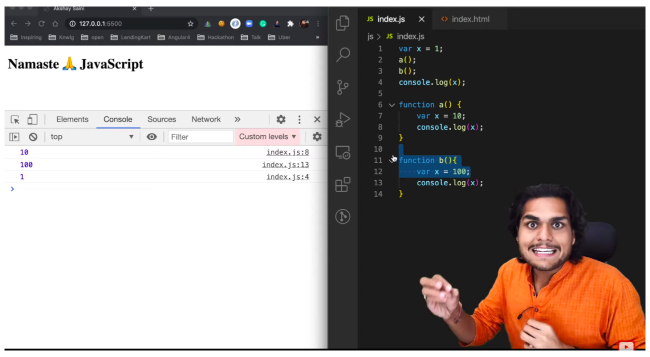
# How Functions Work in JS and Variable Environment

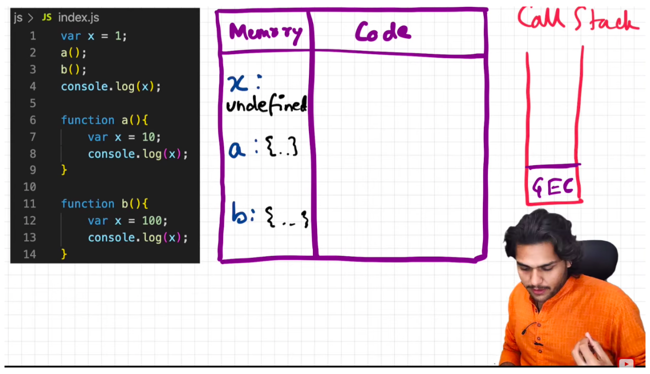
Wednesday, 16 August 2023 12:28 PM



```
10                                   index.js:8
100                                  index.js:13
1                                    index.js:4
>
```

```
1   var x = 1;
2   a();
3   b();
4   console.log(x);
5
6   function a() {
7       var x = 10;
8       console.log(x);
9   }
10
11  function b(){
12      var x = 100;
13      console.log(x);
14  }
```

Now, we will see as to why the above code produces the above output.



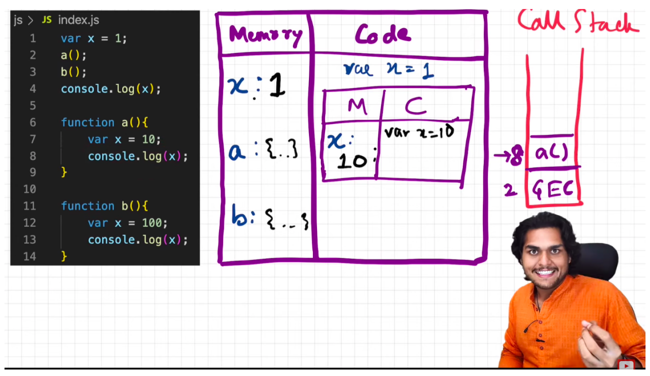
```
1   var x = 1;
2   a();
3   b();
4   console.log(x);
5
6   function a(){
7       var x = 10;
8       console.log(x);
9   }
10
11  function b(){
12      var x = 100;
13      console.log(x);
14  }
```

→ In the first phase i.e the memory execution phase, 'x' is given undefined value and $f^n$ a() & b() are stored as they are in the global EC.

→ After the first phase, we reach the code execution phase, where value of x is replaced from undefined to 1.

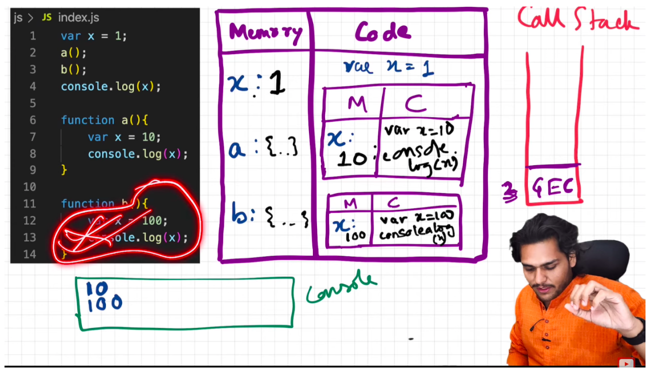
→ When we reach line 2, another EC is pushed to the call stack and there, memory creation phase takes place and another variable is allotted memory which is x and is given undefined value.

Now, code execution phase takes place, and value of x is replaced from undefined to 10. After encountering console.log JS searches for the value of x inside the local EC & 10 is printed. After the $f^n$ ends, the local EC is deleted & popped out of the call stack.



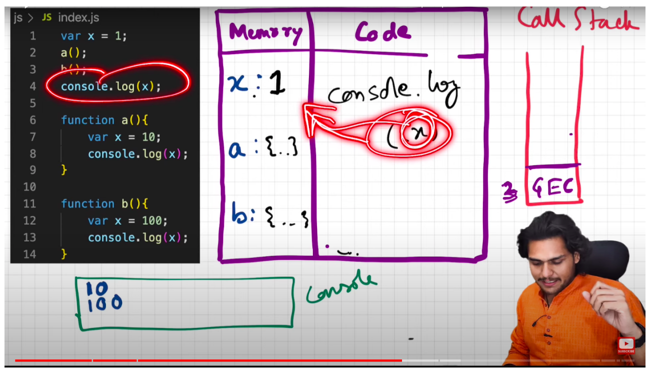
```
1   var x = 1;
2   a();
3   b();
4   console.log(x);
5
6   function a(){
7       var x = 10;
8       console.log(x);
9   }
10
11  function b(){
12      var x = 100;
13      console.log(x);
14  }
```

→ Similarly, same process is repeated when we reach line 3 and 100 is printed.



```
js > JS index.js
1   var x = 1;
2   a();
3   b();
4   console.log(x);
5
6   function a(){
7       var x = 10;
8       console.log(x);
9   }
10
11  function b(){
12      var x = 100;
13      console.log(x);
14  }
```

→ Now, finally when we reach line 4, all local ECs have been deleted, so JS searches for the value of x inside the global EC and 1 is printed.



```
js > JS index.js
1   var x = 1;
2   a();
3   b();
4   console.log(x);
5
6   function a(){
7       var x = 10;
8       console.log(x);
9   }
10
11  function b(){
12      var x = 100;
13      console.log(x);
14  }
```

→ Finally, when the program ends, the Global EC is also deleted and popped off the call stack.