**JavaScript Let**

Variables defined with let cannot be **Redeclared**

Variables defined with let must be **Declared** before use

Variables defined with let have **Block Scope**

## **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**

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

## **Redeclaring Variables**

Redeclaring a variable using the var keyword can impose problems.

Redeclaring a variable inside a block will also redeclare the variable outside the block:

### **Example**

var x = 10;  
// Here x is 10  
  
{  
var x = 2;  
// Here x is 2  
}  
  
// Here x is 2

Redeclaring a variable using the let keyword can solve this problem.

Redeclaring a variable inside a block will not redeclare the variable outside the block:

### **Example**

let x = 10;  
// Here x is 10  
  
{  
let x = 2;  
// Here x is 2  
}  
  
// Here x is 10

## **Difference Between var, let and const**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Variables** | **Scope** | **Redeclare** | **Reassign** | **Hoisted** | **Binds this** |
| Var | No | Yes | Yes | Yes | Yes |
| Let | Yes | No | Yes | No | No |
| const | Yes | No | No | No | No |

Notes:

1. **let and const**

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**.

**b) var**

var does not have to be declared.

var is hoisted.

var binds to this.

## **Redeclaring**

Redeclaring a JavaScript variable with var is allowed anywhere in a program:

var x = 2;  
// Now x is 2  
  
var x = 3;  
// Now x is 3

With let, redeclaring a variable in the same block is NOT allowed:

var x = 2;   // Allowed  
let x = 3;   // Not allowed  
{  
let x = 2;   // Allowed  
let x = 3;   // Not allowed  
}  
{  
let x = 2;   // Allowed  
var x = 3;   // Not allowed  
}

Redeclaring a variable with let, in another block, IS allowed:

let x = 2;   // Allowed  
{  
let x = 3;   // Allowed  
}  
{  
let x = 4;    // Allowed  
}

## **Let Hoisting**

You can use the variable before it is declared:

carName = "Volvo";  
var carName;

let variable before it is declared will result in a ReferenceError:

JavaScript Const

Variables defined with const cannot be **Redeclared.**

Variables defined with const cannot be **Reassigned**

Variables defined with const have **Block Scope.**

## **Must be Assigned**

JavaScript const variables must be assigned a value when they are declared:

### **Correct**

const PI = 3.14159265359;

### **Incorrect**

const PI;  
PI = 3.14159265359;

## **When to use JavaScript const?**

**Always declare a variable with const when you know that the value should not be changed.**

Use const when you declare:

* A new Array
* A new Object
* A new Function
* A new RegExp

## **Constant Objects and Arrays**

The keyword const is a little misleading.

It does not define a constant value. It defines a constant reference to a value.

Because of this you can NOT:

* Reassign a constant value
* Reassign a constant array
* Reassign a constant object

But you CAN:

* Change the elements of constant array
* Change the properties of constant object

## **Constant Arrays**

You can change the elements of a constant array:

// You can create a constant array:  
const cars = ["Saab", "Volvo", "BMW"];  
  
// You can change an element:  
cars[0] = "Toyota";  
  
// You can add an element:  
cars.push("Audi");

But you can NOT reassign the array:

### **Example**

const cars = ["Saab", "Volvo", "BMW"];  
  
cars = ["Toyota", "Volvo", "Audi"];    // ERROR

## **Constant Objects**

You can change the properties of a constant object:

### **Example**

// You can create a const object:  
const car = {type:"Fiat", model:"500", color:"white"};  
  
// You can change a property:  
car.color = "red";  
  
// You can add a property:  
car.owner = "Johnson";

But you can NOT reassign the object:

### **Example**

const car = {type:"Fiat", model:"500", color:"white"};  
  
car = {type:"Volvo", model:"EX60", color:"red"};    // ERROR

## **Block Scope**

Declaring a variable with const is similar to let when it comes to **Block Scope**.

### **Example**

const x = 10;  
// Here x is 10  
  
{  
const x = 2;  
// Here x is 2  
}  
  
// Here x is 10

Redeclaring an existing var or let variable to const, in the same scope, is not allowed:

var x = 2;     // Allowed  
const x = 2;   // Not allowed  
  
{  
let x = 2;     // Allowed  
const x = 2;   // Not allowed  
}  
  
{  
const x = 2;   // Allowed  
const x = 2;   // Not allowed  
}

Reassigning an existing const variable, in the same scope, is not allowed:

### **Example**

const x = 2;     // Allowed  
x = 2;           // Not allowed  
var x = 2;       // Not allowed  
let x = 2;       // Not allowed  
const x = 2;     // Not allowed  
  
{  
  const x = 2;   // Allowed  
  x = 2;         // Not allowed  
  var x = 2;     // Not allowed  
  let x = 2;     // Not allowed  
  const x = 2;   // Not allowed  
}

Redeclaring a variable with const, in another scope, or in another block, is allowed:

### **Example**

const x = 2;       // Allowed  
{  
  const x = 3;   // Allowed  
}  
{  
  const x = 4;   // Allowed  
}

Using a const variable before it is declared will result in a ReferenceError