**Глобальный объект**

Механизм работы функций и переменных в JavaScript очень отличается от большинства языков.

Чтобы его понять, мы в этой главе рассмотрим переменные и функции в глобальной области. А в следующей – пойдём дальше.

**[Глобальный объект](http://learn.javascript.ru/global-object" \l "globalnyy-obekt)**

*Глобальными* называют переменные и функции, которые не находятся внутри какой-то функции. То есть, иными словами, если переменная или функция не находятся внутри конструкции function, то они – «глобальные».

**В JavaScript все глобальные переменные и функции являются свойствами специального объекта, который называется *«глобальный объект»* (global object).**

В браузере этот объект явно доступен под именем window. Объект window одновременно является глобальным объектом и содержит ряд свойств и методов для работы с окном браузера, но нас здесь интересует только его роль как глобального объекта.

В других окружениях, например Node.JS, глобальный объект может быть недоступен в явном виде, но суть происходящего от этого не изменяется, поэтому далее для обозначения глобального объекта мы будем использовать "window".

**Присваивая или читая глобальную переменную, мы, фактически, работаем со свойствами window.**

Например:

var a = 5; // объявление var создаёт свойство window.a

alert( window.a ); // 5

Создать переменную можно и явным присваиванием в window:

window.a = 5;

alert( a ); // 5

## [Порядок инициализации](http://learn.javascript.ru/global-object" \l "poryadok-initsializatsii)

Выполнение скрипта происходит в две фазы:

1. На первой фазе происходит инициализация, подготовка к запуску.

Во время инициализации скрипт сканируется на предмет объявления функций вида [Function Declaration](http://learn.javascript.ru/function-declaration-expression), а затем – на предмет объявления переменных var. Каждое такое объявление добавляется в window.

**Функции, объявленные как Function Declaration, создаются сразу работающими, а переменные – равными undefined.**

1. На второй фазе – собственно, выполнение.

Присваивание (=) значений переменных происходит, когда поток выполнения доходит до соответствующей строчки кода, до этого они undefined.

В коде ниже указано содержание глобального объекта на момент инициализации и далее последовательно по коду:

// На момент инициализации, до выполнения кода:

// window = { f: function, a: undefined, g: undefined }

var a = 5;

// window = { f: function, a: 5, g: undefined }

function f(arg) { /\*...\*/ }

// window = { f: function, a: 5, g: undefined } без изменений, f обработана ранее

var g = function(arg) { /\*...\*/ };

// window = { f: function, a: 5, g: function }

Кстати, тот факт, что к началу выполнения кода переменные и функции уже содержатся в window, можно легко проверить, выведя их:

alert("a" in window); // true, т.к. есть свойство window.a

alert(a); // равно undefined, присваивание будет выполнено далее

alert(f); // function ..., готовая к выполнению функция

alert(g); // undefined, т.к. это переменная, а не Function Declaration

var a = 5;

function f() { /\*...\*/ }

var g = function() { /\*...\*/ };

**Присвоение переменной без объявления**

В старом стандарте JavaScript переменную можно было создать и без объявления var:

a = 5;

alert( a ); // 5

Такое присвоение, как и var a = 5, создает свойство window.a = 5. Отличие от var a = 5 – в том, что переменная будет создана не на этапе входа в область видимости, а в момент присвоения.

Сравните два кода ниже.

Первый выведет undefined, так как переменная была добавлена в window на фазе инициализации:

alert( a ); // undefined

var a = 5;

Второй код выведет ошибку, так как переменной ещё не существует:

alert( a ); // error, a is not defined

a = 5;

Это, конечно, для общего понимания, мы всегда объявляем переменные через var.

**Конструкции for, if... не влияют на видимость переменных**

Фигурные скобки, которые используются в for, while, if, в отличие от объявлений функции, имеют «декоративный» характер.

В JavaScript нет разницы между объявлением вне блока:

var i;

{

i = 5;

}

…И внутри него:

i = 5;

{

var i;

}

**Также нет разницы между объявлением в цикле и вне него:**

for (var i = 0; i < 5; i++) { }

Идентичный по функциональности код:

var i;

for (i = 0; i < 5; i++) { }

В обоих случаях переменная будет создана до выполнения цикла, на стадии инициализации, и ее значение будет сохранено после окончания цикла.

**Не важно, где и сколько раз объявлена переменная**

Объявлений var может быть сколько угодно:

var i = 10;

for (var i = 0; i < 20; i++) {

...

}

var i = 5;

**Все var будут обработаны один раз, на фазе инициализации.**

На фазе исполнения объявления var будут проигнорированы: они уже были обработаны. Зато будут выполнены присваивания.

**Ошибки при работе с window в IE8-**

В старых IE есть две забавные ошибки при работе с переменными в window:

1. Переопределение переменной, у которой такое же имя, как и id элемента, приведет к ошибке:

<div id="a">...</div>

<script>

a = 5; // ошибка в IE8-! Правильно будет "var a = 5"

alert( a ); // никогда не сработает

</script>

А если сделать через var, то всё будет хорошо.

Это была реклама того, что надо везде ставить var.

1. Ошибка при рекурсии через функцию-свойство window. Следующий код «умрет» в IE8-:

<script>

// рекурсия через функцию, явно записанную в window

window.recurse = function(times) {

if (times !== 0) recurse(times - 1);

}

recurse(13);

</script>

Проблема здесь возникает из-за того, что функция напрямую присвоена в window.recurse = .... Ее не будет при обычном объявлении функции.

**Этот пример выдаст ошибку только в настоящем IE8!** Не IE9 в режиме эмуляции. Вообще, режим эмуляции позволяет отлавливать где-то 95% несовместимостей и проблем, а для оставшихся 5% вам нужен будет настоящий IE8 в виртуальной машине.

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

В результате инициализации, к началу выполнения кода:

1. Функции, объявленные как Function Declaration, создаются полностью и готовы к использованию.
2. Переменные объявлены, но равны undefined. Присваивания выполнятся позже, когда выполнение дойдет до них.