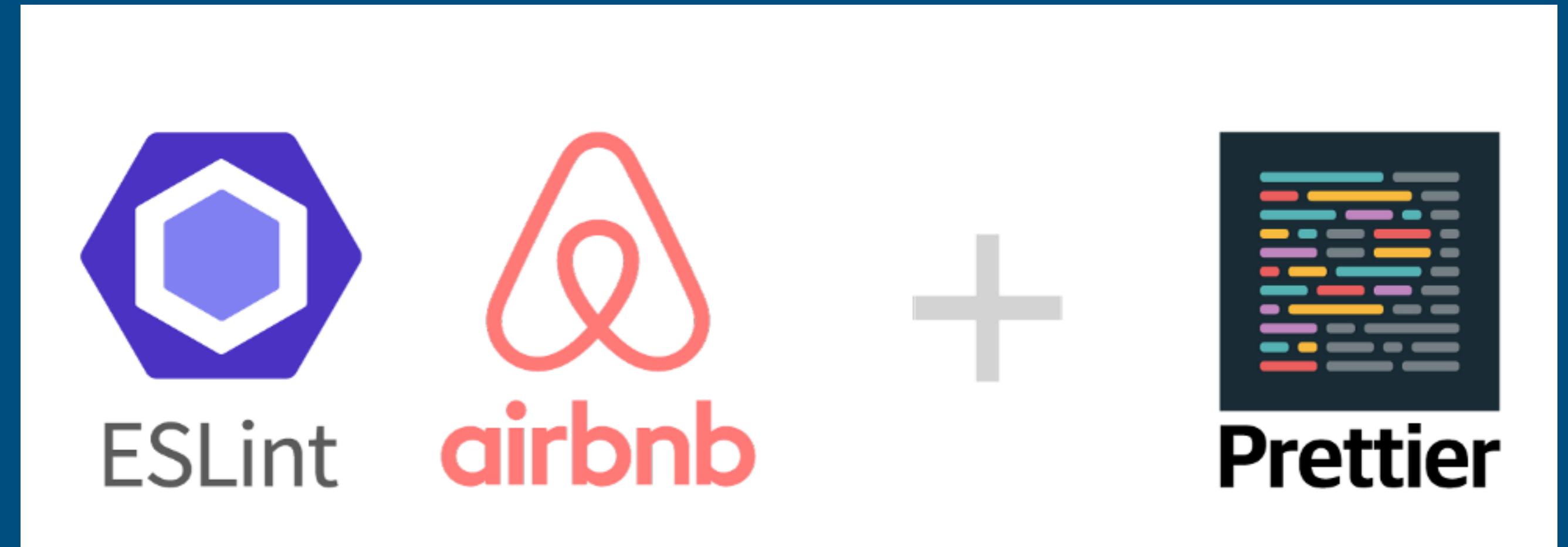


Code Quality Tools



Full Stack Web Development

Hapi Application Code Quality Tools

Hapi Application

Simplest Node Application

package.json

```
{  
  "name": "playtime",  
  "version": "0.1.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \\\"Error: no test specified\\\" && exit 1"  
  },  
  "author": "",  
  "license": "ISC"  
}
```

The screenshot shows a software interface for developing a Node.js application named 'playtime'. The top part of the interface is a code editor displaying the contents of the `package.json` file. The file defines the project name as 'playtime', version as '0.1.0', and main entry point as 'index.js'. It includes a script named 'test' that outputs an error message and exits with status 1. The bottom part of the interface is a terminal window showing the output of running the 'npm start' command, which outputs a message about the default interactive shell being zsh and provides instructions for switching to zsh if desired.

```
playtime – package.json  
playtime ~repos/modules/hdip/2021/prj/ent-wkship  
playtime package.json  
1 {  
2   "name": "playtime",  
3   "version": "0.1.0",  
4   "description": "",  
5   "main": "index.js",  
6   "scripts": {  
7     "test": "echo \\\"Error: no test specified\\\" && exit 1"  
8   },  
9   "author": "",  
10  "license": "ISC"  
11 }  
12  
Terminal: Local +  
The default interactive shell is now zsh.  
To update your account to use zsh, please run `chsh -s /bin/zsh`.  
For more details, please visit https://support.apple.com/kb/HT208050.  
Eamonns-Mac-mini:playtime edeleastar$
```

Minimal Hapi Application

src/server.js

```
import Hapi from "@hapi/hapi";
import path from "path";

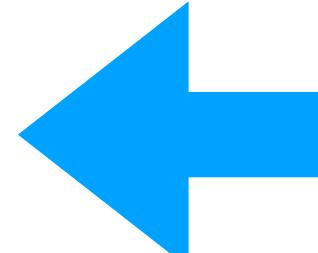
import { fileURLToPath } from "url";

const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);

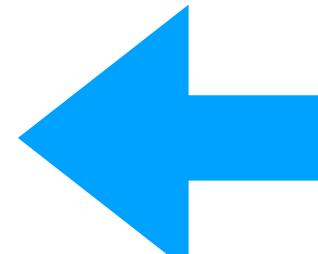
async function init() {
  const server = Hapi.server({
    port: 3000,
    host: "localhost",
  });
  await server.start();
  console.log("Server running on %s", server.info.uri);
}

process.on("unhandledRejection", (err) => {
  console.log(err);
  process.exit(1);
});

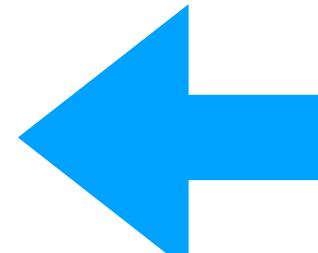
init();
```



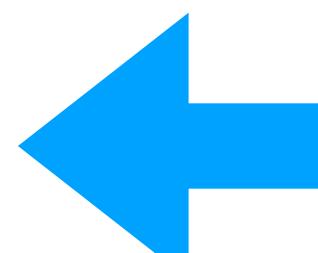
Import Hapi



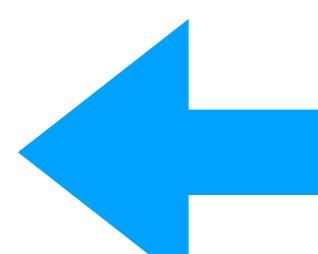
Import utilities to locate folder paths



Initialise a server



Handle failure to start



Start the app

Application Scripts

package.json

```
{  
  "name": "playtime",  
  "version": "0.1.0",  
  "description": "Playtime: a Hapi/node application for managing Play]  
  "main": "src/server.js",  
  "type": "module",  
  "scripts": {  
    "start": "node src/server.js",  
    "lint": "./node_modules/.bin/eslint . --ext .js"  
  },
```

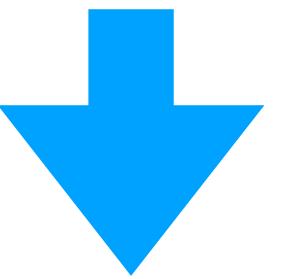
2 scripts:
“start”
“lint”

```
npm run start
```

- Can also do ‘npm run lint’...

Running the application

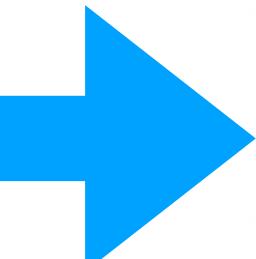
- Start the application ...

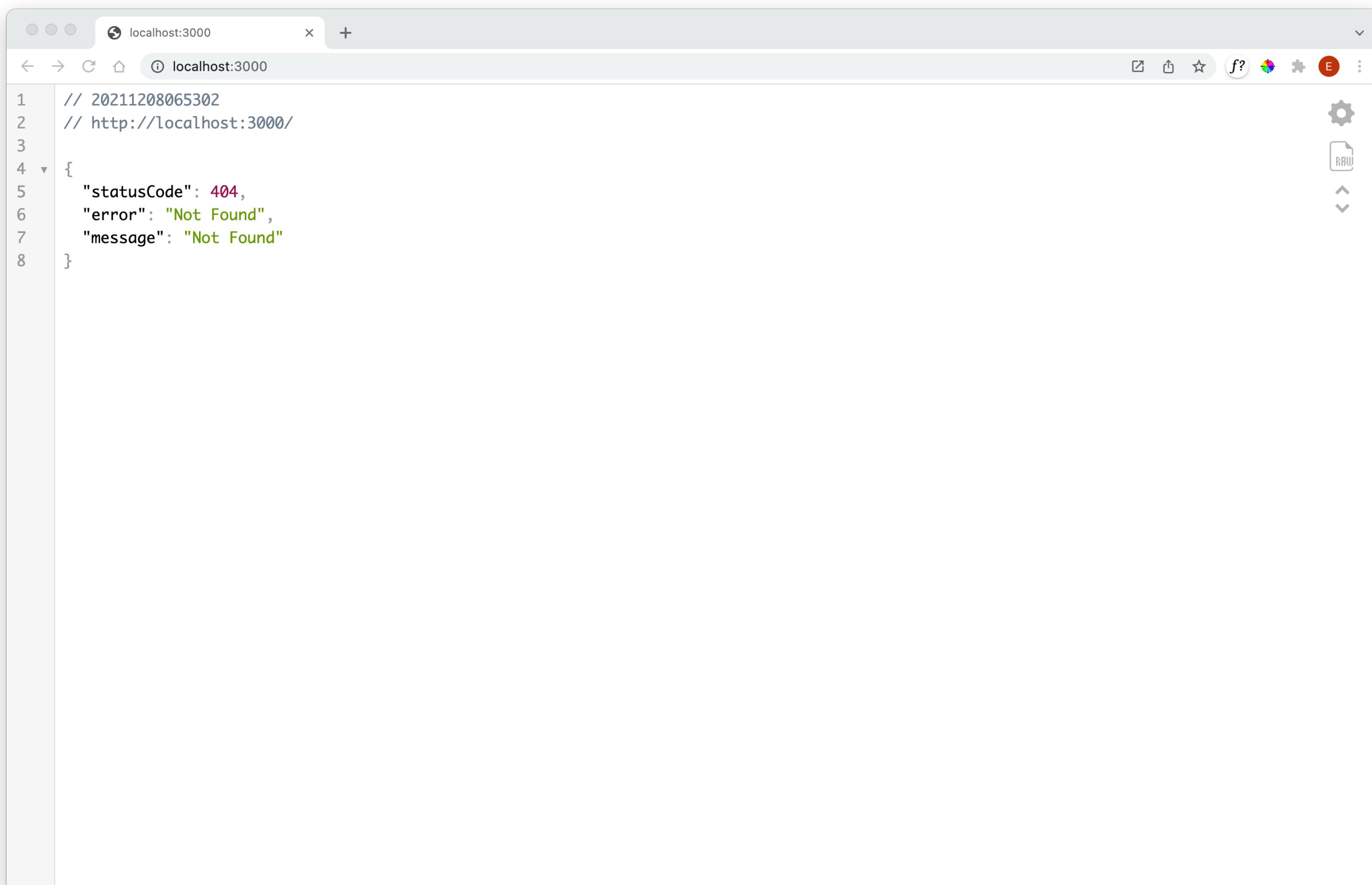


```
npm run start
```

```
> playtime@0.1.0 start
> node src/server.js
```

```
Server running on http://localhost:3000
```

- ...no routes defined yet 



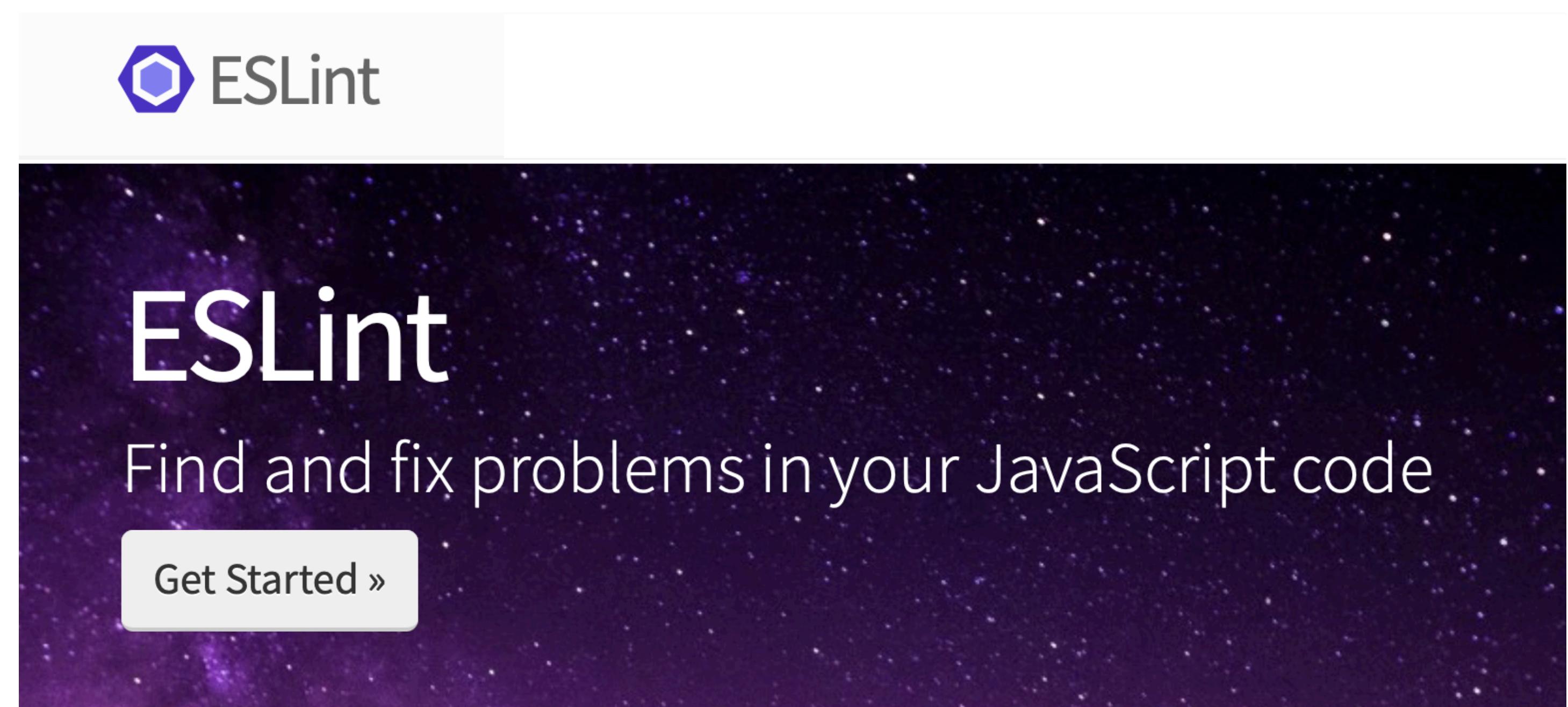
A screenshot of a web browser window titled "localhost:3000". The address bar also shows "localhost:3000". The page content is a JSON object with the following structure:

```
// 20211208065302
// http://localhost:3000/
{
  "statusCode": 404,
  "error": "Not Found",
  "message": "Not Found"
}
```

Code Quality Tools

Static Analysis

- Static program analysis is the analysis of computer software performed without executing any programs
- The term is usually applied to analysis performed by an automated tool.
- The Static Analysis may enforce a set of configurable Code Guidelines, which encapsulate a set of best practice rules



Find Problems

ESLint statically analyzes your code to quickly find problems. ESLint is built into most text editors and you can run ESLint as part of your continuous integration pipeline.

Fix Automatically

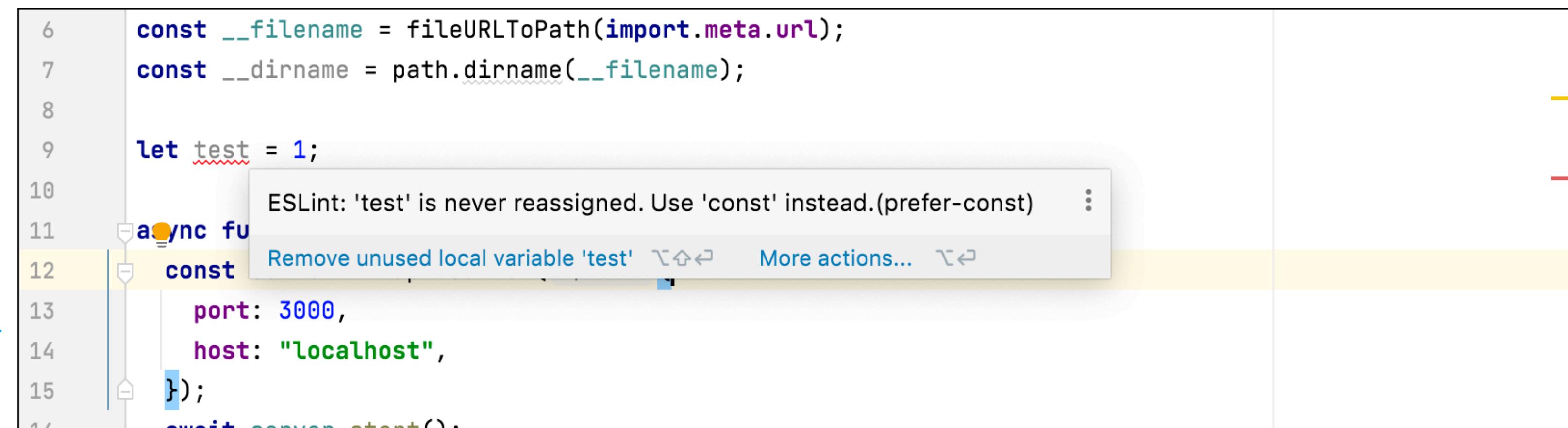
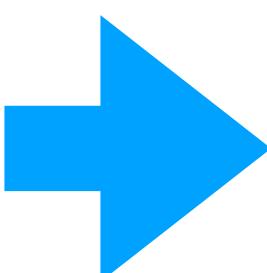
Many problems ESLint finds can be automatically fixed. ESLint fixes are syntax-aware so you won't experience errors introduced by traditional find-and-replace algorithms.

Customize

Preprocess code, use custom parsers, and write your own rules that work alongside ESLint's built-in rules. You can customize ESLint to work exactly the way you need it for your project.

AirBnB Code Style

- AirBnb defined a set of guidelines for use in their applications
- The rationale, documentation and ‘reasonable’ nature of the guidelines has made it a popular standard to aim for.
- Combine ESLint, Airbnb rules + IDE plugins



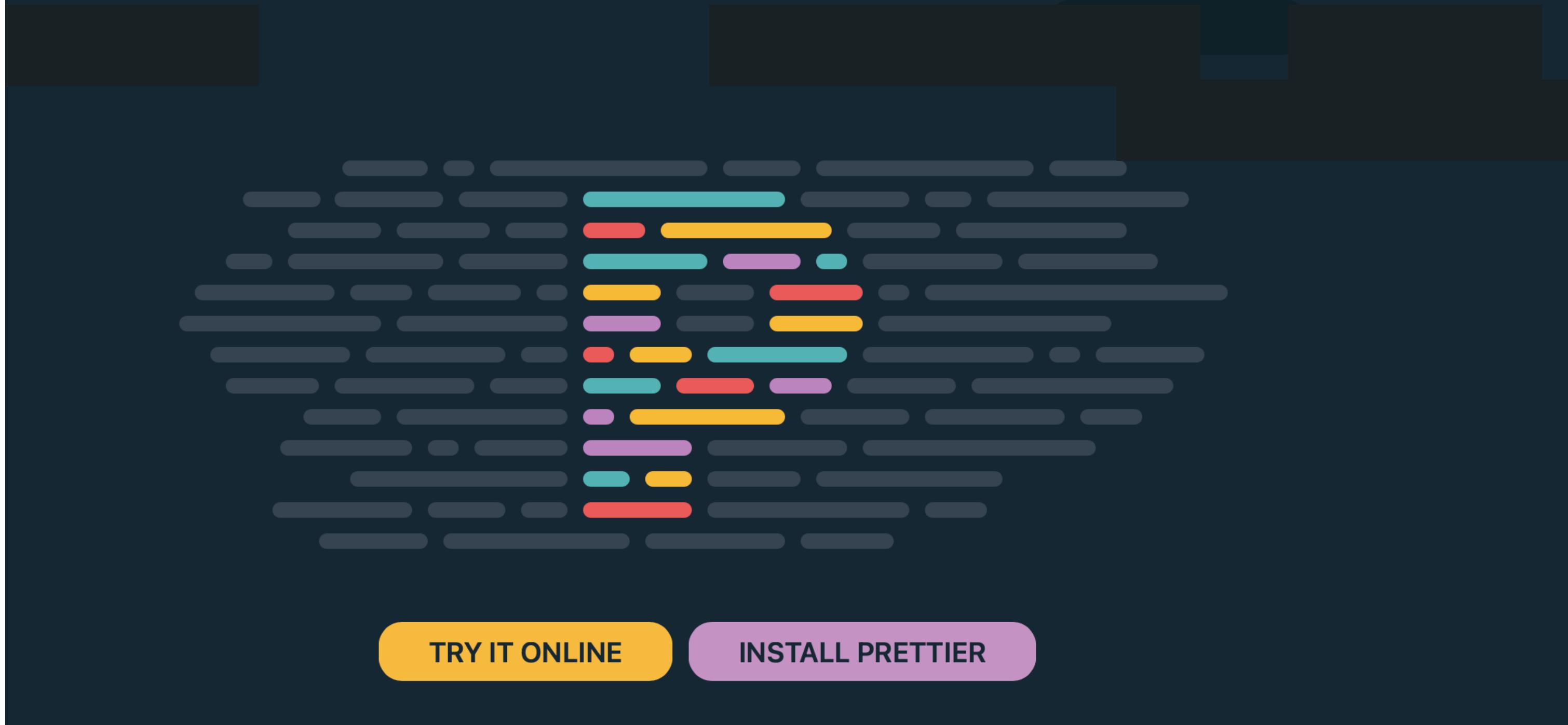
A screenshot of a code editor showing a file with ESLint annotations. The code is as follows:

```
6  const __filename = fileURLToPath(import.meta.url);
7  const __dirname = path.dirname(__filename);
8
9  let test = 1;
10
11  async function() {
12    const port: 3000,
13      host: "localhost",
14    );
15
16    await server.start();
17  }
18
```

The line `let test = 1;` has a red squiggle under `test`. A tooltip appears: "ESLint: 'test' is never reassigned. Use 'const' instead.(prefer-const)". Below the code editor, there is a toolbar with buttons for "Remove unused local variable 'test'" and "More actions...".

Code Formatting

- Prettier a prominent formatter in the Javascript Ecosystem
- “Opinionated” => formatter makes majority of key decisions.
- Some customisations - but far less than typical formatters



The background features a dark blue header with a decorative pattern of horizontal bars in various colors (teal, red, yellow, purple) on the right side. Below the header is a dark grey section containing two buttons: a yellow one labeled "TRY IT ONLINE" and a purple one labeled "INSTALL PRETTIER".

What is Prettier?

- * An opinionated code formatter
- * Supports many languages
- * Integrates with most editors
- * Has few options

Why?

- * You press save and code is formatted
- * No need to discuss style in code review
- * Saves you time and energy
- * And more

Configure Code Quality Tools

.prettierrc.json

```
{  
  "trailingComma": "es5",  
  "tabWidth": 2,  
  "semi": true,  
  "singleQuote": false,  
  "printWidth": 180  
}
```

.eslintrc.json

```
{  
  "env": {  
    "node": true,  
    "es2021": true  
  },  
  "extends": [  
    "airbnb-base",  
    "prettier"  
  ],  
  "parserOptions": {  
    "ecmaVersion": "latest",  
    "sourceType": "module"  
  },  
  "rules": {  
    "quotes": [  
      "error",  
      "double"  
    ],  
    "import/extensions": "off",  
    "import/prefer-default-export": "off",  
    "object-shorthand": "off",  
    "no-unused-vars": "off",  
    "no-underscore-dangle": "off",  
    "no-param-reassign": "off",  
    "no-undef": "off",  
    "func-names": "off",  
    "no-console": "off"  
  }  
}
```

Code Quality Tools

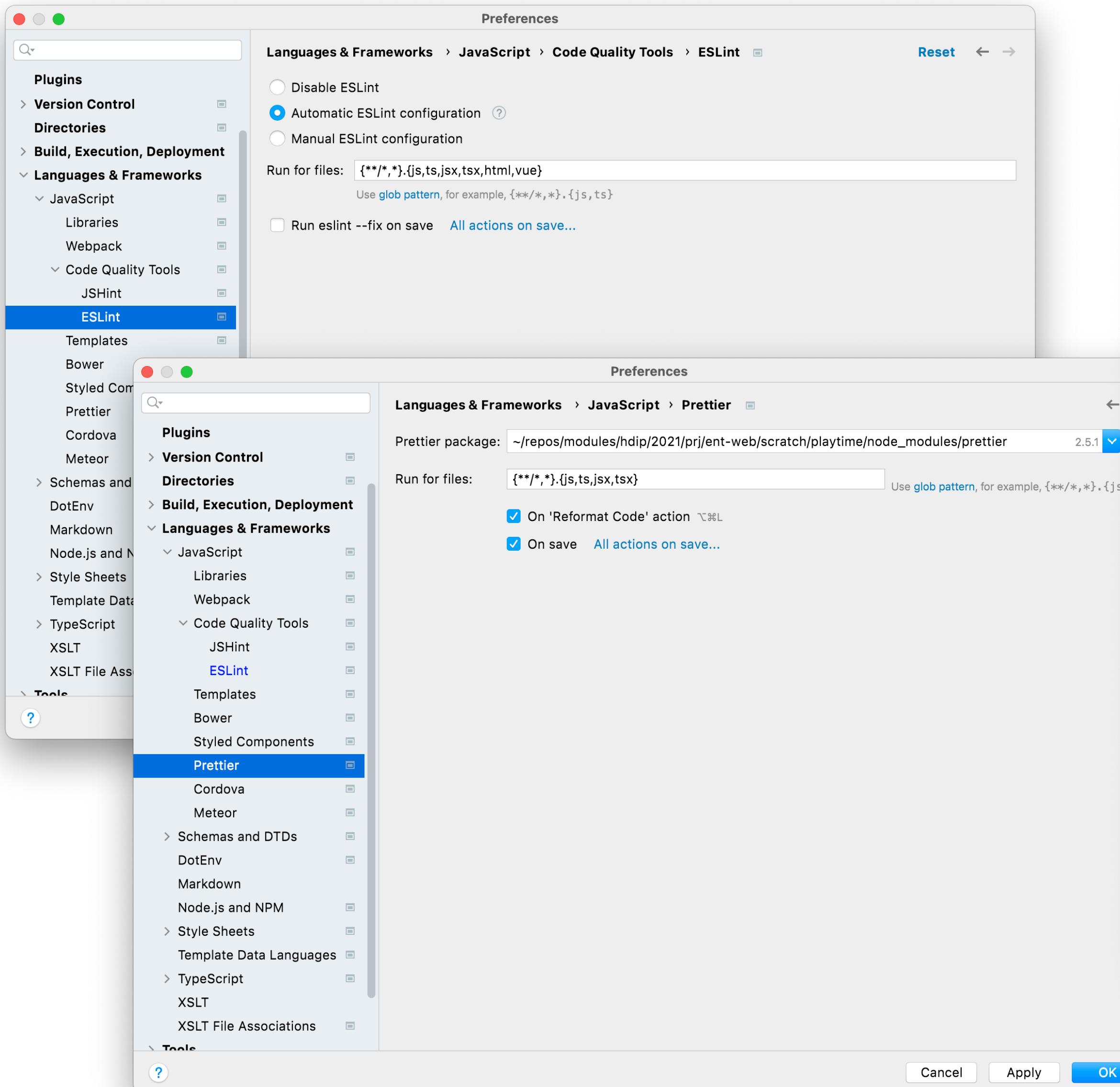
```
npm install -D eslint
npm install -D eslint-config-airbnb-base
npm install -D eslint-config-prettier
npm install -D eslint-plugin-import
npm install -D prettier
```

- ESLint
- Prettier
- Airbnb Code Style

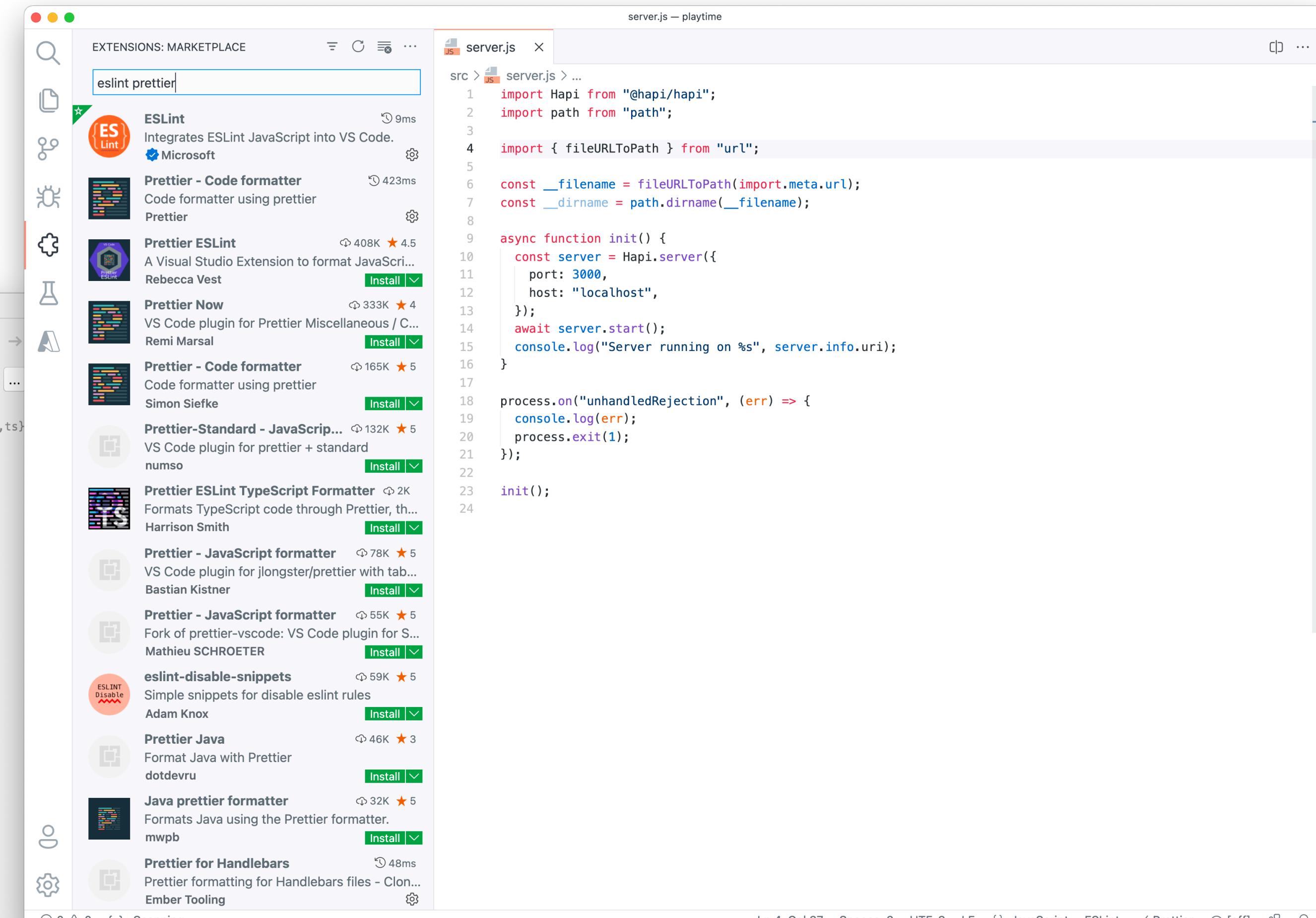
package.json

```
{
  "name": "playtime",
  "version": "0.1.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "eslint": "^8.4.1",
    "eslint-config-airbnb-base": "^15.0.0",
    "eslint-config-prettier": "^8.3.0",
    "eslint-plugin-import": "^2.25.3",
    "prettier": "^2.5.1"
  }
}
```

Code Quality IDE Configurations



WebStorm



VSCode

```
"rules": {  
  "import/extensions": "off",  
  "import/prefer-default-export": "off",  
  "object-shorthand": "off",  
  "no-unused-vars": "off",  
  "no-underscore-dangle": "off",  
  "no-param-reassign": "off",  
  "no-undef": "off",  
  "func-names": "off",  
  "no-console": "off"  
}
```



```
npm run lint
```

```
> playtime@0.1.0 lint
> ./node_modules/.bin/eslint . --ext .js

/Users/edeleastar/repos/modules/hdip/2021/prj/ent-web/scratch/playtime/src/server.js
  9:5  error  'test' is never reassigned. Use 'const' instead  prefer-const

✖ 1 problem (1 error, 0 warnings)
  1 error and 0 warnings potentially fixable with the `--fix` option.
```

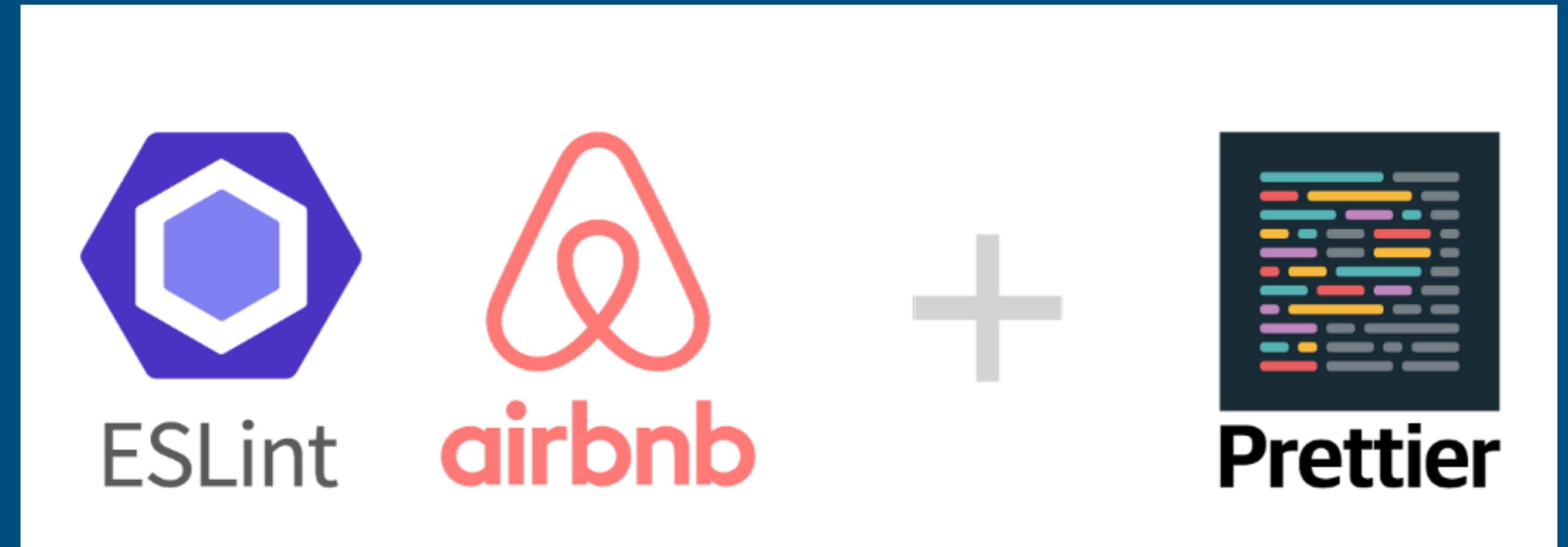


The screenshot shows a code editor window with a file named `server.js`. The file content is as follows:

```
src > JS server.js > ...
1 import Hapi from "@hapi/hapi";
2 import path from "path";
3
4 import 'test' is declared but its value is never read. ts(6133)
5 'test' is never reassigned. Use 'const' instead. eslint(prefer-const)
6 const test: number
7 const
8 View Problem Quick Fix... (⌘.)
9 let test = 1;
10
11 source function init() {
```

An ESLint error message is displayed in a tooltip over the line `import 'test'`:
`'test' is declared but its value is never read. ts(6133)`
`'test' is never reassigned. Use 'const' instead. eslint(prefer-const)`

Code Quality Tools



Full Stack Web Development