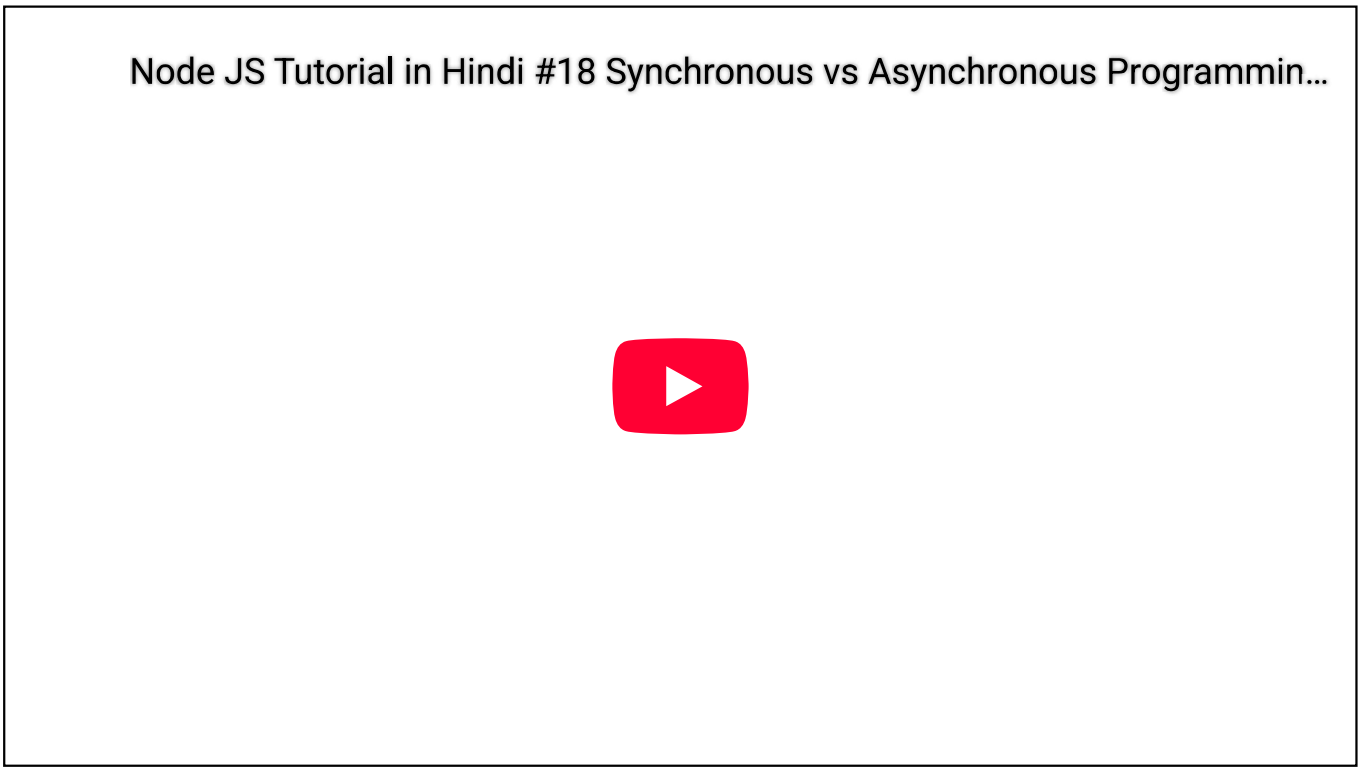
Synchronous vs Asynchronous Programming in node and Javascript



In most programming languages, operations run **synchronously** — one task at a time. But in JavaScript and Node.js, we often use **asynchronous programming** for better speed and performance.

Synchronous Programming

Executes **one task at a time**.

The next line waits until the current one finishes.

Slower in terms of overall performance.

```
console.log("Apple1");
console.log("Apple2");
console.log("Apple3");
```

Each task waits for the previous one to complete.

Asynchronous Programming

Doesn't wait for a task to finish before moving on.

Great for **I/O operations** (file, API, database).

Faster in real-world applications.

```
console.log("Apple1");

setTimeout(() => {
  console.log("Apple2");
}, 2000);

console.log("Apple3");
```

`setTimeout()` runs separately and doesn't block the next line.

When Not to Use Asynchronous Code?

Sometimes, we want code to **wait** until one step is complete.

```
let a = 20;
let b = 0;

setTimeout(() => {
  b = 100;
}, 2000);
```

```
console.log(a + b); // Output: 20 (Not 120!)
```

Here, we **don't need** async because we want b before using it.

Real-Life Async Example

Reading a file without blocking the main thread:

```
onst fs = require("fs");

fs.readFile("text/peter.txt", "utf-8", (err, data) => {
  if (err) return false;
  console.log(data);
});

console.log("End Script");
```

Read FIle in Sync

```
const data = fs.readFileSync("text/peter.txt", "utf-8");

console.log(data);
```