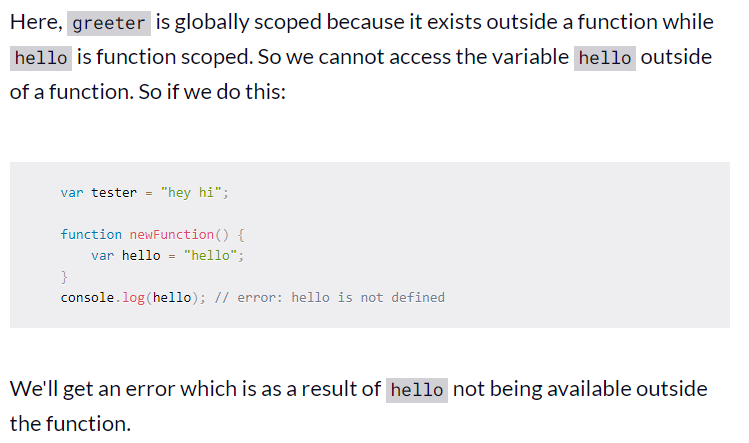
**Var**

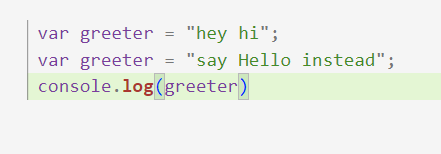
Before the advent of ES6, var declarations ruled.

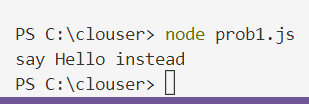
**Scope of var**

**Scope** essentially means where these variables are available for use. var declarations are globally scoped or function/locally scoped.

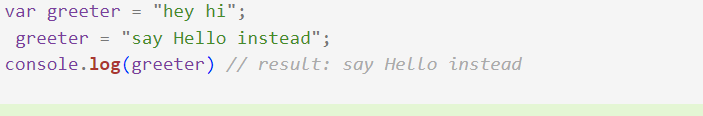


**var variables can be re-declared and updated**

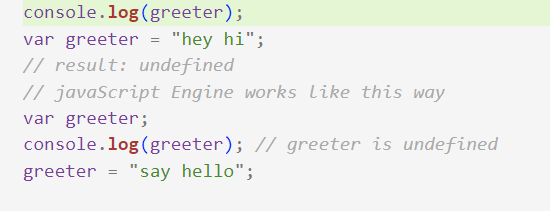




We can writer this way too



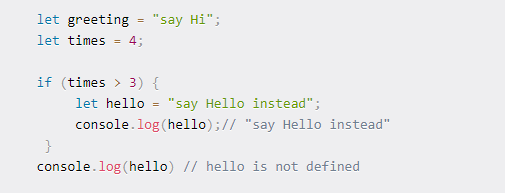
**Hoisting of var**

var variables are hoisted to the top of their scope and initialized with a value of undefined. 

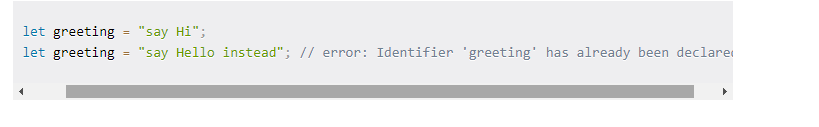
**Problem with var**

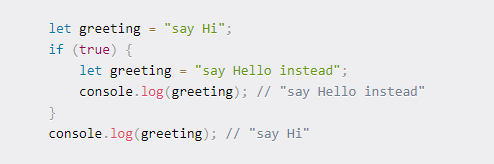


So, since times > 3 returns true, greeter is redefined  to "say Hello instead".If you have used greeter in other parts of your code, you might be surprised at the output you might get. This will likely cause a lot of bugs in your code. This is why let and const are necessary. If it would be hey hi with let. Because of "say Hello instead" inside the block scope. Although times > 3 returns true, it wouldn’t print it.



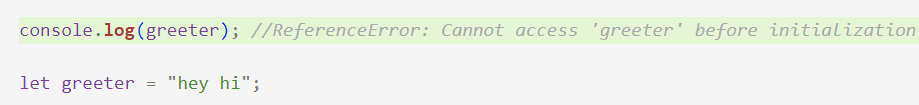
**let can be updated but not re-declared**

****

****

**because both instances are treated as different variables since they have different scopes.**

**Hoisting of let**

****

**Just like var, let declarations are hoisted to the top. Unlike var which is initialized as undefined, the let keyword is not initialized. So if you try to use a let variable before declaration, you'll get a Reference Error.**

**Const**

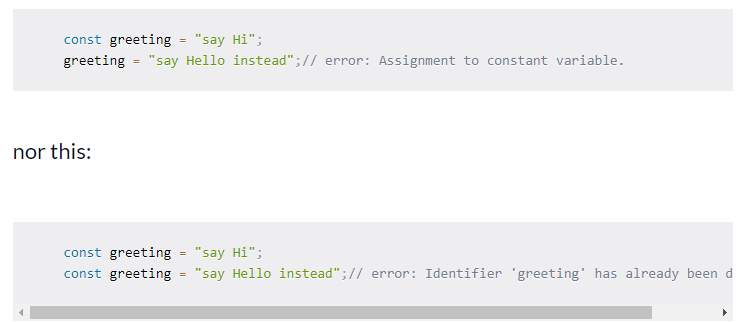
Variables declared with the const maintain constant values. const declarations share some similarities with let declarations.

const declarations are block scoped

Like let declarations, const declarations can only be accessed within the block they were declared.

**const cannot be updated or re-declared**

const doesn’t work with same name variable in same scope like let keyword.

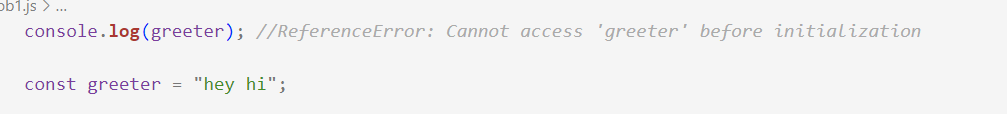




We see at above example that it works at block scope and same name works if it is different scope like let keyword.

**Hoisting of const**

Just like let, const declarations are hoisted to the top but are not initialized.thus, we get *ReferenceError*



Summery

