웹프로그래밍

- Web Programming -

2. 웹개발환경 구축

정보보호학과 이병천 교수

1. 웹프로그래밍 개발환경 구축

- 1. VSCode 설치
- 2. Node.js 설치
- 3. Git 설치
- 4. Github.com 회원가입

VSCode 설치

- VSCode
  - 마이크로소프트에서 개발한 통합개발환경
  - https://code.visualstudio.com/
  - 현재 버전: 1.76 (2023.3), 자동 업데이트
  - 개발자들이 가장 애용하는 에디터

- 유용한 확장기능 설치 활용
  - Korean language pack: 한국어 UI 환경
  - Live Server: 개발용 로컬 웹서버
  - Prettier: 코드 자동 포맷 정리
  - Indent-rainbow: 들여쓰기를 색깔로 구별 표시
  - ES7+React/Redux/React-native snippets: 자바스크립트 문법 체크, 코드 추천
  - 유용한 vscode 확장프로그램 추천 검색 활용

VSCode 설치

- VSCode 설정
  - Default formatter : prettier
    - 코드의 포맷을 prettier로 관리
  - Format on save: on 설정
    - 파일 저장시 default formatter가 자동 포맷
  - Prettier: single quote 설정
    - 따옴표 종류를 지정
  - Prettier: semi 설정 해제
    - 자바스크립트 코드를 semi-colon으로 끝내는 것을 해제
    - Semi-colon을 사용하지 않는 것이 깨끗한 코드로 관리
  - Editor: Word Wrap 설정
    - 한 줄을 넘는 코드는 자동으로 다음 줄로 넘겨서 표시

Node.js 설치

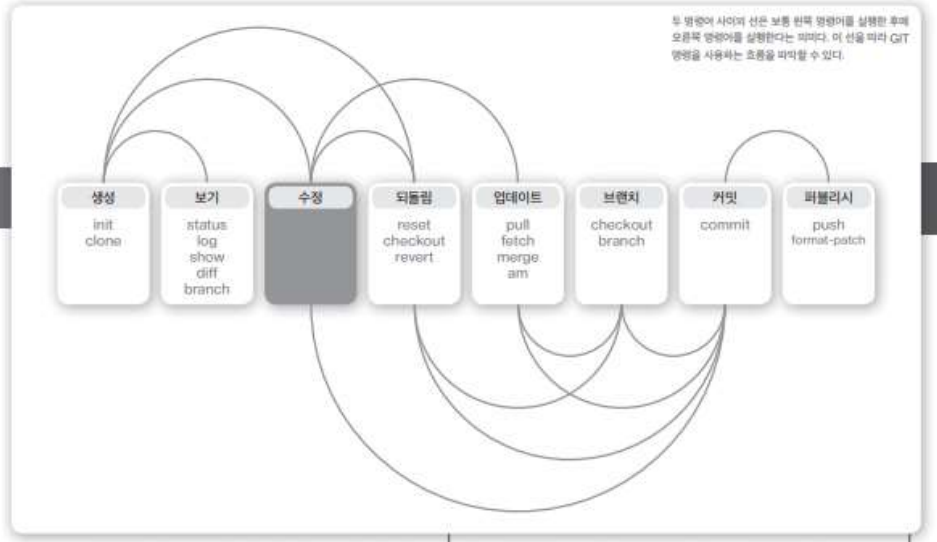
- Node.js
  - Node.js® is an open-source, cross-platform JavaScript runtime environment.
  - Node.js 기반의 웹서버 구축에도 활용
  - https://nodejs.org/en/
  - 최신 버전: 18.14.2 LTS, 19.7.0 Current (2023.3)
  - 자바스크립트 프로그램을 실행하고, 외부 패키지를 설치하여 사용하는데 필요

Git 설치

- Git
  - source 관리를 위한 분산 버전 관리 시스템
  - 최초로 리눅스 토발즈가 리눅스 커널 개발에 이용하려고 개발
  - 프로그램 다운로드 및 설치 https://git-scm.com/downloads
  - 현재 버전: 2.37.3 (2022.8.30)
  - Git GUI clients

GIT 작업 흐름과 명령어

도움말: git "명령어" --help
Git의 글로벌 설정은 $HOME/.gitconfig에 저장 (git config --help)



생성

새 저장소 생성하기
```
cd ~/projects/myproject
git init
git add .
```

기존 저장소 Clone하기
```
git clone ~/existing/repo ~/new/repo
git clone git://host.org/project.git
git clone you@host.org/project.git
```

보기

워킹 디렉터리의 파일 상태 보기
```
git status
```

파일의 변경사항 보기
```
git diff
```

$ID1과 $ID2 사이의 변경사항 보기
```
git diff $id1 $id2
```

커밋 히스토리 보기
```
git log
```

특정 파일의 커밋 히스토리별 변경사항 보기
```
git log -p $file $dir/ec/tory/
```

특정 파일을 누가 언제 고쳤는지 보기
```
git blame $file
```

$ID 커밋 보기
```
git show id
```

$ID 버전의 파일 보기
```
git show $id:$file
```

로컬 브랜치들 보기
```
git branch
```
(* 표시는 현재 브랜치를 나타냄)

범례

$id - 커밋 ID, 브랜치 이름, 태그 이름을 나타냄
$file - 파일 이름
$branch - 브랜치 이름

개념

Git 기본

master : 기본 브랜치
origin : 기본 리모트 저장소
HEAD : 현재 브랜치
HEAD^ : HEAD의 부모
HEAD~4 : HEAD의 부모의 부모의 부모의 부모

되돌림

마지막 커밋 시점으로 되돌리기
```
git reset --hard
```
▲Hard Reset은 되돌릴 수 없음

마지막 커밋 내용을 되돌리고 커밋하기
```
git revert HEAD
```
새로운 커밋 생성

특정 커밋 내용을 되돌리고 커밋하기
```
git revert $id
```
새로운 커밋 생성

마지막 커밋 수정하기
```
git commit -a --amend
```
(잘못 커밋해서 수정하고 싶을 때)

$id 시점의 파일을 꺼내기
```
git checkout $id $file
```

브랜치

브랜치를 Checkout하기
```
git checkout $id
```

$branch1을 $branch2에 Merge하기
```
git checkout $branch2
git merge $branch1
```

새 브랜치 만들기
```
git branch $branch
```

$other와 같은 커밋을 가리키는 브랜치를 새로 만들고 바로 Checkout하기
```
git checkout -b $new_branch $other
```

$branch를 삭제하기
```
git branch -d $branch
```

업데이트

origin에서 최신 데이터를 가져오기
```
git fetch
```
(Merge하지는 않음)

origin에서 최신 데이터를 가져와 Merge하기
```
git pull
```
(Fetch하고 Merge까지 함)

누군가 보낸 패치를 Merge하기
```
git am -3 patch-mbox
```
(충돌이 발생하면 해결 후 git am --resolved)

퍼블리시

현재 모든 수정사항을 커밋하기
```
git commit -a
```

다른 개발자에게 보낼 패치를 작성하기
```
git format-patch origin
```

origin으로 업데이트를 Push하기
```
git push
```

버전이나 마일스톤을 생성하기
```
git tag v1.0
```

유용한 명령어

문제가 발생한 커밋 이진탐색하기
```
git bisect start (이진탐색 시작)
git bisect good $id ($id를 문제 없는 상태로 표시)
git bisect bad $id ($id를 문제 있는 상태로 표시)

git bisect bad/good (현재 상태가 문제가 있는지 없는지 설정)
git bisect visualize (gitk를 실행하여 확인함)
git bisect reset (시작했던 상태로 Checkout 함)
```

저장소의 무결성 검사 및 저장소 청소하기
```
git fsck
git gc --prune
```

워킹 디렉터리에서 'foo()'라는 문자열 검색하기
```
git grep "foo()"
```

Merge 충돌 해결

Merge 충돌 내용 보기
```
git diff (충돌 내용 전체 보기)
git diff --base $file (Merge Base를 기준으로)
git diff --ours $file (현 브랜치를 기준으로)
git diff --theirs $file (Merge할 브랜치를 기준으로)
```

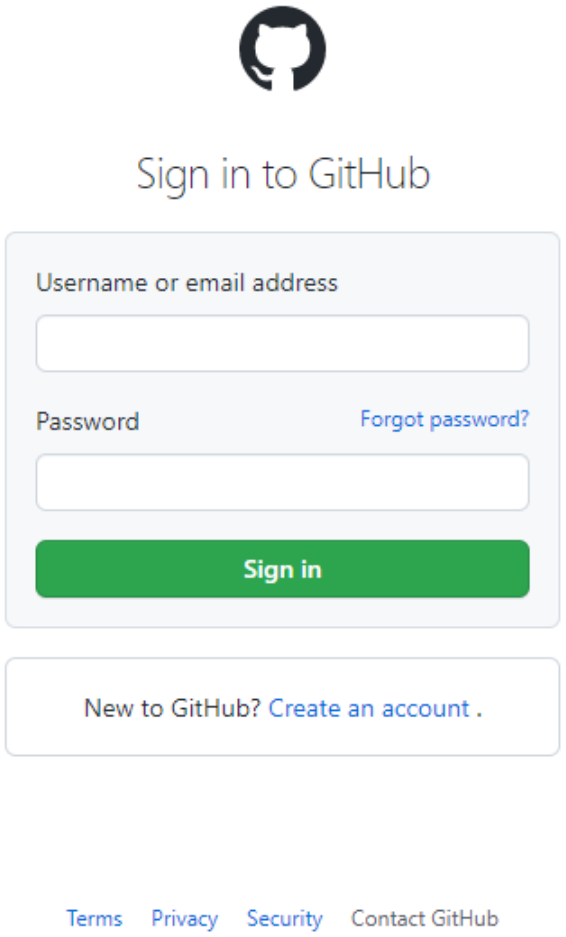
충돌이 생기는 패치 버리기
```
git reset --hard
git rebase --skip
```

충돌 해결 후 Merge 진행하기
```
git add $conflicting_file (충돌 해결한 파일)
git rebase --continue
```

Git 명령어 익히기

Github 회원가입

- Github
  - 소스 관리 포털 서비스
  - https://github.com/
  - 회원가입 (sign up)
  - 로그인 (sign in)



새로운 저장소 만들기

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.

Owner * Repository name *

lbcsultan / crypto-test

Great repository names are short and memorable. Need inspiration? How about stunning-waddle?

Description (optional)

Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.

Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/lbcsultan/crypto-test.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a R

...or create a new repository on the command line

```
echo "# crypto-test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/lbcsultan/crypto-test.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/lbcsultan/crypto-test.git
git branch -M main
git push -u origin main
```

Github 사용을 위한 Git 명령어 흐름

- 0. 프로젝트 폴더로 이동
- 1. 저장소 초기화
  - > git init
- 2. 현재 폴더의 모든 파일들을 git 저장소에 추가
  - > git add .
  - .gitignore에 선언된 폴더, 파일들은 제외
- 3. 커밋, 전송 준비
  - >git commit -am 'first' (구별하기 위한 메시지 추가)
- 4. 브랜치 생성
  - > git branch
- 5. 원격 저장소 지정
  - > git remote
- 6. 푸시, 전송
  - > git push, 실제 코드를 github로 업로드

Vscode에서
Github에 로그인

2. HTML/CSS/Javascript 기초 예제 실습

- HTML5, CSS3, Javascript 예제 실습
  - https://hcjdemo.netlify.app/
  - https://github.com/lbcsultan/hcjdemo

- 주안점
  - 3가지 기술이 함께 사용되는 방법 이해
  - 차분하게 조립해 나가는 능력
  - 따라하기, 변경하기, 설계 및 구현하기

3. 홈페이지 운영하기

- Github에 소스 업로드

- Netlify에서 홈페이지 운영
  - https://www.netlify.com/ 회원가입
  - Github에서 import 하기
  - 홈페이지 주소 설정