

Project Standards là gì?

"Project Standards" là **tập hợp các quy chuẩn và công cụ** được áp dụng trong dự án để đảm bảo:

- Code có **chất lượng cao**
- Được **viết nhất quán**
- **Dễ bảo trì và mở rộng**
- Hạn chế tối đa lỗi **ngay từ khi viết mã**, thay vì đến lúc chạy mới phát hiện

CÁC THÀNH PHẦN CHÍNH

1. ESLint – *Kiểm tra chất lượng mã*

Mục tiêu:

Giúp phát hiện **lỗi cú pháp, lỗi logic, hoặc code không chuẩn** trong JavaScript/TypeScript trước khi chạy.

Cách dùng:

- Cấu hình trong file `.eslintrc.js`
- Cảnh báo/lỗi sẽ xuất hiện trong IDE hoặc khi chạy lệnh lint
- Giúp giữ code sạch và tránh lỗi khó phát hiện

Ví dụ lỗi mà ESLint phát hiện:

- Dùng biến chưa khai báo
 - Có đoạn code không dùng đến
 - Viết function không đúng chuẩn quy định
-

2. Prettier – *Tự động định dạng mã*

Mục tiêu:

Giúp mọi người trong team **viết code theo một style thống nhất** (indent, khoảng trắng, dấu chấm phẩy, v.v.)

Cách dùng:

- Cấu hình trong `.prettierrc`
- IDE sẽ tự động định dạng khi bạn **nhấn lưu (format on save)**

Lợi ích:

- Tự động sửa lỗi style (thay vì phải sửa tay)
 - Giảm xung đột merge do format khác nhau
-


3. TypeScript – *Kiểm tra kiểu dữ liệu*

Mục tiêu:

Giúp kiểm tra kiểu dữ liệu **ngay trong lúc viết code** → phát hiện lỗi khi refactor hoặc thao tác với dữ liệu phức tạp.

Lợi ích:

- Phát hiện lỗi sớm
- Code an toàn, tự tin khi thay đổi logic
- Giúp IDE cung cấp autocomplete tốt hơn

 Lưu ý: TypeScript chỉ **kiểm tra tại build-time**, không ngăn lỗi khi chạy (runtime).

4. Husky – *Chặn commit lỗi*

Mục tiêu:

Không cho commit code nếu nó vi phạm quy chuẩn (lint, format, test...)

Cách dùng:

- Cài đặt Husky để chạy lệnh kiểm tra trước khi commit (git hooks)
- Ví dụ: kiểm tra lint + prettier + tsc trước khi **git commit**

Lợi ích:

- Bảo vệ repo khỏi code lỗi
 - Giúp đồng bộ giữa các thành viên team
-

5. Absolute Imports – *Import tuyệt đối*

Vấn đề với import tương đối:

```
ts
import Button from "../../components/Button";
```

Giải pháp với absolute import:

```
ts
import Button from "@components/Button";
```

Cách cấu hình:

- JS: **jsconfig.json**
- TS: **tsconfig.json**

```
json
{
  "compilerOptions": {
    "baseUrl": ".",
    "paths": {
      "@/*": ["/src/*"]
    }
  }
}
```

Lợi ích:

- Dễ đọc
- Dễ di chuyển file mà không bị gãy đường dẫn

6. File & Folder Naming Convention – Chuẩn hoá tên file/thư mục

Mục tiêu:

Tạo sự **nhất quán** trong cách đặt tên file/folder → dễ tìm, dễ đọc

Quy chuẩn ví dụ:

- Dùng **kebab-case** cho tất cả các file (`user-profile.tsx`)
- Tránh đặt tên tùy tiện hoặc có khoảng trắng

Cách kiểm tra: dùng plugin ESLint:

```
'check-file/filename-naming-convention': [  
  'error',  
  {  
    '**/*. {ts,tsx}': 'KEBAB_CASE',  
  },  
  { ignoreMiddleExtensions: true },  
]
```

💡 Tổng kết: Tại sao nên dùng Project Standards?

Lý do	Lợi ích
✅ Code sạch, ít lỗi	Phát hiện lỗi sớm bằng lint, type check
✅ Dễ bảo trì	Code thống nhất, dễ đọc, dễ hiểu
✅ Làm việc nhóm tốt hơn	Mọi người theo chung một chuẩn
✅ Chất lượng chuyên nghiệp	Giống các dự án production thực tế