04 - Comments

- Comentários são um mal necessário
  - Compensam o que o código não foi capaz de dizer sozinho
  - São sempre um resultado de falha na comunicação
- Podem ficar defasados em relação ao código de verdade.
- Comentários ruins são pior do que nenhum comentário
- Comentários não consertam código ruim
- Exemplo:

```
// Ruim:
   // Check to see if the employee is eligible for full benefits
   if ((employee.flags & HOURLY_FLAG) &&  (employee.age > 65))

// Bom:
        if (employee.isEligibleForFullBenefits())
```

COMENTÁRIOS BONS

- **Comentários legais:**
  - Copyright ou autoria.
- **Comentários informativos:**
  - Descreve o que está sendo feito, embora seja melhor o código ser autoexplicativo
  - Explicar Regex
- **Explicar intenção:**
  - Justificar decisões.
- **Esclarecimento:**
  - Explicar o significado de um valor do código que não podemos alterar nós mesmos
  - Pode ser um risco de estar errado.
- **Alerta:**
  - Avisar que uma função está desligada porque demora muito ou pode apagar algo.
- **TODO:**
  - Descrever tarefas futuras.
  - Não justifica deixar código ruim.
- **Ênfase:**
  - Destacar importância de alguma operação.

COMENTÁRIOS RUINS

- **Murmúrio:**
  - Comentário mal explicado sem sentido.
- **Redudante:**
  - Só repete o que o código já diz por si só sem acréscimo.
- **Enganoso:**
  - Descreve errado o que o código faz
- **Obrigatório:**
  - Exigir um docstring sempre é uma péssima ideia que só polui o código.
- **Registro:**
  - Registro das alterações ao longo do tempo, como um diário.
  - Já foram úteis mas hoje com Git não são mais.
- **Ruído:**

    - Não acrescenta nada. Completamente ignoráveis.
- **Substituição de um bom nome de funções e variáveis:**
    - O código deve falar por si só quando possível
- **Marcadores visuais / Banner:**
    - Desnecessários e poluição visual.
- **Explicação de colchete:**
    - Indicam função muito longa
    - Ifs e whiles e trys devem ser autoexplicativos e fáceis de seguir.
- **Autoria de modificação:**
    - Git já diz quem fez qual alteração, não precisa deixar escrito.
- **Código comentado:**
    - Outros vão ficar com medo de deletar, ficam se acumulando.
    - Não deve existir. Git armazena código antigo para nós.
- **Comentários HTML:**
    - Às vezes usado para renderizar bonito em outra ferramenta.
    - Não deve existir. A outra ferramenta que se encarregue de renderizar certo.
- **Informação não-local:**
    - Comentário deve sempre explicar o código adjacente.
    - Nunca deve explicar coisas do sistema inteiro.
- **Informação demais:**
    - Contextualização histórica.
    - Muitos detalhes, como uma página de um manual.
- **Conexão não-óbvia:**
    - Comentário não pode ser difícil de entender.
    - Comentário não deve precisar de um comentário para explicar a si próprio.
- **Título de função:**
    - Nome e ação da função deve ser autoexplicativo.
- **Docstring em códigos não-públicos:**
    - Desnecessário para código de consumo interno, formalidade a mais só atrapalha.