ES6函数允许传入默认值，函数参数是一单独作用域，函数内部也是一单独作用域，但是在参数和函数内部之间浏览器厂商处理的有问题。

var x = 1;

function foo(x, y = function() { x = 2; }) {

var x = 3;

y();

console.log(x);

}

Chrome64下

foo() // 3

x // 1

firefox48下

foo() // 2

x // 1

主要的差异在与y = function() { x = 2; }执行后，x改变的是参数作用域还是函数内部作用域。  
Chrome是放弃了兼容ES5，但是firefox做了兼容ES5处理。

function(x,y){

var x = x;

var y = y;

}

ES6下参数里的变量不能在函数内部重新用let和const定义，但是var没问题。  
2020-12-10备注

调用foo()时的堆栈信息

foo函数块内：{

this: Window,//指向全局window

x: 3 //x为函数内变量，由undefined变为3

},

foo:{

arguments: Arguments,

x: 2, //y = function() { x = 2; } y函数创建新的堆栈，上层为foo，参数

//x由undefined变为2，可以理解为实参的变化

y: y()

},

window:{

foo: foo(x,y),

x:1

}

实参（实际参数）

真实传给函数的参数，叫实参。实参可以是：常量、变量、表达式、函数等。无论实参是何种类型的量，在进行函数调用时，它们必须有确定的值，以便把这些值传送给形参。

形参（形式参数）

形式参数是指函数名后括号中的变量。因为形式参数只有在函数被调用的过程中才实例化（分配内存单元），所以叫形式参数。形式参数当函数调用完成之后就自动销毁了。因此形式参数只在函数中有效。