[JS中的let和var的区别](https://www.cnblogs.com/fly_dragon/p/8669057.html)

ES6 新增了let命令，用来声明局部变量。它的用法类似于var，但是所声明的变量，只在let命令所在的代码块内有效，而且有暂时性死区的约束。

var a = 99; // 全局变量a

f(); // f是函数，虽然定义在调用的后面，但是函数声明会提升到作用域的顶部。

console.log(a); // a=>99, 此时是全局变量的a

function f() {

console.log(a); // 当前的a变量是下面变量a声明提升后，默认值undefined

var a = 10;

console.log(a); // a => 10

}

// 输出结果：

undefined

10

99

**ES6可以用let定义块级作用域变量**

在ES6之前，我们都是用var来声明变量，而且JS只有函数作用域和全局作用域，没有块级作用域，所以{}限定不了var声明变量的访问范围。  
例如：

{

var i = 9;

}

console.log(i); // 9

ES6新增的let，可以声明块级作用域的变量。

{

let i = 9; // i变量只在 花括号内有效！！！

}

console.log(i); // Uncaught ReferenceError: i is not defined

**let 配合for循环的独特应用**

let非常适合用于 for循环内部的块级作用域。JS中的for循环体比较特殊，每次执行都是一个全新的独立的块作用域，用let声明的变量传入到 for循环体的作用域后，不会发生改变，不受外界的影响。看一个常见的面试题目：

for (var i = 0; i <10; i++) {

setTimeout(function() { // 同步注册回调函数到 异步的 宏任务队列。

console.log(i); // 执行此代码时，同步代码for循环已经执行完成

}, 0);

}

// 输出结果

10 共10个

// 这里面的知识点： JS的事件循环机制，setTimeout的机制等

如果把 var改成 let声明：

// i虽然在全局作用域声明，但是在for循环体局部作用域中使用的时候，变量会被固定，不受外界干扰。

for (let i = 0; i < 10; i++) {

setTimeout(function() {

console.log(i); // i 是循环体内局部作用域，不受外界影响。

}, 0);

}

// 输出结果：

0 1 2 3 4 5 6 7 8 9

**let没有变量提升与暂时性死区**

用let声明的变量，不存在变量提升。而且要求必须 等let声明语句执行完之后，变量才能使用，不然会报Uncaught ReferenceError错误。  
例如：

console.log(aicoder); // 错误：Uncaught ReferenceError ...

let aicoder = 'aicoder.com';

// 这里就可以安全使用aicoder

ES6 明确规定，如果区块中存在let和const命令，这个区块对这些命令声明的变量，从一开始就形成了封闭作用域。凡是在声明之前就使用这些变量，就会报错。  
总之，在代码块内，使用let命令声明变量之前，该变量都是不可用的。这在语法上，称为“暂时性死区”（temporal dead zone，简称 TDZ）。

**let变量不能重复声明**

let不允许在相同作用域内，重复声明同一个变量。否则报错：Uncaught SyntaxError: Identifier 'XXX' has already been declared

例如：

let a = 0;

let a = 'sss';

// Uncaught SyntaxError: Identifier 'a' has already been declared

**总结**

ES6的let让js真正拥有了块级作用域，也是向这更安全更规范的路走，虽然加了很多约束，但是都是为了让我们更安全的使用和写代码。