Best Conventions for Commit Messages

Why Commit Message Conventions Matter

A well-structured commit history improves collaboration, debugging, and version tracking. Clear commit messages help team members and future developers understand changes quickly.

General Best Practices

- Keep messages concise yet descriptive.
- Use imperative mood (e.g., "Fix bug" instead of "Fixed bug").
- Separate subject and body with a blank line.
- Limit the subject line to **50 characters** (or 72 max).
- Wrap the body text at **72 characters per line**.
- Reference issue numbers when relevant (e.g., `Fix #123`).
- Use a consistent style for clarity and automation compatibility.

Recommended Commit Message Format

```
<type>(<scope>): <subject>

<body>

<footer>
```

1. Type (Describes the purpose of the commit)

Common types include:

- **feat**: A new feature
- **fix**: A bug fix
- **docs**: Documentation updates
- **style**: Formatting, white-space, missing semicolons, etc.
- **refactor**: Code restructuring without functionality changes
- **perf**: Performance improvements
- **test**: Adding or updating tests
- **chore**: Maintenance tasks, build scripts, etc.

2. Scope (Optional but helpful)

Defines what part of the codebase the change affects. Examples:

- `feat(auth): add JWT authentication`
- `fix(ui): resolve button alignment issue`

3. **Subject** (Short summary of the change)

- Use **imperative mood** (e.g., "Add login form" instead of "Added login form").
- **Capitalize** the first word.
- **No period** at the end.

4. **Body** (Detailed explanation, optional)

- Explain **why** the change was needed and **how** it was done.
- Provide context if needed.
- Mention trade-offs, limitations, or related tickets/issues.

Example:

```
feat(api): add user authentication

Implemented JWT-based authentication with login and registration endpoints.
Updated middleware to handle token validation. Fixes #42.
```

5. **Footer** (Metadata, optional)

- Used for references like closing issues (`Fixes #123`) or breaking changes.
- Example:

```
BREAKING CHANGE: Updated API endpoint from /v1/users to /v2/users.
```