在学习默认参数的时候发现一个bug

**let** foo **=** 'outer';

**function** bar(func **=** x **=>** foo) {

**let** foo **=** 'inner';

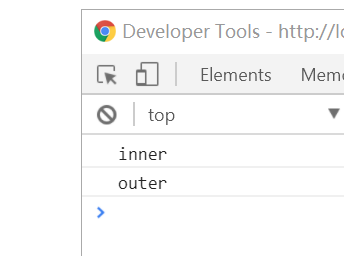
console.log(func()); }

bar(); *//outer*

Babel转义出来

'use strict';  
  
*var* foo = 'outer';  
  
*function bar*() {  
 *var* func = arguments.length > 0 && arguments[0] !== undefined ? arguments[0] : *function* (x) {  
 *return* foo;  
 };  
  
 *var* foo = 'inner';  
 console.*log*(func());  
}  
  
*bar*(); *//inner*

实测



Babel的例子我们很容易理解，有点类似于闭包的概念

Es6的请参考https://www.zhihu.com/question/60394151/answer/176834242

If default value parameter initializers exist, a second [Environment Record](https://link.zhihu.com/?target=http://www.ecma-international.org/ecma-262/7.0/%23sec-environment-records" \t "https://www.zhihu.com/question/60394151/answer/_blank) is created for the body declarations.

A separate Environment Record is needed to ensure that closures created by expressions in the formal parameter list do not have visibility of declarations in the function body.

上面说的意思就是默认初始化是在一个独立的闭包里面初始化而bebal把这个区域放到了内部。

Es6还增加了一个特性就是

function containsAll(haystack, ...needles) {

for (var needle of needles) {

if (haystack.indexOf(needle) === -1) {

return false;

}

}

return true;

}

不定参数，bebal转义是这个样子，从上述看needles是对应第二个参数，其实可以对应第三或者之后的参数

"use strict";

function containsAll(haystack) {

for (var \_len = arguments.length, needles = Array(\_len > 1 ? \_len - 1 : 0), \_key = 1; \_key < \_len; \_key++) {

needles[\_key - 1] = arguments[\_key];

}

var \_iteratorNormalCompletion = true;

var \_didIteratorError = false;

var \_iteratorError = undefined;

try {

for (var \_iterator = needles[Symbol.iterator](), \_step; !(\_iteratorNormalCompletion = (\_step = \_iterator.next()).done); \_iteratorNormalCompletion = true) {

var needle = \_step.value;

if (haystack.indexOf(needle) === -1) {

return false;

}

}

} catch (err) {

\_didIteratorError = true;

\_iteratorError = err;

} finally {

try {

if (!\_iteratorNormalCompletion && \_iterator.return) {

\_iterator.return();

}

} finally {

if (\_didIteratorError) {

throw \_iteratorError;

}

}

}

return true;

}

从实例当中我们发现了Symbol

根据规范，对象属性键只能是string类型或symbol类型,不能是number、boolean，只有string和symbol两种类型。   
<https://blog.csdn.net/neweastsun/article/details/71309317>

《symbol.docx》