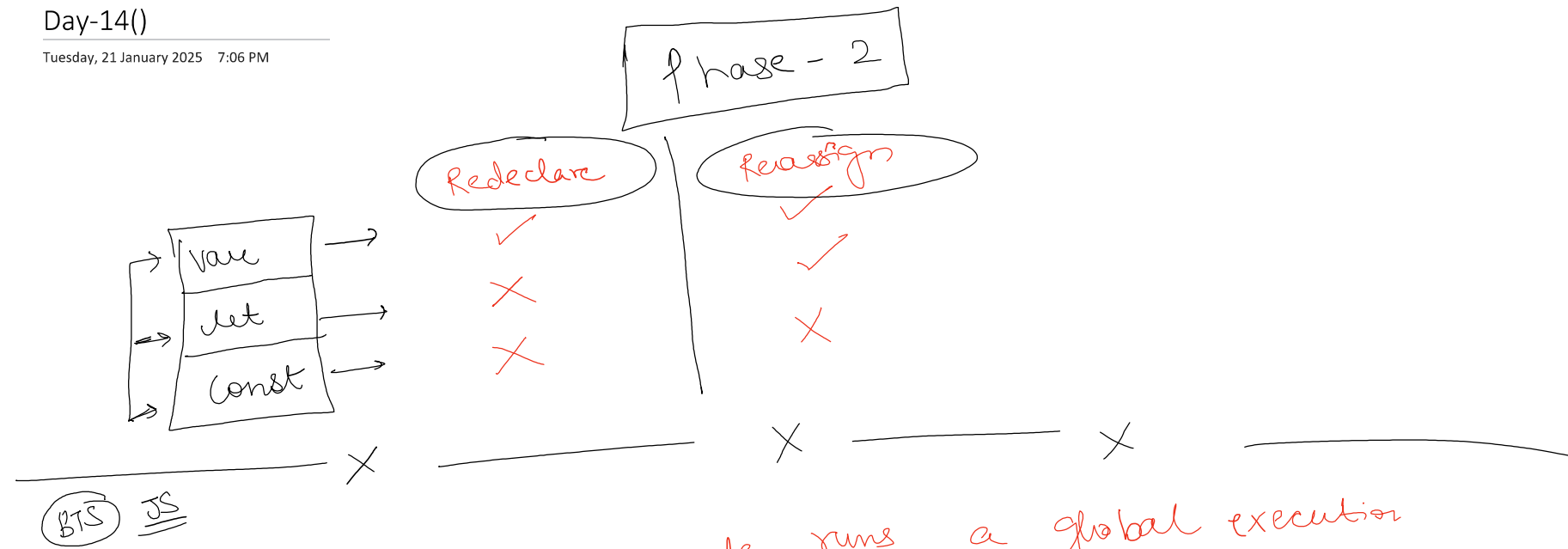


Day-14()

Tuesday, 21 January 2025 7:06 PM



① Whenever a JS code runs a global execution context (GEC) context is created.

② Inside a GEC we have 2 phases

- 2.1 Memory Creation phase (MCP) ①
- 2.2 Code Execution phase (CEP) ②

③ Role of MCP is to allocate memory

- 3.1 to all the variables *
- 3.2 to all the fns *

Memory

MCP runs before CEP *

④ CEP means running the code line by line (0 line)

after MCP is done. *

... is called a new execution (EC) process repeats.

EC

- MCP ✓
- CEP ✓

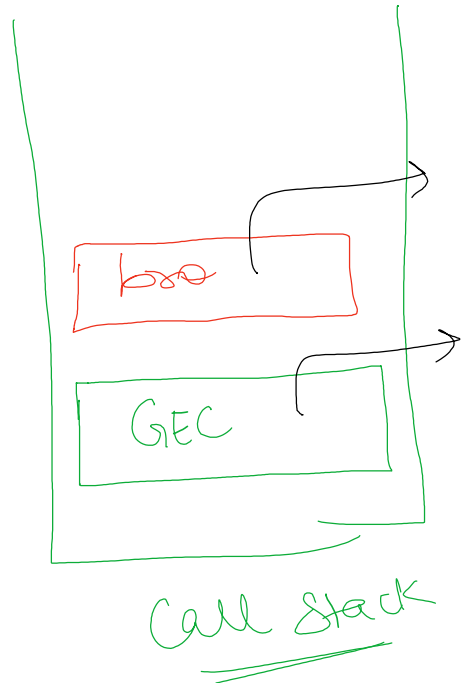
Whenever a ~~frame~~ context is created & the same ~~frame~~
Whenever GEC is created, a global object along with it is also created. (window) *

later on
(6)

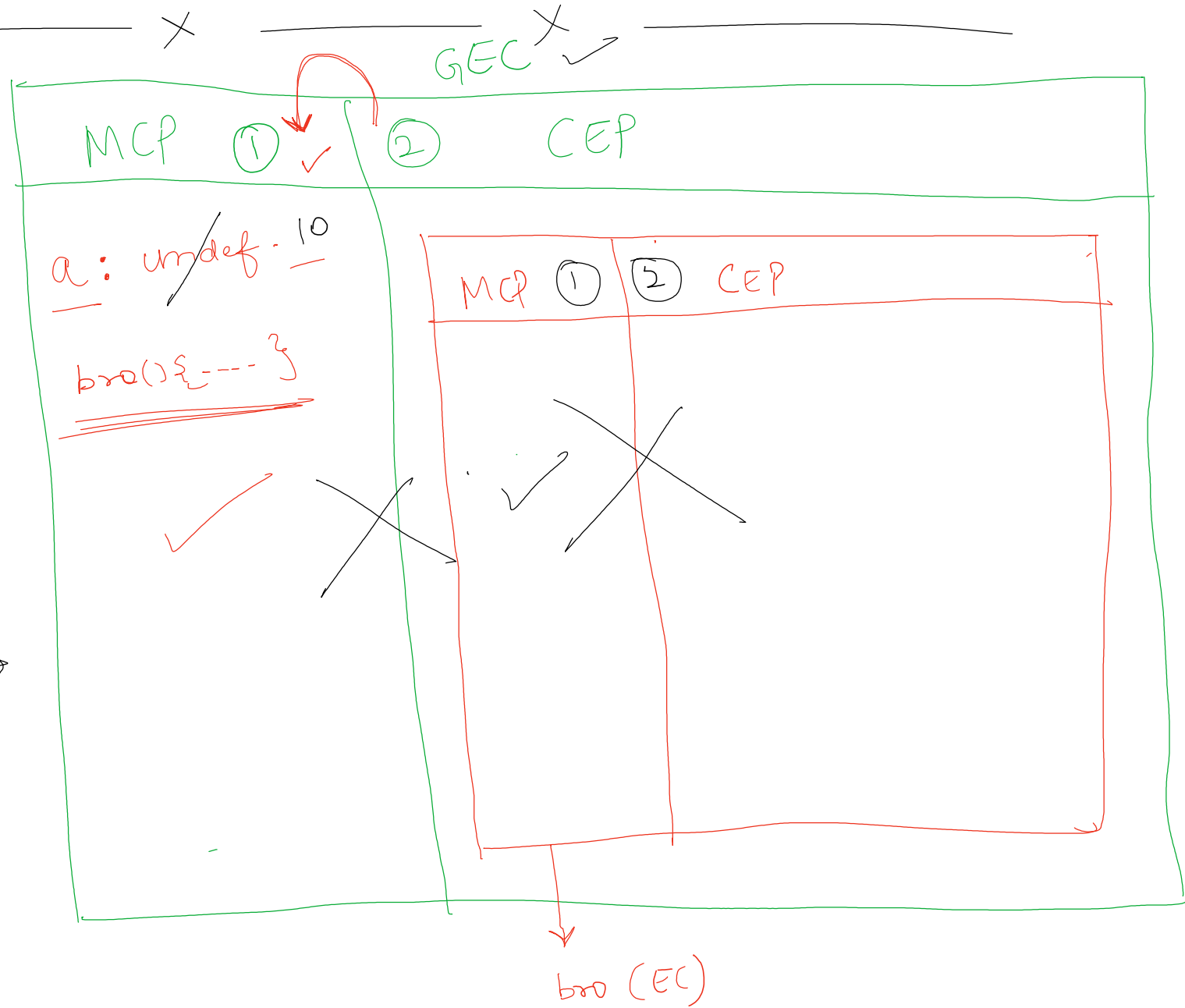
JS

```
1 var a = 10;  
2 console.log(a);  
  
3 function bro(){  
4   console.log("i am bro");  
5 }  
6 bro();
```

10
I am bro
output



empty

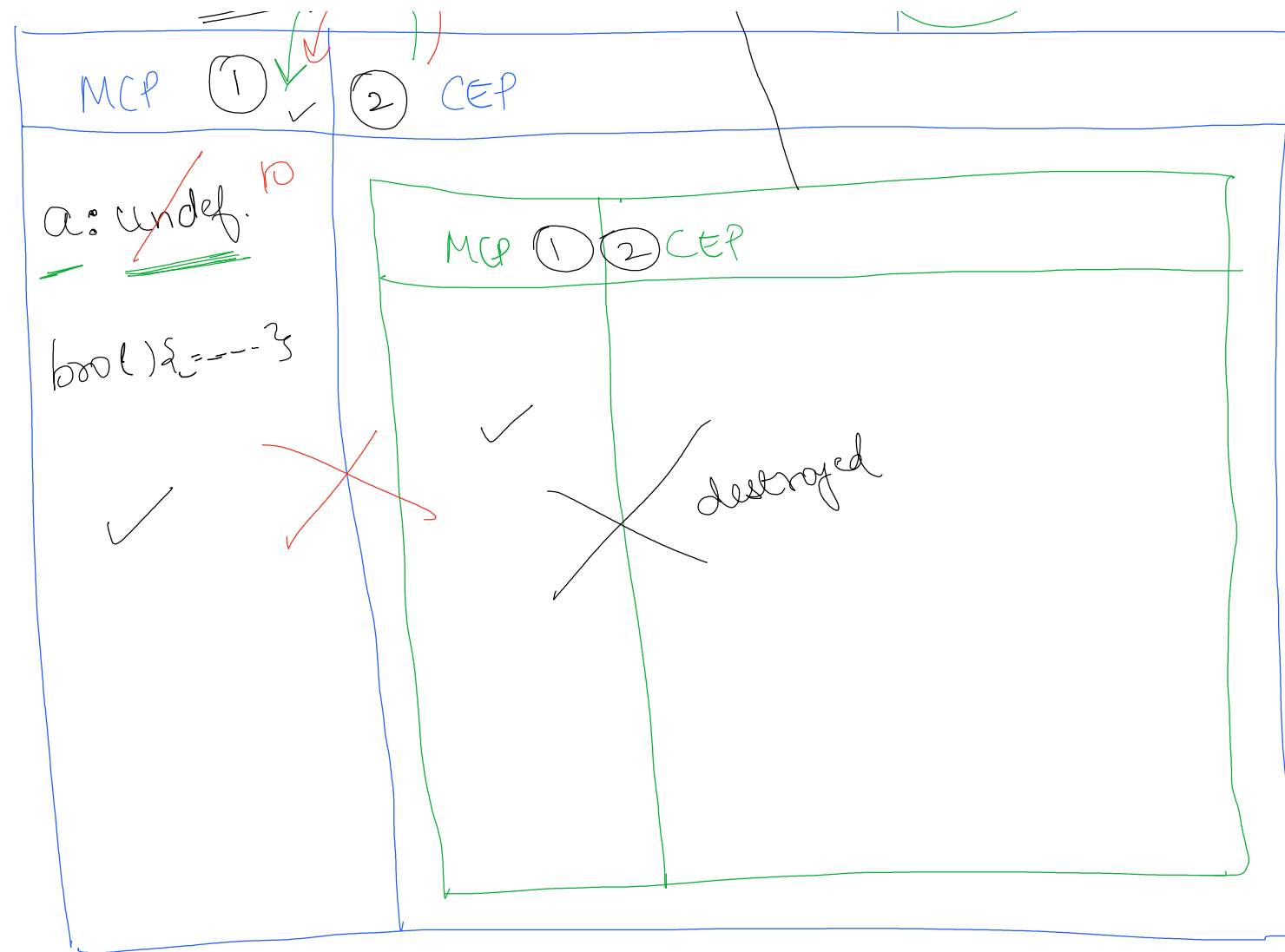


JS Engine

bro (EC)
↑

GEC

undef.

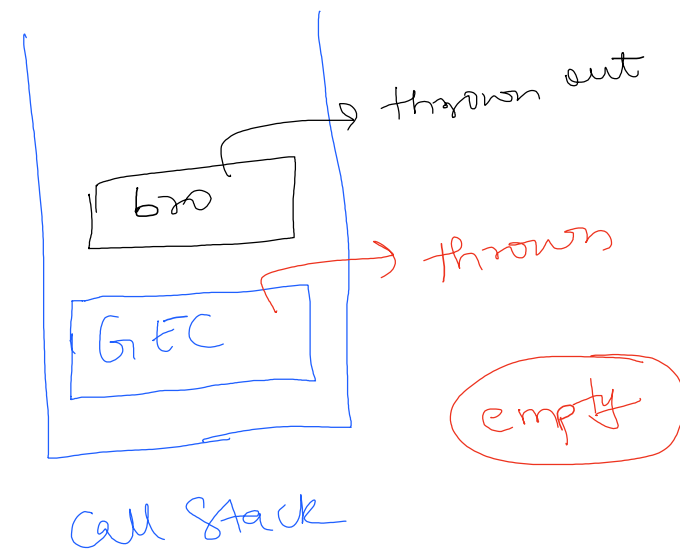


I am bro

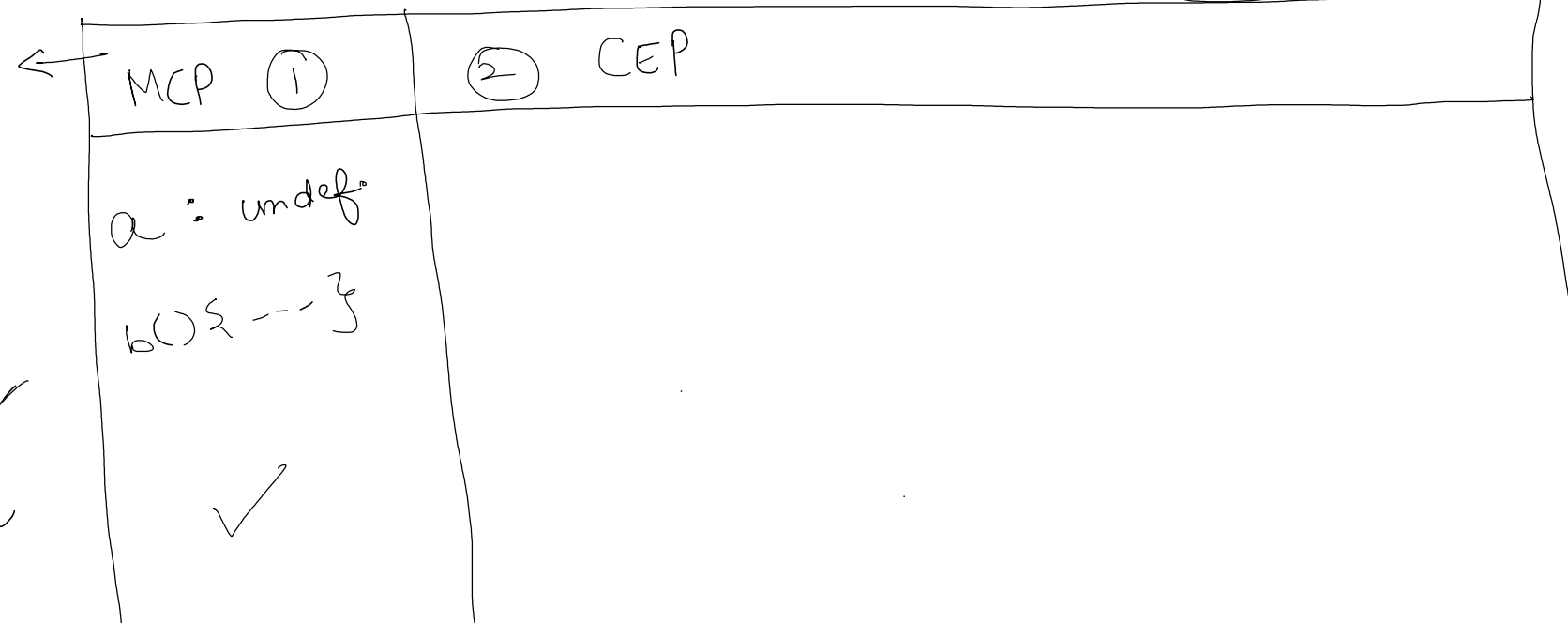
```

1 console.log(a);
2 bro();
3 var a = 10;
4 function bro(){
5   console.log("i am bro");
6 }
  
```

output



(0 line of code)



```

1 console.log(a);
2 var a = 10;
3 b()
4 function b(){
5   console.log(c);
6   var c = 200
7   console.log(c);
8 }
9
  
```

A hand-drawn diagram illustrating a memory layout. On the left, a red-outlined rectangle contains the text "undef." in red. Below this rectangle, the word "output" is written in red cursive. On the right, a blue-outlined rectangle contains a smaller white rectangle with a black border, which in turn contains the text "GEC." in blue. Below the blue rectangle, the letters "CS" are written in blue.



A diagram illustrating the flow of data from a code line to two components, MCP and CEP. A green box on the left contains the text "0 line of code". An arrow points from this box to a green circle labeled "1" inside the MCP component. From circle "1", two arrows point to a green circle labeled "2" inside the CEP component. The CEP component is labeled "GEC" above it. The MCP component is labeled "MCP" above it.

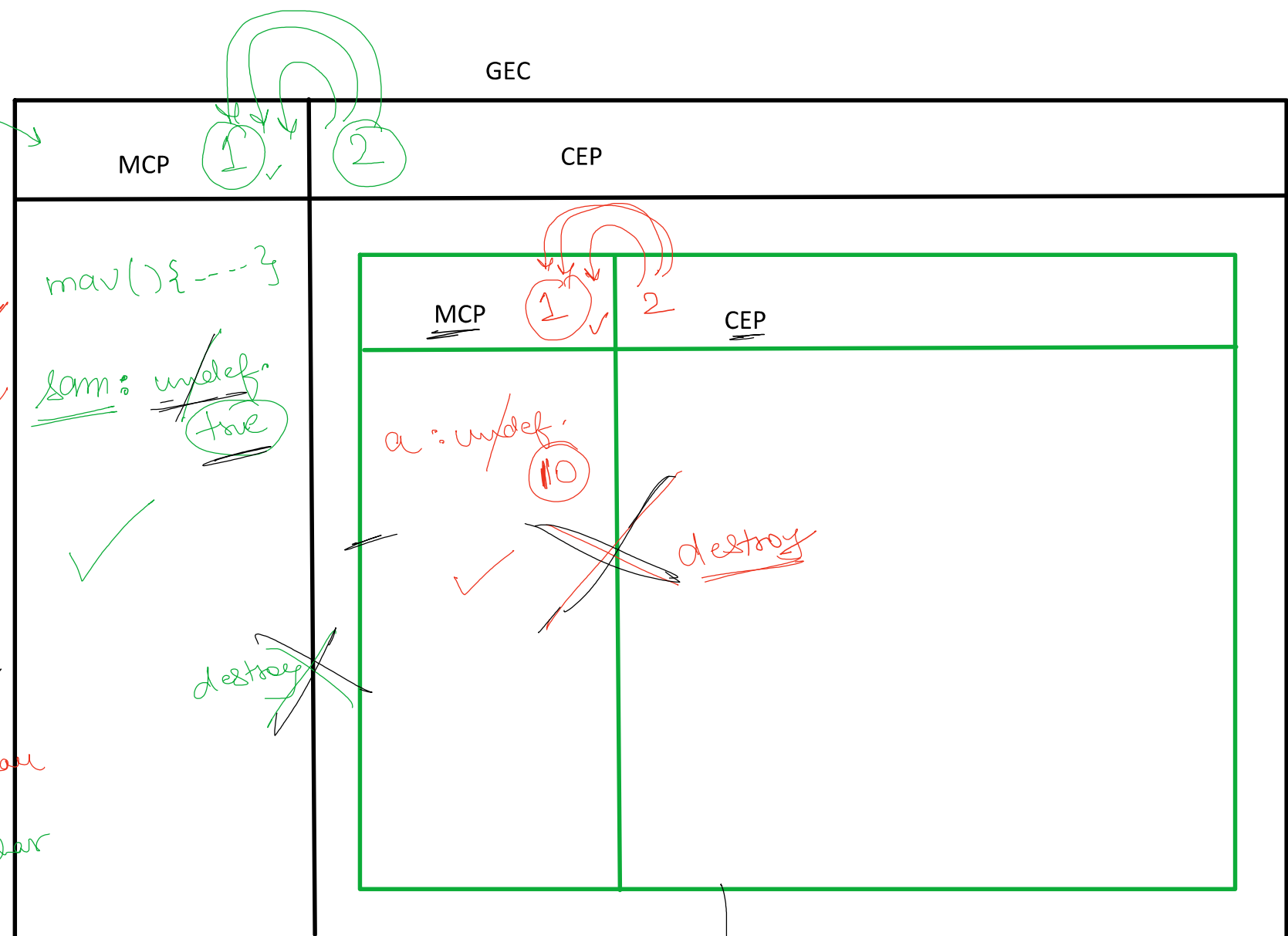
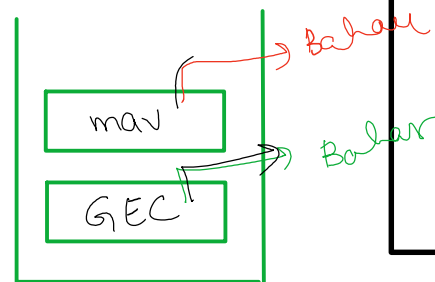
map() ✓

```
function mav(){  
  console.log(a);  
  var a = 10;  
  console.log(a);  
}
```

```
console.log(sam);
```

```
var sam = true;
```

```
console.log(sam);
```



BTS

CALL STACK
empty

undef. ✓
10 ✓
undef. ✓
true ✓

output

max (EC)

Hoisting Ⓢ

✓ console.log(a);
var a = 10;

Whenever you are trying to access a
variable or a fn even before its

declaration Ⓢ

Ⓢ

Ho

DTZ

Temporal
Dead
Zone

general
Hoisting

↓
let | const

7
var
↓
undef'

ur
↓
error (special)