1. **Exercise 01:**

setTimeout(() => { console.log("Hello Fresher Academy") }, 1000);

1. **Exercise 02:**

setInterval(() => { console.log("Hello Fresher Academy") }, 3000);

1. **Exercise 03:**

The above declare an anonimous function and call it.

That function call "console.log(1);" then "console.log(4);", so that the first two output is 1 and 4.

Because "console.log(2);" and "console.log(3);" is called asynchronously in callback queue, so that it come later tham 1 and 4.

Moreover "console.log(2);" is called by setTimeout with 1000 ms delay, and "console.log(3);"is called by setTimeout with 0 ms delay, so that 3 come before 2 in callback queue.

1. **Exercise 04:**

Case 1:

Output: A D B C

A and D were logged synchronously, so A was logged and D come later.

B and C were logged in callback queue, so that B and C come later than A and D.

Case 2:

Output: B D A C

B and D were logged in callback queue by callback function, but A and C were logged in the callback function of another setTimeout so that A and C come later that B and D.

Case 3:

Output: A D C B

Because the variable ‘x’ is declared by the ‘var’ key word it has function scope, so the value of ‘x’ is the newest value that assigned to ‘x’.

The callback of the third setTimeout is called first because it have lowest timeout ‘1’, so that ‘A’ is logged and x is updated to ‘D’

Next, the callback of the second setTimeout is called because the timeout, so that ‘D’ is logged and x is updated to ‘C’

Next, the callback of the first setTimeout is called because the timeout, so that ‘C’ is logged and x is updated to ‘D’

Finally, the callback of the last setTimeout is called because the timeout, so that ‘D’ is logged.

Case 4:

Output: C D

Because setTimeout t1 is cancel immediately after declare, A and B is never log.

However, the callback of setTimeout t2 is called before the clearTimeout function, so C is logged and D also be logged after that.