Node.JS:

The process flow here is based on the event loop. All call back functions are placed in different queues like timer queue, pending queue, immediate queue, Microtask queue. Then they are executed in different phases of the event loop.

Code:

setImmediate(() => { console.log("executes immediately"); });

setTimeout(() => { console.log("timeout"); },3000);

function example(text="default"){

console.log(text);

}

function fun(a,cb){

if(a>10) cb("Greater than 10");

else cb("less than 10");

}

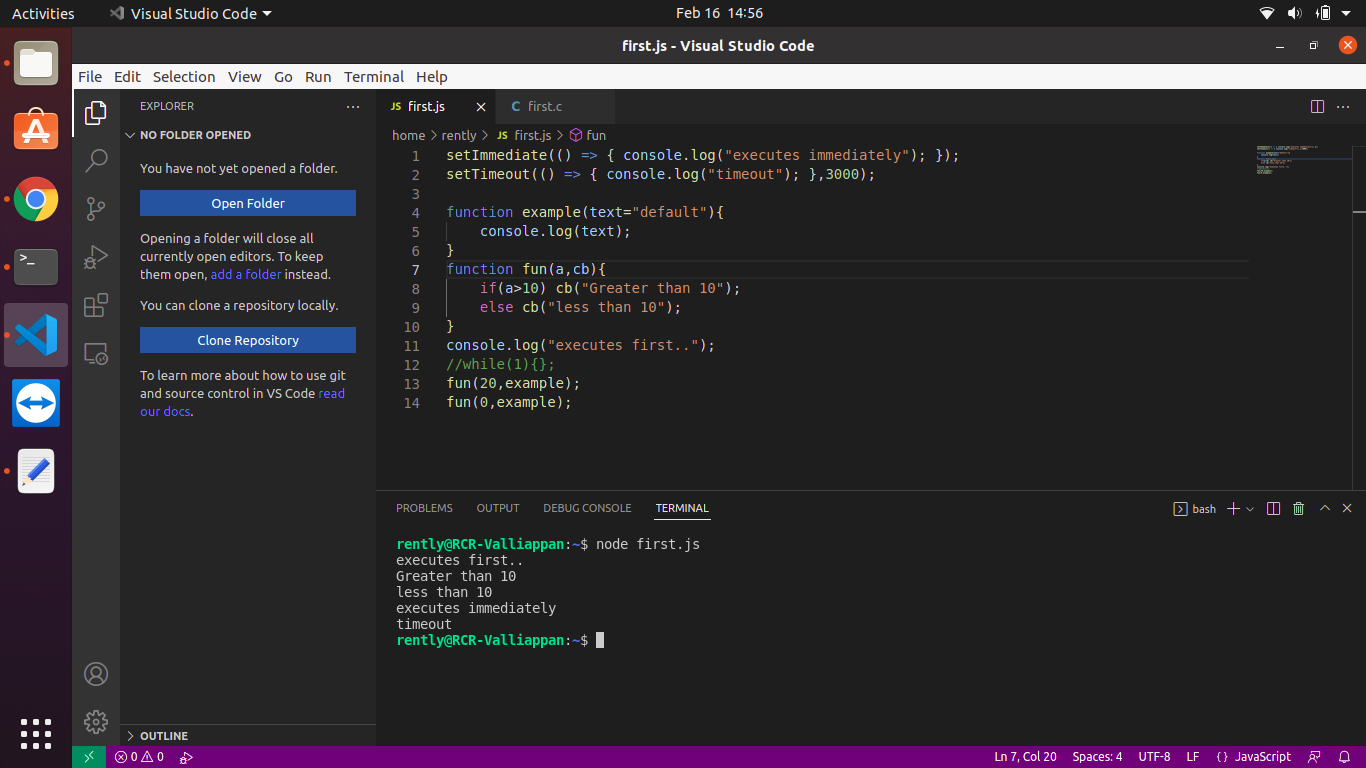
console.log("executes first..");

//while(1){};

fun(20,example);

fun(0,example);

Output:



It executes in a Non-blocking fashion. This is because of the event loop in Node.JS

C program:

In C, the program is executed in a blocking fashion. The program is executed sequentially and so here until the timeout function completes the code following it is not executed.

#include<stdio.h>

#include<time.h>

int timeout ( int seconds )

{

printf("Inside timeout\n");

clock\_t endwait;

endwait = clock () + seconds \* CLOCKS\_PER\_SEC ;

while (clock() < endwait) {}

return 1;

}

void display(){

printf("Just a print statement\n");

}

int main ()

{

printf("This statement prints first\n");

display();

timeout(3);

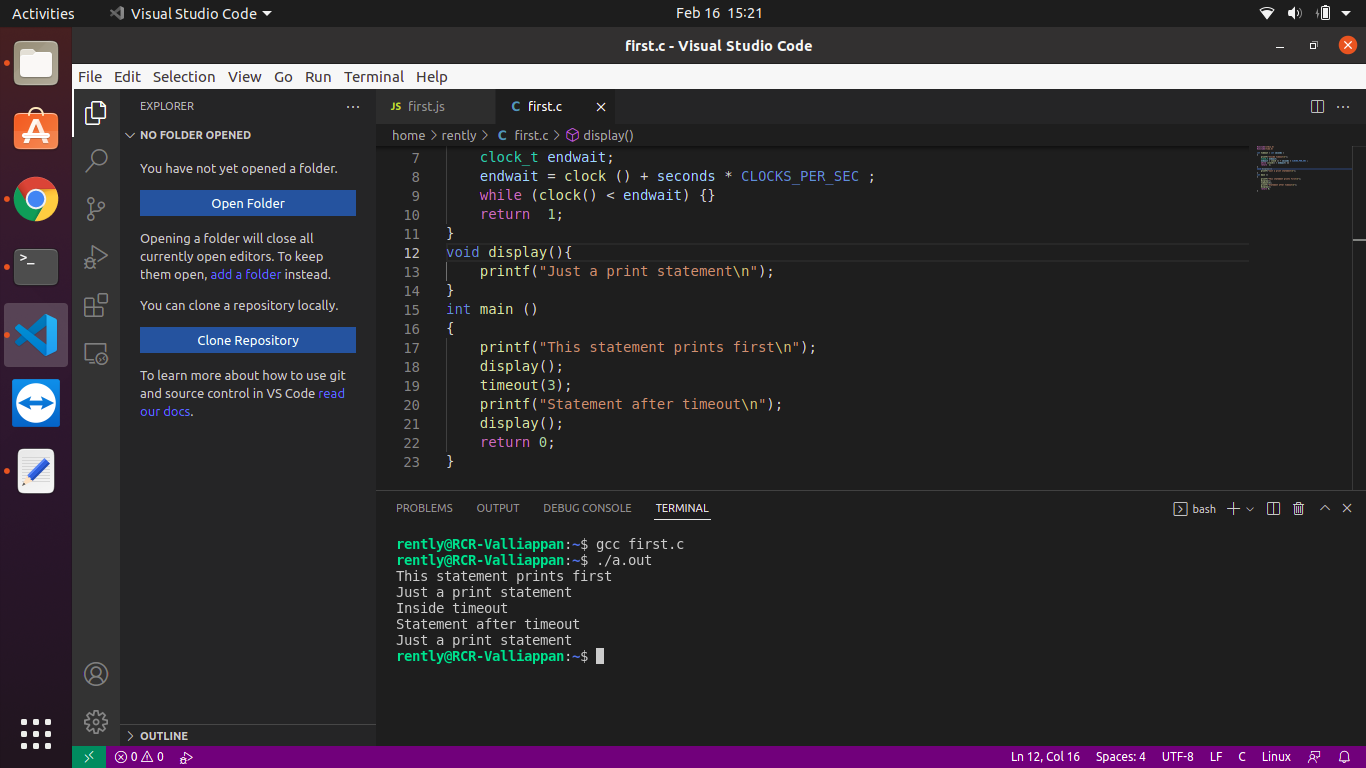
printf("Statement after timeout\n");

display();

return 0;

}

Output:



The control flow is sequential here.

Comparison:

-> So Node.JS uses the event loop for Non-blocking I/O. But the C program follows normal sequential flow control for Blocking-I/O.

-> The event loop is more efficient than the traditional way as it can handle multiple requests without blocking any.