Code Quality Tools

Full Stack Web Development

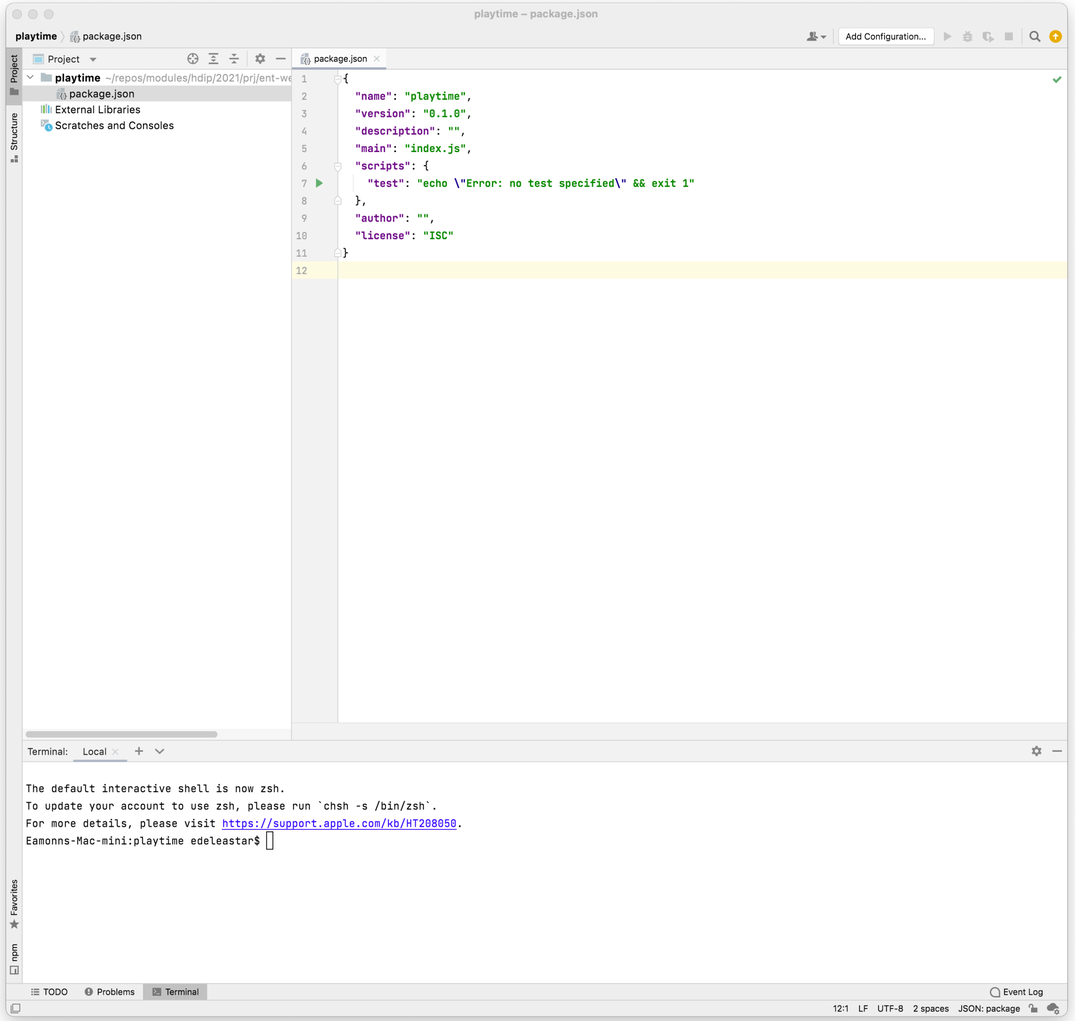
Hapi Application
Code Quality Tools

Hapi Application

Simplest Node Application

package.json

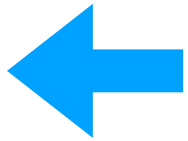
```
{
  "name": "playtime",
  "version": "0.1.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```



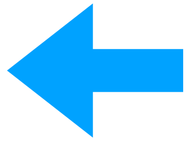
Minimal Hapi Application

src/server.js

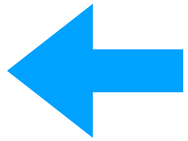
```
import Hapi from "@hapi/hapi";
import path from "path";

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();
```



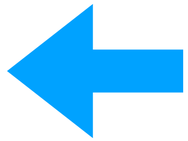
Import Hapi



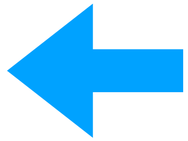
Import utilities to locate folder paths



Initialise a server



Handle failure to start



Start the app

Application Scripts

2 scripts:
“start”
“lint”

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"
  },
```

```
npm run start
```

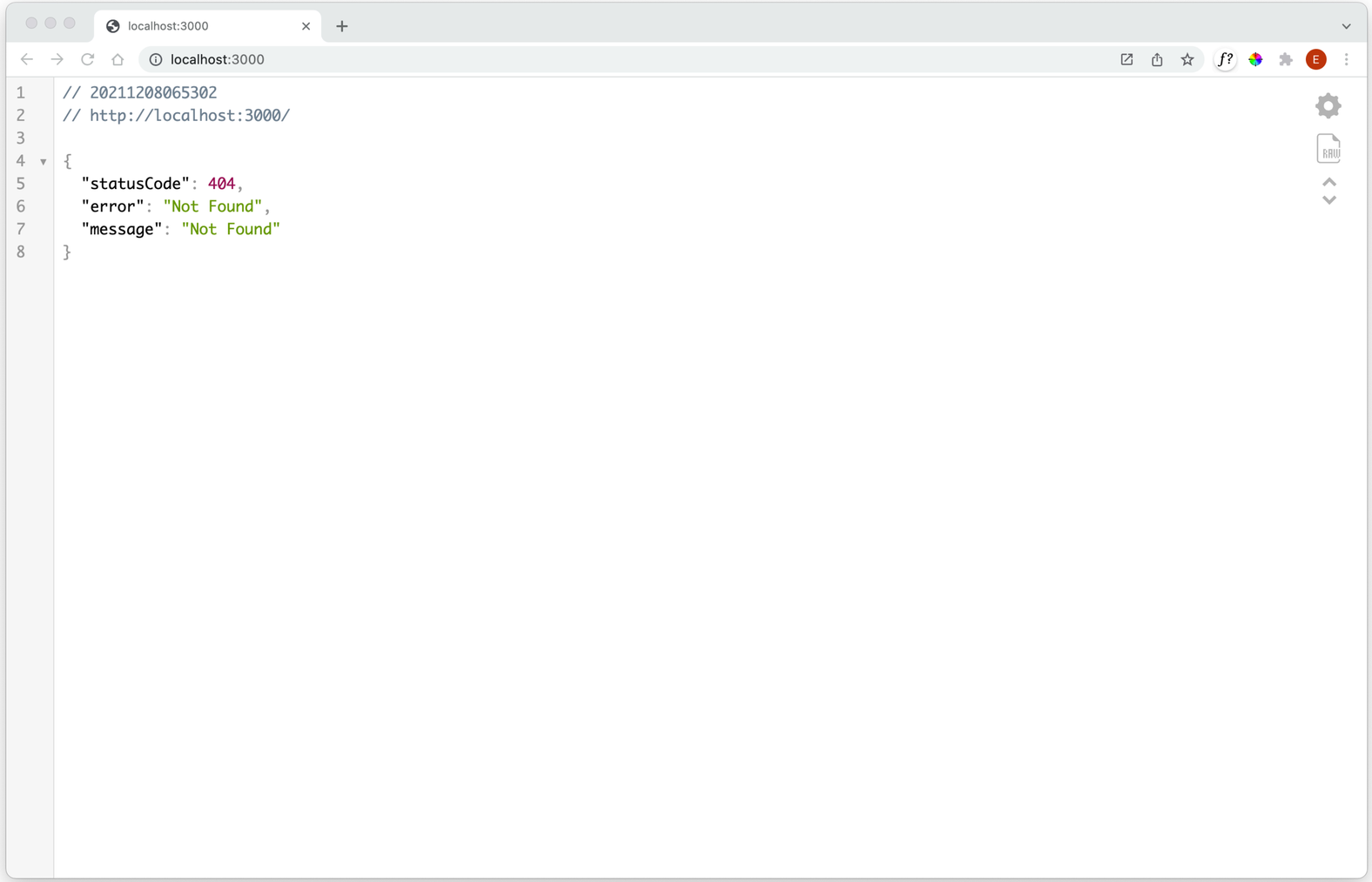
- Can also do ‘npm run lint’…

Running the application

- Start the application …

```
npm run start


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

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

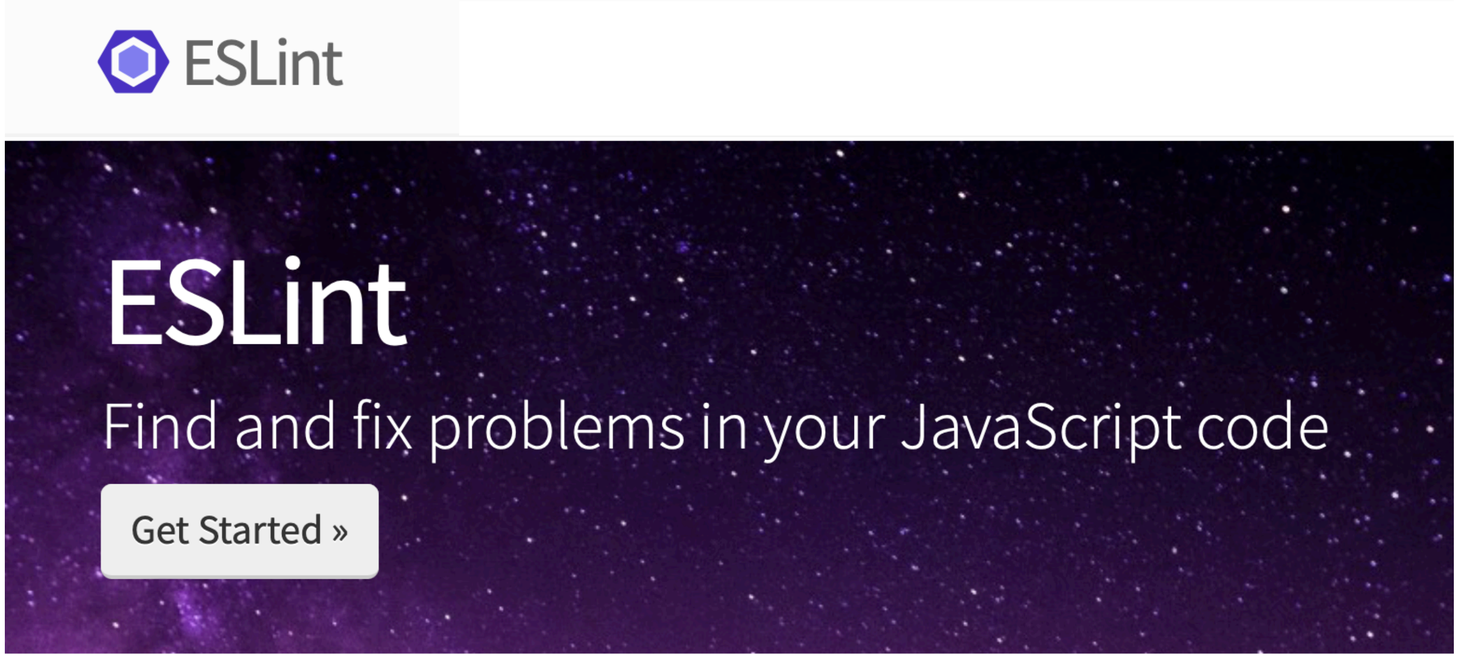
- …no routes defined yet



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

Code Formatting

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

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

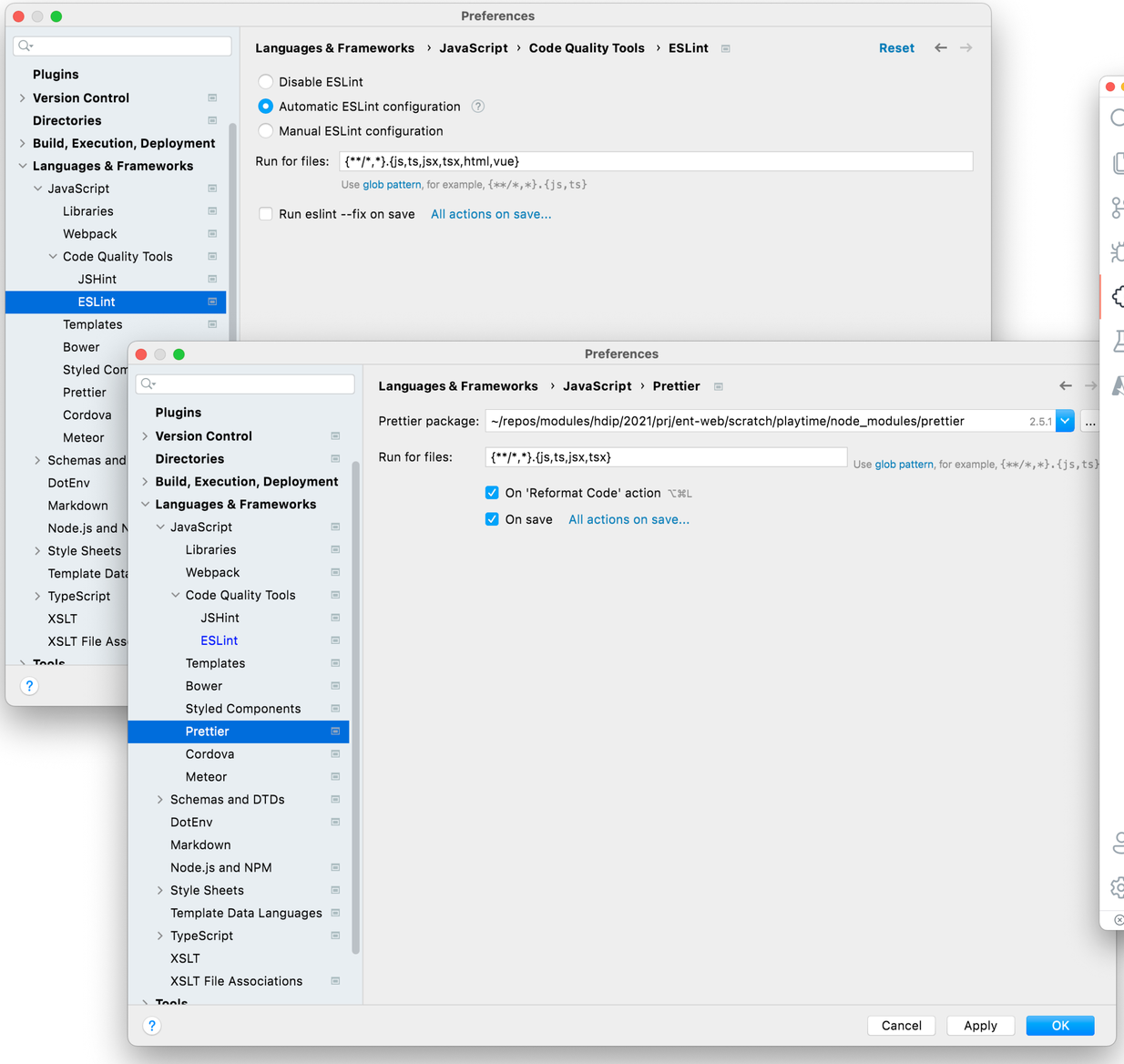
```
npm install -D eslint
npm install -D eslint-config-airbnb-base
npm install -D eslint-config-prettier
npm install -D eslint-plugin-import
npm install -D prettier
```

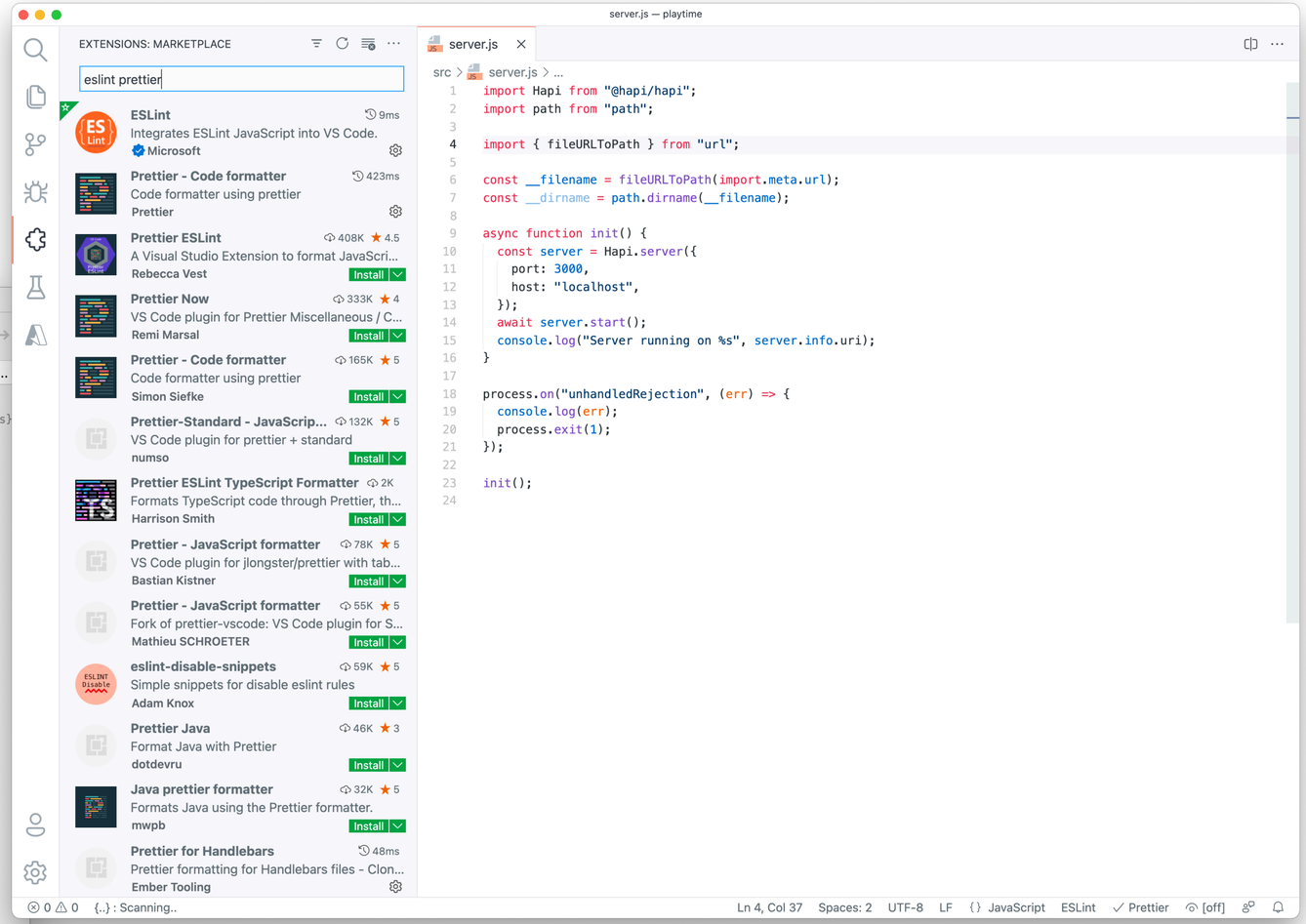
- ESLint
- Prettier
- Airbnb Code Style

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

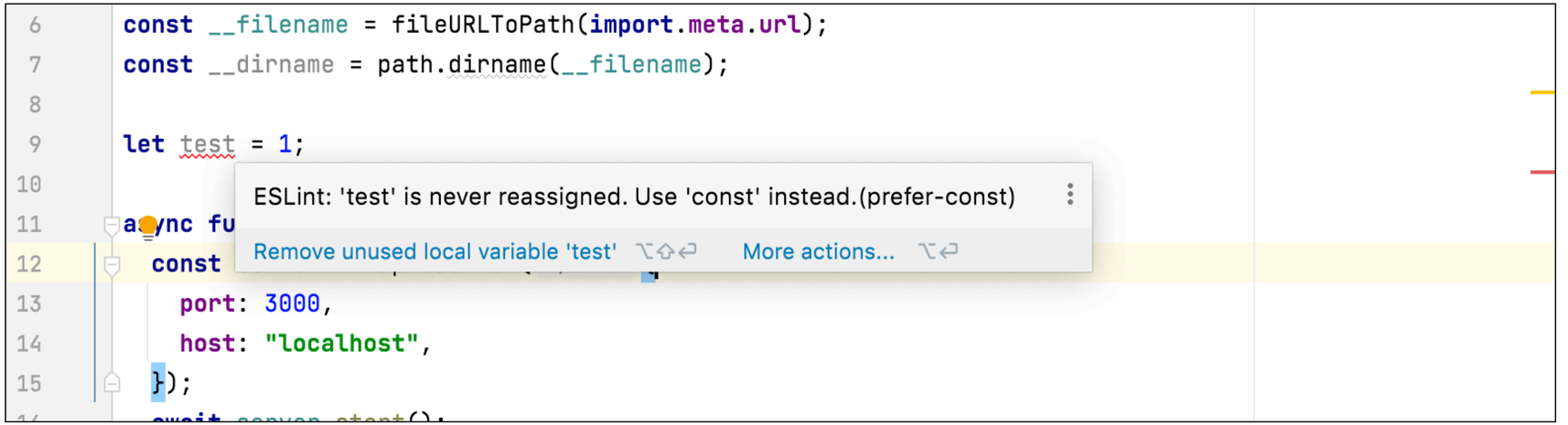




WebStorm

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.
```

Code Quality Tools

Full Stack Web Development