git, github

Connect to Frontend

Revert Everything!

Rename

- Worst
  `$ mv server.py main.py` -> deleted, new file
- Best
  `$ git mv server.py main.py` -> renamed

> 파일의 history를 남기기 위해서는 삭제 후 생성이 아닌 이름바꾸기로 추적

Undoing

`$ git checkout -- .` or `$ git checkout -- {filename}`

Unstaging

```
$ git reset HEAD {filename}
```

Unstaging and Remove

```
$ git rm -f {filename}
```

Edit latest commit

```
$ git commit --amend
```

Edit prior commit

```
$ git rebase -i <commit>
```

abort rebase

```
$ git rebase --abort
```

Complete rebase

```
$ git rebase --continue
```

Reset Commit

Worst case: Reset

ex) 직전 3개의 commit을 삭제한 후, remote에 강제 push

```
$ git reset --hard HEAD~3
$ git push -f origin <branch>
```

- 협업 시 다른 cloned repo에 존재하던 commit log로 인해 파일이 살아나거나, 과거 이력이 깔끔히 사라져 commit log tracking이 힘들어짐.
- solution: 잘못한 이력도 commit으로 박제하고 수정한 이력을 남기자!

Best case: Revert

ex) 현재 HEAD에서 직전의 3개의 commit을 순서대로 거슬러 올라가 해당 내역에 대해 commit, push 수행

```
$ git revert --no-commit HEAD~3..

$ git commit

$ git push origin <branch>
```

- 잘못하기 전 과거로 돌아가 최신을 유지하면서 되돌렸다는 이력을 commit으로 남겨 모든 팀원이 이 사항을 공유하고 주지시킬 수 있음.
- commit을 따로 안할땐 `--no-edit`
- merge commit을 되돌릴 땐 `-m` (`$git revert -m {1 or 2} {merge commit id}`)

Simple project with git

Stopwatch

- 시, 분, 초, 밀리초 가 표현되어야 한다.
- 00:00:00.0 의 형태
- lap 버튼으로 lap 기능을 구현해야 한다.(특정시점 저장 기능)
- 새로고침 시 모든 정보가 리셋되어야 한다
- start 버튼은 동작 후 stop/resume 버튼으로 바뀌며, 누를 때마다 정지/계속 되어야 한다.
- reset 버튼을 누르면 모든 상태가 초기화 된다.(사실상 새로고침)
- 해당 페이지의 디자인은 자유롭게 구성한다.
- 추가기능 구현 환영

Process

1. 팀원 구성 후 요구사항 분석 회의 진행
2. 개인 계정의 repo를 생성하여 진행(collaboration)
3. 각자의 역할은 기술단위가 아닌 기능단위로 정함
4. github projects로 backlog를 만들고 구현을 시작함
5. 모든 대화는 대면이 아닌 projects와 커밋 속 conversation 으로 진행
6. git flow 적극 활용

- 기능의 동작이나 개발완료 유무는 고려사항이 아님!!!!!