

# Chapter 1: Introduction to Node.js

---

## Table of Contents

- [What is Node.js?](#)
  - [Why use Node.js?](#)
  - [Installing Node.js & npm](#)
  - [Understanding the Node.js Runtime and V8 Engine](#)
  - [First Node.js Script](#)
  - [Understanding process, \\_\\_dirname, \\_\\_filename](#)
- 

## What is Node.js?


Imagine you're at a **dhaba** (roadside restaurant) where one waiter serves multiple tables. This waiter doesn't wait for one table to finish eating before taking orders from other tables. He's efficient, quick, and handles everyone simultaneously.

**Node.js is exactly like that smart waiter!**

Node.js is a **JavaScript runtime environment** that allows you to run JavaScript code outside of a web browser. Think of it as a special kitchen where JavaScript can cook up amazing applications.

Key Points:

- **JavaScript Everywhere:** You can now use JavaScript for both frontend (what users see) and backend (server-side) development
- **Single Language:** No need to learn different languages for different parts of your application
- **Fast & Efficient:** Built on Google's V8 engine (the same engine that powers Chrome browser)

 **Real-life Analogy:** Just like how a dhaba waiter can handle multiple customers without getting stuck on one order, Node.js can handle multiple requests without blocking.

---

## Why use Node.js?

### 1. Event-Driven Architecture

Think of Node.js like a **smart traffic signal system**:

```
// Traditional approach (like old traffic signals)
// Stop all traffic, let one car pass, then let the next car pass

// Node.js approach (like smart traffic signals)
// All cars can move simultaneously when their turn comes
```

**What this means:** Instead of waiting for one task to complete before starting another, Node.js can handle multiple tasks at the same time.

## 2. Non-Blocking I/O 🚀

Imagine you're ordering food online:

### Old Way (Blocking):

1. Order pizza online
2. Wait at computer screen for 30 minutes
3. Can't do anything else while waiting
4. Pizza arrives

### Node.js Way (Non-blocking):

1. Order pizza online
2. Get confirmation
3. Go watch TV, play games, work
4. Pizza arrives when ready

```
// Blocking (old way)
const result = readFileSync('data.txt'); // Everything stops here
console.log('This waits until file is read');

// Non-blocking (Node.js way)
readFile('data.txt', (err, data) => {
  console.log('File read complete');
});
console.log('This runs immediately!');
```

## 3. Single-Threaded but Powerful 🍷

Think of Node.js like a **super-efficient dabbawala** (lunch delivery person):

- **One person** (single thread) but **very organized**
- Uses **event loop** (like a smart delivery route)
- Handles multiple deliveries efficiently
- Never gets stuck on one delivery

💡 **Why Single-Threaded?:** Just like how one dabbawala can deliver 50+ lunch boxes efficiently, one Node.js thread can handle hundreds of requests!

---

## Installing Node.js & npm 🛠️

### Step 1: Download Node.js

1. Go to [nodejs.org](https://nodejs.org)
2. Download the **LTS version** (Long Term Support - like a stable recipe)

### 3. Run the installer


## Step 2: Verify Installation

Open your terminal/command prompt and type:

```
node --version  
npm --version
```

You should see something like:

```
v18.17.0  
9.6.7
```

What is npm? 


**npm** (Node Package Manager) is like a **giant online grocery store** for code:

- **Packages:** Ready-made code pieces (like pre-cut vegetables)
- **Dependencies:** Code your project needs (like ingredients for a recipe)
- **Scripts:** Automated tasks (like cooking instructions)

```
# Installing a package (like buying ingredients)  
npm install express  
  
# Creating a new project (like setting up a new kitchen)  
npm init
```

---

## Understanding the Node.js Runtime and V8 Engine

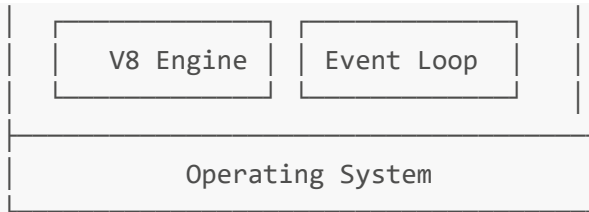
The V8 Engine 

Think of V8 as the **powerful engine** in a sports car:

- **Google Chrome's Engine:** The same engine that runs your Chrome browser
- **Super Fast:** Compiles JavaScript to machine code quickly
- **Optimized:** Constantly improved by Google engineers

Node.js Runtime Architecture 





**Real-life Analogy:** Like a **smart restaurant kitchen**:

- **V8 Engine:** The powerful stove and cooking equipment
- **Event Loop:** The head chef who manages all cooking tasks
- **Node.js Runtime:** The entire kitchen system
- **Your Code:** The recipes you want to cook

---

## First Node.js Script 🎬

### 1. Hello World! 🖱️

Create a file called `hello.js`:

```
console.log("Hello, Node.js World! 🚀");  
console.log("Welcome to the amazing world of backend development!");
```

Run it:

```
node hello.js
```

### 2. Creating a Simple Server 🌐

Think of this like opening a **small restaurant**:

```
// Import the http module (like getting cooking tools)  
const http = require('http');  
  
// Create a server (like setting up your restaurant)  
const server = http.createServer((req, res) => {  
  // Set response headers (like setting up the dining table)  
  res.writeHead(200, { 'Content-Type': 'text/html' });  
  
  // Send response (like serving food)  
  res.end(`  
    <html>  
      <head>  
        <title>My First Node.js Server</title>  
      </head>  
      <body>
```

```

        <h1>🍴 Welcome to My Node.js Restaurant!</h1>
        <p>Your server is running successfully!</p>
        <p>Current time: ${new Date().toLocaleString()}</p>
    </body>
</html>
`);
});

// Start the server (like opening the restaurant)
const PORT = 3000;
server.listen(PORT, () => {
    console.log(`🍴 Restaurant is open! Visit: http://localhost:${PORT}`);
    console.log(`🕒 Server started at: ${new Date().toLocaleString()}`);
});

```

### What this does:

1. **Creates a server** (like opening a restaurant)
2. **Listens for requests** (like waiting for customers)
3. **Serves HTML content** (like serving food)
4. **Runs on port 3000** (like having a specific address)

### 3. Running Your Server 🏃

```
node server.js
```

Then open your browser and go to: <http://localhost:3000>

💡 **Pro Tip:** `localhost` means "this computer" - like saying "my house" instead of your full address.

## Understanding process, \_\_dirname, \_\_filename 📁

### The process Object 🔑

Think of `process` as your **application's ID card**:

```

console.log('Process ID:', process.pid);           // Like your Aadhar number
console.log('Node.js Version:', process.version);  // Like your age
console.log('Platform:', process.platform);        // Like your city
console.log('Current Directory:', process.cwd());   // Like your current location

```

**Real-life Analogy:** Just like how you have personal information (name, age, address), your Node.js application has process information.

### \_\_dirname and \_\_filename 📁

Think of these like **GPS coordinates** for your files:

```
console.log('Current file location:', __filename);
console.log('Current folder location:', __dirname);
```

### Real-life Analogy:

- `__filename` = Your exact GPS coordinates
- `__dirname` = Your neighborhood/area

### Practical Example: File Paths

```
const path = require('path');

// Getting the current file's directory
console.log('File directory:', __dirname);

// Creating a path to a file in the same folder
const dataFile = path.join(__dirname, 'data.txt');
console.log('Data file path:', dataFile);

// Getting file information
console.log('File name:', path.basename(__filename));
console.log('File extension:', path.extname(__filename));
```

### Complete Example: File Explorer

```
const fs = require('fs');
const path = require('path');

console.log('🔍 File Explorer Information:');
console.log('=====');
console.log(`📁 Current Directory: ${process.cwd()}`);
console.log(`📄 Current File: ${__filename}`);
console.log(`📁 File Directory: ${__dirname}`);
console.log(`🔗 File Name: ${path.basename(__filename)}`);
console.log(`📄 File Extension: ${path.extname(__filename)}`);

// List files in current directory
console.log(`\n📁 Files in current directory:');
fs.readdir(__dirname, (err, files) => {
  if (err) {
    console.error('❌ Error reading directory:', err);
    return;
  }

  files.forEach(file => {
    const filePath = path.join(__dirname, file);
```

```
const stats = fs.statSync(filePath);
const type = stats.isDirectory() ? '📁' : '📄';
console.log(`${type} ${file}`);
});
});
```

---

## Summary

### What We Learned:

- ☑ **Node.js** is a JavaScript runtime for server-side development
- ☑ **Event-driven, non-blocking** architecture makes it super efficient
- ☑ **V8 engine** provides the power (like a sports car engine)
- ☑ **npm** is our package manager (like an online grocery store)
- ☑ **Simple servers** can be created with just a few lines of code
- ☑ **process, \_\_dirname, \_\_filename** help us navigate our application

### Key Analogies:

- 🍽️ **Node.js** = Smart restaurant waiter
- 🏎️ **V8 Engine** = Sports car engine
- 📦 **npm** = Online grocery store for code
- 🏠 **localhost** = Your house address
- 📍 **\_\_dirname/\_\_filename** = GPS coordinates

### Next Steps:

In the next chapter, we'll explore **Node.js Core Modules** - the built-in tools that make Node.js so powerful!

---

#### **Practice Exercise:**

1. Create a simple Node.js script that displays your name and current time
2. Create a basic server that shows different messages for different URLs
3. Experiment with `process`, `__dirname`, and `__filename` in your scripts

**Ready for Chapter 2? Let's dive into the amazing world of Node.js Core Modules! 🚀**