GITHUB Cheat Sheet

- new repository

(필수) **Repository name** : 저장소의 이름, 사용자가 원하는 이름 입력

(선택) **Description** : 저장소에 대한 설명(주석)

(필수) **Public, Private** : 무료 계정인 경우 Public만 사용 가능, 개인적인 저장소를 원한다면 결제 후 Private 사용

(선택) **Initialize this repository a README** : 저장소 생성과 함께 README 파일 설치

(선택) **Add .gitignore** : 깃허브로 올리지 않을 파일 선택

(선택) **Add a license** : 저장소 생성과 함께 라이선스 파일의 설치가 자동 수행

- Wiki

Edit message : wiki의 커밋 메시지

Pages : 구성된 모든 md 파일을 순서에 상관 없이 보여줌

- Issue

새롭게 추가될 기능, 개선 해야할 기능, 버그 등

branch의 생성 목적을 issue라고 생각하기

Title, Description 입력

Description 마크다운으로 작성 가능

Assignees : 위탁인, 검토인으로 이슈를 관리할 사람, 봐줬으면 하는 사람 지정

Labels : 해당 이슈에 해당하는 Tags 지정. 개발분야(server/client), 목적, 중요도 등

Projects : 해당 이슈를 담을 Project 지정

Milestone : '이정표' 라는 뜻, 해당 이슈를 담을 마일스톤 선택.

한 이슈당 최대 하나의 마일스톤만 지정

- Label

이슈, PR이 어떤 특징을 갖고 있는지, 어떠한 상태인지 태깅하여 라벨링

- Milestone

프로젝트에서 중요한 이벤트를 표시하는 기준점, 프로젝트의 진행도를 파악하기 위해 사용

Title, Description 입력

(선택) **Due date (기간)** 입력

- Pull Request

1. fork

타겟 프로젝트의 저장소를 자신의 저장소로 fork

2. clone, remote 설정

fork로 생성한 respository에서 clone or download 버튼을 누르고 표시된 URL 복사

$ git clone [URL]

3. branch 생성

$ git checkout -b [branch]

Sidebar : '_Sidebar.md' 파일을 이용해서 목차 생성

파일을 생성하고 _Sidebar.md 파일에 연결고리를 만들어 네이게이트를 할 수 있도록 관리

Footer : '_Footer.md' 모든 페이지 하단에 공통적으로 처리할 수 있는 문구

(다국어정보 및 저작권 등)

Page History : 변경한 사람, 변경 날짜, 커밋 메시지 확인 (커밋 이력 확인)

Revert Changes : 수정 전의 위키 문서로 되돌리기

이슈명, PR명이 너무 길어지면 가독성이 떨어지므로 공통적이면서 태그로 관리하기 좋은 라벨로 표현하면 좋음

커스터마이징 가능

Label name 입력

(선택) **Description** 입력

Color 선택

4. 수정 작업 후 add, commit, push

$ git push origin [branch]

5. Pull Request 생성

push 완료 후 자신의 github 저장소에서 **compare & pull request** 버튼이 활성화 되어있는 걸 확인할 수 있음

버튼을 눌러 PR 생성

6. Merge Pull Request

PR을 받은 관리자는 코드 변경내역 확인 후 merge 여부 결정

7. Merge 이후 동기화 및 branch 삭제

merge가 완료되면 로컬 코드와 원본 코드를 병합하고 최신 상태를 유지하기 위해 동기화

$ git pull post [remote]

$ git branch -d [branch]