좋은 커밋 메세지를 위한 5 단계

출처 : https://www.freecodecamp.org/news/how-to-write-better-git-commit-messages

제안된 지침을 요약하면:

1. 대문자, 구두점(:) : 첫 단어는 대문자로 하고, 구두점으로 끝내지 않습니다.
   기존 커밋을 사용하는 경우 모두 소문자를 사용해야 합니다.

2. 분위기 : 제목줄에 명령형 분위기를 사용 하십시오.
   Example – Add fix: 다크모드 토글 상태
   명령적 분위기는 명령이나 요청을 하는 어조를 나타냅니다.

3. 커밋 유형: 커밋 유형을 지정합니다. 변경 사항을 설명하는 일관된 단어 집합을 사용하는 것이 권장되며 훨씬 더 유용할 수 있습니다.
   Example: Bugfix, Update, Refactor, Bump, and so on.
   추가 정보는 아래의 기존 커밋 섹션을 참조하세요.

4. 길이: 첫 번째 줄은 이상적으로는 50자 이하여야 하며, 본문은 72자로 제한되어야 합니다.

5. 콘텐츠: 직관적으로, 이 문장에서 채우기 단어와 구를 제거하려고 노력하십시오.
   (examples: 그러나, 아마도, 내 생각에, 거의).
   기자처럼 생각하십시오.

기존 커밋

좋은 커밋 메시지의 기본적인 커밋 구조를 살펴보았으니, 견고한 커밋 메시지 생성에 대한 세부 정보를 제공하기 위해 기존 커밋을 소개하고 싶습니다.
D2iQ에서는 엔지니어링 팀 사이에서 좋은 관행인 Conventional Commit을 사용합니다.

기존 커밋은 다음과 같이 일관된 커밋 메시지 구조를 공식화하는 규칙 집합을 제공하는 형식 지정 규칙입니다:

```
<type>[optional scope]: <description>

[optional body]

[optional footer(s)]
```

커밋 유형에는 다음이 포함될 수 있습니다:

feat – 변경 사항과 함께 새로운 기능이 도입되었습니다.
fix – 버그 수정이 발생했습니다.
chore – 수정 사항 또는 기능과 관련이 없고 src 또는 테스트 파일을 수정하지 않는 변경 사항 (for example 종속성 업데이트)
refactor – 버그를 수정하거나 기능을 추가하지 않는 리팩토링된 코드
docs – README 또는 기타 markdown file과 같은 문서 업데이트
style – 공백, 세미콜론 누락 등과 같은 코드 형식 지정과 관련된 코드의 의미에 영향을 주지 않는 변경 사항.
test – 새로운 테스트 또는 수정된 이전 테스트 포함
perf – 성능 향상
ci – 지속적인 통합 관련
build – 빌드 시스템 또는 외부 종속성에 영향을 주는 변경 사항
revert – 이전 커밋을 되돌립니다.

Commit Message 비교

다음 메시지를 검토하고 각 범주에서 얼마나 많은 제안된 지침을 확인하는지 확인하십시오.

Good

Feat: 이미지에 대한 지연 로드 구현으로 성능 향상
```
feat: improve performance with lazy load implementation for images
```

Chore: npm 종속성을 최신 버전으로 업데이트
```
chore: update npm dependency to latest version
```

Fix: 사용자가 구독 양식을 제출하지 못하게 하는 버그 수정
```
Fix bug preventing users from submitting the subscribe form
```

Update: 클라이언트 요청에 따라 바닥글 본문 내 잘못된 클라이언트 전화번호 업데이트
```
Update incorrect client phone number within footer body per client request
```

Bad

fixed bug on landing page
Changed style
oops
I think I fixed it this time?
empty commit messages