CLEAN CODE

NA PRATICA

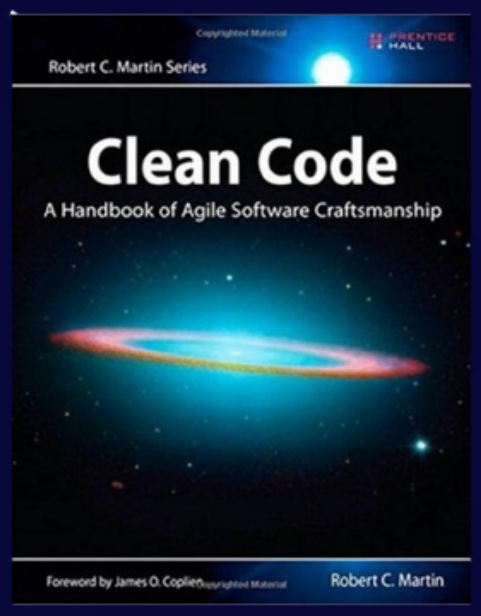
1PT - NOMES SIGNIFICATIVOS

```
const data = []
const results = []
const json = []
const filter = []
const active = []
```

Quanto nome ruim em um só lugar 🤮

Isaac Gomes

CLEAN CODE



Clean Code é um conceito que se refere à escrita de código de forma clara, organizada e legível, facilitando a manutenção, entendimento e evolução de sistemas de software. A ideia central é que o código deve ser simples e direto, evitando complexidade desnecessária e favorecendo a legibilidade para outros desenvolvedores.

Isaac Gomes

Capitulo 1

NOMES SIGNIFICATIVOS

Nomes que revelam seus Propósitos

pensemos um cenário onde um usuário vai ter acesso a um recurso específico no premium da categoria dele... qual seria um bom nome para a const que guarda o user???

Isaac Gomes

Capitulo 1

NOMES SIGNIFICATIVOS

Nomes que revelam seus Propósitos

```
const user = {}
```

❌ - *pouco descritivo*

```
// user represents a premium user
// with access to special features
const user = {}
```

❌ - *comentario para se explicar*

```
const premiumUserWithAccessToXFeature = {}
```

✅ - *autoexplicativo*

Isaac Gomes

Capitulo 1

NOMES SIGNIFICATIVOS

Evite Informações Erradas

pensemos em um cenário onde temos que filtrar usuários banidos pense em um nome que pode te induzir ao erro

Isaac Gomes

Capitulo 1

NOMES SIGNIFICATIVOS

Evite Informações Erradas

```
function filterActiveUsers(users){
    return users.filter(user => user.isBanned);
}
```

❌ *- nome informa errado*

```
function filterBannedUsers(users){
    return users.filter(user => user.isBanned);
}
```

✅ *- nome que informa de maneira correta*

Isaac Gomes

TS

Capitulo 1

NOMES SIGNIFICATIVOS

Distinções significativas

evite usar termos similares
para informações distintas
deixe o mais legível possível

Isaac Gomes

Capitulo 1

NOMES SIGNIFICATIVOS

Distinções significativas

```
const user1 = {}
const user2 = {}
```

❌ - ***nomes similares***

```
const adminUser = {}
const guestUser = {}
```

✅ - ***nomes distintos***

Isaac Gomes

TS

Capitulo 1

NOMES SIGNIFICATIVOS

nomes pronunciáveis

use nomes pronunciáveis,
evite abreviações que podem
gerar confusões

Isaac Gomes

Capitulo 1

NOMES SIGNIFICATIVOS

nomes pronunciáveis

```
const t3rmlntrs: number; = 10
```

❌ *- contem abreviações confusas*

```
const terminalCount: number; = 10
```

✅ *- sem abreviações e claro*

Isaac Gomes

Capitulo 1

NOMES SIGNIFICATIVOS

nomes passíveis de busca

use nomes descritivos que podem ser pesquisados com facilidade

Isaac Gomes

Capitulo 1

NOMES SIGNIFICATIVOS

nomes passíveis de busca

```
const x: number = 25
```

❌ - ***nome sem busca clara***

```
const temperatureInCelsius: number = 25
```

✅ - ***nome passivel de busca***

Isaac Gomes

Capitulo 1

NOMES SIGNIFICATIVOS

Não de uma de espertinho

Não de uma de espertinho evite nomes genéricos ou vagos... mesmo que tenha contexto da criação escreva de maneira que qualquer um possa ler

Isaac Gomes

Capitulo 1

NOMES SIGNIFICATIVOS

Não de uma de espertinho

```
class DataProcessor {
    process(data) {}
}
```

❌ - ***nome generico***

```
class UserProfileProcessor {
    process(userProfile) {}
}
```

✅ - ***nome explicativo***

Isaac Gomes

Capitulo 1

NOMES SIGNIFICATIVOS

Não faça trocadilhos

Não coloque nomes engraçadinhos ou com mais de uma interpretação possível seja objetivo e claro

Isaac Gomes

Capitulo 1

NOMES SIGNIFICATIVOS

Não faça trocadilhos

```
class DateShifter {
    shiftDate(date, days) {
        return new Date(date.getTime() + days * 86400000);
    }
}
const timeTraveler = new DateShifter();
const futureDate = timeTraveler.shiftDate(new Date(), 3);
```

❌ ***- O nome DateShifter parece engraçado ou criativo, mas o termo "shifter" pode ser confundido com outras coisas (por exemplo, manipulação de bits ou mudanças de estado) e não comunica diretamente que a classe lida com cálculos de datas.***

Isaac Gomes

Capitulo 1

NOMES SIGNIFICATIVOS

Não faça trocadilhos

```
class DateCalculator {
    addDaysToDate(date, days) {
        return new Date(date.getTime() + days * 86400000)
    }
}
const dateCalculator = new DateCalculator();
const futureDate = dateCalculator.addDaysToDate(new Date(), 3);
```

✅ - O nome DateCalculator comunica claramente que a classe realiza cálculos com datas. Além disso, o nome do método addDaysToDate explica explicitamente o que o método faz, tornando o código mais fácil de entender para todos que o leem.

Isaac Gomes

Capitulo 1

NOMES SIGNIFICATIVOS

Não faça trocadilhos

```
class DateCalculator {
    addDaysToDate(date, days) {
        return new Date(date.getTime() + days * 86400000)
    }
}
const dateCalculator = new DateCalculator();
const futureDate = dateCalculator.addDaysToDate(new Date(), 3);
```

✅ - O nome DateCalculator comunica claramente que a classe realiza cálculos com datas. Além disso, o nome do método addDaysToDate explica explicitamente o que o método faz, tornando o código mais fácil de entender para todos que o leem.

Isaac Gomes

Gostou?

Curta

Compartilhe

Salve

Isaac Gomes