3차시. 타임머신 기록 생성: 커밋, 이력 기록과 메시지 규칙

Day 1 · 11:00-11:45 (수업 45분) / 11:45-12:00 (휴식 15분)
점심시간 안내: 12:00-13:00 (60분 고정 휴식)
저장소: `portfolio-web`
선수 학습: 2차시 파일 추적과 staged 상태

이 차시에서 하는 일

- 스테이징 영역에 올려둔 파일들을 `git commit`으로 확실한 스냅샷 기록으로 남긴다.
- 커밋 해시(SHA-1 Hash)의 고유 식별자 의미와 직관적인 커밋 메시지 규칙을 익힌다.
- `git log --oneline`과 `git diff`로 프로젝트의 변경 이력을 조회한다.
- vi 에디터가 열렸을 때 당황하지 않고 안전하게 저장하고 빠져나오는 법을 체득한다.
- 실수로 스테이징에 올린 파일을 안전하게 내리는 `git restore --staged`를 실습한다.

의미 있는 작업 단위 기록의 필요성

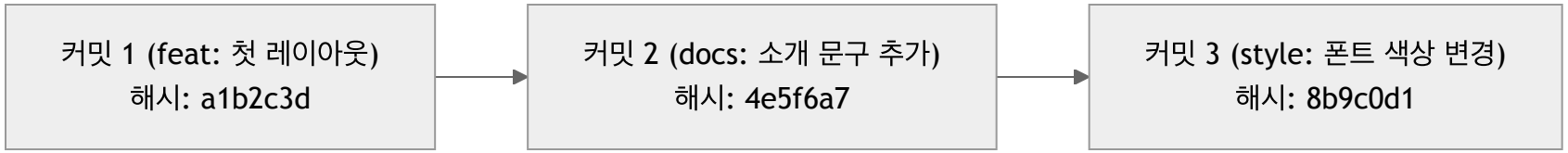
웹페이지에 새로운 섹션을 추가하거나 디자인 스타일을 다듬을 때, 작업 중간중간 확실한 체크포인트를 남겨두어야 합니다. 그래야 기능이 고장 났을 때 방금 전 정상 동작하던 시점으로 언제든 되돌아갈 수 있습니다.

Git에서 커밋(Commit)은 단순한 파일 저장이 아닙니다. "특정 시점에 의미 있는 작업 단위가 완성되었음을 선언하고, 그 순간의 전체 프로젝트 모습을 영구적인 스냅샷으로 남기는 행위"입니다.

또한 내가 어떤 작업을 왜 수행했는지 한눈에 알 수 있도록 설명해 주는 **커밋 메시지(Commit Message)**를 함께 기록해야 협업하는 동료와 미래의 내가 이력을 쉽게 추적할 수 있습니다.

커밋 스냅샷과 해시 식별자의 원리

Git 커밋은 다음과 같은 특징을 지닙니다:



커밋 스냅샷과 해시 식별자의 원리 · 계속

1. **커밋 해시 (Commit Hash)**
   - 각 커밋은 작성자, 작성 시간, 파일 내용 등을 종합하여 만들어진 40자리 16진수 문자열(SHA-1)을 가집니다.
   - 전 세계에서 유일하게 식별되는 지문과 같으며, 보통 앞의 7자리(예: `a1b2c3d`)만으로 커밋을 고유하게 구분합니다.

커밋 스냅샷과 해시 식별자의 원리 · 계속

2. **커밋 메시지 컨벤션 (규칙)**
    - 좋은 커밋 메시지는 첫머리에 변경의 성격을 드러내는 접두사를 붙입니다:
        - `feat:` 새로운 기능이나 화면 구성 요소 추가
        - `docs:` 문서 수정, HTML 본문 문구 수정
        - `style:` CSS 스타일 변경, 여백이나 색상 조정 (코드 로직 변경 없음)
        - `fix:` 버그나 깨진 마크업 수정
        - `refactor:` 기능 변경 없는 코드 구조 개선
        - `chore:` 설정 파일, 패키지 관리 등 기타 작업

vi 에디터와 탈출법

만약 `git commit`을 실행할 때 `-m "메시지"` 옵션을 빠뜨리면 터미널 내에서 기본 텍스트 편집기인 vi(또는 vim)가 열립니다. 마우스가 동작하지 않아 당황하기 쉽지만, 다음 절차를 거치면 안전하게 빠져나올 수 있습니다:

- **저장하고 닫기**: `Esc` 키를 누른 뒤 `:wq`를 입력하고 `Enter`

vi 에디터와 탈출법 · 계속

- **저장하지 않고 취소하기**: `Esc` 키를 누른 뒤 `:q!`를 입력하고 `Enter`

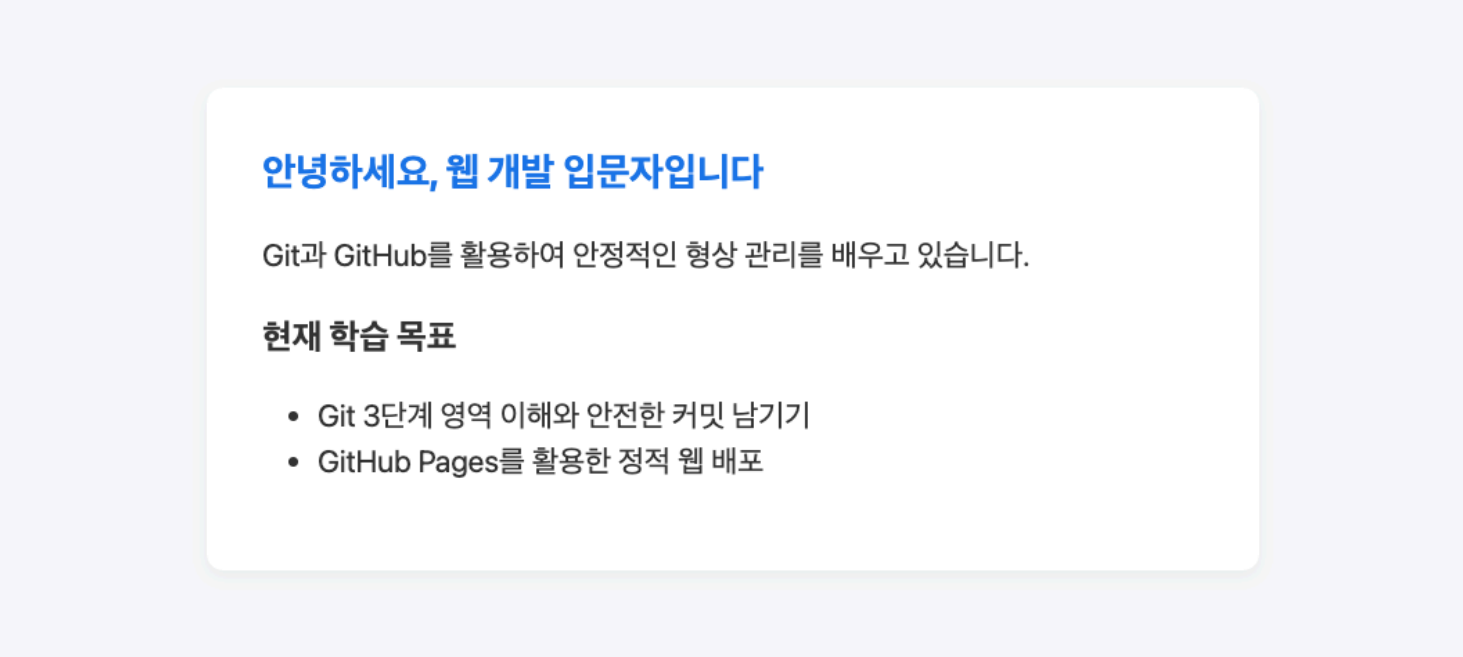
웹페이지 수정 코드

3차시에서 소개 문구를 보강하고 스타일을 다듬는 수정 예시입니다.

HTML 수정 (index.html)

`<main>` 태그 안에 학습 목표 목록을 추가합니다.

HTML 수정 (index.html) · 계속



HTML/CSS 실제 브라우저 렌더 결과

[전체 예제 코드 · portfolio-1.html]

따라하기

1. 2차시에서 스테이징 영역에 올려둔 파일들을 첫 번째 커밋으로 기록합니다: `git commit -m "feat: 첫 웹 페이지 레이아웃 및 스타일 추가"`
2. `git log --oneline`을 실행하여 첫 번째 커밋의 해시(앞 7자리)와 메시지가 출력되는지 확인합니다.
3. VS Code에서 `index.html`을 열고 위의 학습 목표 섹션을 추가한 뒤 저장합니다.
4. `git status`를 입력해 `index.html`이 빨간색 `modified` 상태로 감지되는지 확인합니다.
5. `git diff`를 실행하여 실제로 어떤 HTML 코드가 추가되었는지 확인합니다 (초록색 + 줄).
6. `git add index.html`로 스테이징 영역에 올립니다.
7. 실수로 올린 상황을 가정하고, 스테이징을 안전하게 해제해 봅니다: `git restore --staged index.html`
8. `git status`로 다시 `Changes not staged for commit:` 상태로 내려갔음을 확인합니다.

따라하기 · 계속

9. 다시 `git add index.html`을 실행하고 두 번째 커밋을 작성합니다: `git commit -m "docs: 학습 목표 섹션 및 문구 추가"`
10. `git log --oneline`으로 누적된 2개의 커밋 이력을 확인합니다.
11. 깨끗한 작업 트리 상태(`working tree clean`)를 확인하고, 12:00부터 13:00까지 점심 휴식을 취합니다.

Git 명령

```
git commit -m "feat: 첫 웹페이지 레이아웃 및 스타일 추가"
git log --oneline
git diff
```

Git 명령 · 계속

Git 실습 실행 결과

```
$ git log --oneline -3
bf2ce20 docs: add project guide
7616b25 feat: add portfolio page
```

격리된 실습 저장소의 실제 실행 출력 (원격 배포 화면 아님)

Git 명령 · 계속

[전체 예제 코드 · git-commands-1.sh]

선택 참고 명령

```
git log -p -2
git log --oneline --graph --all
```

에이전트 요청 예시

> 커밋 메시지 컨벤션 feat docs style fix 차이 초보자 눈높이 요약
> vi 에디터 진입 시 저장 종료 및 취소 종료 키 입력 절차 안내

확인법

- [ ] `git log --oneline` 입력 시 2개의 커밋 해시와 메시지가 시간 역순으로 표시된다.
- [ ] `git status` 입력 시 `nothing to commit, working tree clean` 문구가 출력된다.
- [ ] `git diff`를 통해 수정한 라인이 초록색으로 표시되는 원리를 이해한다.
- [ ] vi 에디터 진입 시 `Esc` 후 `:wq` 또는 `:q!`로 빠져나올 수 있다.

오류 복구

증상	원인	대응
vi 에디터가 열려 글자가 안 써짐	`git commit` 시 `-m` 옵션 누락	`Esc`를 누르고 `:wq`를 입력해 저장 종료하거나, `:q!`로 취소한 뒤 `git commit -m "메시지"`로 다시 실행한다.
커밋 메시지에 오타 발생	작성 실수	직전 커밋 메시지는 4차시에서 배울 `git commit --amend`로 안전하게 수정할 수 있다.
`git diff`에 아무것도 안 나옴	파일이 이미 `git add` 되었음	`git diff`는 작업 공간의 변경을 보며, 스테이징된 변경은 `git diff --staged`로 확인한다.

안전 주의

- 커밋 메시지는 "수정 완료", "test", "asdf"처럼 모호하게 적지 말고, 무엇이 바뀌었는지 명확하게 기록합니다.

안전 주의 · 계속

- 점심시간(12:00-13:00)에는 작업 중인 내용을 반드시 커밋해 두거나 저장하여 작업 트리를 안전하게 정리해 둡니다.
- 상세한 Git 명령어 옵션은 부록 2. Git 명령과 옵션 일람을 참조합니다.
- 추가 심화 실습은 보충안 심화, 복습 퀴즈는 보충안 퀴즈, 에이전트 주도 Git은 보충안 에이전트에서 다룹니다.