MWA Homework 02 - NodeJS 01

Written Exercises

1. Explain why do we sometimes want to use `setImmediate` instead of using `setTimeout`?

**setImmediate -** will make the function asynchronous, meaning wait until all synchronous code finish execution. The main reason to use setImmediate instead of setTimeout is when you need to execute a code block as soon as possible [[1]](https://stackoverflow.com/questions/24117267/nodejs-settimeoutfn-0-vs-setimmediatefn), and it should not be delayed even by a single millisecond. setImmediate will execute the code block on the next cycle of the event loop, while setTimeout will wait for the specified delay to pass before it executes the code block. This makes setImmediate the better choice when you need to ensure that code is executed as quickly as possible, such as in animations or real-time applications. Additionally, setImmediate allows you to specify a callback that will be executed when the code is done executing, while setTimeout does not.

2. Explain the difference between `process.nextTick` and `setImmediate`?

The main difference between process.nextTick() and setImmediate() is the order in which they are executed. process.nextTick() is a low-level function that puts its callback at the head of the event queue and is fired immediately after the current operation completes. setImmediate(), on the other hand, puts its callback at the end of the event queue and is executed after I/O operations and timers have completed. Therefore, process.nextTick() will always be called before setImmediate()

3. Name 10 core modules that Node provides by default, and 10 of the Global objects?

Node.js provides a number of core modules by default, including:

1. fs: The file system module provides access to the underlying file system.
2. http: The HTTP module provides an interface for creating and managing web servers.
3. url: The URL module provides methods for creating and parsing URLs.
4. path: The path module provides utilities for working with file and directory paths.
5. querystring: The querystring module provides utilities for parsing and formatting query strings.
6. events: The events module provides the EventEmitter class for creating and managing event emitters.
7. stream: The stream module provides the Stream class for creating and managing streams of data.
8. util: The util module provides utility functions for working with objects, functions, and other data types.
9. buffer: The buffer module provides utility functions for working with binary data.
10. crypto: The crypto module provides a number of cryptographic functions.

Exercise 02

Write a function in Node `factorial(n)` to find asynchronously the factorial of a number. Write your observation (CPU, RAM.. etc) on what happens in `Node Process` when we calculate the factorial of a large number.

\* factorial(3) is 3 \* factorial(2)

\* factorial(2) is 2 \* factorial(1)

\* factorial(1) is 1 \* factorial(0)

\* factorial(0) is 1

```javascript

// your Node code here...

console.log('start');

factorial(3).then(console.log);

console.log('end');

// Test your code in Node.JS CLI, Output:

// start

// end

// 6

```

**Answer:-**

console.log(`Start`);

const factorial = (n) => {

return new Promise((resolve, reject) => {

if (n < 0) {

reject("Number must be a positive integer");

}

let result = 1;

for (let i = 1; i <= n; i++) {

result \*= i;

}

resolve(result);

});

}

factorial(3).then(result => { console.log(result); });

console.log(`End`);