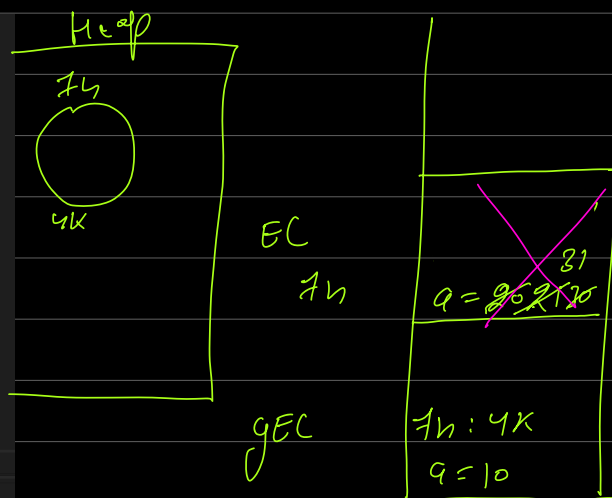


Agenda

- ① few question ground var
- ② shadowing
- ③ host and native object.
- ④ How 'this' behave in js.
- ⑤ chaining in js
- ⑥ inheritance
- ⑦ call, bind, apply.

```
var a = 10;  
console.log("line number 2", a);  
function fn() {  
  var a = 20;  
  console.log("line number 4", a);  
  a++;  
  console.log("line number 7", a);  
  if (a) {  
    var a = 30;  
    a++;  
    console.log("line number 11", a);  
  }  
  console.log("line number 13", a);  
}  
fn();  
console.log("line number 16", a);
```



console

2: 10
4: 20
7: 21
11: 31
13: 31
16: 10

```

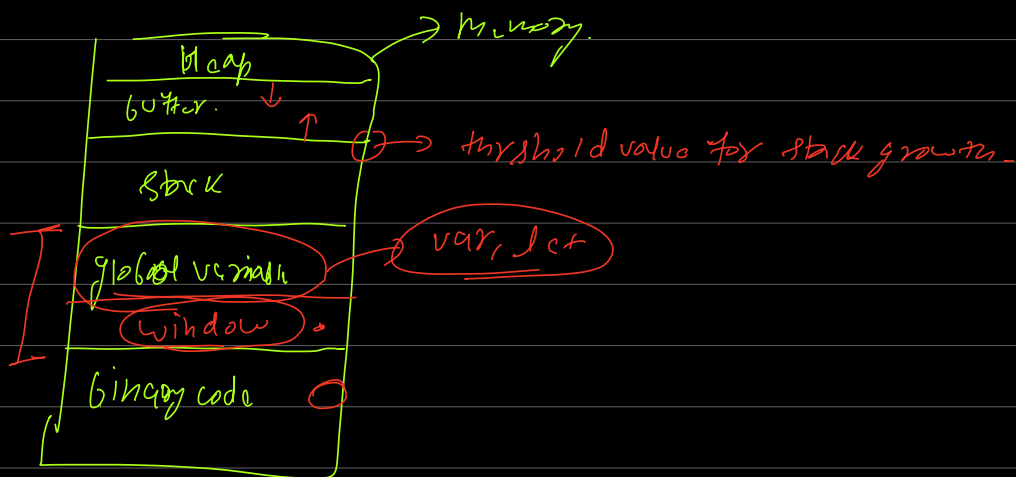
var f = 5
fn c() {
  f = 6;
}
fn o:
log(f)

```

EC

JEC

~~f = 5~~ 6



```

let fruits = "apple";
console.log("21", fruits); // apple
{
  let fruits;
  console.log("25", fruits);
  fruits = "orange";
  {
    let fruits;
    console.log("28", fruits)
  }
  console.log(fruits);
}
console.log(fruits);

```

EC

7: ~

7: ~~orange~~

EC

gEC

7: apple

console

21: apple

25: ~

28: ~

: orange

: apple

```

let cap = {
  // property
  firstName: "Steve",
  // method
  sayHi: function () {
    console.log("Hi from ", this.firstName);
  }
}

```

```

cap.sayHi();
let sayHiAdd = cap.sayHi; 100K

```

```

var firstName = "loki";
sayHiAdd();

```

Heap

200K
100K
cap

100K
sayHi

EC

w | this = 200K

cap: 200K

w | this = w

gEC

Heap

200K
cap
100K
sayHi
100K

EC

window { firstName = 'loki' }

gEC

w	this = w
sayHiAdd = 100K cap: 200K	
w	this = w

```

let cap2 = {
  firstName: "Steve",
  sayHi: function (param) {
    console.log("47", this.firstName);
    const iAmInner = function (param) {
      console.log("49", this.firstName);
    }
    // EC by this kind of call -> window
    iAmInner(20);
  }
}

// // EC by this -> ?? -> cap
cap2.sayHi(10);
  
```

EC

EC

gEC

w	this = w
w	this = 200K
w	this = w

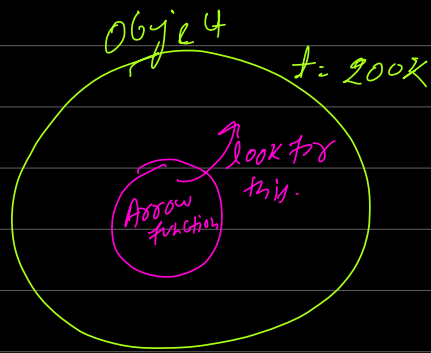
47 : Steve

49 : 20

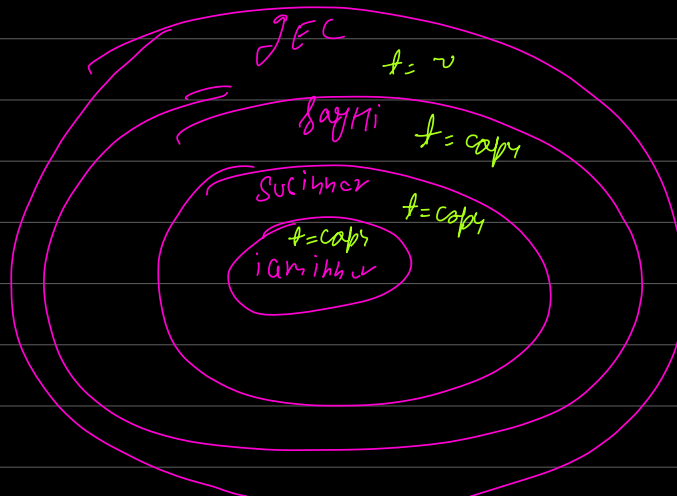
(*) object $\xrightarrow{\text{this = window}}$ simple-function call

object $\xrightarrow{\text{this = object}}$ method call

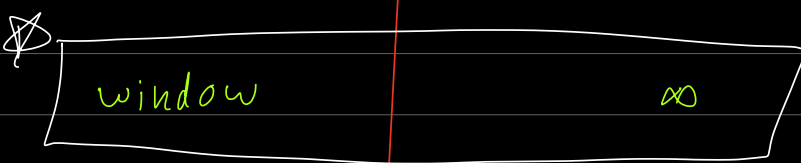
object $\xrightarrow{\text{this = (nearest scope)}}$ arrow call



```
let cap4 = {
  firstName: "Steve",
  sayHi: function () {
    console.log("91", this.firstName);
    // arrow -> does not have it's own this. I am going to cheat it from outside
    const subInner = () => {
      console.log("94", this.firstName);
      const iAmInner = () => {
        console.log("95", this.firstName);
      }
      iAmInner();
    }
    subInner();
  }
}
cap4.sayHi();
```



• eval()	call(),	() => {
↓ = object	↓ = w	↓ = nearest parent

	non-strict (this)	strict (this)
gEC	window	window
Function		∞
method	current object	current object
Arrow fn.	nearest scope 'this'	nearest scope 'this'

Chaining

