



ESLint

Gold - Chapter 8 - Topic 4

Selamat datang di **Chapter 8 Topik 4** online
course **Fullstack Web** dari Binar Academy!





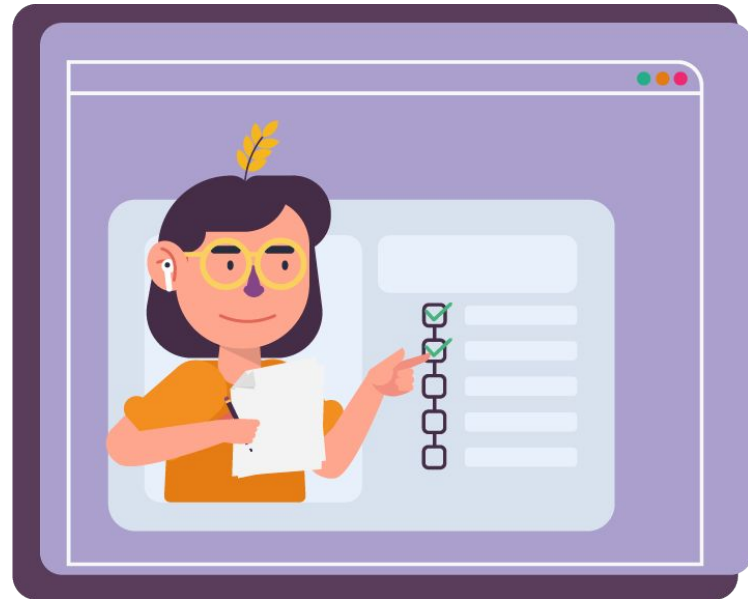
Nah, pada topik keempat, kita bakal bahas penerapan ESLint pada kodingan kita. Cus langsung aja~





Detailnya, kita bakal bahas hal-hal berikut ini:

- Memahami konsep ESLint
- Melakukan instalasi dan konfigurasi ESLint di project Javascript





Dari topik-topik yang sudah dibahas, kamu sudah banyak mencoba melakukan koding kan? Nah, sekarang kita akan mempelajari hal simpel tapi gak kalah penting, yaitu **menulis code dengan standar yang benar**. Atau bahasa developernya **linting**. Bukan linting rokok ya, tapi code linting.





Javascript adalah **bahasa pemrograman super fleksibel**. Dalam satu aspek, itu adalah hal baik karena memungkinkan kita membuat berbagai hal dengan mudah. Disisi lain, **banyak error atau bug potensial muncul dari code yang terlalu “nyeleneh”**. Maka, dibuatlah **ESLint** yang berfungsi sebagai panduan sekaligus pemeriksa code.

Kenapa tidak dimunculkan error nya dari awal saja supaya langsung diketahui? Nah, inilah yang mendasari dibuatnya ESLint~





Kita mulai dari ESLint ya~

Dalam [website resminya](#), ESLint merupakan **tools untuk mengidentifikasi dan melaporkan kode Javascript/ECMAScript tertentu dengan tujuan membuat kode lebih konsisten dan minimalisasi bugs**. Konsekuensinya, ESLint “memaksa” kita sebagai developer untuk menulis kode dengan standar tinggi.





Tools sejenis ini sebenarnya sudah bukan barang asing di dunia pemrograman. Program Lint untuk bahasa C sudah ada sejak tahun 1979. Saat ini, ada berbagai tools serupa di bahasa pemrograman lain misal scss-lint untuk Sass, Pylint untuk Python, coffee-lint untuk Coffeescript, dan Rubocop untuk Ruby.

Setelah tahu pengertiannya, penasaran ga sih gimana caranya ESLint bekerja?



Berikut nih proses tools linting atau beberapa prinsip ESLint dan tools sejenis bekerja!

- Terdapat **sekumpulan aturan (rules) atau pola (pattern)** tentang code. Misal, tidak boleh ada spasi pada kodingan dengan fungsi yang berbeda, harus menggunakan camelcase, dsb.
- ESLint melakukan **scanning pada file project** untuk mencari pelanggaran aturan/pola
- Jika menemukan pelanggaran, ESLint akan **membuat flag berupa warning atau error**
- Proses dapat diintegrasikan dengan code editor atau toolchain untuk build

Kalau sekarang kamu lagi bertanya-tanya, Kenapa sih harus repot-repot pakai ESLint? Kita langsung bahas yuk!



Sab! Coba jelasin secara singkat dong kenapa kita sebagai developer harus repot-repot pakai ESLint? 🤔

Jadi gini, Proses kerja yang tadi kamu bayangkan itu sangat masuk akal. Tapi, sebenarnya paradigma yang-penting-cepat-jadi justru menghambat produktivitas ketika skala aplikasinya semakin besar. Error dan bug itu meningkat eksponensial seiring bertambahnya baris kode.

Pola yang diatur dalam ESLint adalah hasil pengalaman ribuan developer dalam menyelesaikan berbagai macam error dan bug, sampai akhirnya bisa meningkatkan kualitas kode. Nah, jadi dengan kita belajar ESLint, **kita bisa menyelamatkan diri sendiri dan menghilangkan dari error dan bug potensial yang bertubi-tubi.**





Tapi perlu diingat ya guys! **Linter hanya mengevaluasi penulisan dari kodenya saja lho**, kualitas logika program tetap ditentukan oleh developernya. Jadi, kita sebagai developer tetap punya kapasitas yang besar dalam penulisan kode, sehingga tidak hanya berpatok kepada linter.

Seperti kata pepatah, berakit-rakit ke hulu berenang-renang ke tepian. Rumit di ESLint dahulu, hidup tenang di production kemudian~

*Coding-an boleh sama.
Tapi soal kualitas logika
Sabrina oke punya~ 😊*





Setelah kita tahu konsep ESLint dan manfaatnya untuk proses development, sekarang saatnya kita melakukan **setup ESLint**.





Berikut nih Setup ESLint yang akan kita bahas meliputi :

1. **Setup project Node.JS umum**
2. **Setup integrasi dengan code editor**

Kita cek satu persatu, yuk!





1. Setup Project Node.JS umum

Kita dapat melakukan setup di project Node.Js apapun selama di project tersebut sudah ada file package.json. Jika belum ada file tersebut di folder project, ketik perintah berikut di terminal

```
$ npm init
```

Selanjutnya, lakukan instalasi ESLint secara lokal di environment development dengan perintah.

```
$ npm i eslint --save-dev
```

Mengapa instalasi lokal untuk environment development? Pertama, instalasi lokal memungkinkan kita bisa fleksibel dalam konfigurasi ESLint. Kemudian kita hanya memerlukannya untuk development, bukan production.

Untuk inialisasinya kita bisa menggunakan perintah berikut

```
$ ./node_modules/.bin/eslint --init
```






Nah, Setelah inisialisasi di terminal, akan ada opsi penggunaan ESLint di terminal. Pilih dan tekan enter sesuai kebutuhan.

Hasil akhirnya akan ada **file .eslintrc.{js,yml,json}** di folder project kita. File itulah yang menjadi patokan konfigurasi ESLint, meliputi peraturan apa yang dipakai, dan tindakan apa yang diberikan.

Sekarang kita bisa menjalankan ESLint di folder project. Kita perlu memilih file/folder yang dicek oleh ESLint. Ups.. tapi sebelum itu, perlu diperhatikan bahwa ESLint kita ada di instalasi lokal sehingga belum tentu bisa dipanggil langsung. Cara untuk memanggilnya ada 2 :

1. Panggil dengan merujuk node_modules : **\$./node_modules/.bin/eslint app.js**
2. Tambahkan script di package.json seperti tertera di samping, kemudian panggil dengan **\$ npm run eslint**



```
"scripts": {  
  ...  
  "eslint": "eslint app.js"  
}
```



Gambar di samping adalah contoh config ESLint dengan format JSON. Nama standarnya adalah `.eslintrc.json`.

Di dalamnya ada beberapa item config seperti **extends**, **parser**, **parserOptions**, **plugins**, dan **rules**. Tapi config yang berpengaruh besar dalam proses linting ada di poin rules.

Poin rules berisi aturan apa yang berlaku dan bagaimana mengelolanya. Secara umum ada 3 pengelolaan rules :

- “off”** → diperbolehkan
- “warn”** → diperbolehkan, dengan warning di console
- “error”** → tidak diperbolehkan, keluar error

Contoh config ESLint

`.eslintrc.json`

```
{
  "extends": [ "standard" ],
  "parser": "@typescript-eslint/parser",
  "parserOptions": { "ecmaVersion": 11 },
  "plugins": [ "@typescript-eslint" ],
  "rules": {
    "no-useless-constructor": "error",
    "camelcase": "off",
    "no-unused-vars": "warn",
    "callback-return": [ "warn", [ "callback", "next" ] ]
  }
}
```



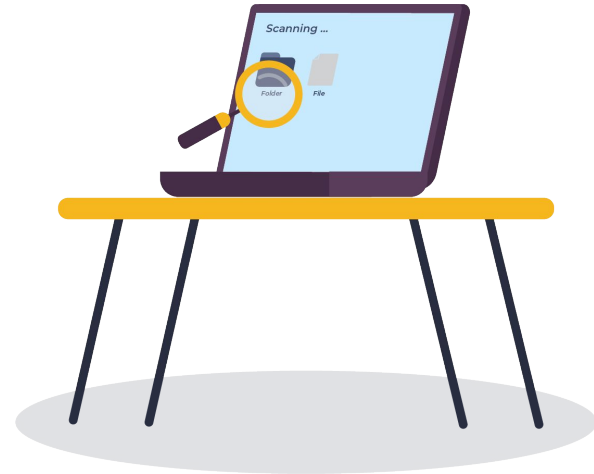

Kalau kamu sudah berhasil set up ESLINT, nantinya di Node.JS kamu akan muncul tampilan kaya gambar disamping ~

```
$ ./node_modules/.bin/eslint --init
? How would you like to use ESLint? ...
  To check syntax only
> To check syntax and find problems
✓ How would you like to use ESLint? - style
? What type of modules does your project use? ...
> JavaScript modules (import/export)
  CommonJS (require/exports)
  None of these
```



Setelah melakukan set up dan config tadi, sekarang kita bisa memilih untuk melakukan scan terhadap file maupun folder.

- **Untuk scan file**, sebut nama file nya (bisa lebih dari satu file, dipisah dengan menggunakan spasi), misalnya :
`./node_modules/.bin/eslint app.js index.js`
- **Untuk scan folder** langsung sebut nama foldernya untuk menandakan scan semua elemen folder, misalnya
`./node_modules/.bin/eslint ./src`





2. Setup ESLint dengan Code Editor

ESLint dapat diintegrasikan dengan berbagai text editor.
Keterangan lengkapnya dapat dilihat di [dokumentasi ESLint](#).

Untuk VSCode sebagai code editor populer, setup ESLint dapat dilakukan dengan install [extension ESLint](#).





Konsep ESLint pada topik ini cukup sederhana.

Karena yang paling penting adalah **penguasaan aturannya** dengan belajar sambil melakukan koding langsung di banyak project.

Untuk lebih menguasainya kamu bisa elaborasi disini ya! <https://eslint.org/docs/rules/>



Saatnya kita
Quiz!





1. File konfigurasi .eslintrc tidak tersedia dalam format ...

- A. .yaml
- B. .xml
- C. .js



1. File konfigurasi .eslintrc tidak tersedia dalam format ...

- A. .yaml
- B. .xml**
- C. .js

File .eslintrc bisa tersedia dalam tiga format : .js atau .yaml atau .json



2. Mode instalasi ESLint yang direkomendasikan adalah ...

- A. Instalasi global di env development
- B. Instalasi lokal di semua env
- C. Instalasi lokal di env development



2. Mode instalasi ESLint yang direkomendasikan adalah ...

- A. Instalasi global di env development
- B. Instalasi lokal di semua env
- C. Instalasi lokal di env development**

Instalasi lokal direkomendasikan untuk fleksibilitas config. Env development direkomendasikan agar tidak menambah beban saat build production



3. ESLint yang dipanggil melalui command line terminal dapat melakukan scanning 2 file sekaligus dengan perintah ...

- A. `$ eslint file1.js file2.js`
- B. `$ eslint (file1.js, file2.js)`
- C. `$ eslint file1.js && file2.js`



3. ESLint yang dipanggil melalui command line terminal dapat melakukan scanning 2 file sekaligus dengan perintah ...

- A. `$ eslint file1.js file2.js`
- B. `$ eslint (file1.js, file2.js)`
- C. `$ eslint file1.js && file2.js`

Scan beberapa file sekaligus dapat dilakukan dengan memasukkan file yang mau di-scan, dipisah dengan spasi. Keterangan lebih lengkap dapat dicek di dokumentasi.



4. Salah satu tujuan mengimplementasikan ESLint dalam project javascript adalah ...

- A. Mempercepat proses development
- B. Mengurangi potensi munculnya bug
- C. Membuat kode menjadi lebih ringkas



4. Salah satu tujuan mengimplementasikan ESLint dalam project javascript adalah ...

- A. Mempercepat proses development
- B. Mengurangi potensi munculnya bug**
- C. Membuat kode menjadi lebih ringkas

ESLint dapat meminimalkan code yang berisiko memunculkan bug. Di sisi lain, ESLint cenderung menambah waktu development karena mendorong standar tinggi. Soal panjang pendeknya code tidak ditangani oleh ESLint.



5. ESLint dapat diintegrasikan dengan toolchain eksternal untuk mempermudah penggunaannya. Contoh tools yang dapat diintegrasikan dengan ESLint adalah ...

- A. Code editor dan build tools
- B. Browser dan database
- C. Repository dan API



5. ESLint dapat diintegrasikan dengan toolchain eksternal untuk mempermudah penggunaannya. Contoh tools yang dapat diintegrasikan dengan ESLint adalah ...

- A. Code editor dan build tools**
- B. Browser dan database
- C. Repository dan API

Code editor memiliki extension yang melakukan linting langsung dalam project. Build tools dapat diintegrasikan untuk mengecek code sebelum build. Opsi lainnya tidak memiliki hubungan langsung dengan kualitas kode.



Referensi dan bacaan lebih lanjut~

1. [Lint Kodemu](#)
2. [Repository](#)
3. [Dokumentasi Resmi](#)





Nah, selesai sudah pembahasan kita di **Chapter 8 Topic 4** ini.

Selanjutnya, kita bakal bahas tentang gimana caranya **handling media** di aplikasi web kita. Yukkk! Gas! Ngeng~



Terima Kasih!



Next Topic

loading...