- Arrow functions:

```
var getName = () => {
    console.log("Namaste JS")
}
```

⇒ Now, if we do getName() before this, it will console "undefined".

-
```
var getName = function() {
    console.log("Namaste JS")
}
```

⇒ It will also be stored as "undefined" during memory creation phase

Episode - 4

✱ Hoco functions work in Javascript and Variable Environment.

→ Code:

```
1  var x = 1
2  a()
3  b()

4  console.log(x)
```

Miracle
Pg.No :
Date: / /20

```
5  function a(){
6      var x = 10
7      console.log(x)
   }
8  function b(){
9      var x=100
10     console.log(x)
   }
```

→ Output:

10
100
1

→ How?

Memory	Code
x: undefined → 1	var x = 1
a: {...}	
b: {...}	

Line: 1 ↓ Global EC is created

L2: x: 1 is assigned and a new EC is created for a(). Control moves to line 6.

M	C
x: undefi. → 10	var x = 10
	console.log(x)

M	C
x: undefi. → 100	var x = 100
	console.log(x)

console.log(x)

Call Stack

~~b()~~
~~a()~~
~~GEC~~

x finds it memory allocation in ~~local EC~~ and assigns it so x = 10 is printed. After exec this EC is deleted → x = 100 and EC deleted

x = 1 is printed and GEC deleted

Miracle
Pg No :
Date: / /20

Episode - 5

Shortest JS Program

→ Shortest Javascript program is an Empty File.

→ How? Why?

- Even in an empty file, a global EC is created.
- Also creates a Window object. (global obj

Window object has lots of functions and methods. Created by the JS engine in the global space.

We can access all this methods and functions anywhere in our JS program.

- Also creates a this keyword.

At the global level, "this" points to the window object.

- So, 3 things are created:

(1) Global Execution Context
(2) Global Object (in case of browsers, it is known as window)
(3) this keyword

- At the global / base level, "this" keyword is equal to window object

this === window (in console, if we write)

⇒ true

- Global Space : Anything that is not inside a function is "global space"

e.g. var a = 10
function b() {
var x = 10
}

a is in global space. x is not.

- In console (Inspect ⇒ console)

window
⇒ Window {..... key : value pairs ....}

this
⇒ Window { key-value pairs .... }

this === window
⇒ true

- Whenever we create any variables or functions in the "global space", they get attached to the global object (window)

- In the previous code, to access variables / functions in the global space, we can do:

```
console.log(window.a)
console.log(a)
console.log(this.a)    (this points to window in global level)
```

Output: All 3 is same, 10

- If we do,

```
console.log(x)
```

⇒ Error: x is not defined. (x is not in global level)