**Устаревшая конструкция "with"**

Конструкция with позволяет использовать в качестве области видимости для переменных произвольный объект.

В современном JavaScript от этой конструкции отказались. С use strict она не работает, но её ещё можно найти в старом коде, так что стоит познакомиться с ней, чтобы если что – понимать, о чём речь.

Синтаксис:

with(obj) {

...код...

}

Любое обращение к переменной внутри with сначала ищет её среди свойств obj, а только потом – вне with.

**[Пример](http://learn.javascript.ru/with" \l "primer)**

В примере ниже переменная будет взята не из глобальной области, а из obj:

var a = 5;

var obj = {

a: 10

};

with(obj) {

alert( a ); // 10, из obj

}

Попробуем получить переменную, которой в obj нет:

var b = 1;

var obj = {

a: 10

};

with(obj) {

alert( b ); // 1, из window

}

Здесь интерпретатор сначала проверяет наличие obj.b, не находит и идет вне with.

Особенно забавно выглядит применение вложенных with:

var obj = {

weight: 10,

size: {

width: 5,

height: 7

}

};

with(obj) {

with(size) { // size будет взят из obj

alert( width \* height / weight ); // width,height из size, weight из obj

}

}

Свойства из разных объектов используются как обычные переменные… Магия! Порядок поиска переменных в выделенном коде: size => obj => window.

## [Изменения переменной](http://learn.javascript.ru/with" \l "izmeneniya-peremennoy)

При использовании with, как и во вложенных функциях – переменная изменяется в той области, где была найдена.

Например:

var obj = {

a: 10

}

with(obj) {

a = 20;

}

alert( obj.a ); // 20, переменная была изменена в объекте

## [Почему отказались от with?](http://learn.javascript.ru/with" \l "pochemu-otkazalis-ot-with)

Есть несколько причин.

1. В современном стандарте JavaScript отказались от with, потому что конструкция with подвержена ошибкам и непрозрачна.

Проблемы возникают в том случае, когда в with(obj) присваивается переменная, которая по замыслу должна быть в свойствах obj, но ее там нет.

Например:

var obj = {

weight: 10

};

with(obj) {

weight = 20; // (1)

size = 35; // (2)

}

alert( obj.size );

alert( window.size );

В строке (2) присваивается свойство, отсутствующее в obj. В результате интерпретатор, не найдя его, создает новую глобальную переменную window.size.

Такие ошибки редки, но очень сложны в отладке, особенно если size изменилась не в window, а где-нибудь во внешнем LexicalEnvironment.

1. Еще одна причина – алгоритмы сжатия JavaScript не любят with. Перед выкладкой на сервер JavaScript сжимают. Для этого есть много инструментов, например [Closure Compiler](http://code.google.com/intl/ru-RU/closure/compiler/) и [UglifyJS](https://github.com/mishoo/UglifyJS). Обычно они переименовывают локальные переменные в более короткие имена, но не свойства объектов. С конструкцией with до запуска кода непонятно – откуда будет взята переменная. Поэтому выходит, что, на всякий случай (если это свойство), лучше её не переименовывать. Таким образом, качество сжатия кода страдает.
2. Ну и, наконец, производительность – усложнение поиска переменной из-за with влечет дополнительные накладные расходы.

Современные движки применяют много внутренних оптимизаций, ряд которых не может быть применен к коду, в котором есть with.

Вот, к примеру, запустите этот код в современном браузере. Производительность функции fast существенно отличается от slow с пустым(!) with. И дело тут именно в with, т.к. наличие этой конструкции препятствует оптимизации.

var i = 0;

function fast() {

i++;

}

function slow() {

with(i) {}

i++;

}

var time = performance.now();

while (i < 1000000) fast();

alert( "Без with: " + (performance.now() - time) );

var time = performance.now();

i = 0;

while (i < 1000000) slow();

alert( "С with: " + (performance.now() - time) );

### [Замена with](http://learn.javascript.ru/with" \l "zamena-with)

Вместо with рекомендуется использовать временную переменную, например:

/\* вместо

with(elem.style) {

top = '10px';

left = '20px';

}

\*/

var s = elem.style;

s.top = '10px';

s.left = '0';

Это не так элегантно, но убирает лишний уровень вложенности и абсолютно точно понятно, что будет происходить и куда присвоятся свойства.

## [Итого](http://learn.javascript.ru/with" \l "itogo)

* Конструкция with(obj) { ... } использует obj как дополнительную область видимости. Все переменные, к которым идет обращение внутри блока, сначала ищутся в obj.
* Конструкция with устарела и не рекомендуется по ряду причин. Избегайте её.