



# What is Linting

Linting is the process of running a program that analyzes source code to flag programming errors, bugs, stylistic errors

The screenshot shows two terminal windows side-by-side. The left window displays ESLint output for a file named 'somefile.js' with a 'max-element' rule. It shows a syntax error at line 4, column 10, where a closing brace '{' is expected. The right window shows the same code after it has been prettified by Prettier. The red arrow points from the ESLint output on the left to the Prettier output on the right, illustrating how ESLint can identify errors that Prettier can then fix.

```
problems ✘ max-element ✘ somefile.js
1
2 let x =      1;
3 if (x) {
4 x = x + 1;
5 }
6 |
```

```
options ✘ max-element ✘ somefile.js
1
2 let x = 1
3 if (x) {
4   x = x + 1;
5 }
6 |
```

[Back to home](#)[Jump To ↗](#)[◀ Prev](#)[Next ➞](#)[Go to Top ↑](#)



1. Multiple developers in a team can have a similar coding pattern

ESLint, Prettier and Pre-commit hooks 1 of 9

3. Automatic Linting so developer doesn't have to manually lint their code

## Tools

1. ESLint - <https://eslint.org/>

2. Prettier - <https://prettier.io/>

3. Husky - <https://typicode.github.io/husky/>

## Two types of lints

### 1. Code analysis (Similar to TS checks)

Analyzes code to find and fix problems according to a set of rules. These problems can include syntax errors, stylistic issues, and potential bugs.

```
let x = 1;  
let y = 1;
```



▼ Can you guess some bad things about this code?



1. x is never used, and hence can be removed  
ESLint, Prettier and Pre-commit hooks 1 of 9 can be a const

## 2. Code formatting (Purely stylistic changes)

```
let x = 1;  
let y = 1  
let firstName = "harkirat";  
let lastName = 'singh';
```

▼ Can you guess some bad things about this code?

1. One line is missing semi colons
2. Spacing is wrong in second line
3. One string uses double quotes, other uses single quotes

Eslint lets you do both 1 and 2

Prettier is used for 2



## ESLint, Prettier and Pre-commit hooks 1 of 9

Ref - <https://eslint.org/>

The screenshot shows the ESLint homepage with a dark background. The main title "Find and fix problems in your JavaScript code" is prominently displayed in large white font. Below it, a subtitle explains ESLint's purpose: "ESLint statically analyzes your code to quickly find problems. It is built into most text editors and you can run ESLint as part of your continuous integration pipeline." A command-line interface button contains the text "npm init @eslint/config@latest". Two buttons at the bottom are "Get Started" and "Become a Sponsor". A sidebar on the right provides information about the latest version (v9.3.0), upcoming version (v9.4.0), and development status (HEAD). An advertisement for EthicalAds is visible in the top right corner.

# No user data  
ethicalads:  
topic: devs  
region: global  
type: image

Monetize your audience: Fund an OSS project or website with EthicalAds, a privacy-first ad network

Ads by EthicalAds

Latest Version

v9.3.0 on 17 May

Upcoming Version

v9.4.0 on 31 May

Development

HEAD on 29 May

Sponsored by:

AUTOMATTIC and more...

## Node.js express app



## ESLint, Prettier and Pre-commit hooks 1 of 9

```
const app = express();
let x = 1;
```

```
app.get("/", (req, res) => {
  res.json({
    message: "Hi there"
  });
});
```

- Add ESLint

```
npm init @eslint/config@latest
```



Make sure you press `space` to actually select the values. If you face issues, try using `yarn` instead of `npm` (`npm i -g yarn`)



ESLint, Prettier and Pre-commit hooks 1 of 9

- ✓ Which framework does your project use? · none
- ✓ Does your project use TypeScript? · typescript
- ✓ Where does your code run? · node

The config that you've selected requires the following dependencies:

- eslint@9.x, globals, @eslint/js, typescript-eslint
- ✓ Would you like to install them now? · No / Yes
- ✓ Which package manager do you want to use? · npm

☕ Installing...

- Explore `eslint.config.mjs`

```
import globals from "globals";
import pluginJs from "@eslint/js";
import tseslint from "typescript-eslint";

export default [
  {languageOptions: { globals: globals.node }},
  pluginJs.configs.recommended,
  ...tseslint.configs.recommended,
];
```

- You might have to install typescript

```
yarn add typescript
```



## ESLint, Prettier and Pre-commit hooks 1 of 9

J,

- Run `npm run lint`

```
@ -> 1-eslint-express npm run lint
> 1-eslint-express@1.0.0 lint
> eslint .

/Users/harkiratsingh/Projects/week-26-eslint-prettier/solution/1-eslint-express/src/index.ts
  5:5  error  'x' is never reassigned. Use 'const' instead  prefer-const
  5:5  error  'x' is assigned a value but never used          @typescript-eslint/no-unused-vars

* 2 problems (2 errors, 0 warnings)
  1 error and 0 warnings potentially fixable with the '--fix' option.
```

- Try removing the extra variable

```
import express from "express";

const app = express();

app.get("/", (req, res) => {
  res.json({
    message: "Hi there"
  });
});
```

- Now run `npm run lint`



ESLint, Prettier and Pre-commit hooks 1 of 9

run lint

```
> 1-eslint-express@1.0.0 lint
> eslint .
```

- Try breaking the indentation

```
import express from "express";
```

```
const app = express();
```

```
app.get("/", (req, res) => {
  res.json({
    message: "Hi there"
  });
});
```

- Notice eslint still **DOESNT** complain



## ESLint, Prettier and Pre-commit hooks 1 of 9

```
1  const app = express();
2
3  app.get("/", (req, res) => {
4    res.json({
5      message: "Hi there"
6    });
7
8  });
9
```

# Code formatting in ESLint

- Update eslint.config.mjs

```
import globals from "globals";
import pluginJs from "@eslint/js";
import tseslint from "typescript-eslint";
```

```
export default [
  {
    // ...
  }
]
```

```
// Indentation rule: 2 spaces
```



ESLint, Prettier and Pre-commit hooks 1 of 9

```
    "indent": ["error", 2],  
    "space-in-parens": ["error", "never"]  
  },  
  pluginJs.configs.recommended,  
  ...tseslint.configs.recommended,  
];
```

```
9 → 1-eslint-express npm run lint  
> 1-eslint-express@1.0.0 lint  
> eslint .  
  
/Users/harkiratsingh/Projects/week-26-eslint-prettier/solution/1-eslint-express/src/index.ts  
  6:1  error  Expected indentation of 2 spaces but found 5 tabs  indent  
  8:1  error  Expected indentation of 2 spaces but found 1 tab  indent  
  
✖ 2 problems (2 errors, 0 warnings)  
  2 errors and 0 warnings potentially fixable with the `--fix` option.
```

## Problem

If you press `tab` while coding, it adds a tab. You need `2 spaces` to be added now that our eslint config says that

ESLint, Prettier and Pre-commit hooks 1 of 9

```
4
5  app.get("/",(req,res) => {
6    |   res.json({
7    |     message: "Hi there"
8    });
9  });
10
11
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

zsh - 1-eslint-express

Ln 6, Col 5 Spaces: 2 UTF-8 LF {} TypeScript

# VSCode plugin

# Problem

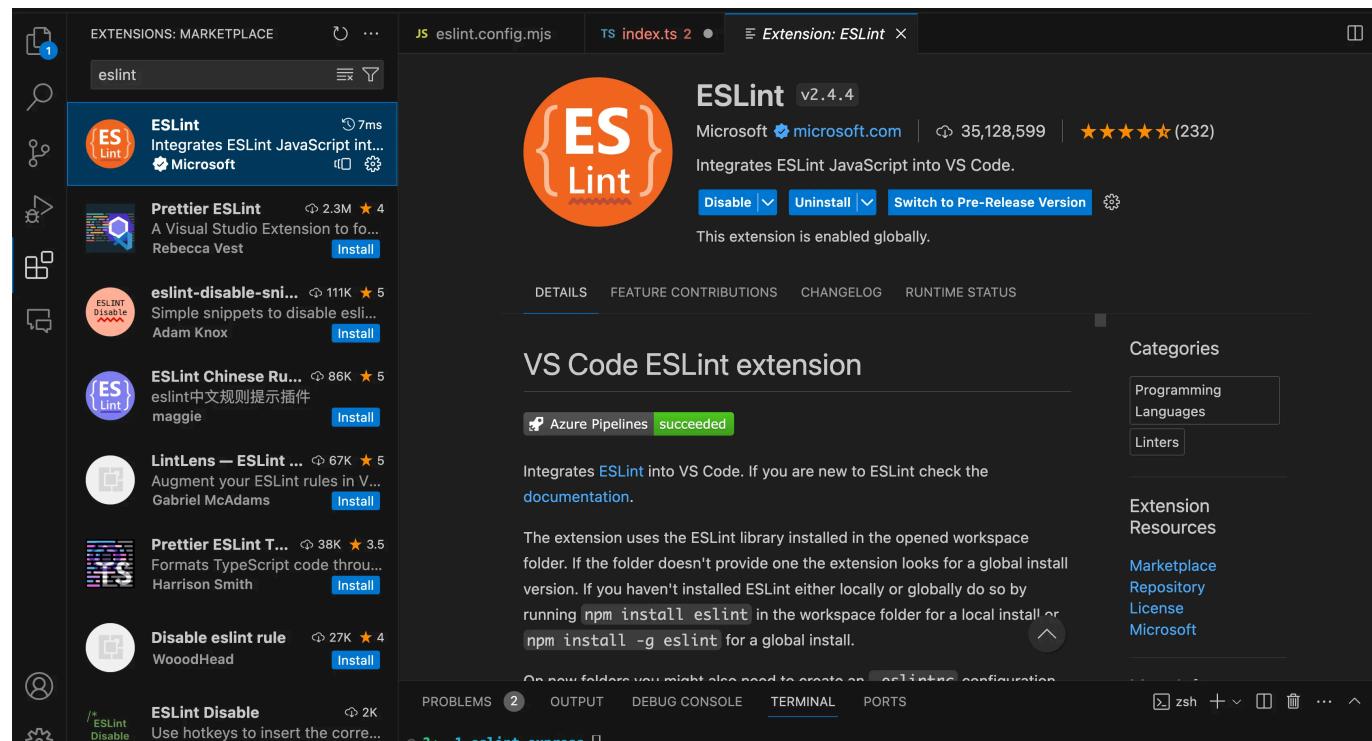


ESLint, Prettier and Pre-commit hooks 1 of 9

our editor directly. We have to run `npm run lint` to be able to see them.

# Solution

- Install the `ESLint` VSCode plugin





## ESLint, Prettier and Pre-commit hooks 1 of 9

```
SOURCE.MAINTAINENCE.COMMIT . EXPERT  
},  
"eslint.validate": ["javascript", "typescript"]  
}
```

- Notice code gets **auto linted** every time you save now

The screenshot shows a Mac OS X desktop environment with the Dock at the bottom. A Microsoft Visual Studio Code window is open, displaying a file named 'index.ts'. The code contains a simple Express.js application. In the editor, several lines of code are underlined in red, indicating ESLint errors. The 'PROBLEMS' tab in the bottom navigation bar shows a count of 1 error. The 'OUTLINE' and 'TIMELINE' tabs are also visible in the bottom left. The status bar at the bottom right shows the current line (Ln 8), column (Col 9), and file type (TypeScript).

```
src > ts index.ts > app.get("/") callback  
1 import express from "express";  
2  
3 const app = express();  
4  
5 app.get("/", (req, res) => {  
6   res.json({  
7     message: "Hi there"  
8   });  
9 });
```



# ESLint in react project

- Initialize a react project

```
npm create vite@latest
```



- vite by default gives you eslint
- Run lint

```
yarn install  
yarn lint
```





# Popular eslint configurations

You can roll your own config, like we did in the last slide

```
module.exports = {
  root: true,
  env: { browser: true, es2020: true },
  extends: [
    'eslint:recommended',
    'plugin:@typescript-eslint/recommended',
    'plugin:react-hooks/recommended',
  ],
  ignorePatterns: ['dist', '.eslintrc.cjs'],
  parser: '@typescript-eslint/parser',
  plugins: ['react-refresh'],
  rules: {
    // ...
    'allowConstantExport': true,
  }
}
```

],



## ESLint, Prettier and Pre-commit hooks 1 of 9

Or you can use a popular one created by other people, for example

1. <https://github.com/airbnb/javascript>
1. <https://github.com/dustinspecker/awesome-eslint#configs>



## ESLint, Prettier and Pre-commit hooks 1 of 9

Using multiple properties of an object. eslint: [prefer-destructuring](#)

Why? Destructuring saves you from creating temporary references for those properties, and from repetitive access of the object. Repeating object access creates more repetitive code, requires more reading, and creates more opportunities for mistakes. Destructuring objects also provides a single site of definition of the object structure that is used in the block, rather than requiring reading the entire block to determine what is used.

```
// bad
function getFullName(user) {
  const firstName = user.firstName;
  const lastName = user.lastName;

  return `${firstName} ${lastName}`;
}

// good
function getFullName(user) {
  const { firstName, lastName } = user;
  return `${firstName} ${lastName}`;
}

// best
function getFullName({ firstName, lastName }) {
  return `${firstName} ${lastName}`;
}
```

## Using airbnb config

- Add dependencies



## ESLint, Prettier and Pre-commit hooks 1 of 9



- Update .eslintrc.cjs

```
module.exports = {  
  root: true,  
  env: { browser: true, es2020: true },  
  parserOptions: {  
    ecmaVersion: "latest",  
    sourceType: "module",  
    project: ["./tsconfig.json", "./tsconfig.node.json"],  
    tsconfigRootDir: __dirname,  
  },  
  extends: [  
    "eslint:recommended",  
    "plugin:@typescript-eslint/recommended",  
    "plugin:react-hooks/recommended",  
    "airbnb",  
    "airbnb/hooks",  
  ],  
  ignorePatterns: ["dist", ".eslintrc.cjs", "vite.config.ts"],  
  parser: "@typescript-eslint/parser",  
  plugins: ["react-refresh"],  
  rules: {  
    "react/jsx-filename-extension": [  
      ,  
    ],  
  },  
};
```

"react-refresh/only-export-components": [



## ESLint, Prettier and Pre-commit hooks 1 of 9

```
  ],
},
};
```

 If you try doing the same in the new config system (for eg, the node.js project), you'll face some issues.

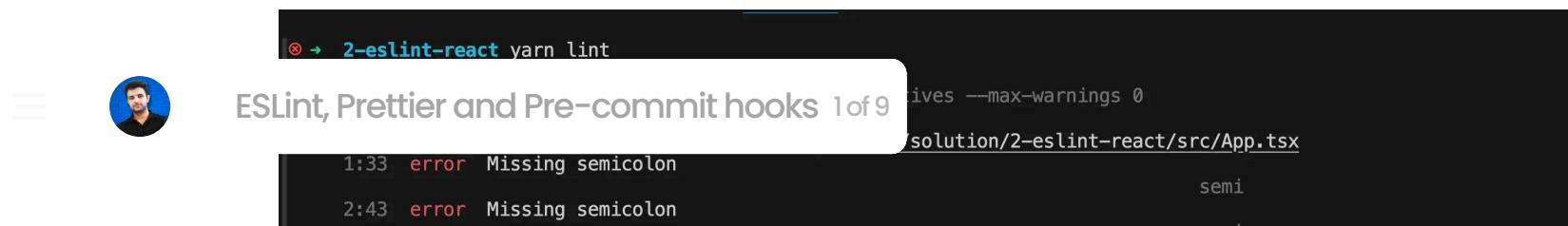
New config system - ▶

Ref - <https://stackoverflow.com/questions/74925642/how-to-use-eslint-config-airbnb-with-the-new-eslint-config-spec>

- Update .eslintrc.cjs (read README.md)

```
parserOptions: {
  ecmaVersion: 'latest',
  sourceType: 'module',
  project: ['./tsconfig.json', './tsconfig.node.json'],
  tsconfigRootDir: '__dirname',
},
```

- Try running the lint command



The terminal window shows the command `2-eslint-react yarn lint` being run. The output indicates 9 errors found, with two specific ones highlighted:

```
✖ 2-eslint-react yarn lint
ESLint, Prettier and Pre-commit hooks 1 of 9
  1:33  error  Missing semicolon
  2:43  error  Missing semicolon
    1 error(s) found in 1 file(s).
    1 warning(s) found in 1 file(s).
```

The errors are located in the file `/solution/2-eslint-react/src/App.tsx`:

```
1:33  error  Missing semicolon
2:43  error  Missing semicolon
```

# ESLint with nextjs

- Create a next application (Select yes on ESLint)

```
npx create-next-app
```



- Check the default config

```
{
  "extends": ["next/core-web-vitals"]
}
```



- Add `.vscode/settings.json`

```
{
  "editor.formatOnSave": true,
  "editor.codeActionsOnSave": [
    "source.organizeImports"
  ]
}
```



}



## ESLint, Prettier and Pre-commit hooks 1 of 9

- Add airbnb config

```
yarn add eslint-config-airbnb eslint-config-airbnb-typescript @typescript-eslint/eslint-plugin
```



- Update the eslint config

```
{  
  "extends": ["next/core-web-vitals", "airbnb", "airbnb/hooks"]  
}
```



- Run `npm run lint`



## ESLint, Prettier and Pre-commit hooks 1 of 9

```
./app/layout.tsx
1:31 Error: Strings must use singlequote. @typescript-eslint/quotes
2:23 Error: Strings must use singlequote. @typescript-eslint/quotes
3:8 Error: Strings must use singlequote. @typescript-eslint/quotes
5:33 Error: Strings must use singlequote. @typescript-eslint/quotes
8:10 Error: Strings must use singlequote. @typescript-eslint/quotes
9:16 Error: Strings must use singlequote. @typescript-eslint/quotes
18:5 Error: 'React' must be in scope when using JSX react/react-in-jsx-scope
19:7 Error: 'React' must be in scope when using JSX react/react-in-jsx-scope

./app/page.tsx
1:19 Error: Strings must use singlequote. @typescript-eslint/quotes
5:5 Error: 'React' must be in scope when using JSX react/react-in-jsx-scope
6:7 Error: 'React' must be in scope when using JSX react/react-in-jsx-scope
7:9 Error: 'React' must be in scope when using JSX react/react-in-jsx-scope
9:11 Error: 'React' must be in scope when using JSX react/react-in-jsx-scope
11:9 Error: 'React' must be in scope when using JSX react/react-in-jsx-scope
12:11 Error: 'React' must be in scope when using JSX react/react-in-jsx-scope
18:15 Error: `{" "}` must be placed on a new line react/jsx-one-expression-per-line
18:16 Error: Strings must use singlequote. @typescript-eslint/quotes
19:13 Error: 'React' must be in scope when using JSX react/react-in-jsx-scope
31:7 Error: 'React' must be in scope when using JSX react/react-in-jsx-scope
```

- Fix some automatically

yarn lint --fix



# Drafts



## ESLint, Prettier and Pre-commit hooks 1 of 9

Ref - <https://prettier.io/>

The screenshot shows the official Prettier website. At the top, there's a navigation bar with links for "Playground", "Docs", "Blog", "Search", "Donate", and "GitHub". Below the navigation is a large, abstract graphic composed of horizontal bars in various colors (blue, green, yellow, red) arranged in a grid-like pattern. Two prominent buttons are centered below the graphic: a yellow "TRY IT ONLINE" button and a purple "INSTALL PRETTIER" button. The main content area has a dark background with white text. On the left, under the heading "# What is Prettier?", there's a bulleted list: "\* An opinionated code formatter", "\* Supports many languages", "\* Integrates with most editors", and "\* Has few options »". On the right, under the heading "# Why?", there's another bulleted list: "\* Your code is formatted on save", "\* No need to discuss style in code review", "\* Saves you time and energy", and "\* And more »". At the bottom of the main content area, the text "Works with the Tools You Use" is visible.

Installing prettier - <https://prettier.io/docs/en/install>

- Install dependency

- Create a `.prettierrc` file in the root



## ESLint, Prettier and Pre-commit hooks 1 of 9

- Create `.prettierignore`

build

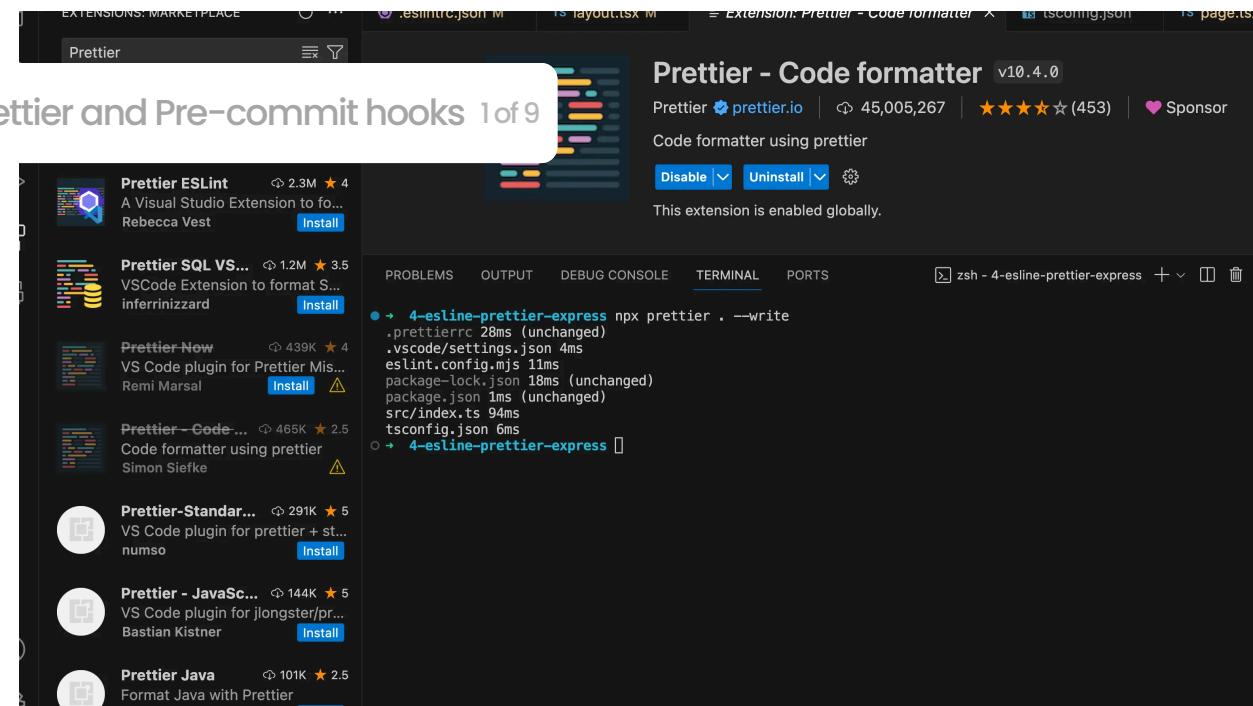
coverage

dist

- Try formatting

```
npx prettier . --write
```

- Add vscode extension



- Update `.vscode/settings.json`

```
{
  "editor.codeActionsOnSave": {
    "source.fixAll.eslint": "explicit"
  },
  "eslint.validate": ["javascript", "typescript"],
  "editor.defaultFormatter": "esbenp.prettier-vscode",
  "[javascript)": {
    "editor.defaultFormatter": "esbenp.prettier-vscode",
  }
}
```

```
"[typescript)": {
```

```
"editor.defaultFormatter": "esbenp.prettier-vscode",
```



## ESLint, Prettier and Pre-commit hooks 1 of 9

```
}
```

- Install `eslint-config-prettier` (this is to ensure they don't conflict with each other's configs)

```
npm install --save-dev eslint-config-prettier
```



- Extend eslint config to use `prettier`

```
"extends": [  
  "next/core-web-vitals",  
  "airbnb",  
  "airbnb-typescript",  
  "airbnb/hooks",  
  "prettier"  
,
```



- Add `prettier` script

```
"prettier": "prettier . --write"
```



- Try running `npm run prettier`



ESLint, Prettier and Pre-commit hooks 1 of 9

# PRE COMMIT HOOKS

Ref - <https://typicode.github.io/husky/>

The screenshot shows the Husky GitHub page. At the top, it displays 'downloads 48M/month' and 'v9.0.1'. The main content area includes a brief introduction, download statistics, and a summary of Husky's capabilities. It highlights that Husky enhances commits and automatically lint commit messages, code, and run tests upon committing or pushing. A 'Features' section lists various capabilities like support for macOS, Linux, Windows, and Git GUIs. Below this, there's a 'And more:' section with additional bullet points. To the right, there's a sidebar titled 'On this page' with links to Features, Sponsors, Used by, and Articles. At the bottom right, there's an advertisement for Figma.

**On this page**

- Features
- Sponsors
- Used by
- Articles

**Features**

- Just 2 kB (gzipped) with no dependencies
- Extremely fast (runs in ~1ms)
- Uses new Git feature (core.hooksPath)
- Supports:
  - macOS, Linux, Windows
  - Git GUIs, Node version managers, custom hooks directory, nested projects, monorepos
  - All 13 client-side Git hooks

And more:

- Branch-specific hooks
- Use POSIX shell to script advanced cases
- Adheres to Git's native hook organization

**Explore design possibilities with Figma. Build prototypes, and easily translate your work into code.**

ADS VIA CARBON

## How to start

The slide has a header with a profile picture, the title 'ESLint, Prettier and Pre-commit hooks 1 of 9', and a subtitle 'Initialize Husky'. It includes icons for copy, download, and refresh.

Terminal command: `npx husky init`

- Update .husky/pre-commit

Terminal command: `npm prettier`

- Try to commit

git init  
git add .  
git commit -m "Init"

Screenshot of a VS Code terminal window showing the execution of the commands:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS zsh + ⌂ ⌂ ... ^

star Done in 11.35s.
● 7-eslint-prettier-husky git:(main) ✘ git commit -m "init"
> 3-next-eslint@0.1.0 prettier
> prettier . --write
.eslintrc.json 23ms (unchanged)
.prettierrc 11ms
.vscode/settings.json 2ms
app/globals.css 17ms (unchanged)
app/layout.tsx 97ms
app/page.tsx 9ms
next.config.mjs 4ms (unchanged)
package-lock.json 42ms (unchanged)
[main 2af8709] init
```

The terminal shows the execution of the commands and the output of the ESLint, Prettier, and Husky initialization process. The status bar at the bottom indicates 'esconfig.json 1ms (unchanged)' and '[main 2af8709] init'.

• You can get past it by using `--no-verify`



## ESLint, Prettier and Pre-commit hooks 1 of 9

git commit --no-verify

