# let 基本使用:

1. 用于声明变量，但是声明的变量只在let命令所在代码块 ( 每个{ } ) 内有效
2. 用于循环中作为循环变量， js引擎会记住上一论循环的值，初始化本轮的变量时，就在上一轮的基础上计算的
3. for 循环设置let 循环变量的那部分是父级作用域,循环体内部是单独的子作用域

# let 细节:

1. 不存在变量提升: 用let声明某个变量之前,这个变量是不存在的,如果一用就会报错
2. 存在暂时性死区: 只要进入当前作用域,所要使用的变量就已存在,但是无法访问,只有等到let声明之后,才可以访问变量
3. 只要块级作用域存在let命令,所声明的变量就会绑定这个区域,不再受外部的影响
4. 在相同作用域内, let不允许重复声明, var允许重复声明(且以后面声明为准)
5. 不与顶层对象的属性挂钩

# 注意：

1. Js是可以多次用var声明（变量初始化）同一个变量名的变量
2. 在函数或者区块内，不写var声明直接赋值，会被定义成全局变量

# 块级作用域引出:

1. var 会覆盖上级作用域的变量
2. var用来计数的变量是全局变量
3. ES5 之前只有全局和函数作用域的概念, 但是像if(){},for(){} 这些语句内其实也是一种作用域, 但是当时只有var 让这些作用域变得跟全局一样, 如果用let,则会绑定那一个{}作为块级作用域

所以 ES6 使用let 作为js新增的块级作用域(let 那个变量,那个变量的作用域就在那一个变量的{ }花括号内), 块级作用域就是 = { } + let

# 块级作用域基本使用:

1. let 那个变量,那个变量的作用域就在那一个变量的{ }花括号内
2. 在{}内,如果要声明函数,使用let 和函数表达式来声明一个函数

do表达式的引出: （提案， 现阶段未实现）

1. 块级作用域本质上就是一个语句,将多个操作封装在一起,没有返回值(除非用var)
2. 如果我想把一些值返回呢?

所以ES6新增,在kuai'ji