JavaScript Variable Scope and Hoisting Explained

jan. 31 2013 [104](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comments)

In this post, we will learn JavaScript’s variable scope and hoisting and all the idiosyncrasies of both.

We must understand how variable scope and variable hoisting work in JavaScript, if want to understand JavaScript well. These concepts may seem straightforward; they are not. Some important subtleties exist that we must understand, if we want to thrive and excel as JavaScript developers.

* Receive Updates

Top of Form



Bottom of Form

**Variable Scope**  
A variable’s scope is the context in which the variable exists. The scope specifies from where you can access a variable and whether you have access to the variable in that context.

[[](https://bovacademy.com/)](https://bovacademy.com/" \t "_blank)

[Become an Elite/Highly Paid Specialized Software Engineer (Frontend, Fullstack, etc.)](https://bovacademy.com/" \t "_blank)

[Within 8–10 Months, Earn MORE than the Avg. New CS Graduate Earns.](https://bovacademy.com/" \t "_blank)

**[New Prices (Monthly Subscription):](https://bovacademy.com/" \t "_blank)** [Basic: $250/month, Pro: $350/month, Premium: $600/month](https://bovacademy.com/" \t "_blank)

[By the founder of JavaScriptIsSexy](https://bovacademy.com/" \t "_blank)

Variables have either a local scope or a global scope.

**Local Variables (Function-level scope)**  
Unlike most programming languages, JavaScript does not have block-level scope (variables scoped to surrounding curly brackets); instead, JavaScript has function-level scope. Variables declared within a function are local variables and are only accessible within that function or by functions inside that function. See my post on [Closures](http://javascriptissexy.com/javascript-closures-in-lovely-detail/) for more on accessing variables in outer functions from inner functions.  
  
Demonstration of Function-Level Scope

|  |  |
| --- | --- |
|  | var name = "Richard"; |
|  | ​ |
|  | ​function showName () { |
|  | var name = "Jack"; // local variable; only accessible in this showName function​ |
|  | console.log (name); // Jack​ |
|  | } |
|  | console.log (name); // Richard: the global variable |

No Block-Level Scope

|  |  |
| --- | --- |
|  | var name = "Richard"; |
|  | ​// the blocks in this if statement do not create a local context for the name variable​ |
|  | ​if (name) { |
|  | name = "Jack"; // this name is the global name variable and it is being changed to "Jack" here​ |
|  | console.log (name); // Jack: still the global variable​ |
|  | } |
|  | ​ |
|  | ​// Here, the name variable is the same global name variable, but it was changed in the if statement​ |
|  | console.log (name); // Jack |

* **If You Don’t Declare Your Local Variables, Trouble is Nigh**  
  Always declare your local variables before you use them. In fact, you should use [JSHint](http://www.jshint.com/) to check your code for syntax errors and style guides. Here is the trouble with not declaring local variables:

|  |  |
| --- | --- |
|  | // If you don't declare your local variables with the var keyword, they are part of the global scope​ |
|  | ​var name = "Michael Jackson"; |
|  | ​ |
|  | ​function showCelebrityName () { |
|  | console.log (name); |
|  | } |
|  | ​ |
|  | ​function showOrdinaryPersonName () { |
|  | name = "Johnny Evers"; |
|  | console.log (name); |
|  | } |
|  | showCelebrityName (); // Michael Jackson​ |
|  | ​ |
|  | ​// name is not a local variable, it simply changes the global name variable​ |
|  | showOrdinaryPersonName (); // Johnny Evers​ |
|  | ​ |
|  | ​// The global variable is now Johnny Evers, not the celebrity name anymore​ |
|  | showCelebrityName (); // Johnny Evers​ |
|  | ​ |
|  | ​// The solution is to declare your local variable with the var keyword​ |
|  | ​function showOrdinaryPersonName () { |
|  | var name = "Johnny Evers"; // Now name is always a local variable and it will not overwrite the global variable​ |
|  | console.log (name); |
|  | } |

* **Local Variables Have Priority Over Global Variables in Functions**  
  If you declare a global variable and a local variable with the same name, the local variable will have priority when you attempt to use the variable inside a function (local scope):

|  |  |
| --- | --- |
|  | var name = "Paul"; |
|  | ​ |
|  | ​function users () { |
|  | // Here, the name variable is local and it takes precedence over the same name variable in the global scope​ |
|  | ​var name = "Jack"; |
|  | ​ |
|  | ​// The search for name starts right here inside the function before it attempts to look outside the function in the global scope​ |
|  | console.log (name); |
|  | } |
|  | ​ |
|  | users (); // Jack |

**Global Variables**  
All variables declared outside a function are in the global scope. In the browser, which is what we are concerned with as front-end developers, the global context or scope is the window object (or the entire HTML document).

* Any variable declared or initialized outside a function is a global variable, and it is therefore available to the entire application. For example:

|  |  |
| --- | --- |
|  | // To declare a global variable, you could do any of the following:​ |
|  | ​var myName = "Richard"; |
|  | ​ |
|  | ​// or even​ |
|  | firstName = "Richard"; |
|  | ​ |
|  | ​// or ​ |
|  | ​var name; //​ |
|  | name; |
|  | ​</pre> |

* If a variable is initialized (assigned a value) without first being declared with the var keyword, it is automatically added to the global context and it is thus a global variable:

|  |  |
| --- | --- |
|  | function showAge () { |
|  | // Age is a global variable because it was not declared with the var keyword inside this function​ |
|  | age = 90; |
|  | console.log(age);// ​ |
|  | } |
|  | ​ |
|  | showAge (); // 90​ |
|  | ​ |
|  | ​// Age is in the global context, so it is available here, too​ |
|  | console.log(age); // 90 |

* Demonstration of variables that are in the Global scope even as they seem otherwise:

|  |  |
| --- | --- |
|  | // Both firstName variables are in the global scope, even though the second one is surrounded by a block {}. ​ |
|  | ​var firstName = "Richard"; |
|  | { |
|  | ​var firstName = "Bob"; |
|  | } |
|  | ​ |
|  | ​// To reiterate: JavaScript does not have block-level scope​ |
|  | ​ |
|  | ​// The second declaration of firstName simply re-declares and overwrites the first one​ |
|  | console.log (firstName); // Bob |

* Another example

|  |  |
| --- | --- |
|  | for (var i = 1; i <= 10; i++) { |
|  | console.log (i); // outputs 1, 2, 3, 4, 5, 6, 7, 8, 9, 10;​ |
|  | }; |
|  | ​ |
|  | ​// The variable i is a global variable and it is accessible in the following function with the last value it was assigned above ​ |
|  | ​function aNumber () { |
|  | console.log(i); |
|  | } |
|  | ​ |
|  | ​// The variable i in the aNumber function below is the global variable i that was changed in the for loop above. Its last value was 11, set just before the for loop exited:​ |
|  | aNumber (); // 11​ |
|  | ​ |

* **setTimeout Variables are Executed in the Global Scope**  
  Note that all functions in setTimeout are executed in the global scope. This is a tricky bit; consider this:

|  |  |
| --- | --- |
|  | // The use of the "this" object inside the setTimeout function refers to the Window object, not to myObj​ |
|  | ​ |
|  | ​var highValue = 200; |
|  | ​var constantVal = 2; |
|  | ​var myObj = { |
|  | highValue: 20, |
|  | constantVal: 5, |
|  | calculateIt: function () { |
|  | setTimeout (function () { |
|  | console.log(this.constantVal \* this.highValue); |
|  | }, 2000); |
|  | } |
|  | } |
|  | ​ |
|  | ​// The "this" object in the setTimeout function used the global highValue and constantVal variables, because the reference to "this" in the setTimeout function refers to the global window object, not to the myObj object as we might expect.​ |
|  | ​ |
|  | myObj.calculateIt(); // 400​ |
|  | ​// This is an important point to remember. |

* **Do not Pollute the Global Scope**  
  If you want to become a JavaScript master, which you certainly want to do (otherwise you will be watching Honey Boo Boo right now), you have to know that it is important to avoid creating many variables in the global scope, such as this:

|  |  |
| --- | --- |
|  | // These two variables are in the global scope and they shouldn't be here​ |
|  | ​var firstName, lastName; |
|  | ​ |
|  | ​function fullName () { |
|  | console.log ("Full Name: " + firstName + " " + lastName ); |
|  | } |

* This is the improved code and the proper way to avoid polluting the global scope

|  |  |
| --- | --- |
|  | // Declare the variables inside the function where they are local variables​ |
|  | ​ |
|  | ​function fullName () { |
|  | var firstName = "Michael", lastName = "Jackson"; |
|  | ​ |
|  | console.log ("Full Name: " + firstName + " " + lastName ); |
|  | } |

* In this last example, the function fullName is also in the global scope.

**Variable Hoisting**  
All variable declarations are hoisted (lifted and declared) to the top of the function, if defined in a function, or the top of the global context, if outside a function.

It is important to know that only variable declarations are hoisted to the top, not variable initialization or assignments (when the variable is assigned a value).

Variable Hoisting Example:

|  |  |
| --- | --- |
|  | function showName () { |
|  | console.log ("First Name: " + name); |
|  | ​var name = "Ford"; |
|  | console.log ("Last Name: " + name); |
|  | } |
|  | ​ |
|  | showName (); |
|  | ​// First Name: undefined​ |
|  | ​// Last Name: Ford​ |
|  | ​ |
|  | ​// The reason undefined prints first is because the local variable name was hoisted to the top of the function​ |
|  | ​// Which means it is this local variable that get calls the first time.​ |
|  | ​// This is how the code is actually processed by the JavaScript engine:​ |
|  | ​ |
|  | ​function showName () { |
|  | var name; // name is hoisted (note that is undefined at this point, since the assignment happens below)​ |
|  | console.log ("First Name: " + name); // First Name: undefined​ |
|  | ​ |
|  | name = "Ford"; // name is assigned a value​ |
|  | ​ |
|  | ​// now name is Ford​ |
|  | console.log ("Last Name: " + name); // Last Name: Ford​ |
|  | } |

**Function Declaration Overrides Variable Declaration When Hoisted**  
Both function declaration and variable declarations are hoisted to the top of the containing scope. And function declaration takes precedence over variable declarations (but not over variable assignment). As is noted above, variable assignment is not hoisted, and neither is function assignment. As a reminder, this is a function assignment: var myFunction = function () {}.  
Here is a basic example to demonstrate:

|  |  |
| --- | --- |
|  | // Both the variable and the function are named myName​ |
|  | ​var myName; |
|  | ​function myName () { |
|  | console.log ("Rich"); |
|  | } |
|  | ​ |
|  | ​// The function declaration overrides the variable name​ |
|  | console.log(typeof myName); // function |

|  |  |
| --- | --- |
|  | // But in this example, the variable assignment overrides the function declaration​ |
|  | ​var myName = "Richard"; // This is the variable assignment (initialization) that overrides the function declaration.​ |
|  | ​ |
|  | ​function myName () { |
|  | console.log ("Rich"); |
|  | } |
|  | ​ |
|  | console.log(typeof myName); // string |

It is important to note that function expressions, such as the example below, are not hoisted.

|  |  |
| --- | --- |
|  | var myName = function () { |
|  | console.log ("Rich"); |
|  | } |

In strict mode, an error will occur if you assign a variable a value without first declaring the variable. Always declare your variables.

Be good. Sleep well. And enjoy coding.

[[](https://bovacademy.com/)](https://bovacademy.com/" \t "_blank)

[Become an Elite/Highly Paid Specialized Software Engineer (Frontend, Fullstack, etc.)](https://bovacademy.com/" \t "_blank)

[Within 8–10 Months, Earn MORE than the Avg. New CS Graduate Earns.](https://bovacademy.com/" \t "_blank)

**[New Prices (Monthly Subscription):](https://bovacademy.com/" \t "_blank)** [Basic: $250/month, Pro: $350/month, Premium: $600/month](https://bovacademy.com/" \t "_blank)

[By the founder of JavaScriptIsSexy](https://bovacademy.com/" \t "_blank)

Posted in: [16 Important JavaScript Concepts](http://javascriptissexy.com/category/16-important-javascript-concepts/), [JavaScript](http://javascriptissexy.com/category/javascript/) / Tagged: [Learn JavaScript](http://javascriptissexy.com/tag/learn-javascript/), [Variable Hoisting](http://javascriptissexy.com/tag/variable-hoisting/), [Variable Scope](http://javascriptissexy.com/tag/variable-scope/)

[Richard](http://javascriptissexy.com/author/richardb/)Thanks for your time; please come back soon. Email me here: javascriptissexy at gmail email, or use the [contact](http://javascriptissexy.com/contact/) form.

104 Comments

1. http://2.gravatar.com/avatar/21ac22581daff15b289162792f1ec519?s=60&d=mm&r=g

Rameş Aliyev

[February 1, 2013 at 2:52 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-1914) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=1914#respond)

In for loop, when i create “i” with “var”, why it becomes global? I try it and see it, its no make sense, is it a bug of js?

Here is live example  
<http://jsbin.com/azawaj/1/edit>

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[February 3, 2013 at 11:59 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-1952) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=1952#respond)

Hello Rames,

If you are referring to the example below, i is a global variable because it was not defined in a function, and it is therefore part of the global scope. Even if the variable i were declared and initialized in the for loop, it would still be a global variable because JavaScript does not have block-level scope, so the block (curly brackets) from the for loop do not create a local scope.

Here is the code:  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
for (var i = 1; i <= 10; i++) { console.log (i); // outputs 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10; }; // i is a the global variable and it is accessible here with the last value it was assigned above function aNumber () { console.log(i); // 11 } aNumber (); // 10 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ The code is valid, just copy and paste it and run it in your browser’s console.

1. http://0.gravatar.com/avatar/3ea1022e47952a4f6114f35ae72072c5?s=60&d=mm&r=g

[isabel marant sneakers](http://isabellemarantsneakers.weebly.com/)

[February 2, 2013 at 9:21 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-1929) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=1929#respond)

“Its always good to read tips as you share with regard to blog placing. As I just now started placing comments with regard to blog plus facing challenge of loads of rejections. I believe that your suggestion will be helpful personally. I will allow you to know if perhaps its improve me too. “

1. http://2.gravatar.com/avatar/8f2541c0c2c963f8d490d9c1997e2764?s=60&d=mm&r=g

Vetri

[February 13, 2013 at 4:55 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-2239) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=2239#respond)

What do you mean by this?

It is important to note that function expressions, such as the example below, are not hoisted.

var myName = function () {  
console.log (“Rich”);  
}

but I can see this hoisted.

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[February 13, 2013 at 2:29 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-2269) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=2269#respond)

@Vetri The function expression, var myName = function () {console.log (“Rich”);}, is not actually hoisted. Instead, if you initialize it after the var myName = “Rich” initialization, the myName variable is assigned a new value, which is the function.

It looks like this when the code is executed:  
var myName;  
myName = “Rich”;

// Then it is assigned a new value, the function  
myName = function () {console.log (“Rich”);}

1. http://0.gravatar.com/avatar/912ddb0dd796437303d60f5b72340677?s=60&d=mm&r=g

Ramon Royo

[February 15, 2013 at 1:51 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-2343) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=2343#respond)

Hi Richard,

thank you for the article, I wasn’t aware of the scope for ‘this’ inside timeouts and intervals.  
I just wanted to point out a small problem in one of the examples, the variable names in the following example, should be surrounded by double quotes, as now they’re checked as properties inside window:

console.log(myName in window);  
console.log(firstName in window);

Should be:

console.log(“myName” in window); // true  
console.log(“firstName” in window); // true

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[February 15, 2013 at 5:08 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-2347) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=2347#respond)

Thanks for pointing out the typo, Ramon. I just fixed it.

1. http://0.gravatar.com/avatar/cf23360bef543714ec0a77b3100e929c?s=60&d=mm&r=g

Sean

[March 6, 2013 at 10:29 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-3557) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=3557#respond)

Great Post!! Quick Question Richard.. So basically theres no need to declare variables outside of a function if i can declare them within a function.. basically in what case would I need to declare variables outside a function when its easier to use them inside a function as below… it seems like local variables are easier to work with then global…

function myFunction () {  
var a = “foo”  
}

function newFunction () {  
var a = “boo”  
}  
myfunction()  
newFunction()  
why would i want to declare a variable outside a function and I can make one inside a function via local scope and use them later???  
Thanks!!!!

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[March 7, 2013 at 3:36 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-3596) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=3596#respond)

*So basically theres no need to declare variables outside of a function if i can declare them within a function.*

You are correct. It is best to just use local variables, unless you need a global variable.

*basically in what case would I need to declare variables outside a function when its easier to use them inside a function as below*

There are times when you do need global variables, for example, when you want any function in your application to have access to the variable. Keep in mind your other functions wouldn’t have access to your local variables, so if you want other functions to access a variable, you would have to declare it as a global variable or pass it via the parameter to a function.

There are many times when you are building a large application where you will have one or more global variables.

1. http://0.gravatar.com/avatar/cf23360bef543714ec0a77b3100e929c?s=60&d=mm&r=g

Sean

[March 7, 2013 at 4:13 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-3597) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=3597#respond)

Great!! thanks for that, closures and hoisting are a bit confusing but after reading this it totaly makes sense…im on my journey reading and learning from this site along with Nicholas Zakas Professional Book … back to reading & coding 

1. http://0.gravatar.com/avatar/69ea6d7b8f7fe6f558a84af746af8d05?s=60&d=mm&r=g

[whaleshark](http://twitter.com/sergi_bosch)

[May 8, 2013 at 12:49 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-7242) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=7242#respond)

In your example for hoisting, you say that the variable assignment will override the function declaration — well, in the Firefox (20) console it does not. Is that a browser bug? I see it works in Chrome.

thanks

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[May 8, 2013 at 9:42 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-7289) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=7289#respond)

Good question and good research, whaleshark.

You are correct that the overriding is working as expected in Chrome, while it is not in Firefox. The reason is that in Firefox, the second use of “myName” is a reassignment, so essentially, Firefox is assigning a new value to the myName variable. It is assigning the function to it.

This is how the code is executed in Firefox:

|  |  |
| --- | --- |
|  | ​ |
|  | ​// A new myName global variable is created in Firefox and assigned the value "Richard"​ |
|  | ​var myName = "Richard"; |
|  | ​ |
|  | ​/\* The same global myName variable is now assigned a new value. The first myName String assignment is literally destroyed (because Stings are immutable—they cannot be changed) and a new myName global variable is created and assigned the value of the following function: \*/​ |
|  | myName = function () { |
|  | console.log ("Rich"); |
|  | } |
|  | ​ |

Now, you might be wondering why it is not executed the same way by both browsers. Welcome to land of Browsers gone rogue. All the browsers have similar but different implementation for many os JavaScript’s API, so you have to watch out for these kinds of side effects, which is why using the same global variable in this manner is not a good idea to begin with.

1. http://1.gravatar.com/avatar/4fde6dee0dfb32e96507fa4cc540ccdd?s=60&d=mm&r=g

A N Sinha

[May 16, 2013 at 4:40 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-8012) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=8012#respond)

Hi Richard,

Thanks your very much 

JavaScript Variable Scope :  
====================  
for (var i = 1; i <= 10; i++) {  
console.log (i); // outputs 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10;  
};  
function aNumber () {  
console.log(i); // 11  
}  
aNumber ();  
=================  
I REALLY SURPRISE  
var i is define inside the for loop it treated as a global variable.

I have test the code in firebug, its treated as a global variable…..

that's great…

thanks your very much 

A.N. Sinha 

* + http://2.gravatar.com/avatar/2b18fede97658dd573b974985c86161e?s=60&d=mm&r=g

Chachaching

[November 19, 2013 at 1:32 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-162339) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=162339#respond)

The behavior you are seeing right now on JavaScript is a new change in JavaScript. The variable scope used to be the same as other languages, but I also noticed the same thing a while back that there is no longer a variable scope WITHIN a function. In other words, once you start using/assigning a new variable inside a function, the variable will live through out the function scope regardless there are other sub scopes inside the function. I am still trying to find out whether this is a bug or intention of JavaScript (and it is definitely unconventional idea for all other computer languages).

* + - http://2.gravatar.com/avatar/2b18fede97658dd573b974985c86161e?s=60&d=mm&r=g

Chachaching

[November 19, 2013 at 1:38 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-162341) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=162341#respond)

One more note, a variable you declared outside any function but inside the script tag will be GLOBAL regardless where you declare it and with or without ‘var’ keyword in front. To me, this idea is silly and it will break any variable scope idea.

* + - http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Of Stanley](http://javascriptissexy.com/) (Author)

[November 22, 2013 at 7:31 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-163151) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=163151#respond)

Good observation, Chachaching.

Modern browsers add different implementations for how to hand variable scope and when to garbage collect said variables.

There was a vibrant discussion on this issue on Hacker News a few months back. If I find the link, I will post it.

1. http://0.gravatar.com/avatar/3c01c944d95c16488d38bf17900b47ba?s=60&d=mm&r=g

Mike

[May 24, 2013 at 11:38 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-8955) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=8955#respond)

Great site! Your articles are perfect for experienced Javascript developers who want to round out their knowledge and get into the fine details. Thanks!

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[May 27, 2013 at 10:06 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-9464) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=9464#respond)

Thanks very much, Mike.

1. http://1.gravatar.com/avatar/4cd1e95835ceb2e79b7c7b06970bf743?s=60&d=mm&r=g

Eddie Romeiro

[July 10, 2013 at 5:29 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-19158) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=19158#respond)

Hi there. Great post and site. In regards to hoisting, there was a little confusion (at least for me) in your explanation about function hoisting. It should be made clear that function declarations get fully hoisted to the top, while function expressions behave like variables. Maybe even add that function declarations get hoisted and turn into an expression at the top. Just my opinion.

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[July 11, 2013 at 7:50 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-19396) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=19396#respond)

Hi Eddie,  
Your opinion is actually the fact: function declarations are indeed hoisted while function expressions are not.

I did note this in the article, maybe you missed it. Here is the quote from the article:

*Both function declaration and variable declarations are hoisted to the top of the containing scope. And function declaration takes precedence over variable declarations (but not over variable assignment). As is noted above, variable assignment is not hoisted, and neither is function assignment. As a reminder, this is a function assignment: var myFunction = function () {}.*

1. http://1.gravatar.com/avatar/1eee90df0c3d3d1af05671d0d3f21d5f?s=60&d=mm&r=g

abeing

[August 30, 2013 at 10:09 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-33768) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=33768#respond)

Hi, great article!  
I did this:  
function doSomething(){  
function formatSomething(element){  
return “”+element.text+””;  
}

anObject({ format: formatSomething });  
anotherObject({format: formatSomething });  
}

a function within a function, which I find very useful, because it worked and I had the function at the place where I need it.  
But is this good coding style?

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Bovell](http://javascriptissexy.com/) (Author)

[August 30, 2013 at 4:17 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-33840) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=33840#respond)

*a function within a function, which I find very useful, because it worked and I had the function at the place where I need it.  
But is this good coding style?*

If your question is whether a function inside a function (better known as a closure) is a good coding style, then the answer is a definite yes.

Closures are used a lot by advanced JavaScript coders, and probably by most JavaScript coders in general. And sometimes it is even tough to avoid using a closure, so feel free to use closures liberally.

I actually wrote an article on just closures—see the archive.

1. http://1.gravatar.com/avatar/40fc0acae1d23d81d942ea4734314912?s=60&d=mm&r=g

Foysal Mamun

[September 30, 2013 at 9:21 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-57923) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=57923#respond)

thanks nice post.

1. http://1.gravatar.com/avatar/14282fa30326b5907eaffc1346e60817?s=60&d=mm&r=g

Deepak

[October 3, 2013 at 7:58 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-66429) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=66429#respond)

what happens when i am not used var for delectation. just used name; instated of var name.?

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Of Stanley](http://javascriptissexy.com/) (Author)

[October 11, 2013 at 2:41 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-86470) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=86470#respond)

If you do not use var, the variable will be a global variable defined on the Window object. But this can cause trouble, especially when using strict mode.

1. http://2.gravatar.com/avatar/8f2cf3602f2f30e752f23c1b129df464?s=60&d=mm&r=g

Gulsen

[October 6, 2013 at 1:31 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-74356) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=74356#respond)

Hi Richard,

Thanks for the great article.

In your example with the celebrity name and ordinary person name, the first call to showCelebrityName() function, it logs “Michael Jackson”, and the second time “Johnny Evers”. Since the ShowOrdinaryPersonName is defined before the first showCelebrityName() call, does that mean that the name variable is changed to “Johnny Evers” once the ShowOrdinaryPersonName() is called but not when it is defined? Just to make sure I understand it correctly…

Thanks,  
Gulsen

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Of Stanley](http://javascriptissexy.com/) (Author)

[October 11, 2013 at 2:56 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-86504) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=86504#respond)

You are correct, Gulsen.

1. http://2.gravatar.com/avatar/2f37f3672ccadf916d68b9ace7a2d5ee?s=60&d=mm&r=g

Vipul

[October 6, 2013 at 11:54 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-75532) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=75532#respond)

Hi Richard,

Thanks for the great article.  
I am new to Javascript, learnt new things from your posts. Actually i am focusing as a Javascript developer as my carrier path. Hope it will help me.

Thanks,  
Vipul

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Of Stanley](http://javascriptissexy.com/) (Author)

[October 11, 2013 at 2:25 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-86433) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=86433#respond)

You are welcome, and much success to you in your JavaScript career.

1. http://2.gravatar.com/avatar/bf7e5a966f9d5db874ee1d17c346fc65?s=60&d=mm&r=g

Alisa

[October 9, 2013 at 10:06 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-82547) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=82547#respond)

Hello,  
In gratitude for your informative article, I offer you some proofreading corrections…  
“Jhonny Evers”  
“global scape”  
“Variable Hosting”

Cheers!

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Of Stanley](http://javascriptissexy.com/) (Author)

[October 11, 2013 at 2:25 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-86431) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=86431#respond)

You rock, Alisa. Thanks very much.

I care much about grammar mistakes: I don’t like them so I like to get rid of them promptly. The trouble is that it seems like you can never get rid of **all** grammar mistakes.

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Of Stanley](http://javascriptissexy.com/) (Author)

[October 11, 2013 at 2:51 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-86493) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=86493#respond)

Hi Alisa,

Those were some difficult to catch errors, IMO. Are you a professional proofreader or a grammar guru? I am very curious to know 

1. http://0.gravatar.com/avatar/3c5286fbaee9987999f16d11c3cd3b39?s=60&d=mm&r=g

Eric Oliveira

[October 11, 2013 at 12:04 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-86175) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=86175#respond)

You are a master of the didactics, your articles are simply amazing! Thank you for this excellent work!

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Of Stanley](http://javascriptissexy.com/) (Author)

[October 11, 2013 at 2:28 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-86437) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=86437#respond)

You made my day, Eric. Thanks very much. I am very happy that others can learn and benefit from y articles.

1. http://2.gravatar.com/avatar/249e198b008d977df2f570f137a8efb2?s=60&d=mm&r=g

yougen

[October 28, 2013 at 11:28 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-123002) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=123002#respond)

Hi,  
“the variable assignment overrides the function declaration” example is not exactly accurate.  
function declaration is always precedence variable declaration. What happens is Execution Context has creation stage and code execution stage. Hoisting happens in creation stage. More info here: <http://davidshariff.com/blog/what-is-the-execution-context-in-javascript/>

I modify the code as following:

console.log(typeof myName); // function , NOT string  
var myName = “Richard”;

function myName () {  
console.log (“Rich”);  
}

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Of Stanley](http://javascriptissexy.com/) (Author)

[October 29, 2013 at 3:55 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-124465) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=124465#respond)

Yougen,

The variable assignment always overrides the function declaration. The reason it shows as function in your example code is because first the variables are hoisted (both the function and variable). Since they haven’t been assigned a value yet, the function declaration takes precedence, as I noted in the article. Because function declaration takes precedence over variable declaration.

But then in the execution context when variable assignment occurs, the variable is assign the value “Richard” and it takes precedence over the function declaration. If the function were a function expression of this form: var myName = function () {}, then it would have had precedence over the variable assignment. But because only the variable is assigned a value, it takes precedence.

Here is proof:

|  |  |
| --- | --- |
|  | ​ |
|  | console.log("Before: " + typeof myName); // Before: function​ |
|  | ​var myName = "Richard"; |
|  | ​ |
|  | ​function myName () { |
|  | console.log ("Rich"); |
|  | } |
|  | ​ |
|  | console.log("After: " + typeof myName); // After: string​ |
|  | ​ |

1. http://2.gravatar.com/avatar/207f6ba6d17b31e53c243f39d11ef0cb?s=60&d=mm&r=g

Fon

[November 5, 2013 at 2:21 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-139740) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=139740#respond)

Hey Richard, thanks for you blog! It’s an amazing learning material.

I want to share with everybody here about my experience of running the example code. At the beginning, I try to run the code with jsFiddle, however, the result sometimes will different. And I am confused. But after I try to just create a html file and run the code within the script tag. The result is correct.

So, I think jsFiddle will change the behavior of scope for some reasons. Richard, do you know anything about this?  
This’s an example about “setTimeout Variables are Executed in the Global Scope”, the output of “console.log(this.constantVal \* this.highValue);” is NaN

<http://jsfiddle.net/fangunxs/tUHPY/>

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Of Stanley](http://javascriptissexy.com/) (Author)

[November 10, 2013 at 2:06 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-149534) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=149534#respond)

The reason it did not work was because you use the “this” keyword inside the settimeout function, and the settimeout function changes the scope; it is run from the global scope.

Read on post on JavaScript’s this for more:  
[Understand JavaScript’s “this” With Clarity, and Master It](http://javascriptissexy.com/understand-javascripts-this-with-clarity-and-master-it/)

Here is the fix for the NaN issue with your code:

|  |  |
| --- | --- |
|  | ​ |
|  | ​var highValue = 200; |
|  | ​var constantVal = 2; |
|  | ​var myObj = { |
|  | highValue: 20, |
|  | constantVal: 5, |
|  | calculateIt: function () { |
|  | var self = this; |
|  | setTimeout (function () { |
|  |  |
|  | console.log(self.constantVal \* self.highValue); |
|  | }, 2000); |
|  | } |
|  | } |
|  | ​ |

1. http://2.gravatar.com/avatar/59112f8c907e034a9b7329e9cd179ffe?s=60&d=mm&r=g

matt

[November 8, 2013 at 9:58 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-147108) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=147108#respond)

function showName () {  
setTimeout(function(){console.log (“First Name: ” + name)},0);  
var name = “Ford”;  
console.log (“Last Name: ” + name);  
}

1. http://0.gravatar.com/avatar/ce74eba86698bdde8b69c237bc4cf65c?s=60&d=mm&r=g

Johnzilla

[November 22, 2013 at 4:49 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-163097) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=163097#respond)

Hi. Great article. I found it very helpful for the most part but I do have a two lingering questions.

1) In your first variable hoisting example, you say in a comment that “The reason undefined prints first is because the local variable name was hoisted to the top of the function.”  
I understand why this is clearly indicative of the variable assignment \*not\* being hoisted, but not how this is at all indicative that the declaration \*is\* being hoisted.

2) With regard to functions, In one place you say that the format “var myFunction = function () {}” is called function assignment, but a bit further down you say that this is called function expression.  
I see that both of these are something different from function declaration, which is written as “function myFunction () {}” but not whether there is a difference between function assignment and function expression.

Thank you and I look forward to reading more of your articles.

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Of Stanley](http://javascriptissexy.com/) (Author)

[November 22, 2013 at 3:59 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-163279) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=163279#respond)

1. All variable and function declarations are hoisted: this is simply a fundamental concept in JavaScript.

2. Ah ha. That is a good catch.  
This, “var myFunction = function () {}”, is indeed both a function expression and a function assignment. Technically, it is called a **function expression. So keep that in mind always, because you will encounter and use that kind of expression a lot.**

**Now, whenever you assign a value to a variable, the execution is called a variable assignment. For example, var city = “New York”, is a variable assignment. Similarly the expression, var cityFunc = function () {}, is also a variable assignment. But because it is an assignment of a function to the variable, we call it a function assignment.**

* + - http://0.gravatar.com/avatar/ce74eba86698bdde8b69c237bc4cf65c?s=60&d=mm&r=g

Johnzilla

[November 24, 2013 at 10:28 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-164054) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=164054#respond)

Cool. Thank you! And thanks in general for all that you’ve done with this blog here. 

1. http://2.gravatar.com/avatar/5aa96ae43acbcf05f88abd50f00e7599?s=60&d=mm&r=g

Sivaram

[November 28, 2013 at 7:22 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-165419) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=165419#respond)

Hi Richard,

I am starting to read your tutorial..its very very nice..  
its amazing when u explain everything…

1. http://1.gravatar.com/avatar/74e4ea3fd64e56918e056b399d5f84d8?s=60&d=mm&r=g

Shaikhul

[November 29, 2013 at 7:28 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-165856) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=165856#respond)

Function Declaration Overrides Variable Declaration When Hoisted – OK

But variable assignment overrides the function declaration – not OK.

ex.  
var hoistMe = ‘hoistme';  
function hoistMe () {};  
typeof hoistMe; // “function”

1. http://1.gravatar.com/avatar/74e4ea3fd64e56918e056b399d5f84d8?s=60&d=mm&r=g

Shaikhul

[November 29, 2013 at 7:34 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-165862) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=165862#respond)

ah sorry ‘But variable assignment overrides the function declaration’ is ok while running all the statement together. but if i run each statement in chrome console one by one it doesn’t override. Any explanation for this ?

1. http://1.gravatar.com/avatar/49136486f5f6f7bda81b25103b5f89b6?s=60&d=mm&r=g

haind

[December 13, 2013 at 3:38 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-171392) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=171392#respond)

Hi Richard. I’m just read your article. What I’m confused is I test some code with html file and command line of Firebug:  
console.log(typeof myName);  
var myName = “Richard”;

I show me 2 different results. ‘undefined’ for html file and ‘string’ for command line. I think it ‘s kind of weird. Maybe you can explain for me?

1. http://0.gravatar.com/avatar/ce1390b91425b090a094e309fa8eb758?s=60&d=mm&r=g

Dydrax

[December 23, 2013 at 9:29 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-173967) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=173967#respond)

what about get a value from local variable and stored in global variable

in example i use a js library canvas like easeljs  
there is a lot of function there

function main(){  
var text=["A","B","C"];  
var object;i=0;  
object.on("click",function(A){i++});//assume when click the i is increase  
console.log(text[i]);//still A why ?  
}

* + http://1.gravatar.com/avatar/ad71bdefdcb4247c58f9cc3e27c0dd4d?s=60&d=mm&r=g

[Richard Of Stanley](http://javascriptissexy.com/) (Author)

[January 7, 2014 at 12:46 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-177466) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=177466#respond)

Hi Dydrax, I don’t understand your question. Could you ask the question again with a bit more detail?

1. http://0.gravatar.com/avatar/c3e2b16938f2fc5726dea7f794232885?s=60&d=mm&r=g

saima

[February 11, 2014 at 3:44 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-183456) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=183456#respond)

One of the best articles on the subject.. Many thanks

1. http://2.gravatar.com/avatar/543ed3bd0091513942ba0481ae86d63d?s=60&d=mm&r=g

sagar

[March 23, 2014 at 3:00 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-191281) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=191281#respond)

I am a server side programmer and ignored JavaScript for last 13-14 years. With frameworks like AngularJS and dojo , JS has got lot of recognition.  
I tried my hands on these frameworks recently and was damn scared with JS syntax.  
Thanks a lot bro, you saved me from turning my back forever to JS.  
You deserve a big HUG.

1. http://1.gravatar.com/avatar/10ad67c293d255b10f87b7acb407507b?s=60&d=mm&r=g

Vinayak

[April 20, 2014 at 2:34 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-203208) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=203208#respond)

Hi Richard,  
Thank you very much for the post.. you cleared many things for me with very good examples and neat explanations.

1. http://2.gravatar.com/avatar/8805043049a2b19facda36cd704bc4c0?s=60&d=mm&r=g

Random Dyood

[June 4, 2014 at 1:54 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-241001) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=241001#respond)

Interesting thing about for-loop iterating variable being global.

Having said that, how does one implement a for-loop without polluting one’s global scope with a multitude of distinct or, a lot worse, shared/overlapping iterator variables. Is there a way to implement these loop constructs with a local-scoped variable?

Thanks in advance. Btw, great blog and JS learning material.

Enjoy!

1. http://2.gravatar.com/avatar/5fe9b4ca4a02db9848642e90ec0c93d2?s=60&d=mm&r=g

Steve

[August 7, 2014 at 6:08 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-268087) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=268087#respond)

Hi Richard,

Love the tutorial / lesson style and workable examples btw!  
I just have one question wrt to ” example where you were showing how variable declaration outside functions have global scope and are accessible from the window object, ..however I don’t understand the ” tag declaration / assignment?

…  
// or  
var name; //  
name;

Which you then say is accessible like so:

…  
// or  
console.log(“myName” in window); // true  
console.log(“firstName” in window); // true

..i.e. what’s the significance of the trailing ‘pre’ tag in both declaration and accessibility examples above?

Thanks in advance,  
Steve.

1. http://2.gravatar.com/avatar/5fe9b4ca4a02db9848642e90ec0c93d2?s=60&d=mm&r=g

Steve

[August 8, 2014 at 12:17 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-268376) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=268376#respond)

oops, I just noticed that all the > and < tags have been parsed in some way, ..the instances of two single quotes above ” in my post should read ‘<pre>’  
(not sure if that will have worked either..)

1. http://2.gravatar.com/avatar/2ec6384dd08995428647409db45ee3b9?s=60&d=mm&r=g

Bipul

[October 10, 2014 at 11:43 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-305732) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=305732#respond)

Hi Richard,

Can you please the scope of “myName” in below 2 code snippets. Confused 8-(  
1.  
var myName=”Ram”;  
var myName=function abc()  
{  
console.log(“Bipul”);  
}

console.log(myName());

2.

var myName=”Ram”;  
var myName=function abc()  
{  
console.log(myName);  
}

myName(); // function

1. http://1.gravatar.com/avatar/df32e2962c912939db5920fd111a08a0?s=60&d=mm&r=g

Igor Fedorov

[November 7, 2014 at 4:39 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-305853) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=305853#respond)

Hi,  
I read about hoisting but could somebody explain why

alert(“Hello”);  
var alert = 2;  
alert(“World”);  
does not show anything but

alert(“Hello”);  
alert = 2;  
alert(“World”);  
shows “Hello”, ” World”

window.alert(“Hello”);  
alert = 2;  
window.alert(“World”);  
shows “Hello”, ” World”

What is a difference between “var alert = 2;” and “alert = 2;”

1. http://1.gravatar.com/avatar/77ce77cd7dd3ea52b2fca9da6573c395?s=60&d=mm&r=g

Javier

[December 8, 2014 at 11:21 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-305964) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=305964#respond)

hi, i’ve followed your articles and they’re awesome, but I’m having problems related to the last example of hoisting variables:

var myName = “Javier”;

function myName() {console.log(“My name is Javier”);}

console.log(typeof myName); // this prints “function” i was hoping this prints “string”. I’ve tried in Chrome’s and Mozilla’s webbrowser.and still getting the same result.

1. http://0.gravatar.com/avatar/6bd852a8af53fbf5e8db7f8db08918f2?s=60&d=mm&r=g

Siddhartha

[April 14, 2015 at 1:41 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-306571) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=306571#respond)

Hey,

Nice article.

I’m not sure I understand why the ‘this’ in ‘myObj.calculateIt()’ wouldn’t point to myObj and belong to the global scope?

Regards,  
Siddhartha

1. http://0.gravatar.com/avatar/fa4c72a694f895ff7f221b45d138cb42?s=60&d=mm&r=g

kanhaiya

[July 14, 2015 at 3:50 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-307202) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=307202#respond)

Hi Richard,

In C or OOps –  
function declaration – showName() // only call the method  
function definition – function showName () { } // only define function

But in Javascript:-  
function declaration :- function showName () { }

Question is why function declaration is different as per c language or oops?

Thanks,  
Kanhaiya

1. http://0.gravatar.com/avatar/cea8b13addf74aeea55a75996ffb2799?s=60&d=mm&r=g

Ravindra

[July 23, 2015 at 11:31 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-307229) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=307229#respond)

I am unable to execute this code by simply copy & paste. Due to some encoding issue, I am getting below error

Uncaught SyntaxError: Unexpected token ILLEGAL

One invisible special characters has been added at the first character for every line. Even dos2unix is unable to fix the issue. Please let me know how to fix this issue

1. http://0.gravatar.com/avatar/3ee05bfad1a1d241eeb9620ce7e4f498?s=60&d=mm&r=g

Naveen

[August 18, 2015 at 2:36 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-307307) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=307307#respond)

Hi Richard,

/ The use of the “this” object inside the setTimeout function refers to the Window object, not to myObj​  
​  
​var highValue = 200;  
​var constantVal = 2;  
​var myObj = {  
highValue: 20,  
constantVal: 5,  
calculateIt: function () {  
setTimeout (function () {  
console.log(this.constantVal \* this.highValue);  
}, 2000);  
}  
}  
​  
​// The “this” object in the setTimeout function used the global highValue and constantVal variables, because the reference to “this” in the setTimeout function refers to the global window object, not to the myObj object as we might expect.​

myObj.calculateIt(); // 400​  
​// This is an important point to remember.

WHY “this” IN THE SETTIMEOUT FUNCTION REFERS TO THE GLOBAL WINDOW OBJECT.  
As in your expalanation about “this” , you mentioned the calling object properties will be manipulated.  
How is it different here ?????????

1. http://0.gravatar.com/avatar/fc35d95890ce715ceb18c78425bc0c90?s=60&d=mm&r=g

Deepal

[October 5, 2015 at 3:14 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-307424) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=307424#respond)

I have seen many tutorials website, but the clarity with which you explain stuff is awesome. Keep up the good work.

1. http://0.gravatar.com/avatar/f84cd7a0afb82390585bc3b067a508e7?s=60&d=mm&r=g

Himanshu

[December 16, 2015 at 7:00 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-307724) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=307724#respond)

Hello Richard,

Thanks for the nice & informative article(s)… 

I ran this in FireFox/Chrome’s console:

function showName () {  
console.log (“First Name: ” + name);  
console.log (“Last Name: ” + name);  
}  
showName ();  
shouldn’t I have got a “Reference Error” for variable name here as I did not have a var name statement?  
I saw this instead

First Name:  
Last Name:  
undefined

When I changed variable name to “Tname” then it showed a Reference Error but why not in the case “name”?

Please clarify.

Regards  
Himanshu

1. http://1.gravatar.com/avatar/a74d009c90ac5790687b7ba78f209d2c?s=60&d=mm&r=g

[Artas](http://twitter.com/artasbartas)

[January 23, 2016 at 2:37 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-307821) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=307821#respond)

Thanks for such an informative article. I feel like I understand the variable scope much better now that I read your blog post. Looking forward to reading more of this stuff.

1. http://1.gravatar.com/avatar/d29ca1914bb5b81ae45def6701c82bee?s=60&d=mm&r=g

sasmita

[January 25, 2016 at 12:16 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-307824) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=307824#respond)

Hi, i want to know .suppose within the function we don’t use “var” keyword for creating a variable.It will be automaically global variable.what will happen if a local variable become a global. This question was asked me in interview. plz let clear abt this question.

1. http://1.gravatar.com/avatar/a4fe5e87784bfd539b9b3aecf3ab57c6?s=60&d=mm&r=g

Balasubramanian

[February 3, 2016 at 12:16 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-307846) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=307846#respond)

var highValue = 200;  
​var constantVal = 2;  
​var myObj = {  
highValue: 20,  
constantVal: 5,  
calculateIt: function () {  
setTimeout (function () {  
console.log(this.constantVal \* this.highValue);  
}, 2000);  
}  
}  
​  
​// The “this” object in the setTimeout function used the global highValue and constantVal variables, because the reference to “this” in the setTimeout function refers to the global window object, not to the myObj object as we might expect.​  
​  
myObj.calculateIt(); // 400​  
​// This is an important point to remember

this is wrong the this.constantVal and this.highValue is not accesible directly inside setTimeout

1. http://1.gravatar.com/avatar/a4fe5e87784bfd539b9b3aecf3ab57c6?s=60&d=mm&r=g

Balasubramanian

[February 3, 2016 at 1:06 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-307847) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=307847#respond)

Hello Richard,

Thanks for the nice article(s)… 

I ran this in FireFox/Chrome’s console:

​var highValue = 200;  
​var constantVal = 2;  
​var myObj = {  
highValue: 20,  
constantVal: 5,  
calculateIt: function () {  
setTimeout (function () {  
console.log(this.constantVal \* this.highValue);  
}, 2000);  
}  
}  
myObj.calculateIt();

I have got a “NaN” as a result.

When I stored this.constantVal and this.highValue inside calculateIt function and using those variables inside the setTimeOut function producing the results.

Please clarify.

Regards  
Balasubramanian. K

1. http://2.gravatar.com/avatar/82bcc2992ffd975777636dca17e02d83?s=60&d=mm&r=g

[krishna](http://java.meritcampus.com/core-java-topics/scope-of-variables-in-same-block)

[February 10, 2016 at 5:23 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-307863) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=307863#respond)

for the concept of scope of variables follow the link:  
<http://java.meritcampus.com/core-java-topics/scope-of-variables-in-same-block>

1. http://2.gravatar.com/avatar/b591350a198f2f87b875c5b07257a368?s=60&d=mm&r=g

Thabo

[February 23, 2016 at 4:14 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-307898) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=307898#respond)

Hi author,thank you, but i don’t understand why the variables declared in the for loop are accessible globally, i understood that JavaScript does not have the “Block scope” even if it has the braces(curly brackets), but a function also has braces but it has a block scope(you cannot access its variables outside that scope), how is a function different to the for loop? in respect to the block-scope part.

1. http://1.gravatar.com/avatar/79895feae95d0cbd46700129b26d7de5?s=60&d=mm&r=g

Prasanna

[February 23, 2016 at 6:29 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-307900) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=307900#respond)

Hello Himanshu,  
By now you might have figured out by name is not throwing exception.If not here is why,

browsers have global variable name ,which indicates the name of the current browser window.So it does not show throw error!!!.

1. http://1.gravatar.com/avatar/79895feae95d0cbd46700129b26d7de5?s=60&d=mm&r=g

Prasanna

[February 23, 2016 at 6:30 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-307901) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=307901#respond)

Hello Himanshu,  
By now you might have figured out why “name” is not throwing exception.If not here is why,

browsers have global variable name ,which indicates the name of the current browser window.So it did not throw error!!!.

1. http://1.gravatar.com/avatar/4b866eba55df2cbd34a5e1996e1088c8?s=60&d=mm&r=g

[Simran](http://www.yourtechchick.com/)

[March 5, 2016 at 6:02 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-307929) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=307929#respond)

Nice article. Here’s a little more to add to javascript hoisting :). <http://www.yourtechchick.com/javascript/what-is-hoisting-in-javascript-javascript-hoisting-explained/>

* + http://0.gravatar.com/avatar/f84cd7a0afb82390585bc3b067a508e7?s=60&d=mm&r=g

Himanshu

[March 26, 2016 at 1:51 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-307979) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=307979#respond)

Nicely written.. 

1. http://2.gravatar.com/avatar/e602eb1d74888d506d916a8d26d030a2?s=60&d=mm&r=g

Caio Mathielo

[March 21, 2016 at 12:46 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-307974) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=307974#respond)

Hello Richard! Great article, extremely detailed and very simple to read/understand. Unfortunately some information is no longer true – i.e. “Unlike most programming languages, JavaScript does not have block-level scope (variables scoped to surrounding curly brackets);”.

The “let” statement (<https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Statements/let>) was implemented in JavaScript 1.7 and it specifically allows the declaration of block-level variables. Would be really nice to see the article updated as it looks like many people end up here when googling for JS scope!

Cheers!

1. http://0.gravatar.com/avatar/f84cd7a0afb82390585bc3b067a508e7?s=60&d=mm&r=g

Himanshu

[March 26, 2016 at 1:52 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-307980) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=307980#respond)

A good point out..thanks.. 

1. http://0.gravatar.com/avatar/07afe186e3fdcb82e337ec509e7733bc?s=60&d=mm&r=g

Eazie

[April 28, 2016 at 1:16 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-308054) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=308054#respond)

Love your article, simple but detailed, thumbs up.

1. http://1.gravatar.com/avatar/d305220abc29724f5fd3dc2c8ed4a72f?s=60&d=mm&r=g

Devdutta Natu

[October 12, 2016 at 3:20 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-308361) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=308361#respond)

Hi Richard,

I tried to hoist the very last thing where is mentioned that :-

It is important to note that function expressions, such as the example below, are not hoisted.

var myName;  
var myName = function () {  
console.log (“Rich”);  
}

but still by using the typeof operator i’m getting the same result as function itself.

1. http://2.gravatar.com/avatar/be6bb241b03442fdeffa5758a2beec1d?s=60&d=mm&r=g

Ganesan Natarajan

[October 14, 2016 at 4:50 pm](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-308369) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=308369#respond)

sorry .. wrong comment on the post. i meant to post it here : <http://javascriptissexy.com/understand-javascript-closures-with-ease/#comment-308368>

1. http://1.gravatar.com/avatar/aec76683aac6b0762ff9e28f7b7c8fc7?s=60&d=mm&r=g

Suganya

[February 12, 2018 at 2:04 am](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/#comment-309029) / [Reply](http://javascriptissexy.com/javascript-variable-scope-and-hoisting-explained/?replytocom=309029#respond)

Is there an article about Javascript Promise Concept? Kindly share the link.