# let和const命令

## let命令

**不存在变量提升**

//var 的情况

console.log(foo); //输出undefined

var foo = 2;

//let的情况

console.log(bar)； // 报错RefernceError

let bar = 2;

**暂时性死区(TDZ)**

只要块级作用域内存在let命令，它所声明的变量就“绑定”这个区域，不再受外部的影响。

var tmp = 123;

if(true){

tmp = “abc”；//RefernceError

let tmp;

}

if（true）

//TDZ开始

tmp = “abc”; //RefernceError

console.log(tmp)；//RefernceError

let tmp; //TDZ结束

console.log(tmp); //undefined

tmp = 123;

console.log(tmp)； //123

}

**不允许重复声明**

function(){

let a = 10;

let a = 1;

}

//报错

function(){

let a = 10;

let a = 1;

}

function func(arg){

let arg;//报错

}

function func（arg）

{

{

let arg;//不报错

}

}

**块级作用域**

var tmp = new Date();

function f(){

console.log(tmp)；

if（false）{

var tmp = “hello world”；

}

}

f(); //undefined

**ES6的块级作用域**

**块级作用域与函数声明**

考虑到环境导致的行为差异太大，应该避免在块级作用域内声明函数。如果确实需要，也应该写成函数表达式，而不是函数声明语句。

//函数声明语句

{

let a = “secret”；

function f（）{

return a;

}}

//函数表达式

{

let a = “secret”;

let f = function(){

return a;

};

}

**do表达式**

本质上，块级作用域就是一个语句,将多个操作封装在一起，没有返回值

{

let t = f();

t=t\*t + 1;

}  
let x = do{

let f = f();

t\*t +1；

}

变量x会得到整个块级作用域的返回值。

**const命令**

const声明一个只读的常量，一旦声明，常量的值就不能改变。

const foo;

//SyntaxError:Missing initializer in const declaration

只声明不赋值，就会报错

与let一样存在暂时性死区，不可重复声明

对于复合型变量，变量名不指向数据，而是指向数据所在的地址，const命令只是保证变量名指向的地址不变，并不保证该地址的数据不变。

const a = [];

a.push(“Hello”)；//可执行

a.length = 0;//可执行

a=[“Dave”]; //报错

如果真的想冻结对象，可以使用Object.freeze方法。

除了将对象本身冻结，对象的属性也应该冻结。下面是一个将对象彻底冻结的函数。

var costantize = (obj)=>{

Object.freeze(obj);

Object.keys(obj).forEach(（key,value）=>{

if(typeof obj[key] === ‘object’){

constantize(obj[key]);

}

})；

};

**顶层对象的属性**

顶层对象，在浏览器指的是window对象，在Node指的是global对象