ECMAScript 2015

ES2015 introduced two important new JavaScript keywords: let and const.

These two keywords provide **Block Scope** variables (and constants) in JavaScript.

Before ES2015, JavaScript had only two types of scope: **Global Scope** and **Function Scope**.

Global Scope

Variables declared **Globally** (outside any function) have **Global Scope**.

Example

var carName = "Volvo";  
  
// code here can use carName  
  
function myFunction() {  
  // code here can also use carName  
}

**Global** variables can be accessed from anywhere in a JavaScript program.

Function Scope

Variables declared **Locally** (inside a function) have **Function Scope**.

Example

// code here can NOT use carName  
  
function myFunction() {  
  var carName = "Volvo";  
  // code here CAN use carName  
}  
  
// code here can NOT use carName

**Local** variables can only be accessed from inside the function where they are declared.

JavaScript Block Scope

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

Before ES2015 JavaScript did not have **Block Scope**.

Variables declared with the let keyword can have Block Scope.

Variables declared inside a block **{}** can not be accessed from outside the block:

Example

{  
  let x = 2;  
}  
// x can NOT 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

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