Keyword	Scope	Reassignment	Hoisting	Nested Scopes	Example
`let`	Block	Allowed	No	Yes	`let x = 1; { let x = 2; ole.log(x); } // outputs 2`
`const`	Block	Not allowed	No	Yes	`const x = 1; { const x = 2; ole.log(x); } // outputs 2`
`var`	Function or al	Allowed	Yes	Yes	`var x = 1; { var x = 2; ole.log(x); } // outputs 2`

Here's a brief explanation of each column:

- `Keyword`: The variable declaration keyword.
- `Scope`: The scope of the variable. `Block` means the variable is only accessible within the curly braces where it's declared. `Function or Global` means the variable is accessible throughout the function or the entire program, depending on where it's declared.
- `Reassignment`: Whether or not the variable's value can be reassigned. `Allowed` means it can be reassigned after it's declared. `Not allowed` means it can't be reassigned after it's declared.
- `Hoisting`: Whether or not the variable can be accessed before it's declared. `Yes` means the variable can be accessed before it's declared, but its value will be `undefined`. `No` means the variable can't be accessed before it's declared.
- `Example`: An example of how the keyword can be used in code.

- `Nested Scopes`: Whether or not the variable can be accessed in nested scopes. `Yes` means the variable declared in an outer scope can be accessed in an inner scope, but a variable declared in an inner scope cannot be accessed in the outer scope.

For example, consider the following code:

javascript

Copy code

```
let x = 1; { let y = 2; console.log(x); // Outputs 1 console.log(y); // Outputs 2 }

console.log(x); // Outputs 1 console.log(y); // ReferenceError: y is not defined
```

In this code, `x` is declared outside the block, and `y` is declared inside the block. Both `x` and `y` can be accessed within the block because the block is a nested scope within the global scope. However, `y` cannot be accessed outside the block because it is not defined in the global scope.

Keep in mind that these are general rules, and there can be exceptions and nuances to how these keywords behave in different situations.