# Vérifiez que votre code respecte les conventions de codage

5–6 minutes

---

Votre mission pour Les Bonnes Pièces est terminée ! Le site affiche correctement les fiches produits et les avis des utilisateurs dans une interface web interactive et dynamique. Bravo ! 

Mais puisque vous êtes un bon développeur, vous ne pouvez pas vous arrêter en si bon chemin.

Le code que vous avez produit est peut-être fonctionnel, mais il doit aussi être facile à lire et correctement présenté. Découvrez avec moi comment utiliser le linter pour vérifier votre code source.

## Adoptez les conventions de codage

Dans un projet informatique, que vous soyez seul ou à plusieurs, je vous recommande de respecter des **conventions de codage**. Il s'agit d'un ensemble de règles sur la façon d'écrire le code informatique.

Ces règles concernent trois aspects de l'écriture du code.

**1. La gestion des erreurs de programmation**, comme des plantages ou un comportement anormal.

> Par exemple : une boucle qui ne se finit jamais car vous avez oublié l'instruction d'incrémentation, ou une condition qui utilise le symbole d'assignation (un seul égal =) au lieu de l'opérateur de comparaison (trois signes égal ===).

**2. L'utilisation de bonnes pratiques** qui vous éviteront des bugs sur le long terme.

> Par exemple : utiliser le triple égal === pour comparer à la fois le type et la valeur, au lieu du double égal == qui ne compare que la valeur sans prendre en compte le type.

**3. L'application d'une mise en forme** pour l'écriture du code source.

> Par exemple : Doit-on utiliser des espaces ou des tabulations ? Doit-on revenir à la ligne pour écrire l'accolade ouvrante ?

Les règles concernant les erreurs de programmation et les bonnes pratiques sont très largement adoptées par les développeurs. En revanche, il est plus difficile d'établir un consensus sur la mise en forme du code. Cet aspect varie souvent d'un développeur à l'autre.

Le linter s'utilise à l'aide de la ligne de commande. Les résultats de la vérification sont donc affichés dans le terminal, où vous retrouverez trois informations pour chaque élément à corriger : la nature de l'erreur, le nom du fichier concerné et la ligne de l'erreur.

Il existe plusieurs conventions de codage publiques et populaires. En plus de faire respecter les bonnes pratiques et de corriger les erreurs de programmation, elles ont l'avantage de proposer une mise en forme par défaut. On peut citer notamment :

- Google JavaScript Style Guide ;
- Airbnb JavaScript Style Guide ;
- JavaScript Standard Style ;
- les options par défaut de ESLint.

Dans ce cours, je vous propose d'utiliser les options par défaut de ESLint, tout en configurant 3 préférences :

- l'utilisation de guillemets doubles ;
- la présence de points-virgules à la fin des instructions ;
- l'indentation par tabulation.

**Utilisez ESLint pour analyser votre code**

Pour vérifier votre code, vous devez d'abord installer le paquet eslint dans votre projet à l'aide de la commande suivante :

```
npm install eslint
```

Ensuite, vous devez dire à ESLint quelles règles appliquer. Pour cela, lancez la commande suivante :

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

Voici, par exemple, les réponses que j'ai données. Vous êtes libre de choisir d'autres réponses pour les quatre dernières questions, car il s'agit de préférences de style.

```
✔ How would you like to use ESLint? · style
✔ What type of modules does your project use? · esm
✔ Which framework does your project use? · none
✔ Does your project use TypeScript? · No / Yes
✔ Where does your code run? · browser
✔ How would you like to define a style for your project? · prompt
✔ What format do you want your config file to be in? · JSON
✔ What style of indentation do you use? · tab
✔ What quotes do you use for strings? · double
✔ What line endings do you use? · unix
✔ Do you require semicolons? · No / Yes
```

En conséquence, la commande npm init a créé un fichier `.eslintrc.json` qui documente ces options choisies au format JSON. Vous pouvez le modifier à la main pour changer d'autres règles si vous le désirez.

Il ne reste plus qu'à lancer ESLint pour effectuer la vérification. Pour cela, nous ajoutons un script dans le fichier package.json :

```
"scripts": {
  "dev": "http-server",
  "lint": "eslint *.js"
 }
```

Ce script exécutera la commande eslint pour tous les fichiers se terminant par ".js". On lance ce script avec la commande npm run lint. Voici le résultat pour mon code source :

```
  30:34  error  Strings must use doublequote                        quotes
  34:49  error  Strings must use doublequote                        quotes
  79:8   error  'graphiqueAvis' is assigned a value but never used  no-unused-vars
  79:28  error  'Chart' is not defined                              no-undef
 123:6   error  'Chart' is not defined                              no-undef
```

```
✖ 5 problems (5 errors, 0 warnings)
  2 errors and 0 warnings potentially fixable with the `--fix` option.
```

Résultat de la commande eslint sur tous les fichiers se terminant par “.js”

On voit qu’il y a cinq erreurs dans le fichier avis.js. À la ligne 34, j’ai utilisé des guillemets simples au lieu de doubles guillemets. Je dois les remplacer par des guillemets doubles. À la ligne 79, j’ai déclaré la constante graphiqueAvis, mais je ne l’ai pas utilisée après. Je dois donc supprimer la déclaration de la constante, tout en laissant la création du graphique (tout ce qu’il y avait à droite du symbole égal).

Dans ce cas, la solution consiste à indiquer à ESLint que Chart est une variable globale, à l’aide de ce commentaire que l’on place en haut du fichier pieces.js :

```
/* global Chart */
```

Une fois ces corrections effectuées, lorsque je relance la commande npm run lint, il n’y a aucun message d’erreur dans le terminal. Tout est bon !

### Récapitulons en vidéo

Vous pouvez revoir les différentes étapes de cette démonstration dans la vidéo ci-dessous :

### En résumé

- Respectez les conventions de codage pour :
- avoir un code de bonne qualité, consistant et facilement lisible ;
- uniformiser l’écriture du code à plusieurs.
- Utilisez la librairie ESLint pour :
- configurer les règles à respecter ;
- détecter automatiquement les erreurs.