# Immediately Invoked Function Expression

or IIFE

An IIFE ( Immediately Invoked Function Expression ) is a JavaScript function that runs as soon as it is defined.

```
( function () {
    statements
} ) ( );
```

It can also reffered as Self-Executing Anonymous Function.

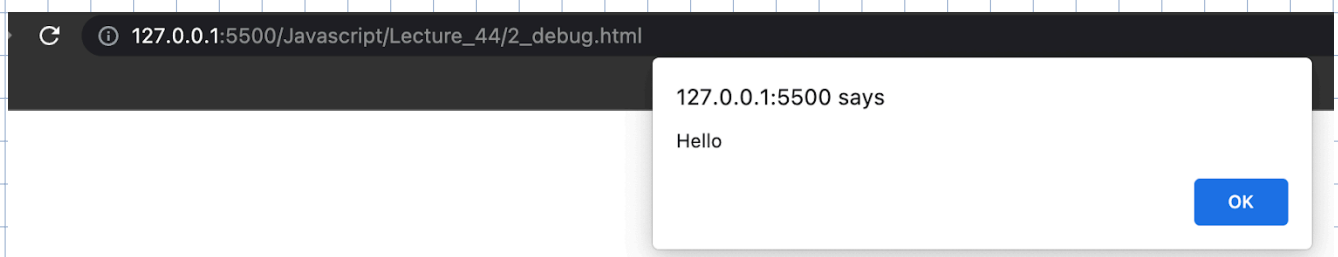
**It contains two major parts:**

1. **The first is the anonymous function with lexical scope enclosed within the Grouping Operator (). This prevents accessing variables within the IIFE idiom as well as polluting the global scope.**
2. **The second part creates the immediately invoked function expression () through which the JavaScript engine will directly interpret the function.**

```
(function(){

    console.log("Hi, I am an IIFE");
    alert("Hello");
})();
// IIFE = immediately invoked function execution
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>For Debug purposes</title>
</head>
<body>
    <script src="1_IIFE.js"></script>
</body>
</html>
```



Ques 1. Contruct a timer using IIFE, prompt, alert

1. Page Load → Ask the user of time?
2. Use setInterval and console.log 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
3. Alert → counted 10 sec

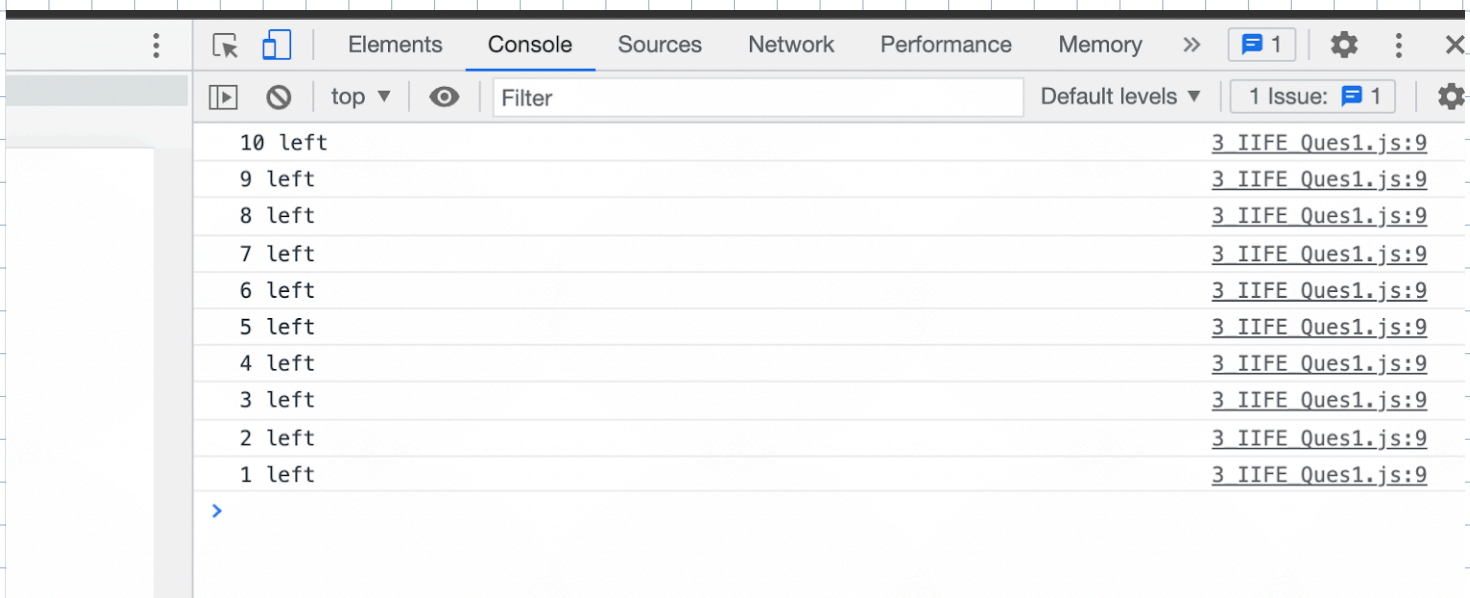
```
(function(){
    let timeUnits = parseInt(prompt("How much to count?"));
    let interval = parseInt(prompt("Log after how much interval"));

    // calls the handleCall function after every interval seconds (passed as millis)
    let iid = setInterval(handleCalls, interval * 1000);
    //returns an id used to stop calling via clearInterval

    handleCalls.orgTU = timeUnits; //Functions can be used as a store of properties (much like object)

    function handleCalls(){
        console.log(timeUnits + " left");
        timeUnits -= interval;

        if(timeUnits <= 0){
            clearInterval(iid);
            alert(handleCalls.orgTU + " has been counted.");
        }
    }
})();
```



# CLOSURE ques

```
function powerCreator(exp){

    let fun = function(base){
        let rv = Math.pow(base, exp);
        return rv;
    }

    return fun;

}

let squarer = powerCreator(2);
let val = squarer(8);
console.log(val);

// how can you change squarer to cuber without calling powerCreator again
// you can change powercreator

// change powerCreator
// to make it a producer of such functions
// whose exponent we can change on a later date
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
→  Lecture_44 git:(main) ✗ node 4_Closure_Ques_1.js
64
```

```
2   function powerCreator(obj){
3
4       let fun = function(base){
5           let rv = Math.pow(base, obj.exp);
6           return rv;
7       }
8
9       return fun;
10
11  }
12
13  let o1 = {
14      exp: 2
15  }
16
17  let squarer = powerCreator(o1);
18  let val = squarer(8);
19  console.log(val);
20
21  o1.exp = 3;
22  let val2 = squarer(7);
23  console.log(val2);
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
→  Lecture_44 git:(main) ✗ node 5_Closure_ans.js
64
343
→  Lecture_44 git:(main) ✗
```

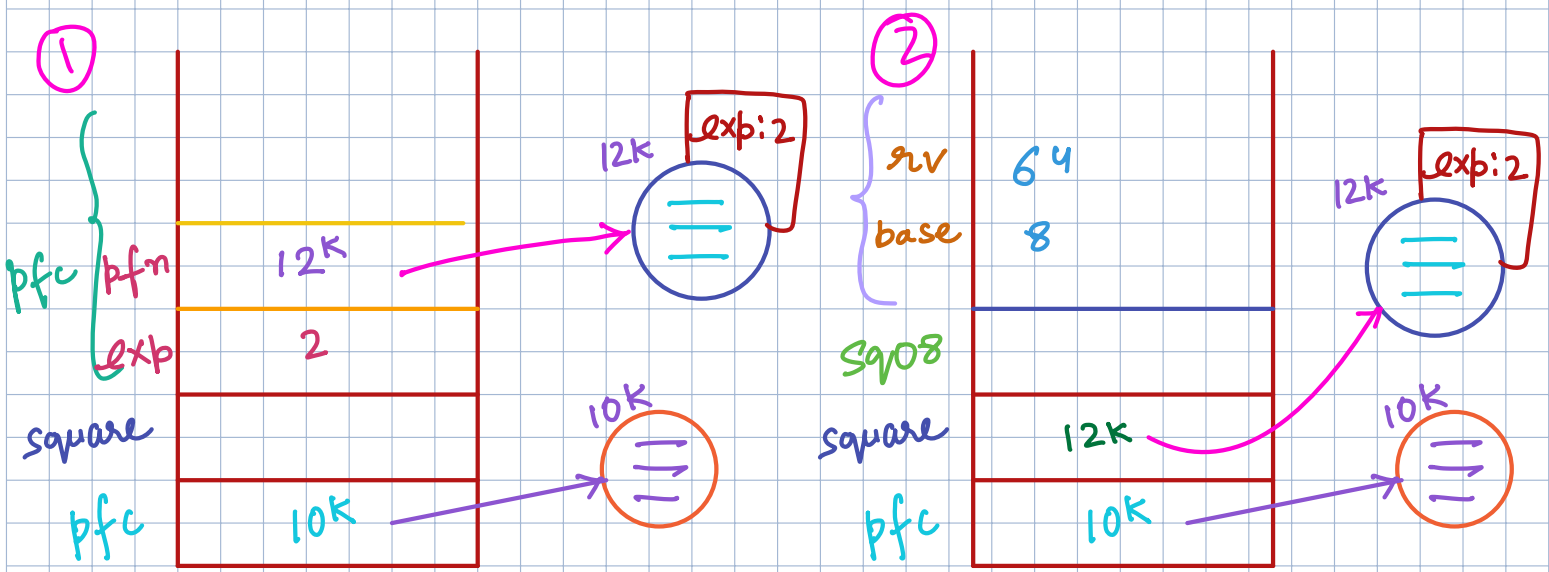
# closure demo → Memory Map diagram

```
2
3   function powerFunctionCreator(exp) {
4       if (typeof exp !== 'number') {
5           console.log("exp must be a number.")
6           return null;
7       }
8
9       let powerFn = function(base) {
10          let rv = Math.pow(base, exp);
11          return rv;
12      }
13      return powerFn;
14  }
15
16  let squarer = powerFunctionCreator(2);
17  let sqo8 = squarer(8);
18  console.log(sqo8);
```

12K return

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

→  Lecture_44 git:(main) ✗ node 6_Closure_demo.js
64
→  Lecture_44 git:(main) ✗
```



```
2   function powerFunctionCreator(obj) {
3       if (typeof obj.exp !== 'number') {
4           console.log("exp must be a number.")
5           return null;
6       }
7
8       let powerFn = function(base) {
9           let rv = Math.pow(base, obj.exp);
10          return rv;
11      }
12      return powerFn;
13  }
14
15  let obj = {
16      exp: 2
17  }
18
19  let squarer = powerFunctionCreator(obj);
20  let sqo8 = squarer(8);
21  console.log(sqo8);
22
23  obj.exp = 3;
24  let cuo8 = squarer(8);
25  console.log(cuo8);
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

→  Lecture_44 git:(main) ✗ node 7_Closure_demo2.js
64
512
→  Lecture_44 git:(main) ✗
```