# Function hoisted

Обратите внимание, что присваивание значений переменным не поднимается вместе с их объявлением. Поднимаются только объявления переменных. В случае с функциями, поднимается вся функция целиком. Существуют два основных способа объявить функцию, давайте их рассмотрим:

function test() {

foo(); *// TypeError "foo is not a function"*

bar(); *// "this will run!"*

var foo = function () { *// функциональное выражение, присвоенное локальной переменной 'foo'*

alert("this won't run!");

}

function bar() { *// объявление функции с именем 'bar'*

alert("this will run!");

}

}

test();

в этом случае поднимается только функция bar. Идентификатор «foo» также поднимается, но не анонимная функция — она остаётся на месте.  
  
Вот мы и описали основные моменты «поднятия» переменных и функций. Конечно, JavaScript не был бы сам собой, если бы не было особых случаев, в которых всё немного сложнее.

References obj

var obj = { a: 1 };

(function(obj) {

obj = { a: 2 };

})(obj);

console.log(obj.a);

При вызове анонимной функции создастся локальная переменная obj в её области видимости. А затем создаётся новый объект {a : 2}, ссылка на который попадает в локальную переменную obj, но переменная из верхнего скоупа будет всё так же ссылаться на старый объект.

Контекст вызова

Logger = function(logFn) { \_logFn = logFn; this.log = function(message) { \_logFn(new Date() + ": " + message); } } var logger = new Logger(console.log); logger.log("Hi!"); logger.log("Wazzup?");

В консоли мы увидим TypeError: Illegal invocation  
А всё потому что при вызове logger.log(), контекст выполнения функции — logger  
  
Чтобы починить можно вспомнить про встроенные методы функций .apply(), .call(), .bind()

< rightLogger = new Logger(console.log.bind(console))

> Logger {log: function}

< rightLogger.log("It's works")

> Sat Oct 04 2014 00:32:49 GMT+0400 (MSK): It's works

(function() {

var x = 1;

function x() {};

console.log(x); })()

Функции объявленный при помощи function declaration имеют больший приоритет и понимаются выше var. Поэтому интерпретатор сначала выполнит function x() {};, а затем var x = 1;  
  
< (function() {  
var x = 1;  
  
function x() {};  
  
console.log(x);   
})()  
  
> undefined  
1