

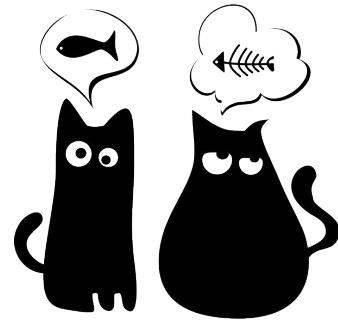
# Java*Script* и this

Репортаж



ХОЧЕШЬ ЗНАТЬ  
ЧЕМУ this  
РАВНЯЕТСЯ  
ПРОВЕРЬ КАК  
ТВОЯ ФУНКЦИЯ  
РАБОТАЕТСЯ

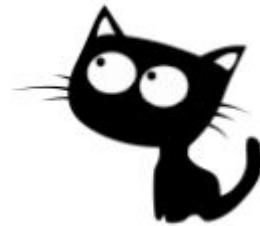
ХОЧЕШЬ ЗНАТЬ  
ЧЕМУ `this` РАВНЯЕТСЯ



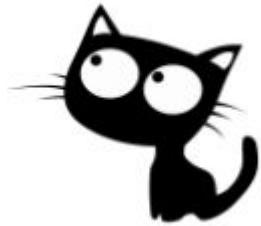
ПРОВЕРЬ ...



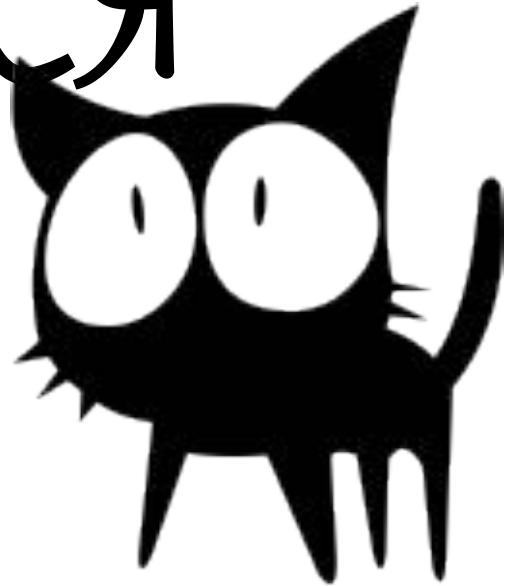
КАК  
И  
КАКАЯ



# ФУНКЦИЯ

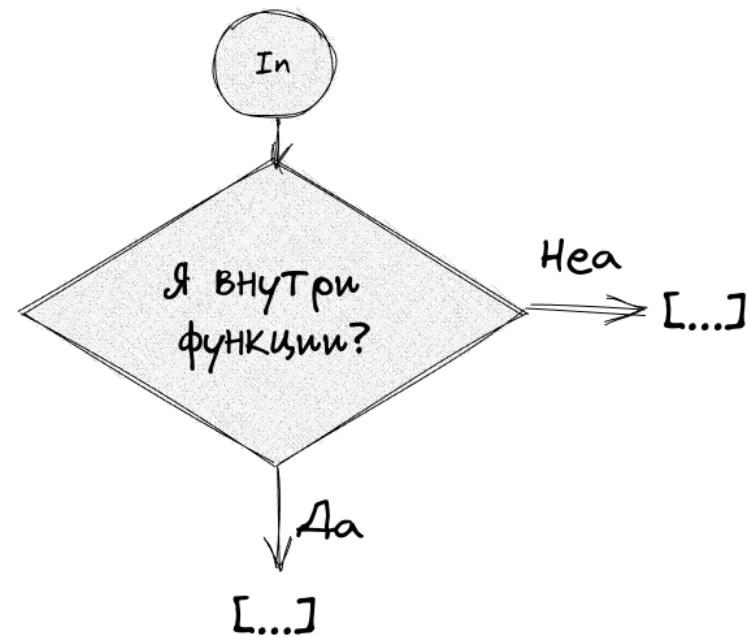
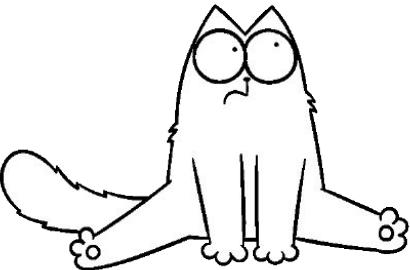


# ЗАПУСКАЕТСЯ



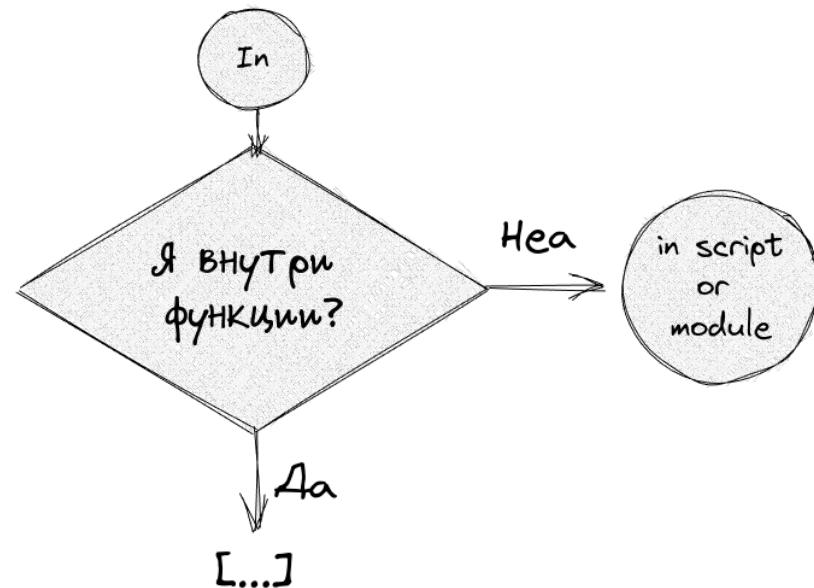
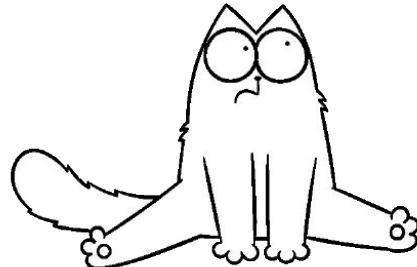
# this и Global Environment

```
1 "use strict";  
2 console.log( "this is: ", this );
```



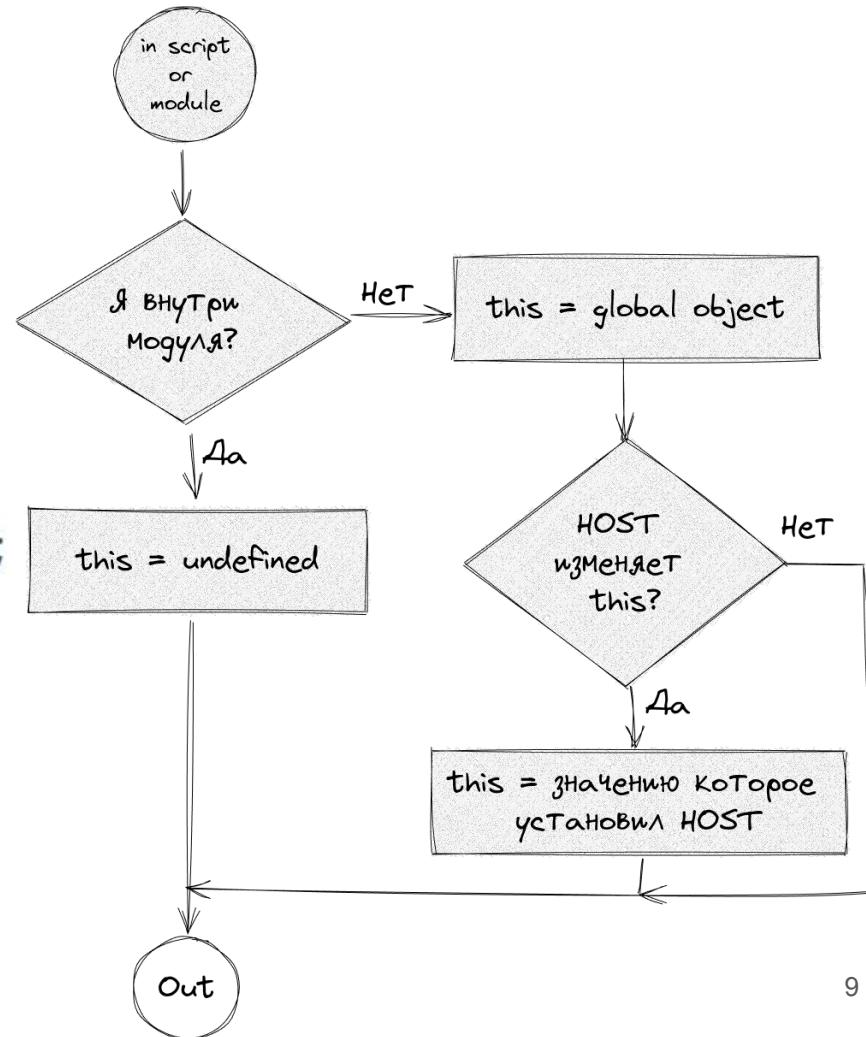
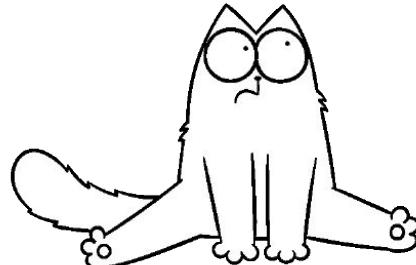
# this и Global Environment

```
1 "use strict";
2 console.log( "this is: ", this );
```



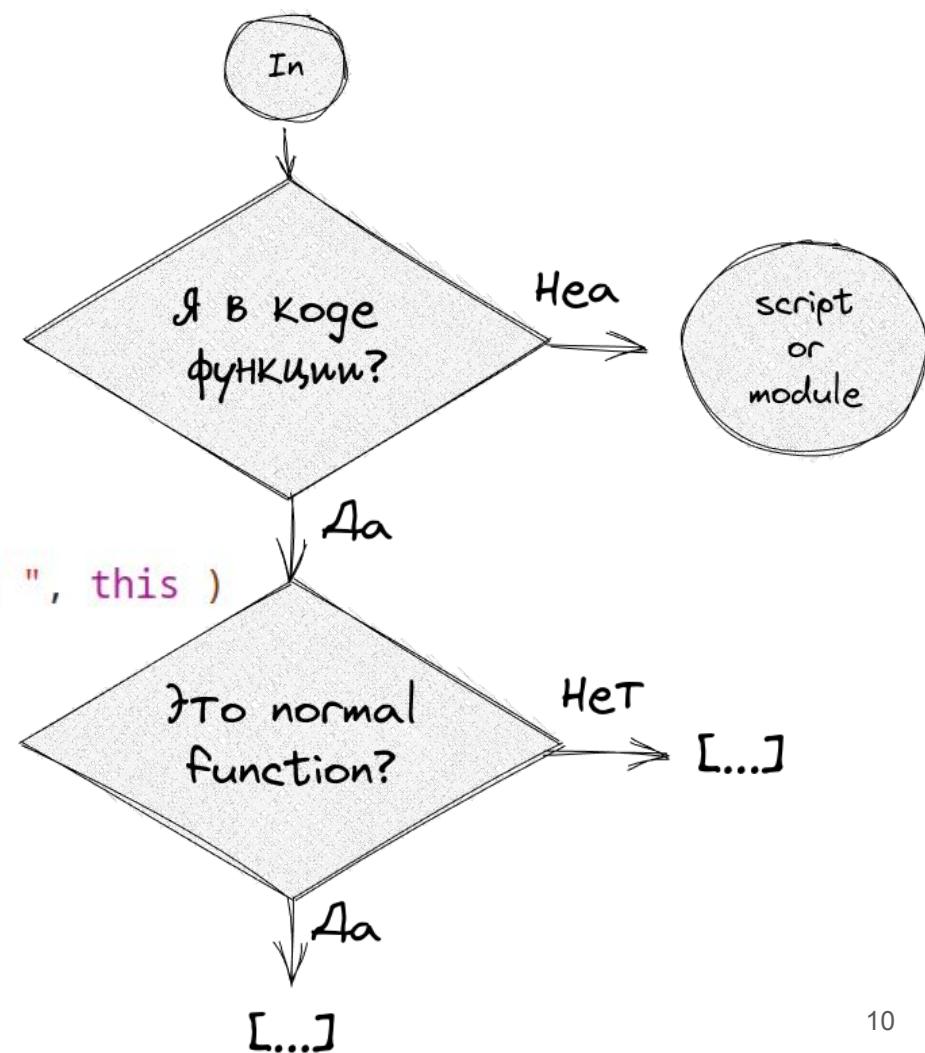
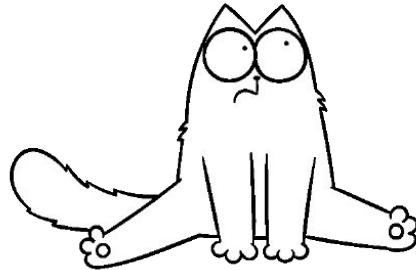
# script или module

```
1 "use strict";
2 console.log( "this is: ", this );
```



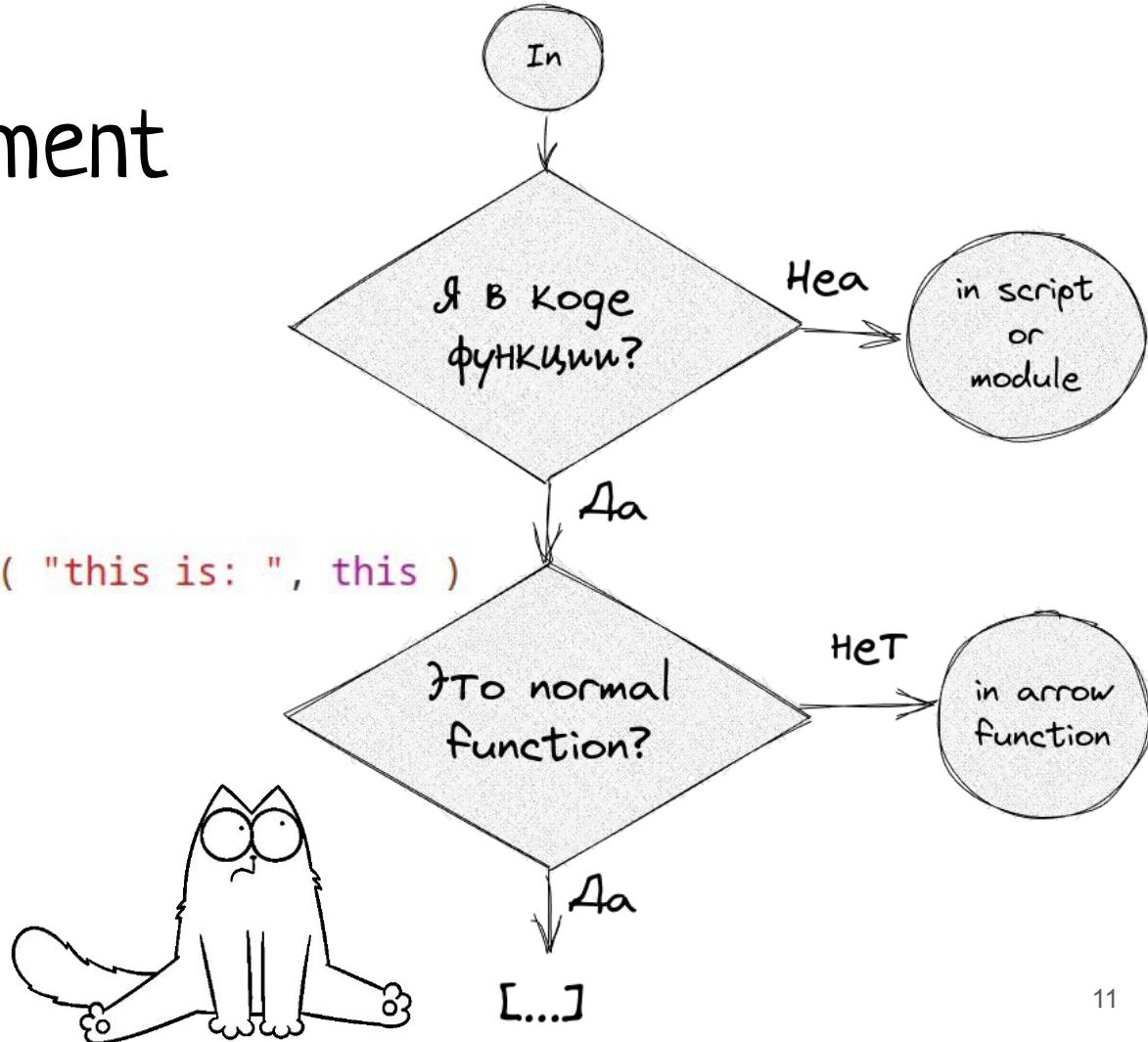
# Function Environment

```
1 "use strict";
2 function doLogThis() {
3     var doArrowThing = (
4         () => console.log( "this is: ", this )
5     );
6     doArrowThing();
7 }
8 doLogThis();
```



# Function Environment

```
1 "use strict";
2 function doLogThis() {
3     var doArrowThing = (
4         () => console.log( "this is: ", this )
5     );
6     doArrowThing();
7 }
8 doLogThis();
```



# Arrow Function Environment

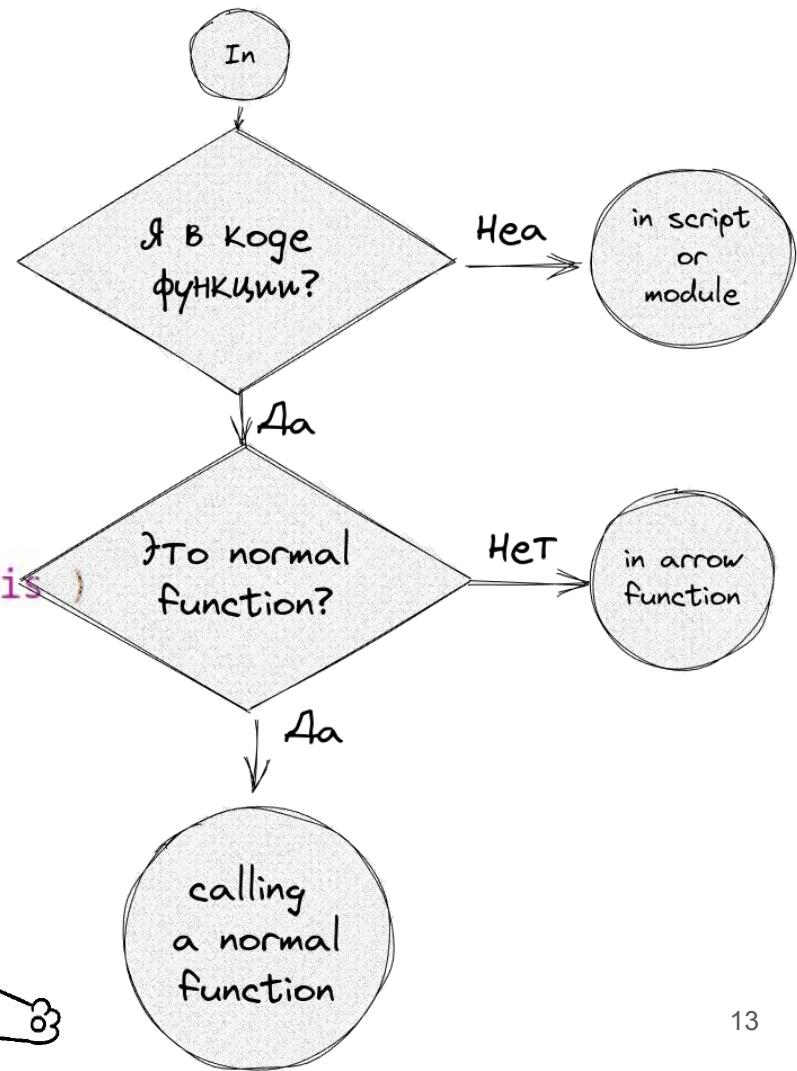
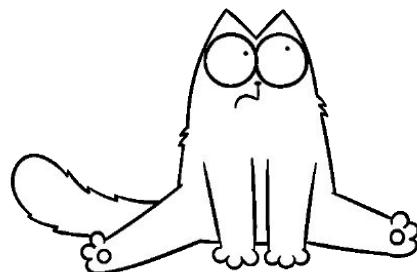
```
1 "use strict";
2 function doLogThis() {
3     var doArrowThing = (
4         () => console.log( "this is: ", this )
5     );
6     doArrowThing();
7 }
8 doLogThis();
```



# Пример 2.2

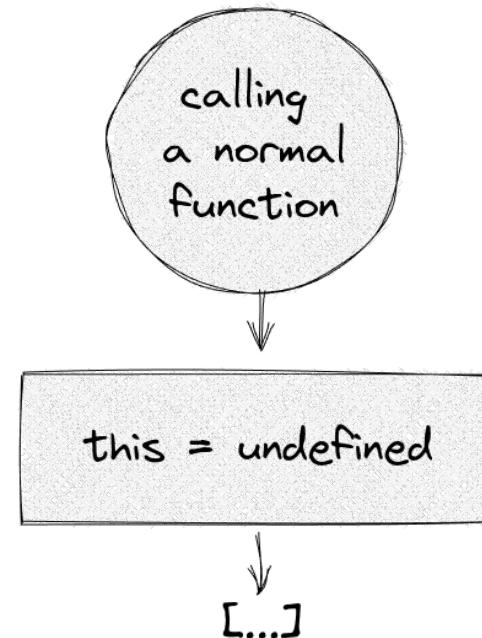
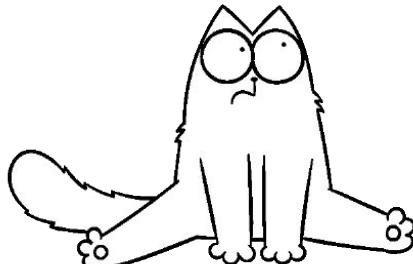
## Function Environment

```
1 "use strict";
2 function doLogThis() {
3     var doArrowThing = (
4         () => console.log( "this is: ", this )
5     );
6     doArrowThing();
7 }
8 doLogThis();
```



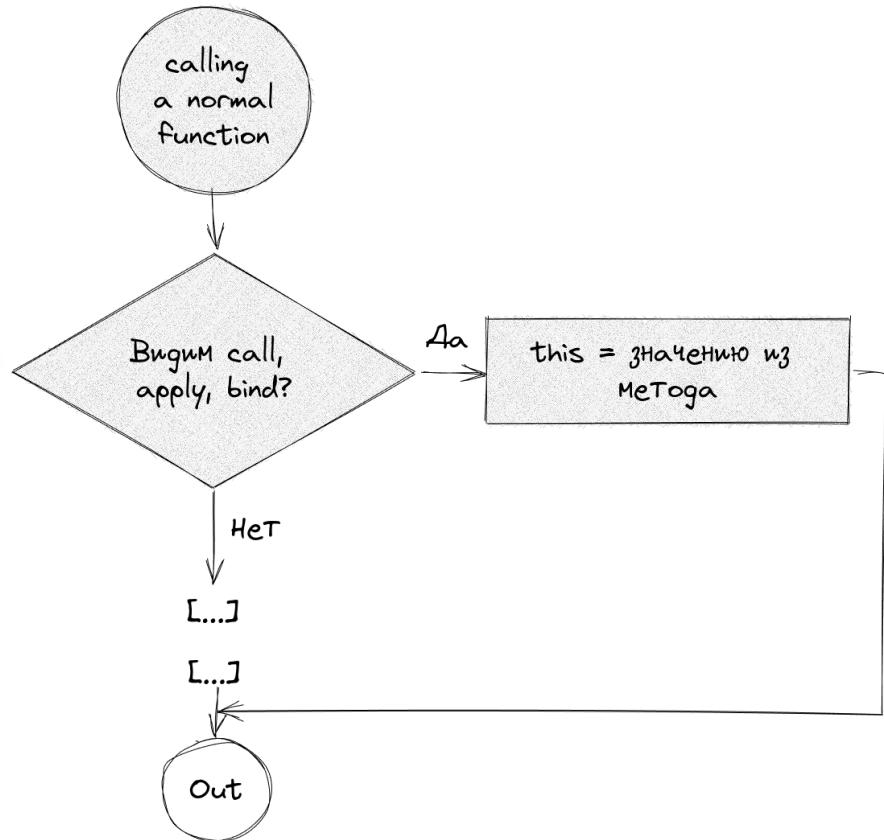
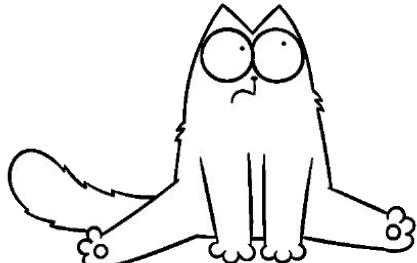
# this и вызов функции

```
1 "use strict";
2 function doLogThis() {
3     console.log( "this is: ", this );
4 }
5 doLogThis();
```



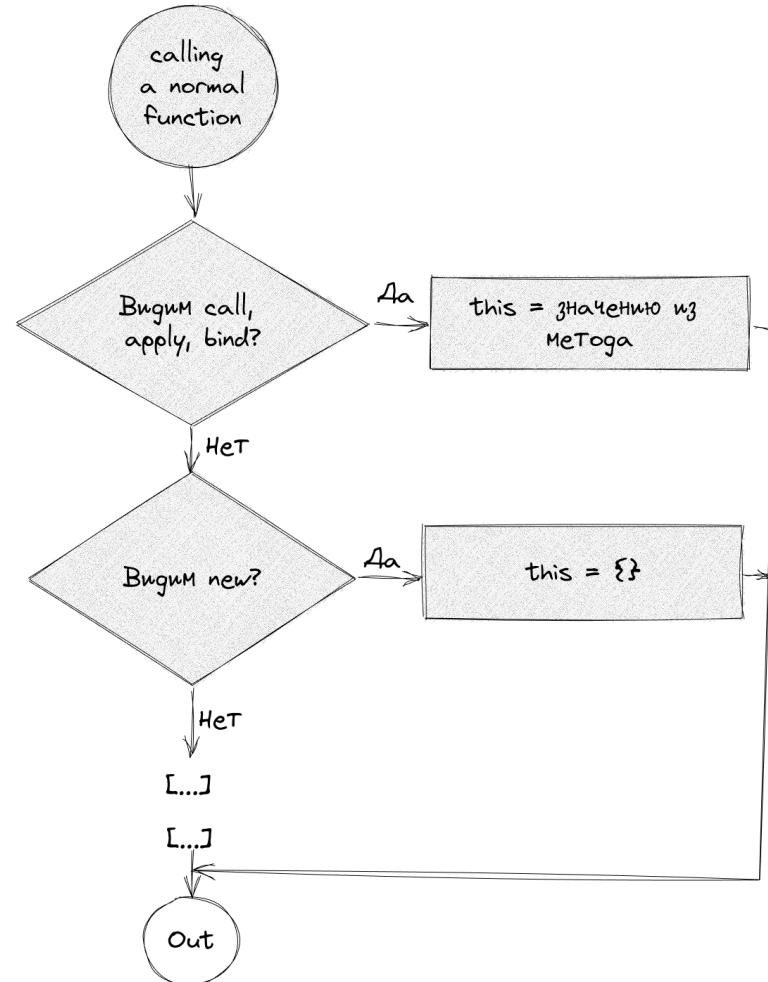
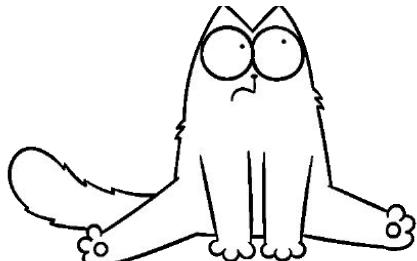
# this и call, apply, bind

```
1 "use strict";
2 function doLogThis() {
3     console.log( "this is: ", this );
4 }
5
6 var thisArg = { name: "thisArg" };
7 doLogThis.call( thisArg );
8 doLogThis.apply( thisArg );
9 doLogThis.bind( thisArg )();
```



# this и new

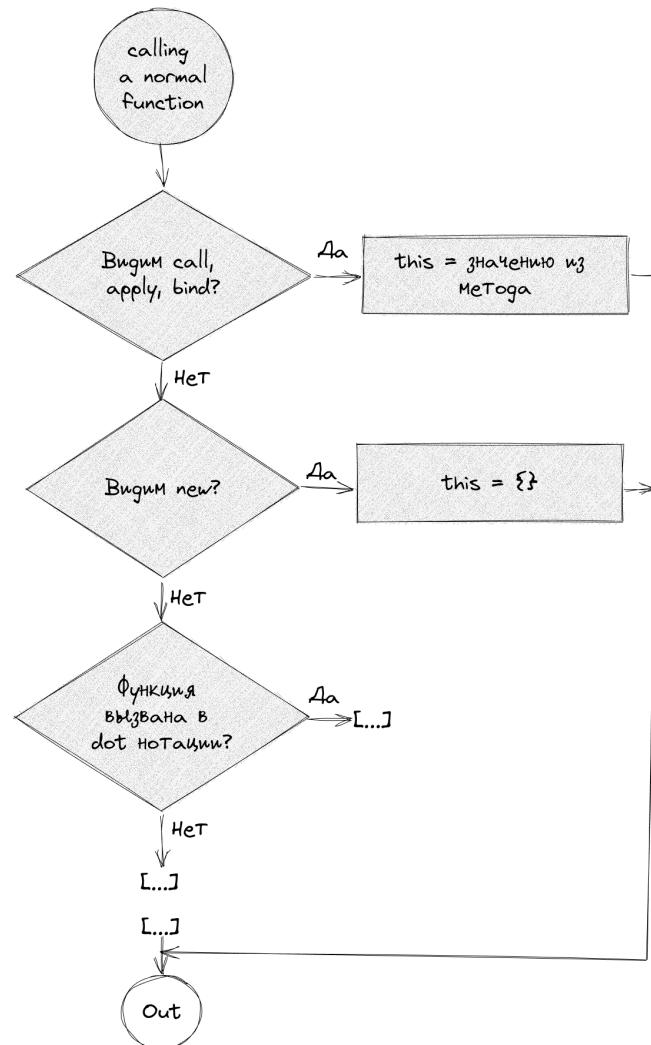
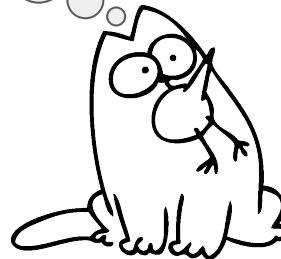
```
1 "use strict";
2 function doLogThis() {
3     console.log( "this is: ", this );
4 }
5
6 new doLogThis();
7 new doLogThis;
```



# this и дот нотация

```
1 "use strict";
2 function doLogThis() {
3     console.log( "this is: ", this );
4 }
5 doLogThis();
```

Че за врот  
нотация?



# Что такое dot нотация?

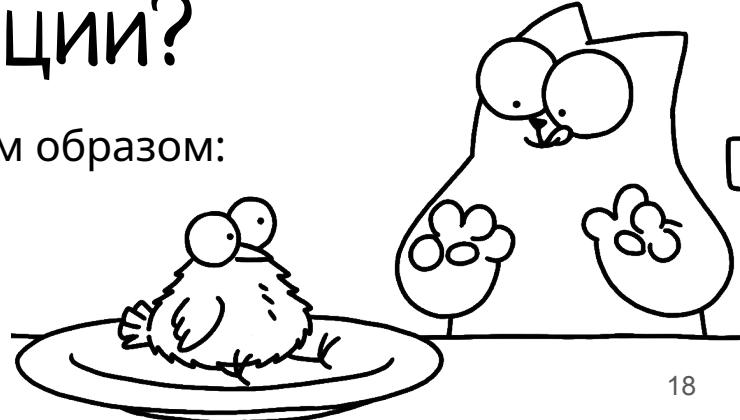
Дот нотацией (dot notation) в JavaScript, называют синтаксис, когда два идентификатора разделены между собой точкой (dot). Например:  
theObj . theProperty ;

Его полным аналогом является синтаксис: theObj [ "theProperty" ] ;  
<https://tc39.es/ecma262/#sec-property-accessors>

# Вызов функции в dot нотации?

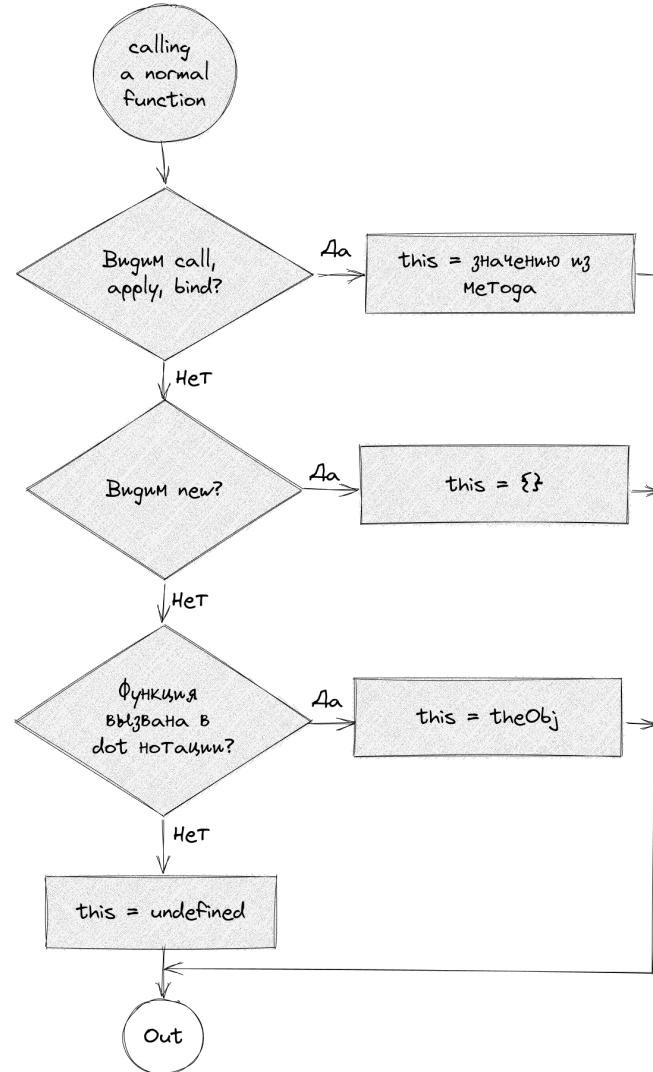
Вызов функции в dot нотации выглядит следующим образом:

```
theObj . doThing () ;  
theObj [ "doThing" ] () ;
```



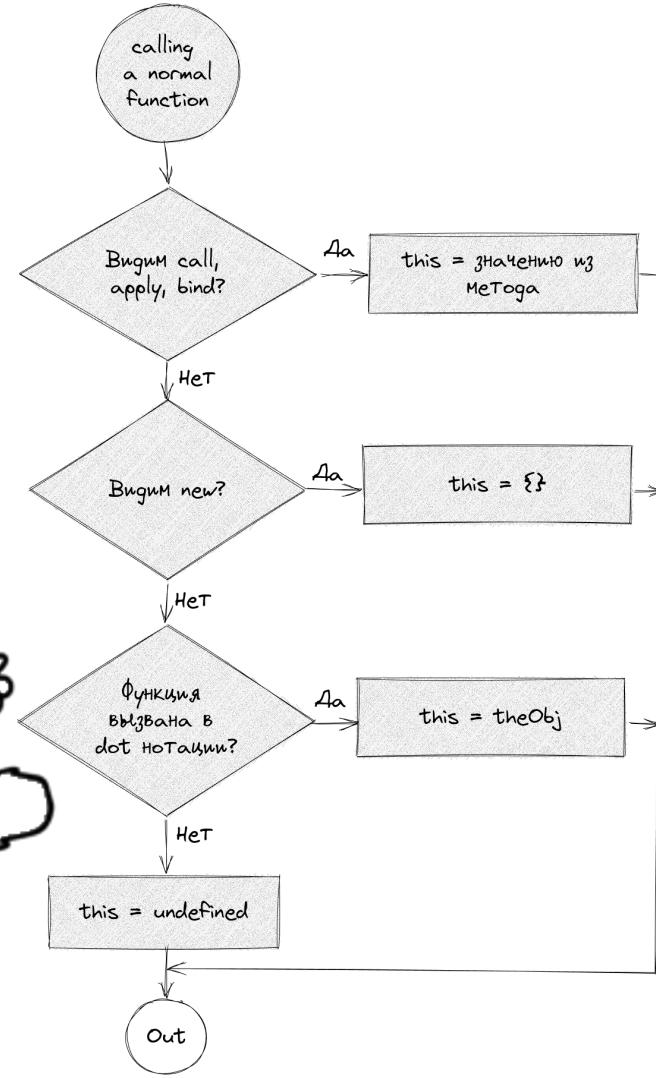
# this и дот нотация

```
1 "use strict";
2 function doLogThis() {
3   console.log( "this is: ", this );
4 }
5 doLogThis();
```



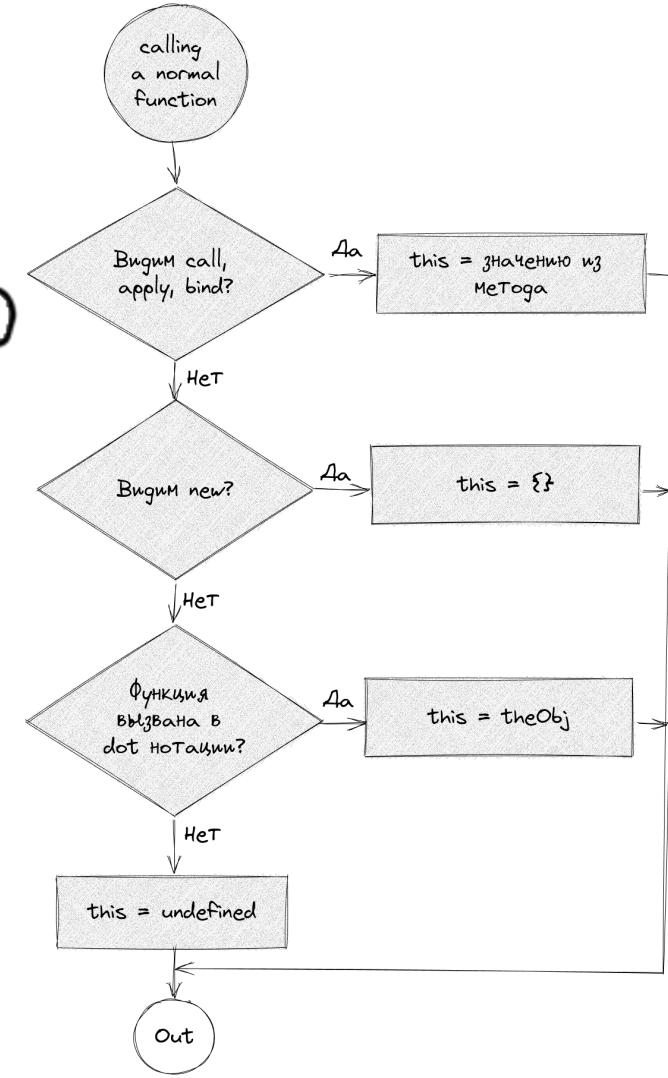
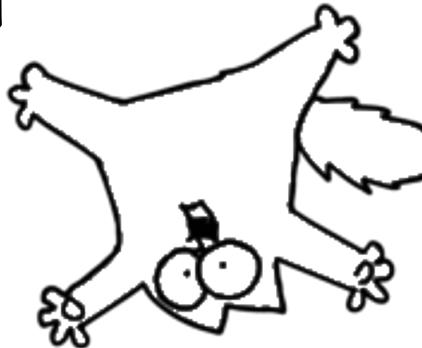
# this и дот нотация

```
1 "use strict";  
2  
3 function doLogThis() {  
4     console.log( "this is: ", this );  
5 }  
6  
7 const theObj = {  
8     name: "Murych"  
9 };  
10 theObj.doLogThis = doLogThis;  
11  
12 theObj.doLogThis();
```



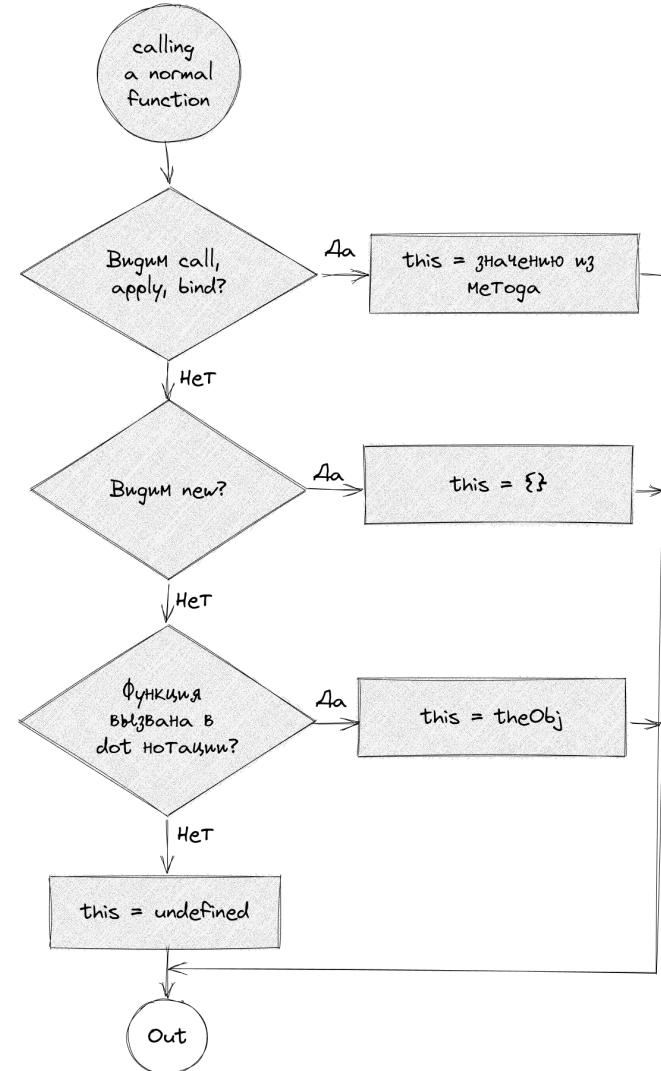
# Пример наоборот

```
1 "use strict";
2
3 const theObj = {
4     name: "Murych",
5     doLogThis: function () {
6         console.log( "this is: ", this );
7     }
8 };
9
10 var doLogThisGlobal = theObj.doLogThis;
11 doLogThisGlobal();
```



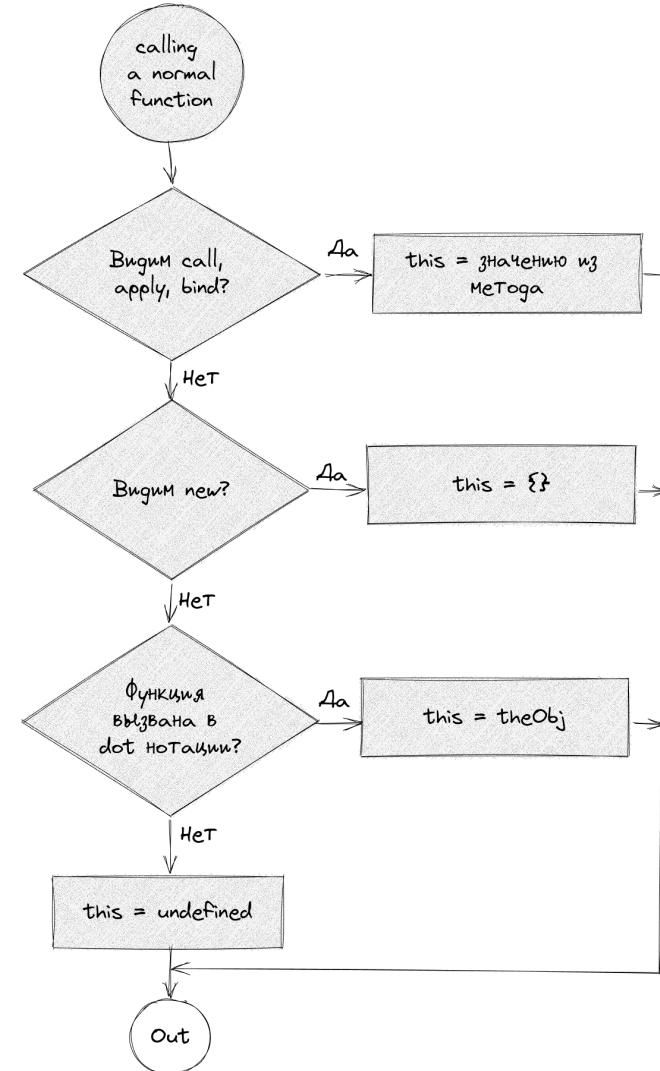
# Пример наоборот - 2

```
1 "use strict";
2
3 const theObj = {
4   name: "Murych",
5   doLogThis: function () {
6     console.log( "this is: ", this );
7   }
8 };
9
10 setTimeout( theObj.doLogThis, 1 );
```



# Пример наоборот - 3

```
1 "use strict";
2
3 const theObj = {
4   name: "Murych",
5   doLogThis: function () {
6     console.log( "this is: ", this );
7   }
8 };
9
10 var doLogThis = theObj.doLogThis;
11 setTimeout( doLogThis, 1 );
```



# API call

```
1 "use strict";  
2  
3 function doHandleClick() {  
4     console.log( "this is:", this )  
5 }  
6  
7 document  
8     .body  
9     .addEventListener( "click", doHandleClick );
```



# Что такое this?

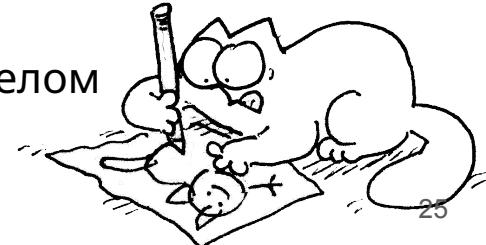
this - это выражение языка JavaScript, поведение которого очень похоже на поведение идентификатора, с той лишь разницей, что связать значение с this, мы можем только особой формой вызова normal function.



# Что такое normal function?

Поскольку способов определить normal function большое множество, проще всего о них думать как о всех функциях, которые отличаются от arrow function.

То есть это все функции, у которых между аргументами и их телом функции отсутствует символьная пара =>



# Какие формы вызова меняют this?

1. Вызов любой normal function, по умолчанию, связывает this с undefined
2. Ключевое слово new, указанное перед вызовом функции, например new doThing(); связет this с normal object без заданных property
3. Используя call, apply или bind - мы можем задать this явным образом

```
doThing.call( thisArg, ... args);  
doThing.apply( thisArg, ... args);  
doThing.bind( thisArg, ... args)();
```
4. И в случае вызова функции в dot нотации, this будет связан со значением идентификатора, который стоит перед точкой:  
`theObj.doThing();`



# this и API

Вызов любого внешнего API может связывать this со значением, которое пришло на ум создателям API.

В любой нотации и формах.

Как реализовано конкретное API, можно узнать только из его документации.

Например в html5 есть API addEventListener, которое позволяет привязывать к определенным событиям стандарта HTML5 функцию - обработчик.

Описание этого API находится вот тут:

<https://dom.spec.whatwg.org/#concept-event-listener-inner-invoke>

Где прямо заявляется о том, что вызов обработчика, будет осуществлен с привязкой this к объекту событие которого он обслуживает.



# Вместо ИГОГО

1. this в JavaScript это не контекст. Никогда им не был. Никогда им не будет.
2. this в JavaScript это особый идентификатор, который определен локально для всех normal function и по умолчанию задан как undefined для strict mode или как global object для non strict mode.
3. Значение this для функции, может быть изменено **только в момент вызова этой функции и зависит от формы/способа ее вызова.**
4. Внешнее API может связать this с произвольным значением в зависимости от фазы луны или левой пятки автора API.  
Но только в случае, если это будет normal function для которой мы не задали this посредством метода bind

# Особенности this в non strict mode

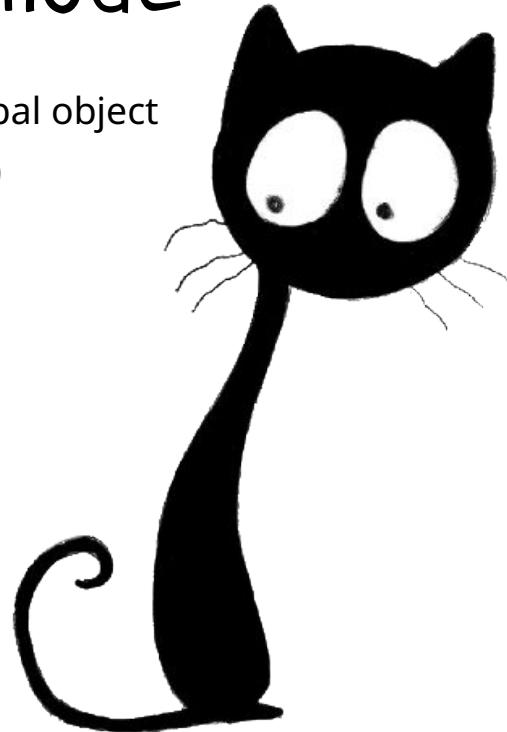
При вызове normal function, this связывается не с undefined, а с global object

В случае вызова с использованием dot нотации: theObj . doThing()

this будет связан со значением ToObject( theObj );

Например:

```
1 ~ String.prototype.doThingStrict = function () {
2     "use strict";
3     console.log( "this is:", this )
4 };
5 ~ String.prototype.doThing = function () {
6     console.log( "this is:", this )
7 };
8 "Yo".doThingStrict();    // this будет связан со значением "Yo"
9 "Yo".doThing();         // this будет связан со значением new String("Yo")
```





Для депеш:

asForJs:

Электропочта:

@demimuruch

<https://t.me/AsForJavaScript>

demimuruch@gmail.com

demi@murych.com