Commit Message sao cho chuẩn?

Speaker: Lê Văn Việt

1. Thói quen commit!

2. Vì sao nên sử dụng Conventional Commits?

- Đầu tiên, và quan trọng nhất: là nó giúp cho những người làm cùng dự án có thể dễ dàng nắm bắt được công việc chỉ bằng cách nhìn vào danh sách commit (chức năng nào đã làm rồi, bug nào đã được fix), từ đó có thể contribute một cách chính xác, nhanh chóng.
- Giúp có thể tự động đánh số phiên bản và sinh CHANGELOGs.
- Thông báo giữa các thành viên trong nhóm, các bên liên quan hoặc public cho cộng đồng về thay đổi của commit
- ...

3. Cấu trúc của một commit message

- Commit message theo chuẩn conventional commits sẽ có cấu trúc:

```
# [] là phần tùy chọn, có thể có hoặc không, <> là phần bắt buộc
<type>[optional scope]: <description>

[optional body]

[optional footer(s)]
```

3. Cấu trúc của một commit message

- `type`: ngoài 2 loại chính là `fix` và `feat` ra thì cũng có thêm một số loại khác
  - **fix**: pull request này thực hiện fix bug của dự án
  - **feat** (feature): pull request này thực hiện một chức năng mới của dự án
  - **refactor**: pull request này thực hiện refactor lại code hiện tại của dự án (refactor hiểu đơn giản là việc "làm sạch" code, loại bỏ code smells, mà không làm thay đổi chức năng hiện có)
  - **docs**: pull request này thực hiện thêm/sửa đổi document của dự án
  - **style**: pull request này thực hiện thay đổi UI của dự án mà không ảnh hưởng đến logic.

3. Cấu trúc của một commit message

- `type`: ngoài 2 loại chính là `fix` và `feat` ra thì cũng có thêm một số loại khác
  - **perf**: pull request này thực hiện cải thiện hiệu năng của dự án (VD: loại bỏ duplicate query, ...)
  - **vendor**: pull request này thực hiện cập nhật phiên bản cho các packages, dependencies mà dự án đang sử dụng.
  - **chore**: từ này dịch ra tiếng Việt là việc lặt vặt để chỉ những thay đổi không đáng kể trong code (ví dụ như thay đổi text chẳng hạn).

3. Cấu trúc của một commit message

- `scope`:
  - **KHÔNG BẮT BUỘC**
  - thay đổi code trong pull request này sẽ có phạm vi ảnh hưởng đến đâu (là danh từ)
  - đứng ngay sau `<type>` và được đặt trong dấu ngoặc đơn.

```
# scope ở đây là lang
feat(lang): add polish language
```

3. Cấu trúc của một commit message

- **`description`**: mô tả khái quát, ngắn gọn về những thay đổi được thực hiện trong pull request.

```
# mô tả ngắn gọn pull request này sửa lỗi chính tả trong CHANGELOG
docs: correct spelling of CHANGELOG
```

3. Cấu trúc của một commit message

- `body`:
  - **KHÔNG BẮT BUỘC**
  - NẾU CÓ, bắt buộc phải cách phần `description` 1 dòng trắng (blank line)
  - Mô tả chi tiết bổ sung cho phần description phía trên về những thay đổi được thực hiện trong pull request
  - Không giới hạn về số dòng và không nhất thiết phải theo 1 format nhất định (free-form).

3. Cấu trúc của một commit message

- `footer(s)`:
  - **KHÔNG BẮT BUỘC**
  - Nằm ở phía sau body (nếu có body) và phân cách bằng một dòng trắng (blank line)
  - Chứa các thông thông tin mở rộng của pull request ví dụ như là danh sách người review, link/id của các pull request có liên quan.
  - Nếu có nhiều footer thì mỗi footer phải chứa một word token là `:<space>` hoặc `<space>#`
  - Để phân biệt với phần body thì các từ ghép dùng làm token (ngoại trừ `BREAKING CHANGE`) sẽ ngăn cách nhau bằng dấu `-` thay vì `<space>` (Reviewed by => Reviewed-by)

3. Cấu trúc của một commit message

- `footer(s)`:

```
# 1 commit message gồm body(gồm nhiều đoạn) và nhiều footer
fix: correct minor typos in code #type: description

see the issue for details #body paragraph 1

on typos fixed. #body paragraph 2

Reviewed-by: Z #footer 1 sử dụng :<space>
Refs #133 #footer 2 sử dụng <space>#
```

3. Cấu trúc của một commit message

- `BREAKING CHANGE` (quan trọng nên cần viết IN HOA và **bôi đậm**):
  - Nếu sự thay đổi code trong pull request của bạn làm thay đổi lớn khiến cho nó không còn tương thích với phiên bản trước nữa thì bạn sẽ phải đánh dấu pull request này là `BREAKING CHANGE`.
  - Có 2 cách để đánh dấu 1 pull request là `BREAKING CHANGE`: đó là đặt từ khóa `BREAKING CHANGE:` vào đầu footer hoặc là `!` liền sau `type/scope` hoặc sử dụng cả 2.

3. Cấu trúc của một commit message

```
# sử dụng từ khóa `BREAKING CHANGE`
feat: allow provided config object to extend other configs
<blank line>
BREAKING CHANGE:<space>`extends` key in config file is now used for extending other config files

# sử dụng ! liền sau type trong commit
refactor!: drop support for Node 6

# sử dụng kết hợp cả 2
refactor!: drop support for Node 6
<blank line>
BREAKING CHANGE:<space>refactor to use JavaScript features not available in Node 6.
```

Q&A...
Hỏi và đáp

Bài kế tiếp?