6.7 Scope and the global object

The var keyword and scope

In addition to declaring variables with `let`, a variable can be declared with the ***var*** keyword. Ex: `var x = 6;` declares the variable `x` with an initial value of 6. When JavaScript was first created, `var` was the only way to declare a variable. The `let` keyword was added to JavaScript in 2015.

Both `let` and `var` declare variables but with differing scope. A JavaScript variable's ***scope*** is the context in which the variable can be accessed.

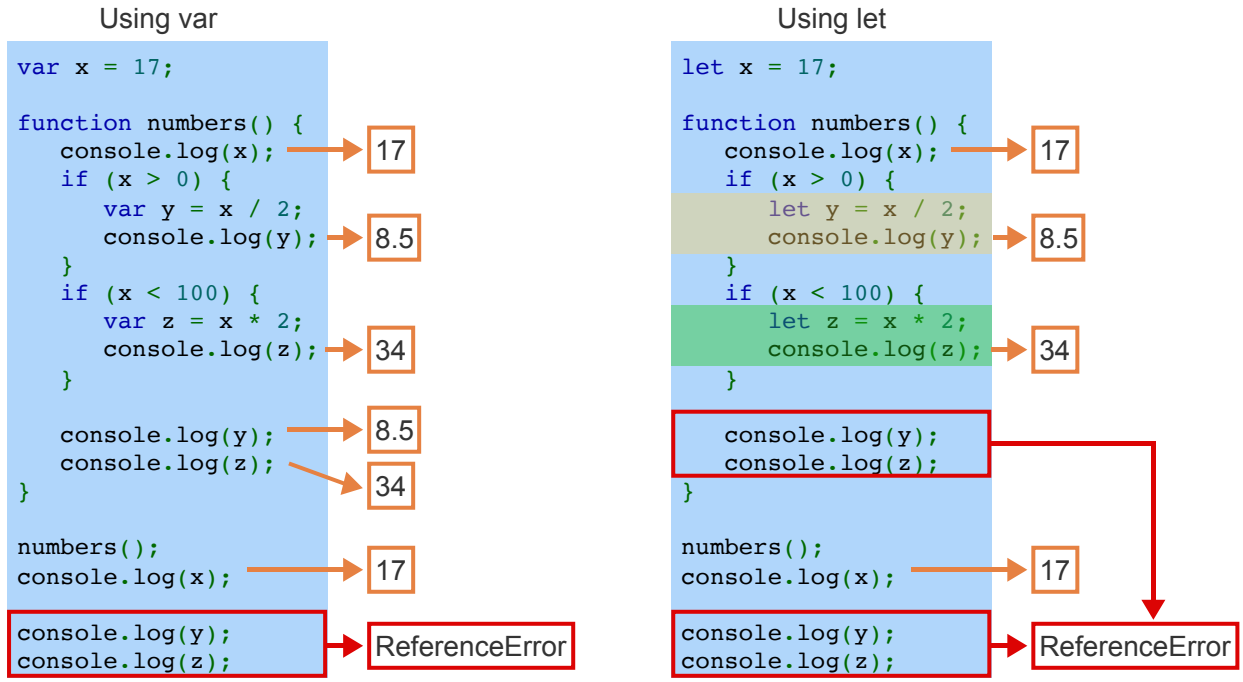
A variable declared inside a function has ***local scope***, so only the function that defines the variable has access to the ***local variable***. A variable declared outside a function has ***global scope***, and all functions have access to a ***global variable***.

A variable declared inside a function with `var` has ***function scope***: the variable is accessible anywhere within the function, but not outside. A variable declared inside a function with `let` has ***block scope***: the variable is accessible only within the enclosing pair of braces.

A variable declared using `var` or `let` that is not inside a function creates a global variable that is accessible from anywhere in the code.

PARTICIPATION ACTIVITY 6.7.1: var vs. let scoping.

Start 2x speed



1. var x = 17; declares x with global scope. x is accessible everywhere, so each console.log(x) statement logs x as 17.
2. The var y declaration exists inside the numbers() function. So both console.log(y) statements inside the function log y as 8.5.
3. Similarly, the var z statement is inside the function, so both console.log(z) statements inside the function log z as 34.
4. y and z are not accessible outside the numbers() function. The console.log() statements that exist outside the function throw a ReferenceError when executed.
5. Code that uses let instead of var has similar behavior for the global variable x.
6. The first log statement for y is in y's scope (yellow), and the first log statement for z is in z's scope (green). So, 8.5 and 34 are logged.
7. All remaining calls to log y or z are out of scope and throw a ReferenceError.

Feedback?

PARTICIPATION ACTIVITY 6.7.2: Local and global variables.

Refer to the code below.

```
function multiplyNumbers(x, y) {
   var answer = x * y;
   return answer;
}

var z = multiplyNumbers(2, 3);
console.log(answer);
```

1) The `answer` variable has _____ scope.
   - global
   - local (selected)

Correct

The `answer` variable is declared as a local variable inside `multiplyNumbers` and therefore has local scope.

2) The `z` variable has _____ scope.
   - global (selected)
   - local

Correct

The `z` variable is not declared inside a function and therefore has global scope.

3) The `console.log(answer);` line _____.
   - logs 6
   - logs undefined
   - throws a ReferenceError (selected)

Correct

The local variable `answer` is not accessible outside the multiplyNumbers() function, so a ReferenceError is thrown.

Feedback?

PARTICIPATION ACTIVITY 6.7.3: var vs. let scoping.

1) In the code below, which variables are in scope on the blank line?

```
function oneToTen() {
   let a = 1;
   for (var i = a; i <= 10; i++) {

console.log(i);
      _____
   }
}
```

- a only
- i only
- both a and i

Correct

The function's braces are the block scope for variable `a`. `i` is declared with `var` and is also in scope anywhere inside the `oneToTen()` function.

2) After calling functions `f1()` and `f2()` in the code below, which variable has global scope?

```
function f1() {
   let x = 100;
}

function f2() {
   var y = 200;
}
```

- x
- y
- neither

Correct

Both `x` and `y` are only available in functions `f1()` and `f2()`, respectively. Neither has global scope.

3) On the blank line in the code below, variable `k` _____.

```
function sumOfSquares() {
   let sum = 0;
   for (let i = 1; i < 5; i++) {
      let j = i * i;
      sum += j;
      var k = sum;
   }
   _____
}
```

- is out of scope and not accessible
- is in scope and has a value of 30
- is in scope, but has an undefined value

Correct

`k` is in scope anywhere inside the function. The most recent assignment of `sum` to `k` leaves `k` with a value of 30 at the end of the loop.

Feedback?

Global variables and the global object

Before developer code is run, JavaScript implementations create ***the global object***: an object that stores certain global variables, functions, and other properties. When running JavaScript code in a web browser, global variables are usually assigned as properties to the global `window` object. Therefore, a global variable called `test` is accessible as `window.test`.

Developers must be careful when assigning global variables, because a global variable could replace an existing window property. Ex: `window.location` contains the URL the browser is displaying. Assigning `location = "Texas"` causes the web browser to attempt to load a web page with the URL "Texas", which likely does not exist.

Three cases exist when assigning to a global variable X:

- X has been declared with `var`, in which case a property named "X" is added to the global object.
- X has been declared with `let`, in which case a property named "X" is not added to the global object, but X is still accessible from anywhere in the code.
- X has not been declared with `var` or `let`, in which case the variable becomes a property of the global object, even if assigned to inside a function.

Figure 6.7.1: Example with accidental global variable.

```
function calculateTax(total) {
   // Missing "var" so tax becomes a global variable!
   tax = total * 0.06;
   return tax;
}

var totalTax = calculateTax(10);

// tax is accessible because tax is global
console.log(tax);
```

```
0.6
```

Feedback?

Good practice is to always declare variables used in functions with `var` or `let`, so the variables do not become global.

PARTICIPATION ACTIVITY 6.7.4: Variable scope and functions.

1) What is output to the console?

```
function addNumber(x) {
   sum += x;
}

let sum = 0;
addNumber(2);
addNumber(5);
console.log(sum);
```

7

Check Show answer

Correct

7

sum = 0 + 2 + 5 = 7

2) What is output to the console?

```
function multiplyNumbers(x, y) {
   answer = x * y;
   return answer;
}

var z = multiplyNumbers(2, 3);
console.log(answer);
```

6

Check Show answer

Correct

6

The global variable `answer` contains 2 * 3 = 6.

3) If `window` is the global object, what is the value of `window.result` after running the following code?

```
function subtractNumbers(x, y) {
   result = x - y;
}

var a = subtractNumbers(7, 6);
var b = subtractNumbers(11, 3);
var c = subtractNumbers(9, 1);
```

8

Check Show answer

Correct

8

`result` is not declared with `let` or `var`, so assigning to `result` is the same as assigning to `window.result`. The most recent assignment comes from the call to `subtractNumbers(9, 1)`, which assigns 9 - 1 = 8 to `window.result`.

Feedback?

Exploring further:

- var (MDN)
- Global object (MDN)

How was this section? Provide section feedback