CS472-2024-10A-10D-01 Web Application Programming

Lab 8: Node Event Loop

October 10th, 2024

Awais Waheed - 618677

1. **Please answer the following:**

   1. **What is LibUV?**
      libuv is a C library that was first created for Node.js to handle non-blocking I/O operations. It uses an event-driven, asynchronous I/O model, which means the CPU can keep working on other tasks while waiting for I/O operations to complete. This helps make better use of system resources, like the CPU and network. Instead of waiting for things like file reads or network requests to finish, libuv uses callbacks to let you know when those tasks are done, keeping things efficient and responsive.

   2. **Explain the difference between setImmediate(f) and setTimeout(f, Time)?**
      setImmediate(f) runs after I/O events have been processed, during the check phase of the event loop. So, it's designed for tasks that need to execute after I/O operations but as soon as possible in the next event loop iteration.

      setTimeout(f, t) runs after at least 0 milliseconds, during the timers phase of the event loop. Although the delay is set to 0, the callback is still queued and may be delayed by other timers or tasks.

      So in **summary**, **setImmediate(f)** executes after I/O events and is meant to run after the current event loop's I/O operations complete. **setTimeout(f, 0)** introduces a delay (even if 0ms), and its execution may be postponed by other tasks or timers.

   3. **Explain the difference between process.nextTick(f) and setImmediate(f)?**
      process.nextTick(f) runs before any I/O or timers in the current event loop, giving it the highest priority. It’s designed for tasks that need to be executed immediately after the current operation, but before any I/O events.

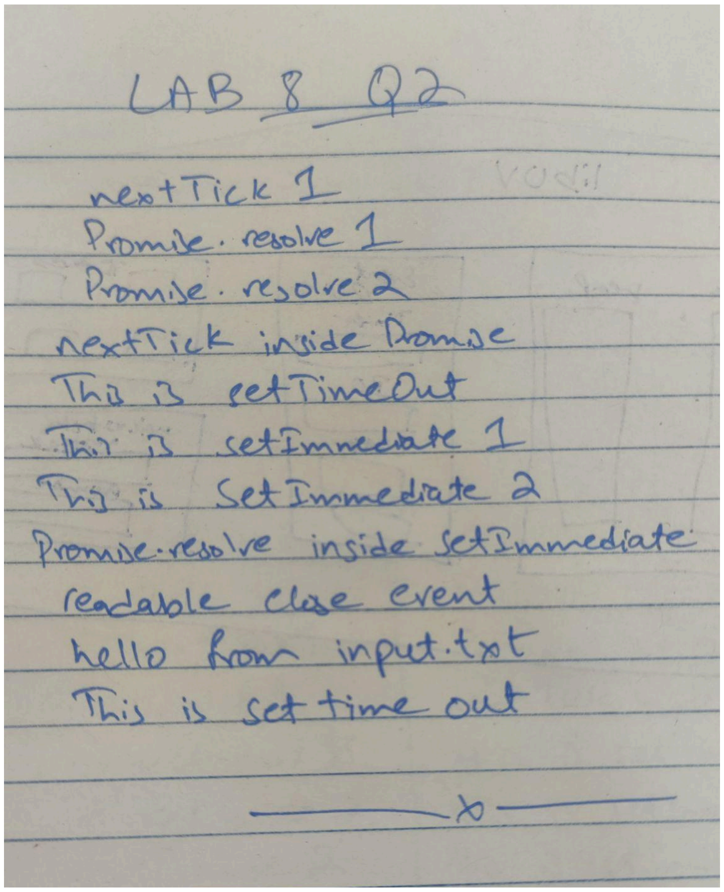
      setImmediate(f) runs after I/O events, during the check phase of the next event loop iteration. It’s ideal for tasks that can wait until the next event loop iteration but need to run as soon as possible.

      So in **summary**, **process.nextTick(f)** executes before I/O and timers, while **setImmediate(f**) runs after I/O events, in the next iteration.

2. Pls write down the output without executing the following code snippets and check it with the result.

```
const fs = require('fs');
//you may assume input.txt is in the same folder
const rd = fs.createReadStream("input.txt");
rd.close();
rd.on("close", () => console.log('readableStream close event'))
fs.readFile('input.txt', "utf-8", (error, data) => {
    if (error) console.log(error);
    else console.log(data)
});
setTimeout(() => console.log("this is setTimeout"), 5000);
setTimeout(() => console.log("this is setTimeout"), 0);

setImmediate(() => console.log("this is setImmediate 1"));
setImmediate(() => {
    console.log("this is setImmediate 2")
    Promise.resolve().then(() => console.log('Promise.resolve inside setImmediate'));
});
Promise.resolve().then(() => console.log('Promise.resolve 1'));
Promise.resolve().then(() => {
    console.log('Promise.resolve 2')
    process.nextTick(() => console.log('nextTick inside Promise'));
});
process.nextTick(() => console.log('nextTick 1'));
```



LAB 8 Q2

nextTick 1
Promise. resolve 1
Promise. resolve 2
nextTick inside Promise
This is setTimeOut
This is setImmediate 1
This is SetImmediate 2
Promise.resolve inside setImmediate
readable close event
hello from input.txt
This is set time out

—x—