07

24th Week • 158 -207

JUNE

09.00

10.00

11.00

12.00

01.00

JS : Scopes

02.00

03.00

04.00

05.00

06.00

07.00

@ ☎

JULY 2017

WK	M	T	W	T	F	S	S
27	31					1	2
28	3	4	5	6	7	8	9
29	10	11	12	13	14	15	16
30	17	18	19	20	21	22	23
31	24	25	26	27	28	29	30

THURSDAY 2017

24th Week • 159-206

JUNE 08

Scope represents visibility of variables, functions.

ie If I initialize a variable, functiin in what part it is visible, so that anyone can use it.

→ deciphering this particular fact, comes under scope.

Everything inside our code, will be used in one of the following 2 ways:-

↓

variables, functions.

(i) Either it will be getting some value assigned to it.

(ii) Either some value will be retrieved from it.

Compiled language:- In such languages, there is a process called as Compiler.

Compiler:- Compiler reads your whole code altogether.

If we have 3 bugs in our code, compiler will tell us know that there are 3 mistakes, we need to rectify.
Once, we rectify them then re-compilatiin happens again & then execution happens. (C, C++)

Interpreted language:- In such languages, there is something involved known as Interpreter.

Interpreter:- Interpreter does not reads your whole code at once.

It starts reading line by line. If there are 3 issues, then at the very first issue itself, interpreter will stop the code execution then and there. It will not move forward.

(Shell scripting)

✉ @ ☎

JS is not purely compiled nor interpreted language.

In JS code execution happens in 2 steps:—

① Parsing ② Execution.

What happens in:—

Parsing :— Scope Resolution.

JS reads the whole code in the first go, and then for every variable and functions that it sees. It tries to allocate a particular scope.

Types of Scope:—

① Global ② Function ③ Block

Global Scope:— If we define a variable inside global Scope, then that variable will be accessible everywhere. /func.

Function Scope:— If we define a variable /func. inside function Scope, then that variable /func. will be accessible in the whole function.

Block Scope:— If we define a variable /func. inside Block Scope. Then that variable /func. will be accessible in the whole Block. { }

Only, let and const declaration allowed. Not Var.

JULY 2017

WK	M	T	W	T	F	S	S
27	31					1	2
28	3	4	5	6	7	8	9
29	10	11	12	13	14	15	16
30	17	18	19	20	21	22	23
31	24	25	26	27	28	29	30

SATURDAY 2017

24th Week • 161-204

JUNE 10

How to initialize something in different scopes :-

If we define something inside the function scope; then our agenda is that we dont want to use it outside the function or maybe within another function.

```
function func() {
    const x = "declared inside";
    console.log("Inside");
    console.log(x);
}
```

Function Scope

console.log(x) → this will cause error.

The variable was intended to remain in the scope of the function func().

Global Scope

```
const x = "global val";

function fun() {
    console.log("Inside");
    console.log(x);
}

console.log(x);
```

Block Scope

```
{
    const x = 12;
}

console.log(x) → Error
```

SUNDAY 11

Because, this variable is intended to remain in the scope of the block.

Var :- When, we use (var); whatever variable we are declaring, that variable will be having either function Scope or Global Scope

Example:

```
var x = 1;                          → This (x) is acting as a Global Variable

if (x===1) {
    var x = 2;                      → This is a Block Scope.
                                      var x = 2; This (x) is actually the Global Variable (x) which is being referred.
    console.log(x); → (2)
}

console.log(x) → (2)
```

Note: (Var) also supports Hoisting.

let :- Whenever we are using (let); whatever variable we are declaring that variable will be having Block Scope.

Example:

```
let x = 1;

if (x===1) {
    let x = 2;                      → This is a Block.
                                      let x=2; means, a new (x) variable has been declared whose scope is within this IF block.
    console.log(x) → (2)
}

console.log(x); → (1)
```

```
{
    var x = 10;
}
console.log(x);
```

(Var) doesn't support Block scoping. So, even if we declare (var) inside a Block. The value of (x) can be accessible everywhere.
ie (x) will act as global Scope.

```
{
    let x = 10;
}
console.log(x)
```

This will throw an error because, the (x) creates a block scope because of let. which is only accessible within that block.

```
function fun(){
    var y = 20;
}
console.log(y)
```

This will throw an error. Because (var) is declared inside a function. Therefore, the scope it takes is function scope. And, since it takes function scope (y) will only be accessible within that function.

```
function fun(){
    console.log(x);
    var x = 10;
}
```

→ undefined.

When, we declare (var) inside a function then (x) will get function scope. Meaning the (x) can be accessible anywhere inside the function, even before declaration.

```
function fun(){
    console.log(x);
    let x = 12;
}
```

This will throw an error. Because (x) will have a Block scope (because of let). If (x) is declared inside a function. which means (x) can be accessible after the point of declaration. ∴ we are accessing before this, we are getting an error.

2017 WEDNESDAY

14 25th Week • 165-200 JUNE

JUNE 2017

WK	M	T	W	T	F	S	S
23				1	2	3	4
24	5	6	7	8	9	10	11
25	12	13	14	15	16	17	18
26	19	20	21	22	23	24	25
27	26	27	28	29	30		

Whenever, we declare something with "let", it is only accessible below it. ✓

What goes on behind the scenes in Parsing:- || and Execution

In the Parsing phase js will read the code one line at a time. and start allocating the variables their corresponding scope.

For every variable, js encounters it will ask the Scope Manager, does the var. already declared in a particular scope.

```
var teacher = "Sanket";
```
→ global scope (formal declaration)

```
function fun() {
    var teacher = "Pulkit";
    console.log(teacher);
}
```
this is a formal declaration, we think about the scope. in Parsing phase
→ fun() → global Scope
& since, we declared a function. So it creates its own scope.
Let's say "scopefun"
→ scopefun

```
function gun() {
    var student = "Sarthak";
    console.log(student);
}
```
Formal declaration global Scope.
& the func. gun() creates its own scope.
"ScopeGun"
→ "Scope Gun"

```
fun();
gun();
console.log(teacher);
```

Formal declaration means something which is defined using var, let, const, function.

JULY 2017

WK	M	T	W	T	F	S	S
27	31					1	2
28	3	4	5	6	7	8	9
29	10	11	12	13	14	15	16
30	17	18	19	20	21	22	23
31	24	25	26	27	28	29	30

THURSDAY 2017

25th Week • 166 - 199

JUNE 15

Outside, the function Global Scope.

Inside the function, we have scope local to the function

↓

i.e function has its own new Scope. called function Scope.

Now, execution phase will start.

During run-time we allocate value.

① var teacher = "Sanket"

∴ The bucket teacher which is in the Global Scope gets value "Sanket".

② function fun() { ⟶ Until and unless we call this function, we are not going to execute this. So, we skip this for now

}

③ function gun() { ⟶ Until and unless we call this function, we are not going to execute this. So, we skip this for now.

}

④ fun(); ↔ Is there a function in Global Scope called "fun" ✓

Now, we call it.

Inside Scope fun do we have teacher ✓

∴ teacher gets value "Pulkit".

console.log(teacher) ⟶ Inside Scope fun, do I have variable teacher Please print its value.

⑤ gun(); ⟶ Is there a function in global Scope called "gun" ✓

We call it.

Inside Scope Gun do we have student ✓

2017 FRIDAY

16 25th Week • 167-198
JUNE

JUNE 2017

WK	M	T	W	T	F	S	S
23				1	2	3	4
24	5	6	7	8	9	10	11
25	12	13	14	15	16	17	18
26	19	20	21	22	23	24	25
27	26	27	28	29	30		

∴ Variable Student gets value "Sarthak".

09.00

console.log (Student)

10.00

→ Inside Scope Gun do we have variable Student ✓
Please print its value.
Therefore "Sarthak" gets printed.

11.00

& we come out of the function.

12.00

01.00

⑤ console.log (teacher) → Do we have a ~~global~~ variable named student inside global scope ✓

02.00

Please print its value
∴ "Sanket" gets printed.

03.00

04.00

var teacher = "sanket"; → Formal declaration, global Scope

06.00

function fun() { → Formal declaration global Scope.
∴ Function has been declared. A function scope of "Fun Scope" initialized.

07.00

var teacher = "Pulkit";
content = "JS";
console.log (teacher)

Fun Scope
NO other formal declarations.

}

function gun() { → Formal declaration Global Scope.
∴ Function has been declared. A function scope of "Gun Scope" initialized.

var student = "Sarthak";
✉ @ ☎ console.log (Student);

Gun Scope
No other formal declarations.

}

fun();
gun();
console.log (teacher);
console.log (content);

JULY 2017

WK	M	T	W	T	F	S	S
27	31					1	2
28	3	4	5	6	7	8	9
29	10	11	12	13	14	15	16
30	17	18	19	20	21	22	23
31	24	25	26	27	28	29	30

SATURDAY 2017

25th Week • 168 - 197

JUNE 17

Execution Phase:—

09.00

(1) var teacher = "Sanket" → Do we have a variable (teacher) inside Global Scope ✓
10.00 ∴ teacher gets value "Sanket" assigned to it.

11.00

(2) function fun() { → This is a function declaration. But we are not calling the
12.00 function over here. Hence, this func. code set will be skipped.
01.00 }
02.00

(3) function gun() { → This is a function declaration. But we are not calling
03.00 the function. Hence, this func. code set will be skipped.
04.00 }
05.00

(4) fun() → Do we have a function (fun) inside Global Scope ✓
06.00 ∴ call it.
var teacher = "Pulkit"
07.00 Inside scope Funscope : Do we have a variable teacher. ✓

Please assign value "Pulkit" to variable teacher. SUNDAY 18

Inside scope Funscope. Do we have a variable content X

∴ We go one scope out from current scope.
∴ Now, we reach Global Scope.

Inside Global Scope, Do we have a variable content X

∴ We were trying to assign a value to a variable and we did not find the variable in any scopes. It will automatically become Global Variable.

2017 MONDAY

19 26th Week • 170-195

JUNE

JUNE 2017

Wk	M	T	W	T	F	S	S
23				1	2	3	4
24	5	6	7	8	9	10	11
25	12	13	14	15	16	17	18
26	19	20	21	22	23	24	25
27	26	27	28	29	30		

∴ The variable content during Execution Phase will become Global variable.
and it will get assigned the value JS.

console.log (teacher) → Do we have a variable teacher inside FunScope ✓
∴ Please print its value
∴ Pulkit gets printed.

09.00
10.00
11.00
12.00
01.00

⑤ gun() → Do we have a function (gun) inside Global Scope? ✓
∴ Call it.
var student = "Sarthak";
Do we have a variable student inside GunScope? ✓
∴ please assign value "Sarthak" to student.

02.00
03.00
04.00

console.log (student);
Do we have a variable student inside GunScope ✓
please print its value.
∴ "Sarthak" gets printed.

05.00
06.00

⑥ console.log (teacher) → Do we have a variable called teacher inside global Scope ✓
∴ please print its value.
∴ "Sanket" gets printed.

07.00

⑦ console.log (content) → Do we have a variable content inside Global Scope ✓
{We made it during execution phase}
∴ please print its value.
∴ (JS) gets printed.

TUESDAY 2017

26th Week • 171-194

JUNE 20

```
var teacher = "Sanket";
function fun() {
    var teacher = "Pulkit";
    content = "JS";
    console.log(teacher);
}
function gun() {
    var student = "Sarthak";
    console.log(student);
}
console.log(content);
fun();
gun();
console.log(teacher);
```

→ Formal declaration, global scope.

Formal declaration, global scope.
Also, since we're initializing function. So function scope, "Funscope" gets initialized.

Funscope

No other formal declarations.

→ Formal declaration, global scope
Also, since we're initializing fun. So, fun. scope, "Gunscope" gets initialized.

Gunscope

No other formal declarations.

No other formal declarations.

Execution phase:-

① var teacher = "Sanket" → Do we have variable teacher in Global Scope ✓
∴ please assign value "Sanket" to it.
∴ teacher = "Sanket".

② function fun() { → ∵ We've declared a function but have not called it.
So, we will skip this func. codeset.
.
.
}

2017 WEDNESDAY

21 26th Week • 172-193 JUNE

③ function gun() { ⟶ ∵ We're declaring the function and not calling it
∴ We will skip this function codeset.
}

④ console.log(content) ⟶ Do we have a variable (content) declared in Global Scope ✗

∵ We don't have any such variable and also we are trying to print its value.
∴ We will get error.

∴ The concept of Autoglobal basically states that,
If we are declaring a variable without var, let, const then that variable
① will become global scope if we are assigning value to that variable.
② will throw ref error, if we are trying to get its value.

To prevent auto-globals from happening. we can use Strict Mode.

JULY 2017

WK	M	T	W	T	F	S	S
27	31					1	2
28	3	4	5	6	7	8	9
29	10	11	12	13	14	15	16
30	17	18	19	20	21	22	23
31	24	25	26	27	28	29	30

THURSDAY 2017

26th Week • 173 - 192

JUNE 22

①
```
console.log ('hi');
console.log ("hello");
console.lo ("bye");
```
op: hi, hello, error

```
console.log ('hi');
console.log ("hello");
console.log ("bye");
```
op: error

Explain why is that?

Part One :- console is an object, inside which there exists a function log.

When, parsing is done console is already declared inside the Global Scope.

When execution phase comes up,

Inside global scope do we have (console)? ✓

Do we have (log) inside (console)? ✓

∴ It executes (log) function.

∴ hi, hello gets executed.

For, (bye) we don't have a function (lo) inside (console).

∴ It throws an error.

Part Two :- The parsing only, it will throw an error because there exists a Syntactical issue.

23 26th Week • 174-191 JUNE

```
function fun() {
    var x = 10;
    function gun() {
        var y = 20;
        console.log(x);
        console.log(y);
    }
    console.log(x);
    console.log(y);
}
fun();
```

→ Formal declaration global Scope of "FunScope" gets initialized. ✓

→ FunScope (formal declaration)

→ Formal declaration FunScope of "GunScope" gets initialized. ✓

→ GunScope ✓

NO other formal declaration inside function gun.

NO other formal declaration inside function fun.

Execution phase:-

① function fun() { → This is a function declaration. ∴ we are not calling it here.
Hence, we will skip this function codeset as of now.
}

② fun() → Do we have a function fun inside global Scope ✓
∴ Call it.

① var x = 10;

Do we have variable called (x) inside Funscope ✓

∴ Please assign the value (10) to x.

∴ x = 10 ✓

(B) function gun() { → this is a function declaration. ∵ we are not calling the func. here.

∴ For the time being, we can skip this function codeset.

}

(C) console.log(x) → Do we have a variable (x) inside Funscope ✓

∴ Please print its value.

∴ 10 gets printed. ✓

(D) console.log(y) → Do we have a variable (y) inside Funscope ✗

∴ look outside one scope outside. ∴

Do we have a variable (y) inside global scope ✗

∵ We don't have variable (y) inside global scope and we are trying to print its value

∴ We will get <u>error</u>.

26 27th Week • 177-188
JUNE

```
function fun() {
    var x = 10;
    function gun() {
        var y = 20;
        console.log(x);
        console.log(y);
    }
    gun();
    console.log(x);
    console.log(Y);
}
fun();
```

→ Formal declaration, global scope
∵ Func. has been initialized, ∴ FunScope gets initialized.

→ FunScope (formal declaration)

→ Formal declaration, FunScope
∵ Func. has been initialized. ∴ GunScope gets initialized.

→ GunScope (formal declaration)

No other formal Declarations inside func. (gun).

No other formal declarations inside func. (fun).

No other formal declarations inside Global Scope.

Execution Phase:-

① function fun() { ⟶ This is a function initialization. and we are not calling it.
∴ We will skip this function codeset.
}

② fun() ⟶ Do we have a function (fun) inside Global Scope ✓
∴ call it ✓

(A) var x = 10.

Do we have a variable (x) inside FunScope? ✓
∴ Please assign value (10) to the variable.

(b) function gun() { → This is a function initialization.
∵ We are not calling the function here.
∴ We will skip it.
}

(c) gun() → Do we have a function (gun) inside FunScope? ✓
∴ Call it.

(i) var y = 20;
Do we have a variable declared inside GunScope? ✓
∴ please assign value 20 to the variable y.

(ii) console.log(x);
Do we have a variable (x) declared inside GunScope? ✗
↓
∴ Go one scope outside.
∴ We are now inside FunScope.
Do we have a variable (x) declared inside FunScope? ✓
∴ please print the value of x
∴ 10 gets printed.

(iii) console.log(Y);
Do we have a variable (Y) declared inside Gunscope? ✓
∴ please print the value.
∴ 20 gets printed.

28 27th Week • 179-186
JUNE

(D) console.log(x) → Do we have a variable (x) inside Funscope? ✓
∴ please print the value.
∴ 10 gets printed.

(E) console.log(Y) → Do we have a variable (Y) declared inside Funscope
X
↓
Go one scope outside
∴ Now, we are at Global Scope.
Do we have a variable (Y) declared inside Global Scope
X

Now, since we do not have the variable (Y) inside Global Scope and at the same time we are trying to print its value.
∴ It will throw us an error.

Lexical Scoping:- During Parsing phase, we determine the scope.
i.e. Scope Resolution is done before Execution Step.

Dynamic Scoping:- During Runtime phase, we determine the Scope.
Eg Bash Scripting.

var teacher = "Sanket"; → Formal declaration (Global Scope) ✓

function ask(question){
✉ @ ☎
console.log(teacher, question);
}

Formal declaration (Global Scope) ✓
∴ Function has been initialized.
∴ Ask Scope is initialized.

No Formal declaration inside Ask Scope.

```
function fun() {
    var teacher = "Pulkit";
    ask("Why?");
}
fun();
```

→ Formal declaration (Global Scope) ✓
∵ Function has been initialised.
∴ FunScope has been initialised.

FunScope ✓
No formal declarations inside func. (fun).

No formal declarations inside global scope.

Execution Phase:-

(1) var teacher = "Sanket" → Do we have a var (teacher) inside global Scope ✓
∴ please assign value "Sanket" to teacher.

(2) function ask(question){ → This is a func. declaration. ∵ It is not being called here.
∴ We can skip this codeset for now.
}

(3) function fun(){ → This is a func. declaration. ∵ It is not being called here.
∴ We can skip this codeset for now.
}

(4) fun(); → Do we have a function (fun) inside Global Scope ✓
∴ call it.

(A) var teacher = "Pulkit";
Do we have a var (teacher) inside the scope FunScope ✓
∴ please assign value "Pulkit" to teacher.

(B) ask("Why?")

30 27th Week • 181-184
JUNE

Do we have a func. (ask) inside the scope FunScope ✗

↓

Go one scope out
∴ Now, we are at Global Scope.

Do we have a func. (ask) inside the Global Scope? ✓
∴ Call it.

(1) console.log (teacher, question);

For teacher

Do we have a variable (teacher) inside the scope AskScope ? ✗

↓

∴ Go one scope out
∴ Now, we are at the Global Scope.

Do we have a variable (teacher) inside Global Scope? ✓

∴ It becomes [console.log ("Sanket", queslim)]

For queslim

Do we have a variable queslim in AskScope ? ✓

Because, function ask (question) {

}

∴ assign value "why" to variable question.

∴ console.log ("Sanket", "why");

∴ It will print "Sanket" "why"

AUGUST 2017

WK	M	T	W	T	F	S	S
32		1	2	3	4	5	6
33	7	8	9	10	11	12	13
34	14	15	16	17	18	19	20
35	21	22	23	24	25	26	27
36	28	29	30	31			

Temporal Dead Zone:-

The region before the declaration of a variable which is having a block scope. is called TDZ.

SATURDAY 2017

27th Week • 182-183

JULY 01

```
var teacher = "Sanket";                    → Formal declaration (Global Scope)
function fun() {                           → Formal declaration (Global Scope)
                                             Fun Scope get initialized.
    var teacher = "Pulkit";
    let content = "JS";                    → Funscope
                                             Due to the usage of let, content
                                             will be accessible post its declaration.
                                             & also available within func. (fun).
    if (content == "JS") {
        let hours = "120+";                → This also has a block scope and
                                             it cannot be accessed outside the IF
                                             & also accessible post its declaration.
        console.log(hours);
    }                                      No other formal declarations inside (fun).
    console.log(teacher, content);
}
fun();                                     No other formal declarations inside
                                           Global Scope.
console.log(teacher);
console.log(content);
```

SUNDAY 02

Execution Phase:-

① var teacher = "Sanket" → Do we have any variable (teacher) inside global Scope ✓
∴ please assign value "Sanket" to (teacher).

② function fun() { → This is a function declaration. ∴ We are not calling it here. So, we can skip this function content as of now.

}

09.00 ③ fun() ⟶ Do we have a function (fun) inside global scope? ✓
∴ call it.

10.00 (A) var teacher = "Pulkit";

11.00 Do we have a variable (teacher) inside scope FunScope ✓

12.00 ∴ Please assign value "Pulkit" to variable (teacher)

01.00 (B) let content = "JS";

02.00 Please assign value (JS) to variable (block) (content).

03.00 (C) if (content == JS) { ⟶ Is content accessible ✓
∴ Please do comparison → true
04.00 ∴ Inside IF block

05.00 (i) let hours = "120+"

06.00 Please assign "120+" to variable (block) (hours).

07.00 (ii) console.log (hours) ⟶ Is the variable hours accessible ✓
∴ please print 120+.

(D) console.log (teacher, content)

For teacher

Do we have a variable (teacher) inside the scope FunScope? ✓

∴ (teacher) gets the value "Pulkit"

@☎

For Content

Can we access the variable (block) content? ✓

∴ (content) gets the value "JS".

AUGUST 2017

WK	M	T	W	T	F	S	S
32		1	2	3	4	5	6
33	7	8	9	10	11	12	13
34	14	15	16	17	18	19	20
35	21	22	23	24	25	26	27
36	28	29	30	31			

TUESDAY 2017

28th Week • 185-180

JULY 04

∴ We print Pulkit. JS.

09.00

(E) console.log (teacher) → Do we have a variable (teacher) inside global scope? ✓

10.00

∴ please print its value.

11.00

∴ "Sanket" gets printed.

12.00

01.00 (F) console.log (content) → Do we have a variable (content) inside global scope? ×

02.00

∵ We do not have a variable (content) inside global scope and we are trying to access its value.

03.00

∴ We will get an (error) over here.

04.00

05.00

06.00 Another extension :-

07.00

```
function fun() {
    var teacher = "Pulkit";
    let content = "JS";
    if (content == "JS") {
        let hours = "120+";
        console.log(content, hours);
    }
}
```

This content can be accessible over here as well. Because the scope of the content lies post its declaration. And, we are calling it within a nested section in function (fun).

@ ☎

05 28th Week • 186-179
JULY

```
{
    function fun () {
        return "123";
    }
}
console.log (fun);
```

This is a normal block.
Inside, the block we create a function (fun).

Then this (fun) can be accessed outside of the block as well

∴ Here, the o/p will be "123".

i.e The function (fun) will get the outside scope and not the block scope.
If the outside scope is global. It will take that.
If the outside scope is function. It will take that.