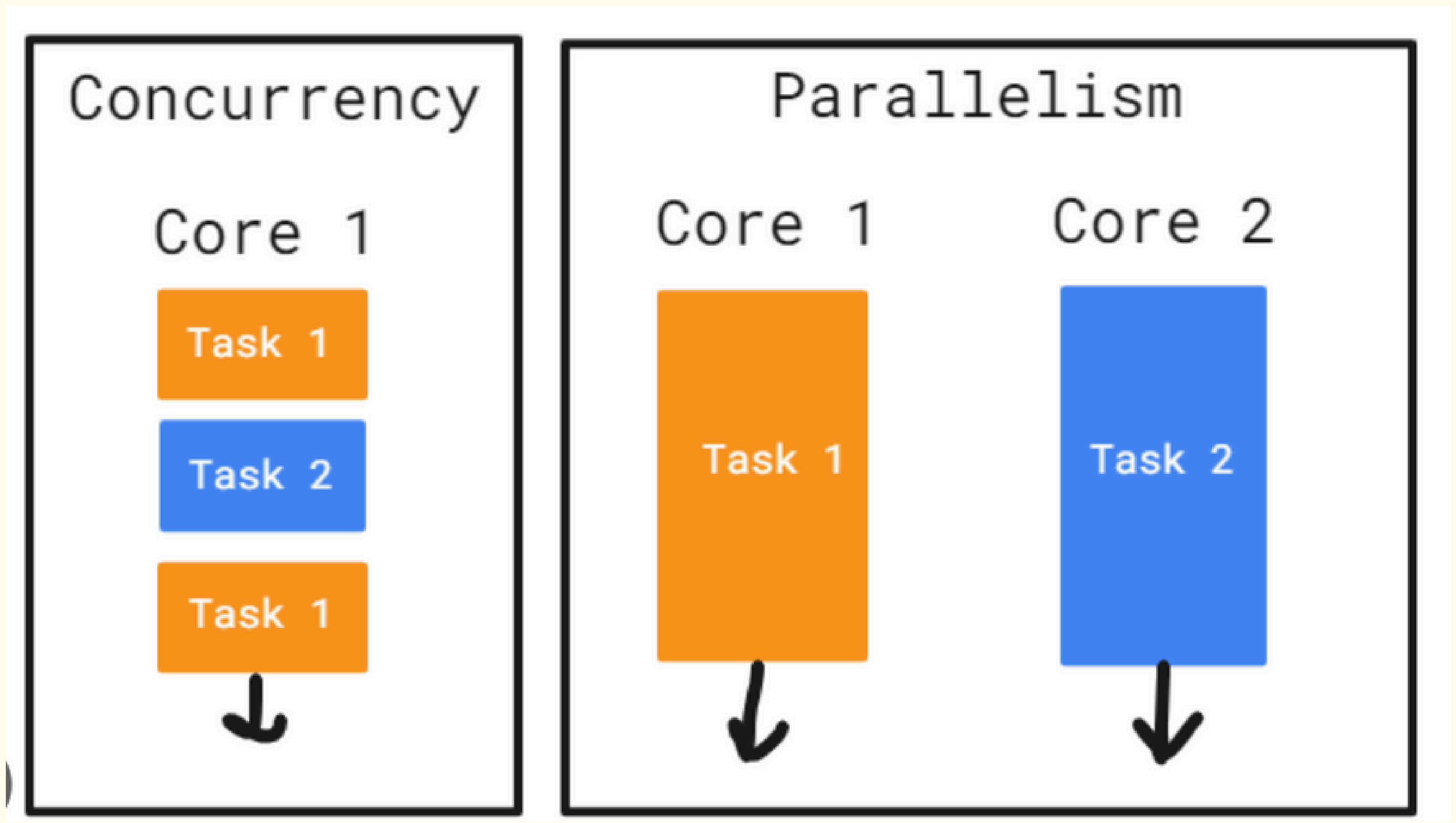
Parallel Programming in JavaScript:

Interview Question

What is Parallel Programming?

Parallel programming means running multiple tasks at the same time to improve performance, typically by using multiple CPU cores or threads. It’s about simultaneous execution of code to speed up processing.



JavaScript and Parallelism

- JavaScript itself is **single-threaded,** meaning all code runs on one main thread.

- However, JavaScript can still handle multiple operations at the same time through **asynchronous APIs** and **web platform features.**

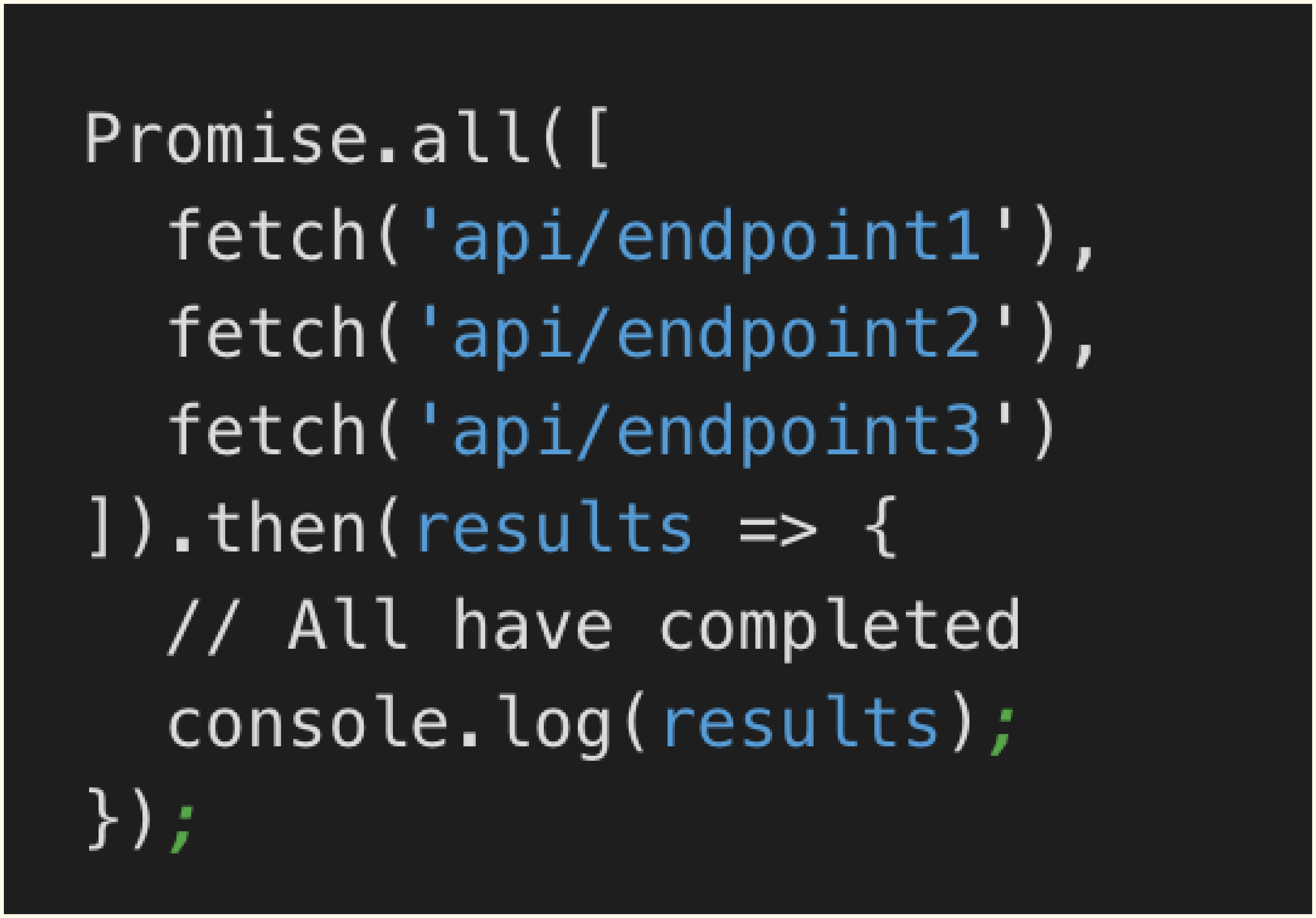
- True parallel execution in JavaScript is possible via **Web Workers** (in browsers) or **Worker Threads** (in Node.js).

How Does JavaScript Achieve Parallel Work?

Asynchronous Tasks Running in Parallel (Logical Parallelism)

- When you start multiple asynchronous operations (like fetching data), these underlying tasks run in parallel via the browser’s or Node.js’s thread pool or system APIs.

- You can start several promises at once and wait for all of them with **Promise.all()**. This isn’t true multithreading in JS itself but achieves similar benefits by letting tasks run independently outside the main thread.



```
Promise.all([
  fetch('api/endpoint1'),
  fetch('api/endpoint2'),
  fetch('api/endpoint3')
]).then(results => {
  // All have completed
  console.log(results);
});
```

Web Workers — True Parallel Threads

- For CPU-intensive tasks or trulyparallel code execution, JavaScript supports Web Workers.
- These spawn separate threads that run independently, allowing parallel processing without blocking the main UI thread.

When to Use Parallel Programming in JS?

- Running multiple API calls simultaneously **(Promise.all()).**
- Processing large data sets or complex computations without freezing the UI **(Web Workers).**
- Handling file processing, image or video manipulation, or heavy calculations.

Follow on

@Duvvuru Kishore