Variables in JavaScript

by:

- Saepul Bahri
- Wira Harsa
- Bul Joseph
- Amaradipta

Variables are used to store information to be referenced and manipulated in a computer program

Scope refers to the area where a function or variable is visible and accessible to other code

Variable in Javascript:

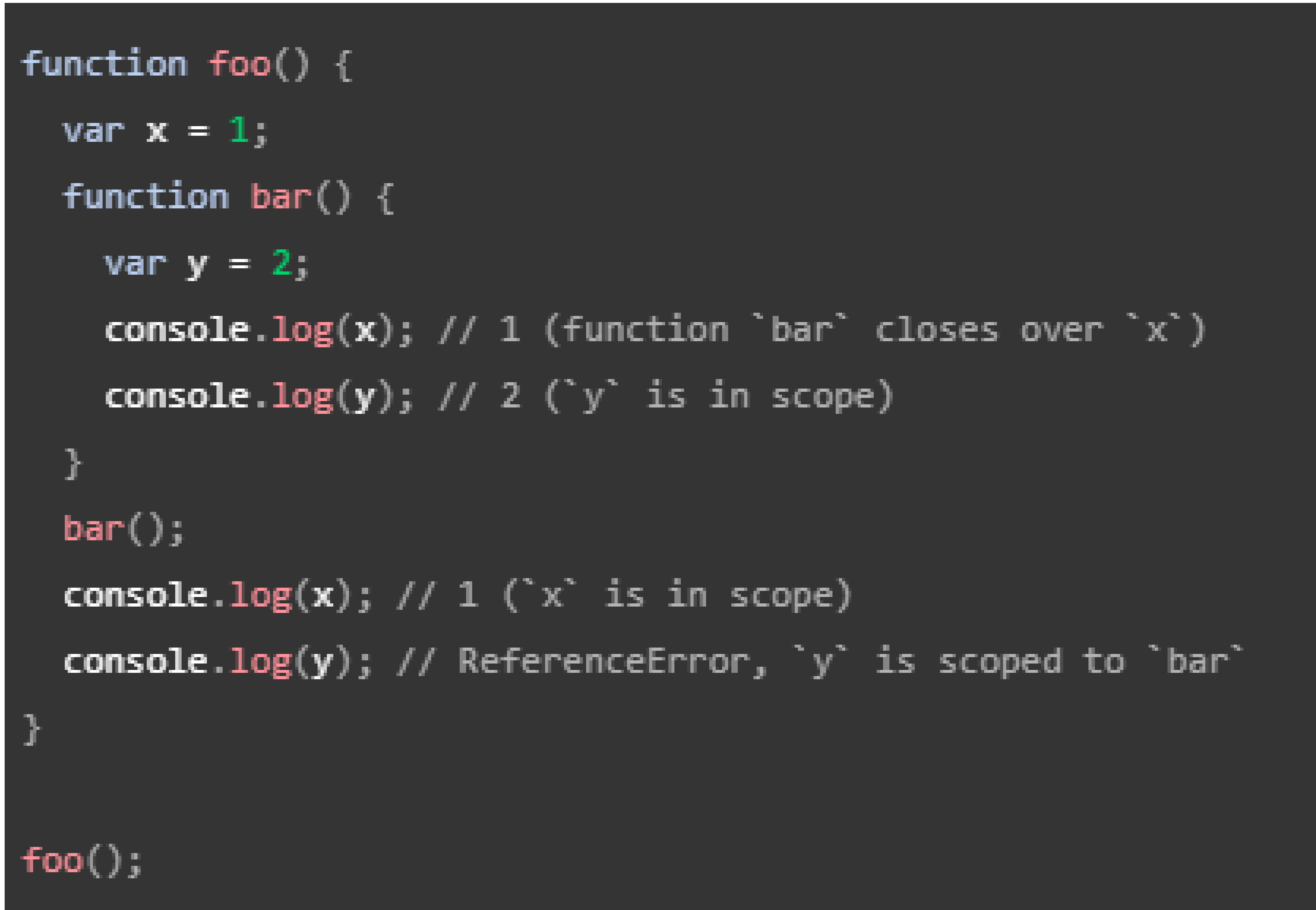
- Var
- Let
- Const

Var:

The var statement declares function-scoped or globally-scoped variables, optionally initializing each to a value

Variable declared with ‘var’ can be redeclared and the value can be changd within its scope.

For example:



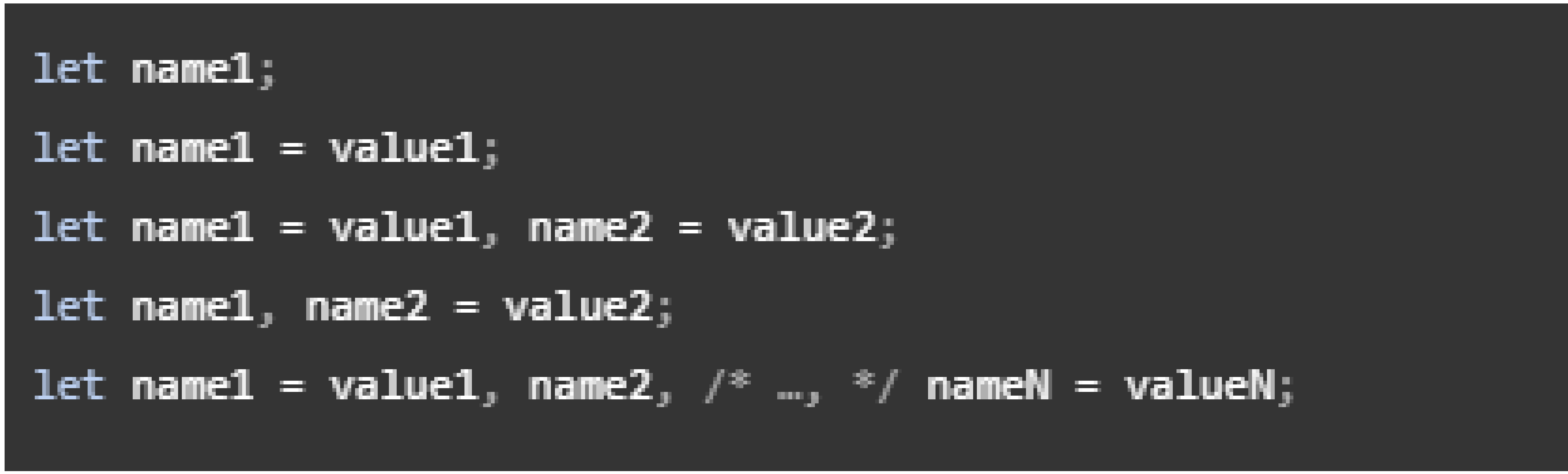
```
function foo() {
  var x = 1;
  function bar() {
    var y = 2;
    console.log(x); // 1 (function `bar` closes over `x`)
    console.log(y); // 2 (`y` is in scope)
  }
  bar();
  console.log(x); // 1 (`x` is in scope)
  console.log(y); // ReferenceError, `y` is scoped to `bar`
}

foo();
```

Let:
The let declaration declares re-assignable, block-scoped local variables, optionally initializing each to a value.

Variable declared with ‘let’ can be updated but cannot be redeclared.

For example:



```
let name1;
let name1 = value1;
let name1 = value1, name2 = value2;
let name1, name2 = value2;
let name1 = value1, name2, /* …, */ nameN = valueN;
```

‘Let’ common use case:

- Block statement
- switch statement
- try...catch statement
- Body of one of the for statements, if the let is in the header of the statement
- Function body
- Static initialization block

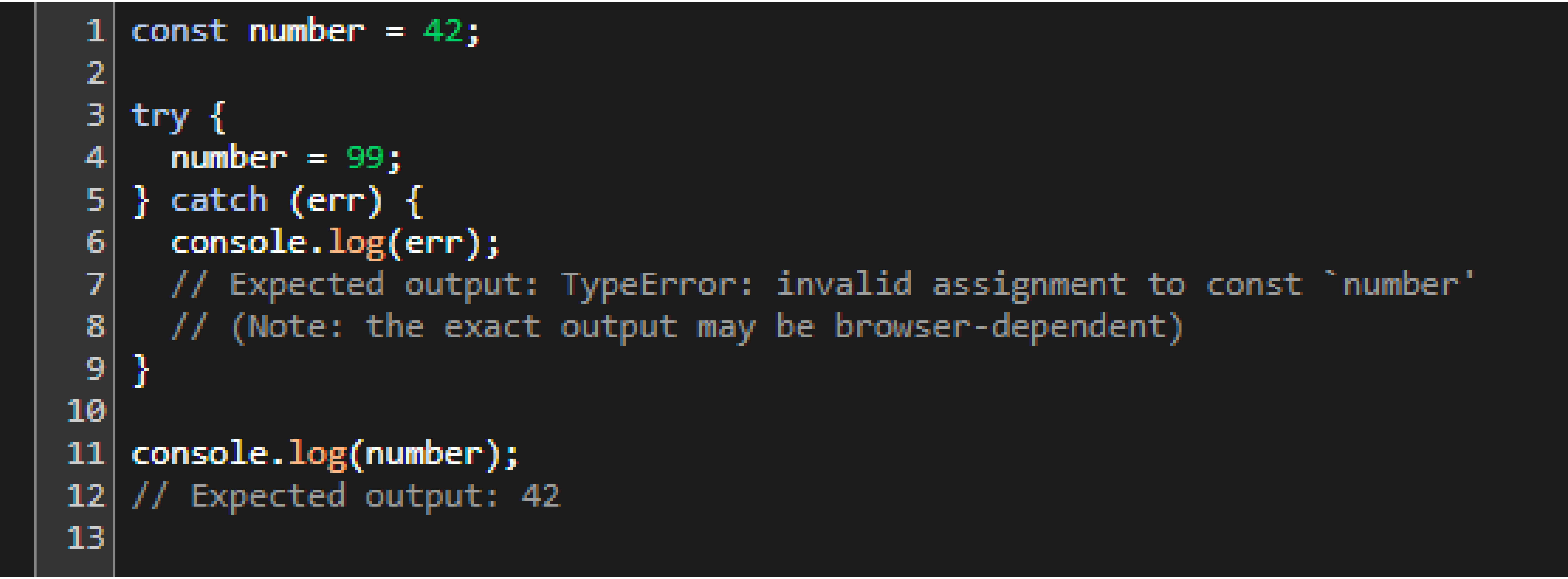
Compared with var, let declarations have the following differences:

- **let declarations are scoped to blocks as well as functions.**
- **let declarations can only be accessed after the place of declaration is reached. For this reason, let declarations are commonly regarded as non-hoisted.**
- **let declarations do not create properties on globalThis when declared at the top level of a script.**
- **let declarations cannot be redeclared by any other declaration in the same scope.**
- **let begins declarations, not statements. That means you cannot use a lone let declaration as the body of a block (which makes sense, since there's no way to access the variable).**

Const:

The ‘const’ declaration declares block-scoped local variables. The value of a constant can't be changed through reassignment using the assignment operator, but if a constant is an object, its properties can be added, updated, or removed.

For example:



```javascript
const number = 42;

try {
  number = 99;
} catch (err) {
  console.log(err);
  // Expected output: TypeError: invalid assignment to const `number'
  // (Note: the exact output may be browser-dependent)
}

console.log(number);
// Expected output: 42

```

- **const declarations are scoped to blocks as well as functions.**
- **const declarations can only be accessed after the place of declaration is reached. For this reason, const declarations are commonly regarded as non-hoisted.**
- **const declarations do not create properties on globalThis when declared at the top level of a script.**
- **const declarations cannot be redeclared by any other declaration in the same scope.**
- **const begins declarations, not statements. That means you cannot use a lone const declaration as the body of a block (which makes sense, since there's no way to access the variable).**

So just in case you missed the differences, here they are:

- **var declarations are globally scoped or function scoped while let and const are block scoped.**
- **var variables can be updated and re-declared within its scope; let variables can be updated but not re-declared; const variables can neither be updated nor re-declared.**
- **They are all hoisted to the top of their scope. But while var variables are initialized with undefined, let and const variables are not initialized.**
- **While var and let can be declared without being initialized, const must be initialized during declaration.**