

Node.js & JavaScript Learning Notes (Beginner Friendly)

This file explains everything in the simplest possible way—so even if you're just starting out, you're not lost. Expect simple language, easy examples, and a little humor to keep you awake. 😊

Day 1: Getting Started with Node.js & JavaScript

What is Node.js?

Node.js is like a superhero that lets JavaScript work **outside the browser**. Normally, JavaScript needs a browser like Chrome. But Node.js lets you run `.js` files directly from your computer's terminal.

🐱 Try this:

```
node app.js
```

Is React also a runtime like Node.js?

Nope! React is just a **toolbox** to help you build user interfaces (like buttons, forms, etc.) inside the browser. It reacts (get it?) when your data changes, and updates the page automatically.

Setup Checklist

- Make a folder for your code.
- Open it in VS Code.
- Run `npm init -y` to create a starter file.
- Make a file like `hello.js` and write:

```
console.log("Hello Node.js");
```

Boom! You just ran backend code! 🧠

Day 2: JavaScript Functions & Git Basics

What is `(err) => {}`?

It's a **shortcut** for writing a function. You don't need the word `function`, you just write the arrow.

🐱 Example:

```
const sayHi = (name) => {  
  console.log(`Hello, ${name}`);  
}  
sayHi("You");
```

What does `path.join()` **do?**

It glues file paths together in a safe way, no matter what operating system you're using (Windows, Mac, Linux).

🐱 Example:

```
const path = require("path");  
const fullPath = path.join("folder", "file.txt");  
console.log(fullPath);
```

What is UTF-8?

It's just a way to **encode text** so computers can read/write files properly. Without it, reading files might look like alien code. 🙄

🐱 Example:

```
fs.readFile("notes.txt", "utf-8", (err, data) => {  
  console.log(data);  
});
```

Where does `data` **come from in functions like** `readFile()` **?**

You write the variable name in your function, and Node magically gives it to you. 😊

🐱 Example:

```
fs.readFile("file.txt", "utf-8", (err, data) => {  
  console.log(data); // Node gives you this 'data'  
});
```

What does `git branch -M main` **do?**

It **renames your main branch** to `main` (instead of the older default `master`). Keeps things modern.



Day 3: The Magic of the File System (`fs`)

What can you do with `fs`?

Think of `fs` like a librarian that lets you:

- **Write to files**

```
fs.writeFile("notes.txt", "Hello world!", (err) => {});
```

- **Add more text** (like writing in a diary)

```
fs.appendFile("notes.txt", "\nAnother line", (err) => {});
```

- **Read files** (duh)

```
fs.readFile("notes.txt", "utf-8", (err, data) => console.log(data));
```

- **Delete files** (carefully!)

```
fs.unlink("notes.txt", (err) => {});
```

- **Make a folder**

```
fs.mkdir("myFolder", (err) => {});
```

- **List what's inside a folder**

```
fs.readdir("./", (err, files) => console.log(files));
```

- **Rename or move files**

```
fs.rename("old.txt", "new.txt", (err) => {});
```

Can I use `import` instead of `require`?

Yes, if you either:

- Rename the file to `.mjs`, or
- Add `"type": "module"` in your `package.json`

🐱 Example:

```
import fs from "fs";
```

Mini Task

1. Make a folder called myFiles
2. Write to notes.txt
3. Read & print it
4. Add more lines
5. Rename the file
6. Delete it like a boss 😎

👉 Need the raw version? [Read the original notes here](#)

You've made it through 3 days—each day getting closer to becoming a real Node wizard . Onward to Day 4!