alvin777 21 марта 2016 в 09:25

# Стандарт ECMA-262 (JavaScript) в картинках, часть 2

JavaScript

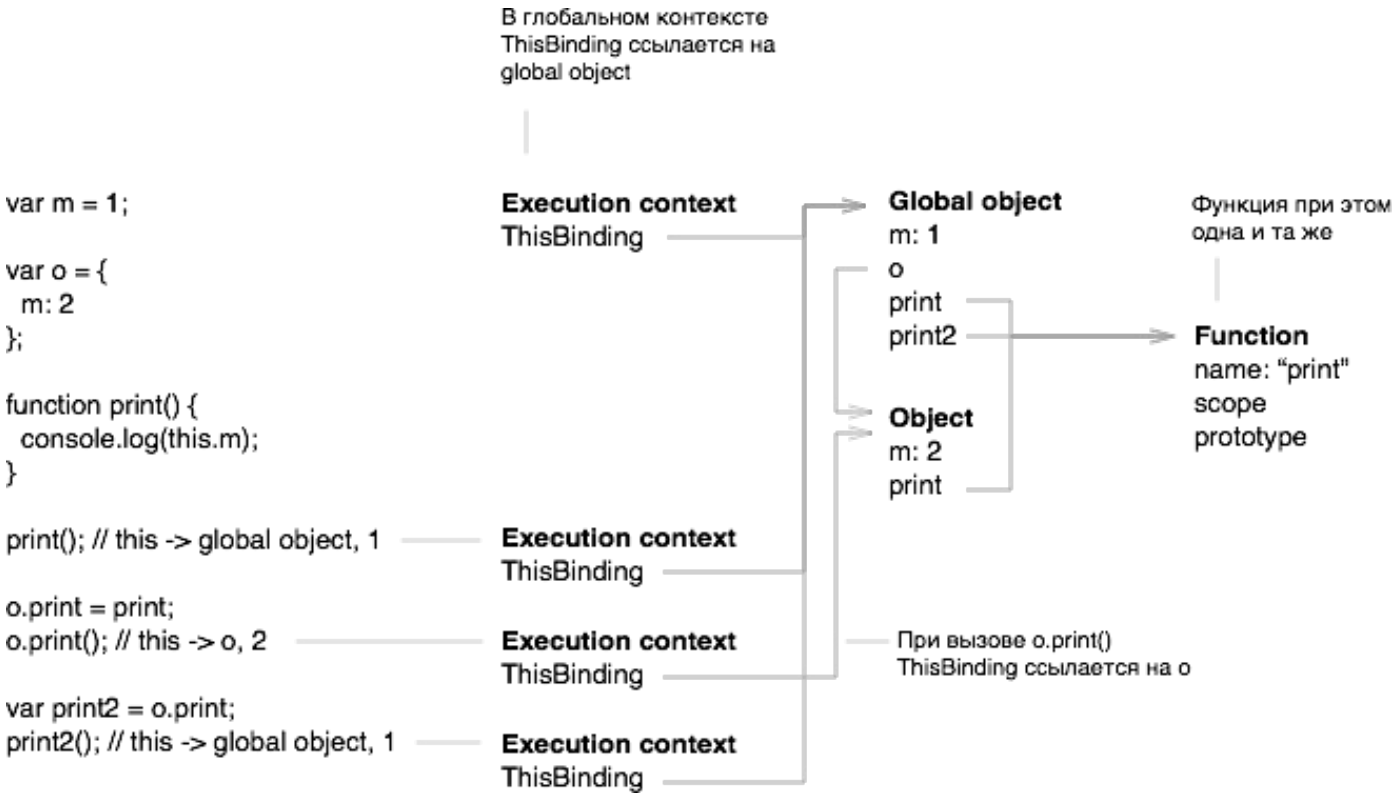
Tutorial


В первой части статьи рассматривались структуры *execution context*, *lexical environment* и объекты *Function*. Вторая часть посвящена использованию *this*.

## Как работает this

В *execution context* кроме *VariableEnvironment* есть поле *ThisBinding*. При поиске обычных переменных используется *VariableEnvironment*, при обращении через *this* – *ThisBinding*. *ThisBinding* устанавливается в *execution context* в зависимости от способа вызова функции. Если функция вызвана через точку *o.f()*, то *ThisBinding* будет указывать на объект *o*. Во всех остальных случаях *ThisBinding* будет ссылаться на *global object*. При этом функция может быть одна и та же.

Например, создадим функцию *print* и объект *o*, а затем добавим ссылку на функцию в объект. Получается, что функцию можно вызывать как `print()`, так и как `o.print()`. *ThisBinding* при этом будет различаться. Если скопировать ссылку на *o.print* в новую переменную *print2*, то при вызове *print2 ThisBinding* будет указывать на глобальный объект, несмотря на то, что ссылка была взята из объекта.



Еще раз, если вызов "через точку", то *ThisBinding* указывает на то, что до точки. В противном случае, *ThisBinding* ссылается на *global object*.

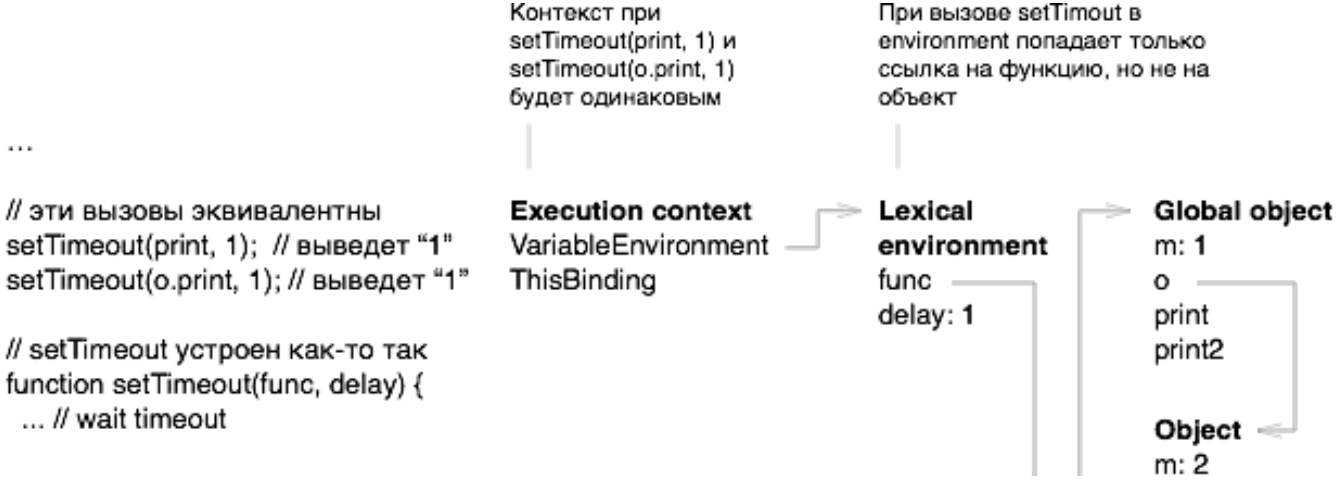
Подробнее про то, как устанавливается *this* при вызове функции, можно прочитать в следующих разделах стандарта:

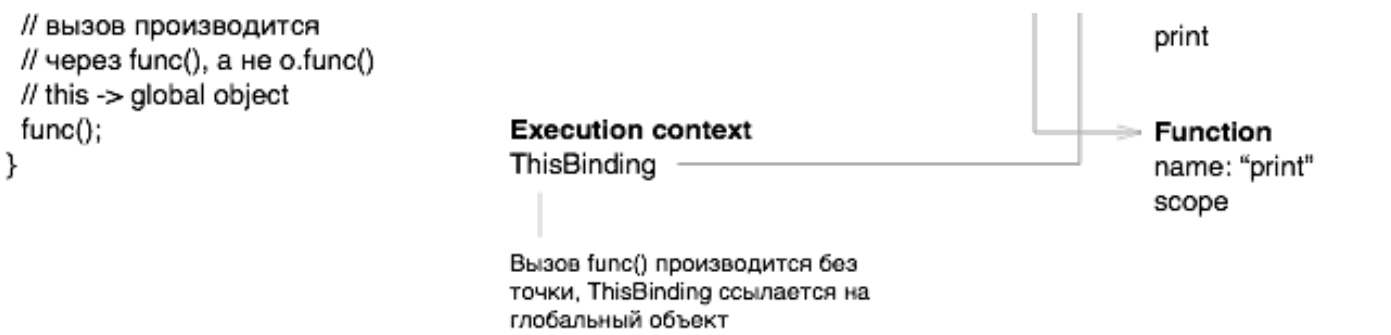
- 10.4.3 Entering Function Code
- 11.2.3 Function Calls
- 13.2.1 [[Call]]

## Что происходит при вызове setTimeout

При использовании *callback'ов* передается ссылка на функцию, но не на объект. Поэтому при вызове *callback'a this* будет указывать на *global object*.

В случае с *setTimout* первым аргументом передается ссылка на функцию *func*. После указанной задержки *setTimeout* её вызывает. Вызов производится без точки, просто как `func()`, поэтому *ThisBinding* указывает на *global object*.






Стандартный способ решить эту задачу – сохранить ссылку на объект в *environment* вспомогательной функции. Имея ссылку на объект можно воспользоваться функцией *Function.prototype.call*. При вызове функции с помощью *call* можно явно указать ссылку, которая будет записана в *ThisBinding* контекста.

Так при вызове `print.call(o)` будет создан контекст, в *ThisBinding* которого будет записана ссылка на *o*.

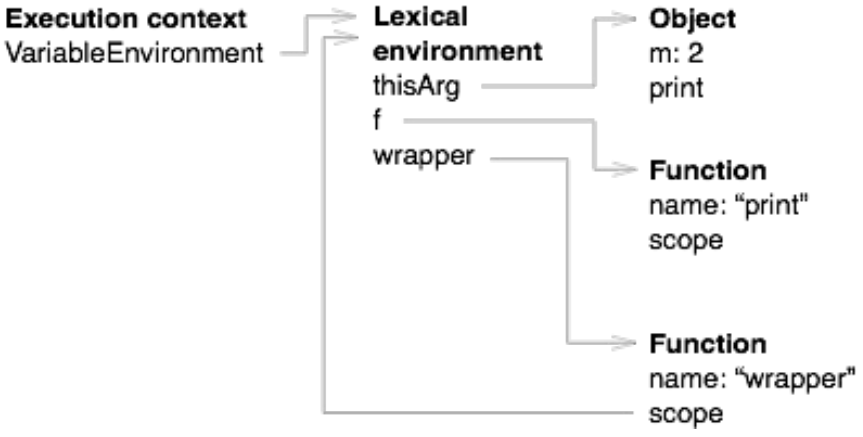
Такая функция как правило называется *bind* и может быть реализована следующим образом.

```
...
// bind возвращает функцию, которая вызывает функцию f
function bind(thisArg, f) {
  return function wrapper() {
    f.call(thisArg);
  }
}

// пользоваться bind надо так
setTimeout(bind(o, o.print), 1);
```

При вызове функции *bind* создается *environment* и объект *Function* с именем *wrapper*. В *thisArg* и *f* сохраняются ссылки на объект и функцию *print*. В *scope* нового объекта *Function* кладется ссылка на *environment*. При выходе из *bind* контекст уничтожается, но *environment* остается, так как на него ссылается *wrapper*.

...

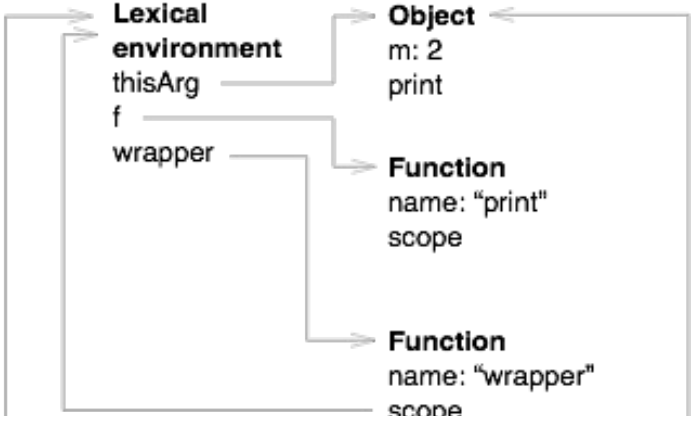
```
function bind(thisArg, f) {
  return function wrapper() {
    f.call(thisArg);
  }
}

setTimeout(bind(o, o.print), 1);
```

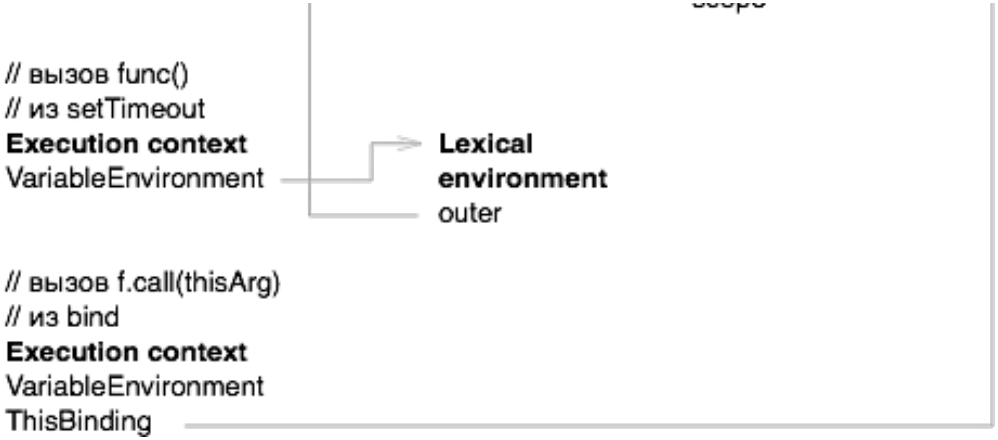



При вызове *setTimeout* через *delay* миллисекунд вызовется функция *wrapper*. При создании *environment* в поле *outer* записывается значение *scope* вызываемой функции. В данном случае *outer* будет ссылаться на *environment* функции *bind*. При вызове `f.call(thisArg)` из *wrapper* обе переменные будут найдены в этом *environment*. Затем при вызове *call* будет создан контекст для *print*, где *ThisBinding* будет указывать на *o*.

...

```
function bind(thisArg, f) {
  return function wrapper() {
    f.call(thisArg);
  }
}

setTimeout(bind(o, o.print), 1);
```






В JavaScript есть стандартный метод *Function.prototype.bind*, который позволяет запомнить не только *this*, но и аргументы функции.

```
// вместо
setTimeout(bind(o, o.print), 1);

// правильнее использовать стандартный bind
setTimeout(o.print.bind(o), 1);
```

В следующей части поговорим о прототипном наследовании.

**Теги:** javascript, ecmascript, ecma-262

**Хабы:** JavaScript

7,0 Карма

0,0 Рейтинг

**Краснояров Станислав** @alvin777

Full-stack разработчик

## ПОХОЖИЕ ПУБЛИКАЦИИ

25 марта 2016 в 10:11

### Стандарт ECMA-262 (JavaScript) в картинках, часть 3

+6 12,8k 124 1

16 марта 2016 в 14:06

### Стандарт ECMA-262 (JavaScript) в картинках, часть 1

+8 23,7k 146 1

31 декабря 2011 в 12:23

### ISO 8601 и ECMAScript — головная боль от разночтения стандартов

+48 8,4k 27 37

## ЗАКАЗЫ

Требуется html верстка 2х проектов

500 ₽ за час • 16 откликов • 137 просмотров

Фронтенд PHP+JS

450 000 ₽ за проект • 29 откликов • 110 просмотров

Разработка небольшой браузерной игры

35 000 ₽ за проект • 6 откликов • 48 просмотров

HTML верстка

10 000 ₽ за проект • 18 откликов • 151 просмотр

Сделать лендинг по дизайнам + Продвижение

15 000 ₽ за проект • 12 откликов • 94 просмотра

Больше заказов на Хабр Фрилансе



## Комментарии 1

**Apathetic** 21 марта 2016 в 22:16 +3

Тот же вопрос: где картинки?

Только полноправные пользователи могут оставлять комментарии. Войдите, пожалуйста.

ЧТО ОБСУЖДАЮТ

Сейчас | Вчера | Неделя

**Что случилось с транспортом и путешествиями за июль**

1,8k 5

**Советы руководителю от руководителя**

38,9k 187

Какая асинхронность должна была бы быть в Python

3,1k 9

Twitter будет блокировать ссылки на экстремистский контент

1,6k 17

Веселье продолжается: шорт-лист конкурса хабрамемов

Мегапост

## САМОЕ ЧИТАЕМОЕ

Сутки | Неделя | Месяц

USB-флешки: заряжать нельзя игнорировать

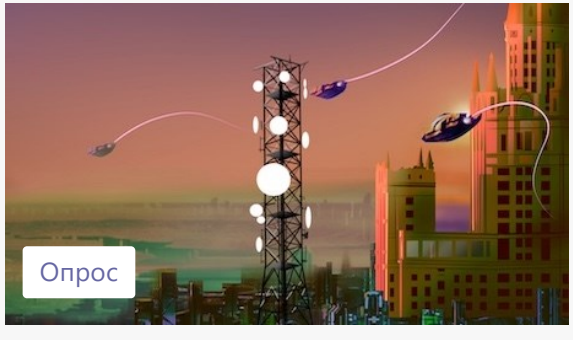
+157 75,2k 215 141

Где прячется Российская электроника

+102 31,6k 61 145

Не надо портить инженерам десктопы своими мобильными решениями,

## ПАРТНЕРСКИЕ МАТЕРИАЛЫ




«Серебряная пуля» или нет? Как Big Data меняет IT

одумайтесь

+206 45,3k 88 392

Коронавирус: первые итоги пандемии и карантина

+34 14,9k 43 103

Моё знакомство с AppGallery: как я воспользовался возможностями Huawei и нашёл точку роста для своего проекта

Мегапост


Познакомился, присмотрелся и в топ: опыт работы с AppGallery

Разместить