Don’t Use Enums in TypeScript

Embracing Structural Typing with Alternatives

The Problem with Enums

- Not part of native JavaScript
- Break TypeScript's structural typing
- Restrict value passing
- Introduce unexpected behavior
- Complicate type inference

```
enum LogLevel {
  DEBUG,
  WARNING,
  ERROR,
}

function log(message: string, level: LogLevel) {
  console.log(`[${level}] ${message}`);
}

log('This works', LogLevel.DEBUG);
```

The Pitfalls of const enums

TypeScript

```ts
enum LogLevel {
  DEBUG,
  WARNING,
  ERROR,
}

function log(message: string, level: LogLevel) {
  console.log(`[${level}] ${message}`);
}

log('This works', LogLevel.DEBUG);
console.log(Object.values(LogLevel))
```

Transpiled JavaScript

```ts
var LogLevel;
(function (LogLevel) {
    LogLevel[LogLevel["DEBUG"] = 0] = "DEBUG";
    LogLevel[LogLevel["WARNING"] = 1] = "WARNING";
    LogLevel[LogLevel["ERROR"] = 2] = "ERROR";
})(LogLevel || (LogLevel = {}));
function log(message, level) {
    console.log("[".concat(level, "] ").concat(message));
}
log('This works', LogLevel.DEBUG);
console.log(Object.values(LogLevel));
```

```
[0] This works
["DEBUG", "WARNING", "ERROR", 0, 1, 2]
```

The Solution: POJO with 'as const'

```
const LOG_LEVEL = {
  DEBUG: 'debug',
  WARNING: 'warning',
  ERROR: 'error'
} as const;
// Type magic
type LogLevel = typeof LOG_LEVEL[keyof typeof LOG_LEVEL];

function log(message: string, level: LogLevel) {
  console.log(`[${level}] ${message}`);
}

log('This works', LOG_LEVEL.DEBUG);
```

- Uses native JavaScript object
- Preserves structural typing
- Allows flexible value passing
- Consistent with TypeScript philosophy
- Provides better type inference

Best Practices

- Avoid using enums, including const enums
- Use POJOs with 'as const' for constant values
- Embrace structural typing principles
- Write idiomatic JavaScript with TypeScript types

Remember

- Enums break structural typing
- POJOs are more flexible and idiomatic
- TypeScript is about shape, not names
- Good TypeScript is good JavaScript

Questions?

Have you encountered issues with enums in your TypeScript projects?

Thank You!

Remember: In TypeScript, shape matters more than names!