

Prérequis

- [Node.js](#) (v16+)
- Git installé
- Un compte GitHub
- VS Code (recommandé) avec l'extension ESLint

1 Initialisation du projet et installation d'ESLint

1. Créez un nouveau projet et initialisez Git :

```
mkdir tp-eslint-git && cd tp-eslint-git  
git init  
npm init -y
```

2. Installez ESLint :

```
npm install eslint --save-dev
```

3. Configurez ESLint :

```
npx eslint --init (ou npm init @eslint/config@latest)
```

ð Configuration recommandé :

- What do you want to lint? Javascript
- How would you like to use ESLint? To check syntax and find problems
- What type of modules? CommonJS
- Which framework? None of these
- Does your project use TypeScript? No
- Where does your code run? Node

4. Créez un .gitignore :

```
echo "node_modules/" >> .gitignore  
echo ".eslintcache" >> .gitignore
```

2 Test d'ESLint sur un fichier JavaScript

- Créez un fichier `app.js` avec des erreurs :

```
const x=10;
console.log(x);

function test(){
  console.log('test')
}

test()
```

Lancez ESLint :

```
npx eslint app.js
```

- Observez les erreurs (indentation, `console.log`, etc.).

Troubleshooting: Si aucune erreur, il se peut que la configuration de votre `.mjs` soit minimale et n'active aucune règle. Modifiez le comme suit et lancez à nouveau Eslint

```
export default defineConfig([
  {
    files: ["**/*.{js,mjs,cjs}"],
    plugins: { js },
    languageOptions: {
      globals: globals.node,
      ecmaVersion: "latest",
      sourceType: "commonjs"
    },
    rules: {
      "indent": ["error", 2],
      "quotes": ["error", "single"],
      "semi": ["error", "always"],
      "no-console": "warn",
      "no-unused-vars": "error"
    }
  },
  js.configs.recommended
]);
```

2. Corrigez le fichier manuellement ou avec --fix :

```
npx eslint --fix app.js
```

3]Intégration avec Git Hooks (Husky)

Husky permet d'automatiser la vérification du code avant qu'il ne soit commisé. Ainsi le code 'non-conforme' n'entre pas dans votre dépôt.

1. Installez Husky (pour les hooks Git) :

```
npm install husky --save-dev
```

2. Initialiser Husky(crée automatiquement .husky/pre-commit) et configurez le hook :

```
npx husky init
```

```
echo "npx eslint ." > .husky/pre-commit
```

3. Testez le hook :

```
git add .
```

```
git commit -m "Test du hook ESLint"
```

- Si le code contient des erreurs, le commit est bloqué.

4 Configuration avancée d'ESLint

1. Modifiez `eslint.config.mjs` pour personnaliser les règles :

```
{  
  //....  
  languageOptions: {  
    globals: {  
      ...globals.browser,  
      ...globals.node  
    },  
    //....  
  },  
  rules: {  
    //....  
    "eqeqeq": ["error", "always"],  
    "curly": ["error", "all"],  
    "space-before-function-paren": ["error", "never"],  
    "comma-dangle": ["error", "never"],  
    "object-curly-spacing": ["error", "always"],  
    "array-bracket-spacing": ["error", "never"]  
  }  
}
```

2. Ajoutez un script `lint` dans `package.json` :

```
"scripts": {  
  "lint": "eslint .",  
  "lint:fix": "eslint . -fix"  
}
```

3. Testez la configuration :

```
npm run lint
```

5 Mise en place de GitHub Actions

1. Créez un dépôt GitHub et liez-le
2. Ajoutez un workflow CI (.github/workflows/lint.yml) :

```
name: Lint Code
on: [push, pull_request]
jobs:
  lint:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v4

      - name: Setup Node.js
        uses: actions/setup-node@v4
        with:
          node-version: '20'
          cache: 'npm'

      - name: Install dependencies
        run: npm ci

      - name: Run ESLint
        run: npm run lint
```

3. Poussez les changements et vérifiez l'exécution sur GitHub :

```
git add .
```

```
git commit -m "Ajout de GitHub Actions"
```

```
git push
```

- Allez dans l'onglet Actions de votre dépôt pour voir le résultat.

6 Simulation travail d'équipe

1. Créez une branche avec du code non conforme :

```
git checkout -b feature/ajout-fonction
```

- Ajoutez un fichier avec des erreurs ESLint (ex : utils.js).

2. Essayez de commiter (le hook doit bloquer) :

```
git add .
```

```
git commit -m "Code non conforme"
```

3. Corrigez les erreurs et créez une Pull Request :

```
npx eslint --fix .
```

```
git add .
```

```
git commit -m "Correction des erreurs ESLint"
```

```
git push --set-upstream origin feature/ajout-fonction
```

- Vérifiez que GitHub Actions valide la PR.
- Soumettez vos repositories GIT et vos résultats d'erreur dans le readme