Hello,

KDT 웹 개발자 양성 프로젝트

5기!

with

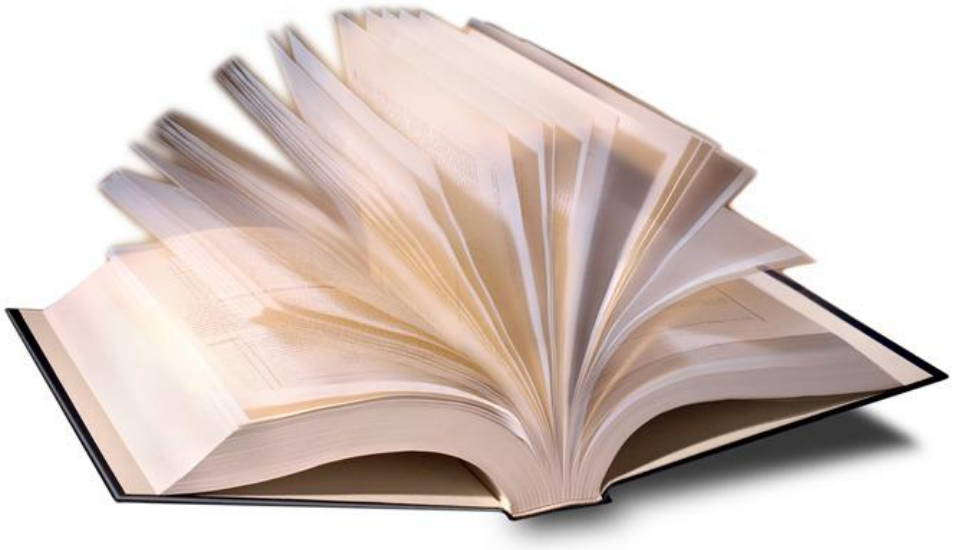
Why?

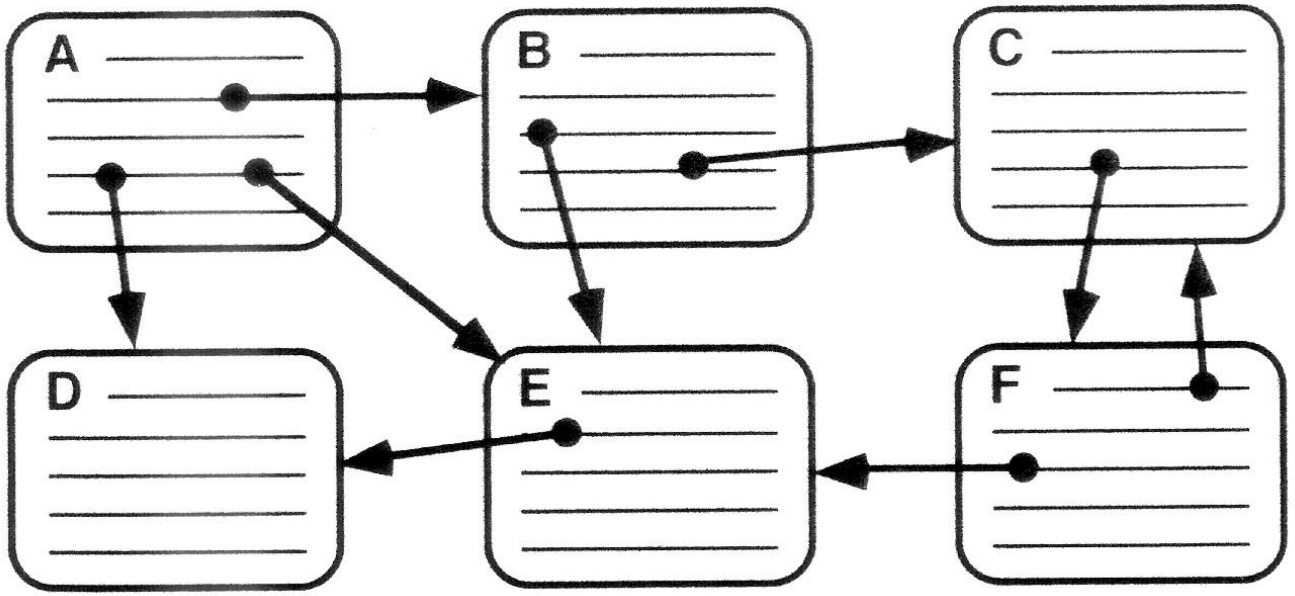

PURPOSE

What?

HyperText Markup Language

- HyperText?





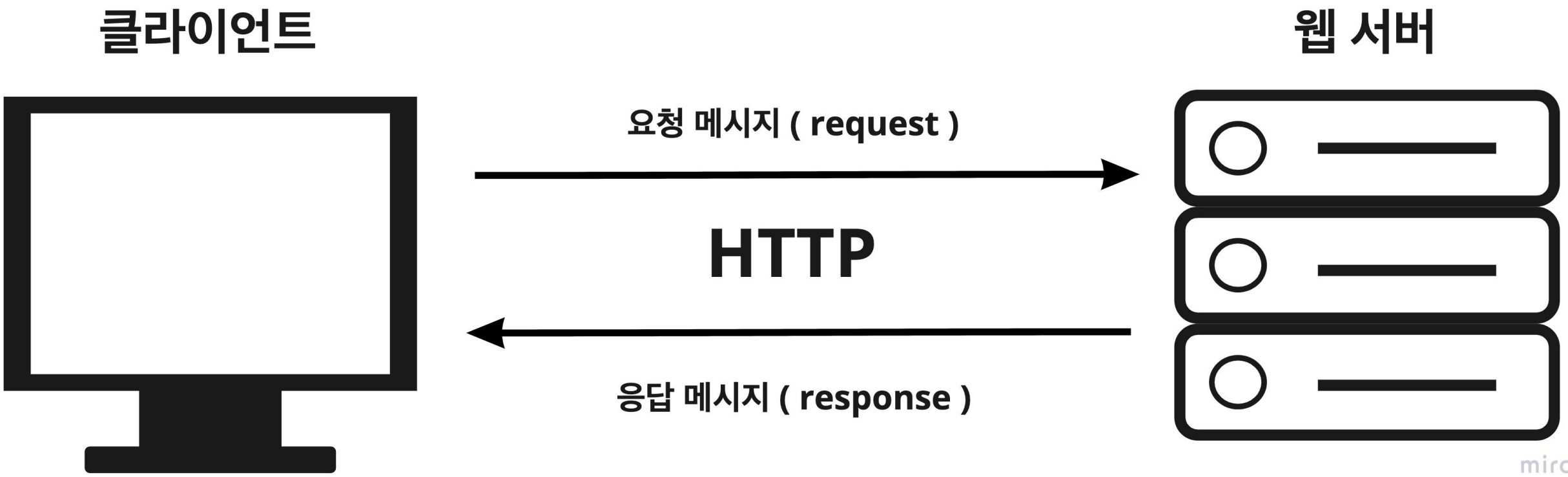
HyperText Markup Language

- Markup

탄생! 인터넷(International Network)

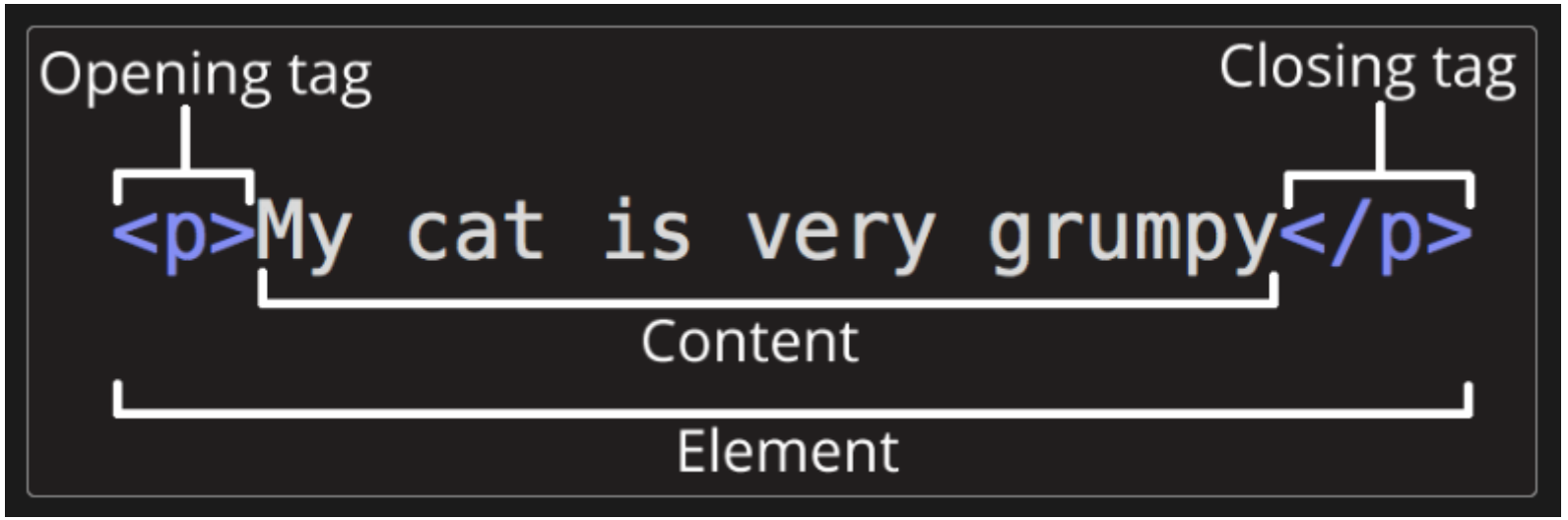
- 1973, '빈튼 서프'와 '밥 간'이 인터네셔널 네트워크 발명
- 1980, 미국 국방성이 연구용으로 'ARPA NET'을 구축
- 1983, 미국 국방성이 군사용 네트워크는 'MIL NET'으로 분리하여 'ARPA NET'은 민간용으로 사용 시작
- 1989, '팀 버너스리'가 월드 와이드 웹의 하이퍼텍스트시스템, URL, HTTP, HTML을 설계 및 개발
- 1993, HTML 1.0 발표
- https://www.w3.org/Protocols/HTTP/AsImplemented.html

HTTP(Hyper Text Transfer Protocol)



HTML
기초 문법

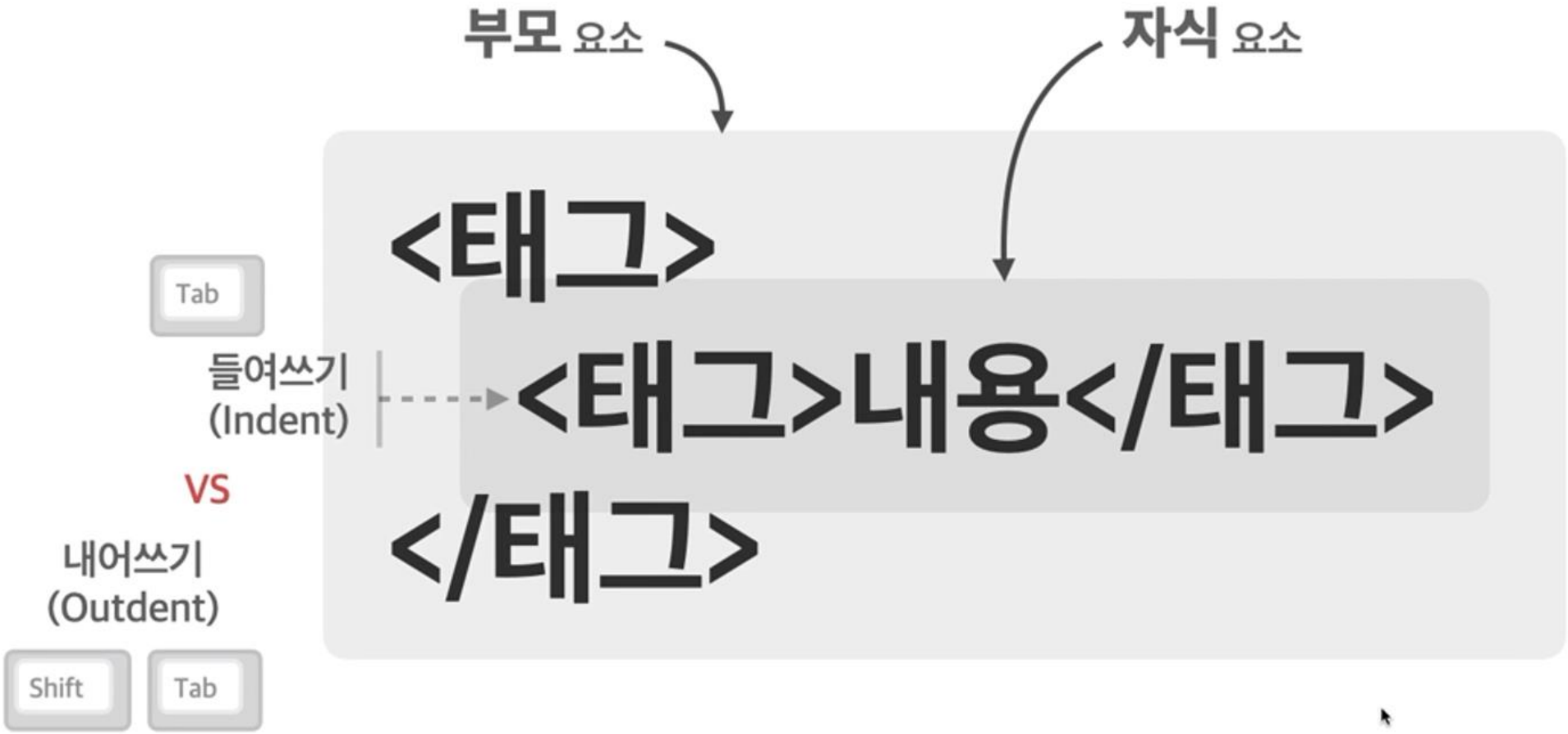
HTML 기본 문법, 요소(Element)



- HTML 요소(Element)는 시작 태그(Opening tag)와 종료 태그(Closing tag) 그리고 태그 사이에 위치한 내용(Content)로 구성

중첩

HTML 기본 문법, 중첩(Nested)



빈 요소

편리함!

HTML 1/2/3/4/5

<태그>

VS

<태그 />

안전함!

XHTML / HTML5

주석

HTML 기본 문법, 주석(Comments)

```
1  <!--주석은 컴퓨터가 아닌 사람을 위한 것이며
2      화면에 표시되지 않는다.-->
3  <p>Hello, KDT World!</p>
```

- 주석(comment)는 개발자에게 코드를 설명하기 위해 사용
- 브라우저는 주석을 화면에 표시하지 않는다.

태그의 속성

이미지의 경로

이미지의 이름
(대체 텍스트 / Alternate Text)

```
<img src="./cat.jpg" alt="고양이" />
```

HTML
문서의 구조

HTML 구조 설명

<html>

<head>

<body>

</html>

<head>

기초 코드를 불러와 봅시다!

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body></body>
</html>
```

<body>

<body> 태그

- HTML 문서의 내용을 담는 태그
- 웹페이지를 구성하는 대부분의 요소가 body 태그 내에 기술
- HTML 태그를 사용하여 사용자가 원하는 문서를 작성

<tag>

글자를 다루는 <tag>

이미지 삽입하기

이미지를 넣어 봅시다!

- 이미지를 넣을 때 사용하는 <img> 태그
- 빈 요소, 닫는 태그가 필요하지 않음

이미지의 경로

이미지의 이름
(대체 텍스트 / Alternate Text)

```
<img src="./cat.jpg" alt="고양이" />
```

문서의
영역 구분하기

```
<h1>이젠 정말 코딩 뿐이야</h1>
<p>코딩 열심히 해야지!</p>
<hr />
<img src="study.png" alt="공부를 결심하는 사진" width="300" />
<p>정말 공부 뿐이야</p>
```

```
<div>
  <h1>이젠 정말 코딩 뿐이야</h1>
  <p>코딩 열심히 해야지!</p>
  <hr />
</div>
<div>
  <img src="study.png" alt="공부를 결심하는 사진" width="300" />
  <p>정말 공부 뿐이야</p>
</div>
```

태그의 전역 속성

태그의 전역 속성

- 모든 HTML 태그에서 공통으로 사용할 수 있는 속성
- ID : 요소에 고유한 이름을 부여하는 식별자 속성
- CLASS : 요소를 그룹 별로 묶을 수 있는 식별자 속성
- STYLE : CSS 를 적용할 때 사용하는 속성
- TITLE : 요소의 추가 정보를 제공하는 속성, 이미지에 툴팁 제공.

링크

HTML의 꽃 하이퍼링크, <a>

- 기존 문서나 텍스트의 선형성, 고정성의 제약에서 벗어나 사용자가 원하는 정보를 취득할 수 있는 기능을 제공
- Anchor 의 약자인 <a> 태그 사용
- 속성 값
  - href : Hypertext Reference 의 약자, 이동할 페이지의 링크
  - target : 링크 된 문서를 열었을 때 문서가 열릴 위치 표시
    - _blank(새 탭) / _self(현재 탭)

목록

목록, or

- 순서 없는 목록
- 순서 있는 목록
- 은 속성 사용 가능
  - <ol type="?"> : 말머리 기호 변경(1, A, a, I, i)
    - 1 / A / a / I / i
  - <ol start="?"> : 시작 값 변경
  - <ol reversed> : 역순으로 시작

사용자 입력 받기

선택 메뉴를 만드는, <select>

- 선택 메뉴(드롭 다운)를 만드는 태그!
- <select> : select 폼 생성
  - Name : select 박스의 이름
- <option> : select 폼의 옵션 값 생성
  - Value : 실제적으로 전달 되는 값
  - Selected : 최초에 선택 된 값으로 설정
- <optgroup> : option 을 그룹화
  - Label : optgroup 이름 설정
- Disabled : 옵션은 보이지만 선택을 못하도록 설정

입력 값 받기! <input>

- type
  - Button
  - Text
  - Password
  - Checkbox
  - Radio
  - Date
  - Color


지금 시작합니다

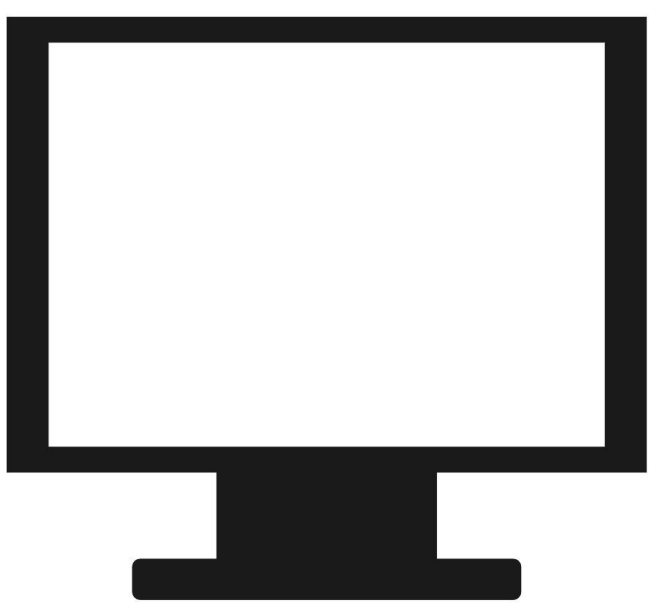
사용자 입력 보내기

사용자 입력을 보내는, <form>

- 사용자가 HTML 문서에서 작성한 내용을 보내기 위해서 사용하는 태그 입니다!
- 그런데 어디로 보내죠!?
- 서버로 보냅니다~!

클라이언트

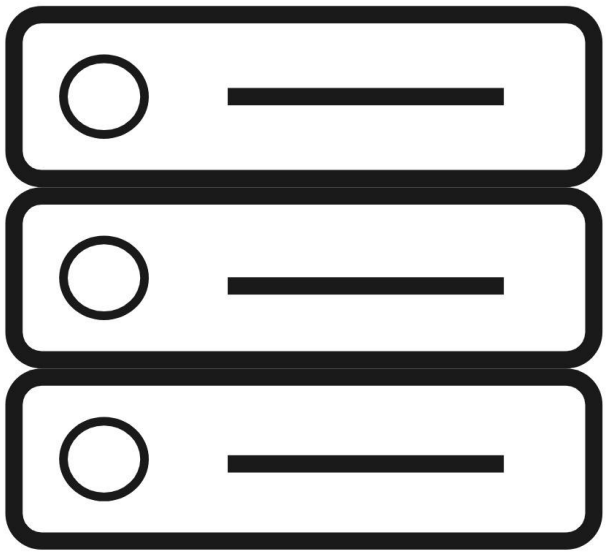
웹 서버



요청 메시지 (request)

HTTP

응답 메시지 (response)



로그인

아이디

패스워드

로그인

보안접속 아이디찾기 | 비밀번호찾기

SNS 계정으로 로그인하기

아직 회원이 아니신가요?

간편 회원가입하기

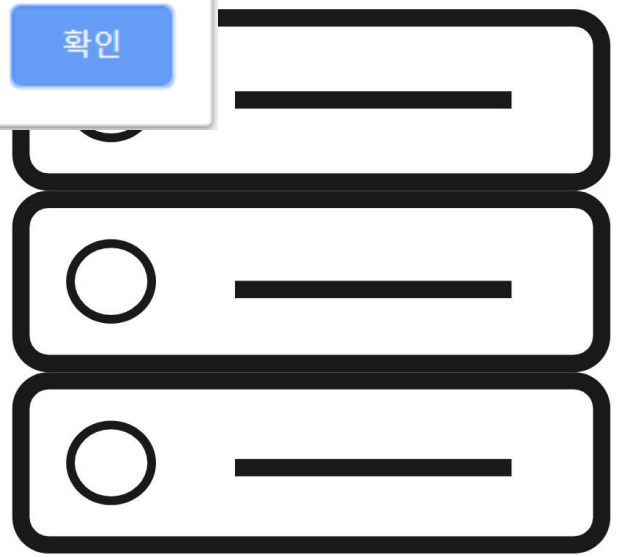
이 페이지 내용:

아이디 혹은 비밀번호가 틀렸습니다.

확인

RESPONSE

웹 서버



miro

사용자 입력을 보내는, <form>

```
<form action="http://3.34.177.57:4000" method="GET">
  <input type="text" name="id" placeholder="아이디" />
  <br />
  <input type="password" name="pw" placeholder="비밀번호" />
  <br />
  <input type="submit" value="제출" />
</form>
```

로그인

아이디

비밀번호

제출

- 사용자가 보내기를 원하는 데이터를 <form> 태그로 감싸고, submit 으로 제출을 하면 됩니다~!
- <input type="submit" /> 을 클릭하면 form 태그에 감싸진 input 요소들의 데이터가 전송 됩니다~!

사용자 입력을 보내는, <form>

- Action : 데이터를 받을 서버의 주소를 입력
- Method : 데이터를 보내는 방식
  - GET : 서버로 부터 정보를 가져만 올 때 / 전달 되는 정보가 주소 창에 보임
  - POST : 서버에게 데이터를 추가, 수정, 삭제 한 후 응답을 받음 / 전달 되는 정보가 주소 창에 보이지 않음


CATS
vs.
DOGS

```
<body>
    <h1>강아지 vs 고양이</h1>
    <form action="http://3.34.177.57:4000" method="GET">
      <input type="text" name="name" />
      <br />
      <select name="choice">
        <option value="dog">강아지</option>
        <option value="cat">고양이</option>
        <option value="whatever">상관 없음</option>
      </select>
      <input type="submit" value="제출" />
    </form>
  </body>
```

```
데이터 통신 서버가 4000에서 작동 중입니다!
{ name: '이효석', choice: 'dog' }
```

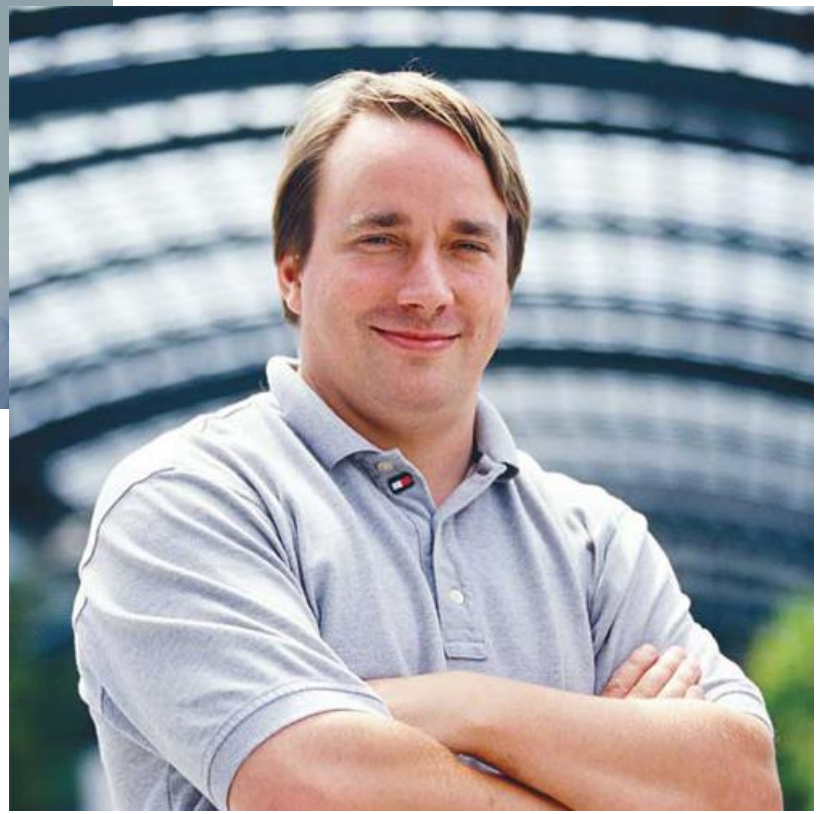
git

git

Git(깃)은 컴퓨터 파일의 변경사항을 추적하고
여러 사용자들 간에 해당 파일 작업을 조율하기 위한
대표적인 버전 관리 시스템(VCS)입니다.

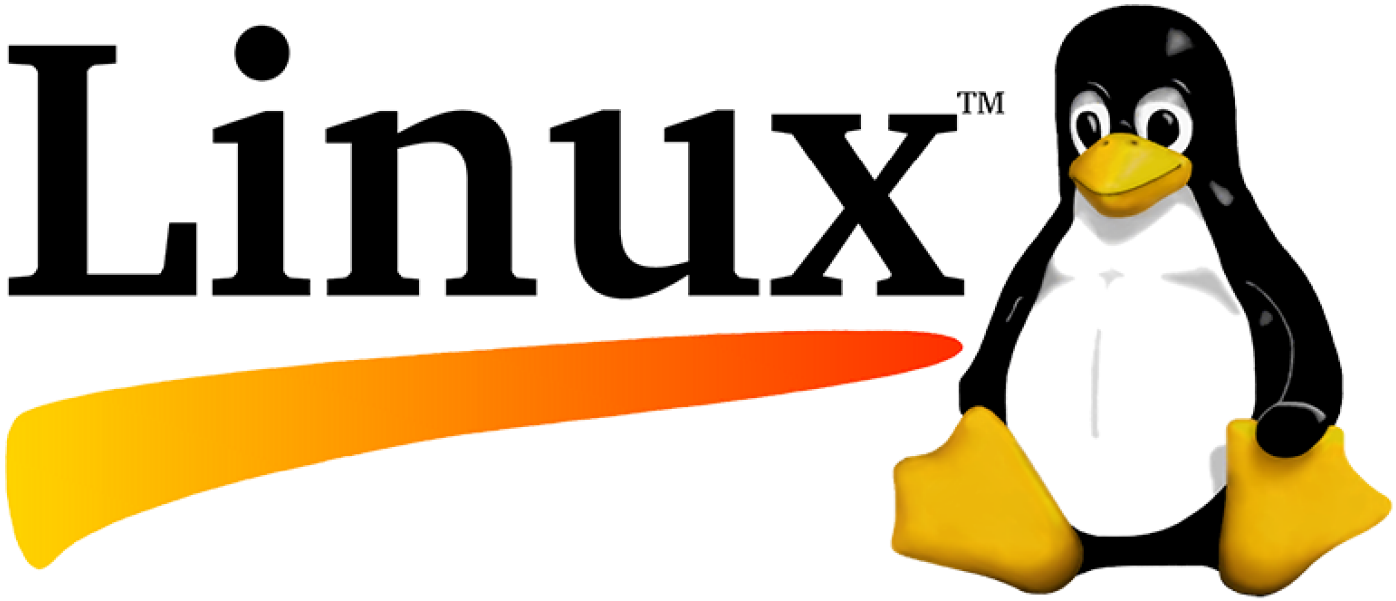
VCS(Version Control System)

세계 3대 개발자!?



Linus Benedict Torvalds

Linux(Linus + unix)



VCS(Version Control System)

Git이 하는 일은!

버전 관리

버전 관리


v_0.1
v_1.0
v_1.0_최종
v_1.3_제발_최종
v_1.5_더 이상은Naver


Commit
Commit
Commit
Commit
Commit

Git의 특징!

- 폴더 단위로 관리! → 하나의 프로젝트 단위로 관리
- Git에 대한 모든 정보는 .git 폴더에 있습니다
  - 일종의 블랙 박스!
- 커밋은 로컬(= 여러분의 컴퓨터)에서만 일어나는 행위입니다!
- 따라서, 깃 허브는 여러분의 로컬에 있는 깃 정보를 온라인에 저장해 주는 서비스를 제공하는 것입니다!
- 깃 허브의 리포지토리는 여러분 로컬의 폴더의 개념과 같아요!

깃 설치하기!

Git, 설치하기(Windows)

https://git-scm.com/

Download for Windows

Click here to download the latest (**2.36.1**) **64-bit** version of **Git for Windows**. This is the most recent maintained build. It was released **24 days ago**, on 2022-05-09.

Other Git for Windows downloads

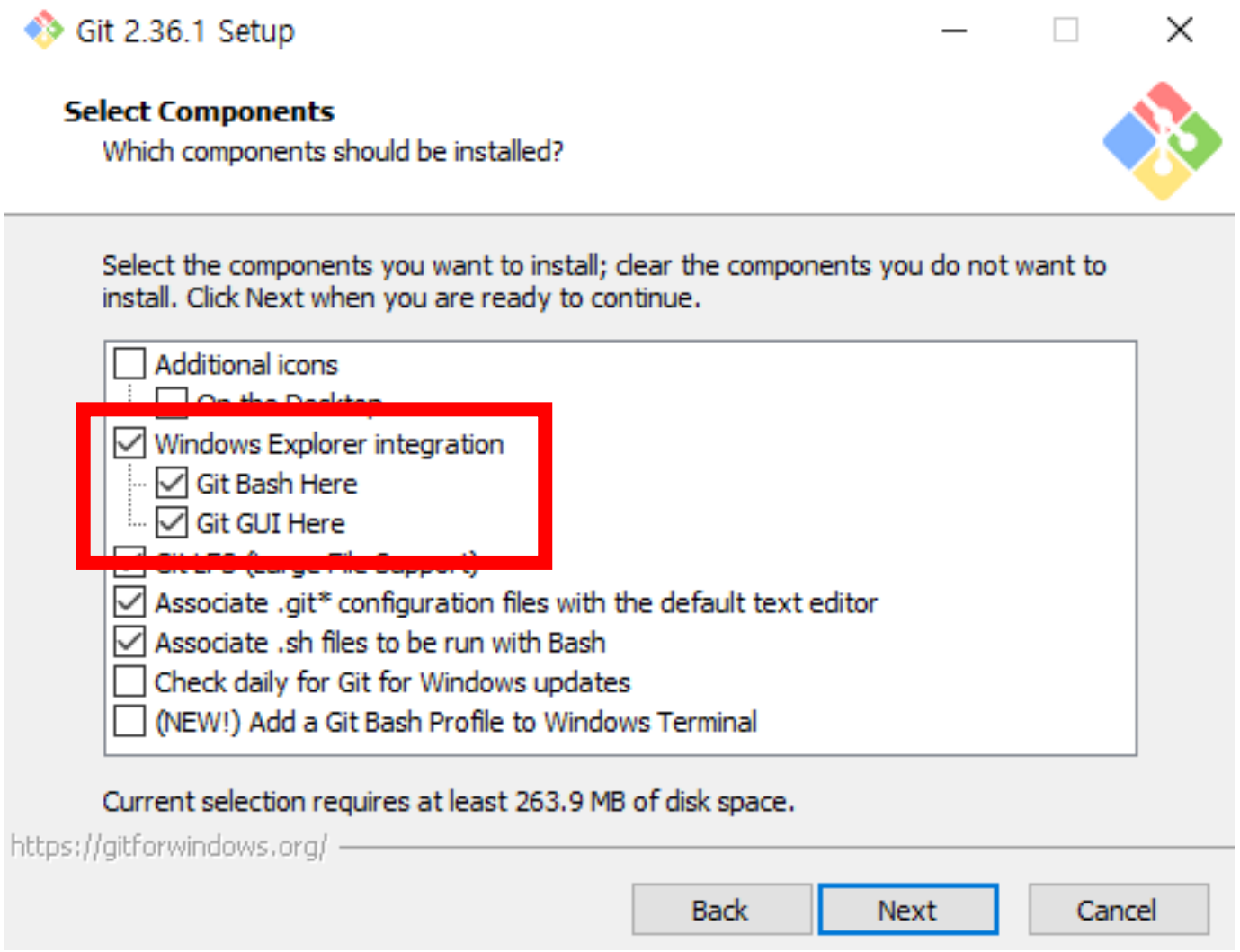
Standalone Installer
32-bit Git for Windows Setup.

64-bit Git for Windows Setup.

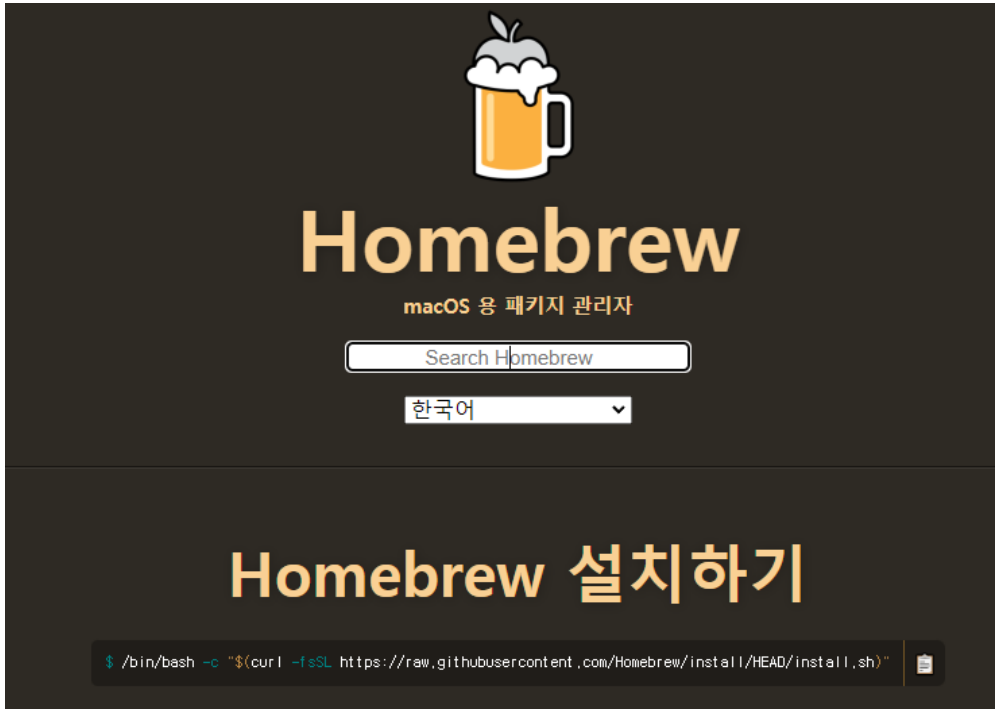
Portable ("thumbdrive edition")
32-bit Git for Windows Portable.

64-bit Git for Windows Portable.

Git, 설치하기(Windows)



Git, 설치하기(Mac)



https://brew.sh/index_ko

brew install git

Desktop

https://desktop.github.com/

Commit?

project

로컬에 설치

사용자
(local, 로컬)

Git hub 사용을 위한 세팅 시작!

- git init
- git config --global init.defaultBranch main
- 개행 문자 관련 처리(윈도우는 CR, LF / 맥은 LF 만)
  - git config --global core.autocrlf true (Window, CRLF → LF)
  - git config --global core.autocrlf input (Mac, LF 만 사용)
- git config --global user.name "프로필 이름"
- git config --global user.email "이메일 주소"
- git config --global --list

```
# 개행 문자(Newline) 설정
## macOS
$ git config --global core.autocrlf input
## Windows
$ git config --global core.autocrlf true

# 사용자 정보
## 커밋(버전 생성)을 위한 정보 등록
$ git config --global user.name 'YOUR_NAME'
$ git config --global user.email 'YOUR_EMAIL'

# 구성 확인
## Q키를 눌러서 종료!
$ git config --global --list
```

project

로컬에 설치

사용자
(local, 로컬)

```
$ git init
# 현재 프로젝트에서 변경사항 추적(버전 관리)을 시작.
```

git

사용자
(local, 로컬)

master

index.html

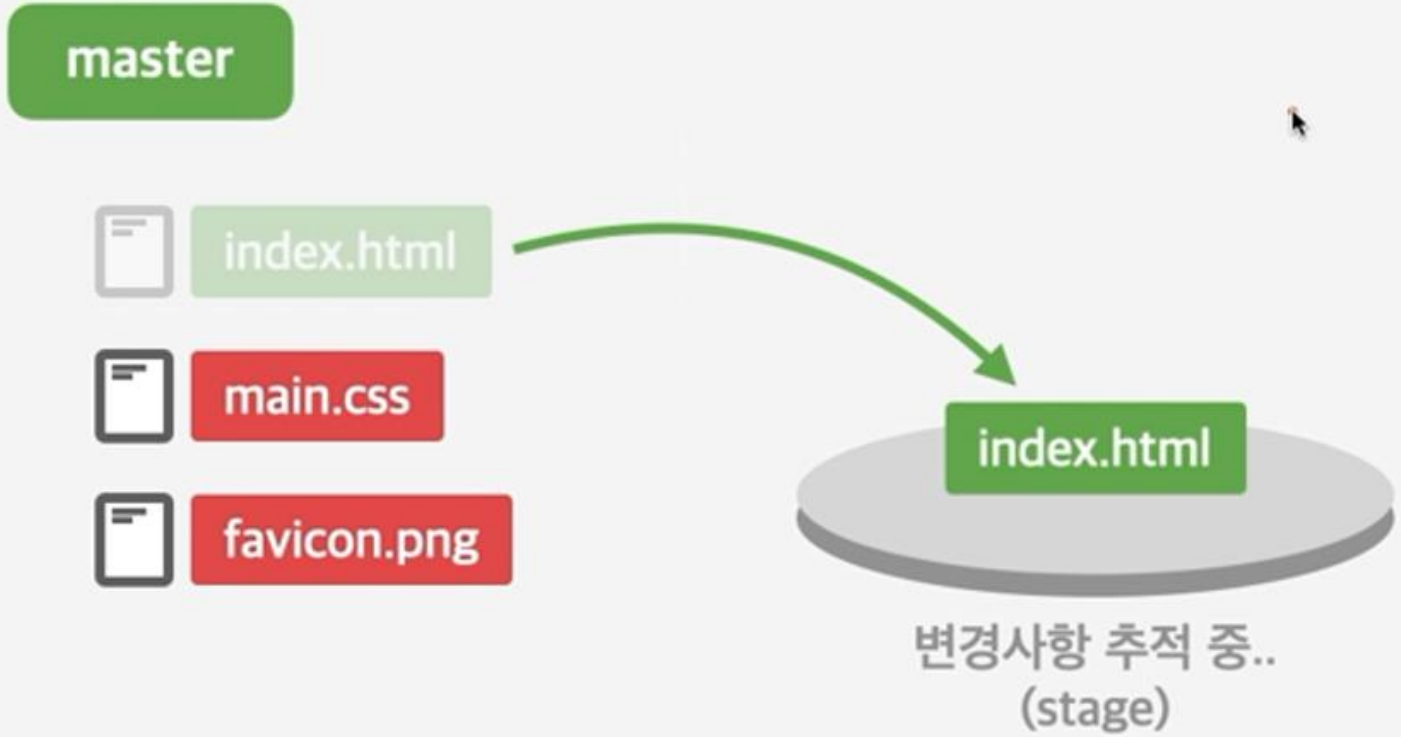
main.css

favicon.png

변경사항 추적 중..
(stage)

```
$ git add index.html
# 변경사항을 추적할 특정 파일(index.html)을 지정.
```

git 사용자 (local, 로컬)



```
$ git add .
# 모든 파일의 변경사항을 추적하도록 지정.
```

git 사용자 (local, 로컬)

master

```
$ git commit -m '프로젝트 생성'
# 메시지(-m)와 함께 버전을 생성.
```

git

사용자
(local, 로컬)

master

- index.html
- main.css
- favicon.png

● 프로젝트 생성

Git, 커밋 메시지!?

```
$
```

git 사용자 (local, 로컬)

master

- index.html
- main.css
- favicon.png
- main.js 파일 생성

프로젝트 생성

```
$ git add .
# 모든 파일의 변경사항을 추적하도록 지정.
```

git 사용자 (local, 로컬)

master

- index.html
- main.css
- favicon.png
- main.js

프로젝트 생성

```
$ git commit -m 'main.js 추가'
# 메시지(-m)와 함께 버전을 생성.
```

git 사용자 (local, 로컬)

master

- index.html
- main.css
- favicon.png
- main.js

main.js 추가

프로젝트 생성

```
$
```

git 사용자 (local, 로컬)

master

index.html 수정함

main.css 수정함

favicon.png

main.js

main.js 추가

프로젝트 생성

```
$ git add .
# 모든 파일의 변경사항을 추적하도록 지정.
```

git 사용자 (local, 로컬)

master

- index.html
- main.css
- favicon.png
- main.js

main.js 추가

프로젝트 생성

```
$ git commit -m 'index.html 수정'
# 메시지(-m)와 함께 버전을 생성.
```

git

사용자
(local, 로컬)

master

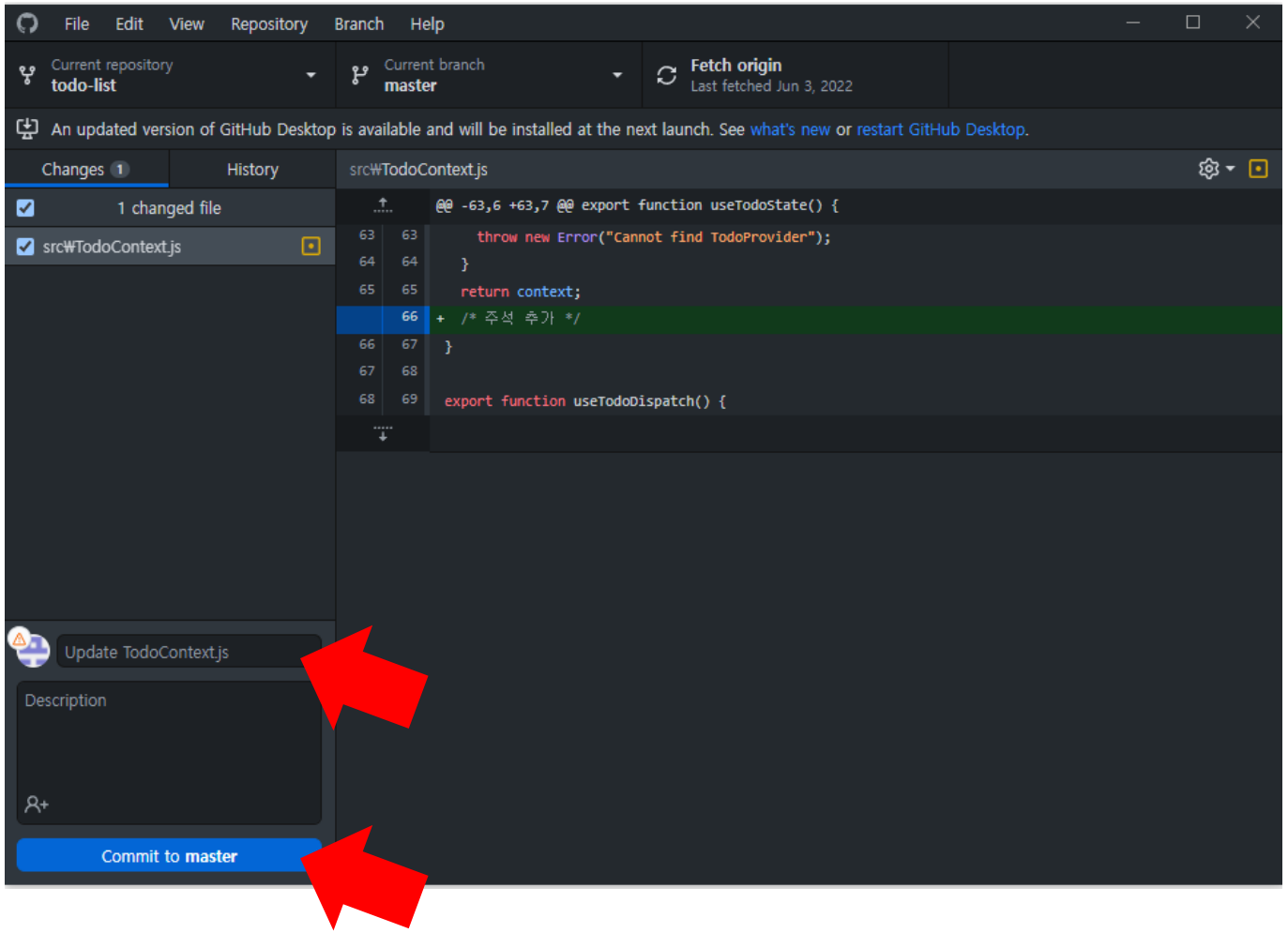
- index.html
- main.css
- favicon.png
- main.js

- index.html 수정
- main.js 추가
- 프로젝트 생성

Git, 커밋하기 on CLI

- git status
- Untracked files?
- git add 가 추천 되네요?
- git add .
- git status
- git commit -m “커밋 메세지”
- git log

Git, 커밋하기 on GUI



실습, 깃 커밋 연습

- 깃 커밋을 위한 빈 폴더 만들기
- 해당 폴더에 index.html 파일 생성
- 해당 내역 커밋하기(by CLI)
- 해당 폴더에 test.html 파일 생성
- 해당 내역 커밋하기(by CLI)
- 각각의 파일을 수정하고 커밋하기(by CLI)
- 같은 작업을 깃 허브 데스크탑으로도 수행 (별도의 폴더 생성)

Push!?

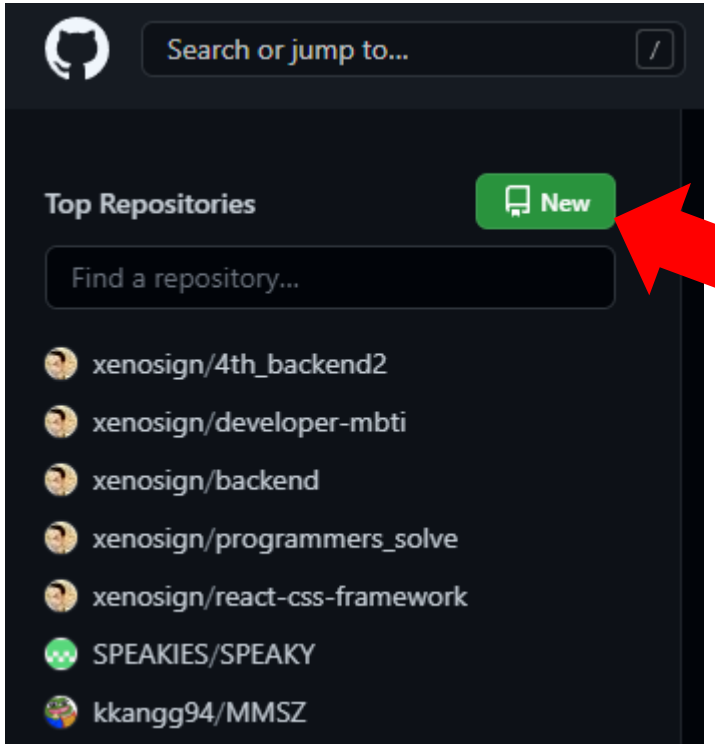

MEMZ
BSOD INCOMING
ENJOY THE NYAN CAT

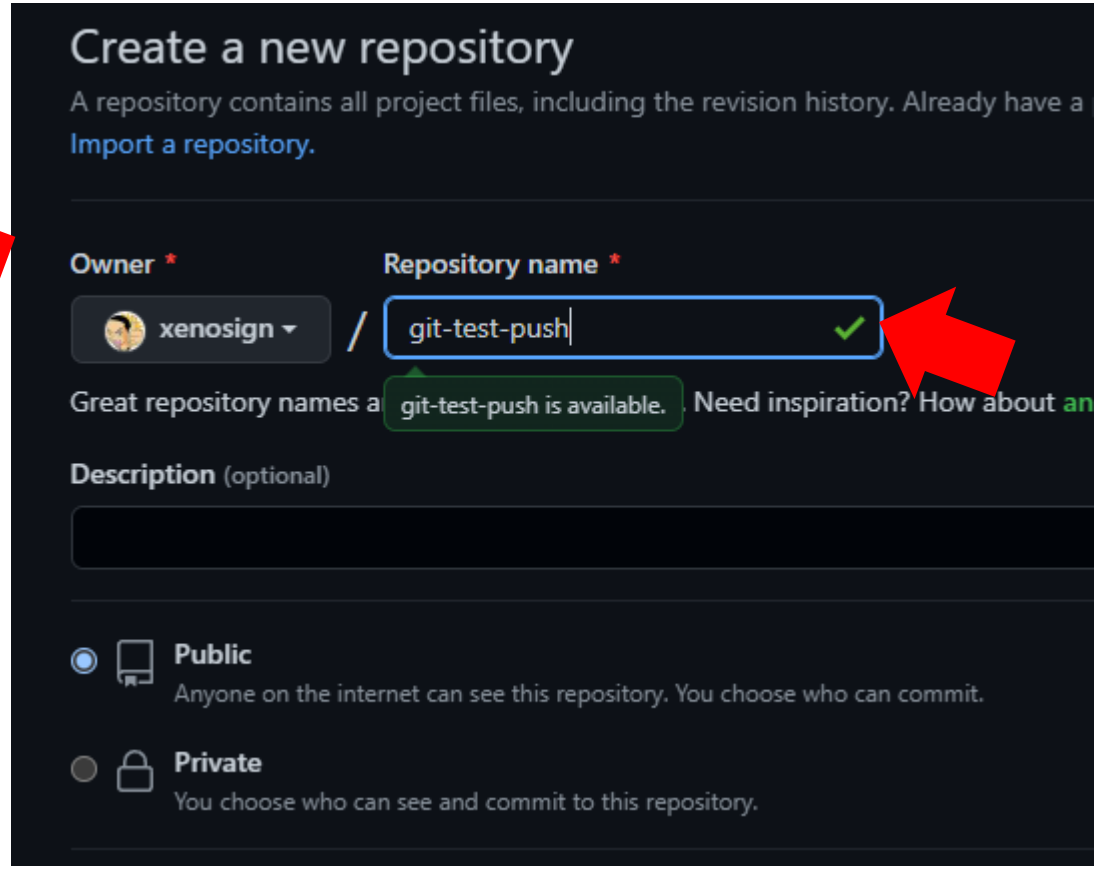

GitHub

GitHub
(remote, 원격)

원격 저장소
(Repository)

원격 저장소 생성하기


Search or jump to...
/
Top Repositories
New
Find a repository...
xenosign/4th_backend2
xenosign/developer-mbti
xenosign/backend
xenosign/programmers_solve
xenosign/react-css-framework
SPEAKIES/SPEAKY
kkangg94/MMSZ


Create a new repository
A repository contains all project files, including the revision history. Already have a
Import a repository.
Owner *
Repository name *
xenosign
/
git-test-push
Great repository names a
git-test-push is available.
Need inspiration? How about an
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.

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

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

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

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

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

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

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

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

git 사용자 (local, 로컬)

GitHub (remote, 원격)

```
$ git remote add origin https://github.c...
# origin이란 별칭으로 원격 저장소를 연결.
```

master

- index.html
- main.css
- favicon.png
- main.js

원격 저장소 (Repository)

git 사용자 (local, 로컬)

GitHub (remote, 원격)

```
$ git push origin master
# origin이란 별칭의 원격 저장소로 버전 내역 전송.
```

master

- index.html
- main.css
- favicon.png
- main.js

원격 저장소 (Repository)

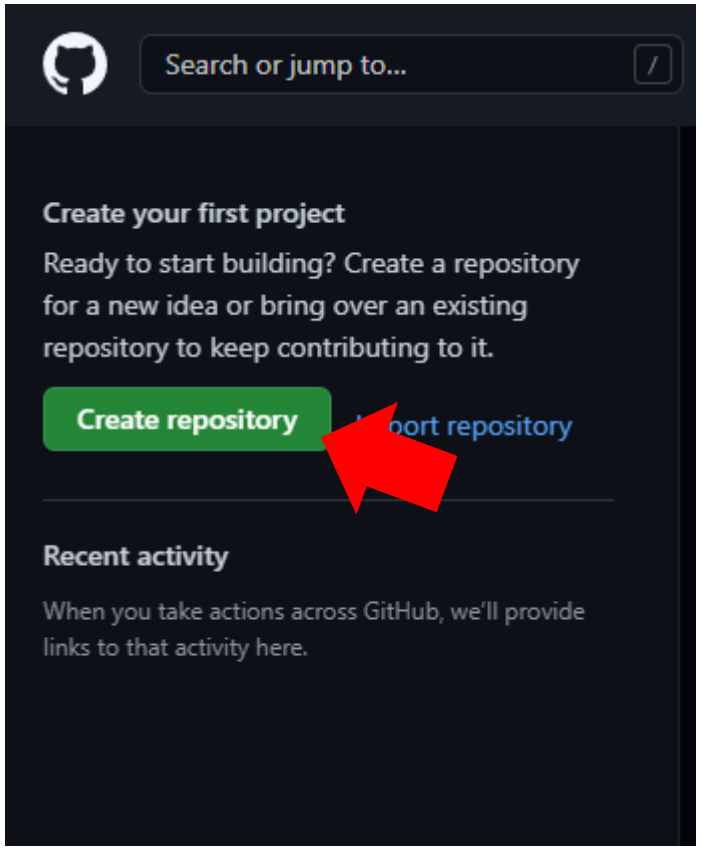
Push, 온라인에 깃을 저장하자!

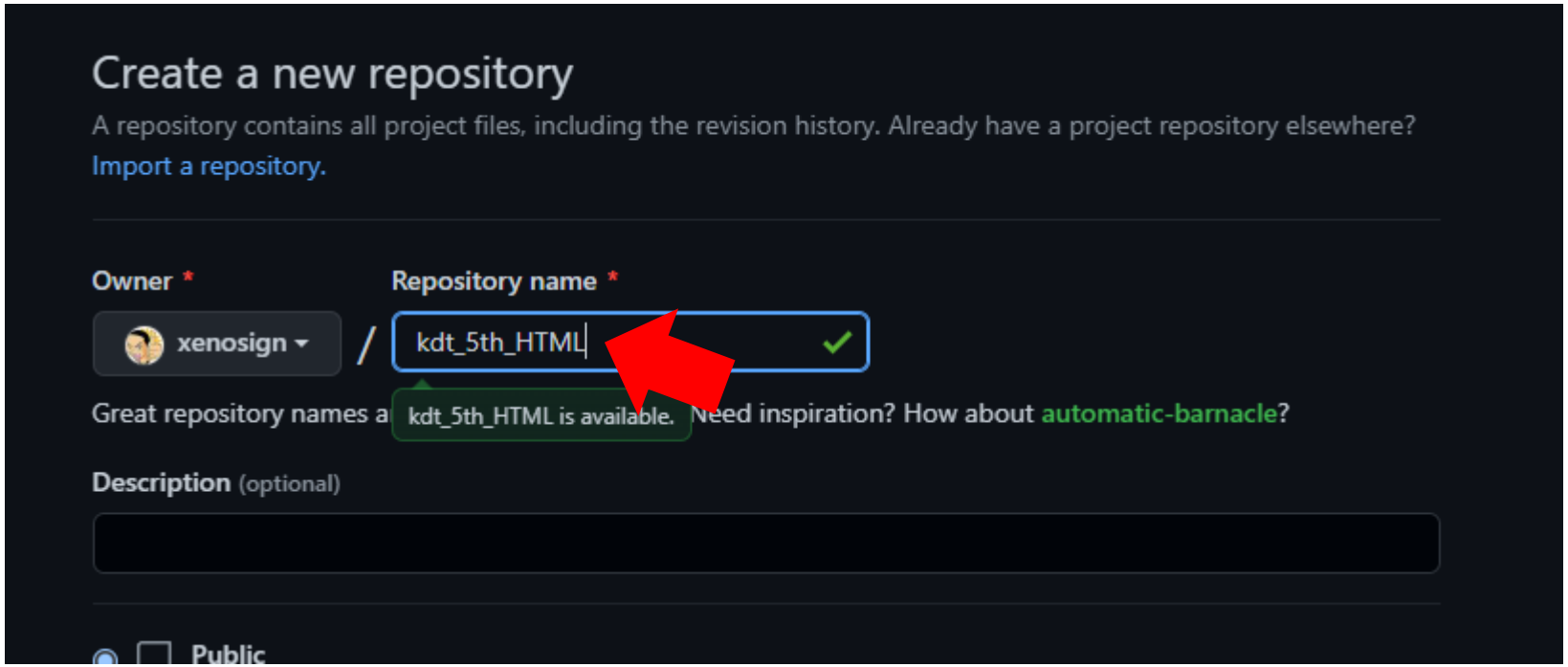
- Commit 은 여러분의 컴퓨터에서만 일어나는 일입니다!
- 즉, 온라인에 해당 내역을 올리지 않으면 github은 아무것도 모르는 거죠

Push, 온라인에 깃을 저장하자!

- Github에 여러분의 커밋 내역을 올리려면 반드시 push를 해주어야 합니다!
- Git이 관리하는 폴더 하나(=프로젝트) → github 리포지토리 하나
  - 두 개를 하나에 올리면, 코드의 변화를 추적하는 깃의 특성상 황당하겠죠?
  - 자동차 2대의 블랙 박스를 하나의 블랙 박스에 넣는 것과 비슷합니다!

Github 에 올리기! – Repo 생성





Github 에 올리기! – 안내 따라하기!

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

```
echo "# kdt_1st_HTML" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/kdtTetz/kdt_1st_HTML.git
git push -u origin main
```

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

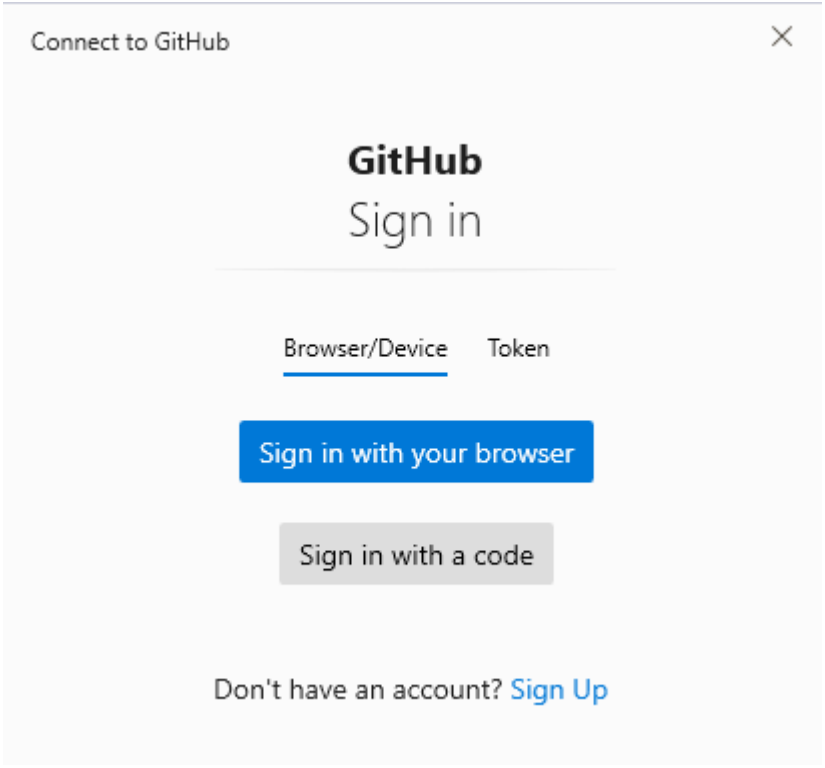
```
git remote add origin https://github.com/kdtTetz/kdt_1st_HTML.git
git branch -M main
git push -u origin main
```

- 상황에 맞는 코드를 복사해서 Vscode 에 붙여 넣기!

Github 에 올리기!(Push)

- echo "# ss" >> README.md
- git init
- git add .
- git commit -m "커밋 메세지"
- git branch -M main
- git remote add origin https://github.com/TetzSpreatics/ss.git
- git push -u origin main

window!

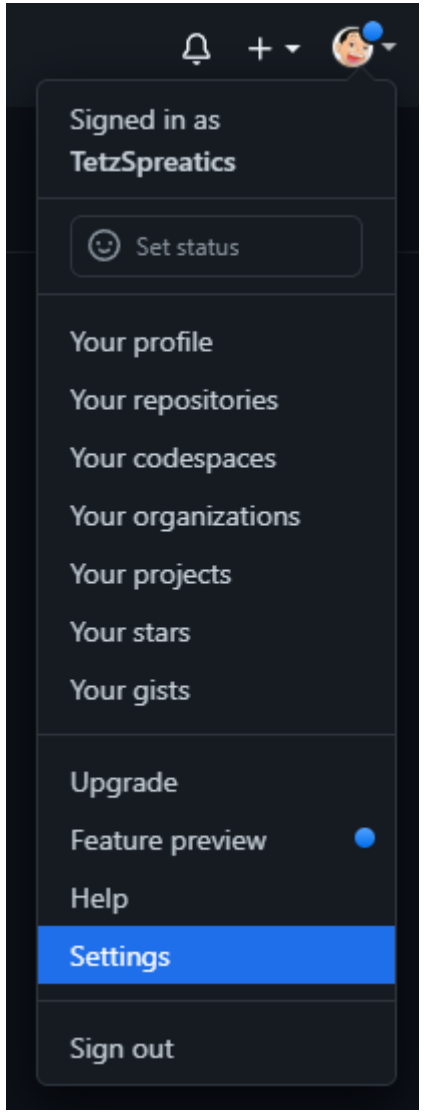


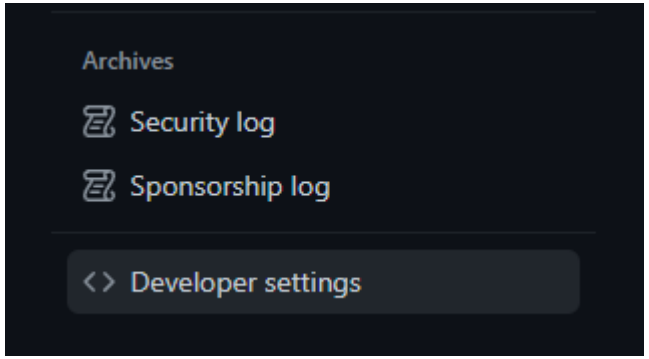
- 인터넷 브라우저로 깃허브에 로그인을 하면 인증이 됩니다!
- 혹, 여기서 다른 진행이 된다면 잠시 기다려 주세요!

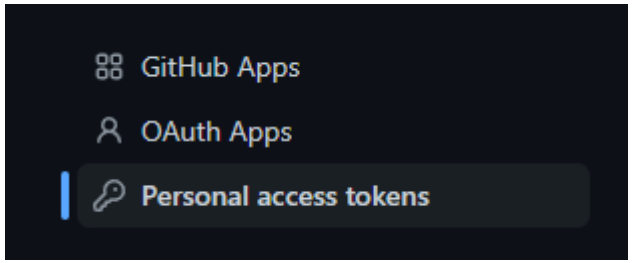
mac!

```
 ~/g/tcp-test  main ⇡1 git push
Username for 'https://github.com': xenosign
Password for 'https://xenosign@github.com':
오브젝트 나열하는 중: 4, 완료.
오브젝트 개수 세는 중: 100% (4/4), 완료.
Delta compression using up to 8 threads
오브젝트 압축하는 중: 100% (2/2), 완료.
오브젝트 쓰는 중: 100% (3/3), 263 bytes | 263.00 KiB/s, 완료.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/xenosign/tcp-test.git
   45aebc3..a938f5a  main -> main
```

Access Token 발행!







토큰 저장해 두기!

Settings / Developer settings

- GitHub Apps
- OAuth Apps
- Personal access tokens
  - Fine-grained tokens (Beta)
  - Tokens (classic)

Personal access tokens (classic)

Generate new token | Revoke all

Tokens you have generated that can be used to access the GitHub API.

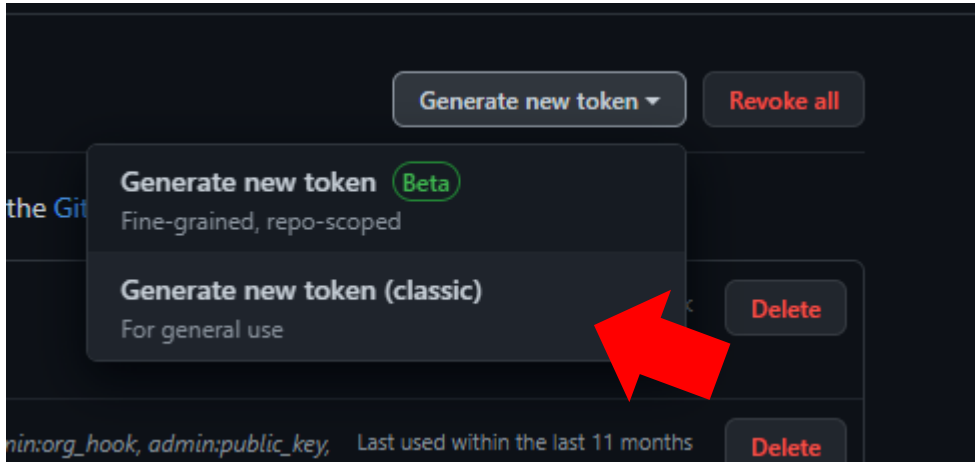
git — *repo*
Last used within the last week — Delete

⚠ This token has no expiration date.

toy project — *admin:enterprise, admin:gpg_key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, delete:packages, delete_repo, gist, notifications, repo, user, workflow, write:discussion, write:packages*
Last used within the last 11 months — Delete

Expired *on Sat, May 28 2022.*

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.


Generate new token
Revoke all
Generate new token
Beta
Fine-grained, repo-scoped
Generate new token (classic)
For general use
Delete
Last used within the last 11 months
Delete

New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

What's this token for?

Expiration *

30 days ken will expire on Sat, Feb 25 2023

Select scopes

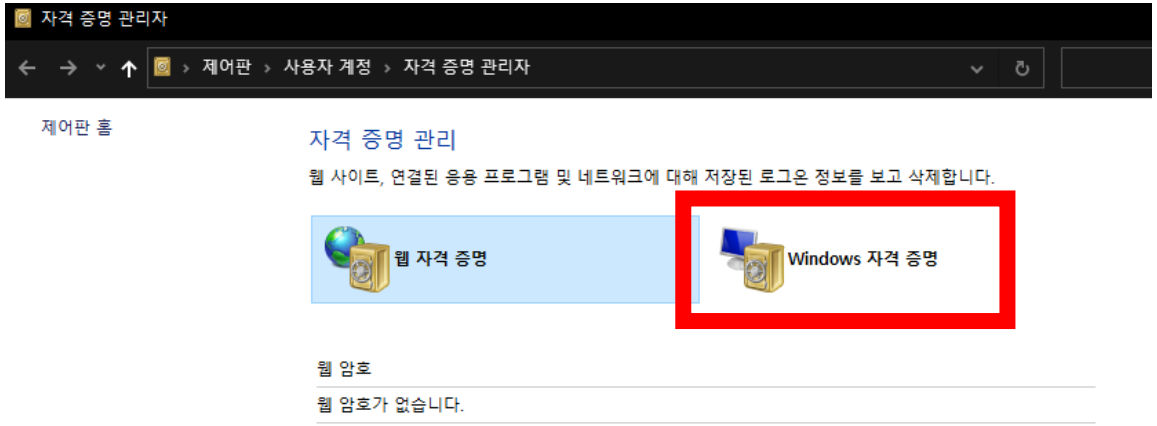
Scopes define the access for personal tokens. Read more about OAuth scopes.

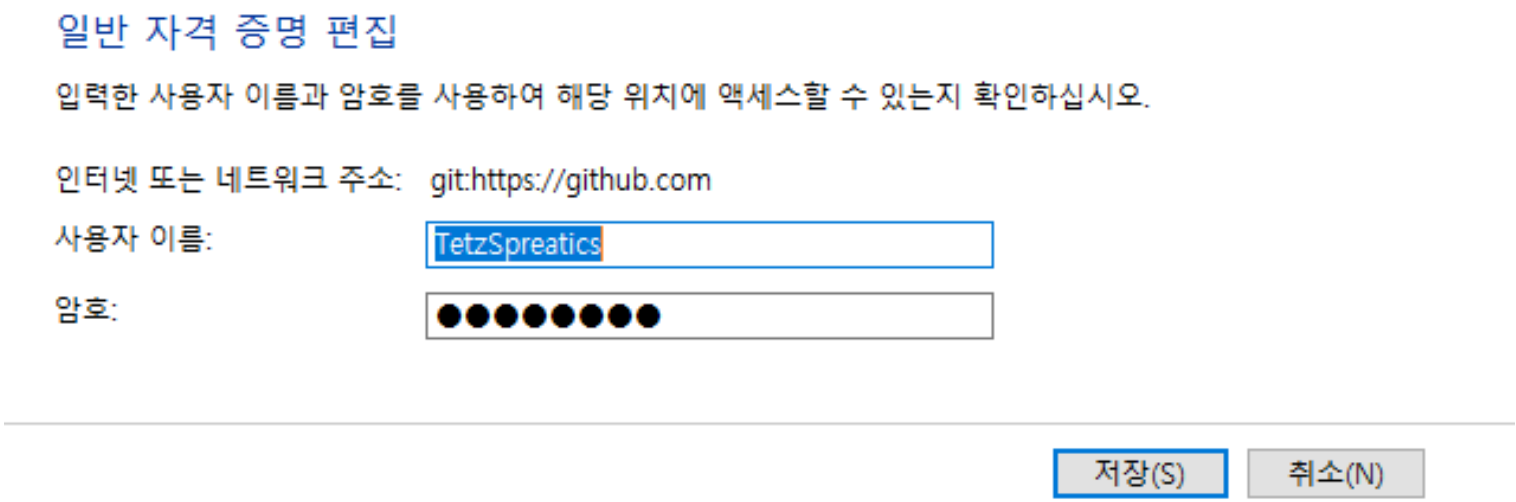
repo	Full control of private repositories
repo:status	Access commit status
repo_deployment	Access deployment status
public_repo	Access public repositories
repo:invite	Access repository invitations
security_events	Read and write security events
workflow	Update GitHub Action workflows
write:packages	Upload packages to GitHub Package Registry

```
 ~/g/tcp-test  main ⇡1 git push
Username for 'https://github.com': xenosign
Password for 'https://xenosign@github.com':
오브젝트 나열하는 중: 4, 완료.
오브젝트 개수 세는 중: 100% (4/4), 완료.
Delta compression using up to 8 threads
오브젝트 압축하는 중: 100% (2/2), 완료.
오브젝트 쓰는 중: 100% (3/3), 263 bytes | 263.00 KiB/s, 완료.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/xenosign/tcp-test.git
   45aebc3..a938f5a  main -> main
```

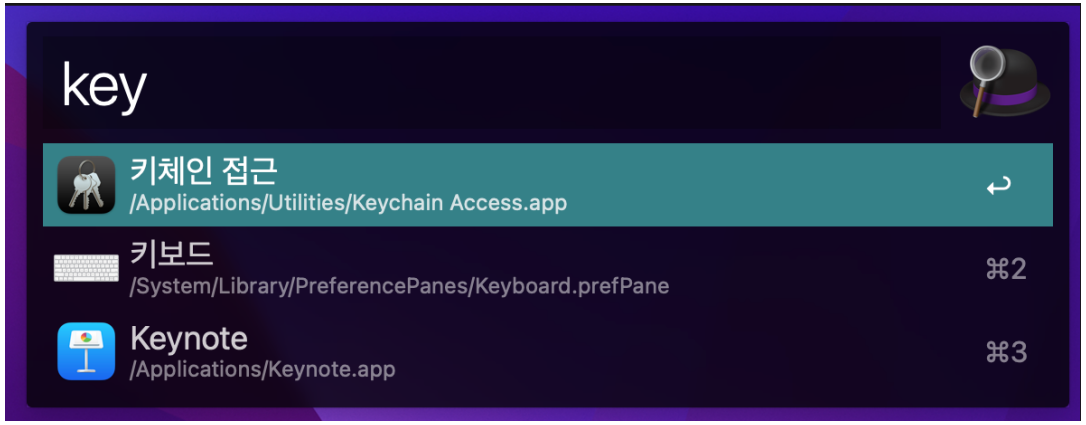
문제 사항 처리

For Windows


자격 증명 관리자
제어판 › 사용자 계정 › 자격 증명 관리자
제어판 홈
자격 증명 관리
웹 사이트, 연결된 응용 프로그램 및 네트워크에 대해 저장된 로그온 정보를 보고 삭제합니다.
웹 자격 증명
Windows 자격 증명
웹 암호
웹 암호가 없습니다.


일반 자격 증명 편집
입력한 사용자 이름과 암호를 사용하여 해당 위치에 액세스할 수 있는지 확인하십시오.
인터넷 또는 네트워크 주소: git:https://github.com
사용자 이름: TetzSpreatics
암호: ●●●●●●●●
저장(S)
취소(N)

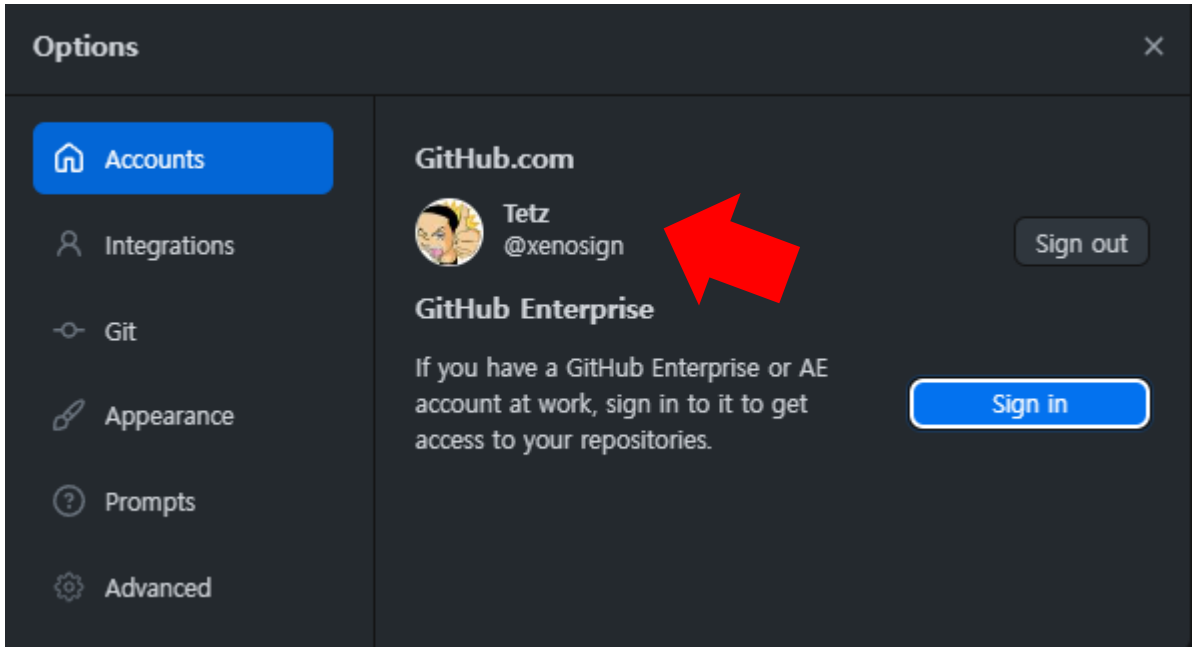
For Mac

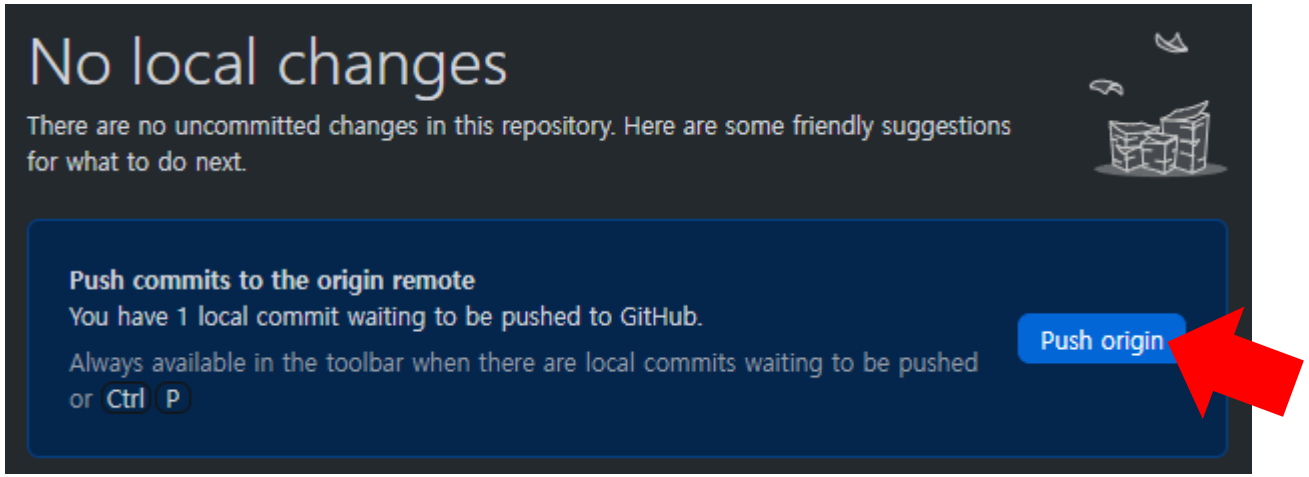

key
키체인 접근
/Applications/Utilities/Keychain Access.app
↩
키보드
/System/Library/PreferencePanes/Keyboard.prefPane
⌘2
Keynote
/Applications/Keynote.app
⌘3


키체인 접근
git
기본 키체인
로그인
로컬 항목
시스템 키체인
시스템
시스템 루트
모든 항목
암호
보안 메모
내 인증서
키
인증서
github.com
종류: 인터넷 암호
계정: xenosign
위치: https://github.com
수정일: 2022. 2. 27. 오후 9:41:42
이름
종류
수정일
만료
키체인
로그인
로그인
로그인
github.com
속성
접근 제어
이름: github.com
종류: 인터넷 암호
계정: xenosign
위치: https://github.com
설명:
암호 보기:
변경 사항 저장

귀찮을 땐!?

Git, 푸쉬하기 on GUI





블랙 박스와 비슷합니다!

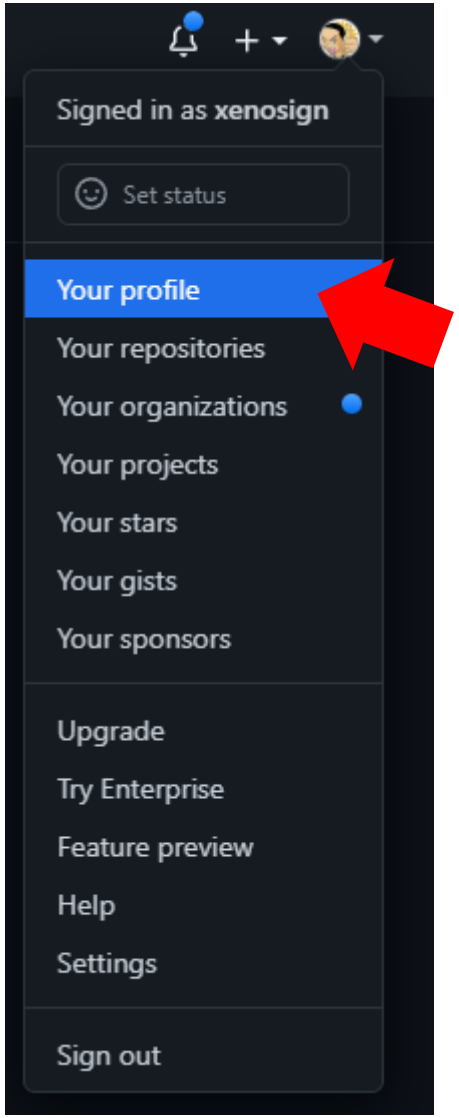
- 하나의 프로젝트 = 하나의 폴더 = 한대의 차
- A 라는 차가 사고가 났을 때, B 라는 차의 블랙 박스를 확인하면 될까요?
- 또, 급하다고 해서 C 라는 차의 블랙 박스 데이터를 D 라는 차에 덮어 씌워도 될까요?

실습, 깃 푸쉬 연습

1. 실습을 위한 github repository 생성
2. 1의 폴더에 main.html 파일 생성 후 커밋
3. 해당 내역을 생성한 github repository 에 푸쉬하기

모는

이미 심어 졌습니다!


Signed in as xenosign
Set status
Your profile
Your repositories
Your organizations
Your projects
Your stars
Your gists
Your sponsors
Upgrade
Try Enterprise
Feature preview
Help
Settings
Sign out


640 contributions in the last year
Contribution settings
We're celebrating 100 million developers! Read our blog post.
Play animation
Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec Jan
Mon
Wed
Fri
Learn how we count contributions
Less More


space 공감
EBS HD
나는 사는 게 재밌다
매일매일 신난다

귀찮으면

Github desktop

CSS


KIM LE NA
KIM LE NA
KIM LE NA
KIM LE NA
KIM LE NA

VS


VISA
VISA
MasterCard
VISA

신문검색 | 국외검색 NAVER 서치센터 | 광고안내

맞춤정보 서비스, MyNAVER
매일매일 핫&쿨, WebToday

삼성생명 오픈 이벤트

네이버를 나의 홈페이지로,,,
네오넷의 부동산 급매물 서비스!

NAVER 검색

분류에서 검색

제목에서 검색 Quick Help 확장검색

NAVER 소식

네이버 포탈서비스 오픈! MyNAVER
네이버 웹문서수 3,141,838 건

NAVER 분류

- 건강/의학 — 건강관리, 병원, 의학
- 교육 — 대학, 시험,자격증, 유학
- 뉴스/미디어 — 신문, 잡지, 텔레비전
- 레크리에이션 — 스포츠, 게임, 여행, 레저
- 사업/경제 — 기업, 취업, 온라인쇼핑
- 사회과학 — 경제학, 사회학, 언어학
- 사회/문화 — 결혼, 기관/단체
- 연예/오락 — 연예인, 영화, 음악, 유머
- 인문/예술 — 디자인, 인문과학, 박물관
- 자연과학 — 공학, 컴퓨터과학, 생물학
- 정부/공공기관 — 정치, 한국정부, 국제기구
- 지역정보 — 대한민국, 서울, 국가
- 참고자료 — 도서관, 사람찾기, 사전
- 컴퓨터 — 인터넷, S/W, O/S, 통신

NAVER 추천

New! 크리스마스
행사,이벤트 여행 온라인쇼핑
영화,비디오 취업,채용 투자,재테크

AT & T 전화접속서비스

도움말 | 새소식 | 홈페이지등록 | 다른검색사이트

Microsoft Internet Explorer

VS

NAVER whale 눈부심 없는 편안~함 내가 다크 모드 쓰는 이유 다운로드 3일 동안 보지 않기

네이버를 시작페이지로

NAVER

메일 카페 블로그 지식iN 쇼핑 쇼핑LIVE Pay TV 사전 뉴스 증권 부동산 지도 VIBE 책 웹툰 더보기 미세 좋음 초미세 좋음

4월 11일 부터 자동차보험료를 내렸습니다!
개인용 자동차 평균 1.2% ↓, 4/11 보험시작일 기준
바로확인

네이버를 더 안전하고 편리하게 이용하세요
NAVER 로그인
아이디 · 비밀번호찾기 회원가입

연합뉴스 › '오자범위내 접전' 인천 계양을...이재명-윤형선 총력전
뉴스홈 · 연예 스포츠 지방선거

증시 | 다우 32,893.24 ▲256.05 +0.78%

뉴스스탠드 › 구독한 언론사 · 전체언론사

연합뉴스TV KBS 한국경제TV 매일경제 헤럴드경제 뉴스1
동아일보 노컷뉴스 KBS WORLD 이데일리 프레시안 BLOTER
Net Korea sportalkorea NewDaily 디지털정글 DAILY NK YTN 사이언스
CEO스코어데일리 Media오늘 Byline Network 비즈한국 더쿠쿠프 뉴스핌

제8회 지방선거 사전투표2일차
투표 안내 › 투표소 찾기 › 후보자 정보 › 투표시간 오전 6시 ~ 오후 6시

오늘 읽을만한 글 주제별로 분류된 다양한 글 모음
681 개의 글 | 관심주제 설정

N 인플루언서
번아웃 증후군을 이겨내는 법
도서 블로거 @힐링의숲 추천
직장인 마음 치유 도서

트렌드쇼핑 › 상품 쇼핑몰 MEN
쿠팡 · G마켓 · 옥션 · 11번가 · 티몬 · 올리브영
위메프 · SSG닷컴 · GS샵 · 롯데홈 · CJ온스타일 · 패션플러스

1 / 18

네이버에

CSS 가 없다면?

확장 프로그램

확장 프로그램 더보기

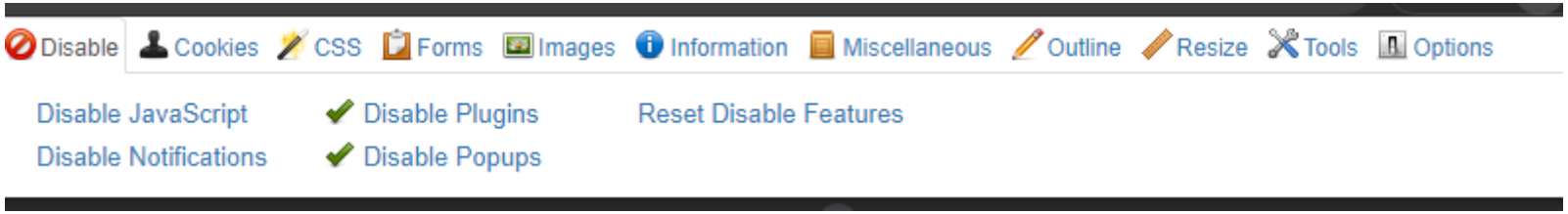
WEB DEVELOPER

Web Developer

chrispederick.com

Adds a toolbar button with various web developer tools.

★★★★½ 2,824 개발자 도구



Disable Cookies CSS

- Disable All Styles
- Disable Browser Default Styles
- Disable Embedded Styles
- Disable Inline Styles

뉴스스탠드 바로가기 주제별캐스트 바로가기 타임스퀘어 바로가기 쇼핑캐스트 바로가기 로그인 바로가기
매일 쓰는 브라우저 보안이 걱정된다면, **안전하고 빠른 최신 브라우저 웨일로 업데이트 하세요.**다운로드 3일 동안 보지 않기

네이버

네이버를 시작페이지로 쥬니어네이버 해피빈

검색

통합검색
검색어를 입력해 주세요.
검색

한글 입력기
자동완성 레이어
최근검색어
전체삭제

- @txt@ *@date@.* *삭제*

검색어 저장 기능이 꺼져 있습니다.
설정이 초기화 된다면 도움말을 확인해주세요.
최근 검색어 내역이 없습니다.
설정이 초기화 된다면 도움말을 확인해주세요.
도움말 자동저장 끄기

자세히보기
관심사를 반영한 컨텍스트 자동완성*도움말*
컨텍스트 자동완성
컨텍스트 자동완성

ON/OFF 설정은
해당기기(브라우저)에 저장됩니다.

자세히

동일한 시간대/연령/남녀별 사용자 그룹의
관심사에 맞춰 자동완성을 제공합니다.

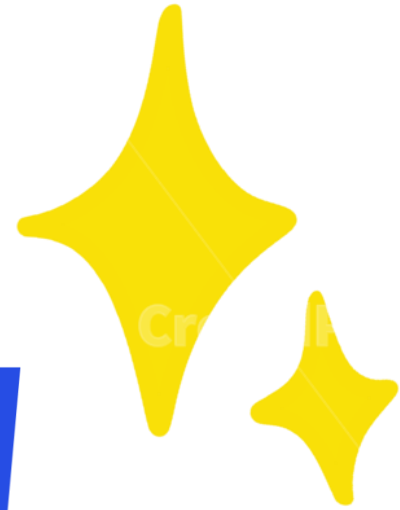
로그인 자세히
컨텍스트 자동완성 레이어 닫기
도움말 신고 자동완성 끄기

- 메일
- 카페
- 블로그

그래서 HTML에 Style을 입히고자

- HTML5 이전 버전에서는 HTML 자체에서 Style을 컨트롤 할 수 있는 태그 <font>, 등을 사용
- HTML은 결국 텍스트를 전달하는 문서 이자 문서의 구조!
- HTML 본연의 기능이 아닌 디자인 적 요소가 추가되면서 많은 혼란을 야기!

CSS



百聞不如一見

```
<body>
    <p>안녕하세요</p>
    <p style="color: blue; font-weight: bold; background-color: yellowgreen">
      안녕하세요
    </p>
</body>
```

```
<body>
    <h1>KDT</h1>
    <h1 style="color: aqua; background-color: blueviolet">KDT</h1>
    <h1 style="background-color: greenyellow; width: 500px">KDT</h1>
    <h1 style="background-color: greenyellow; width: 500px; padding: 10px">
      KDT
    </h1>
    <h1 style="background-color: greenyellow; width: 500px; margin: 10px">
      KDT
    </h1>
</body>
```

CSS를 적용하는 방식

- 인라인 스타일
- 내장 Style
- 파일 링크

인라인(inline) 방식은 불편해요!!

인라인 방식

요소의 style 속성에 직접 스타일을 작성하는 방식
(선택자 없음)

```
<div style="color: red; margin: 20px;"></div>
```

- 각각 태그마다 전부 스타일을 적어줘야 함
- 아래에서 같은 스타일을 가진 태그를 사용하려고 해도 코드를 복붙 필요
- 즉, 재사용이 전혀 불가능!

그래서 탄생! 내장 Style

```
<style>
  div {
    color: red;
    margin: 20px;
  }
</style>
```

내장 방식

<style></style>의 내용(Contents)으로
스타일을 작성하는 방식

CSS의 기본 문법!

```
<style>
  div {
    color: red;
    margin: 20px;
  }
</style>
```

- **선택자** : div 에 스타일을 적용
- **{}** : div 에 적용할 스타일을 {} 안에 정의
- **속성명** : 어떠한 속성을 정의 하겠다
- **속성값** : 속성을 ~~ 한 값으로 정하겠다

CSS의 주석

```
div {
  /* 안의 내용은
  주석으로 처리 됩니다 */
}
```

- /* */ 내부의 내용은 주석으로 처리

내장 Style 의 한계

- 그런데 다른 HTML 파일에는 어떻게 쓰죠?
- 혹시나 스타일이 바뀌면 모두 바꿔줘야 하나요?

그래서 탄생! 링크 방식

```
<link rel="stylesheet" href="./css/main.css">
```

링크 방식

<link /> 로 외부 CSS 문서를
가져와서 연결하는 방식

main.css

```
div {
  color: red;
  margin: 20px;
}
```

그래서 탄생! 링크 방식

- 모든 Style을 하나의 CSS 파일에 넣고, 필요한 HTML 파일에서 해당 파일을 링크해서 사용하는 방식
- 재사용? OK
- 스타일 변경 발생? OK
- 그런데 모든 것을 하나의 파일에서 관리하는 것이 맞을까?
  - 다른 CSS 파일의 코드를 가져와야 할때는?

그래서 탄생! @import 방식

```
<link rel="stylesheet" href="./css/main.css">
```

main.css

```
@import url("./box.css");

div {
  color: red;
  margin: 20px;
}
```

box.css

```
.box {
  background-color: red;
  padding: 20px;
}
```

@import 방식

CSS의 @import 규칙으로 CSS 문서 안에서
또 다른 CSS 문서를 가져와 연결하는 방식

3가지 CSS 참조 방식

인라인 방식 > 내장 방식 ? 링크 방식

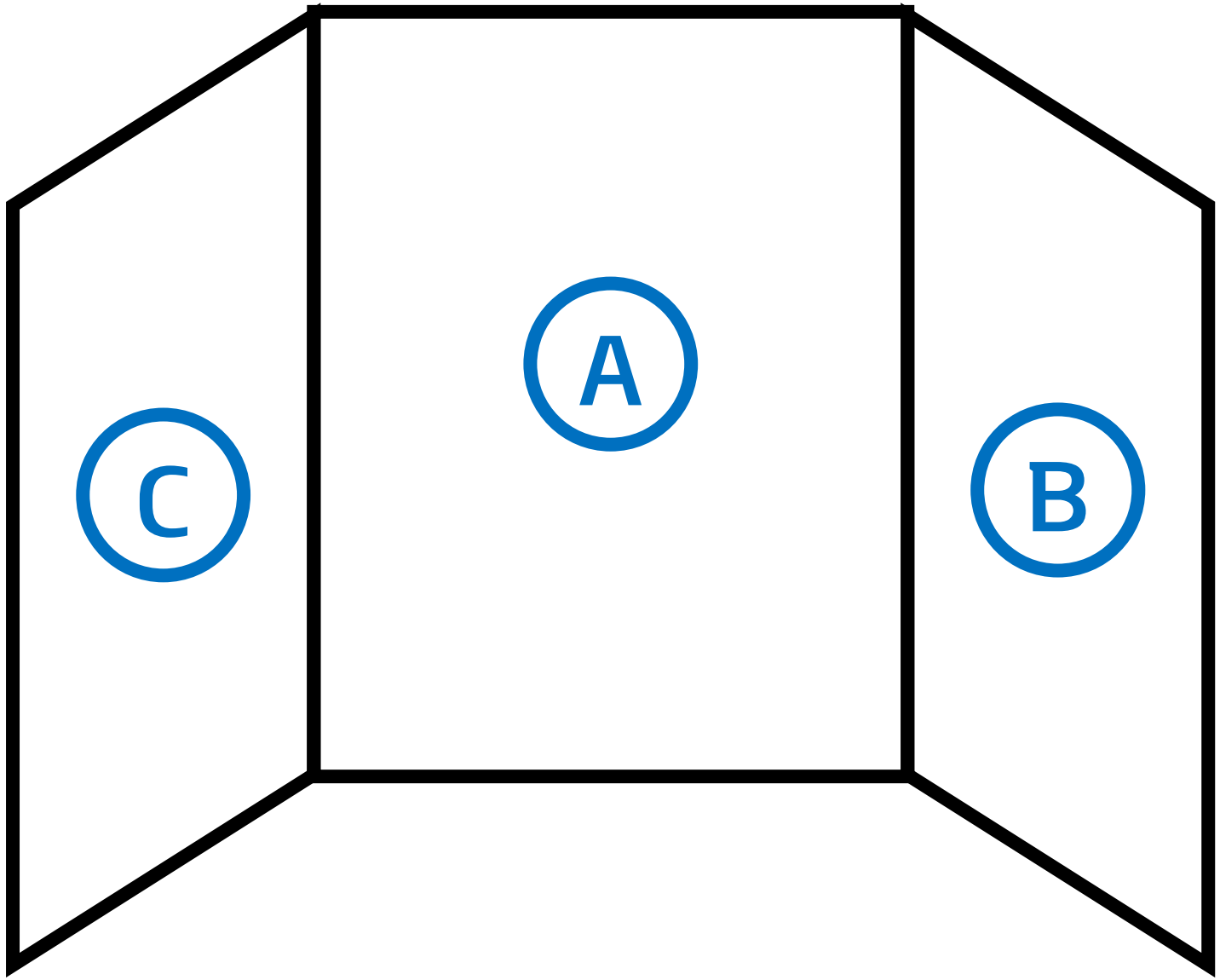
- 그런데 3가지 방식이 겹치면 어떻게 되죠?
- 기본 룰은 가장 늦게 읽히는 것이 우선으로 적용 됩니다!
- 인라인 방식은 내장, 링크 방식에 무조건 우선!
- 내장, 링크 방식은 늦게 쓰여진 것이 우선!

실습, 모든 CSS 적용 방법 해보기

- HTML 문서를 만들고 <h1>, <h2>, <h3>, <p>, 태그로 글자를 쓰기
- <h1> 태그는 인라인 방식으로 배경을 초록, 글자를 흰색
- <h2> 태그는 내장 방식로 배경을 파랑, 글자를 흰색
- <p> 태그는 링크 방식으로 배경을 빨강, 글자를 흰색
- 태그는 @import 방식으로 배경을 오렌지, 글자는 파란색
- <h3> 태그는 배경을 검정, 글자를 흰색으로 선언. 단, 내장 방식과 링크 방식에 전부 선언 한 뒤, 내장 방식이 적용 되도록 설정!

한번

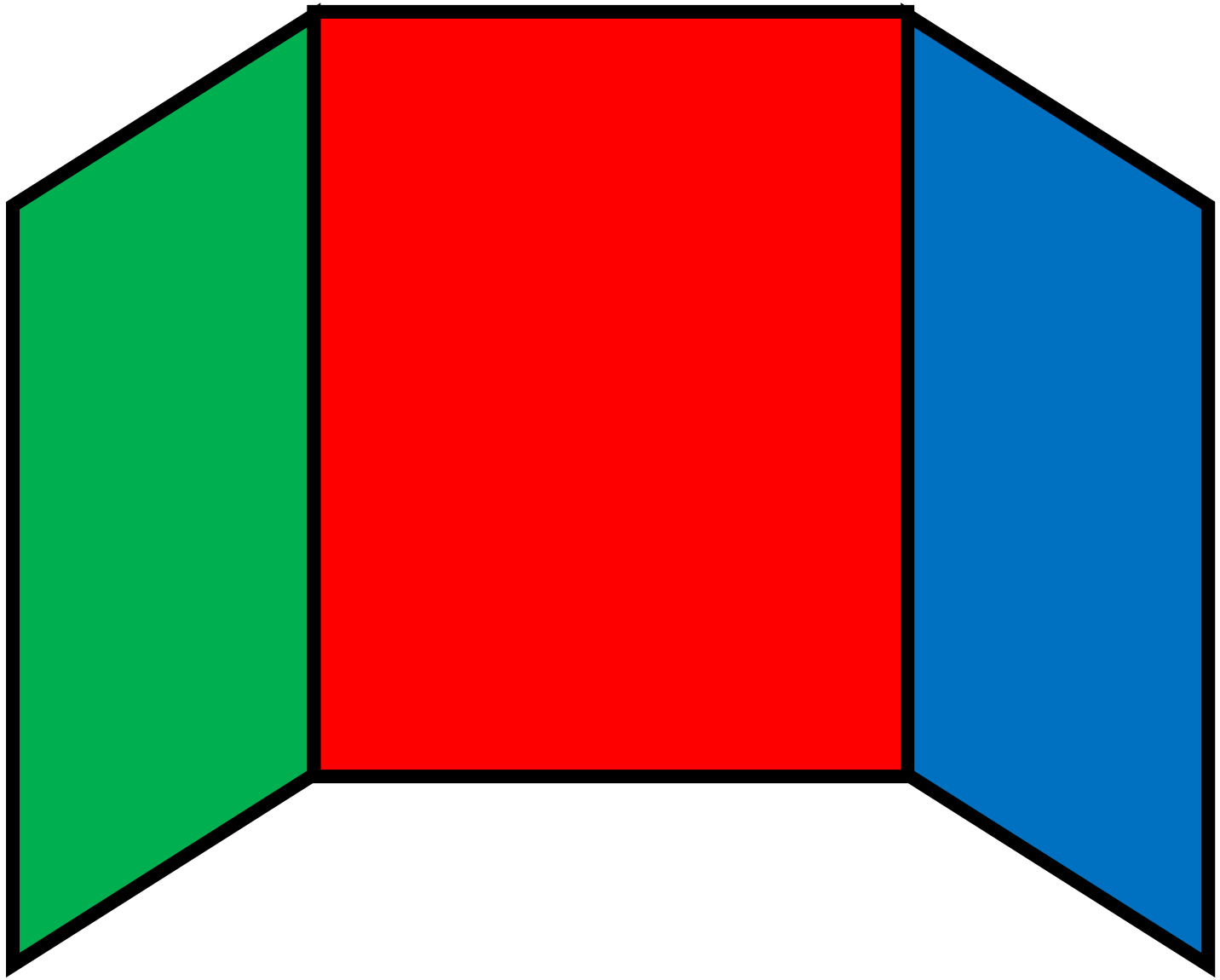
생각해 보세요!


A
C
B


2"
SERIES P-200-L

"빨간, 파란, 초록색으로 예쁘게 칠해주세요"

"A는 빨간색
B는 파란색
C는 초록색으로
예쁘게 칠해주세요!"



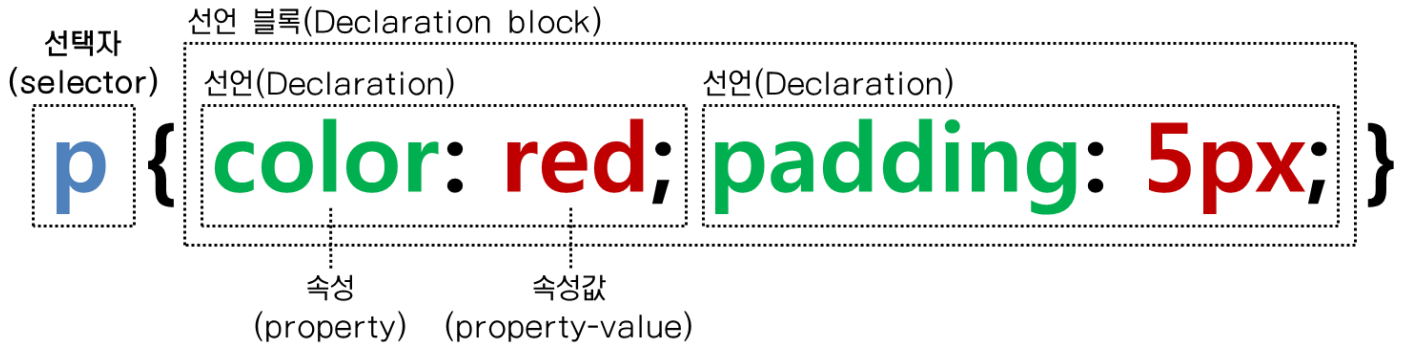

2"
SERIES P-200-L

CSS를 적용 할 때도 어떤 곳에 적용할 지 정확하게 불러주는 것이 중요합니다~!

CSS

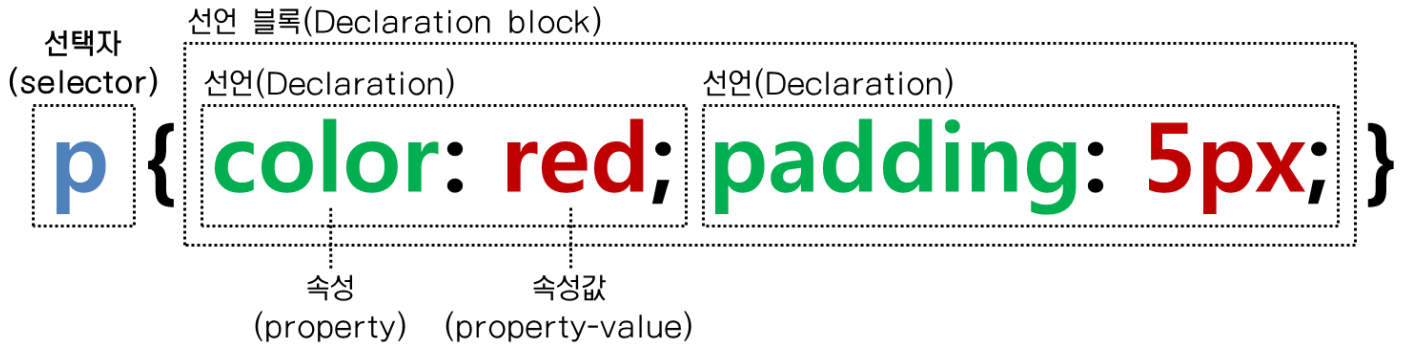
선택자!

CSS, 선택자(Selector)



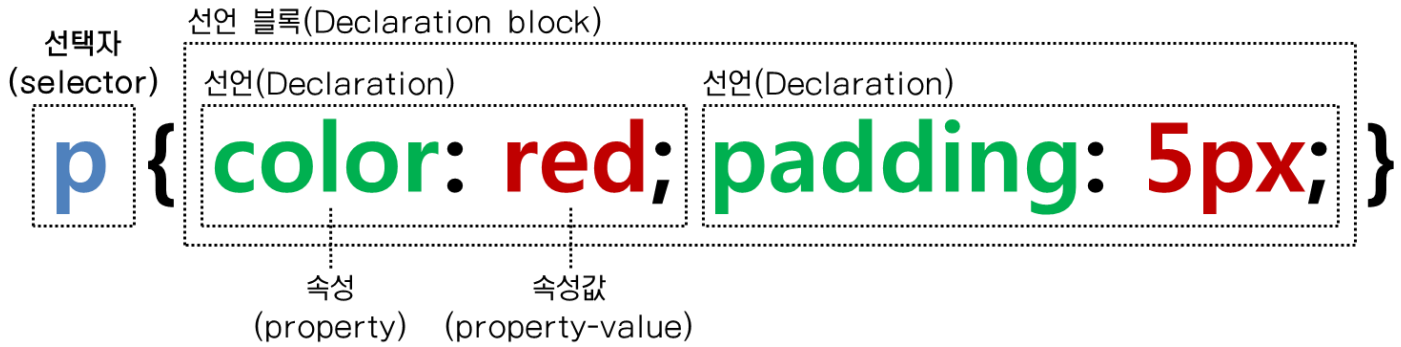
- CSS는 HTML 요소의 style(design, layout etc)을 정의 하여야 하므로 CSS 사용을 위해서는 스타일을 적용하고자 하는 HTML 요소를 선택할 수 있어야 한다
- 선택자는 스타일을 적용하고자 하는 HTML 요소를 선택하기 위해 CSS에서 제공하는 수단

CSS, 속성(Property)



- 선택자로 HTML 요소를 선택하고 { } 내에 속성 값을 지정하여 다양한 style 을 정의
- 속성은 표준 스펙으로 이미 지정되어 있는 것을 사용, 사용자가 임의로 정의 할 수 없다
- 여러 개의 프로퍼티를 연속해서 지정할 수 있으며 세미콜론(;)으로 구분

CSS, 값(Value)



- 값은 해당 속성에 사용할 수 있는 값을 키워드나 크기 단위 또는 색상 단위 등의 특정 단위로 지정

CSS 선택자

기본 | 복합 | 가상 클래스 | 가상 요소 | 속성

기본 선택자!

- 기본 선택자니까 기본적인 선택자겠죠?
- 별도의 테크닉 없이, 순수하게 무엇인가를 호출 할 때 사용합니다!
- 종류
  - 전체 선택자
  - 태그 선택자
  - Class 선택자
  - ID 선택자

* 기본 전체 선택자 (Universal Selector)

모든 요소를 선택.

```html
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li>오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span>오렌지</span>
</div>
```

선택

```css
* {
  color: red;
}
```

ABC

기본 | 태그 선택자 (Type Selector)

태그 이름이 ABC인 요소 선택.

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li>오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span>오렌지</span>
</div>
```

선택

```
li {
  color: red;
}
```

.ABC

기본 | 클래스 선택자 (Class Selector)

HTML class 속성의 값이 ABC인 요소 선택.

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li class="orange">오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span class="orange">오렌지</span>
</div>
```

선택

```
.orange {
  color: red;
}
```

#ABC

기본 | 아이디 선택자 (ID Selector)

HTML id 속성의 값이 ABC인 요소 선택.

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li id="orange" class="orange">오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span class="orange">오렌지</span>
</div>
```

선택

```
#orange {
  color: red;
}
```

실습, 기본 선택자 활용

```
<h1>KDT 선택자 예제</h1>
<p>선택자 사용 연습을 해봅시다</p>
<span>기본 선택자를 위한 예제입니다</span>
<ul>
  선택자로 선택하기
  <li id="first">1</li>
  <li class="second">2</li>
  <li>3</li>
  <li>4</li>
</ul>
```

- 전체 태그에 대한 글자색 → 하얀색
- 태그 배경색 → 빨간색
- id 가 first 인 태그 배경색 → 파란색
- class 가 second 인 태그 배경색 → 초록색

CSS 선택자

기본 | 복합 | 가상 클래스 | 가상 요소 | 속성

복합 선택자!

- 특수한 요소를 호출하고 싶을 때, 기본 선택자만으로는 선택이 불가능한 경우에 사용
- 종류
  - 일치 선택자
  - 자식 선택자
  - 후손 선택자
  - 인접 형제 선택자
  - 일반 형제 선택자

ABCXYZ

복합 | 일치 선택자 (Basic Combinator)

선택자 ABC와 XYZ를 **동시에 만족**하는 요소 선택.

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li class="orange">오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span class="orange">오렌지</span>
</div>
```

선택

```
span.orange {
  color: red;
}
```

ABC > XYZ

복합 | 자식 선택자 (Child Combinator)

선택자 ABC의 자식 요소 XYZ 선택.

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li class="orange">오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span class="orange">오렌지</span>
</div>
```

선택

```
ul > .orange {
  color: red;
}
```

ABC XYZ

복합 **하위(후손) 선택자 (Descendant Combinator)**

선택자 ABC의 **하위** 요소 XYZ 선택.
'띄어쓰기'가 선택자의 기호!

```
<div>
  <ul>
    <li>사과</li>
    <li>딸기</li>
    <li class="orange">오렌지</li>
  </ul>
  <div>당근</div>
  <p>토마토</p>
  <span class="orange">오렌지</span>
</div>
<span class="orange">오렌지</span>
```

선택

```
div .orange {
  color: red;
}
```

ABC + XYZ

복합 | 인접 형제 선택자 (Adjacent Sibling Combinator)

선택자 ABC의 다음 형제 요소 XYZ 하나를 선택.

```
.orange + li {
  color: red;
}
```

선택

```
<ul>
  <li>딸기</li>
  <li>수박</li>
  <li class="orange">오렌지</li>
  <li>망고</li>
  <li>사과</li>
</ul>
```

ABC ~ XYZ

복합 | 일반 형제 선택자 (General Sibling Combinator)

선택자 ABC의 다음 형제 요소 XYZ 모두를 선택.

```css
.orange ~ li {
  color: red;
}
```

선택

```html
<ul>
  <li>딸기</li>
  <li>수박</li>
  <li class="orange">오렌지</li>
  <li>망고</li>
  <li>사과</li>
</ul>
```

실습, 일치 선택자 활용

```
<h1>복합 선택자 사용 연습</h1>
<p>동물원에 왔어요</p>
<div class="zoo">
  <ul>
    여긴 사파리!
    <li>곰</li>
    <li id="tiger">호랑이</li>
    <li>팬더</li>
    <li class="lion">사자</li>
    <li>사육사</li>
    <li>사육사2</li>
  </ul>
  <span class="lion">사파리 밖의 사자</span>
</div>
<p class="lion">동물원 밖의 사자</span>
```

- 동물원 밖의 사자만 선택, 배경을 빨간색
- 사파리 안의 사자만 선택, 배경을 초록색
- 호랑이만 선택, 배경을 노란색
- 사자를 전부 선택, 글자를 하얀색
- 동물원 안의 사자만 선택, 폰트를 두껍게
- 호랑이 뒤에 있는 사육사를 선택, 배경을 하늘색
- 사파리 사자 뒤에 있는 사육사를 모두 선택, 배경을 오렌지색

CSS 선택자

기본 복합 가상 클래스 가상 요소 속성

가상 클래스 선택자!

- 사용자의 행동에 따라 변화하는 가상 상황에 따라서 요소 선택 시
- 각 요소의 상황에 따라 사용자가 원하는 요소를 선택 할 때 사용
- 특정 요소를 부정 할 때 사용
- 종류
  - 가상 클래스 선택자
  - 사용자의 행동에 따라 변화 : Hover, Active, Focus
  - 요소의 상황 : first-child, last-child, nth-child
  - 부정 선택 : not

ABC:hover

가상 클래스 선택자 (Pseudo-Classes) HOVER

선택자 ABC 요소에 마우스 커서가 올라가 있는 동안 선택.

화면에 출력!

NAVER

NAVER

선택

```
a:hover {
    color:  red;
}
```

```
<a href="https://www.naver.com">NAVER</a>
```

ABC:active

가상 클래스 선택자 (Pseudo-Classes) ACTIVE

선택자 ABC 요소에 마우스를 클릭하고 있는 동안 선택.

화면에 출력!

NAVER

NAVER

선택

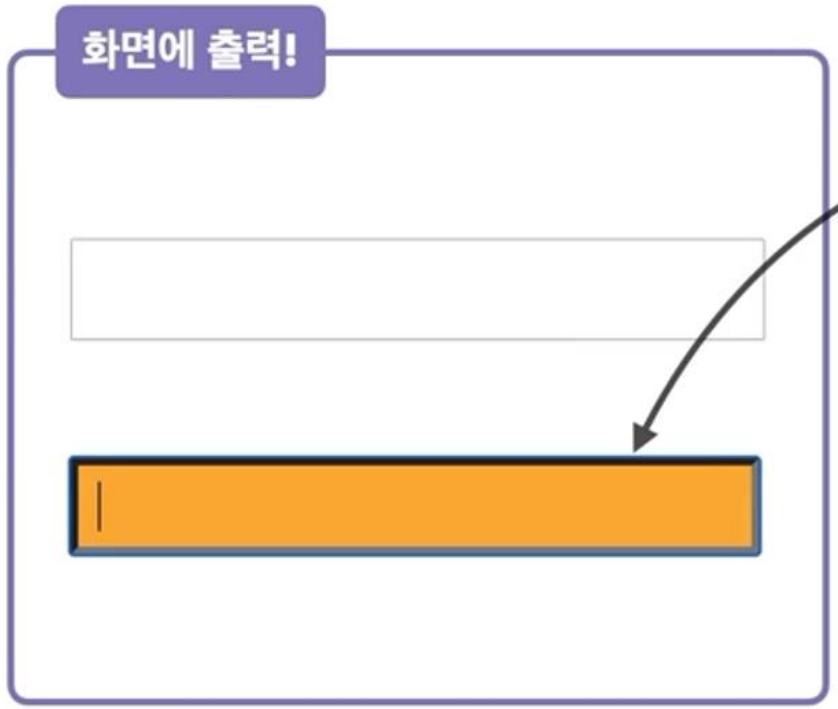
```
a:active {
    color:  red;
}
```

```
<a href="https://www.naver.com">NAVER</a>
```

ABC:focus

가상 클래스 선택자 (Pseudo-Classes) FOCUS

선택자 ABC 요소가 포커스되면 선택.




```css
input:focus {
  background-color: orange;
}
```

```html
<input type="text" />
```

Focus가 안되는 요소에 강제로 추가하기!

- 속성으로 tabindex="-1" 값을 주어서 포커스 효과를 줄 수 있음
- 속성 이름을 보고 유추 할 수 있듯, Tab 키를 눌러서 포커스를 하는 순서를 지정하는 속성
- "-1" 이외의 값을 넣을 경우 기존 페이지의 포커스 흐름이 깨지기 때문에 비추천

실습, 가상 클래스 선택자 1

- <h1>, <a> 은 1개, <input type="text"> 태그는 2개 만들기
- <h1> 태그에 마우스가 올라가면 글자가 빨간색
- <a> 태그를 클릭 하고 있으면 배경이 파란색
- <input> 태그 중 하나의 <input> 태그에 포커스가 가면 입력창이 오렌지색으로 변경

ABC:first-child

가상 클래스 선택자 (Pseudo-Classes) | FIRST CHILD

선택자 ABC가 형제 요소 중 첫째라면 선택.

```
.fruits span:first-child {
  color: red;
}
```

선택

```
<div class="fruits">
  <span>딸기</span>
  <span>수박</span>
  <div>오렌지</div>
  <p>망고</p>
  <h3>사과</h3>
</div>
```

?

```
.fruits div:first-child {
  color: red;
}
```

ABC:last-child

가상 클래스 선택자 (Pseudo-Classes) | LAST CHILD

선택자 ABC가 형제 요소 중 막내라면 선택.

```
.fruits h3:last-child {
  color: red;
}
```

선택

```
<div class="fruits">
  <span>딸기</span>
  <span>수박</span>
  <div>오렌지</div>
  <p>망고</p>
  <h3>사과</h3>
</div>
```

ABC:nth-child(n)

가상 클래스 선택자 (Pseudo-Classes) NTH CHILD

선택자 ABC가 형제 요소 중 (n)째라면 선택.

```
.fruits *:nth-child(2) {
  color: red;
}
```

선택

```
<div class="fruits">
  <span>딸기</span>
  <span>수박</span>
  <div>오렌지</div>
  <p>망고</p>
  <h3>사과</h3>
</div>
```

ABC:nth-child(n)

가상 클래스 선택자 (Pseudo-Classes) | NTH CHILD

선택자 ABC가 형제 요소 중 (n)째라면 선택.

```
<div class="fruits">
  <span>딸기</span>
  <span>수박</span>
  <div>오렌지</div>
  <p>망고</p>
  <h3>사과</h3>
</div>
```

선택

```
.fruits *:nth-child(2n) {
  color: red;
}
```

n은 0부터 시작!
(Zero-Based Numbering)

```
.fruits *:nth-child(n+2) {
  color: red;
}
```

n은 0부터 시작!
(Zero-Based Numbering)

ABC:not(XYZ)

부정 선택자 (Negation) NOT

선택자 XYZ가 아닌 ABC 요소 선택.

```
.fruits *:not(span) {
  color: red;
}
```

선택

```
<div class="fruits">
  <span>딸기</span>
  <span>수박</span>
  <div>오렌지</div>
  <p>망고</p>
  <h3>사과</h3>
</div>
```

실습, 가상 클래스 선택자 2

```
<body>
  <div class="zoo">
    <p class="predator">곰</p>
    <p class="predator">사자</p>
    <p class="predator">호랑이</p>
    <p class="predator">재규어</p>
    <div>기린</div>
    <p class="predator">치타</p>
    <h3>얼룩말</h3>
    <span class="predator">하이에나</span>
  </div>
</body>
```

- 사자, 재규어, 치타, 하이에나 선택 → 배경 오렌지
- 곰만 선택 → 배경 보라색
- 초식 동물만 선택 → 배경 초록색
- 하이에나만 선택 → 배경 빨간색 / 마우스가 올라가면 배경
- 호랑이만 선택 → 배경 파란색

CSS 선택자

기본 복합 가상 클래스 가상 요소 속성

가상 요소 선택자!

- 선택 된 요소의 앞, 뒤에 별도의 Content 를 삽입하는 선택자
- 반드시 content 라는 속성을 사용
- 빈 값(“”) 이라도 넣어 주어야 적용이 됨
- 종류
  - After : 요소의 뒤에 내용 삽입
  - Before : 요소의 앞에 내용 삽입

ABC::before

인라인(글자) 요소

가상 요소 선택자 (Pseudo-Elements) BEFORE

선택자 ABC 요소의 내부 앞에 내용(Content)을 삽입.

```
.box::before {
  content: "앞!";
}
```

화면에 출력!

앞! Content!

::before

```
<div class="box">
  
  Content!

</div>
```

ABC::after

가상 요소 선택자 (Pseudo-Elements) AFTER

선택자 ABC 요소의 내부 뒤에 내용(Content)을 삽입.

인라인(글자) 요소

```
.box::after {
  content: "뒤!";
}
```

화면에 출력!

Content! 뒤!

::after

```
<div class="box">

  Content!

</div>
```

실습, 가상 요소 선택자

- 여기요! 만들기
- 여기요 앞에, "택시" 넣기
- 겨이요 뒤에, "빨리" 넣기

가상 요소 선택자!

- 실제로 의미 없는 HTML 태그를 만들지 않고 요소 삽입이 가능하여 매우 자주 사용!
- 예를 들어 쇼핑몰 페이지에 메뉴에 Hot, 추천 등을 넣기 위해 별도의 태그를 삽입 하는 것이 아니라 가상 요소 선택자를 활용하여 처리하면 편리함!

CSS 선택자

기본 복합 가상 클래스 가상 요소 속성

속성 선택자!

- 지정한 특정 속성을 가지고 있는 태그를 선택하는 선택자
- 종류
  - 특정 속성만 지정
  - 속성과 속성의 값을 지정

[ABC]

속성 선택자 (Attribute) ATTR

속성 ABC을 포함한 요소 선택

```
[disabled] {
  color: red;
}
```

선택

```
<input type="text" value="HEROPY">
<input type="password" value="1234">
<input type="text" value="ABCD" disabled>
```

[ABC="XYZ"]

속성 선택자 (Attribute) ATTR=VALUE

속성 ABC을 포함하고 값이 XYZ인 요소 선택.

```
[type="password"] {
  color: red;
}
```

선택

```
<input type="text" value="HEROPY">
<input type="password" value="1234">
<input type="text" value="ABCD" disabled>
```

실습, 가상 요소 선택자

```
<input type="text" placeholder="이름" />
<input type="password" value="pw" />
<input type="text" value="000-0000-0000" />
<input type="text" placeholder="핸드폰" />
<input type="text" placeholder="주민번호" disabled />
```

- Disabled 속성을 가진 태그 선택 → 배경 빨간색
- placeholder 속성이 “이름” 값을 가지는 태그 선택 → 배경 오렌지
- placeholder 속성을 가지지 않는 태그 선택 → 배경 파란색

스타일

상속!

```css
.animal {
  color: red;
}
```

선택

```html
<div class="ecosystem">생태계
  <div class="animal">동물
    <div class="tiger">호랑이</div>
    <div class="lion">사자</div>
    <div class="elephant">코끼리</div>
  </div>
  <div class="plant">식물</div>
</div>
```

?

스타일 상속!

- 자식 요소가 별도의 Style 속성이 없을 경우, 부모의 속성 값을 그대로 상속하는 CSS 특성!
- 단, 상속이 되는 속성과 상속이 안되는 속성으로 구분

상속되는 CSS 속성들..

모두 글자/문자 관련 속성들!

(모든 글자/문자 속성은 아님 주의!)

font-style : 글자 기울기
font-weight : 글자 두께
font-size : 글자 크기
line-height : 줄 높이
font-family : 폰트(서체)
color : 글자 색상
text-align : 정렬
...

강제 상속!

- 자동 상속 되지 않는 속성의 값을 “inherit”을 주어 부모의 CSS 속성 값을 그대로 상속하게 하는 방법

```css
.parent {
  width: 400px;
  height: 200px;
  background-color: red;
  border: 1px solid #000;
}
.child {
  width: 100px;
  height: inherit;
  top: 100px;
  right: 100px;
  position: fixed;
  border: 1px solid #000;
}
```

선택자 우선 순위!

- CSS 참조 방식 우선 순위와는 다른 개념입니다!
- 같은 HTML 요소가 여러 선택자에 의해서 선택 되었을 경우 어떤 선택자의 CSS 속성을 우선 적용할지 결정하는 방법
- 1 순위 : 점수가 높은 선택자가 우선!
- 2 순위 : 점수가 같으면, 가장 마지막에 읽힌 선택자가 우선!

```css
div              { color: red !important; }
#color_yellow    { color: yellow; }
.color_green     { color: green; }
div              { color: blue; }
*                { color: darkblue; }
body             { color: violet; }
```

선택

```html
<div
  id="color_yellow"
  class="color_green"
  style="color: orange;">
  Hello world!
</div>
```

과연 글자색은??

!important - 9999999999점

ID 선택자 - 100점

Class 선택자 - 10점

태그 선택자 - 1점

전체 선택자 - 0점

상속 - X

```
div                { color: red !important; }
#color_yellow      { color: yellow; }
.color_green       { color: green; }
div                { color: blue; }
*                  { color: darkblue; }
body               { color: violet; }
```

인라인 선언 - 1000점

```
<div
  id="color_yellow"
  class="color_green"
  style="color: orange;">
  Hello world!
</div>
```

선택자, 우선 순위 점수

- !important : 9999999999 점
- 인라인 방식 : 1000 점
- #ID 선택자 : 100 점
- .Class 선택자 : 10 점
- 태그 선택자 : 1 점
- 전체 선택자(*) : 0 점

21점 `.list li.item { color: red; }`

21점 `.list li:hover { color: red; }`

11점 `.box::before { content: "Good "; color: red; }`

101점 `#submit span { color: red; }`

22점 `header .menu li:nth-child(2) { color: red; }`

1점 `h1 { color: red; }`

10점 `:not(.box) { color: red; }`


수고하셨습니다!