Cannot be Redeclared

Variables defined with let **can not be redeclared**.

You can not accidentally redeclare a variable declared with let.

With let you can **not** do this:

let x = "John Doe";  
  
let x = 0;

With var you can:

var x = "John Doe";  
  
var x = 0;

Block Scope

Before ES6 (2015), JavaScript had **Global Scope** and **Function Scope**.

ES6 introduced two important new JavaScript keywords: let and const.

These two keywords provide **Block Scope** in JavaScript.

Variables declared inside a { } block cannot be accessed from outside the block:

Example

{  
  let x = 2;  
}  
// x can NOT be used here

Variables declared with the var keyword can NOT have block scope.

Variables declared inside a { } block can be accessed from outside the block.

Example

{  
  var x = 2;  
}  
// x CAN be used here

Difference Between var, let and const

|  |
| --- |
|  |
|  | Scope | Redeclare | Reassign | Hoisted | Binds this |
| var | No | Yes | Yes | Yes | Yes |
| let | Yes | No | Yes | No | No |
| const | Yes | No | No | No | No |

What is Good?

let and const have **block scope**.

let and const can not be **redeclared**.

let and const must be **declared** before use.

let and const does **not bind** to this.

let and const are **not hoisted**.

What is Not Good?

var does not have to be declared.

var is hoisted.

var binds to this.