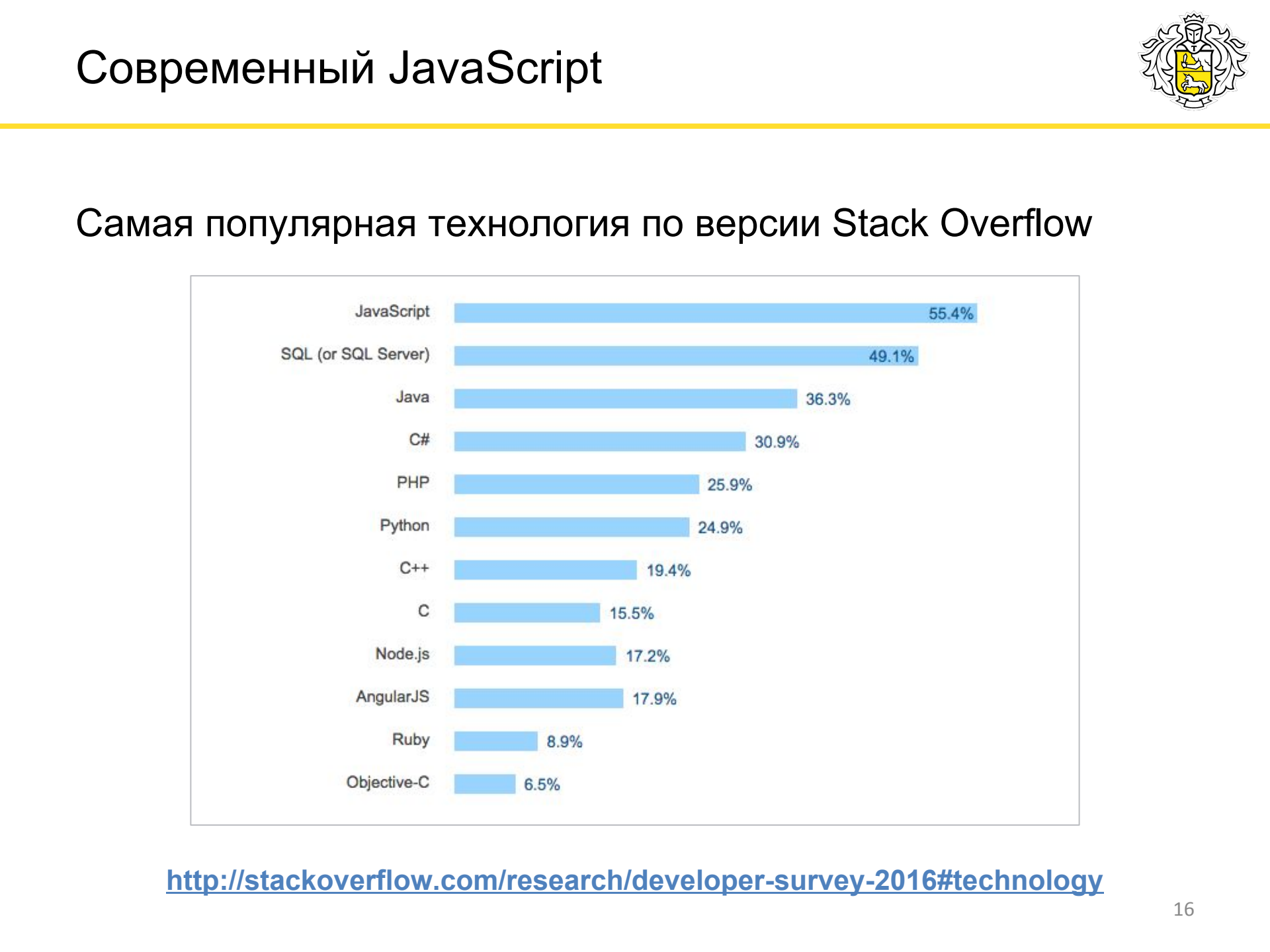
Краткая история JavaScript. Часть 3

ECMAScript

ECMAScript

ECMAScript

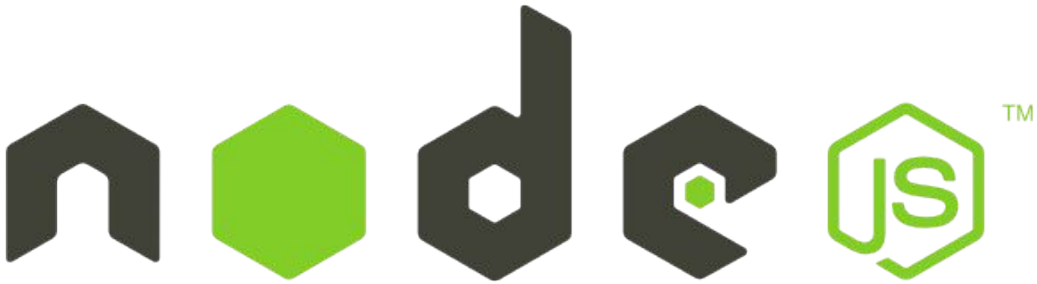
Современный JavaScript



Самая популярная технология по версии Stack Overflow

http://stackoverflow.com/research/developer-survey-2016#technology

Не только в браузере



ИНСТРУМЕНТЫ

Редакторы кода

IDE и редакторы кода:

WebStorm

Atom

Visual Studio Code

SublimeText

Онлайн-песочницы:

JS Bin

Plunker

JSFiddle

Демка

Запуск скриптов

JavaScript – интерпретируемый язык

Внешние скрипты, порядок исполнения

Eslint

Dev Tips

https://umaar.com/dev-tips/

https://umaar.com/dev-tips/125-optimise-web-dev-workflow/

ПЕРЕМЕННЫЕ

Переменные

```
var x; // undefined
x = 'Вася'; // ‘Вася’
x = 'Миша';
x = 4;
x = x + 1; // 5

var y = 10; // 10
```

https://learn.javascript.ru/variables

Допустимые значения

```
var _ = 1;
var $ = 1;
var o_O = 1;
var test1 = 1;
var camelCase = 1;
var CONST_CASE = 3.14;
```

Зарезервированные слова

abstract	arguments	boolean	break	byte
case	catch	char	class*	const
continue	debugger	default	delete	do
double	else	enum*	eval	export*
extends*	false	final	finally	float
for	function	goto	if	implements
import*	in	instanceof	int	interface
let	long	native	new	null
package	private	protected	public	return
short	static	super*	switch	synchronized
this	throw	throws	transient	true
try	typeof	var	void	volatile
while	with	yield		

http://www.w3schools.com/js/js_reserved.asp

Const и let

ES6

```
var I_AM_CONST = 10;
var i = 0;
i = i + 1;
```

```
const I_AM_CONST = 10;
I_AM_CONST = 11; // TypeError
let i = 0;
i = i + 1;
```

https://learn.javascript.ru/let-const

var – функциональная область видимости

```
function foo() {
   var x = 5;
   if (x > 6) {
      var y = 7;
   }
}
```

var – функциональная область видимости

```
console.log(x); // ReferenceError: x is not defined
function foo() {
   var x = 5;
   if (x > 6) {
      var y = 7;
   }
}
console.log(x); // ReferenceError: x is not defined
```

var – функциональная область видимости

```
console.log(x); // ReferenceError: x is not defined
function foo() {
   var x = 5;
   console.log(x); // 5
   if (x > 6) {
      var y = 7;
   }
}
console.log(x); // ReferenceError: x is not defined
```

var – функциональная область видимости

```
console.log(x); // ReferenceError: x is not defined
function foo() {
   var x = 5;
   console.log(x); // 5
   console.log(y); // undefined
   if (x > 6) {
      var y = 7; // никогда не выполнится
   }
}
console.log(x); // ReferenceError: x is not defined
```

const, let – блочная область видимости

```
console.log(x); // Error:
function foo() {
   var x = 5;
   console.log(x); // 5
   console.log(y); // undefined
   if (x > 6) {
      var y = 7;
   }
}
console.log(x); // Error
```

```
console.log(x); // Error

if (true) {
   console.log(x); // Error
   const x = 5;
   console.log(x); // 5
}

console.log(x); // Error
```

Что в итоге использовать?

1. const
2. let
3. ~~var~~

ТИПЫ В JAVASCRIPT


АГА, РАССКАЗЫВАЙ
В JAVASCRIPT НЕТ ТИПОВ
risovach.ru

BOOLEAN

Boolean

```
const a = true;
const b = false;

const check = 3 > 5;
if (check) {
   ...
}
```

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Boolean

СТРОКИ

Строки

```
const str1 = 'Ночь темна и полна ужасов';
const str2 = "Зима близко";
```

https://learn.javascript.ru/string

Строки

```
const str1 = 'Ночь темна и полна ужасов';
const str2 = "Зима близко";
const str3 = ' "Ты смотришь слишком много сериалов," — считают мои коллеги';
```

Специальные символы

```
'— Валар моргулис\n— Валар дохаэрис';
// — Валар моргулис
// — Валар дохаэрис
```

Юникод

```
'\u0024'; // $

'\uD83D\uDE00'; // 
```

https://medium.com/reactnative/emojis-in-javascript-f693d0eb79fb

Шаблонные строки

ES6

```
const name = 'Джон Сноу';
const str = `Ничего ты не знаешь, ${name}!`;
// Ничего ты не знаешь, Джон Сноу!
```

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/template_strings

Шаблонные строки

ES6

```
`
— Валар моргулис
— Валар дохаэрис
`
```

Конкатенация

```
'И то,' + 'и другое'; // 'И то,и другое'
```

Методы строк

```
const str = 'qwerty';
str.length; // 6
str[0]; // 'q'
```

Преобразования строк

```
const str = 'Привет;
str.toUpperCase(); // 'ПРИВЕТ
str.toLowerCase(); // 'привет'
str; // 'Привет <- исходная строка не изменилась
```

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/String#Methods_of_String_instance

Методы строк

String.fromCharCode()
String.fromCodePoint()
String.raw()
string.charAt()
string.charCodeAt()
string.codePointAt()
string.concat()
string.includes()
string.endsWith()
string.indexOf()
string.lastIndexOf()
string.localeCompare()
string.match()
string.normalize()

string.quote()
string.repeat()
string.replace()
string.search()
string.slice()
string.split()
string.startsWith()
string.substr()
string.substring()
string.toLocaleLowerCase()
string.toLocaleUpperCase()
string.toLowerCase()
string.toSource()
string.toString()

string.toUpperCase()
string.trim()
string.trimLeft()
string.trimRight()
string.valueOf()
string[@@iterator]()

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/String

ЧИСЛА

Числа

```
let n = 10;
n = 3.14159265359; // все числа float-ы
0xff; // 255
3e5; // 30000
2 + 3; // 5
2 - 3; // -1
5 * 5; // 25
12 / 4; // 3
8 % 3; // 2
```

http://learn.javascript.ru/number

Числа

```
5 / 0
```


5÷0=

Бесконечность

Infinity

Бесконечность

```
Infinity + 5;
```

Бесконечность

Infinity

Бесконечность

```
Infinity * 2;
```

Бесконечность

Infinity

Бесконечность

```
Infinity * -1;
```

Бесконечность

```
-Infinity;
```

Бесконечность

```
1 / Infinity;
```

Бесконечность

0

Бесконечность

```
-1 / Infinity;
```

Бесконечность

```
-0
```

Бесконечность

```
Infinity - Infinity;
```

Не число

NaN;

Не число

```
NaN === NaN; // false
isNaN(5); // false
isNaN('строка'); // true
```

http://learn.javascript.ru/number#проверка-на-число

Методы чисел

isNaN()
isFinite()
isInteger()
isSafeInteger()
toInteger()
parseFloat()
parseInt()

number.toFixed()
number.toLocaleString()
number.toPrecision()
number.toSource()
number.toString()
number.valueOf()

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Number

Преобразование в число

```
Number('100'); // 100
parseInt('100'); // 100
parseInt('100раз23'); // 100
parseInt('раз 100'); // NaN
parseInt('FF', 16); // 255
```

Преобразование в число

```
parseFloat('3.14'); // 3.14
parseFloat('314e-2'); // 3.14
parseFloat('3.14чототам'); // 3.14
parseFloat('FF'); // NaN
+'3.14'; // 3.14
```

Округление, Math

```
Math.floor(3.14); // 3
Math.ceil(3.14); // 4
Math.round(3.14); // 3
Math.trunc(3); // 3
```

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Math

Дробные числа

```
0.1 + 0.2;
```

Дробные числа

```
0.30000000000000004;
```

Отсечение дробной части

```
const n = 12.34;
n.toFixed(1); // '12.3' — строка
```

Отсечение дробной части

```
const n = 0.1 + 0.2;
+n.toFixed(10); // 0.3
```

UNDEFINED

Undefined

```
var x; // undefined
```

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/undefined

NULL

Null

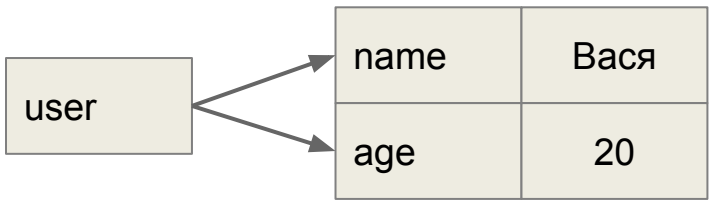
```
var x = null;
```

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/null

OBJECT

Object

```
var user = { name: 'Вася', age: 20 };
```



https://learn.javascript.ru/object

В переменных сохраняются:
для примитивов — значения
для объектов — ссылки

Примитивы и объекты

```
var a = 1;
var b = a;
b = 2;
a; // ?
```

Примитивы и объекты

```
var a = 1;
var b = a;
b = 2;
a; // 1
```

Примитивы и объекты

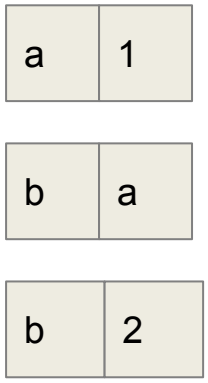
```
var a = {x: 1};
var b = a;
b.x = 2;
a; // ?
```

Примитивы и объекты

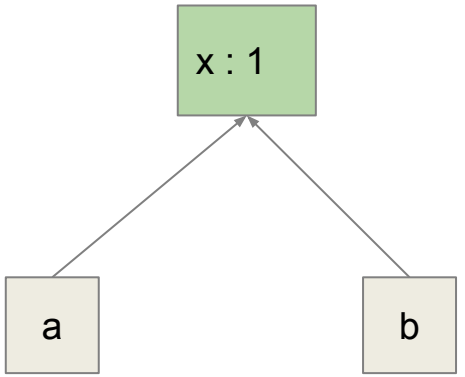
```
var a = {x: 1};
var b = a;
b.x = 2;
a; // Object {x: 2}
```

Примитивы и объекты

```
var a = 1;
var b = a;
b = 2;
a; // 1
```




```
var a = {x: 1};
var b = a;
b.x = 2;
a; // Object {x: 2}
```



ОПЕРАТОРЫ

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators

Арифметические операторы

```
2 + 3; // 5
2 - 3; // -1
5 * 5; // 25
12 / 4; // 3
10 % 3; // 1
(1 + 2) * 4; // 12
```

Приоритет операций

```
3 + 4 * 5; // 23
```

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/Operator_Precedence

Приоритет операций

```
3 + 4 * 5; // 23
```

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/Operator_Precedence

Перегруженный +

Конкатенация строк

```
2 + '3'; // '23'
'2' + 3; // '23'
'foo' + true; // 'footrue'
```

https://learn.javascript.ru/operators

Перегруженный +

Сложение чисел

```
2 + true; // 3
true + true; // 2
2 + null; // 2
1 + 2 + '3'; // '33'
1 + (2 + '3'); // '123'
```

Перегруженный +

Унарный +

```
+(2 - 3); // -1
+'2' + +'3'; // 5
```

Инкремент и декремент

```
let i = 1, j = ++i; // i = 2, j = 2;
let i = 1, j = i++; // i = 2, j = 1;
let i = 1, j = --i;  // i = 0, j = 0;
let i = 1, j = i--;  // i = 0, j = 1;
```

Сравнение

```
2 > 1; // true
2 >= 1; // true
2 === 1; // false
2 !== 1; // true
```

Сравнение с приведением типов

При сравнении значений разных типов используется числовое преобразование

```
'2' > 1; // true
'01' == 1; // true
true == 1; // true
false == ''; // true
```

Строгое сравнение

Сравнение без приведения типов.
Всегда используйте строгое сравнение!

```
'1' == 1; // true
'1' === 1; // false
true == 1; // true
true === 1; // false
false == ''; // true
false === ''; // false
```

Логические операторы

```
// НЕ | NOT
!a;

// И | AND
a && b;

// ИЛИ | OR
a || b;
```

Truthy / falsy

```
!0 && 'foo' || !true

!0 // true
'foo' // true
!true // false
true && true || false // true
```

Эквивалентно false

false

0

-0

"" (пустая строка)

null

undefined

NaN

Короткий цикл вычислений: &&

```
first && second
```

Если first истинно, результатом операции будет second
Иначе результатом будет first

Короткий цикл вычислений: ||

```
first || second
```

Если first истинно, результатом операции будет first
Иначе результатом будет second

Короткий цикл вычислений

```
42 || 'сорок два'; // 42

false || 42 || veryComplexMath(14236487); // 42

42 && 'что-нибудь' && null; // null
```

ПРЕОБРАЗОВАНИЕ ТИПОВ

Преобразование типов

```
"10" + " попугаев"
```

http://learn.javascript.ru/types-conversion

Преобразование типов

"10 попугаев"

Преобразование типов

```
10 + " попугаев"
```

Преобразование типов

"10 попугаев"

Преобразование типов

```
"10" + 1
```

Преобразование типов

```
"101"
```

Преобразование типов

```
"10" - 1
```

Преобразование типов

```
9
```

Преобразование типов

```
"10попугаев" - 1
```

Преобразование типов

```
NaN
```

Преобразование типов

```
true + "false"
```

Преобразование типов

```
"truefalse"
```

Преобразование типов

```
let x;
"123" + x;
```

Преобразование типов

"123undefined"

Преобразование типов

```
Number(123)
```

Преобразование типов

```
123
```

Преобразование типов

```
+"123"
```

Преобразование типов

```
123
```

Преобразование типов

```
0 + undefined
```

Преобразование типов

```
NaN
```

Преобразование типов

```
null + 0
```

Преобразование типов

```
0
```

Преобразование типов

```
true + 1
```

Преобразование типов

```
2
```

Преобразование типов

1 - **false**

Преобразование типов

```
1
```

Преобразование типов

```
+"  \n  123  \n  \n"
```

Преобразование типов

```
123
```

Преобразование типов

```
+null
```

Преобразование типов

```
0
```

Преобразование типов

```
null >= 0
```

Преобразование типов

```
null >= 0 // true
```

Преобразование типов

```
null >= 0 // true
null > 0
```

Преобразование типов

```
null >= 0 // true
null > 0 // false
```

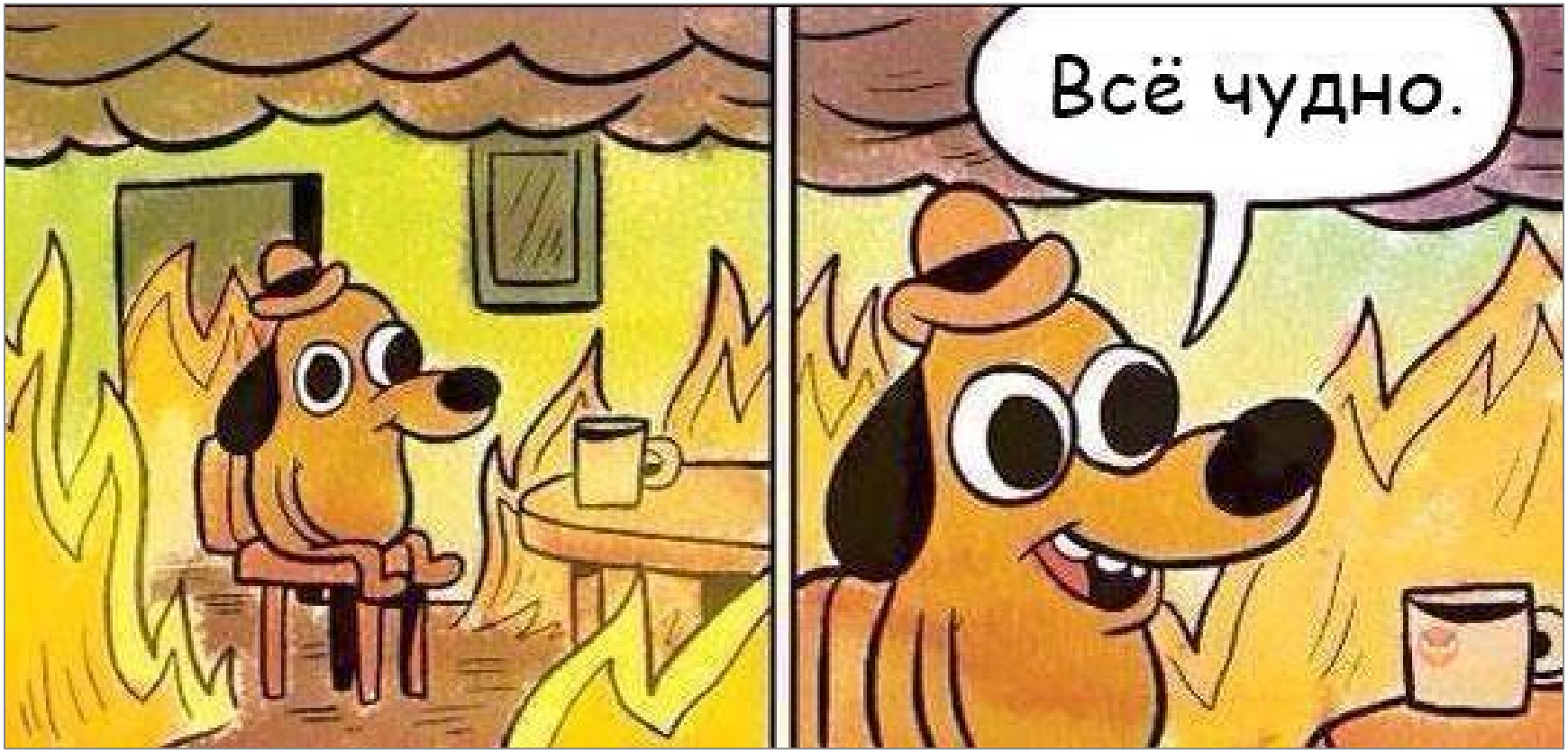
Преобразование типов

```
null >= 0 // true
null > 0 // false
null == 0
```

Преобразование типов

```
null >= 0 // true
null > 0 // false
null == 0 // false
```

Преобразование типов



Преобразование типов

```
!true // false
```

Преобразование типов

```
!true // false
!!true // true
```

Преобразование типов

```
!123
```

Преобразование типов

```
!123 // false
```

Преобразование типов

```
!123 // false
!!123
```

Преобразование типов

```
!123 // false
!!123 // true
```

Преобразование типов

```
!!"0"
```

Преобразование типов

```
!!"0" // true
```

Преобразование типов

```
!!"0" // true
!!" "
```

Преобразование типов

```
!!"0" // true
!!" " // true
```

Преобразование типов

```
!!"0" // true
!!" " // true
!!""
```

Преобразование типов

```
!!"0" // true
!!" " // true
!!"" // false
```

УПРАВЛЯЮЩИЕ КОНСТРУКЦИИ

if, if … else, ?:

```
if (2 > 5) {
   // ...
} else {
   // else не обязателен
}

var x = (2 > 5) ? 'yep' : 'nope';
```

While

```
var i = 0;
while (i < 3) {
   console.log(i);
   i++;
}
```

https://learn.javascript.ru/while-for#цикл-while

For

```
var i;

for (i = 0; i < 3; i++) {
   console.log(i);
}
```

https://learn.javascript.ru/while-for#цикл-for

For

```
for (var i = 0; i < 3; i++) {
   console.log(i);
}

console.log(i); // 3
```

For

```
for (var i = 0; i < 3; i++) {
   console.log(i);
}

console.log(i); // 3
```

ES6

```
for (let i = 0; i < 3; i++) {
   console.log(i);
}

console.log(i); // Error
```

Switch

```
switch(x) {
  case 'value1':  // if (x === 'value1')
  // ...
  break;

  case 'value2':  // if (x === 'value2')
  // ...
  break;

  default:
  // ...
  break;
}
```

https://learn.javascript.ru/switch

Что почитать

JavaScript: от начала до конца
Mozilla Developer Network

Дайджесты, подкасты:
Подкаст Веб-стандартов
Radio JS
Дайджест Хабра
JavaScript Weekly
FrontEnd Focus

Книги:
Илья Кантор «Современный учебник Javascript»
Marijn Haverbeke «Выразительный JavaScript» (ссылка, вторая)
Дэвид Флэнаган «JavaScript. Подробное руководство», 6-е издание

ФУНКЦИИ

Простая функция

```
function simple() {
  console.log('Ghbdtn!');
}

simple(); // Ghbdtn!
```

https://learn.javascript.ru/function-basics

Аргументы функции

```
function hello(name) {
  console.log('Здравствуй, ' + name + '!');
}

hello('Вася'); // Здравствуй, Вася!
```

Аргументы функции

```
function hello(name) {
  console.log('Здравствуй, ' + name + '!');
}

hello('Вася'); // Здравствуй, Вася!
hello();
```

Аргументы функции

```
function hello(name) {
  console.log('Здравствуй, ' + name + '!');
}

hello('Вася'); // Здравствуй, Вася!
hello(); // Здравствуй, undefined!
```

Аргументы функции

```
function hello(name) {
  console.log('Здравствуй, ' + name + '!');
}

hello('Вася'); // Здравствуй, Вася!
hello(); // Здравствуй, undefined!
hello('Вася', 'Миша'); // Здравствуй, Вася!
```

Видимость

```
hello('Вася'); // Здравствуй, Вася!

function hello(name) {
   console.log('Здравствуй, ' + name + '!');
}
```

Return

```
function helloReturned(name) {
  // ... тут ваш клёвый код
  return 'Здравствуй, ' + name + '!';

}

console.log(helloReturned('Вася')); // Здравствуй, Вася!
```

Return

```
function helloReturned(name) {
  // ... тут ваш клёвый код
  return 'Здравствуй, ' + name + '!';
  console.log('Эта фраза не покажется');
}

console.log(helloReturned('Вася')); // Здравствуй, Вася!
```

Анонимная функция

```
function (name) {
  return 'Здравствуй, ' + name + '!';
};
```

Функциональное выражение

```
var helloExpression = function (name) {
   return 'Здравствуй, ' + name + '!';
};
```

Функциональное выражение

```
console.log(helloExpression('Вася')); // ошибка

var helloExpression = function (name) {
   return 'Здравствуй, ' + name + '!';
};
```

Стрелочная функция

ES6

```
const helloExpression = (name) => {
   return 'Здравствуй, ' + name + '!';
};
```

Стрелочная функция

ES6

```
const helloExpression = (name) => 'Здравствуй, ' + name + '!';
```

Пример посложнее

```
function helloOrBye(name, isBye) {
   if (isBye) {
      console.log('Чао, дорогой ' + name + '!');
   } else {
      console.log('Здравствуй, ' + name + '!');
   }
}
```

Пример посложнее

```
function helloOrBye(name, isBye) {
   if (isBye) {
      console.log('Чао, дорогой ' + name + '!');
   } else {
      console.log('Здравствуй, ' + name + '!');
   }
}

helloOrBye('Вася', true); // Чао, дорогой Вася!
helloOrBye('Вася', false); // Здравствуй, Вася!
helloOrBye('Вася'); // Здравствуй, Вася!
```

Множественный return

```
function helloOrBye(name, isBye) {
   if (isBye) {
      return 'Чао, дорогой ' + name + '!';
   }
   return 'Здравствуй, ' + name + '!';
}
```

Тернарный if

```
function helloOrBye(name, isBye) {
  return isBye ? 'Чао, дорогой ' + name + '!' : 'Здравствуй, ' + name + '!';
}
```

Тернарный if

```
function helloOrBye(name, isBye) {
  return (isBye ? 'Чао, дорогой ' : 'Здравствуй, ') + name + '!';
}
```

Fat arrow

ES6

```
const helloOrBye = (name, isBye) => {
  return (isBye ? 'Чао, дорогой ' : 'Здравствуй, ') + name + '!';
};
```

Fat arrow

ES6

```
const helloOrBye = (name, isBye) => (isBye ? 'Чао, дорогой ' : 'Здравствуй, ') + name + '!';
```

Template string

ES6

```
const helloOrBye = (name, isBye) => `${isBye ? 'Чао, дорогой' : 'Здравствуй,'} ${name}!`;
```

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/template_strings

ЗАМЫКАНИЕ, ОБЛАСТИ ВИДИМОСТИ

Локальные переменные

Локальные переменные

```
function billy() {
  var map = 'карта сокровищ';
  return '– У меня нет никакой карты!';
}
```

Локальные переменные

```
function billy() {
  var map = 'карта сокровищ';
  return '– У меня нет никакой карты!';
}
console.log('– Где карта, Билли?');
console.log(billy());
```

Локальные переменные

```
function billy() {
  var map = 'карта сокровищ';
  return '– У меня нет никакой карты!';
}
console.log('– Где карта, Билли?');
console.log(billy()); // – У меня нет никакой карты!
```

Локальные переменные

```
function billy() {
  var map = 'карта сокровищ';
  return '– У меня нет никакой карты!';
}
console.log('– Где карта, Билли?');
console.log(billy()); // – У меня нет никакой карты!
console.log(map); // Error
```

Локальные переменные

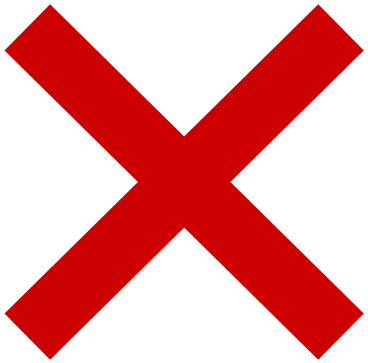
```
function billy() {
  var map = 'карта сокровищ';
  console.log('*взляд на сундук, где лежит ' + map + '*');
  return '– У меня нет никакой карты!';
}

console.log('– Где карта, Билли?');
console.log(billy());
// *взляд на сундук, где лежит карта сокровищ*
// – У меня нет никакой карты!
console.log(map); // Error
```

Локальные переменные

```
function billy() {
  var map = 'карта сокровищ';
  lookup();
  return '– У меня нет никакой карты!';

  function lookup() {
    console.log('*взляд на сундук, где лежит ' + map + '*');
  }
}
```

Локальные переменные

```
function billy() {
  var map = 'карта сокровищ';
  lookup();
  return '– У меня нет никакой карты!';
}

function lookup() {
  console.log('*взляд на сундук, где лежит ' + map + '*');
}
```



ВНЕШНИЕ ПЕРЕМЕННЫЕ

Область видимости

```
var name = 'Вася';
function whoAreYou() {
  var name = 'Вова';
  console.log(name);
}
whoAreYou(); // Вова
console.log(name); // Вася
```

Область видимости

```
var name = 'Вася';
function whoAreYou() {
   name = 'Вова';
   console.log(name);
}
whoAreYou(); // Вова
console.log(name); // 'Вова'
```

Область видимости

```
var name = 'Вася';
function whoAreYou() {
  name = 'Вова';
  console.log(name);
}
whoAreYou(); // Вова
console.log(name); // 'Вова'
```

```
var name = 'Вася';
function whoAreYou() {
  var name = 'Вова';
  console.log(name);
}
whoAreYou(); // Вова
console.log(name); // Вася
```

Замыкание

```
function createCounter () {
  var value = 0;

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

http://learn.javascript.ru/closures

Замыкание

```
function createCounter () {
  var value = 0;

  return function () {
     return value++;
  }
}
var counter = createCounter();
counter(); // ?
```

Замыкание

```
function createCounter () {
   var value = 0;

   return function () {
      return value++;
   }
}
var counter = createCounter();
counter(); // 0
```

Замыкание

```
function createCounter () {
  var value = 0;

  return function () {
     return value++;
  }
}
var counter = createCounter();
counter(); // 0
counter(); // ?
```

Замыкание

```
function createCounter () {
  var value = 0;

  return function () {
     return value++;
  }
}
var counter = createCounter();
counter(); // 0
counter(); // 1
```

Замыкание

```
function createCounter () {
  var value = 0;

  return function () {
     return ++value;
  }
}
var counter = createCounter();
counter(); // 1
counter(); // 2
```

Замыкание

```
var value = 5;

function createCounter () {
   var value = 0;

   return function () {
      return ++value;
   }
}
var counter = createCounter();
counter(); // ?
```

Замыкание

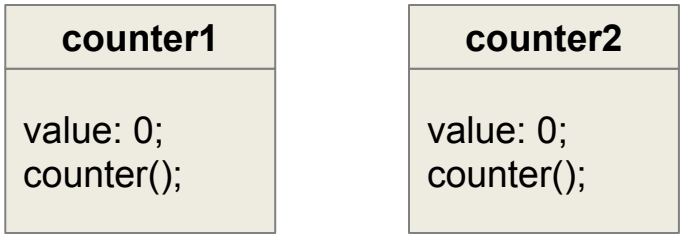
```
var value = 5;

function createCounter () {
   var value = 0;

   return function () {
      return ++value;
   }
}
var counter = createCounter();
counter(); // 1
```

Замыкание

```
function createCounter () {
   var value = 0;

   return function () {
      return ++value;
   }
}
var counter1 = createCounter();
var counter2 = createCounter();
counter1(); // 1
counter1(); // 2
counter2(); // 1
```

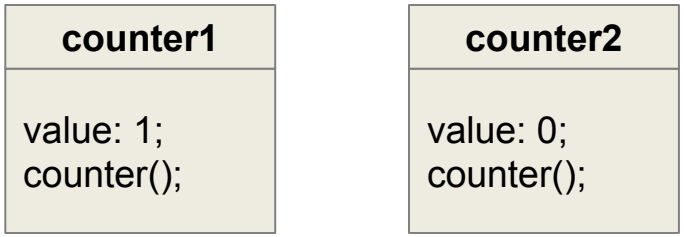
Замыкание

```
function createCounter () {
   var value = 0;

   return function () {
      return ++value;
   }
}
var counter1 = createCounter();
var counter2 = createCounter();
```



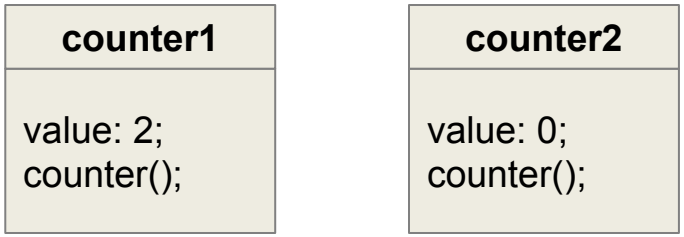
Замыкание

```
function createCounter () {
   var value = 0;

   return function () {
      return ++value;
   }
}
var counter1 = createCounter();
var counter2 = createCounter();
counter1(); // 1
```



Замыкание

```
function createCounter () {
   var value = 0;

   return function () {
      return ++value;
   }
}
var counter1 = createCounter();
var counter2 = createCounter();
counter1(); // 1
counter1(); // 2
```



Замыкание

```
function createCounter () {
   var value = 0;

   return function () {
      return ++value;
   }
}
var counter = createCounter();
var counter2 = createCounter();
counter1(); // 1
counter1(); // 2
counter2(); // 1
```

counter1
value: 2; counter();

counter2
value: 1; counter();

Замыкание

```
function createCounter () {
  var value = 0;

  return {
      next: function () {
          return ++value;
      },
      set: function (newValue) {
          value = newValue;
      },
      reset: function () {
        value = 0;
      }
  }
}
```

Замыкание

```
function createCounter () {
  var value = 0;

  return {
    next: function () {
      return ++value;
    },
    set: function (newValue) {
      value = newValue;
    },
    reset: function () {
     value = 0;
    }
  }
}

var counter = createCounter();
counter.next(); // 1
counter.next(); // 2
counter.set(5);
counter.next(); // 6
```

Особенности замыканий

```
for (var i = 0; i < 10; i++) {
   setTimeout(function(){
      console.log(i);
   }, 1000);
}
```

Особенности замыканий

```
for (var i = 0; i < 10; i++) {
   setTimeout(function(){
      console.log(i);
   }, 1000);
}

//(10) 10
```

Особенности замыканий

```
i = 0;
setTimeout(function(){
    console.log(i);
}, 1000);
```

1. Исполнить **fn** через *1000мс*

Особенности замыканий

```
i = 1;
setTimeout(function(){
    console.log(i);
}, 1000);
```

1. Исполнить **fn** через *1000мс.*
2. Исполнить **fn** через *1000мс.*

Особенности замыканий

```
i = 9;
setTimeout(function(){
    console.log(i);
}, 1000);
```

1. Исполнить **fn** через *1000мс.*
2. Исполнить **fn** через *1000мс.*

…

10. Исполнить **fn** через *1000мс.*

Особенности замыканий

```
i = 9;
setTimeout(function(){
    console.log(i);
}, 1000);
```

1. Исполнить **fn** через *1000мс.*
2. Исполнить **fn** через *1000мс.*

…

10. Исполнить **fn** через *1000мс.*

Особенности замыканий

```
i = 10;
/* цикл завершен */
```

1. Исполнить **fn** через *1000мс.*
2. Исполнить **fn** через *1000мс.*

…

10. Исполнить **fn** через *1000мс.*

…

Особенности замыканий

```
i = 10;
/* цикл завершен */
```

1. Исполнить **fn** через *1000мс.*
2. Исполнить **fn** через *1000мс.*

…

10. Исполнить **fn** через *1000мс.*

…

прошла 1000мс

...

Особенности замыканий

```
i = 10;
/* цикл завершен */
function(){
    console.log(i);
}
```

1. Исполнить **fn** через *1000мс.*
2. Исполнить **fn** через *1000мс.*

…

10. Исполнить **fn** через *1000мс.*

…

прошла 1000мс

…

честно выполняем 10 функий

Особенности замыканий

```
i = 10;
/* цикл завершен */
function(){
    console.log(i);
}
console.log(i); // 10
```

1. ~~Исполнить **fn** через *1000мс.*~~
2. Исполнить **fn** через *1000мс.*

…

10. Исполнить **fn** через *1000мс.*

…

прошла 1000мс

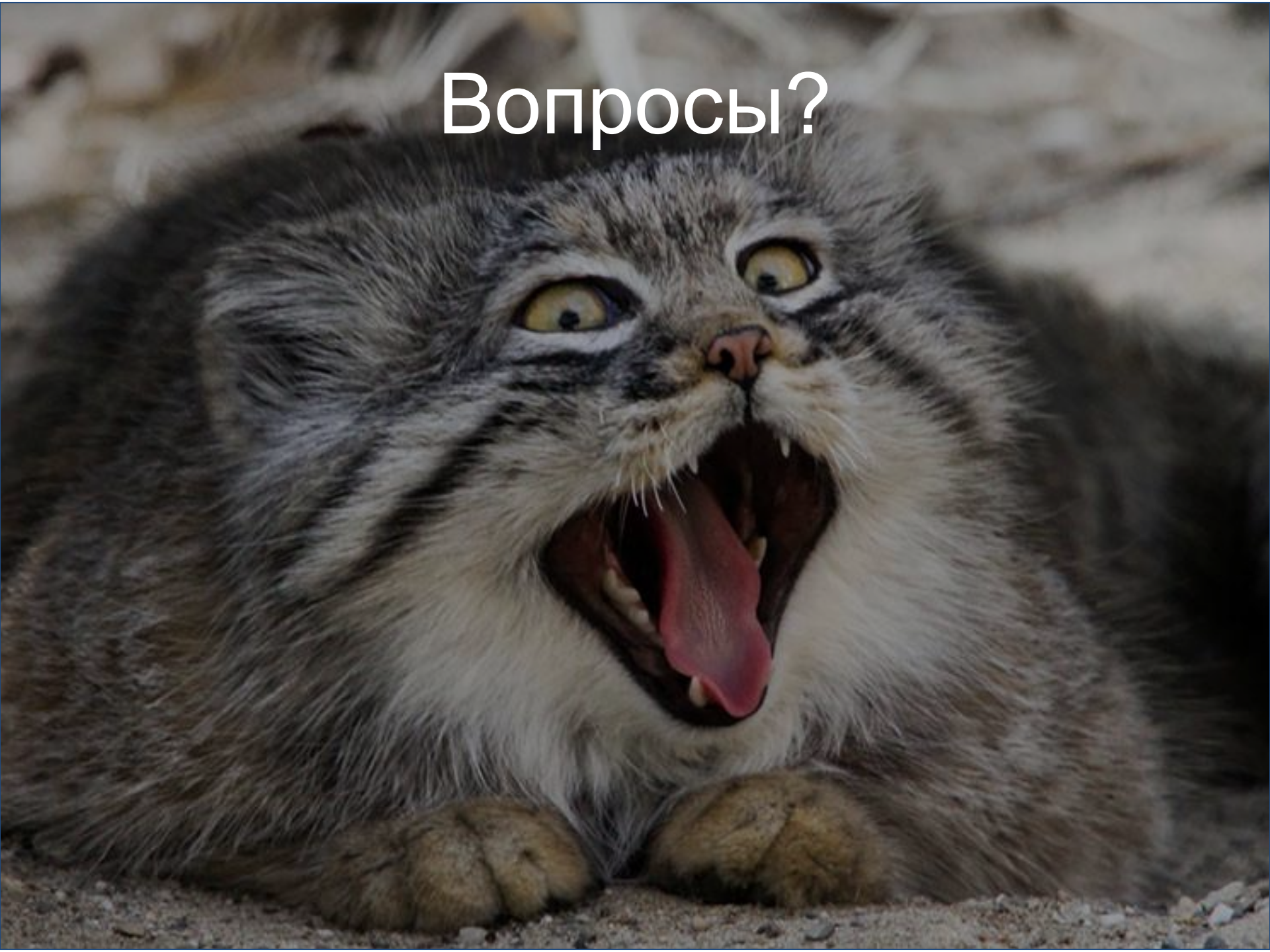
…

честно выполняем 10 функий

Особенности замыканий

```
i = 10;
/* цикл завершен */
function(){
    console.log(i);
}
console.log(i); // 10
console.log(i); // 10
```

~~1. Исполнить **fn** через *1000мс.*~~
~~2. Исполнить **fn** через *1000мс.*~~
…
10. Исполнить **fn** через *1000мс.*
…
прошла 1000мс
…
честно выполняем 10 функий


Вопросы?