Server-side Programming in Node.js

CS5610: Web Development

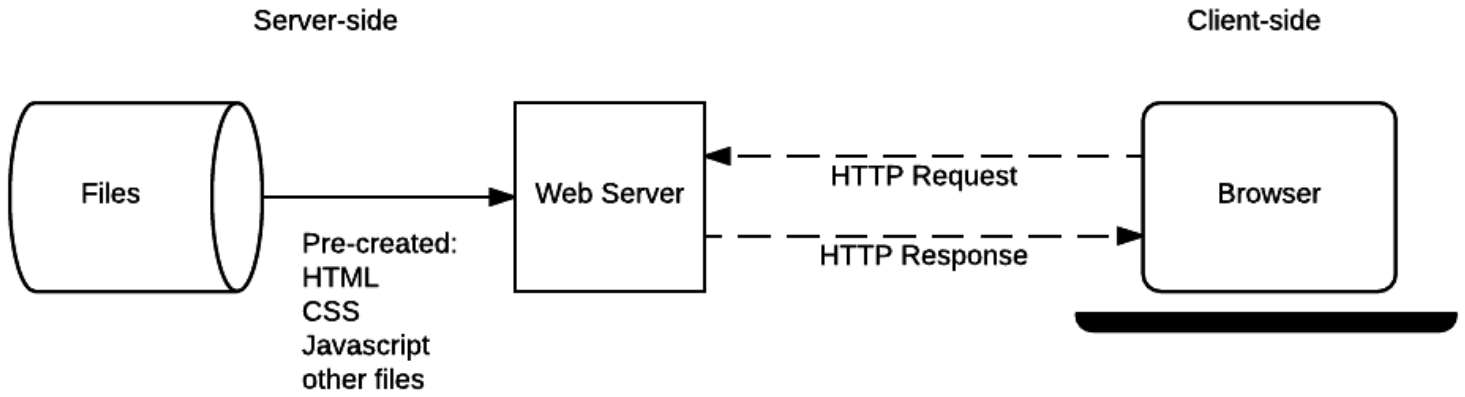
Joydeep Mitra

Pre-class Activity

- Clone the repo at https://github.com/CSE-316-Software-Development/learn-node.git
- Fork the repo.
- Create a new branch with today's date.
  - Submit all quiz activities and submit the repo link with branch name to Canvas.

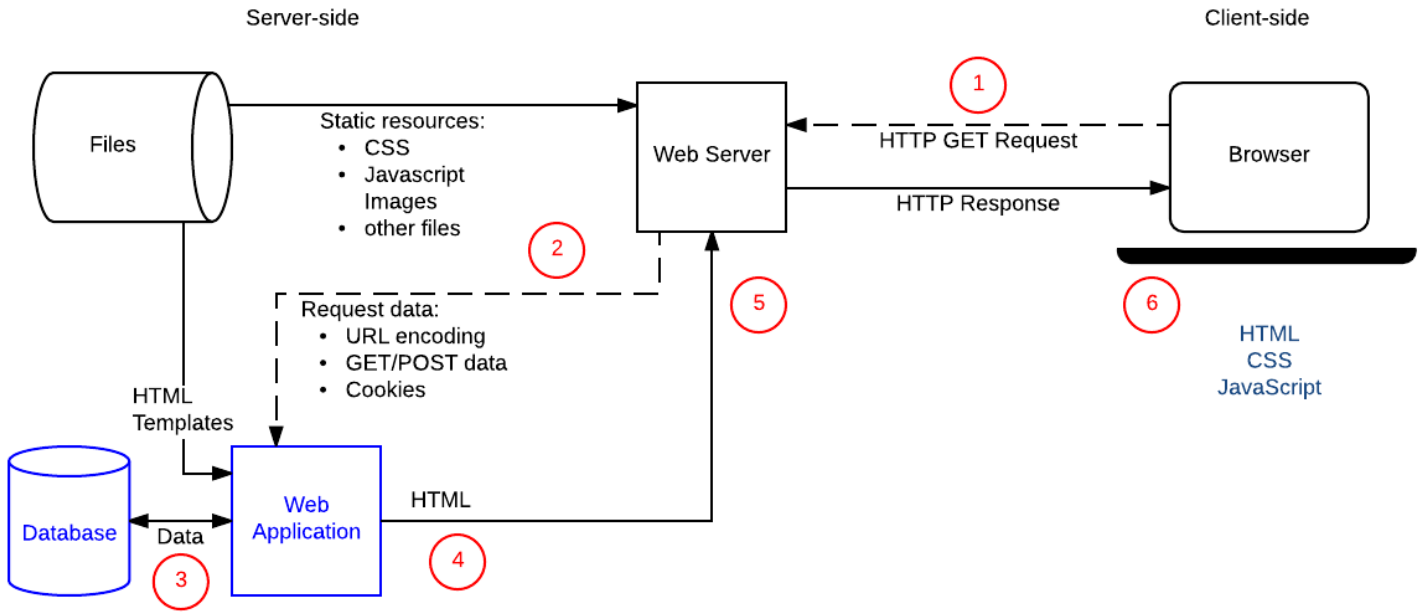
What is a Web Server?

- A Web server is a computer that stores the web server software and the resources of a website (e.g., images, HTML, CSS, and JavaScript).
- The web server software is a program that controls how the resources are accessed by several clients (e.g., browsers).
- Web browsers communicate with web servers using the HTTP protocol.



Static vs. Dynamic Web Server

- A static web server returns the same content for a particular request.
- A dynamic server allows us to send custom content for a request based on inputs provided by a user.



Server-side Programming in Node.js

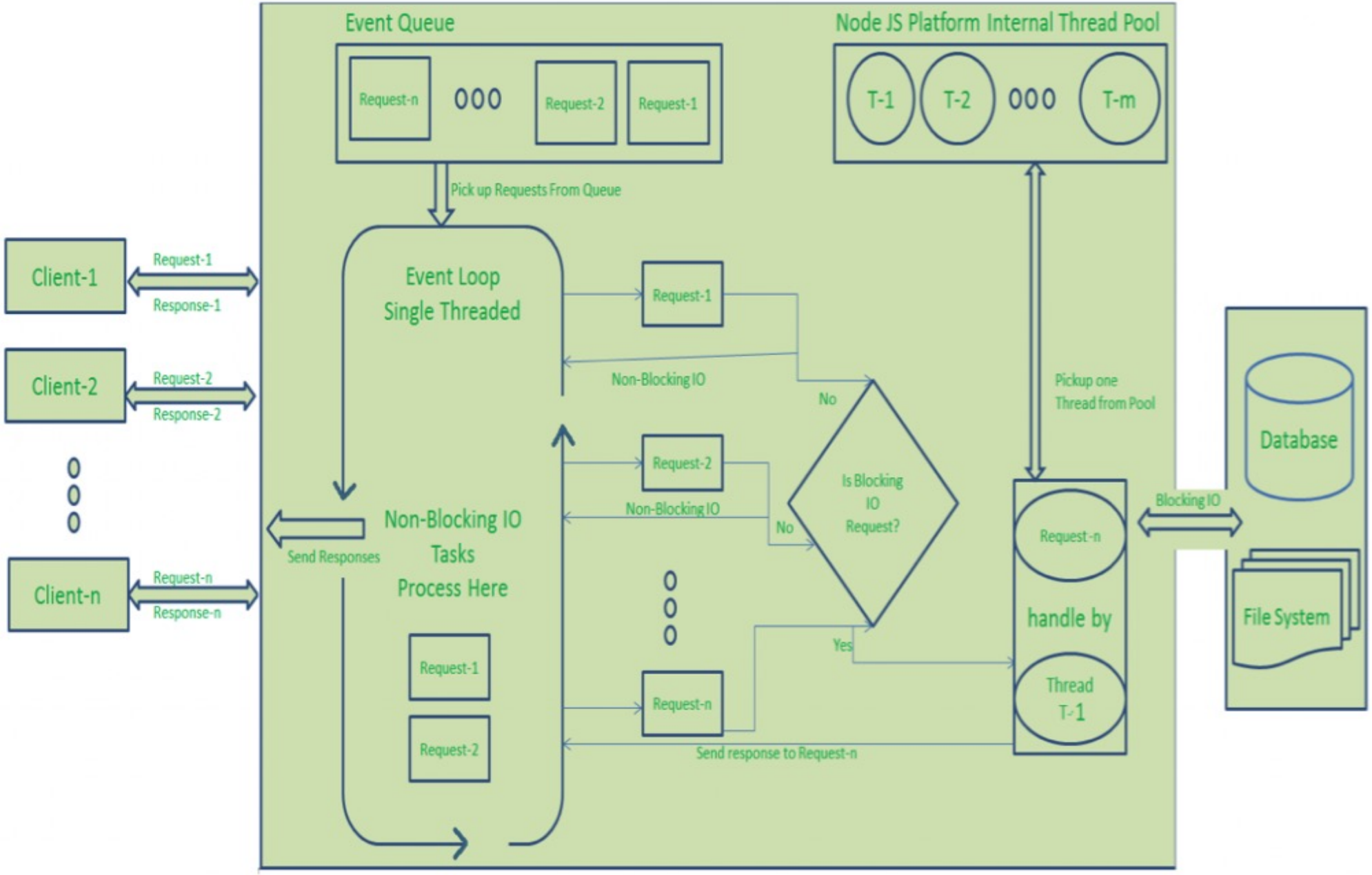
- Node.js is an open-source runtime environment for executing JavaScript outside the browser.
  - *has several libraries and in-built APIs that facilitate the programming and execution of server-related tasks such as file management, database communication, etc.*
- The example in *basics/helloserver.js* shows an implementation of a NodeJS server.
  - Note why and how the *http* library is used.
  - Note why and how the *process* library used.
  - Note how we export and import our custom config module.

For You to Do

- Change the code in *basics/quiz/helloserver.js* such that the server returns a successful response with a json object containing the hostname and the port on which the server is running if the request URL does not end with ‘/home’.

NodeJS Architecture

- Node.js is a **single threaded event loop architecture**
  - A single thread of execution.
  - Each incoming requests is processed synchronously.
    - Long running tasks are delegated to a worker thread selected from a pool of finite worker threads.
  - Different from a classic multi-threaded architectures.



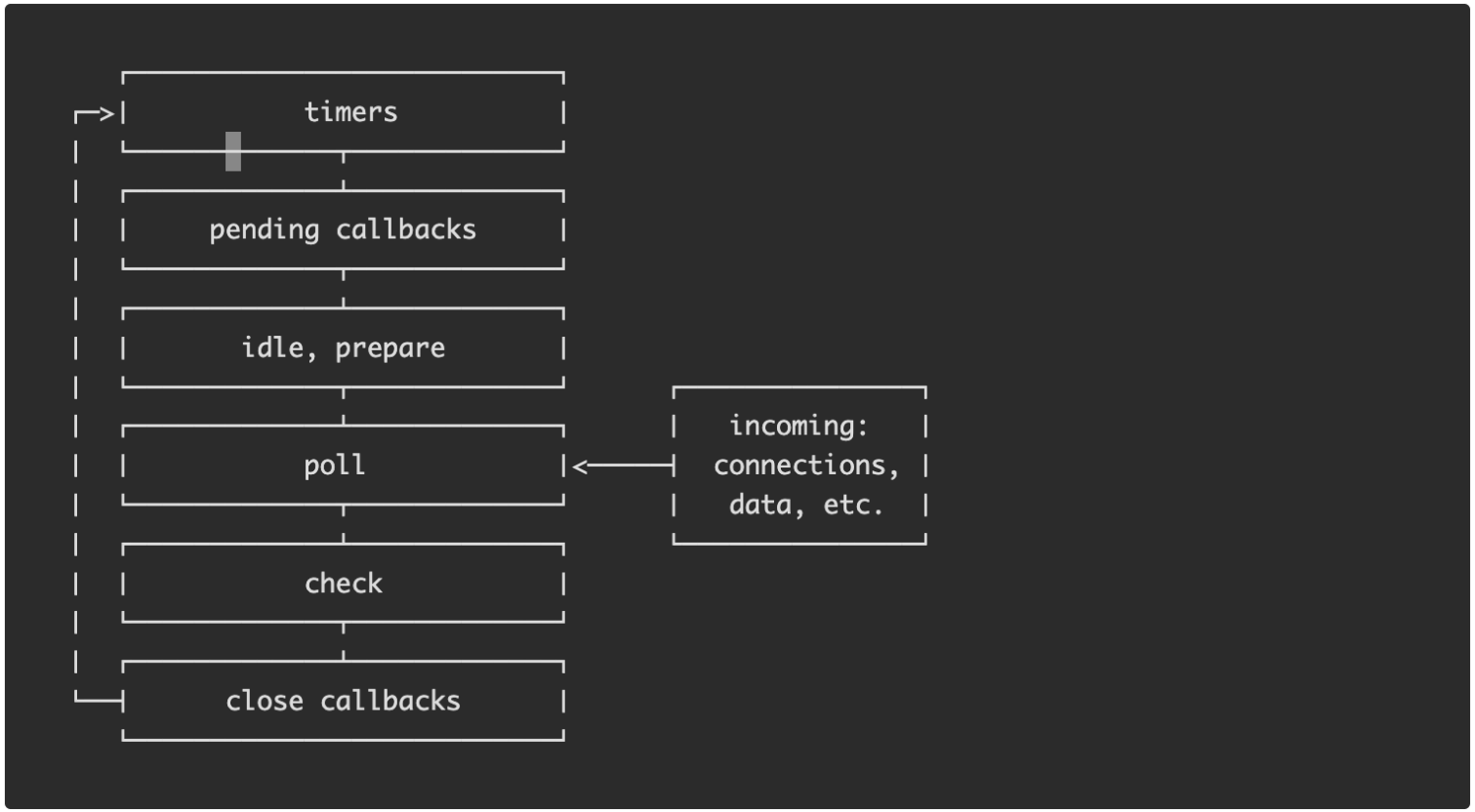
Node JS Application/Node JS Server

For You to Do

- Explore the code in "operations/blocking.js" and "operations/nonblocking.js". What will be the order of the messages logged in the console. Why?
- There is runtime error in "operations/quiz/syncdelete.js”. What is the error? Can you fix it?

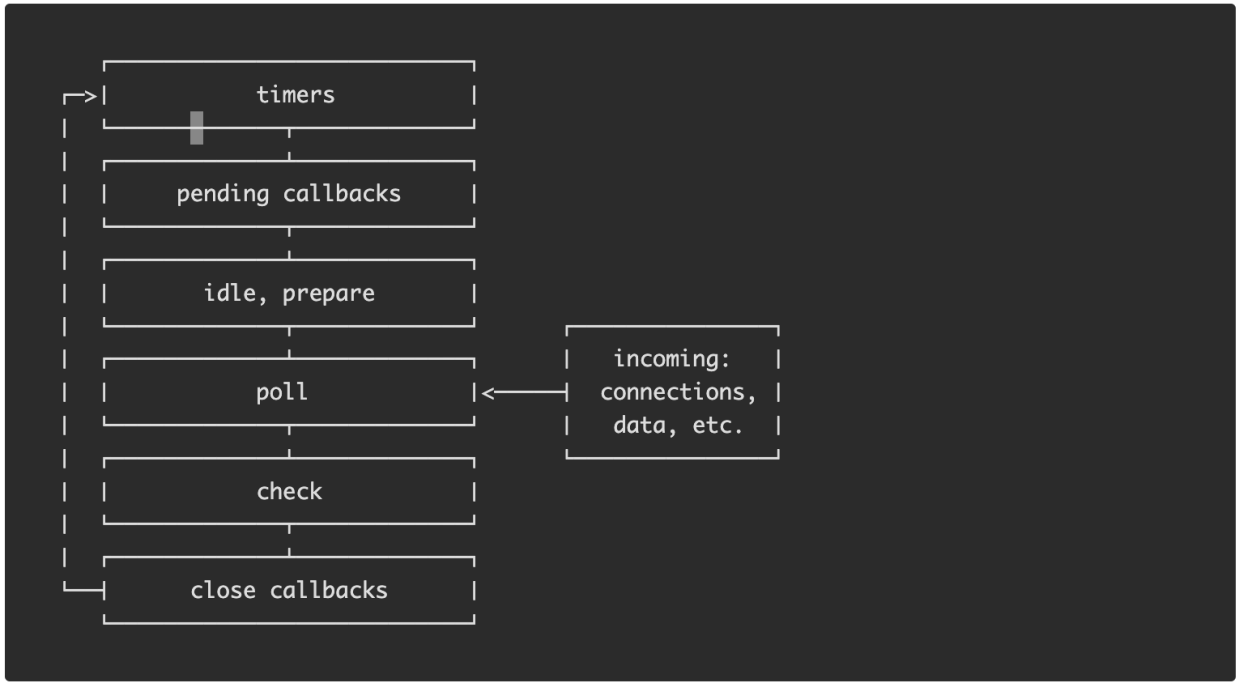
Understanding the Event Loop

- On server start, Node:
  - Initializes the event loop.
  - Runs the input script (server).
  - Runs the event loop.
- The event loop has several phases.



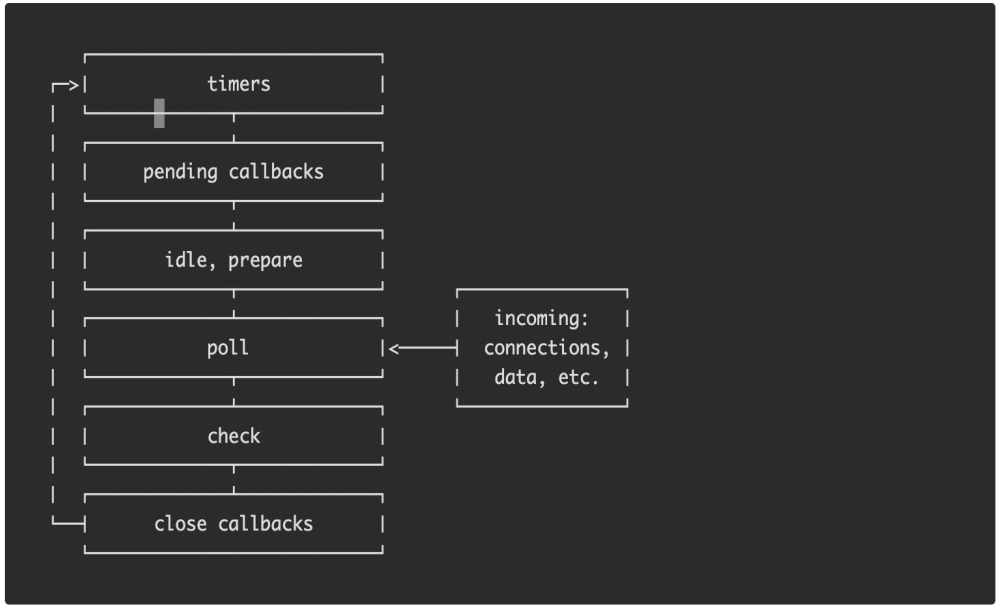
Phases of the Event Loop

- **timers**: queue of callbacks scheduled by `setTimeout()` and `setInterval()`.
- **pending callbacks***:* queue of callbacks that are complete and ready to execute.
- *Idle, prepare:* used internally
- **poll**: waits for I/O events and timer events, executes related pending callbacks when complete.
- **check**: queue of `setImmediate()` callbacks
- **close callbacks***:* queue of close event callbacks (e.g., `socket.on('close', function() { … })`



The **poll** Phase

- The event loop waits for events in the poll phase (**polling**).
  - If I/O event, then process it.
  - Otherwise, run callbacks in *pending callbacks.*
- If an I/O event is complete move corresponding callback to *pending callbacks.*
- Otherwise,
  - move *check* callbacks to *pending callbacks.*
  - If a timer has expired move the callback to *pending callbacks*.
- I/O events always have priority, followed by check, and then timer.



For You to Do

- In *eventloop/timer.js*, what will be the order of execution? Briefly explain the order of execution w.r.t the event loop phases.
  - Describe your answer in *quiz/fytd-timer.txt.*
- In *eventloop/poll.js*, what will be the order of execution? Briefly explain the order of execution w.r.t the event loop phases.
  - Describe your answer in *quiz/fytd-poll.txt.*

For You to Do

- Explain the order of execution in terms of the event loop for *eventloop/poll_timer.js*.
  - Will the order of execution change if the delay in the while loop is changed from 10s to 150s? Why or why not?
  - Will the order of execution change if timeout is changed to 0? Why or why not?
- Describe your answer in *quiz/fytd-poll-timer.txt*

For You to Do

- Explain the order of execution in terms of the event loop for *eventloop/immediate.js*.
  - Will the order of execution change if timeout is changed to 0? Why or why not?
  - Change the script in *eventloop/quiz/immediate.js* such that the immediate callback runs first.
- Describe your answer in *quiz/fytd-poll-immediate.txt.*

Event Emitters

- Event emitters are used to define custom events, which can be used to trigger specific activities such as cleaning up memory or closing files or closing network connections.
- Consider the example in *eventloop/eventemit.js*.
  - Defines and handles event to cleanup timers.
  - Cleaning up timers is recommended to prevent memory leaks.

Points to Note

- Server-side Programming allows us to manage and deliver custom content to client applications.
- Node.js provides a single-threaded runtime environment to write server-side programs in JavaScript.
- The event loop and its phases makes it easy to write and manage code that executes asynchronously.
- Long running tasks on the event loop blocks it, which degrades performance.
  - It is the programmer's responsibility to ensure that the event loop is not blocked.