**[[Scope]] для new Function**

**[Присвоение [[Scope]] для new Function](http://learn.javascript.ru/scope-new-function" \l "scope-Function)**

Есть одно исключение из общего правила присвоения [[Scope]], которое мы рассматривали в предыдущей главе.

При создании функции с использованием new Function, её свойство [[Scope]] ссылается не на текущий LexicalEnvironment, а на window.

**[Пример](http://learn.javascript.ru/scope-new-function" \l "primer)**

Следующий пример демонстрирует как функция, созданная new Function, игнорирует внешнюю переменную a и выводит глобальную вместо неё:

var a = 1;

function getFunc() {

var a = 2;

var func = new Function('', 'alert(a)');

return func;

}

getFunc()(); // 1, из window

Сравним с обычным поведением:

var a = 1;

function getFunc() {

var a = 2;

var func = function() { alert(a); };

return func;

}

getFunc()(); // 2, из LexicalEnvironment функции getFunc

**[Почему так сделано?](http://learn.javascript.ru/scope-new-function" \l "pochemu-tak-sdelano)**

**Продвинутые знания**

Содержимое этой секции содержит продвинутую информацию теоретического характера, которая прямо сейчас не обязательна для дальнейшего изучения JavaScript.

Эта особенность new Function, хоть и выглядит странно, на самом деле весьма полезна.

Представьте себе, что нам действительно нужно создать функцию из строки кода. Текст кода этой функции неизвестен на момент написания скрипта (иначе зачем new Function), но станет известен позже, например получен с сервера или из других источников данных.

Предположим, что этому коду надо будет взаимодействовать с внешними переменными основного скрипта.

Но проблема в том, что JavaScript при выкладывании на «боевой сервер» предварительно сжимается минификатором – специальной программой, которая уменьшает размер кода, убирая из него лишние комментарии, пробелы, что очень важно – переименовывает локальные переменные на более короткие.

То есть, если внутри функции есть var userName, то минификатор заменит её на var a (или другую букву, чтобы не было конфликта), предполагая, что так как переменная видна только внутри функции, то этого всё равно никто не заметит, а код станет короче. И обычно проблем нет.

…Но если бы new Function могла обращаться к внешним переменным, то при попытке доступа к userName в сжатом коде была бы ошибка, так как минификатор переименовал её.

**Получается, что даже если бы мы захотели использовать локальные переменные в new Function, то после сжатия были бы проблемы, так как минификатор переименовывает локальные переменные.**

Описанная особенность new Function просто-таки спасает нас от ошибок.

Ну а если внутри функции, создаваемой через new Function, всё же нужно использовать какие-то данные – без проблем, нужно всего лишь предусмотреть соответствующие параметры и передавать их явным образом, например так:

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

var a = 1, b = 2;

alert( sum(a, b) ); // 3

**[Итого](http://learn.javascript.ru/scope-new-function" \l "itogo)**

* Функции, создаваемые через new Function, имеют значением [[Scope]] не внешний объект переменных, а window.
* Следствие – такие функции не могут использовать замыкание. Но это хорошо, так как бережёт от ошибок проектирования, да и при сжатии JavaScript проблем не будет. Если же внешние переменные реально нужны – их можно передать в качестве параметров.