🗓️ Day 6: Understanding the Event Loop

✅ The Event Loop is how Node.js handles **asynchronous tasks** without blocking.
✅ Node is **single-threaded** but can manage many operations **in parallel** via the event loop.

✅ Phases of the Event Loop

- **Timers:** setTimeout, setInterval
- **Pending Callbacks:** System callbacks
- **Idle/Prepare:** Internal
- **Poll:** Waiting for new I/O
- **Check:** setImmediate
- **Close Callbacks:** socket.on('close')

✅ Example:

```
js
CopyEdit
console.log('Start');

setTimeout(() => {
  console.log('setTimeout');
}, 0);

setImmediate(() => {
  console.log('setImmediate');
});

process.nextTick(() => {
  console.log('process.nextTick');
});

console.log('End');
```

✅ Expected Output:

```
arduino
CopyEdit
Start
End
process.nextTick
setTimeout
setImmediate
```

✅ Explanation:

- `process.nextTick` runs *before* the next phase.
- `setTimeout` waits for the **Timers phase**.
- `setImmediate` runs in the **Check phase**.

✅ Why learn this?

- Understand when your callbacks run.
- Avoid blocking the event loop.
- Write responsive servers.

✅ Mini Task:

- Write your own script using `setTimeout`, `setImmediate`, and `process.nextTick`.
- Predict the output before running!