JS的变量和函数声明都具有hoisting（提升）机制，JS引擎在正式执行之前进行一次预编译，在这个过程中，首先将变量声明及函数声明提升到**当前作用域**的顶端，然后进行接下来的处理。

这种提升机制使用变量及函数可以使用在前、定义在后

例：

函数内的作用域

var foo = 5;

function func(){

if(!foo){

var foo = 10;

}

console.log(foo); //10

}

func();

//预编译之后

var foo = 5;

function func(){

var foo;

if(!foo){

foo = 10;

}

console.log(foo);

}

func();

var foo = 5;

function func(){

if(!foo){

var foo = 10;

}

foo = 20；

console.log(foo); //20

}

func();

全局作用域：

为什么输出10，如果函数外也有一个同名变量为什么不是输出5。

如果在当前作用域（函数内作用域）存在此变量声明，无论它在什么地方声明，引用此变量时就会在当前作用域中查找，不会去外层作用域查找。

只有在当前作用域不存在此变量才到父作用域中查找。

var num = 100;

(function(){

console.log(num);

var num = 200;

})();

===>undefined

//预编译之后

var num = 100;

(function(){

var num;

console.log(num);

num = 200;

})();

如果当前作用域声明了多个同名变量，这变量的同一个标识符都被提升到顶端，其他按顺序执行。

即只提升了变量的声明，变量赋值语句不提升。

JS没有块级作用域，if/for等语句块里的变量也是取外部当前作用域的

声明、定义、初始化

声明宣称一个名字的存在

定义则为这个名字分配存储空间

初始化则为这个名字分配存储空间赋初值

var V = 10; //3个事件：声明一个变量、定义这个变量的数据类型、赋值（初始化）

var v;声明变量v

v = 100;(定义并)初始化变量v

JS是动态语言，其变量并没有固定的类型，其存储空间大小会随初始化与赋值而变化，所以定义在JS中没有什么太多的关系。

函数提升：

定义函数有2种不同的方式

函数声明： 函数表达式：

function func(){

console.log();

}

var func = function(){

console.log();

};//要用;结束

将一个不带名字的函数声明赋值给了一个变量，通过变量可以访问此函数，也存在变量提升效果。

function func(){

var foo;

foo = function foo(){

console.log(2);

}

foo();

var foo = function(){

console.log(1);

};

foo();

foo();

}

func();

function func(){

函数的提升优先权是最高的，它就远被提升到当前作用域最顶端，然后才是函数表达式、变量按顺序执行。

foo();

var foo = function(){

console.log(1);

};

foo();

function foo(){

console.log(2);

}

foo();

}

func();

具名函数表达式：可以为函数表达式指定一个名字，但这并不会使函数表达式成为函数声明。这个具名函数表达式的名字不会进入名字空间，也不会被提升，而只在这个具名函数的作用域内有效。

var foo = function func(){};

JS中一个名字（name）以四种方式进入作用域，其优先级顺序如下：

1、语言内置：所有的作用域中都有this和arguments关键字

2、形式参数：函数的参数在函数作用域都是有效的

3、函数声明：形如function foo(){}

4、变量声明：形如var V / var V = function(){}

从优先级所示，如果一个变量的名字和函数的名字相同，那么函数的名字会覆盖变量的名字，无论函数在代码中的顺序位置如何；但名字的初始化却是按其在代码中书写的顺序进行，不受优先级的影响。

(function(){

var foo;

console.log(typeof foo); //这里foo是函数

function foo(){}

foo = “foo”;

console.log(typeof foo); //这里foo是字符串类型

})();