SOEN 287

Chapter 4: JavaScript (3)

Dr. Yuhong Yan
CSE, Concordia University
Winter, 2023

Topics

- History
- Basic Syntax

This set of sides is modified from the slides accompanied the textbook.

`let` – variable declaration

- The `let` keyword was introduced in ES6 (2015)
- Variables defined with `let` cannot be redeclared.
- Variable defined with `let` must be declared before use.
- Variable defined with `let` have block scope

Variables defined with **`let`** cannot be redeclared

```
let x = "John Doe";

let x = 0;

// SyntaxError: 'x' has already been declared
```

- With var you can

```
var x = "John Doe";

var x = 0;
```

`let` variable should be redeclared before use

```
let x;

x = 0;
```

- With var you can

```
x = "John Doe";

var x;
```

Block scope from ES6 (2015)

- With **`let`** and **`const`**, you can have block scope
- Block scope: variables declared inside a { } cannot be accessed from outside

```
{
  let x = 2;
}
// x can NOT be used here
```

- With var, there is no block scope (a pitfall), only global and function scope

```
{
  var x = 2;
}
// x CAN be used here
```

`Const` – variable declaration

- The `const` keyword was introduced in ES6 (2015)
- Variables defined with `const` cannot be redeclared.
- Variable defined with `const` cannot be reassigned.
- Variable defined with `const` have block scope.

Example on var's global and function scope:

```
var scope = "global scope";
function checkscope(){
    var scope = "local scope";
    function nested() {
        var scope = "nested scope";
        alert("1-" + scope); //prints, what?
    }
    nested();
    scope = "local scope";
    for (var i =0; i < 10; i++){
        var scope = "nested scope";
    }
    alert("2-" + scope);
}

checkscope();
alert("3-"+scope);
```

→**SHOW** `scope1.html` **and display**

iClicker Question

```
var name = "global";
{
   var name = "in scope";
   alert(name);
}
alert(name);
```

- What are the outputs?
  A. global, in scope
  B. in scope, in scope
  C. global, global
  D. in scope, global

Answer: B

var Location Does Not Matter

- You can use a variable *before* it is declared. It will be undefined. BUT you must declare this variable!
- Otherwise, see the following slides…
- But, best practice is to declare all at the start of the function.

```
alert("0-"+scope);
var scope = "global";
function checkloc() {
        alert ("1-" + scope);
        var scope = "functional";
        alert ("2-" + scope);
}
checkloc();
```

→**SHOW** `scope2.html` **and display**

What if you do not declare this global variable?

You get an error!

```
alert("0-"+scope);
// or remove the following line
scope = "global"; //no var used,
function checkloc() {
        alert ("1-" + scope);
        var scope = "local";
        alert ("2-" + scope);
}
checkloc();
```

→**SHOW** `scope2_2.html` **and display**

What if you do not declare this functional variable?

You access the global scope!

```
alert("0-"+scope);
var scope = "global";
function checkloc() {
     alert ("1-" + scope);
     // try to add the following line, change
     //var scope;
}
checkloc();
```

→**SHOW** `scope2_3.html` **and display**

What if you use this functional variable without declare it?

It becomes in global scope!

You can access it at the global scope!

BUT you must call the function first, AND no declaration in the function

```
checkloc(); //you must call the function first
alert("0-"+scope);
function checkloc() {
        scope = "functional";
        alert ("1-" + scope);
}
```

→**SHOW** `scope2_4.html` **and display**

How to access global variable from function

Use this one! No ambiguity

```
var scope = "global";
alert("0-"+scope);
function checkloc() {
        alert ("1-" + window.scope);
        var scope = "local";
        alert ("2-" + scope);
}
checkloc();
```

→**Modify** `scope2_5.html` **and display**

The End