**✅ Step-by-Step: .prettierrc**

**🔹 Purpose:**

Prettier is used to format your code consistently. It's **opinionated**, fast, and works with many editors. It handles things like indentation, semicolons, quotes, etc.

**📂 File: .prettierrc**

{

"semi": true, // Add semicolons at the end of statements

"singleQuote": true, // Use single quotes instead of double

"printWidth": 100, // Wrap lines at 100 characters

"tabWidth": 2, // Set indentation to 2 spaces

"trailingComma": "all", // Add trailing commas wherever possible (ES5+)

"bracketSpacing": true, // Print spaces between brackets

"arrowParens": "always", // Always include parens in arrow functions

"endOfLine": "lf" // Use LF (Line Feed) for line endings

}

**✅ Best Practice Add-ons**

You can also add this in package.json for strict enforcement:

"prettier": "@your-org/prettier-config"

Or define it in a separate npm package for multi-repo usage.

**✨ Benefits:**

* Works with VSCode auto-save
* Used by .husky/pre-commit for safe commits
* Avoids formatting fights during code reviews

**✅ Step-by-Step: .eslintrc.js**

**🔹 Purpose:**

ESLint enforces code quality rules. For NestJS (TypeScript-based), ESLint replaces TSLint.

**📂 File: .eslintrc.js**

module.exports = {

root: true,

parser: '@typescript-eslint/parser', // Use TypeScript parser

parserOptions: {

project: ['./tsconfig.json'], // Point to TypeScript config

tsconfigRootDir: \_\_dirname,

sourceType: 'module',

},

plugins: ['@typescript-eslint', 'prettier'],

extends: [

'plugin:@typescript-eslint/recommended', // TypeScript rules

'plugin:prettier/recommended', // Integrate Prettier with ESLint

],

rules: {

// Customize rules here

'@typescript-eslint/no-explicit-any': 'warn',

'@typescript-eslint/explicit-module-boundary-types': 'off',

'@typescript-eslint/no-unused-vars': ['warn', { argsIgnorePattern: '^\_' }],

'prettier/prettier': [

'error',

{

singleQuote: true,

semi: true,

trailingComma: 'all',

printWidth: 100,

},

],

},

ignorePatterns: ['dist', 'node\_modules'],

};

**✅ What This Does:**

| **Feature** | **What it controls** |
| --- | --- |
| parser | Uses TypeScript ESLint parser |
| extends | Uses recommended rules from TypeScript and Prettier |
| rules | Customize specific rules to match NestJS and team style |
| ignorePatterns | Ignore generated files like dist/ or node\_modules |

**💡 Example in Action**

// Without Prettier + ESLint

function hello(name){console.log("Hello, "+name)}

// With it enforced

function hello(name: string): void {

console.log('Hello, ' + name);

}

**📌 How These Two Work Together**

* **Prettier** handles formatting.
* **ESLint** handles code quality and safety rules.
* They’re **integrated** via plugin:prettier/recommended in .eslintrc.js.

**🧪 Optional Add-on (production-ready)**

Add this lint script in package.json:

"scripts": {

"lint": "eslint --ext .ts src/",

"format": "prettier --write \"src/\*\*/\*.ts\""

}