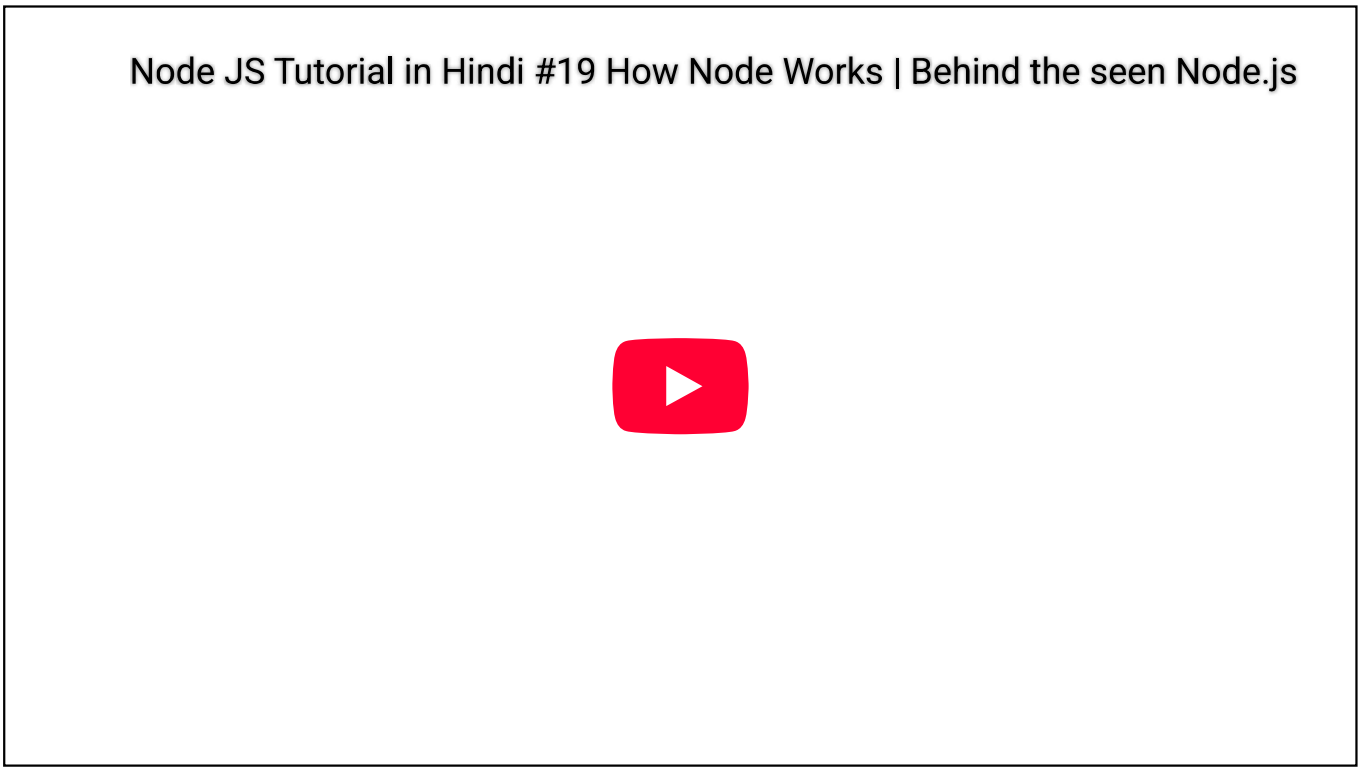
How Node Works Behind the seen Node.js



Let's understand how Node.js works behind the scenes — especially how the **Call Stack**, **Node APIs**, and **Event Loop** come together.

Main Function Execution

When Node.js runs code, it first registers the **main()** function in the **call stack**. Then other functions are called in order.

```
const x = 1;
const y = x + 2;
console.log("sum is " + y);
```

`main()` is added to the call stack automatically.

After `console.log()`, it exits the stack.

Only `main()` remains and exits last.

Example with a Function

```
const listLocations = (locations) => {
  locations.forEach((location) => {
    console.log(location);
  });
};

const myLocations = ["philly", "nyc"];

listLocations(myLocations);
```

Call Stack Flow:

`listLocations` enters call stack

`forEach` runs each item

`console.log("philly")`

Stack clears, then next item:

`console.log("nyc")`

Stack becomes empty

Node APIs & setTimeout

Node APIs handle asynchronous code like `setTimeout`.

```
console.log("Starting up");

setTimeout(() => {
  console.log("Two Seconds");
}, 2000);
```

```
setTimeout(() => {
  console.log("Zero Seconds");
}, 0);

console.log("Finishing up");
```

Call Stack Output Order:

Starting up
Finishing up
(then Node APIs hold timers)

Event Loop then executes callbacks when the call stack is empty:

3. Zero Seconds

4. Two Seconds

Event Loop

Keeps checking if the **call stack** is empty.
If empty, moves code from **callback queue** to **call stack**.