The Block Scope and Let Keyword

The Block Scope

Three kinds of scopes:

1. Global Scope
2. Function Scope
3. Block-level Scope

You've learnt about global and function scopes. Now have a look at the block-level scope.

In simple terms, the block-level scope is the one enclosed within curly braces {}. It includes if-else statements, switch conditions, and loops.

Even this is a block too -

```
{
  var c = 10;
   console.log(c) ;
}
```

Accessibility of variables declared using var, let

- Variables declared using **var** can be accessed outside the block-level scope.
- Variables declared using **let** can be accessed only within the block-level scope. An error will be thrown if you try accessing them outside the block-level scope.

How does using let within a block-level scope solves the problem of avoiding global variables?

- If a variable is declared using **let** then it can only be declared once.

  For example,

let c = 3;
console.log(c);
let c = 4;
	let c = 3;
	console.log(c);
	c = 4;

This will throw an error because ‘c’ can not be **declared again.**	This will not throw an error as ‘c’ **can be defined again**, although it can’t be re-declared.

But

var c = 3;
console.log(c);
var c = 4;
	var c = 3;
	console.log(c);
	c = 4;

This will not throw an error because ‘c’ can be **declared again.**	This will not throw an error as ‘c’ **can be defined again as well.**

- **Two ways to solve the problem of global variables**

    1. Suppose two JS files are attached to the same HTML document. Also, global variables with the same names are created in both files using the **let** keyword**.**

        Since all the global variables in both the files will share the same global scope, javascript will throw an error here. As the variables declared using **let** in the first file can not be re-declared in the second file.
        Hence it **won’t allow creating global variables with the same name**.

    2. If you want variables with the same names in both the JS files without having any error, then you can enclose your code within curly braces to make its scope block-level. Then use the **let** keyword to declare your variables within them.